Design and Implementation of a 3D Laser Scanner: Using Object Rotation and Plane of Light Scanning

Diploma Thesis

Bernhard Mörtl

Supervisor

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Paul O'Leary

University of Leoben

Institute for Automation

December 2010

Abstract

The aim of this work is to investigate the performance of 3D-scanning using a laser-plane of light sensor combined with a turntable. To this end a complete measurement system was designed, implemented and tested. A new calibration concept was conceived and the required calibration objects and procedures were implemented. There are two main steps involved: calibration of the plane of light measurement unit; and registration of the axis of rotation relative to the laserplane of light.
The complete system, optical measurement, rotation control and surface reconstruction has been implemented and successfully tested on a series of engineering objects. The acquired data can also be exported to a CAD an STL data file.

Zusammenfassung

Das Ziel dieser Arbeit ist es die Effizienz eines Lichtschnittsensors in Verbindung mit einem Drehtisch zum Zwecke der 3D Rekonstruktion von technischen Bauteilen zu untersuchen. Hierzu wurde ein komplettes Messsystem entworfen, implementiert und getestet. Es wurde ein neues Konzept zur Kalibration erdacht, die dazu notwendigen Kalibrationsobjekte gefertigt und die Kalibrationsmethode implementiert. Dazu sind 2 Schritte erforderlich: die Kalibration des Lichtschnittensors; und die Registration der Drehachse relativ zur Laserebene.

Das gesamte System, die optische Messeinrichtung, die Steuerung des Drehtisches und die 3D Oberflächenrekonstruktion wurden implementiert und wurden an einer Reihe von technischen Bauteilen getestet. Die gewonnenen Messdaten können auch in Form eines STL Datenformats für ein CAD Programm nutzbar gemacht werden.

Contents

Declaration Of Authorship

I certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other University.

Bernhard Mörtl
Leoben, December 2010

Acknowledgements

I would like to thank my supervisor o.Univ.-Prof.Dipl-.Ing.Dr.techn Paul O'Leary for his support and guidance during this work and also during a long part of my study here in Leoben. He always had an open ear for my problems and gave me lots of good ideas and technical advice.

I also want to thank Ing. Gerold Probst who was a great help in installing all the hardware components for my experimental setup.

Furthermore I want thank the whole staff at the Institute of Automation, where I spend some great years. Especially I am grateful to Doris Widek for her support by countless small things.

A special gratitude to my colleagues Gappi, Lampi, Mösi, Reini and Steff for the great time we had and all the foolery we did together. I will look back with pleasure at our time in Leoben.

Thanks to my family for enabling my education and for always giving me a big backing.

Last but not least I want to thank Verena for giving me the essential motivation and mental support during the last month.

Chapter 1

Introduction

Nowadays 3-dimensional high precision measurement for applications such as reverse engineering are in great demand. There are two main applications for reverse engineering: In quality control where the geometry of a manufactured object is compared with its CAD[1] drawing to determine deviations; and to obtain geometric data for objects where no CAD data is available. There are several approaches how reverse engineering can be done, depending on the kind of object which should be measured or the demanded precision, a suitable method has to be chosen. In principle it can be distinguished between contact or non contact methods and between active and passive ones. The book of Vinesh Raja [3] offers a good insight into the diversity of methods for reverse engineering and the complete theoretical procedure from scanning an object to getting a 3D computer model. Contact scanners offer very high accuracy in measurement, but are very slow for complex geometries. Also soft and deformable materials cannot be measured with such devices. Furthermore contact scanning devices are very expensive because of their strict manufacturing tolerances. Non-contact methods are providing a much higher scanning rate and they do not get in contact with the measurement object. The problem of state of the art non-contact scanners is grounded in the missing precision for real engineering applications. Because of increasing cost pressure and the universal applicability its getting more and more interesting for engineering to use non contact scanners, so it is necessary to improve their accuracy.

[1]Computer aided design

1.1 Previous Work on Non-contact Methods

Hartley,Zisserman [4], Fitzgibbon et al.[5] have done considerable investigation on 3D model reconstruction using passive methods like shape from silhouette [6] and shape from shading [7] combined with rotational motion on a turntable. These methods are only suitable for convex objects i.e there are no concave areas in the surface. After taking a closer look at the mathematics of how the axis of rotation is determined it turned out that this method seemed to be intransparent. Methods such as voxel-carving, also called space-carving [8], are able to reconstruct concave objects, but are only applicable to objects which are clearly textured. These mentioned methods are suitable for creating models for virtual reality, for scanning archaeological findings or works of art. If a precision, better than 0.2 mm, is required non-contact methods are presently not reliable.

1.2 Plane of Light Method

The plane of light method, which is an active scanning method, is one of the best approaches to realize a non-contact scanner for these kind of problems, it is well known, countless of papers have been published on this topic. O'Leary et al. [9] have published numerous papers dealing with investigation on the field of light-sectioning and surface inspection. With the use of a camera, a laser and the principle of triangulation, it is possible determine the depth information on a test specimen, from the lateral displacement of the laserline. It has proofed its reliability in many industrial environments where continuous quality control of contours or surfaces within production lines is required.
The reason why laserscanning is so multi-purpose is, that it offers the following advantages:

1. it is a contact free non destructive measurement method;
2. detailed scans can be realized;
3. its possible to scan arbitrary objects;
4. it requires a relatively simple measurement setup; and
5. it is a fast scanning method which can be integrated into a production line.

Some difficulties and limitations of light - sectioning are:

1. highly reflective object surfaces can disperse the laser in an inappropriate manner;
2. the laserline could be hidden due to occlusion;
3. for high quality measurements an extensive calibration process is necessary; and
4. the accuracy of state of the art laserscanners are limited to about 0.1 mm.

Adding linear motion either to the object or the measurement unit it is possible to capture and reconstruct a surface or one view of an object. But for reverse engineering applications, it is necessary to capture the whole geometry of a specimen so it would be meaningful for objects with a longitudinal axis to use rotational motion instead of linear motion. Laserscanning combined with a turntable is an approach on which very few papers has been published, so it offers a very interesting field of investigation.

1.3 Outline

In this work an experimental setup of a 3 - dimensional laserscanner is presented. It uses a PLC [1] controlled turntable to add rotational motion to the measurement target. Also the problem of an insufficient calibration process, where the laserline and the rotation axis of the turntable are misaligned, is addressed. The mathematical background, some concrete test measurements and a possible way of converting the scanned data into a CAD compatible file format is presented.
The last chapter summarizes the complete work and offers a perspective to further fields of investigation on the topic of laserscanning.

[1]Programmable logic controller

Chapter 2

Reverse Engineering

Reverse engineering is a widely used method in diverse section of the technical world. It is the process of getting a blue-print, a 3D computer model or simply a duplicate of a structural element without having any drawings of it.

Software developers are using it to implement objects which appear in real life into video games or other kind of software. Designers are creating their concepts mostly in plaster, foam rubber or something else. To enable serial production it is essential to have a CAD model. Another field of application for reverse engineering is, if a spare part for a machine is required and its original manufacturer no longer produces this parts. Many more cases could be listed in which the application of reverse engineering can be a practical method to simplify technical problems. The theoretical background in this chapter is mainly related to [3] which offers a good and entire explication on reverse engineering.

2.1 Generic Process

Although there are several different methods of reverse engineering the general operation procedure which is shown in figure 2.1 is the same. The chosen method for scanning the target depends on:

- the application the result should be used for
- the size of the part
- the complexity

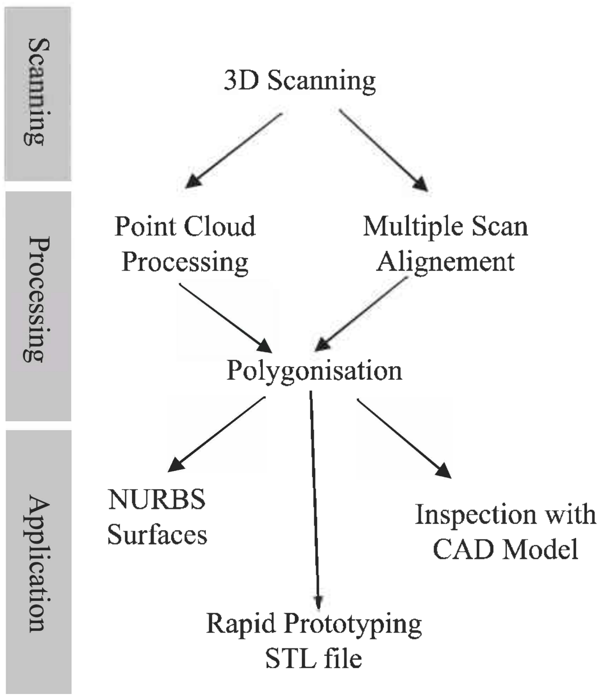


Figure 2.1: Generic process of reverse engineering. Adopted from [3]

- the material of the scanned part
- the required accuracy
- the part geometry

2.1.1 Phase 1 - Scanning

The selection of the proper scanning method is essential for a successful 3D shape reconstruction. There are

- Contact Scanners
- and Noncontact Scanners.

Contact Scanners
are working inversely to a CNC[1] mill. Where a CNC machine moves its cutting head to predefined coordinates in order to produce the part, the scanner is touching the surface of the measurement target with a calibrated sensing head, in order to gain the coordinates on the surface. In Figure 2.2 a contact scanner is shown. On the one hand these types

[1]computerized numerical control

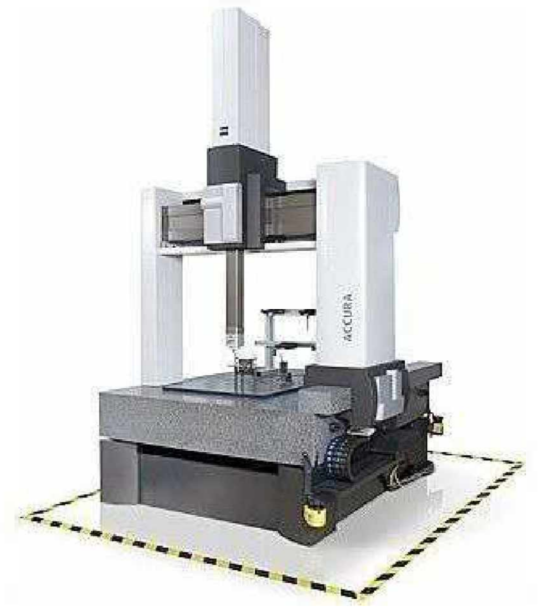


Figure 2.2: Coordinate measuring machine (CMM) from Carl Zeiss Group©.

of scanners have the advantage of high accuracy, but on the other hand they can be very slow because each point coordinate is acquired sequentially at the top of the probe. Also the material of the measurement target is limiting the application of contact scanning. Soft and easy deformable materials cannot be accurately scanned. Tolerances between 0.01 and 0.02 mm are state of the art.

Noncontact Scanners
are working without any physical contact with the measurement target. They are using lasers, structured light or CCD [2] cameras to obtain point coordinates on the object. Depending on the processing capacity of the system noncontact scanners can capture a great amount of points within a short time period. Although the accuracy of noncontact 3D scanners are constantly improving, their tolerances are higher than these of the contact scanners. Reflections on shiny, specular surfaces and deficiencies in the calibration process are the main problems in the application of noncontact scanners for reverse engineering. This type of 3D scanners can be further divided into

- active
- and passive

methods. Active methods are the light-sectioning method or the structured light projection. Passive methods are for example shape from silhouette, shape from shading or

[2]charged coupled device

shape from texture.
The end of every scanning process leads do a dataset which includes all the X,Y and Z coordinates of the scanned point cloud.

2.1.2 Phase 2 - Processing

This phase includes the import of the scanned data set into the computation environment, reducing the noise and if necessary also the number of points. For noncontact scanning methods like the light sectioning the processing phase implies the extraction of the laserline on the object. This has to be done with great care, not to loose to much information of the specimen. Sometimes it is necessary to combine multiple scans of an object to get a proper data set for reconstruction. Output of the processing phase is a merged, smoothed point data set in a format, which can be easily used for further processing.

2.1.3 Phase 3 - Application

To use the acquired data set in a meaningful way for reverse engineering it should be converted into a CAD model. In a CAD format the element can be imported into construction drawing, it can be measured or simply produced in a CNC machining center to make physical copies of it. The most popular CAD format for reverse engineered elements is the STL[3] format. It is easy to understand and to build up and the majority of all CAD software packages e.g. AutoCAD®, CATIA®, SolidWorks® etc. are able to open and import this file format.

[3]Surface tessellation language

Chapter 3

Principle of Operation

3.1 Light-sectioning Principle

The light-sectioning technique [2] is a widely used method to measure cross sectional geometries of objects. A plane of light, produced by a laser, is projected onto the surface of a specimen. The scattering of the laserlight produces a visible line; the form clearly depends on the geometry of the object. Given the position of the camera $(\varphi_1, \varphi_2, d)$ relative to the plane of light, it is possible to reconstruct the surface geometry from the camera observations. A correct measurement requires a previous calibration process (see Section 5.1) of the whole setup. In Figure 3.1 a schematic drawing of a light-sectioning setup is shown. A complete 3D reconstruction of an object requires a series of observations across the whole object. Under constant motion, the measurement data is recorded in equal time intervals corresponding to a constant spatial interval. As long as the camera and the plane of light don't change their relative position, it makes computationally no difference whether the object or the measurement equipment is moving. The reconstruction of the measured object can be performed by assembling all acquired observations. Most commonly, linear motion is used in conjunction with light-sectioning, e.g. during surface inspection in quality control. A turntable offers a simple solution to generating rotational motion, enabling the scanning of complete structural elements. Rotational motion is particularly well suited for the measurement of rotationally invariant objects (see Figure 3.3). Except the different kind of motion, the principle for a turntable setup, shown in Figure 3.2 stays the same.

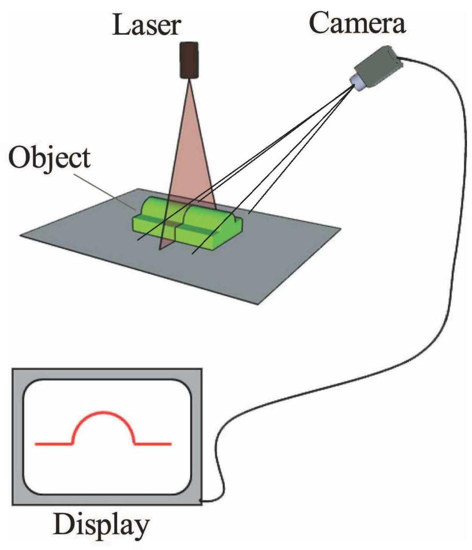


Figure 3.1: Schematic drawing of a light sectioning setup. A laser projects a plane of light onto the surface of a specimen. Observing this scene with a camera, the surface geometry can be determined from the vertical displacement of the laserline in the image.

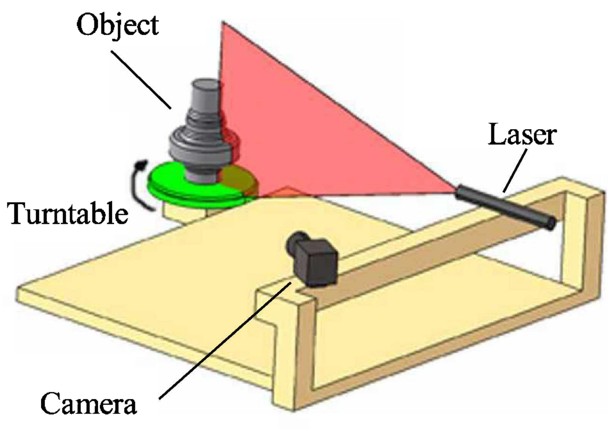


Figure 3.2: Schematic diagram of the 3D scanner consisting of a laserplane of light measurement unit combined with a turntable which produces rotational motion.

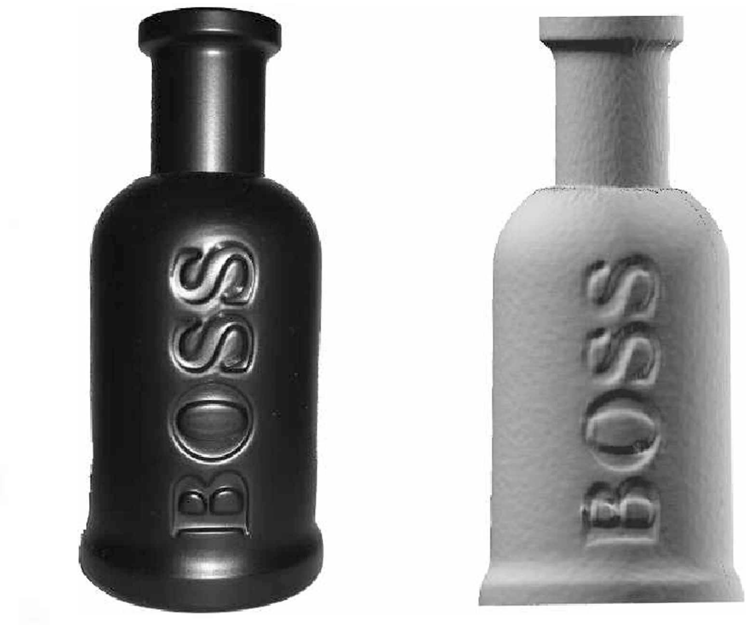


Figure 3.3: The above object has an axis of rotation around which it is predominantly invariant. Left: photograph of the object. Right: result of the scan with the proposed system.

Laser	*Lasiris Inc. red light laser 600-710 nm Power < 5 mW*
Camera	*Pulnix TM-1400 CL; resolution: 1024x1024 gray scale*
Transmission	*Wittenstein TP 004A - MF2 - 50 - 0B1*
Frame grabber	*National Instruments NI PCI-1428*
Synchronous motor	*Bernecker & Rainer 8MSA2M*
Control	*Bernecker & Rainer ACOPOS 1045*

Table 3.1: List of the used hardware for the experimental measurement setup

3.2 Experimental Setup and Data Acquisition

For this work a complete experimental measurement setup was designed and implemented in the laboratory. The aim was to implement a measurement system which can be used to scan the geometry of small objects. The system has been designed to accommodate objects with a height of $h \approx 15$ cm and a diameter of $d \approx 10$ cm. Standard industrial automation components have been used to implement the system (see Table 3.1). An optical interference filter, matched to the wavelength of the laser, can be added to the system if high intensity ambient light is present or to be expected.

A Bernecker & Rainer (B&R) three phase synchronous motor combined with transmission (50 : 1 gear reduction), actuates the turntable. This combination offers a very

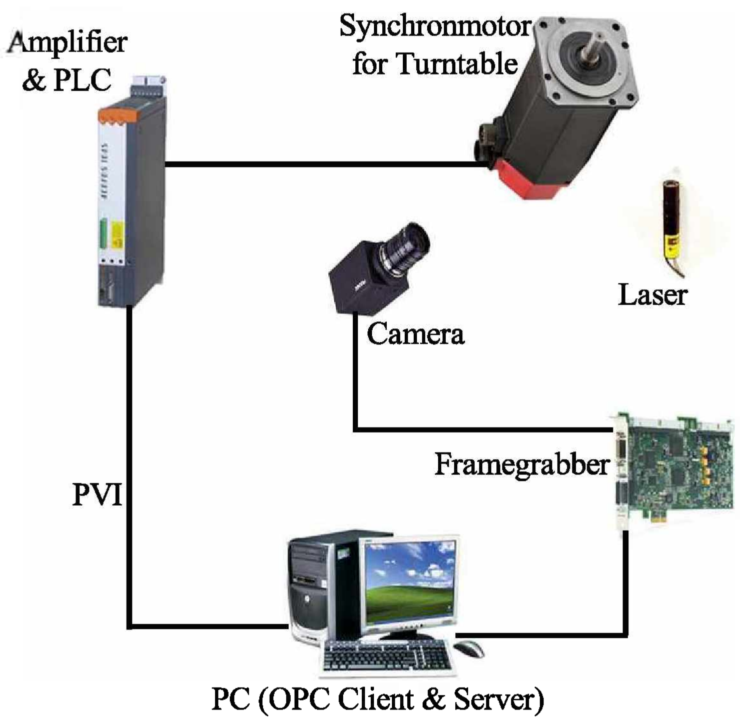


Figure 3.4: Schematic diagram for the major components forming the 3D scanner and their interconnection.

fine grading of 50000 steps for a complete rotation of 360 degrees. The motor is connected to a B&R amplifier with a built-in PLC. The PLC which is interconnected over PVI[1] to a OPC[2] server. In this special case the OPC server and OPC client are running on the same personal computer from which the whole system is controlled. Images are acquired by a Pulnix TM -1400 CL gray scale camera with a resolution of 1024x1024 pixels. A camera-link interface is used to transfer the image data to the memory of the PC via a PCI frame grabber. Matlab® is used to control the turntable and the camera within one program.
The use of an OPC server facilitates the simple installation of such measurement systems, but adds the problem, that it is not real-time capable. During the measurements, it must be ensured that the spatial interval between acquired images is uniform. It can not be assumed that the cyclic time of an OPC system is constant; the cyclic time can vary between 20 and 30 milliseconds to read one process variable. Therefore it is presently not possible to scan an object in continuous motion. This was not considered to be detrimental at the present time, since the investigation of the whole system is in the focus of this

[1]Process variable interface
[2]OLE for process control

work here. To address this problem a concrete example is given:

If a step size ω of 1 degree for the scanning process is chosen, the motor has to make 139 steps until the next image is taken. Depending on the adjusted speed, the motor makes 5 to 10 steps during the one cyclic period of the OPC server. Therefore the actual position of the turntable is not registered reliably to the system. To assure constant step size during the measurement, the turntable has to intercept his motion after every 139 steps until the OPC client has registered its actual position. This leads to a longer data acquisition time, but guarantees proper datasets for further processing.

The smaller the step size between the captured datasets the higher then the resolution of the reconstructed surface geometry. Decreasing the step size to gain further resolution goes along with a proportional increase in the amount of data, e.g. a dataset containing 500 images has a size of about 150 MB.
Choosing the step size is a compromise between the desired resolution and the available scanning- and computation time. Presently a step size of 1 degree offers a satisfying resolution for the reconstruction of structural elements and can be handled with a state of the art personal computer within an acceptable time period.
To some extend the application may determine the required resolution; for reverse engineering high resolution (> 0.2 mm) may be required, whereas when scanning objects for virtual or augmented-reality scenes there is no need for high resolution.

Chapter 4

Mathematical Background

4.1 Homogenous Coordinates

For all the following computations homogeneous coordinates are a crucial. Starting from a 3-dimensional coordinate system a point is defined as

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} . \tag{4.1}$$

If the homogeneous notation [10] is applied to $\boldsymbol{p}$ the homogeneous component w is added, which leads to

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} . \tag{4.2}$$

A special form of the homogeneous notation are the affine coordinates. In this case the homogeneous component w is set to 1 by

$$x^a = \frac{x}{w}, \ y^a = \frac{y}{w}, \ z^a = \frac{z}{w} \ \text{and} \ w^a = 1. \tag{4.3}$$

All the declarations from above are also valid for 2-dimensional coordinates.

4.1.1 Homogeneous Line Equation

The common definition of a line equation in 2-space is

$$y = ax + b. \tag{4.4}$$

written in homogeneous notation leads to

$$l_1x + l_2y + l_3w = 0. \tag{4.5}$$

which can be easily converted into matrix form as

$$\lambda \begin{bmatrix} l_1 & l_2 & l_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = 0. \tag{4.6}$$

In this equation λ is the affine coordinate.

4.2 Line Fitting

The homogeneous line equation can be formulated as follows:

$$L_1 = l_1x + l_2y + l_3 = 0 \tag{4.7}$$

It can be easily seen that there are three homogeneous components and as a consequence 2 degrees of freedom.
Given a set of n points

$$\mathsf{C} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \dots & \\ x_n & y_n & 1 \end{bmatrix} \tag{4.8}$$

and searching the best line approximation is a problem which can be divided in the following steps:

1. Determine the centroid of the C;

$$M_{\mathsf{C}} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix} \tag{4.9}$$

2. make the data set mean free

$$_m\mathrm{x}_i = x_i - \bar{x} \qquad \text{and} \qquad _m\mathrm{y}_i = y_i - \bar{y}; \tag{4.10}$$

3. setting up the design matrix

$$\mathsf{D} = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \tag{4.11}$$

and reducing it by applying the singular value decomposition

$$\mathsf{C} = \mathsf{U}, \mathsf{S}, \mathsf{V}^T \tag{4.12}$$

4. choosing the best solution

$$L = [\mathsf{V}(:, end)', 0]\,; \tag{4.13}$$

5. computing the normal distance from the origin

$$n = -LM_{\mathsf{C}}; \tag{4.14}$$

6. last step is to perform a back substitution

$$L(3) = n; \tag{4.15}$$

To illustrate the result of a least squares fit Figure 4.1 shows a noisy data set to which a line fit is applied.

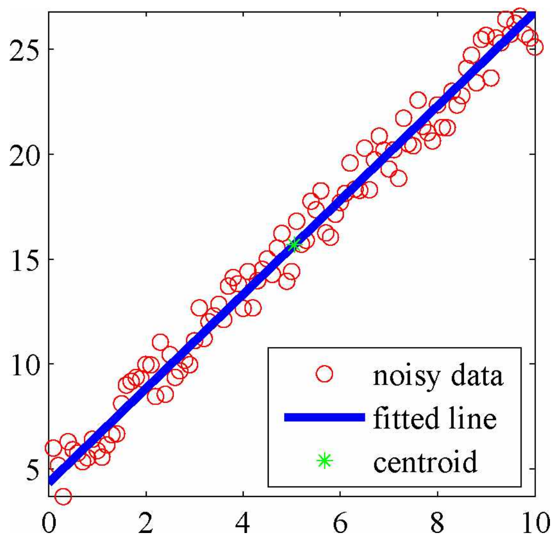


Figure 4.1: Fitted line through noisy data.

4.3 Plane Fitting

Fitting a plane to a given set of points is done in the same manner as fitting a plane. It differs in the equation to be minimized. The equation of a plane can be formulated as

$$E_1x + E_2y + E_3z + E_4 = 0, \tag{4.16}$$

where $[E_1, E_2, E_3]^T$ is the normal vector of the plane.
For a set of noisy data this equation could be written in matrix formulation as

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ \vdots & \dots & & \\ x_n & y_n & z_n & 1 \end{bmatrix} \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} \tag{4.17}$$

Finding a non-trivial solution for minimizing the residual $[e_1, e_2, \dots, e_n]^T$ leads also to the singular value decomposition.

4.4 Homography

A homography is a homogeneous projection of a point $\boldsymbol{p}$ on a plane to a Point $\boldsymbol{p'}$ on another plane. The homogeneous definition of a point is

$$\boldsymbol{p} = \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{4.18}$$

Projecting $\boldsymbol{p}$ to $\boldsymbol{p'}$ is written as

$$\boldsymbol{p} = \mathsf{H}\boldsymbol{p}, \tag{4.19}$$

where H is a 3x3 projection matrix which has the form

$$\mathsf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} . \tag{4.20}$$

The projected point $\boldsymbol{p'}$ is written as

$$\boldsymbol{p'} = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} . \tag{4.21}$$

Because one of the nine entries of H is a scaling factor, the remaining eight degrees of freedom can be determined with at least four corresponding points in each of the two planes.

The affine coordinates of the projected point $\boldsymbol{p'}$ which are x'^a and y'^a, are computed by dividing through the homogeneous coordinate w'

$$\frac{x'^a}{w'} = \frac{h_{11}x_0 + h_{12}y_0 + h_{13}}{h_{31}x_0 + h_{32}y_0 + h_{33}} \tag{4.22}$$

$$\frac{y'^a}{w'} = \frac{h_{21}x_0 + h_{22}y_0 + h_{23}}{h_{31}x_0 + h_{32}y_0 + h_{33}} \tag{4.23}$$

Rewriting the two equations form above leads to:

$$-h_{11}x_0 - h_{12}y_0 - h_{13} + h_{21}0 + h_{22}0 + h_{23}0 + x^r_a(h_{31}x_0 + h_{32}y_0 + h_{33}) = 0 \tag{4.24}$$

$$h_{11}0 + h_{12}0 + h_{13}0 - h_{21}x_0 - h_{22}y_0 + h_{23} + y_a^r(h_{31}x_0 + h_{32}y_0 + h_{33}) = 0 \tag{4.25}$$

In matrix form for n corresponding points, where $n \geq 4$, it can be written as:

$$G\boldsymbol{h} = 0. \tag{4.26}$$

4.5 Coordinate Transformation [1]

Given a fixed coordinate system a transformation of a point $\boldsymbol{p}$ to a point $\boldsymbol{p'}$ coordinate system can be done if its location and direction relative to the fixed one is known. The transformation is written as

$$\boldsymbol{p'} = \mathsf{T}\boldsymbol{p}, \tag{4.27}$$

where

$$\mathsf{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \Delta x \\ r_{21} & r_{22} & r_{23} & \Delta y \\ r_{31} & r_{32} & r_{33} & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.28}$$

which can be divided into the rotational and a translational part; every arbitrary euclidean coordinate transformation can be done with a translation and a rotation.

4.5.1 Translational Part

The translational part of T is

$$\mathsf{T}_{tr} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.29}$$

which defines the location of the origin of the local coordinate system.
Given the point $\boldsymbol{p}$ in the fixed coordinate system Ω with its origin $\begin{bmatrix} 0, & 0, & 0 \end{bmatrix}^T$

$$\boldsymbol{p} = \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} \tag{4.30}$$

and the position of the origin of the local coordinate system Ω' relative to Ω

$$\boldsymbol{\Omega}' = \begin{bmatrix} x_{\Omega'} \\ y_{\Omega'} \\ 1 \end{bmatrix}. \tag{4.31}$$

The translational part can be determined by computing the difference between two origins as

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} x_{\Omega'} \\ y_{\Omega'} \\ 1 \end{bmatrix} \tag{4.32}$$

4.5.2 Rotational Part

The Rotational part of the transformation matrix are again consisting of three matrices; they are determining the orientation of the local coordinate system. Each of these matrices describes the rotations (ξ, δ, ϵ) around one coordinate axis (X, Y, Z). The three matrices have the form

$$\mathsf{R}_\xi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\xi) & -\sin(\xi) & 0 \\ 0 & \sin(\xi) & \cos(\xi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.33}$$

$$\mathsf{R}_\delta = \begin{bmatrix} \cos(\delta) & 0 & \sin(\delta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\delta) & 0 & \cos(\delta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.34}$$

$$\text{and} \quad \mathsf{R}_\epsilon = \begin{bmatrix} \cos(\epsilon) & -\sin(\epsilon) & 0 & 0 \\ \sin(\epsilon) & \cos(\epsilon) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.35}$$

Concatenating the 3 rotation matrices and the translation part as

$$\mathsf{T} = \mathsf{R}_\xi \mathsf{R}_\delta \mathsf{R}_\epsilon \mathsf{T}_{tr} \tag{4.36}$$

leads to Equation 4.29. Thus every point in Ω can be transformed into Ω'.

4.6 1. Moment of the Intensity [2]

The width of a laserline always includes more than one pixel; starting from the position with the maximum intensity (x_{max}, y_{max}) the 1. moment is computed to determine exactly the center of the laserline.
This computation is done as follows:

$$x_{int} = \frac{\sum\limits_{x=x_{max}-m}^{x=x_{max}+m} \sum\limits_{y=y_{max}-n}^{y=y_{max}+n} x I^{p(x,y)}}{\sum\limits_{x=x_{max}-m}^{x=x_{max}+m} \sum\limits_{y=y_{max}-n}^{y=y_{max}+n} I^{p(x,y)}} \tag{4.37}$$

$$y_{int} = \frac{\sum\limits_{x=x_{max}-m}^{x=x_{max}+m} \sum\limits_{y=y_{max}-n}^{y=y_{max}+n} y I^{p(x,y)}}{\sum\limits_{x=x_{max}-m}^{x=x_{max}+m} \sum\limits_{y=y_{max}-n}^{y=y_{max}+n} I^{p(x,y)}} \tag{4.38}$$

The variables in these two equations are:

x_{int}	...x-position of the first moment of the intensity
y_{int}	...y-position of the first moment of the intensity
x_{max}	...x-position of the maximum intensity value
y_{max}	...y-position of the maximum intensity value
$I(x,y)$	...Intensity on the position x, y
p	...weighting factor of the intensity
m	...half of the horizontal pixel distance which is included into computation
n	...half of the vertical pixel distance which is included into computation

The two variables m and n are defining the area in which the computation is performed; starting from the maximum intensity position the area is spanned in x-direction $\pm m$ and in y-direction $\pm n$. These variables can be chosen manually by the user; a value between 3 and 6 is reasonable for the present application.
The exponent p determines, how much the intensity is weighted; a high intensity value has a bigger influence of the position on the first moment.

This computation can be done either in two or only in one direction; due to the fact that the laserlines in this project are almost vertical a computation in y-direction was not performed.

Chapter 5

Calibration and Registration

A complete calibration is a prerequisite for performing exact measurements. There are several aspects to the calibration of the system at hand:

1. Calibration of the plane of light measurement system;
2. Registration of the axis of rotation to the plane of light.

Without a proper calibration not even the best and most expensive setup would deliver satisfying results. There are two approaches to calibrate a plane of light system; the first one is a geometrical method, called triangulation, where the exact position and orientation of the camera $(\varphi_1, \varphi_2, d)$ relative to the plane of light has to be known; the second method computes a homogeneous projection matrix H between the camera and laser coordinate system; no positioning knowledge of the camera is required.

5.1 Calibration Process

In this work the calibration process is done according to Fauster, Schalk and Tratnig [11]. Within the calibration procedure the connection between three different coordinate systems has to be established:

1. world coordinate system (3-dimensional);
2. pixel coordinate system (2-dimensional);
3. laser coordinate system (2-dimensional);

Therefor a 3-stage calibration target, which is shown in Figure 5.1, is used. This target contains 30 circular red-light LED's; their positions are known a-priori from the design drawing. Putting this target onto the turntable two different calibration images were taken. An exact alignment of the target is not necessary; the planes containing the LED's should be, as far as possible, perpendicular to the plane of light and all the 30 LED's have to be visible on the images. The turntable is not moved during the calibration. The first picture is an image of the 30 activated LED dots. The second contains only the scattering of the plane of light from the laser on the surface of the target (see Figure 5.2).
A homogeneous transformation of two planes between two coordinate systems requires four corresponding points. Based on this fact it is possible to compute the three homography matrices $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$; one for each stage of the calibration target. The homogeneous coordinates of the LED dots are exactly known. Finding their center in the image is done by means of image processing. With an adaptive thresholding method, the gray scale image from the camera is converted into a binary image. Applying some basic morphological operations only the 30 LED dots remain. Now from each dot, the first moment, also called the center of gravity (COG) in x- and y-direction is calculated by:

$$C_x = \frac{\sum_{i=1}^{n} x_i}{n} \tag{5.1}$$

and

$$C_y = \frac{\sum_{i=1}^{n} y_i}{n}. \tag{5.2}$$

Once acquired the 30 center points of the LED's (see Figure 5.3), the homogeneous transformation matrices $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$ can be calculated. It is possible to transform every point $\boldsymbol{p}_i$ in one of the three planes into the corresponding point $\boldsymbol{p}_i'$ in another coordinate system. This computation is written as

$$\boldsymbol{p}_i' = \mathsf{H}\boldsymbol{p}_i. \tag{5.3}$$

A detailed description of how to calculate homography matrices is given in Section 4.4.
Now the laserline has to be extracted from the second calibration image. To get the exact position of this laserline some more complex computations have to be applied. A rough estimation of its location is found via the adaptive thresholding method which leads to a binary image containing the laserline. Instead of calculating the COG of the laserbeam from the binary image the first moment of its intensity is estimated from the gray scale image; its location is acquired from the binary image. Computing the 1. moment of

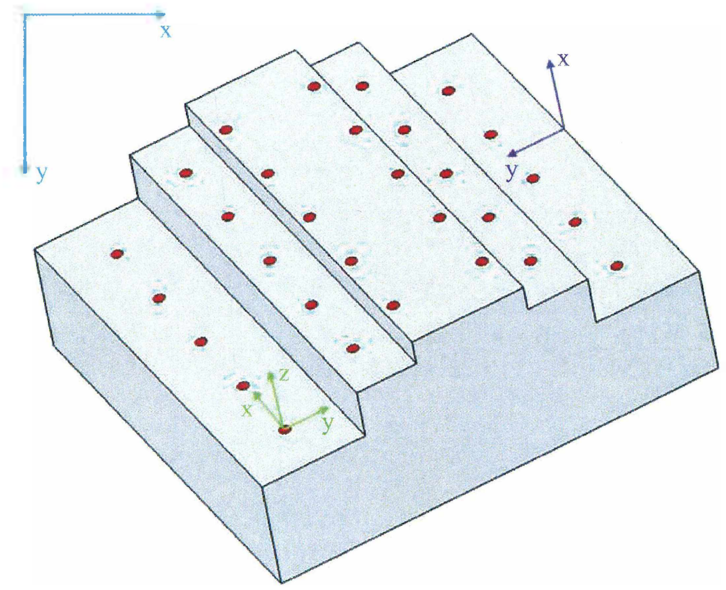
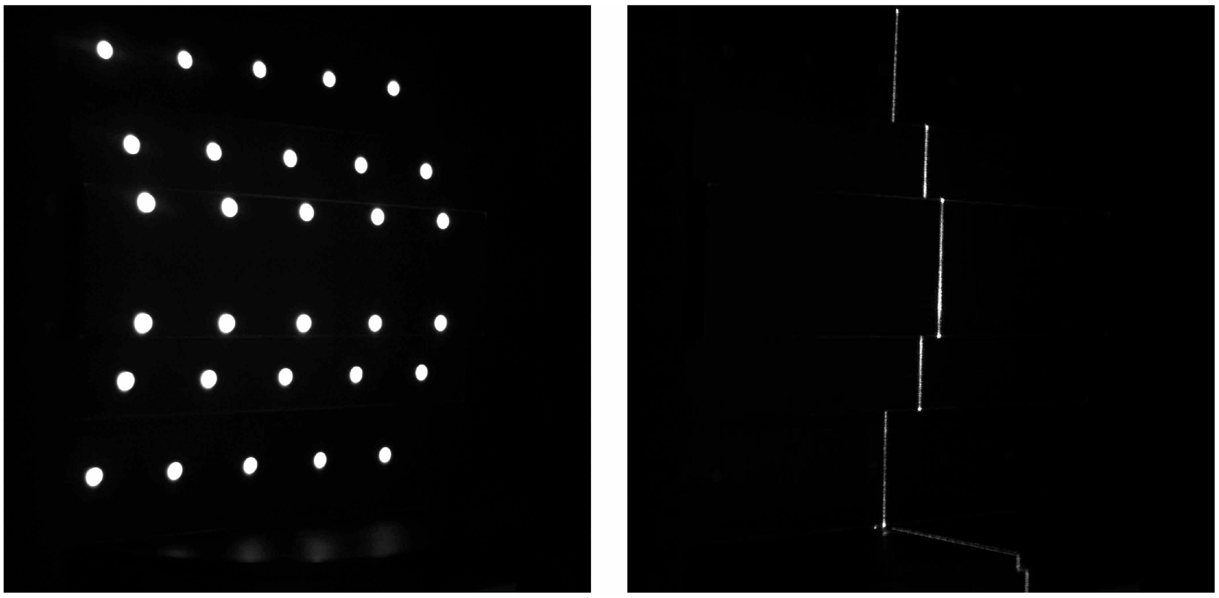


Figure 5.1: Calibration target with the 3 coordinate systems. Green: World coordinate system. Blue: Pixel coordinate system. Purple: Laser coordinate system



(a) LED's (b) Plane of light

Figure 5.2: Calibration images.

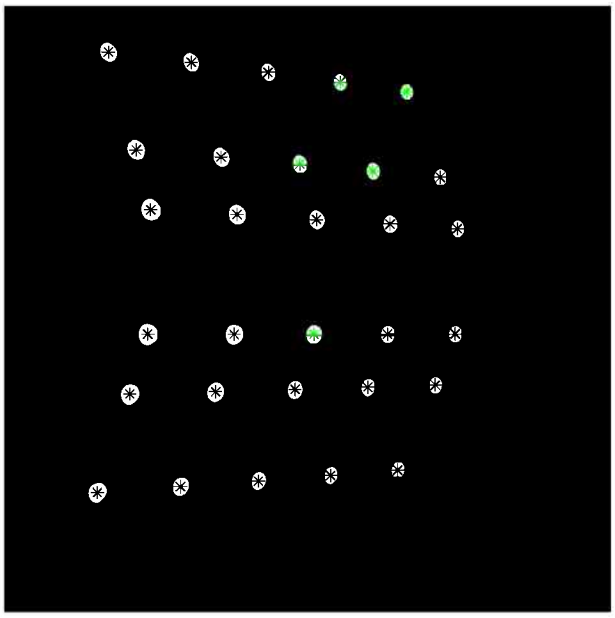


Figure 5.3: Binary calibration image with the computed COG's.

the intensity is an improvement which leads to a better and much smoother estimation. Figure 5.4 illustrates the improvement achieved using the first intensity moment. Next step is to segment the laserline on the 3 stages of the target. Performing a least-squares line fitting (see Section 4.2) to the points of the 3 line segments is a further improvement in determining the laserline; their intersection with lines fitted through the centers of gravity of the LED's brings up 6 high precision points on the laserline (see Figure 5.5).

These six points are now transformed, by using the three homography matrices, into homogeneous world coordinates. Replacing the homogeneous coordinate z of the points with the pre-known height information of the calibration target, the results are the real 3-dimensional coordinates of these points on the target; the lowermost stage is determined as $z = 0$.

Next step within the calibration process a plane through the six intersection points is fitted. To correct potential misalignments between the y-axis of the world- and the laser coordinate system a euclidean coordinate transformation is applied (see section 4.5). This operation yields to a 4x4 transformation matrix; applying this matrix to the six points of intersection they are transformed into the laser coordinate system. Now a homography between these points in the pixel and the laser coordinate system is calculated.

Storing this 3x3 transformation matrix, it can be used to measure the dimensions of every object on which the laser is projected. As previously mentioned in this section the camera and the laser must not move with respect to each other, otherwise the calibration process has to be done again.

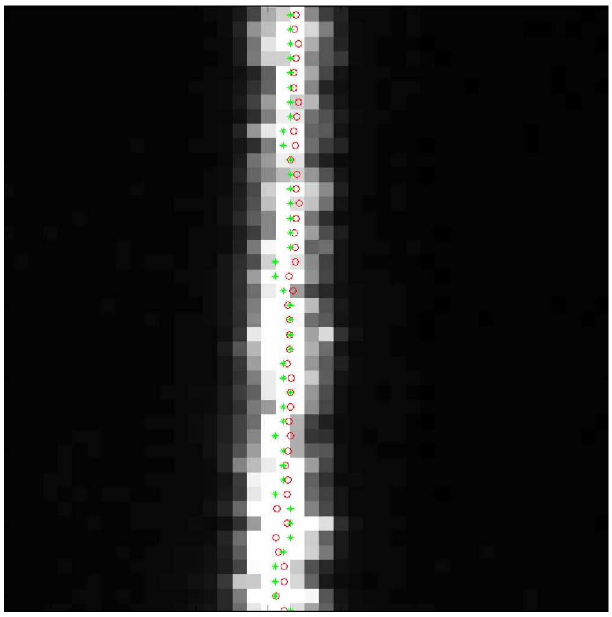


Figure 5.4: Points on the laserline computed with two different methods The green "*" are representing the points of the laserline computed with the cog method, the red "o" are calculated by the first moment of the intensity.

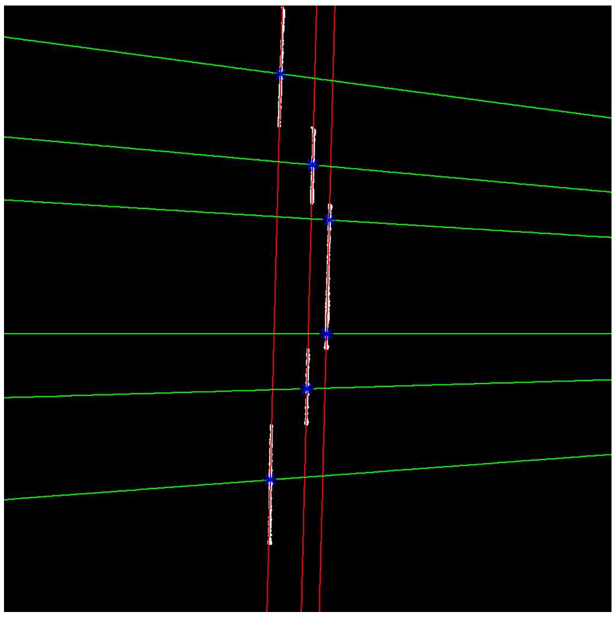


Figure 5.5: Calibration image with the fitted lines through the LED's (horizontal green lines) and the laserline (vertical red lines). The blue points are marking their points of intersection.

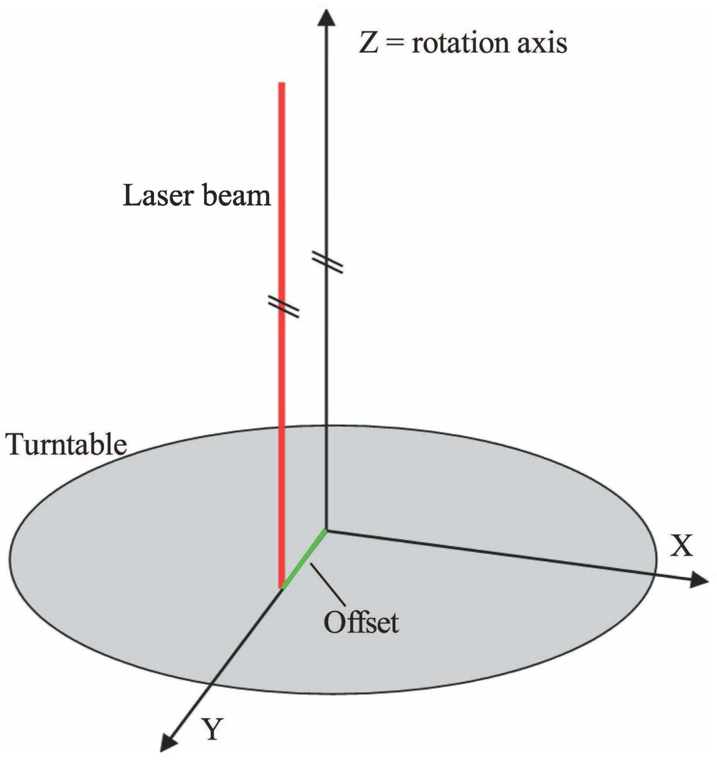


Figure 5.6: Misalignment where the laserbeam is parallel to axis of rotation, but has a lateral offset.

5.2 Offset Computation

In order to perform accurate measurements it is also necessary to register the axis of rotation with respect to the laser beam. The position of the rotation axis respectively the position of the turntable has no influence on the calibration the light sectioning head.
It can be distinguished between two types of misalignment of the laserplane of light relative to the center of the turntable:

- the plane of light is parallel to the axis of rotation but has a lateral displacement; or
- there is a tangential deviation between plane of light and axis of rotation.

Generally a combination of these two misalignments occurs. In Figure 5.6 a drawing is shown which illustrates these type of problem. Assuming that production tolerances and adjustment of the measurement equipment are not faultless the elimination of this source of error has to be considered. The issue of production tolerances, which may lead to misalignment, are not seriously addressed in the literature [12] and [13]. Actually the

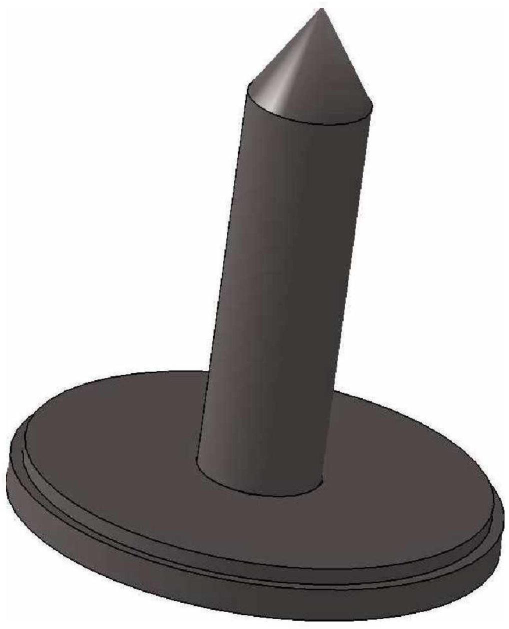


Figure 5.7: Calibration target consisting of a disk and a cylinder.

issue is sometimes ignored totally or addressed in the following manner:

"*The laser projectors are positioned so that the projected beams are as vertical to the rotational axis of the turntable as possible.*" [14]

This is inadequate for a scientific publication to address a problem which can cause considerable error in measurements. To register the turntable a calibration target was designed (see Figure 5.7); the manufacturing accuracy of the target is ± 0.01 mm. This target permits the calculation of the two different misalignments mentioned before. Each of these computations requires a careful previous calibration of the light sectioning head. (see Section 5.1). In Section 5.1 it has been mentioned that an exact alignment of the 3-stage calibration target is not strictly necessary; it is also possible to eliminate deviations between the rotation axis and the zero stage of the calibration target.

5.2.1 Parallel Displacement

Once the light sectioning system is correctly calibrated it is possible to measure distances from a pre-defined reference level to any object which is visible; in this case the reference level is the lowermost plane of the 3-stage calibration target. It can not be ensured, that the axis of rotation and the reference level have exactly the same distance from the light sectioning equipment. Figure 5.8 shows a schematic drawing of the geometry of how the two offsets h and u are computed. All the physical dimensions of the registration target are known exactly. The two necessary dimensions are the diameter r of the cylinder and the diameter R of the disk; the cylinder is exactly screwed into the center of the disk which is also mounted with self-centering counter-sunk screws on the axis of rotation.

Measuring the distances from the reference level to the edge of the disk L_1 and also to the shell of the cylinder L_2 the real length x between shell and rim can be calculated by,

$$x = L_1 - L_2. \tag{5.4}$$

Considering the trigonometric relationships shown graphically in Figure 5.8 the following relationship can be determined; substituting for L_1 and L_2

$$L_1 = (R - r) + r(1 - \cos(\varphi)) \qquad \text{and} \tag{5.5}$$

$$L_2 = R(1 - \cos(\eta)), \tag{5.6}$$

where the angles φ and η are

$$\varphi = \sin^{-1}\left(\frac{h}{r}\right), \qquad \text{and} \tag{5.7}$$

$$\eta = \sin^{-1}\left(\frac{h}{R}\right). \tag{5.8}$$

Now backsubstituting into Equation 5.4 yields to

$$x = (R - r) + r\left[1 - \cos\left(\sin^{-1}\left(\frac{h}{r}\right)\right)\right] - R\left[1 - \cos\left(\sin^{-1}\left(\frac{h}{R}\right)\right)\right]. \tag{5.9}$$

This simplifies to

$$x = -r\sqrt{\frac{h^2}{r^2} - 1} + R\sqrt{1 - \frac{h^2}{R^2}}. \tag{5.10}$$

Except for the lateral offset h, the variables in Equation 5.9 are fully determined with the measurements and the knowledge of the target dimensions. Solving for h yields

$$h = \pm\frac{1}{2}\left[\frac{\sqrt{2r^2x^2 - R^4 + 2R^2r^2 + 2R^2x^2 - r^4 - x^4}}{x}\right]. \tag{5.11}$$

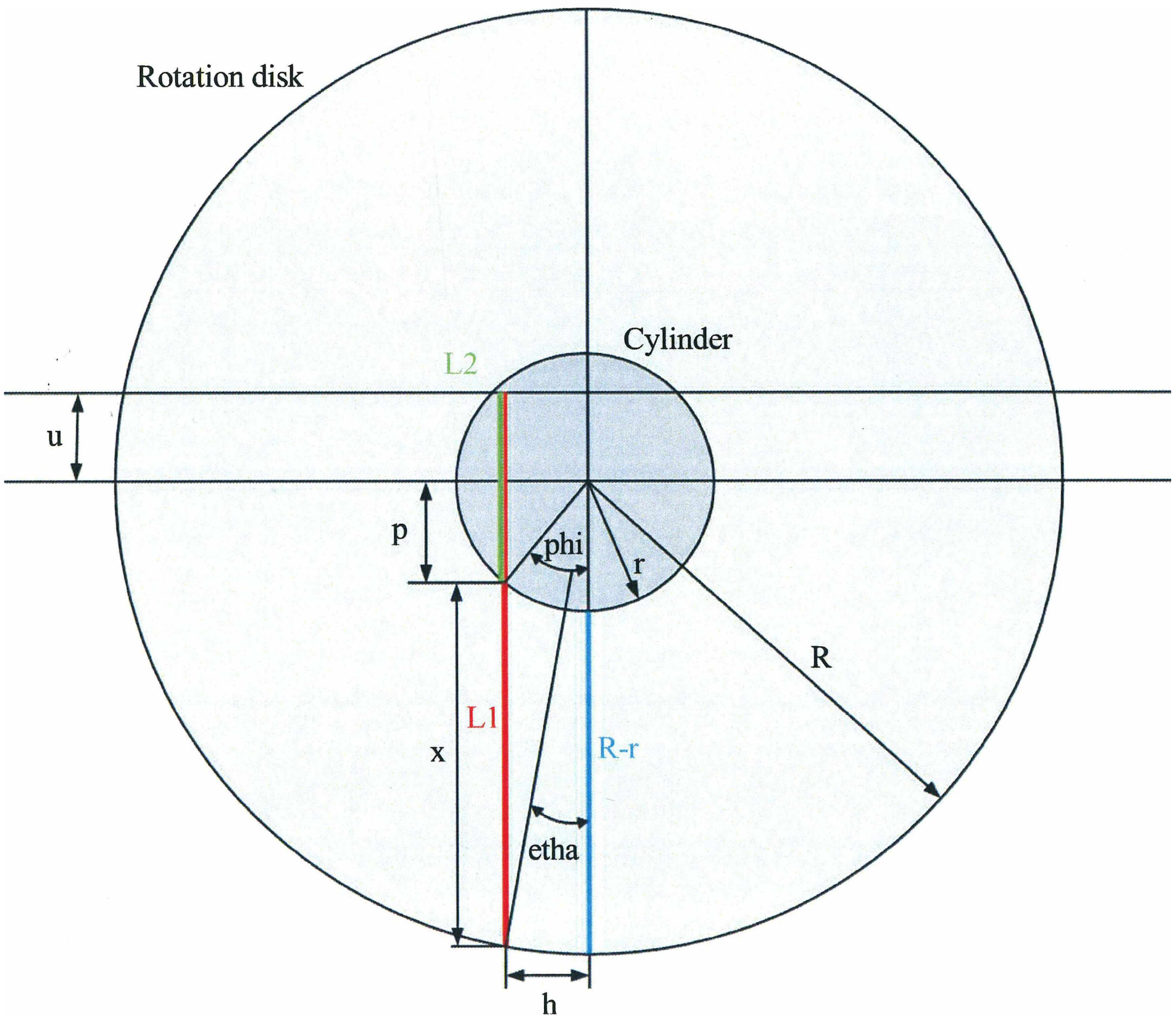


Figure 5.8: Schematic drawing of the parallel offset computation.

This can be also written as

$$h = \pm\frac{1}{2}\left[\frac{\sqrt{-(r+R+x)(r+R-x)(-r+R-x)(x+R-r)}}{x}\right]. \qquad (5.12)$$

It has to be mentioned that an offset in lateral direction is not always a fault in aligning the rotation axis. A lateral offset can be caused deliberately to avoid occlusions of the plane of light due to the object. After calculating the lateral offset the correction of the error in depth information can be performed. The depth offset u is defined as

$$u = L_1 - p, \qquad (5.13)$$

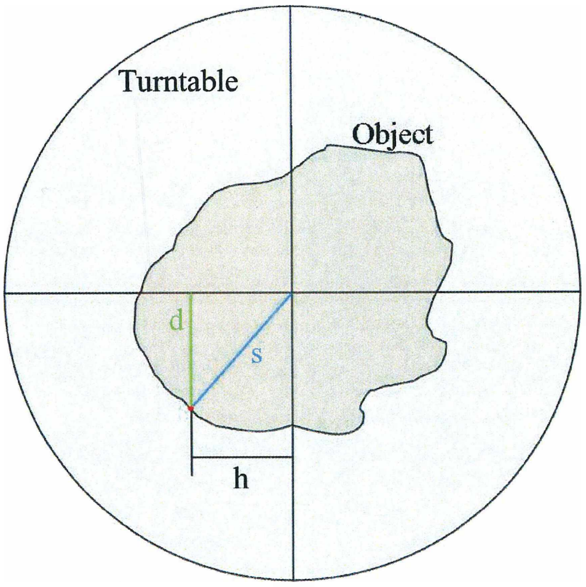


Figure 5.9: Schematic drawing of how the turntable is registered.

where p is

$$p = r\cos(\varphi). \tag{5.14}$$

So it is possible to correct any offset where the laserbeam stays parallel to the axis of rotation; a precondition is, that the lateral offset is smaller than the radius of the registration cylinder. Otherwise the laserline is not visible on the cylinder shell and a calculation is not possible.

5.3 Turntable Registration

To register the turntable in order to perform correct measurements, further computations are necessary. At first the error in setting up the calibration target has to be eliminated. Depending on the prefix of the calculated depth offset u its value has to be either added or subtracted from the x-value of all the measurement data. So the distance between every point on the object surface and the reference level, where the x-coordinate of the laser is zero, can be determined correctly.

Next step is to eliminate the lateral offset h. For a proper reconstruction the real distance between the object surface and the axis of rotation has to be determined. If there is a lateral offset between the rotation axis and the laserbeam, the measured distance d (see Figure 5.9) is not equal to the distance s which is required for reconstruction. This problem can be solved very easy, applying the pythagorean theorem, s can be calculated as

$$s = \sqrt{h^2 + d^2}. \tag{5.15}$$

This computation has to be performed for every measured point on the object surface. By this means it can be ensured, that it is possible to carry out reliable measurements.

Chapter 6

Test Measurements

A set of test measurements were performed to verify the correct functionality and accuracy of the designed system. As mentioned in Section 2 there are 3 steps, which must be performed so that a 3D model can be used for engineering:

- Image acquisition;
- Processing;
- Conversion to CAD compatible format.

In this chapter the first two steps are described in detail. The object shown in Figure 6.1 is used to illustrate the process; all images and graphics come from one and the same measurement. Following the detailed analysis of a single measurement; finally results from measuring a series of objects are presented.

6.1 Image Acquisition

The object to be measured is placed on the turntable. An exact positioning is not necessary; however, the eccentricity must be limited (in the range of 10 mm) to ensure that the laserplane impinges upon the object during the complete 360° rotation.
During these measurements, the effects of ambient light was reduced by shading the object from the light sources in the vicinity.

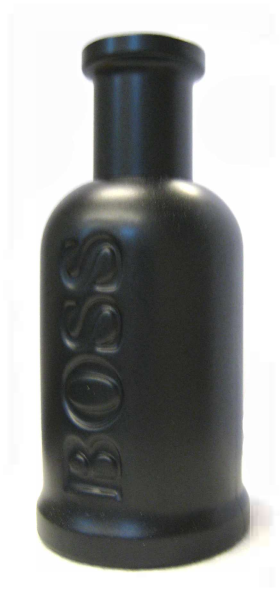


Figure 6.1: Object which has been scanned to illustrate the scanning process.

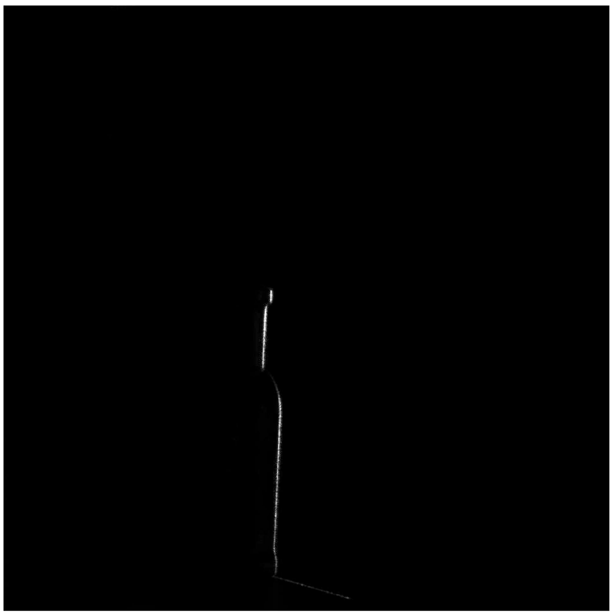


Figure 6.2: One frame taken in equal spatial intervals around the whole object.

6.1.1 Scanning Routine

The whole scanning procedure is implemented in Matlab®; the image acquisition; and the actuation of the turntable.

Image Acquisition

Acquiring images from an industrial camera requires a framegrabber; for this work the camera is connected to the used device via a Camera Link interface. The *Image Acquisition* toolbox in Matlab® is used to communicate directly with the framegrabber. The following code sequence inside the scanning routine initializes the camera for further usage:

```
        vid = videoinput('ni');
        triggerconfig(vid,'manual');
        set(vid, 'ReturnedColorSpace', 'grayscale');
        start(vid);
```

Listing 6.1: Code sequence to initialize the camera.

To acquire an image, from the video running in background, a snapshot had to be taken; the command `getsnapshot(vid);` executes this event.

Turntable Control

The process variables, required to actuate the turntable, are controlled from Matlab® by using OPC. The OPC client (Matlab®) communicates with an OPC server which is connected via a PVI interface to the PLC.
There are two sets of variables; read-only variables such as:

- `'rem_ready'` which indicates the readiness of the system;
- `'rem_act_pos'` reports the actual position of the turntable,

and the variables which can be changed by the user:

- `'rem_init'` initializes the system by setting the variables to default;
- `'rem_nom_pos'` sets the position which should be reached by the turntable;
- `'rem_speed'` designate the speed of the turntable;

- `'rem_to_pos'` starts the motion and the turntable drives to the designated position.

The configuration of the OPC connection in Matlab® is written as:

```
da = opcda('localhost','B&R.PviOPC.2')
pause(0.05); % waiting for a response of the OPC server
connect(da)
input('Press RETURN TO START!')
grp = addgroup(da)
```

Listing 6.2: Code sequence to initialize the OPC server for further usage.

With the command `additem(grp,'variableName')` the process variables are defined; all the variables must exist on the PLC under the same name.

The image acquisition process requires the choice of an adequate resolution for the desired application; a high resolution raises the number of taken images by shortening the angle φ between them. The programmed scanning routine starts the acquisition sequence autonomously and ends with storing the data set into one file. This file is a structured array containing n separate datasets; one for each image ($1024x1024$ pixels) taken in equal spatial intervals around the object. Figure 6.2 shows an example of one of these images.

6.2 Processing

The processing part includes the following steps:

- extraction of the laserline;
- transformation of the pixel coordinates into lasercoordinates;
- concatenation of the separate scans into one structured point cloud; and
- the optimization of the dataset.

6.2.1 Extraction of the Laserline

The position of the laser in the image reflects the geometry of the object. Consequently, it is necessary do determine the position of the laser in the image with high accuracy;

independent of the illumination conditions.

A typical acquired image is shown in Figure 6.2. It should be observed, that the intensity of the laserline in the images is not uniform. Furthermore, there may be occlusions in some measurements, i.e., the laserline is not visible in some portions of the image. Consequently there are two steps required in determining the position of the laserline:

1. Is the maximum intensity $I_{m,j}$ in an individual row sufficiently high that it may be assumed that the laser is visible in the row, and
2. in the rows where the laser is visible, the position of the laser must be determined. The local intensity of the observed laserline depends on the local surface properties and the angle with which the laser impinges on the surface.

Direct binarization of the image with a fixed global threshold t, does not deliver satisfactory results (see Figure 6.3(a)).

The position of the laser is determined in the following manner:

1. The maximum intensity $I_{m,j}$ is determined in each row,if it is greater than a predefined threshold t, then the row is subjected to further processing.
2. An individual threshold t_r is determined for each row by multiplying the maximum value by a predefined factor λ
$$t_r = \lambda I_{m,j} \tag{6.1}$$
3. Starting from the maximum intensity value in each row the standard deviation of the remaining pixels in the row is computed; the pixels within a predefined deviation σ are considered as the pixels of the laserline.
4. The first geometric moment (see Section 4.6) for all these pixels in a row are computed.

This procedure leads to a set of points which gives a smooth estimation of the laserline in pixel coordinates. Figure 6.3(b) illustrates the result of a computation done in this manner. Implemented in Matlab® the binarization could be done in the manner shown in Listing 6.3.

```
%defining the limit of the standard deviation
devLim = 5;
% performing all he following computations for each row
for i = 1  to noRows
    row = F(i,:);
  [mVal, mAt] = max(row);
  %defining the global threshold
  threshGlobal = lambda * IntMax
  if mVal > threshGlobal
    threshRow = mVal * kappa;
    % finding all the pixel grater than the local threshold
    ind = find( row > threshRow );
    % checking the standard deviation of the pixels
    if std(ind) < devLim
        pix(i,ind) = 1;
    end
    end
end
```

Listing 6.3: Pseudo code of how the implementation of the the adaptive threshold binarization could be done.

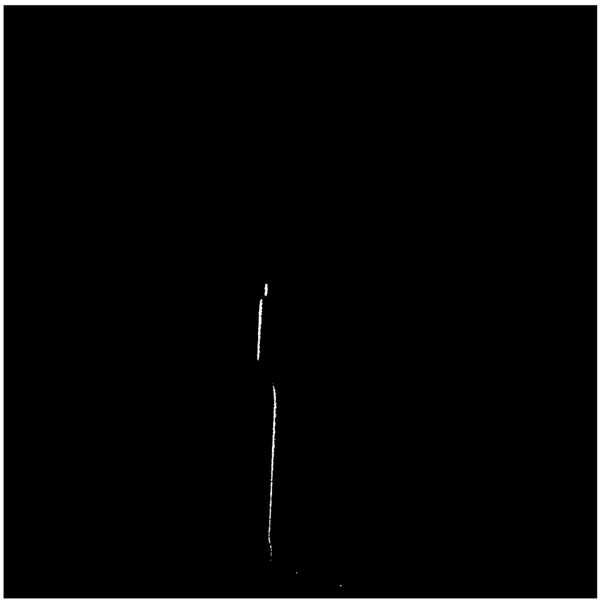
Figure 6.3(b) illustrates the result of using an adaptive threshholding method.
The binarized image is used to extract the center of the laserline. The 1. moment of the intensity (see Section 4.6) from the white pixels in the image are computed; this leads to a set of points which gives a smooth estimation of the laserline in pixel coordinates.

6.2.2 Transformation

With the projection matrix P, acquired from the calibration, the homogeneous coordinates of the points on the laserline are transformed from the pixel- into the laser coordinate system with

$$\mathsf{L} = \mathsf{PG}, \tag{6.2}$$

where G is a $3xk$ matrix containing the k homogeneous coordinates of the laserline and L contains the coordinates of the corresponding points in the lasercoordinatesystem. Converting them to homogeneous coordinates and applying the computations for the turntable registration (see Section 5.3), the height and also the depth information from points on the object are now available in mm. Plotting these points lead to an image which is shown in Figure 6.4. This computations have to be done for all acquired images.


(a) Conventional

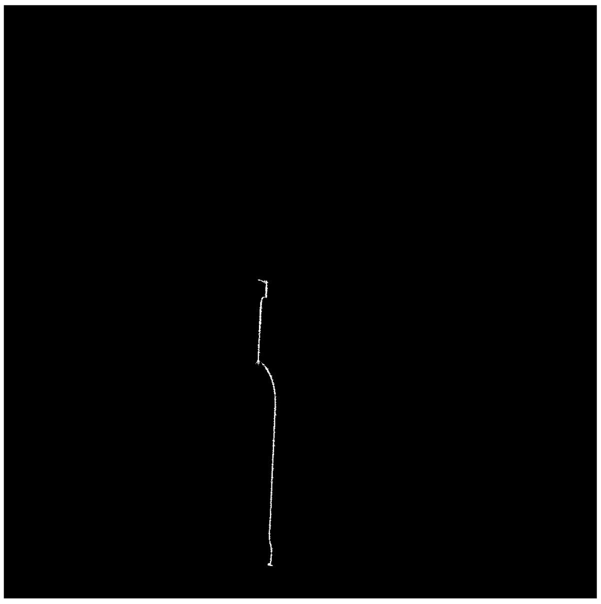

(b) Adaptive

Figure 6.3: Difference between the Matlab® and the adaptive thresholding method.

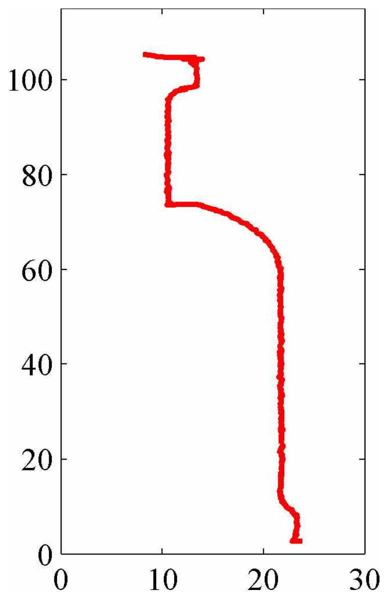


Figure 6.4: Points on the laserline containing the correct dimensions in *mm*.

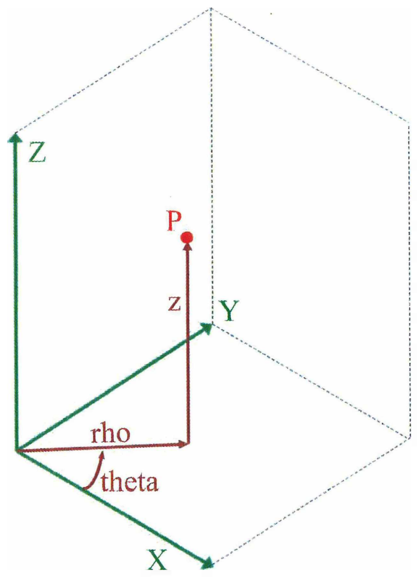


Figure 6.5: Cylindrical and cartesian coordinate system

6.2.3 Concatenation

In order to build a 3-dimensional surface model, the individual data sets from each image have to be merged into one structured point cloud. The two dimensional lasercoordinates have to be transformed into 3-dimensions; the missing coordinate comes from the rotation angle of the table which is determined during the scanning process.

For the reconstruction of a rotating object the use of a cylindrical coordinate system is suitable. Figure 6.5 illustrates the two coordinate systems. The relations between the cylindrical (ρ, θ, z) and the cartesian coordinates (X, Y, Z) are:

$$\theta = \arctan\left(\frac{x}{y}\right) \tag{6.3}$$

$$\rho = \sqrt{x^2 + y^2} \tag{6.4}$$

$$\zeta = Z. \tag{6.5}$$

For the measurement θ comes directly from the information of the rotation angle in the scan; the height z correspond to the Y value of the lasercoordinate system; and ρ comes from the X information of the lasercoordinate system. So it is possible to concatenate data from each scan into one structured dataset (see Figure 6.6). Now the dataset consists of three matrices ρ, φ and z which contain the coordinates of each point.

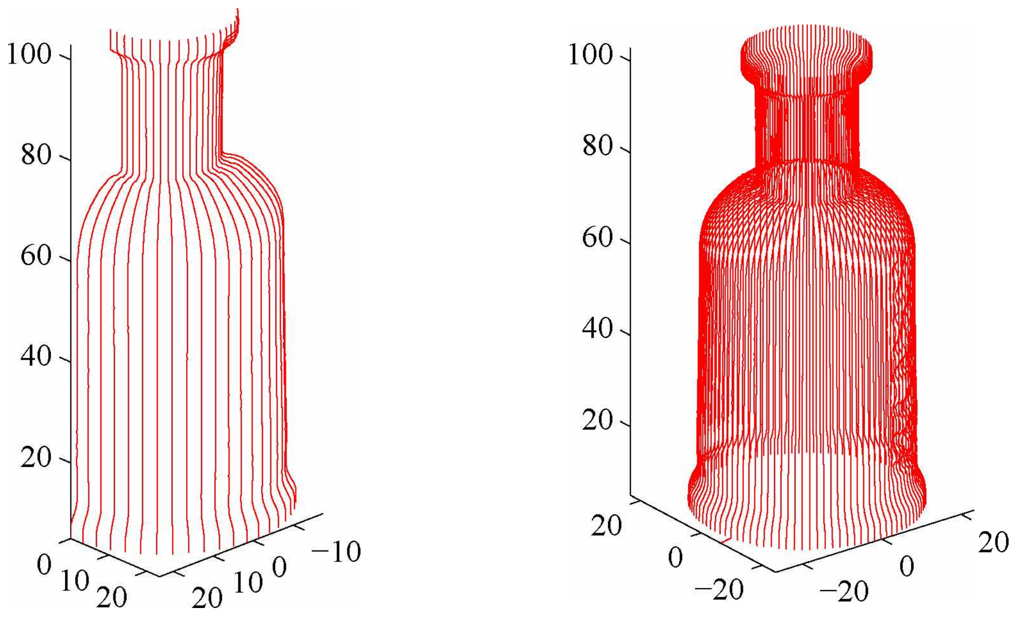


Figure 6.6: Partly and complete concatenated point cloud

6.2.4 Point Cloud Processing

Performing point cloud processing to the data set have no influence on the accuracy of a measurement; it improves the structure of the acquired data for further use by

1. patching the bad spots in the dataset;
2. smoothing the surface geometry; and
3. reducing the amount of points.

In this work the main focus were placed on improving the accuracy and reproducibility of the measurements; only basic computations on patching and smoothing the data were performed. Figure 6.7 illustrates the improvements gained from the padding and smoothing process.

Data Patching

If the laserline has been occluded at some sections during the scan there are bad spots in the dataset. Due to the fact, that surface information in these regions are not available,

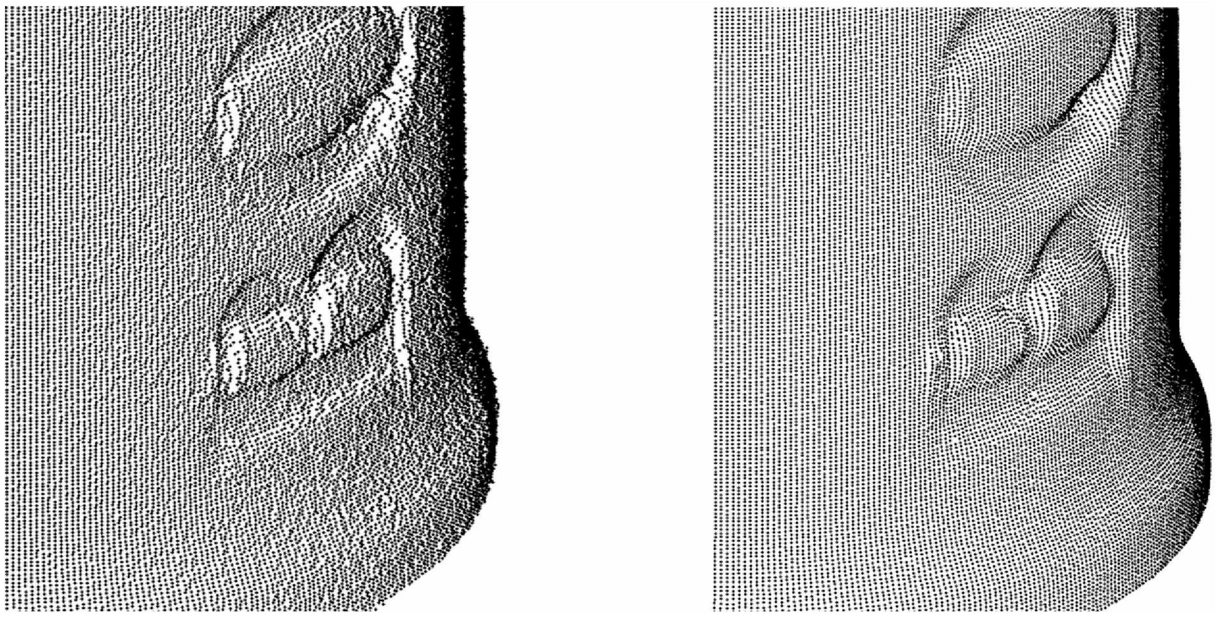


Figure 6.7: Comparison between original (left) and smoothed and patched data set (right).

the imperfections cannot be eliminated completely; but in order to avoid leakage in the dataset it has to be padded. The defective areas in the matrices, which are represented by NaN's, are removed by a linear interpolation between the boundary values of each hole. If the leak occurs at the margin of the dataset an interpolation is not possible.

The first step in the patching procedure is to eliminate the NaN's on the margin of a dataset. All the NaN's at the margin are replaced by the first value. In Matlab® the elimination of NaN's at the beginning and end of each column could be written as:

```
[noRows, noCols] = size(data);
for k =1 to noCols
    col = data(:,k);
    if isnan(col(1))
        at = 1;
        indices = at;
        while (isnan(col(at + 1))) and (at < noCols)
            at = at + 1;
            indices = [indices, at];
        end
        col(indices) = col(at + 1);
    end
end
```

Listing 6.4: Pseudo code for the elimination of the NaN's at the border of the data set.

The next step is to perform the linear interpolation between the remaining holes in the dataset.

```
at = 1;
while at ≤ (noRows - 2)
    at = at + 1;
    if isnan( col(at) )
        % start of a hole has been found
        start = at;
        % find the end of the hole
        while isnan(col(at))
            at = at + 1;
        end
        stop = at;
        % generate the linear interpolation
        vals = linspace( col(start-1), col(stop), stop - start + 1);
        % fill the hole
        col(start:stop) = vals';
    end
end
```

Listing 6.5: Pseudo code for interpolating between the gaps in the dataset.

Smoothing

To eliminate small distortions of the laserline, mainly caused by reflections, a standard smoothing routine from the Matlab® library is applied on the data. The command "`Xs = smooth(X)`" uses a moving 5 point average to smooth the dataset either line by line or column by column. Some other smoothing filters such as the "Savitzky-Golay", or the "Quadratic Fit" are distributed in Matlab®. Figure 6.7 illustrates the improvements gained from the padding and smoothing process.

6.3 Example Scans

The general applicability of the 3D scanner is demonstrated with measurements of additional objects. At the right side the 3-dimensional surface reconstruction is shown, whereas on the left side there is the original photograph of this object. The first two examples are almost rotational invariant; the reconstruction is possible without any holes in the dataset. On the other three objects the laserline was partially hidden during the scanning process; a reconstruction of the occluded areas is not possible (see Section 6.5).

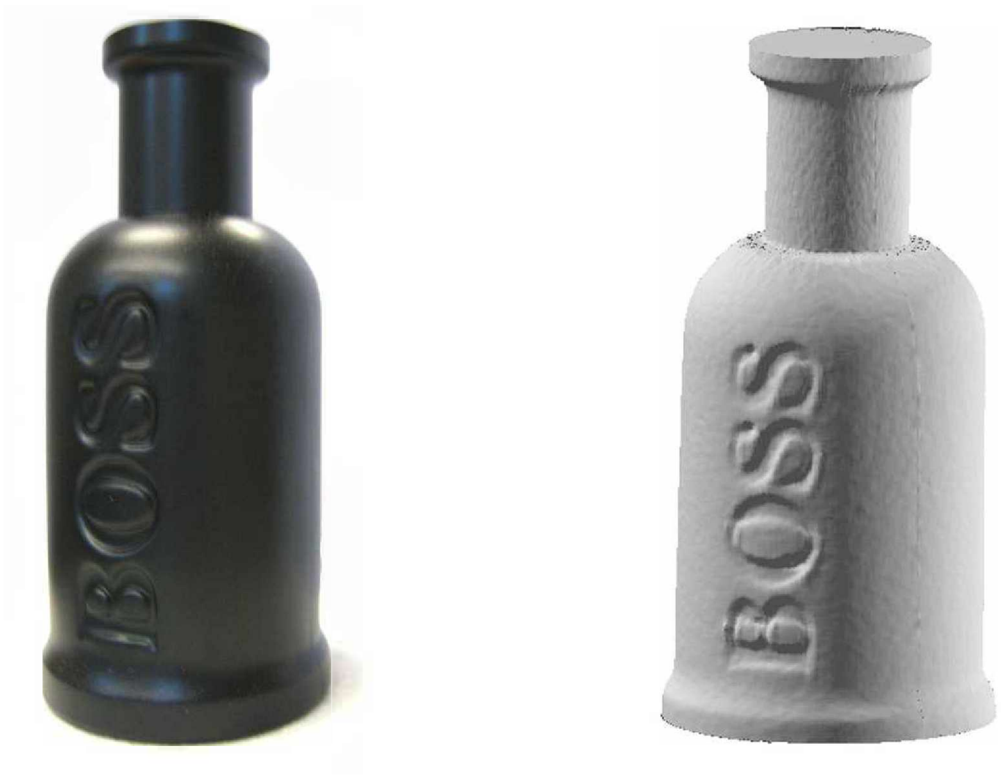


Figure 6.8: Example of the scan of a parfum bottle (left:photograph, right: reconstruction).

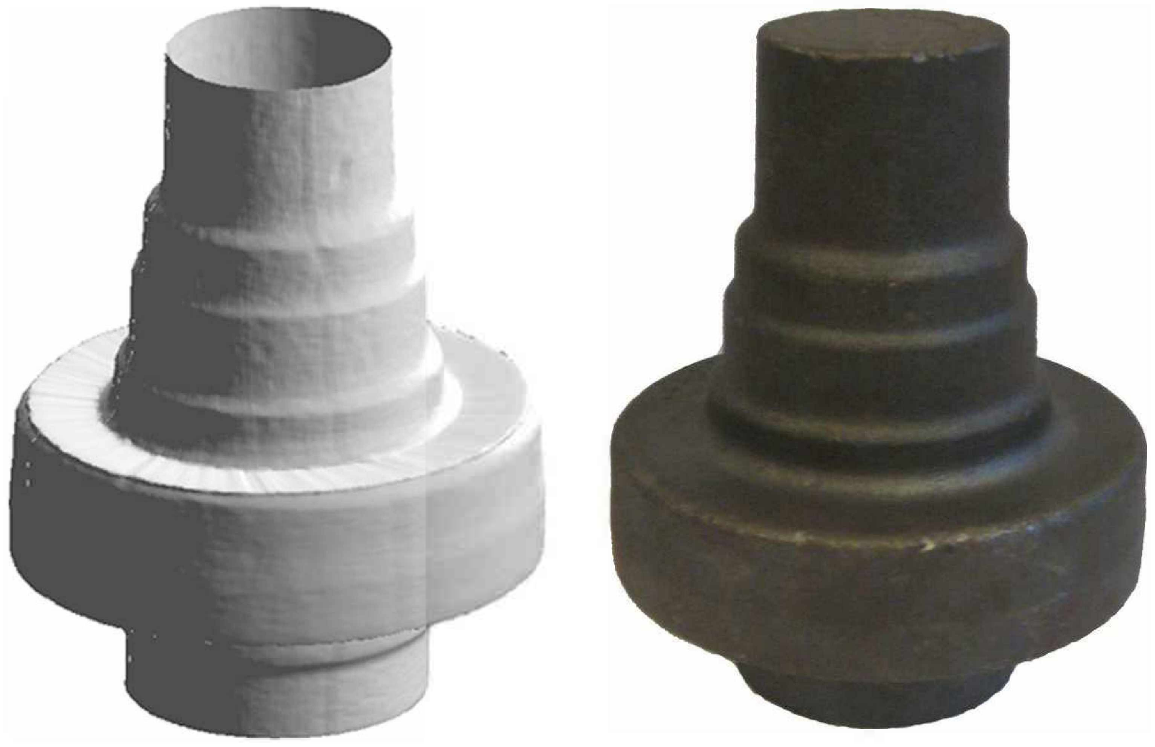


Figure 6.9: Example of a scanned rotational invariant casting part(left:photograph, right: reconstruction).

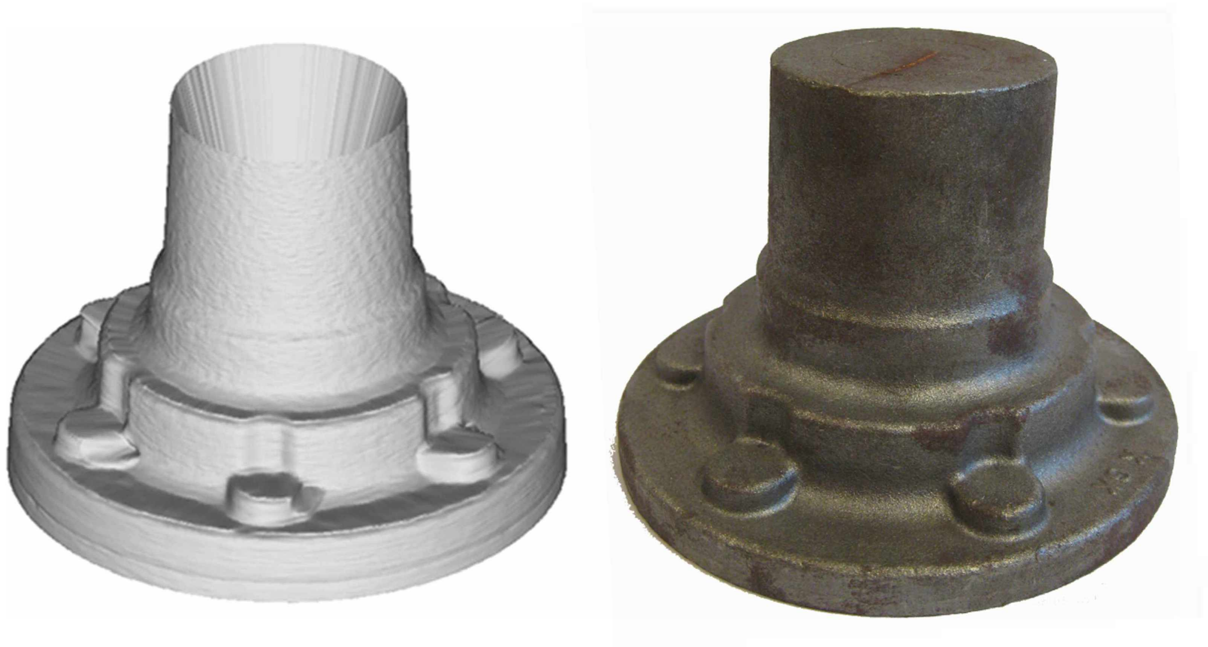


Figure 6.10: Example of a scanned mainly rotational invariant casting part(left:photograph, right: reconstruction).

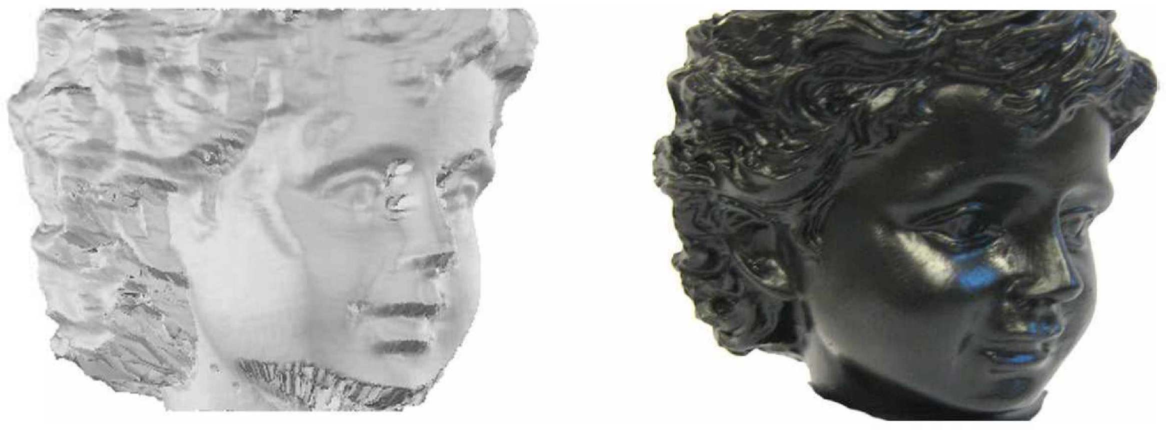


Figure 6.11: Scanned head of a little statue (left:photograph, right: reconstruction)).

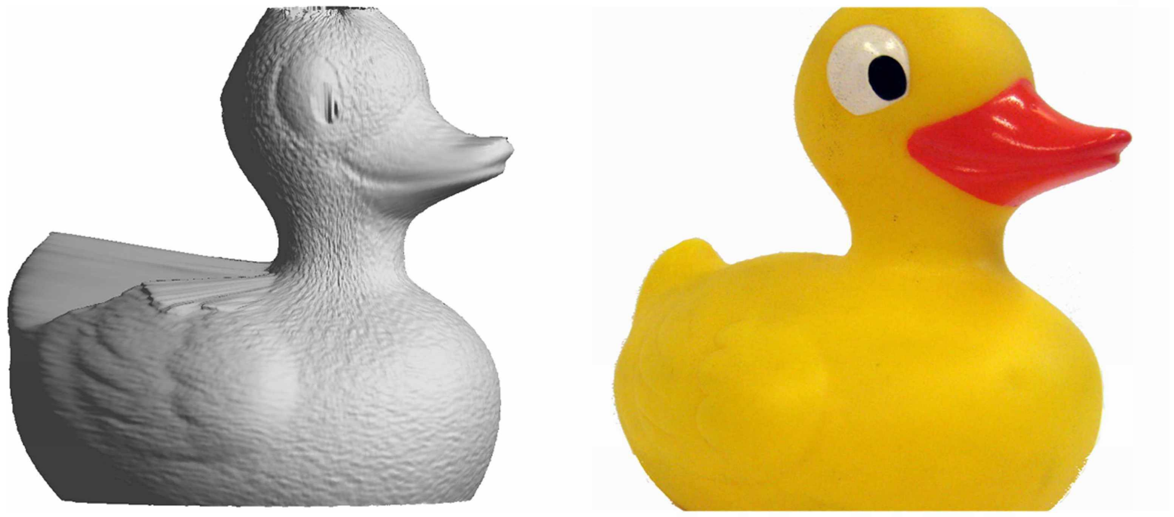
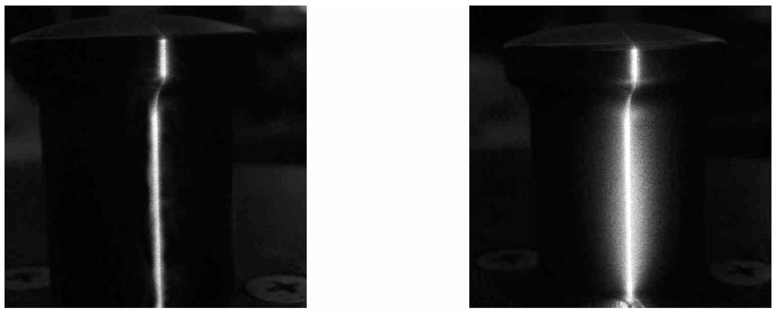


Figure 6.12: Scanned rubber duck (left:photograph, right: reconstruction).

6.4 Accuracy

It has to be mentioned, that the absolute measurement accuracy of a non contact laser-scanner could never compete with the accuracy achieved with a coordinate measuring machine (CMM); their accuracy is with 0.01 to 0.02 mm approximately 10 times higher. Accuracy means not only the absolute exactness of the measurement, this term also implies the reproducibility of measurements. Unfortunately there was no calibrated test specimen available to demonstrate the absolute accuracy of the present setup; however it is higher than the accuracy which could be achieved by a human with a sliding caliper.



(a) Matt (b) Glossy

Figure 6.13: Reflection of the laserline on two different surfaces.

But for many reverse engineering and quality control applications a high reproducibility factor is required which can be ensured with this system. The advantages of such laserscanners are lying in the capability to reproduce complex surface geometries within a short time period and high quality, which could be never done with a CMM.

6.5 Problems

There are several difficulties that can appear during the measurement:

1. reflections of the laser on the surface;
2. influence of ambient light; and
3. occlusion of the laserline;

Reflections
can occur when the surface, on which the laserbeam impinges, is too glossy; it gets almost impossible to extract the laserline exactly when its shape is not clearly defined. Figure 6.13 shows an example of a suitable and an unfeasible surface. In order to solve this problem, the surface can be either coated with matt paint or powdered with e.g. chalk.

Ambient light
leads to almost the same problem as the reflections. To avoid ambient light in an image the use of two methods are possible:

- usage of an optical interference filter which matches to the wavelength of the used laser (see Figure 6.14); or

- to shade the object completely from distracting light sources.

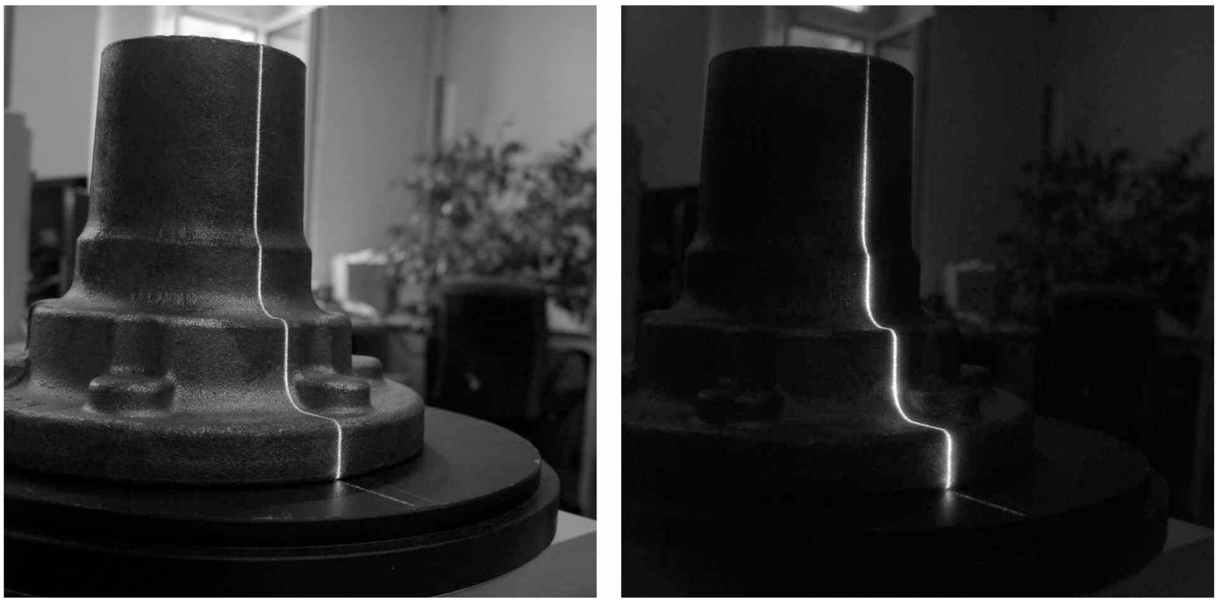


Figure 6.14: Image taken without an optical interference filter (left) and image taken with filter (right).

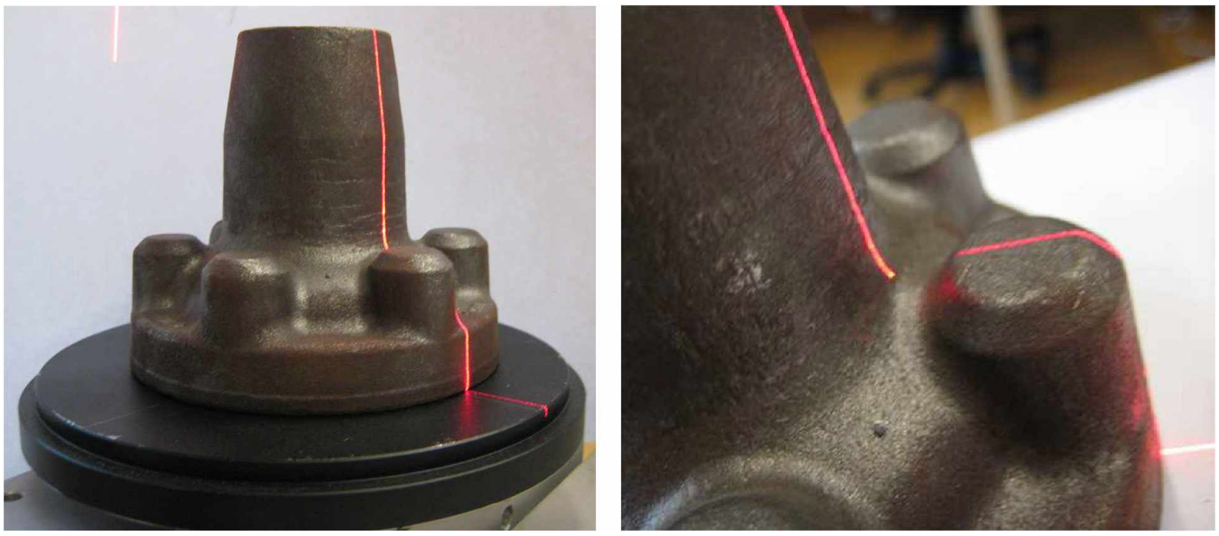


Figure 6.15: Occlusions of the laserline on the surface

Occlusions

of the laserline are causing leakage in the dataset. If the laserline is hidden by some parts of the object (see Figure 6.15) and could not be captured by the camera over the complete height, the information in these areas are missing. An arrangement of two or more cameras and lasers can be used to acquire complete datasets.

Besides an improvement of the surface quality, the use of two or more cameras implicates a massive increase of data and processing time. Furthermore the combination of more cameras requires more diligence in calibration.

Chapter 7

Generating CAD Data

To use a 3-dimensional scan of an object in a meaningful engineering application it is necessary to convert it into a file format that is compatible with a CAD software program. The most popular file format for reverse engineered objects is the surface tesselation language (STL) [15], because of its robustness and simplicity. This file format allows the interchange of data between the CAD software packages of various producers.

In this chapter the export of a scanned object in the STL file format is shown. It is also demonstrated that the import of this file format is straight forward in any CAD program.

The STL file is an easy to understand format with an almost primitive structure. Actually it consists only of the coordinates of the vertices and their normal vector to describe an object. The STL file can be generated in either an ASCII or Binary format. The ASCII code is easier to read and understand so in this work the focus lies only on the STL format written in ASCII code. However, the ASCII files are significantly larger than the binary format.

7.1 Data Export

The STL data has the following structure:

```
solid solidname
    facet normal Nx Ny Nz
```

```
        outer loop
            vertex X1 Y1 Z1
            vertex X1 Y2 Z2
            vertex X3 Y3 Z3
        endloop
    endfacet
    ...
    ...
    ...
    facet normal Nx Ny Nz
        outer loop
            vertex Xn-2 Yn-2 Zn-2
            vertex Xn-1 Yn-1 Zn-1
            vertex Xn Yn Zn
        endloop
    endfacet
```

Listing 7.1: Structure of a STL file.

STL defines the surface as a collection of triangular patches. Each patch is defined by the coordinates of their vertices x_1, y_1, z_1; x_2, y_2, z_2 and x_3, y_3, z_3. Additionally a surface normal vector with the coordinates $\boldsymbol{n}_x$, $\boldsymbol{n}_y$ and $\boldsymbol{n}_z$ is used to define the surface orientation. Given three matrices $\mathsf{M}_x(i,j)$, $\mathsf{M}_y(i,j)$ and $\mathsf{M}_z(i,j)$ containing the data points of an object surface, where $i = 1, ...m$ and $j = 1, ...n$, The triangles are formed in a manner shown in the pseudo code (see Listing 7.2); the procedure for the two other matrices are the same.

```
for j = 1 : n
  for i = 1 : (m-1)
    connect Mx(i,j) to Mx(i+1,j);
  end
end
for j = 1 : (n-1)
    for i = 1 : m
    connect Mx(i,j) to Mx(i,j+1);
  end
  for i = 1 : (m-1)
    connect Mx(i,j) to Mx(i+1,j+1);
  end
end
```

Listing 7.2: Pseudo code how the vertices are concatenated to patches.

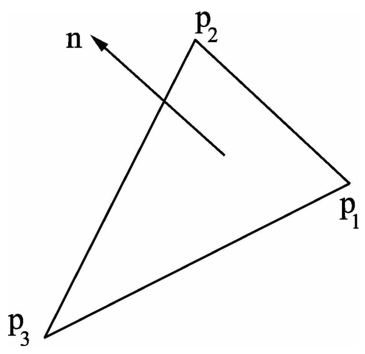


Figure 7.1: Triangle and its normal vector used to generate a STL file.

After concatenating the vertices, to form a triangle, the normalized surface vector for the patch has to be defined. By definition the vertices $(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$ of the triangle, such as in Figure 7.1, must be arranged in a counterclockwise manner. The normals can be computed as follows:
Forming the two vectors $\boldsymbol{V}_{32}$ and $\boldsymbol{V}_{31}$

$$\begin{aligned} \boldsymbol{V}_{31} &= \boldsymbol{p}_1 - \boldsymbol{p}_3 \\ \boldsymbol{V}_{32} &= \boldsymbol{p}_2 - \boldsymbol{p}_3 \end{aligned} \tag{7.1}$$

Calculating the vertex normal with the cross-product:

$$\boldsymbol{n} = \boldsymbol{V}_{31} \times \boldsymbol{V}_{32} \tag{7.2}$$

A high scanning rate, which causes a big amount of data points, also ends up in a huge STL file after the triangulation. To reduce file size a reduction of data points is required. In this work a data reduction is not implemented, but for the sake of completeness it is mentioned here. Without loosing too much information of the object surface it is only useful to reduce data points in planar regions. In principle the data reduction is performed by comparing the normalized normal vectors of the neighboring triangles. On a planar surface the normal vectors of all the triangles are pointing almost in the same direction; fewer but bigger triangles don't change anything in accuracy but leads to a smaller STL file; for detailed information how data reduction is performed see [16].

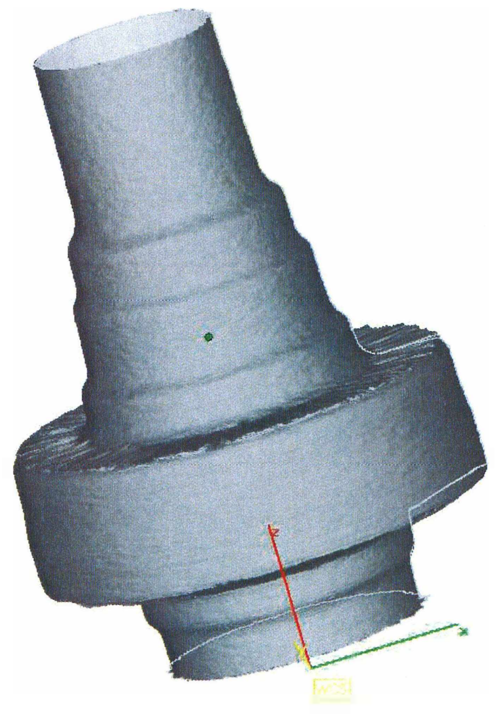


Figure 7.2: STL object imported into MiniMagics 2.0

7.2 Data Import

As the standard file format for rapid prototyping and reverse engineering the STL format is compatible to nearly all commercial CAD software packages. Figure 7.2 shows an example of an imported object into the MiniMagics 2.0 software. Once the dataset is imported into an arbitrary CAD software package it is possible to convert it into another file format, generate a blue-print or edit the scanned object.

Chapter 8

Conclusion

In this work a complete experimental 3D laserscanner was designed and implemented. It has been shown that such a system is well suited to perform measurements of rotational invariant objects. A complete calibration process as well as a new approach to registering the turntable were implemented; the entire mathematical background of the used method has been presented. With a specially designed registration target it is possible to eliminate assembly errors in the setup. Applying a careful calibration in connection with the registration procedure the laserplane of light system is suitable for reverse engineering applications; a very high reproducibility of the system has been achieved.
Also a method to convert the acquired 3D data into a STL file format was implemented, so that data can be exported to the most common CAD systems (e.g. CATIA®, AutoCAD®, SolidWorks® etc.)

8.1 Problems of this method

One of the problems which appeared during this work is the problem of holes in the scanned dataset, where either the laser cannot be captured by the camera due to occlusion, or the laser is not able to reach the surface spot because of undercuts. Another difficulty is the variety of surface features an object can have; the scattering of the laserbeam is reflected differently on the surface, this makes the retrieval of a proper threshold and consequently the extraction of the laserline nearly impossible.

8.2 Future Work

Although this work made a great improvement using a 3D laserscanner as a reverse engineering tool, there is a big field of unexplored topics left:

1. installation of two or more lasers to avoid occlusion areas on the surface;
2. installation of two ore more cameras do circumvent occlusions;
3. investigate reflectance models of various surfaces in combination with the used laser;
4. apply high sophisticated surface fitting algorithms in order to get smoother surfaces and get more independent from runaway values in the measurement data; and
5. use better algorithms to reduce the amount of points after the scanning process, to reduce the size of the generated CAD file format.

Furthermore the calibration, which is supposed to be the most important step could be improved with the construction of a calibration target especially designed for rotational motion scanners. With a new target a non-static calibration process is conceivable. A target whose dimensions (height and depth) matches better to the size of the test specimens would also lead to more accuracy in measurement.

Bibliography

[1] A. Gfrerer, *Analytische Projektive Geometrie. Lecture notes.* Graz University of Technology, Austria, 2003.

[2] R. Ofner, "Three-dimensional measurement via the light-sectioning method," Ph.D. dissertation, University of Leoben, Leoben, AUT, 2000.

[3] V. Raja and K. J. Fernandes, *Reverse Engineering: An Industrial Perspective*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[4] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[5] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3d model construction for turn-table sequences," in *Proceedings of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, ser. SMILE'98. London, UK: Springer-Verlag, 1998, pp. 155–170. [Online]. Available: http://portal.acm.org/citation.cfm?id=646488.694783

[6] X. Liu, H. Yao, X. Chen, and W. Gao, "Shape from silhouettes based on a centripetal pentahedron model," *Graphical Models*, vol. 70, no. 6, pp. 133 – 148, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6WG3-4T72WX1-1/2/44e75247dbc29bd5bd9c3920a0003714

[7] P. Eisert, "3-d geometry enhancement by contour optimization in turntable sequences," 1998.

[8] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," in *International Journal of Computer Vision*, vol. 35, 1997, pp. 1067–1073. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.7242

[9] F. Pernkopf and P. O'Leary, "Image acquisition techniques for automatic visual inspection of metallic surfaces," *NDT & E International*, vol. 36, no. 8, pp.

609 – 617, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/B6V4C-49506KB-2/2/7c4ce97f2593235b7242e950b435721a

[10] H. J. Bartsch, *Taschenbuch Mathematischer Formeln*, 18th ed. Fachbuchverlag Leipzig im Carl Hanser Verlag, 1999.

[11] M. T. Ewald Fauster, Peter Schalk, "Calibration method for light sectioning measurement system," *Skiathos: WESEAS Press*, 2002.

[12] A. D. B. Carlo Colombo, Dario Comanducci, "Low-cost 3d scanning by exploiting virtual image symmetries," *Journal of Multimedia VOL. 1 No. 6*, 2006.

[13] S. Y. Z. Vladimir A. Knyaz, "Vision based technique for photorealistic 3d reconstruction of historical items," 1.

[14] Y. Yemez and C. Wetherilt, "A volumetric fusion technique for surface reconstruction from silhouettes and range data," *Computer Vision and Image Understanding*, vol. 105, no. 1, pp. 30 – 41, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6WCX-4KWTFND-2/2/2f58873cb767abed1f09b32b48287ee5

[15] T. Wu and E. Cheung, "Enhanced stl," *The International Journal of Advanced Manufacturing Technology*, vol. 29, pp. 1143–1150, 2006, 10.1007/s00170-005-0001-5. [Online]. Available: http://dx.doi.org/10.1007/s00170-005-0001-5

[16] Y. Z. W. Y. H. Chen, C.T. Ng, "Generation of an stl file from 3d measurement data with user-controlled data reduction," *International Journal of Advanced Manufacturing Technology, 1999*, vol. 15, pp. 127–131, 1999.

List of Figures

List of Tables

Listings

Appendix A

Code

Listing A.1: Image acquisition code.

```
%Purpose:
%Image acquisition routine for a 3D laser scanner using a turntable.
%
% Return Parameters: frame, cell array containing the acquired images
%
% References : none
%
% Author :   Bernhard Mörtl
% Date :     17 August 2010
% Version : 1.0
%
% (c) 2010, Bernhard Mörtl,
% Instutite for Automation, University of Leoben, Leoben, Austria
% email: automation@unileoben.ac.at, url: automation.unileoben.ac.at
%
% History:
%   Date:                   Comment:
%   17 August 2010          Original Version 1.0
%==========================================================================
clear all;
close all;
clc;
% Defining the stepsize
noPics = 500;
stepSize = round(50000/noPics);
%
```

```
% Initializing the camera
vid = videoinput('ni');
triggerconfig(vid,'manual');
set(vid, 'ReturnedColorSpace', 'grayscale');
start(vid);
%
% Initializing the OPC connection
da = opcda('localhost','B&R.PviOPC.2')
pause(0.05);
connect(da);
input('Press RETURN TO START!')
grp = addgroup(da)
set(grp,'updateRate',0.00)
%
% Defining the group variables
itm1 = additem(grp,'rem_act_pos')
itm2 = additem(grp,'rem_init')
itm3 = additem(grp,'rem_ready')
%
possoll = additem(grp,'rem_nom_pos');
posstart = additem(grp,'rem_to_pos');
posspeed = additem(grp,'rem_speed');
%
% Initializing the turntable
pause(0.05);
write(itm2,1);
pause(0.05);
write(itm2,0);
pause(2);
ready=read(itm3);
%
if ready.Value==1
    % Scanning process
    for i = 1:noPics
        % taking an image
        frame{i} = getsnapshot(vid);
        % moving to next position
        write(possoll, stepSize);
        pause(0.05);
        write(posspeed, 1100);
        pause(0.05);
        write(posstart,1);
        pause(0.05);
        write(posstart,0);
        pause(0.5);
    end
end
% storing the acquired images on the harddisk
save('Scan','frame');
```

Listing A.2: Calibration algorithm.

```
%Purpose:
%Calibration routine of a light sectioning equipment.
%
% Return Parameters: ProjektionKameraLaser, 4x4 transformation matrix
%
% References : none
%
% Author :  Bernhard Mörtl
% Date :    25 August 2010
% Version : 1.0
%
% (c) 2010, Bernhard Mörtl,
% Instutite for Automation, University of Leoben, Leoben, Austria
% email: automation@unileoben.ac.at, url: automation.unileoben.ac.at
%
% History:
%   Date:                Comment:
%   25 August 2010       Original Version 1.0
%==========================================================================
%
close all;
clear all;
setupGraphics(12);
%
% Load the two calibration images
load calibration_LAS;
Laser = frame;
load calibration_LED;
%
% convert calibration image into binarye
thresh = graythresh(frame);
frameBW = im2bw(frame, thresh);
frameBW = bwareaopen(frameBW, 50);
%
%labeling the LED dots
[labels, noLabels] = bwlabel(frameBW);
%
if noLabels > 30
    error('please choose a lower threshhold')
elseif noLabels < 30
    error('please choose a higher threshhold');
end
%
```

```
%compute the center of gravitiy from all the LED dots
for i = 1:noLabels
    temp = labels;
    indices = find(labels ≠ i);
    temp(indices) = 0;
    [zeile, spalte] = find(temp);
    cog{i} = computeCOG(zeile, spalte);
end
% Define the Points in the Pxel and in the world coordinate system
PixelHinten = [ cog{2}(1) cog{8}(1) cog{14}(1) cog{20}(1) cog{26}(1)...
                cog{1}(1) cog{7}(1) cog{13}(1) cog{19}(1) cog{25}(1);...
                cog{2}(2) cog{8}(2) cog{14}(2) cog{20}(2) cog{26}(2)...
                cog{1}(2) cog{7}(2) cog{13}(2) cog{19}(2) cog{25}(2);...
                    1         1          1          1          1 ...
                    1         1          1          1          1];
%
PixelMitte = [ cog{4}(1) cog{10}(1) cog{16}(1) cog{22}(1) cog{28}(1)...
               cog{3}(1) cog{9}(1)  cog{15}(1) cog{21}(1) cog{27}(1);...
               cog{4}(2) cog{10}(2) cog{16}(2) cog{22}(2) cog{28}(2)...
               cog{3}(2) cog{9}(2)  cog{15}(2) cog{21}(2) cog{27}(2);...
                   1         1           1          1          1 ...
                   1         1           1          1          1];
%
PixelVorne = [ cog{6}(1) cog{12}(1) cog{18}(1) cog{24}(1) cog{30}(1)...
               cog{5}(1) cog{11}(1) cog{17}(1) cog{23}(1) cog{29}(1);...
               cog{6}(2) cog{12}(2) cog{18}(2) cog{24}(2) cog{30}(2)...
               cog{5}(2) cog{11}(2) cog{17}(2) cog{23}(2) cog{29}(2);...
                   1          1          1          1          1  ...
                   1          1          1          1          1   ];
%
WeltHinten = [ 174     174      174      174      174 ...
                24      24       24       24       24;...
               -70     -35        0       35       70 ...
               -70     -35        0       35       70;...
                 1       1        1        1        1 ...
                 1       1        1        1        1 ];
%
WeltMitte = [ 139      139      139      139      139 ...
               59       59       59       59       59;...
              -70      -35        0       35       70 ...
              -70      -35        0       35       70;
                1        1        1        1        1 ...
                1        1        1        1        1 ];
%
```

```
WeltVorne = [ 119.5    119.5    119.5    119.5    119.5...
               79.5     79.5     79.5     79.5     79.5 ;...
              -70      -35        0       35       70 ...
              -70      -35        0       35       70;...
                1        1        1        1        1 ...
                1        1        1        1        1];
%
%computation of the homography between pixel- and world coordinate system
hHinten = homography(PixelHinten, WeltHinten);
hMitte = homography(PixelMitte, WeltMitte);
hVorne = homography(PixelVorne, WeltVorne);
%
%perform adaptive Threshholding
LaserBW = adaptThresh(Laser);
LaserBW(910:end,:) = 0;
LaserBW = bwmorph(LaserBW, 'dilate');
LaserBW = bwareaopen(LaserBW, 500);
%
%labeling the laser segments
[laserLabels, noLaserLabels] = bwlabel(LaserBW);
if noLaserLabels ≠ 5
    error('please take another set of calibration pictures');
end
abwLimit = 10;
LaserN = zeros(size(LaserBW));
for i = 1:length(LaserBW)
    temp = LaserBW(i,:);
    indices = find(temp);
    if std(indices) ≤ abwLimit
        LaserN(i,indices) = Laser(i,indices);
    end
end
%
% Computing the first geometric moment of the laser line
m = 5;
n = 3;
Schwerpunkt = zeros(2,1024);
for i = 1:length(LaserN)
    %
    temp = LaserN(i,:);
    [C cols] = max(temp);
    rows = i;
    %
    if (rows ≥ 6) && (¬isempty(cols)) && (C > 0) && (rows ≤ 1018)
```

```
            j = cols - m;
            k = rows - n;
            SummeObenX = [];
            SummeUntenX = [];
            %
            for k = k : rows + n
                for j = j :cols + m

                    if isempty(SummeObenX)
                        %
                        SummeObenX = 0 + j*LaserN(k,j)^2;
                        SummeUntenX = 0 + LaserN(k,j)^2;
                    else
                        SummeObenX = SummeObenX + j*LaserN(k,j)^2;
                        SummeUntenX = SummeUntenX + LaserN(k,j)^2;
                        %
                    end
                end
            end
            if ¬isempty(SummeObenX)
                Schwerpunkt(2,i) = SummeObenX / SummeUntenX;
            end
        end
    end
    %
    % segmenting the point on the laserline on the 3 stages of the target
    Schwerpunkt(1,:) = 1:1024;
    indices = find(isnan(Schwerpunkt(2,:)));
    Schwerpunkt(2,indices) = 0;
    [rows cols] = find(Schwerpunkt == 0);
    Schwerpunkt(:, cols) = [];
    SchwerpunktTemp = Schwerpunkt;
    indexVorne = find(SchwerpunktTemp(2,:) > 491);
    SchwerpunktTemp(:,indexVorne) = 0;
    indexMitte = find(SchwerpunktTemp(2,:) > 445);
    SchwerpunktTemp(:,indexMitte) = 0;
    indexHinten = find(SchwerpunktTemp(2,:));
    %
    % fitting lines through the center points of the LED dots
    QuerLinieHintenObenP = [cog{2}(2)  cog{8}(2)  cog{14}(2) ...
                             cog{20}(2) cog{26}(2);            ...
                             cog{2}(1)  cog{8}(1)  cog{14}(1) ...
                             cog{20}(1) cog{26}(1);            ...
                                 1          1          1      ...
                             1          1];
%
QuerLinieHintenUntenP =[cog{1}(2)  cog{7}(2)   cog{13}(2)...
                        cog{19}(2) cog{25}(2);            ...
                        cog{1}(1)  cog{7}(1)   cog{13}(1)...
                        cog{19}(1) cog{25}(1);            ...
                             1          1           1     ...
                             1          1];
%
QuerLinieMitteObenP =  [cog{4}(2)  cog{10}(2)  cog{16}(2)...
                        cog{22}(2) cog{28}(2);            ...
                        cog{4}(1)  cog{10}(1)  cog{16}(1)...
                        cog{22}(1) cog{28}(1);            ...
                             1          1           1     ...
                             1          1];
%
QuerLinieMitteUntenP = [cog{3}(2)  cog{9}(2)   cog{15}(2)...
                        cog{21}(2) cog{27}(2);            ...
                        cog{3}(1)  cog{9}(1)   cog{15}(1)...
                        cog{21}(1) cog{27}(1);            ...
                             1          1           1     ...
                             1          1];
%
QuerLinieVorneObenP = [cog{6}(2)   cog{12}(2)  cog{18}(2)...
                       cog{24}(2)  cog{30}(2);            ...
                       cog{6}(1)   cog{12}(1)  cog{18}(1)...
                       cog{24}(1)  cog{30}(1);            ...
                            1           1          1      ...
                            1           1];
%
QuerLinieVorneUntenP = [cog{5}(2)  cog{11}(2)  cog{17}(2)...
                        cog{23}(2) cog{29}(2);            ...
                        cog{5}(1)  cog{11}(1)  cog{17}(1)...
                        cog{23}(1) cog{29}(1);            ...
                             1          1          1      ...
                             1          1];
%
QuerLinieHintenUnten = fitLine(QuerLinieHintenUntenP);
QuerLinieHintenOben = fitLine(QuerLinieHintenObenP);
QuerLinieMitteUnten = fitLine(QuerLinieMitteUntenP);
QuerLinieMitteOben = fitLine(QuerLinieMitteObenP);
QuerLinieVorneUnten = fitLine(QuerLinieVorneUntenP);
QuerLinieVorneOben = fitLine(QuerLinieVorneObenP);
%
```

```
%% fit lines through the laserline on the calibration target
LaserlinieHintenPixel = [Schwerpunkt(2,indexHinten); ...
                         Schwerpunkt(1,indexHinten); ...
                         ones(1,length(indexHinten))];
%
LaserlinieMittePixel = [Schwerpunkt(2,indexMitte);   ...
                        Schwerpunkt(1,indexMitte);   ...
                        ones(1,length(indexMitte))];
%
LaserlinieVornePixel = [Schwerpunkt(2,indexVorne);   ...
                        Schwerpunkt(1,indexVorne);   ...
                        ones(1,length(indexVorne))];
%
LaserLineHinten = fitLine(LaserlinieHintenPixel);
LaserLineMitte = fitLine(LaserlinieMittePixel);
LaserLineVorne = fitLine(LaserlinieVornePixel);
%
%% intersect the laserlines with the lines through the LED dots
PointHintenOben = int2Lines(QuerLinieHintenOben, LaserLineHinten);
PointHintenOben = convert2affine(PointHintenOben);
PointHintenOben = xy2yx(PointHintenOben);
PointHintenUnten= int2Lines(QuerLinieHintenUnten, LaserLineHinten);
PointHintenUnten = convert2affine(PointHintenUnten);
PointHintenUnten = xy2yx(PointHintenUnten);
%
PointMitteOben = int2Lines(QuerLinieMitteOben, LaserLineMitte);
PointMitteOben = convert2affine(PointMitteOben);
PointMitteOben = xy2yx(PointMitteOben);
PointMitteUnten= int2Lines(QuerLinieMitteUnten, LaserLineMitte);
PointMitteUnten = convert2affine(PointMitteUnten);
PointMitteUnten = xy2yx(PointMitteUnten);
%
PointVorneOben = int2Lines(QuerLinieVorneOben, LaserLineVorne);
PointVorneOben = convert2affine(PointVorneOben);
PointVorneOben = xy2yx(PointVorneOben);
PointVorneUnten= int2Lines(QuerLinieVorneUnten, LaserLineVorne);
PointVorneUnten = convert2affine(PointVorneUnten);
PointVorneUnten = xy2yx(PointVorneUnten);
%
% compute the world coordinates of the interesction points
LPointHintenWorld = convert2affine(hHinten*...
                    [PointHintenOben PointHintenUnten]);
LPointHintenWorld(3,:) = 0;
%
```

```
LPointMitteWorld = convert2affine(hMitte*...
                     [PointMitteOben PointMitteUnten]);
LPointMitteWorld(3,:) = 20;
%
LPointVorneWorld = convert2affine(hVorne*...
                    [PointVorneOben PointVorneUnten]);
LPointVorneWorld(3,:) = 30;
%
PWelt = [LPointHintenWorld LPointMitteWorld LPointVorneWorld;...
         1 1 1 1 1 1];
%
LaserT = FitLaserPlane_berni1(LPointHintenWorld,...
                              LPointMitteWorld, LPointVorneWorld);
LaserKoordinatensystem = LaserT * PWelt;
KalibPunkteLaser = LaserKoordinatensystem(1:3,:);
KalibPunkteLaser(2,:) = abs(KalibPunkteLaser(2,:));
KalibPunkteLaser(3,:) = 1;
%
KalibPunktePixel = [PointHintenOben PointHintenUnten PointMitteOben ...
                     PointMitteUnten PointVorneOben PointVorneUnten];
%
% compute the homography between
ProjektionKameraLaser = homography( KalibPunktePixel,KalibPunkteLaser);
save('Projection','ProjektionKameraLaser');
```

Listing A.3: Offset computation.

```
function [offsetSeitlich, offsetTiefe] = computeOffset(l1,l2)
%
%Purpose:
%compute the misalignment between the laser plane and the roation axis of
%an 3D measurment system using a turntable.
%
% Input parameters: l1
%                   l2
%
% Return Parameters: OffsetSeitlich
%                    OffsetTiefe
%
% References : none
%
% Author :  Bernhard Mörtl
% Date :    30 August 2010
% Version : 1.0
%
% (c) 2010, Bernhard Mörtl,
% Instutite for Automation, University of Leoben, Leoben, Austria
% email: automation@unileoben.ac.at, url: automation.unileoben.ac.at
%
% History:
%   Date:               Comment:
%   30 August 2010      Original Version 1.0
%=======================================================================
% define the radius of the two circles of the calibration target
r = 20;
R = 75;
%compute the measured distance between the big and the small circle
x = l2 - l1;
if x ≤ R-r
    error('The distance x has to be at least equal or bigger than R-r ');
end
%
% Compute the horizontal offset of the laserline in respect to the
% rotation axis
offsetSeitlich = sqrt((2 * r ^ 2 * x ^ 2 - R ^ 4 + 2 * R ^ 2 * r ^ 2 ...
                 + 2 * R ^ 2 * x ^ 2 - r ^ 4 - x ^ 4)) / x / 2;
%
% Compute missalignement of the calibration target (the distance between
% the zero-line of the calibration target and the rotation axis)
phi = asin(offsetSeitlich/r);
```

```
l_real = r*cos(phi);
offsetTiefe = l1 - l_real;
```

Listing A.4: Computation of an acquired image sequence.

```
%Purpose:
% Proccessing routine for scanned 3D data acquired with a light
% sectioning system and a turntable.
%
% References : none
%
% Author :  Bernhard Mörtl
% Date :    10 September 2010
% Version : 1.0
%
% (c) 2010, Bernhard Mörtl,
% Instutite for Automation, University of Leoben, Leoben, Austria
% email: automation@unileoben.ac.at, url: automation.unileoben.ac.at
%
% History:
%   Date:               Comment:
%   10 September 2010      Original Version 1.0
%==========================================================================
close all;
clear all;
setupGraphics(12);
%
% Load the projectionmatrix form the calibration and the measurement data
%load valid;
load boss;
load Projection;
load Offset;
%
% extract the laserline from each dataset
for s = 1 to length(frame)
    % binarizing the images with the adptive trhesholding algorithm
    FrameBW{s} = adaptThresh(frame{s}, 0.5, 0.001);
    FrameBW{s} = bwmorph(FrameBW{s}, 'dilate', 5);
    FrameBW{s} = bwareaopen(FrameBW{s}, 500);
    %
    FrameN = zeros(size(FrameBW{s}));
    indices = find(FrameBW{s});
    FrameN(indices) = frame{s}(indices);
    %
    m = 5;
    n = 3;
    LaserLineReady{s} = nan(3,1024);
    for i = 1 to length(FrameN)
```

```
        %
        temp = FrameN(i,:);
        [C cols] = max(temp);
        rows = i;
        %
        if (cols ≥6) and (rows ≥ 6) and (¬isempty(cols)) and (C > 0) ...
                and (rows ≤ 1018) and (cols ≤ 1018)
            %
            j = cols - m;
            k = rows - n;
            SummeObenX = [];
            SummeUntenX = [];
            %
            %compute the first geometric moment of the laser line
            for k = k to rows + n
                for j = j to cols + m
                    %
                    if isempty(SummeObenX)
                        %
                        SummeObenX = 0 + j*FrameN(k,j)^2;
                        SummeUntenX = 0 + FrameN(k,j)^2;
                    else
                        SummeObenX = SummeObenX + j*FrameN(k,j)^2;
                        SummeUntenX = SummeUntenX + FrameN(k,j)^2;
                        %
                    end
                end
            end
            if ¬isempty(SummeObenX)
                temp = [i;(SummeObenX / SummeUntenX);1];
                LaserLineReady{s}(1,i) = i;
                % transform the points of the laser into the laser
                % coordinate system.
                LaserLineReady{s}(:,i) = ...
                    convert2affine(ProjektionKameraLaser*temp);

            end
        end
    end
end
%
% convert the coordinates into cylindrical coordinates
for i = 1 to length(LaserLineReady)
    RHO(:,i)= LaserLineReady{i}(1,:);
    Zpol(:,i)= LaserLineReady{i}(2,:);
end
THETA = zeros(size(RHO));
for i = 1 to length(LaserLineReady)-1
    angle = (2*pi/(length(LaserLineReady)))*i;
    THETA(:,i+1) = angle;
end
THETA(:,end) = 2*pi;
%
% define the height of the object (values in other regions are deleted)
indices = find(Zpol < 0);
Zpol(indices) = nan;
RHO(indices) = nan;
%
indices = find(Zpol < 105);
Zpol(indices) = nan;
RHO(indices) = nan;
%
% delete ne NaN's in the dataset
Zpol = patchcolumnnans(Zpol);
RHO = patchcolumnnans(RHO);
%
%perform the basic smoothing algorithm of Matlab
for i = 1:length(RHO(:,1))
     Zpol(i,:) = smooth(Zpol(i,:));
     RHO(i,:) = smooth(RHO(i,:));
end
%
% for plotting the data the cylindrical cooordinates have to be back
% converted into cartesian coordinates
for i = 1:length(RHO)
    [X(i,:),Y(i,:),Z(i,:)] = pol2cart(THETA(i,:),RHO(i,:),Zpol(i,:));
end
%
% convert into an STL file format
createSTL('boss_stl',X, Y, Z);
%
% Plot the results
fig1 = figure;
gr = 0.8;
S = surf( X, Y, Z, 'EdgeColor', 'none', ...
    'FaceColor', [gr, gr, gr],'FaceLighting', 'phong' );
material dull;
camlight headlight;
```

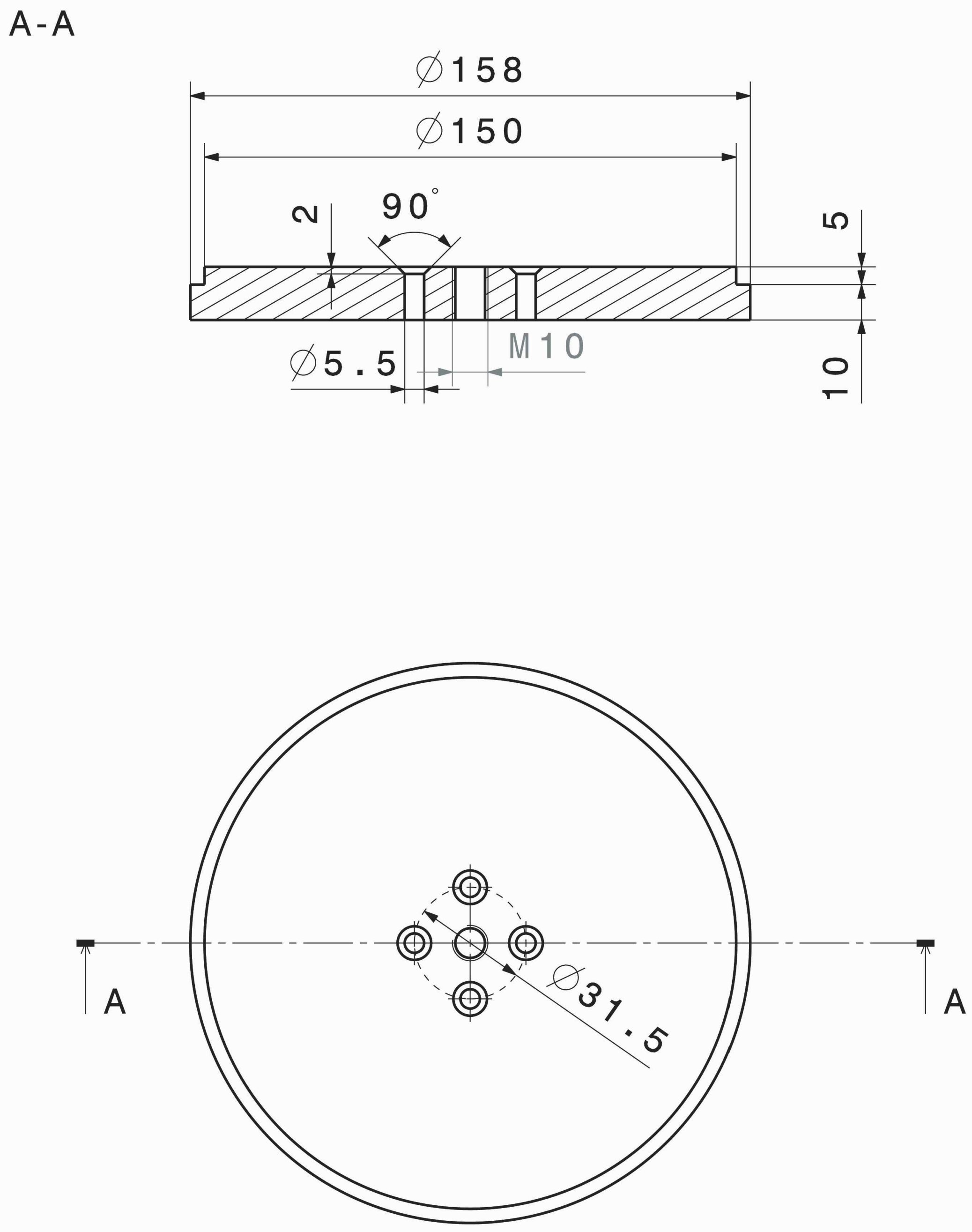
```
axis equal;
axis off;
```

Listing A.5: Routine to create STL files from the scanned data.

```
function createSTL('boss_stl',X, Y, Z);
% This method creates a stl data file out of the catresian coorindates
% of a surface representation of a recontructed object.
% It is trimmed down program from Bill McDonald's SURF2STL.
%
% References : Bill McDonald SURF2STL
%
% Author :  Bernhard Mörtl
% Date :    19 September 2010
% Version : 1.0
%
% (c) 2010, Bernhard Mörtl,
% Instutite for Automation, University of Leoben, Leoben, Austria
% email: automation@unileoben.ac.at, url: automation.unileoben.ac.at
%
% History:
%   Date:                Comment:
%   19 September 2010      Original Version 1.0
%==========================================================================
 fid = fopen(filename,'w');
%
if (fid == -1)
    error( sprintf('Unable to write to %s',filename) );
end
%
title_str = sprintf('Created',datestr(now));
%
fprintf(fid,'solid %s\r\n',title_str);
%
nfacets = 0;
%
% concatenate the vertices of the triangles
for i=1:(size(z,1)-1)
    for j=1:(size(z,2)-1)

        p1 = [x(i,j)     y(i,j)     z(i,j)];
        p2 = [x(i,j+1)   y(i,j+1)   z(i,j+1)];
        p3 = [x(i+1,j+1) y(i+1,j+1) z(i+1,j+1)];
        val = local_write_facet(fid,p1,p2,p3,mode);
        nfacets = nfacets + val;

        p1 = [x(i+1,j+1) y(i+1,j+1) z(i+1,j+1)];
        p2 = [x(i+1,j)   y(i+1,j)   z(i+1,j)];
        p3 = [x(i,j)     y(i,j)     z(i,j)];
        val = local_write_facet(fid,p1,p2,p3,mode);
        nfacets = nfacets + val;

    end
end
% Finishing writing the file
fprintf(fid,'endsolid %s\r\n',title_str);
fclose(fid);
%
%function to write one facet to the file
function num = local_write_facet(fid,p1,p2,p3,mode)
if any( isnan(p1) | isnan(p2) | isnan(p3) )
    num = 0;
    return;
else
    num = 1;
    n = local_find_normal(p1,p2,p3);

    fprintf(fid,'facet normal %.7E %.7E %.7E\r\n', n(1),n(2),n(3) );
    fprintf(fid,'outer loop\r\n');
    fprintf(fid,'vertex %.7E %.7E %.7E\r\n', p1);
    fprintf(fid,'vertex %.7E %.7E %.7E\r\n', p2);
    fprintf(fid,'vertex %.7E %.7E %.7E\r\n', p3);
    fprintf(fid,'endloop\r\n');
    fprintf(fid,'endfacet\r\n');
end
%
% compute the normalized normal vector of the triangles
function n = local_find_normal(p1,p2,p3)
v1 = p2-p1;
v2 = p3-p1;
v3 = cross(v1,v2);
n = v3 ./ sqrt(sum(v3.*v3));
```

Appendix B

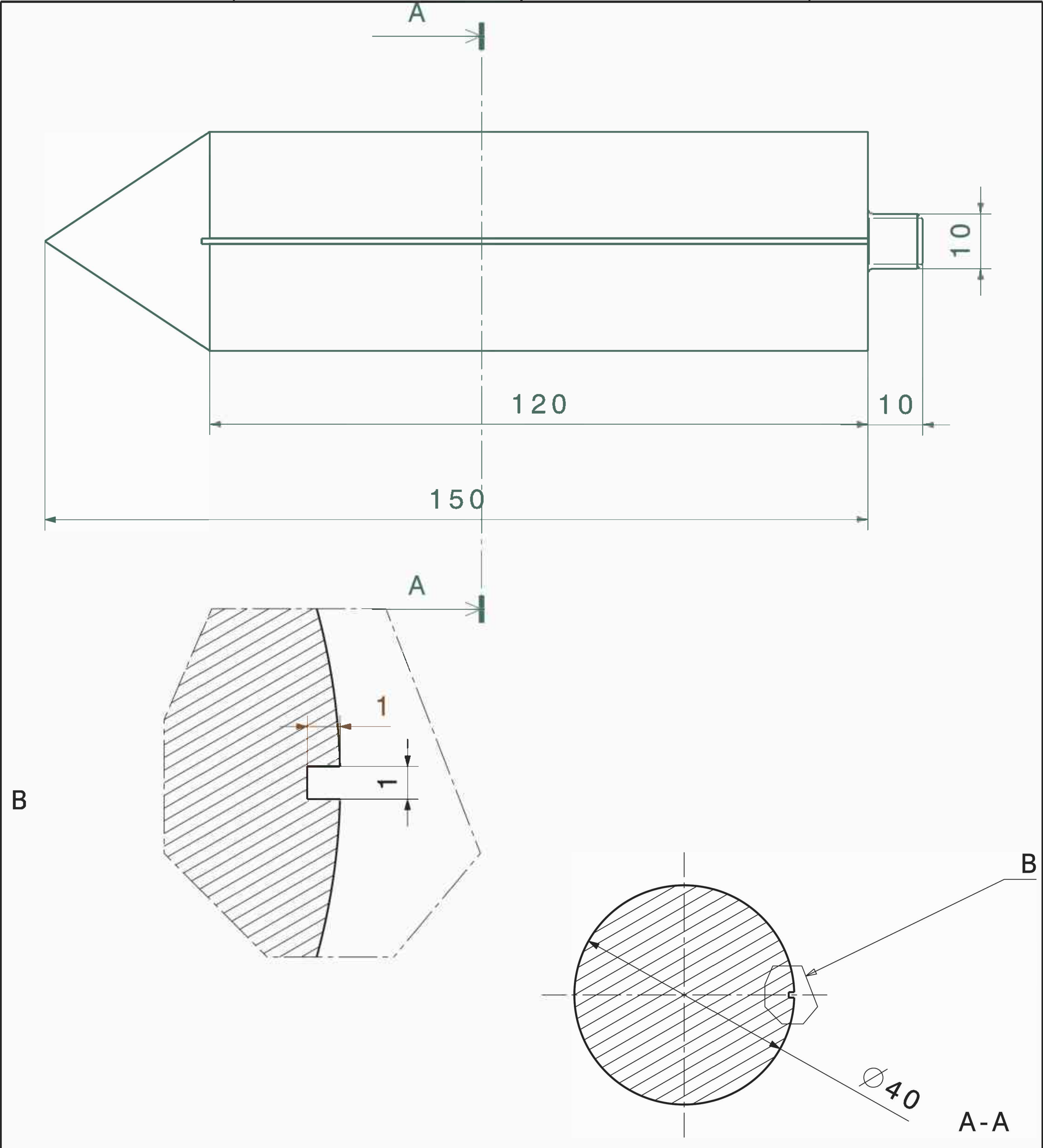
Calibration Targets



DESIGNED BY: Bernhard Mörtl	
DATE: 06.07.2010	Drehteller
CHECKED BY:	
DATE:	
SIZE A4	Lehrstuhl für Automation
SCALE 1:2	WEIGHT (kg)
DRAWING NUMBER 1	SHEET 1/1

I	_
H	_
G	_
F	_
E	_
D	_
C	_
B	_
A	_

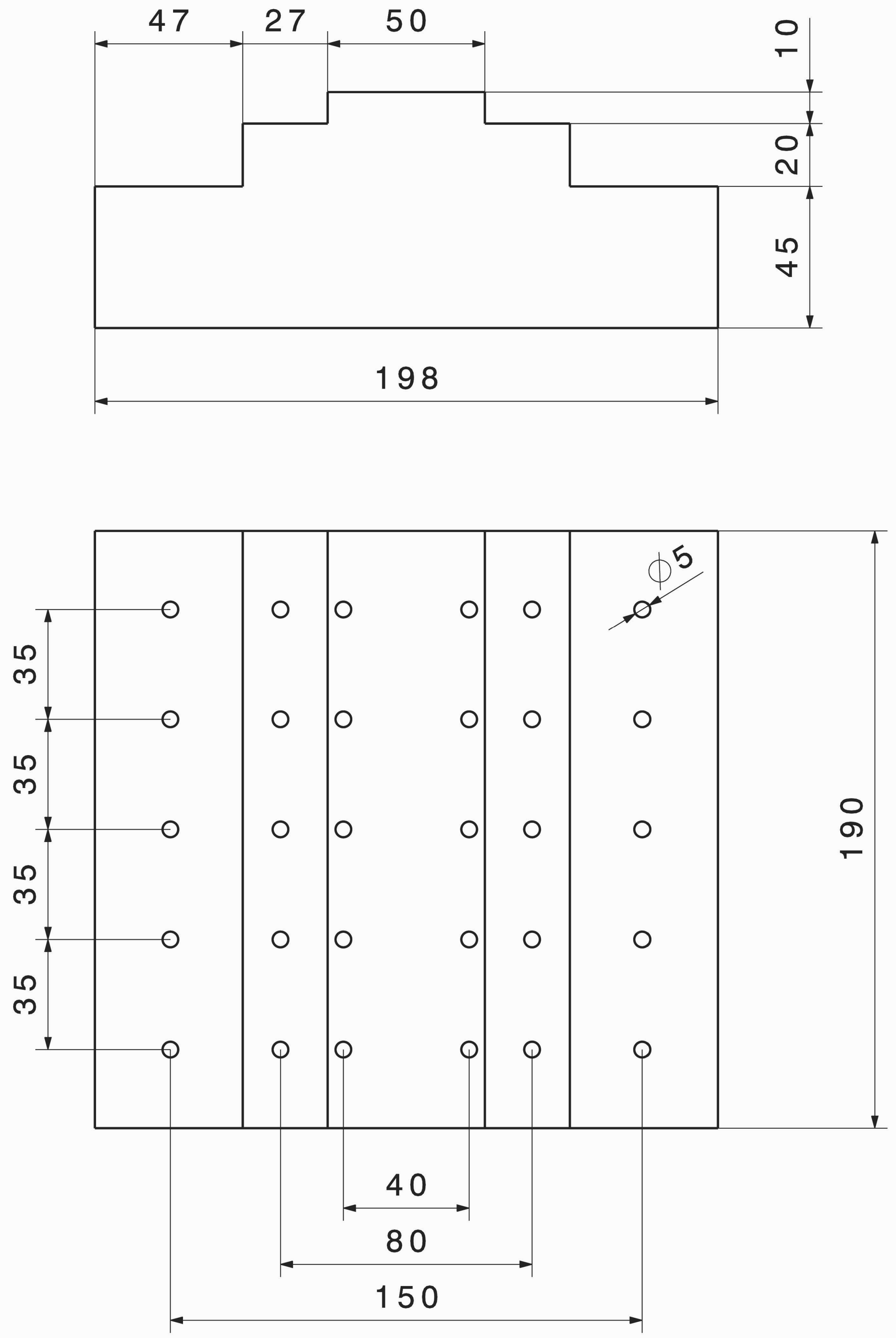
This drawing is our property; it can't be reproduced or communicated without our written agreement.



DESIGNED BY: Bernhard Mörtl DATE: 06.07.2010 CHECKED BY: DATE:	Kalibrationsspitze		
SIZE A4		Lehrstuhl für Automation	
SCALE 1:1	WEIGHT (kg)	DRAWING NUMBER	SHEET 1/1

I	_
H	_
G	_
F	_
E	_
D	_
C	_
B	_
A	_

This drawing is our property; it can't be reproduced or communicated without our written agreement.



DESIGNED BY: Bernhard Mörtl DATE: 09.08.2010	Kalibrationstarget	I _
CHECKED BY: DATE:		H _
		G _
		F _
SIZE A4	Lehrstuhl für Automation	E _
		D _
SCALE 1:2	WEIGHT (kg)	DRAWING NUMBER	SHEET 1/1	C _
		B _
		A _

This drawing is our property; it can't be reproduced or communicated without our written agreement.