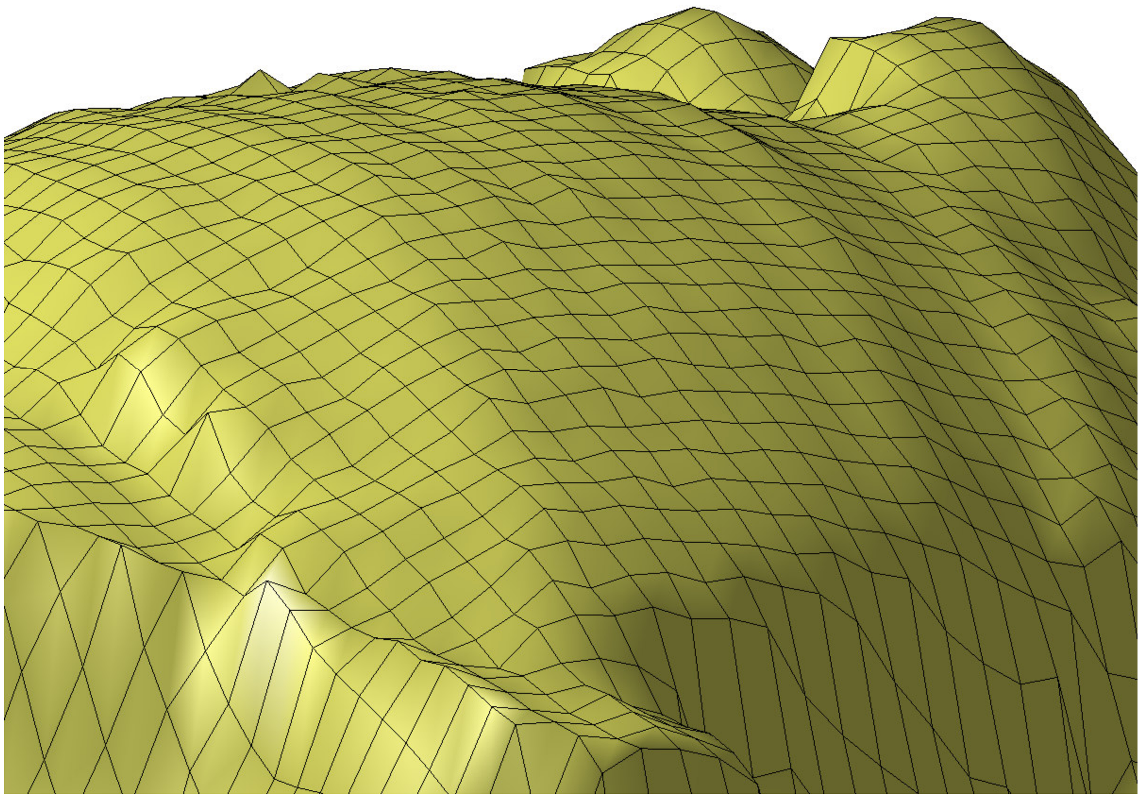
3D scanning using multibeam laser



Kenn Tornslev

Lyngby 2005
Master Thesis IMM-Thesis-2005-83
IMM

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-THESIS: ISSN 1601-233X

Abstract

In this project a complete 3D laser scanning system which uses a multibeam laser and a single camera is presented. The system is build from scratch and is aimed at small matchbox size object. The geometry of the scanner is adapted to fit an existing system which produces multispectral images of varies object and surface samples. The system as a whole can however easily be scaled to accommodate larger object and can therefore also be used for other applications.
The physical movement of the laser and camera is limited by certain constraints imposed by the existing system and this motivates the use of the multibeam laser. The introduction of multiple laser lines within the same image creates some challenges for the analysis software. Two different approached to deal with there issues are presented and compared.
The multibeam laser is incapable of drawing straight lines, save for the center line, and therefore a more advanced model describing the "laser surfaces" is applied and optimized.

Keywords:
Image enhancement, image segmentation, line detection, camera calibration, 3D reconstruction, data optimization.

Resumé

Dette projekt præsenterer et komplet 3D scanner system som bruger en flerstråle laser samt et enkelt kamera. Systemet er bygget fra bunden og er anvendeligt for små objekter i tændstikæske størrelsesorden. Scannerens geometri er tilpasset et eksisterende system der laver fler-spektrale billeder af diverse objekter og overflader. Systemet som helhed kan let skaleres så større objekter kan scannes og kan derfor også bruges i en anden anvendelse.
Kameraet og laserens bevægelighed i forhold til objektet er underlagt visse begrænsninger for at opnå kompatibilitet med det eksisterende system og dette er bevæggrunden for brugen af flerstråle laseren. Billeder med flere laserlinier giver større udfordringer for analyse softwaren. To forskellige tilgange til dette problem præsenteres og sammenlignes.
Flerstråle laseren er bortset fra den midterste linie ikke i stand til at tegne rette linier, derfor anvendes en mere avanceret model til at beskrive "laser-overfladerne".

Nøgleord:
Billedforbedring, billedsegmentering, linie detektering, kamera kalibrering, 3D rekonstruktion, data optimering.

Preface

This thesis was prepared at the Image Analysis and Computer Graphics section of the Department of Informatics and Mathematical Modeling, at The Technical University of Denmark, DTU in partial fulfillment of the requirements for the degree Master of Science in Engineering, M.Sc.Eng.

The algorithms developed during this project have primarily been written in Matlab 7. The source code is available on the enclosed DVD along with the image sequences constituting the datasets and calibration data. If you have obtained a copy with no enclosed DVD, please go to: *www.tornslev.dk/kenn/thesis* and look for available downloads. At the time of writing only the source code can be downloaded due to lack of space on the server.

Lyngby, Oktober 2005

Kenn Tornslev

Acknowledgements

I would like to thank my project supervisor Jens Michael Carstensen and co-supervisor Henrik Aanæs for help and inspiration during this project. Also I would like thank Kenneth H Møller for insightful discussion and ideas during the course of this project. Finally I would like to thank my family who have put up with seeing far less of me than usual.

Contents

1 Introduction

Everyday we humans interact with all sorts of objects without the need to pay special attention to it. This is possible due to our ability to perceive the geometry of the world around us. Getting a computer to perceive the geometry of an object is however a more complicated matter. Nevertheless there exist several systems where computers via a digital camera can obtain geometrical information about an object. The perhaps most accurate of these is a 3D laser scanner. While being around for some years now 3D laser scanners are still a relatively young technology in rapid progress. As 3D laser scanners continue to evolve and the cost of components decrease this popular technology finds it way into a still increasing number of new applications. The ability to create a digital 3D model of an object in a few minutes, which hereafter can be transferred worldwide via internet almost instantly, makes it an attractive tool for a lot of applications.

In this thesis a complete 3D laser scanner system is presented. The scanner is build using a laser and a single camera and is different from conventional scanner because it uses a multibeam laser that generates a fan of lines.

2 Motivation and objectives

The vast majority of existing laser scanners uses a single laser in combination with one or more cameras, where the laser draws a single straight line. In this project the idea of using a multibeam laser, which draws a fan of lines, is explored. The motivation for using this type of laser is that the required movement of the laser and camera relative to the object been scanned can be drastically reduced. This opens up a new spectrum of possibilities for incorporating the laser scanner into other systems where the space or possible movement is limited.

2.1 Objectives

The objective of this project is twofold. Firstly is the construction of a laser scanner system including every thing from assembling the hardware to writhing the necessary algorithms to analyze and process the acquired data. Secondly is the use of the multibeam laser. While solving a crucial issue related to the physical setup, namely reducing the required motion of the laser and camera relative to the object, the introduction of multiple laser lines within the same image creates some very hard challenges for the software part. The aim is to construct an algorithm capable of handling these challenges and to examine the advantages and limitations of this technique.

2.2 Thesis overview

The process scanning an object involves the following steps: A sequence of images where the laser sweeps across the object must be acquired. Next the images must be analyzed – the projected laser lines must be detected, extracted and represented mathematically. Hereafter the relationship or correspondence between the detected line pieces must be sorted out. Finally the obtained 2D projections of the laser lines must be converted to a series of 3D points using a mathematical model of the emitted laser lines.

- Section 3: Introduction to Laser Scanner Systems – explains how laser scanners work in general. Also an overview of different possible geometrical configurations of the scanner is given. This is followed by a brief overview of existing scanner systems.
- Section 4: Assembling the Scanner – describes how the laser was assembled and explains different issues related to geometrical setup of the scanner.
- Section 5: Line Detection – describes the process of detecting, extracting and representing the laser lines projected onto the object being scanned.

- Section 6: Line Segment Correspondence – explains the process of sorting out the correspondence of line segments originating from different lines. Two approaches to this problem are proposed and compaired.
- Section 7: 3D Reconstruction – describes how the 2D projections of the intersection curves between object and laser are converted into 3D points. Also the calibration of the camera and laser are explained. Finally the resulting 3D reconstructions of the scanned objects are presented.
- Section 8: Future Work – presents ideas for future work and improvements to the scanner.
- Section 9: Conclusion – gives an overall conclusion on the project.

3 Introduction to Laser Scanner Systems

In general, the purpose of a laser scanner system is to create a digital 3D model of some surface or object. Normally a laser that emits a single flat sheet-of-light is used meaning that if you point the laser at a wall it draws a straight line. The sheet or plane itself is not visible, only the intersection between the sheet and whatever is put in front of the laser is visible. The intersection between a plane and some object is always some sort of line or curve in space. Given that the object in question is solid this curve will lie on its surface and thus obtaining this curve we obtain some 3D information about the surface. Consequently a laser scanner system extracts 3D information about a surface by performing a 3D reconstruction of the intersection between surface and the shet-of-light emitted by the laser. Performing this 3D reconstruction involves the following few steps. First an image of the intersection is acquired by a digital camera. Figure 3.1 shows an image of such an intersection. Next the intersection is extracted from the image by the use of digital image analysis techniques. The intersection curve would typically be represented by a number of discrete points in 2D image coordinates.

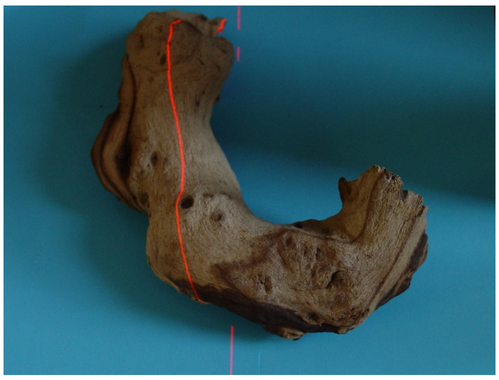


Figure 3.1: Picture of an object and the laser plane projected onto this object yielding the red intersection curve seen on the object.

The final step is to convert these points from 2D to 3D. This is achieved by calculating a line of sight from the camera and into the physical world corresponding to each of the points. Hereafter the intersection between this line of sight and the laser plane is calculated. This constitutes the 3D reconstruction the object and laser intersection. However one single intersection only gives us a thin slice of the objects surface, so consequently the intersection must be moved and the process repeated until there are enough slices to compose a good representation of the object. The following section will give a more thorough outline of the steps involved including setting up and calibrating the system.

3.1 Analysis of the acquired image

The acquired digital image contains the intersection curve projected onto the 2D image plane of the camera. Assuming that the laser is the only significant light source the intersection curve will consist of pixels which are brighter than background. The purpose of the image analysis is to produce a mathematical representation of the curve based on the pixel values in the digital image. This mathematical representation may just be a series of discrete point coordinates but the curve could also be represented by a spline curve or something similar. If the signal to noise ratio of the curve is strong, deciding which pixels belong to the curve can be done simply by setting a threshold value. Otherwise some sort of enhancement technique must be performed in order to increase the pixel values on the curve. Ideally the intersection curve is infinitely thin but in reality it may be as wide as a millimeter or so. This raises another aspect of the image analysis part – namely not only finding the curve within the image but also finding the middle of the curve. How to approach this will be described later.

3.2 Converting 2D points to 3D points

The final objective of a laser scanning system is to obtain 3D information or in other words simply spatial positions in the physical world. Therefore a coordinate system in which these coordinates can be represented is needed. A natural choice is to use the camera as a reference for this coordinate system. Here it's assumed that the camera is modeled by the pinhole camera model, that the camera's center of projection is the center of the coordinate system and that the z-axis coincide with the optical axis of the camera. Having defined this coordinate system and assuming that the camera's focal length is known we can now convert any point in the image to a direction or line of sight that goes from the image plane through the projection center and out into the physical world. Figure 3.2 shows an illustration of this.

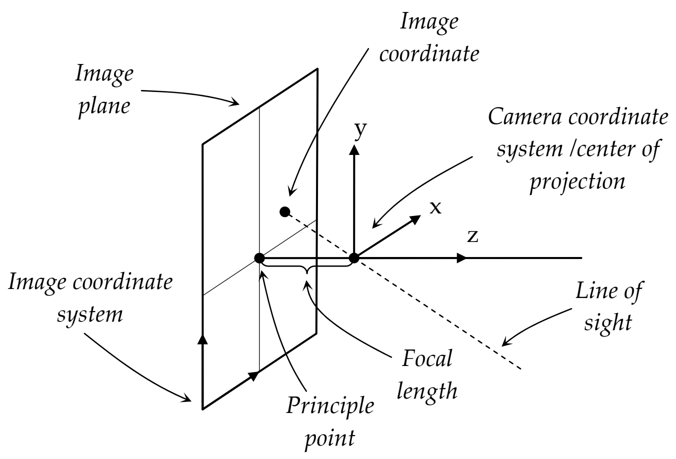


Figure 3.2: Illustration of the relationship between the image coordinate system within the image plane and the camera coordinate system. Also notice how an image coordinate corresponds to at line of sight into the physical world.

Once we have the line of sight corresponding to an image coordinate, this coordinate can be converted into a 3D point by intersecting the line of sight with the sheet-of-light emitted by the laser. This does however require that we have a mathematical description of this surface given in the camera coordinate system.

3.2.1 Calibrating the camera

In order to obtain the focal length a camera calibration must be performed. However a camera calibration provides much more information than the focal length alone. It also gives an estimate of the principal point, which is the image coordinates of the point where the optical axis intersects the image plane. Finally the camera calibration provides information about the radial and possibly (depending on the calibration) the tangential distortion of the lens. This information essentially describes how the camera and lens differs from the pinhole model. Utilizing this information to correct for distortions introduced by lens yields a much more precise reconstruction of the 3D point than else would have been possible.

3.2.2 Obtaining a mathematical description of the laser plane

In order to calculate the intersection point between the line of sight and the laser plane a mathematical description of this plane is required. One way to go about this is to acquire a series of images where a flat object is placed orthogonal to the optical axis and at a series of knows distances to the camera.

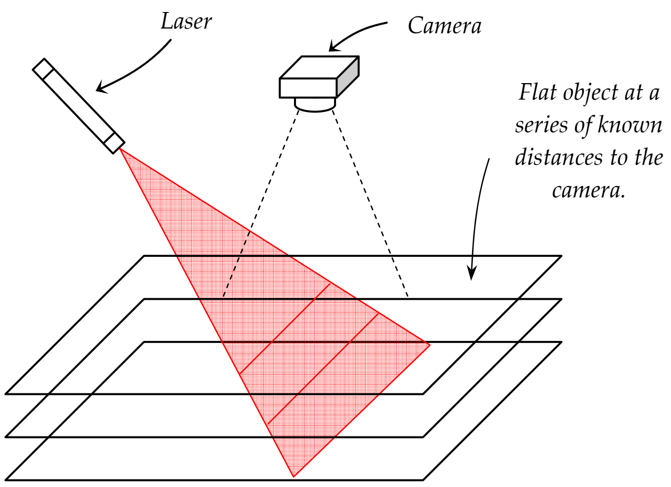


Figure 3.3: Obtaining intersection curves at a known distance to the camera in order to get at mathematical description of the laser plane.

At each distance we obtain the intersection curve in 2D, but since we know at which distance each intersection occurred the 2D intersection curves can be converted to 3D curves directly. Using these curves the angle and position of the laser plane can be estimated, thus yielding the mathematical representation we sought.

3.2.3 Moving the intersection between object and light

A single intersection between the object and the laser plane only gives us 3D information about a thin slice of the object's surface. If we want more information, the intersection must be moved around on the surface. This requires moving either the laser or the object. Figure 3.4 show four examples of this. Either one can be preferable depending on the given application. In a production line where the objects moves on a conveyer belt then setup A is an obvious choice. If the object in question is big or heavy and therefore impractical to move then setup B or C will be preferable. Setup D is ideal for small portable scanners. It is also possible to combine two or more movements in the same setup. The motivation for doing this is to increase the coverage (this phenomenon is described in the following section).

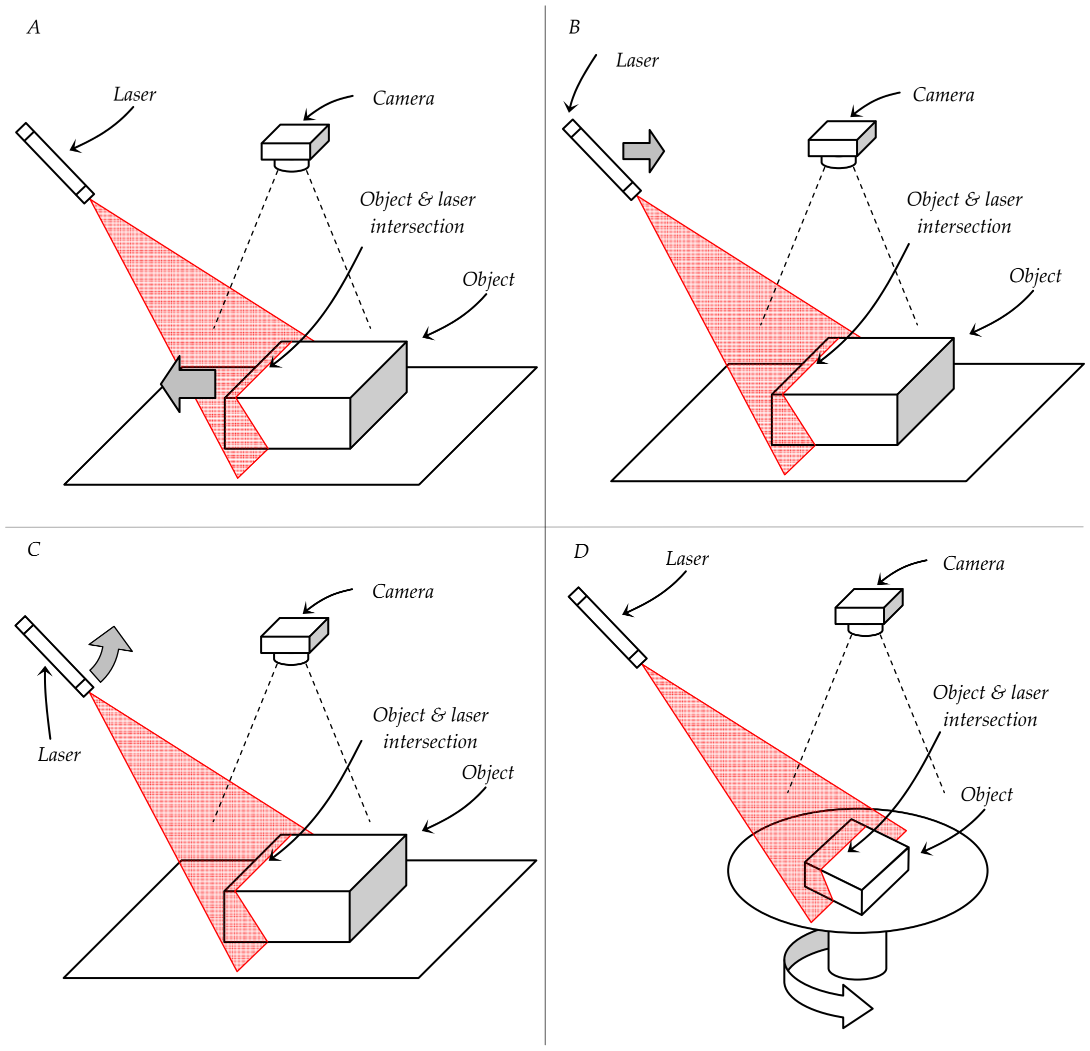

A
Laser
Camera
Object & laser intersection
Object
B
Laser
Camera
Object & laser intersection
Object
C
Camera
Laser
Object & laser intersection
Object
D
Laser
Camera
Object & laser intersection
Object

Figure 3.4: Four different scanning setups – the camera is stationary in all cases. (a) The laser is stationary and the objects moves to the left making the object and laser intersection move across the object to the right. (b) The object is stationary and the laser moves to the right again making the object and laser intersection move across the object to the right. (c) The object is stationary and the laser rotates making the object and laser intersection sweep across the object. This has a similar effect as in the two previous examples but the angle between the optical axis of the camera and the laser varies as the laser rotates. (d) The object rotates while the laser is stationary. This makes the intersection curve rotate on the objects surface.

3.2.4 Shadowing

In order for the laser triangulation to work the camera and laser must be reasonably spaced and these therefore see the object from each there point of view. For objects with some vertical variations some points on the object that are visible from the cameras point of view might not from the lasers point of view or vice versa.

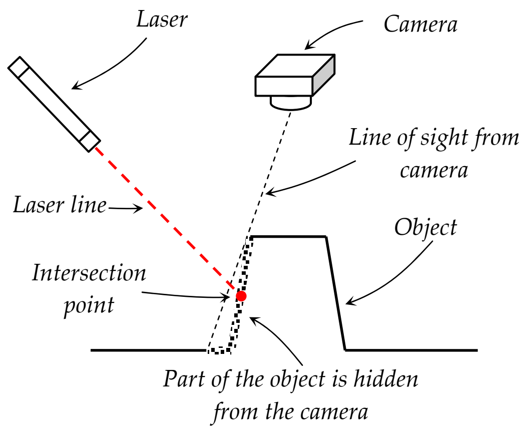

Laser
Camera
Line of sight from camera
Laser line
Object
Intersection point
Part of the object is hidden from the camera

Figure 3.5: Illustration of a situation where a part of the object is hidden from the laser. While the laser actually sweeps across this area no data is obtained as the object a laser intersection is not visible from the cameras point of view.

Figure 3.5 shows an illustration of a situation where a part of the object is hidden from the camera. Even though the laser sweeps over this part of the object the camera never sees this and thus no data from this area is recorded.

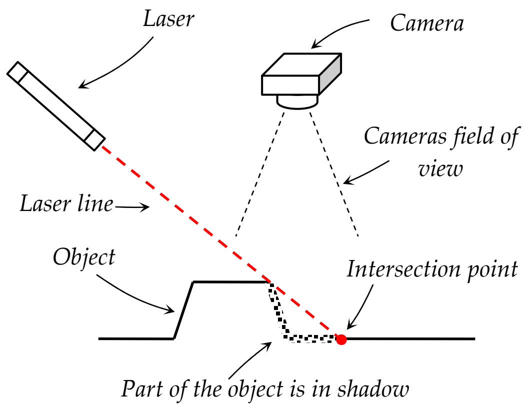


Figure 3.6: Illustration of a situation where part of the object lie in shadow relative to the lasers point of view. While this part of the object is fully visible from the cameras point of view no data is obtained as the intersection curve skips this part as the object is scanned.

Figure 3.6 shows an illustration of another situation where a part of the object lies in shadow relative to the lasers point of view. Both situations result in the same thing: a part of object is not scanned. Different objects cause different

amount of shadowing and consequently the coverage of the acquired data may differ from object to object. Some laser scanner systems move the intersection curve on the object by both rotating and translating the object. Using a combination of rotation and translation it is possible increase the coverage and possibly to get 100% coverage. However it always possible to find objects that can cause problem ever using this technique. Other systems simply use two or more cameras in an arrangement where the laser is pointing orthogonal to the object. This approach also eliminates the problem for a great number of objects but very concave shaped objects can still cause problems.
The setup used in this project uses a single camera only and there is no possibility to rotate the object. Therefore it's not possible to obtain data from parts of the object that lie in shadow. As a consequence this system is best suited for objects without large vertical variations.

3.3 Existing 3D laser scanner systems

3D laser scanning systems have been around for some years now. Specialized systems for a large variety of applications are available.
In assembly lines 3D scanners can perform quality control by verifying measures and tolerances.
In the construction industry 3D laser scanners are used to check that large construction parts are placed correctly.
In the movie industry, handheld 3D scanners are used to get sculptured creatures of different kind into the computer.

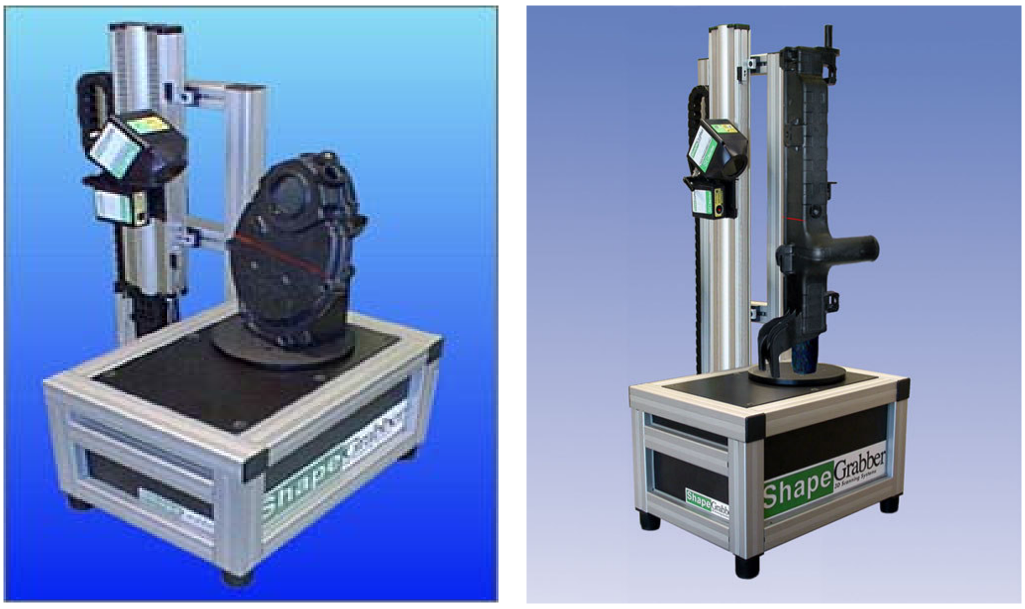


Figure 3.7: ShapeGrapper AI 300 (left) and AI600 (right) systems from ShapeGrapper, Canada.

Figure 3.7 shows two 3D scanner systems from ShapeGrapper in Canada. Objects are placed on a rotary table and the laser and camera can move on the vertical rail. Thus, this system combines rotation of the object with translation of the laser and camera. The pictures are borrowed from [27].

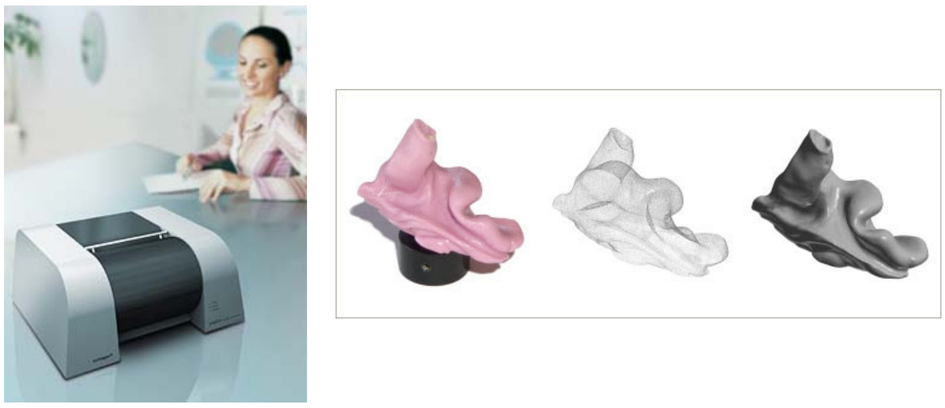


Figure 3.8: (left) Legato scanner from 3Shape. (right) An ear impression (physical object), the digital model shown as a wireframe and finally the shaded digital model.

Figure 3.8 shows the Legato 3D scanner from 3Shape and an ear impression scanned by this scanner. The digital model can be used to build customized hearing aids. The illustration are borrowed from [29].

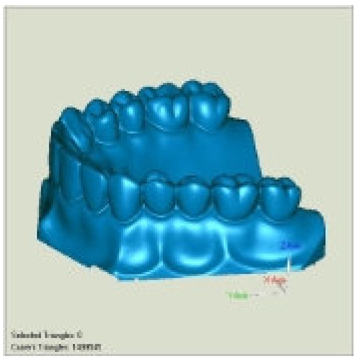


Figure 3.9: Digital model of a set of teeth created by a 3D laser scanner from Laser Design, Minneapolis, USA.

Figure 3.9 shows a digital model of a set of teeth created by a laser scanner from Laser Design. 3D digital models of teeth are used in the dental industry when building crowns, caps etc. The image is borrowed from [28].

3.4 Multi beam laser system

All the presented system has one thing in common – they all use a laser that emits a single sheet-of-light. One of the main purposes of this project is to explore the possibilities of using a laser that emits multiple sheets-of-light at slightly different angles. Figure 3.10 shows an illustration of this.

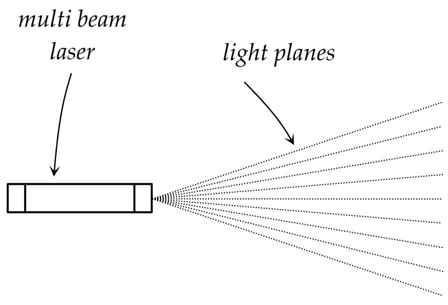


Figure 3.10: Illustration of a multi beam laser and the emitted sheets-of-light.

The motivation for using a multi beam laser is the ability to cover a greater area faster without moving the laser or object very far. Also the amount of data per image increases as there are multiple intersections within the same image. Figure 3.11 shows how a multi beam laser is capable of covering an area moving only very little compared to a single beam laser.

This feature gives a great advantage in applications where the movement of the laser or object is limited. Also since the required physical movement of the setup is reduced so is the time necessary to perform this movement.

However the use of a multi beam laser is not all advantageous, it also introduces some problematic issues. Although the intersection curves are continues in a mathematical sense the may not be in practice. This is because the object can cast a shadow on it self. Even though the intersection curve is continues on the object it may not be from the camera's point of view. As a consequence the 2D projection of the intersection curve captured by the camera may consist of a number of curve segments rather than one continues curve. The fact that there is multiple intersection curves within the same image poses the risk of mixing up these curve segments. One of the big challenges in this project is to handle these curve segments and figure out which constitute a given line.

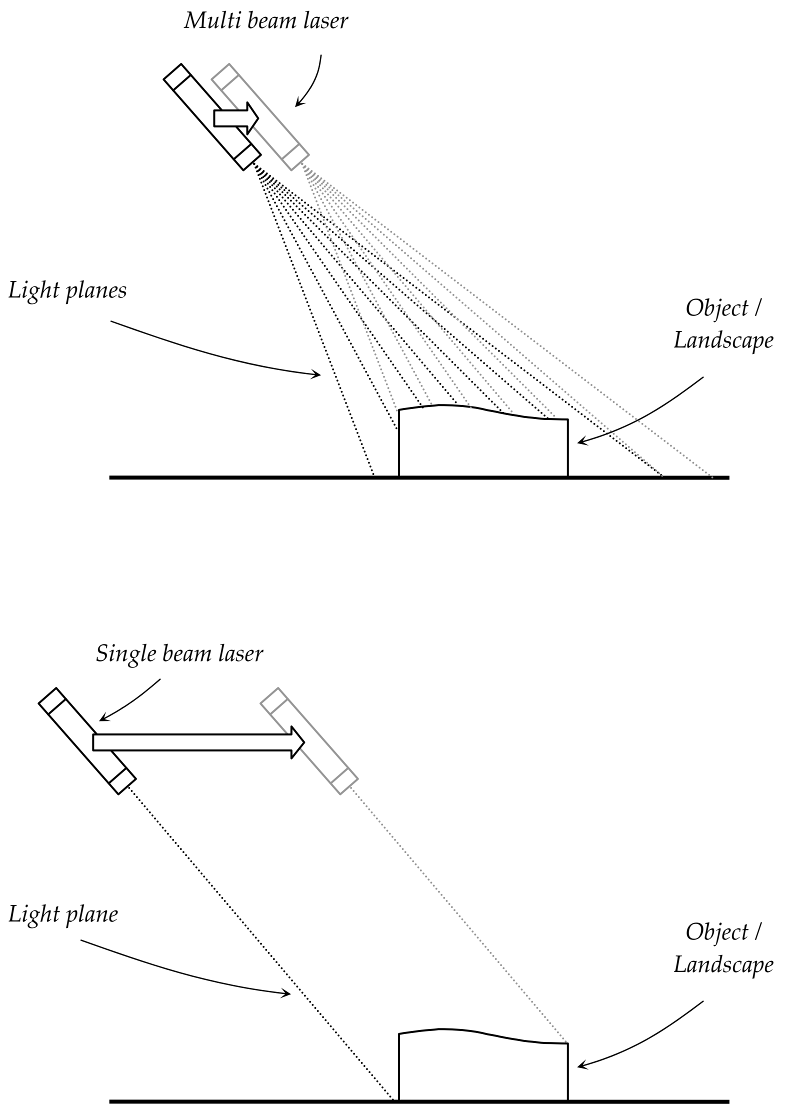


Figure 3.11: Illustration of the laser coverage. A multi beam laser (above) only need to move very little compared to a single beam laser (below) in order to cover a given area.

4 Assembling the Scanner

The overall goal of this project is to create a system capable of scanning an object or small landscape using a camera and a laser. This problem is basically a classical triangulation problem – the camera sees a line drawn by the laser. The physical position of a point on the line can be found by intersecting the laser line and the cameras line of sight to that particular point.

However the aim here is to investigate this problem with a special setup in mind, in order to integrate the scanning part with an existing system that produces multispectral images of varies objects and surfaces. Therefore this system is bound to the geometry of the existing system. The laser and camera are both mounted on a horizontal bar, the camera pointing directly downwards and the laser pointing on the area seen by the camera. The type of laser used produces 15 individual lines a few degrees apart.

As the bar is moved downwards the laser lines sweeps across the object.

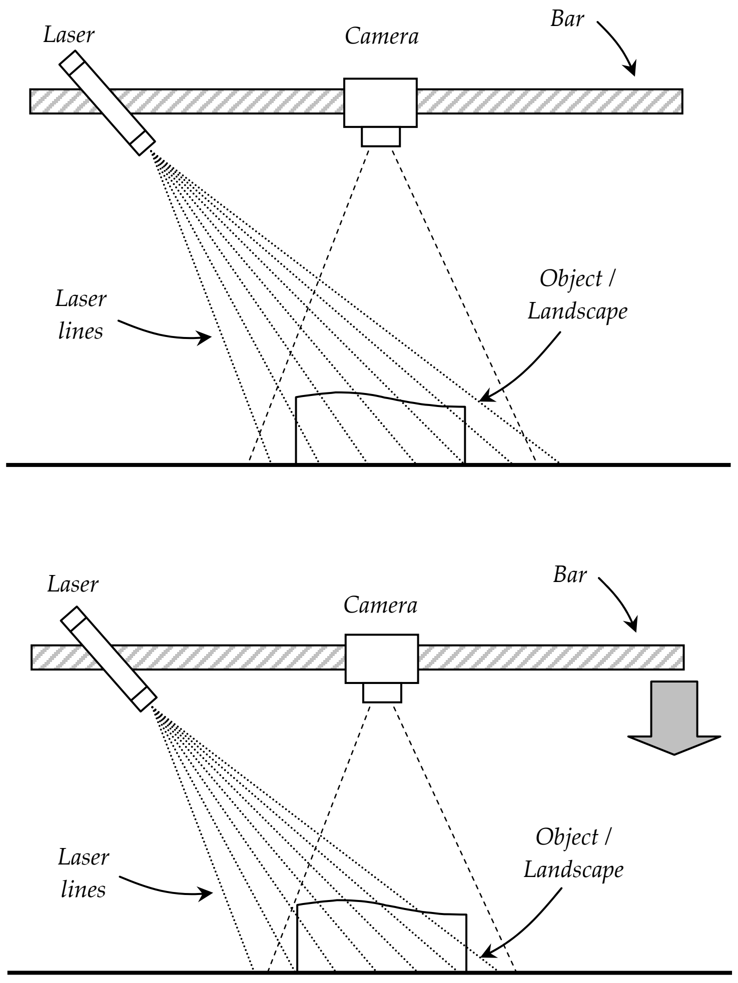


Figure 4.1: Illustration of the basic setup. Both the laser and the camera are mounted on a horizontal bar. The camera points directly down while the laser is mounted at an angle pointing on the area seen by the camera. As the bar is moved vertically the intersection curves moves sideways across the object.

In this way the intersection curves between the object and the laser planes covers the entire object, which is therefore scanned by the laser.

4.1 Choosing the setup parameters

The above illustration gives a general idea of the setup but there are a lot of specific details to consider which put different constraints on the setup. For the intended purpose the camera and laser will be hanging approximately half a meter above the ground. This is therefore one constraint. At this height the lens is adjusted to focus on the ground. For this purpose a text document was laid beneath the camera. Hereafter experiments with the camera was carried out to see how much it could be moved from its initial height and still keep the text in focus. It turned out that turning the aperture of the lens to the smallest possible setting, meaning allowing the smallest possible amount of light which maximizes the focus range, yield a working range of 10 centimeters centered on the initial height. Hence this is another constraint – namely the vertical working range of the bar. Fortunately this is consistent with the geometry of the existing system. This leaves us with choosing a position for the laser. Again the setup dictates that the laser is placed at approximately the same height as the camera. However the horizontal distance to the camera as well as the angle of the laser can be more freely chosen. The starting point for choosing these settings is the camera and its vertical working range. Recalling that the laser is mounted on the same bar as the camera we know that the laser is bound to the same vertical movement. The angle that we chose for the laser will govern how much intersection curves will move as the bar is moved from the lowest position to the highest or vice versa. Actually each intersection curve will travel at slightly different speeds as bar moves since they all have slightly different angles of incident. So to ensure total coverage we must arrange the laser such that the slowest traveling intersection curve will reach the starting position of the second slowest intersection within the limitations of the working range. The slowest traveling intersection curve is the one with the steepest incident angle, hence the left most line in the above illustration. A certain amount of overlap is however desirable and therefore it has been chosen that the lines should overlap three times – meaning that e.g. line one end at the position where line four starts.

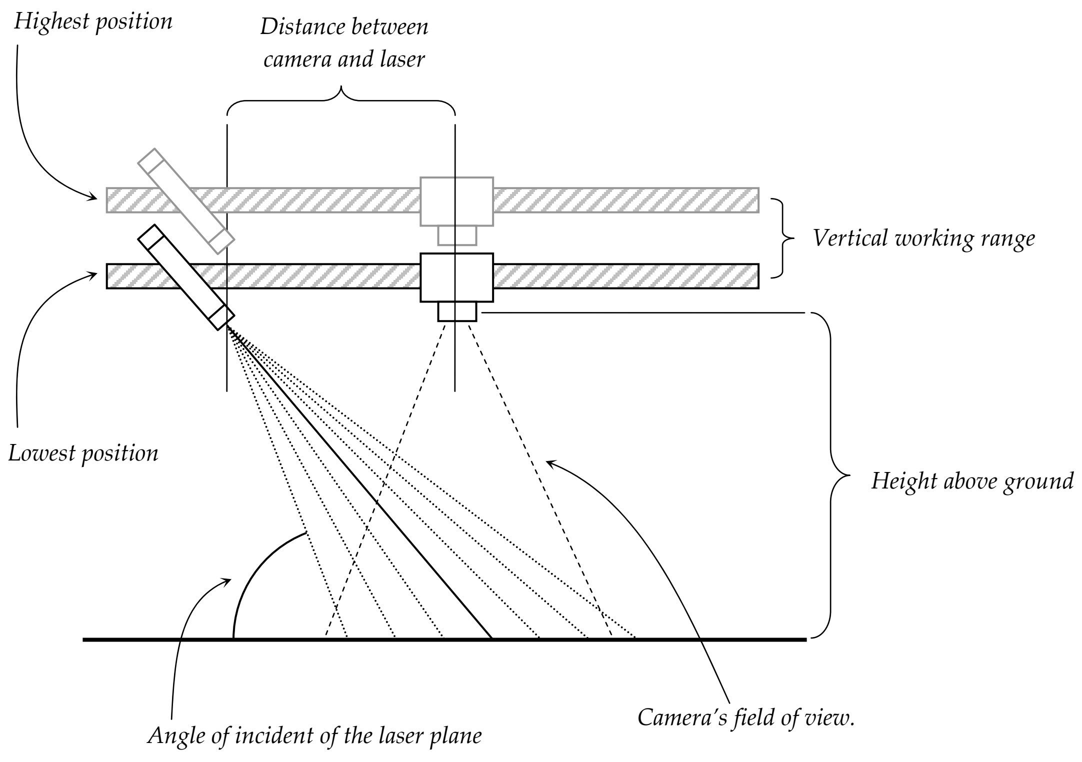


Figure 4.2: The specific camera, lens and laser set a number of constrains on the setup. These include the vertical working range, the height above ground, the distance between the camera and laser and the incident angle of the laser planes.

Figure 4.2 illustrates the parameters of the setup which must be carefully chosen in order to ensure that the system works properly. Given this setup it's not possible to utilize all the 15 laser planes emitted by the laser. At its highest position the camera is only able to see six of the intersection curves. As the bar is moved some of the lines move out of the camera's field of view and other move in. Within its working range the camera see a total of eight intersection lines. Table 4.1 shows the chosen values of the setup parameters.

Setup parameter	Chosen value
Height above ground (lowest position)	350 mm
Vertical working range	100 mm
Distance between laser and camera	200 mm
Incident angle of the centre laser plane	85 degrees

Table 4.1: The chosen value of the setup parameters (approximate values).

4.2 Assembling the setup

Figure 4.3 shows four photographs of the assembled setup. The laser and camera are mounted on the horizontal bar which in turn is mounted on a reprostand. The elevation of the horizontal bar is controlled by the long vertical threaded pipe section (the thin white metal rod next to the black tower of the reprostand) which also supports the weight of the bar. In order to move the bar vertically the threaded rod must be turned manually. Thus recording a sequence of a hundred frames can be a long tedious process. The award for this hard work is that the distance traveled by the bar between consecutive frames is very uniform.

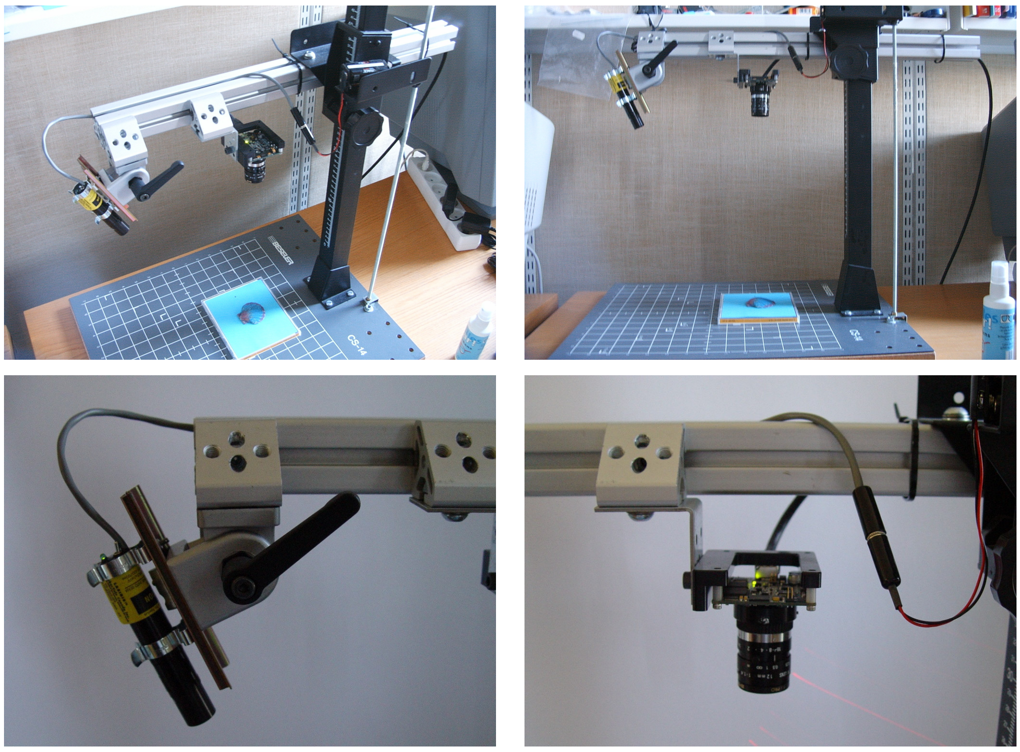


Figure 4.3: (top row) The assembled setup seen from two different angles. A shell is placed on a cd-cover beneath the camera. (bottom row) Close view of the mounted laser and camera respectively.

The entire setup has been build from material at hand from the image lab at IMM, DTU. Finding and assembling the proper fittings and attachments was a bit of a challenge. Initially the there were no rod controlling the elevation of the bar and data were recorded by lowering the bar by hand while the camera were continuously shooting frames. However it showed impossible to control the motion of the bar steadily enough and therefore this approach was abandoned. For the threaded rod approach to work a nut had to be mounted on the sliding part of the reprostand. The first attempt to attach it with two-part adhesive only held for a few days and then the nut fell of. Finally the nut was attached by point welding and this solution has lasted so far.

5 Line Detection

5.1 Introduction

An acquired image sequence contains the intersection curve between a given object and the sheets-of-light emitted by the laser. It is this curve that is of interest and the objective is to detect and extract the 2D shape that is projected onto the CCD chip of the camera. It is points on this 2D shape that later shall be converted into 3D points. The process of extracting such a curve is known as line detection which along with the related edge detection is well described image analysis techniques. In [9] edge enhancement by the use of the Sobel and Prewitt filter masks is presented. In [12] Green presents a tutorial on the popular Canny edge detector and Christov [6] uses the Canny edge detector in connection with wavelets. In [18] a framework for multiscale edge detection is given. Konishi et al. [16] presents a method for statistical edge detection. Line detection by the well known Hough Transform is presented in [9] and Davies [10] presents a method for truncating the Hough transform parameter space. Obviously there exist numerous methods and techniques for detecting lines and edges so it's merely a matter of choosing a suitable technique for the specific problem at hand. We do have some a priori knowledge about the lines that are to be detected. The lines are horizontal with some vertical variations (possibly even discontinuities) caused the geometry of the object been scanned. For this reason a method like the Hough transform is not well suited because it requires that the line is described by a known function such as a line or a parabola. The edge finding techniques can be used but they produce two edges per line, thus the lines are detected indirectly. It is more convenient to detect the lines directly. The method presented in the following section is very similar to the Sobel and Prewitt filter techniques where a filter mask is moved around the image. The mask is designed to give a high response when a certain shape is encountered – in this case a horizontal line of a certain width. This process enhances the line enough that line and background can be separated using a threshold value. This approach has been chosen because it's simple yet very effective.

5.2 Overview

The line detection is responsible for finding and extracting the line segments from the images, hereafter storing them using a suitable mathematical representation. The chosen mathematical representation of the lines is merely a series of discrete points. The lines could also have been represented by spline curves but as there is no obvious benefit from this we're sticking with discrete point not to complicate mattes unnecessarily.

5.3 Ensuring optimal conditions for the algorithm

Even before the line detection algorithm is designed we can take steps to ensure that it's given optimal conditions. In other words we want to make it as easy as possible for the algorithm. One thing that can be done is to eliminate other light sources. Having done this the only other remaining light source is daylight. This light source can obviously not be eliminated but it can be shielded of to some extent. In order to further strengthen the signal to noise ratio of the laser an optical filter is used. This filter ideally only allow light with a wavelength equal to that of the laser to pass and block out light of any other wavelength. In practice the undesirable light can only be attenuated by the filter.

5.4 Adjusting camera settings

As mentioned the purpose of the image analysis algorithm is to detect line segments within the image. These line segments are characterized by pixel values significantly brighter than the background.

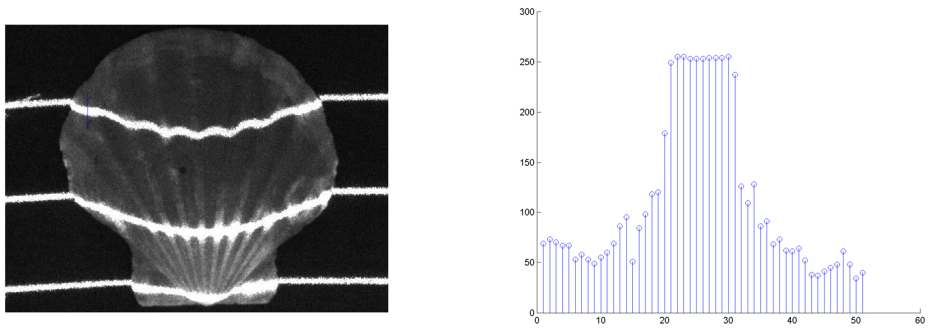


Figure 5.1: (left) Image of a shell with projected laser lines at high exposure. (right) Image data (pixel values) from a cross section of the topmost laser line.

By adjusting the exposure and brightness settings of the camera it is possible to ensure that the pixel values of the line segments are saturated or near saturation. This makes it very easy to detect the lines by setting a threshold value. However if we look at a cross section of a line, see Figure 5.1, the pixel values surrounding the line will be noisy with some DC offset overlaid. The pixel values of the line itself, which it approximately five to ten pixels wide, will be saturated. The shift between background noise and line thus happens in one jump from one pixel to the next. So looking at this cross section we can only determine if a given pixel is part of the line or not. This basically means that there is no way we can obtain the position of the line with sub pixel precision. However the actual intensity of the cross section of the laser line more or less resembles a gauss bell. By reducing the exposure and brightness settings on the camera it is possible to catch this phenomenon. The motivation for doing this is that we can try to estimate the middle of the bell with sub pixel precision. Obviously the more precise the line position can be estimated the better the final 3D reconstruction. Therefore it's only natural to pursuit any possibility of estimating the line position with sub pixel accuracy. However in doing so, we have made the line detection problem harder than before. Recall that the exposure and brightness settings were reduced in order to "see" the gauss bell. This means that the pixel values of the line overall have been significantly reduced.

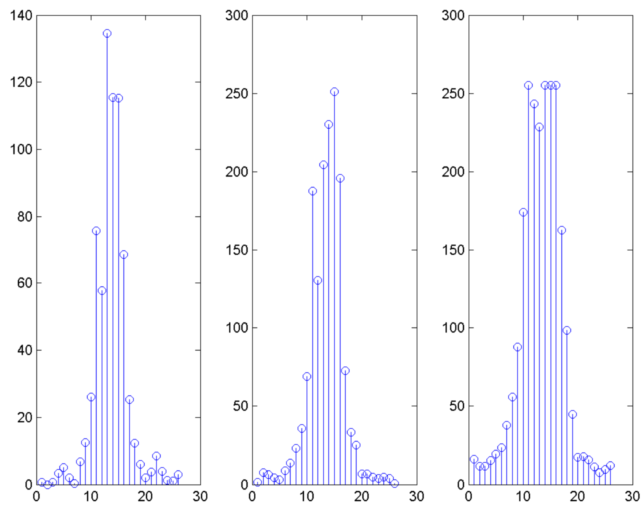


Figure 5.2: The three figures shows pixel intensities of a laser line cross section for increasing exposure values. In the latter case the pixel intensities have reached saturation at the center of the line.

In Figure 5.2 the same laser line cross section is shown for different exposures. The optimal exposure setting yields an image where the pixel values at the center of the line barely reaches saturation – the second in this case. This should make it possible to fit a model to the data and thereby estimating the center of the line.

However before such a model can be fitted a starting guess of the center value must be provided. This leads us back to the issue of detecting the line. Given the depicted cross section above it should be relatively easy to set a threshold value that detects the line. Unfortunately the cross section for the lines

throughout an entire image is not very uniform. The observed intensity of the laser is very sensitive to different kinds of surfaces. In fact if merely the color of a surface changes so does the intensity of the reflected laser line. The explanation is that the color of a surface is determined by how strongly different wavelengths of light are reflected with respect to each other. A laser is characterized by emitting light in a very narrow frequency band and is often specified at a single wavelength – 462 nanometers as an example. While one color might reflect this wavelength very strongly another color may reflect the same wavelength weakly. Thus the intensity of the reflected light may vary very much on varying colors. However the reflectance properties of a surface can not be characterized by its color alone, the smoothness of the surface also plays a great role. If a given surface is very smooth the majority of the incoming light will be reflected at an angle identical to the angle of the incoming light – only mirrored. Conversely on a rough surface the incoming light will be reflected more uniformly distributed in all directions. When scanning a surface with at least some height variations we must expect that some small patches on the surface are situated such that the normal divides the angle between the laser and camera in two. If the surfaces in question have a high specular reflectance, see [24] page 41-44, the majority of the incoming light will be reflected directly into the camera. The effect is that the camera is blinded by the laser, so to speak, at this particular location within the image. In the remaining part of the image where the above property doesn't hold true the amount of light reflected into the camera is far less. Thus the observed intensity of the laser may vary a great deal throughout an image.
The point of all this is that while we can adjust the camera settings to fit a given surface in general we can not be guaranteed to have the desired signal strength of the laser line throughout the entire image.

5.5 Designing the algorithm

The image analysis is a sequential process that consist of the following few steps. First the presence of the line is detected, then the center is roughly estimated and finally a more precise estimate of the center is found. The following sections will give a description of this process.

5.5.1 Detecting the line

The setup of the system dictates that the laser lines primarily appear as horizontal lines within the image. Whenever the laser line is projected onto anything that is elevated from the ground the line appears to move downwards within the image. This is merely a consequence of how the laser and camera are positioned relative to each other.

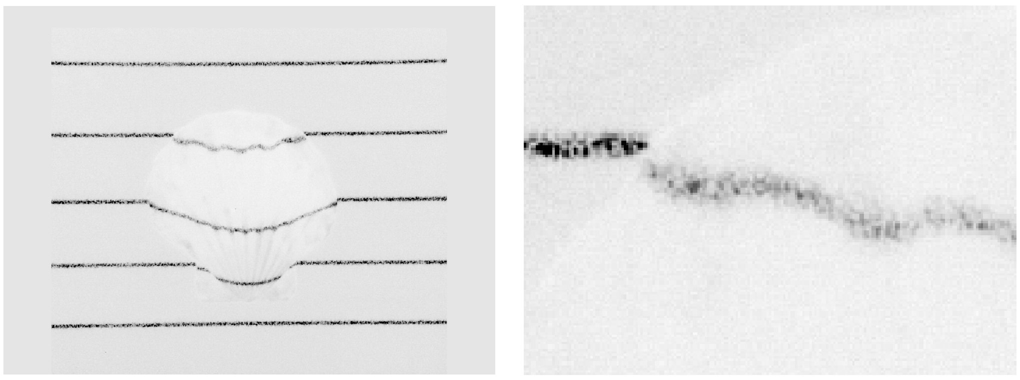


Figure 5.3: An image (negative) of five laser lines projected onto a shell. (left) entire image – (right) Zoom of the second line from the top as it touches the left edge of the shell.

Figure 5.3 shows the first image from the obtained dataset of a shell. Only five of the fifteen lines are falls within the cameras field of view and of these only three are actually projected onto the shell. Notice that the line is not entirely smooth – this is the effect of the so called speckle phenomenon, see [1]. In an attempt to suppress this phenomenon a median filter applied to the image. The median filter is chosen because it can suppress noise without smoothing the image. The median filter is especially suited for a type of noise known as "Salt and Pepper" noise where individual pixels are either much brighter or much darker than the surrounding pixels, see [15]. Applying a median filter replaces a given pixel value by the median of the pixel values in a local neighborhood of that pixel. The speckle noise present in the lines bear some similarity to "Salt and Pepper" noise which is why median filtering is well suited here. There are alternative methods to reduce speckle noise. In [25], [26] methods for reducing speckle noise in SAR-images are presented. These methods are however highly complex compared to the median filter which is why the latter is chosen.

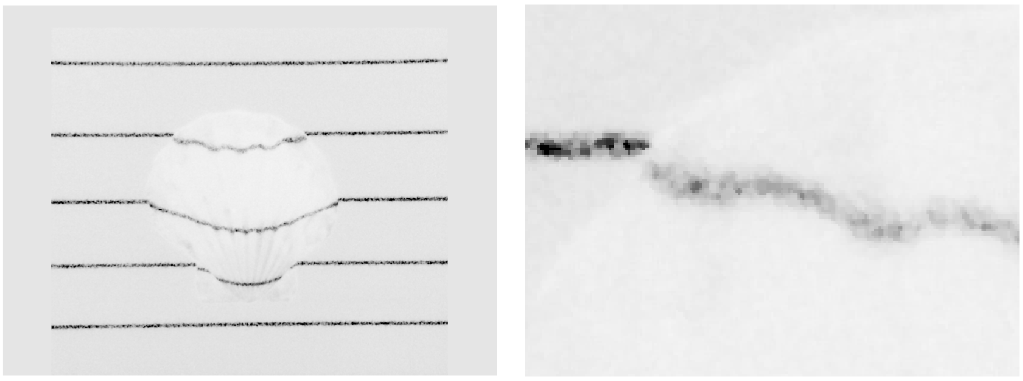


Figure 5.4: Result of median filter (size 3x3) applied to the source image. (left) Entire image – (right) Zoom. The median filter both replaces very light pixels with slightly darker pixels as well as very dark pixels with slightly lighter pixels. The result is that the speckle effect is suppressed to some degree.

Figure 5.4 shows the result of the median filtering – we can see that the speckle noise is somewhat reduces, it is however not completely gone. In order to further suppress the speckle noise the, a low pass filter is applied to the rows of the image.

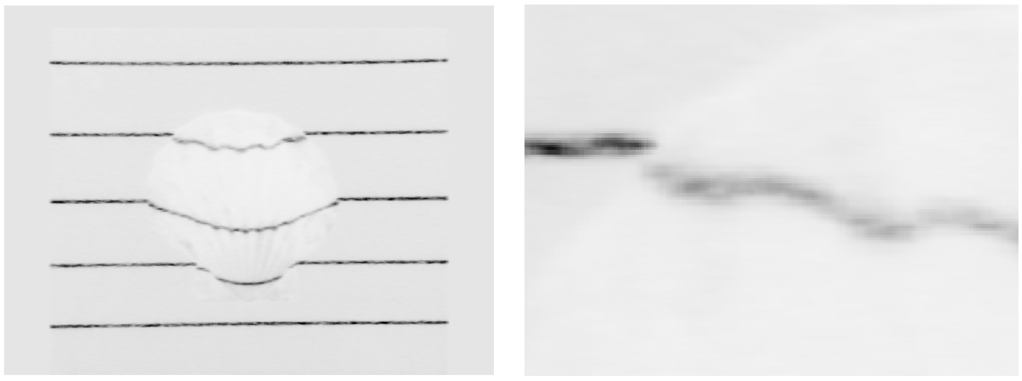


Figure 5.5: Result of lowpass filtering realized by a 1x10 averaging filter, hence the image is only smoothed in the horizontal direction. (left) Entire image – (right) Zoom.

Figure 5.5 shows the image after low pass filtering - the achieved effect is a horizontal blurring of the image. Since the lines are primarily horizontal the blurring further reduces the speckle noise. However the information that we want to obtain from the image is the vertical variations of the lines. Hence horizontal blurring is a potential dangerous technique to apply in this situation because it can wash out the information we are pursuing. Therefore it must be applied with care and kept to a minimum, meaning using the shortest possible averaging filter that will accomplish the desired result.

The next step in detecting the lines is an enhancement of the image with respect to the lines and for this purpose a filter mask is applied to the image. The mask is designed to give a high response when a horizontal line is encountered. The laser lines have a very uniform thickness of approximately 5 pixels. The filter mask is chosen to match this thickness.

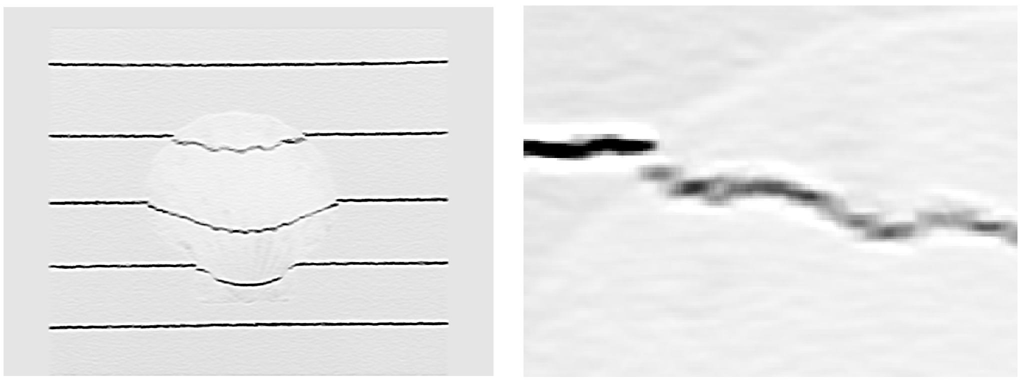


Figure 5.6: Result of line enchantment realized by vertical shape filtering (shape: [-1 -1 1 1 1 1 1 -1 -1]^T). (left) Entire image – (right) Zoom.

Figure 5.6 shows the effect of the shape filter applied the columns of the image. The lines are now so strong relative to the background that they can be detected simply by setting a threshold value. The result of this is a several pixel thick line - hence the desired result. Nevertheless white pixels do occur within the lines from time to time. This undesired phenomenon can be eliminated by modifying the threshold technique a little. Instead of a single threshold of the image, two threshold passes are used but in between the image is blurred. The purpose of blurring is to fill holes within the lines.

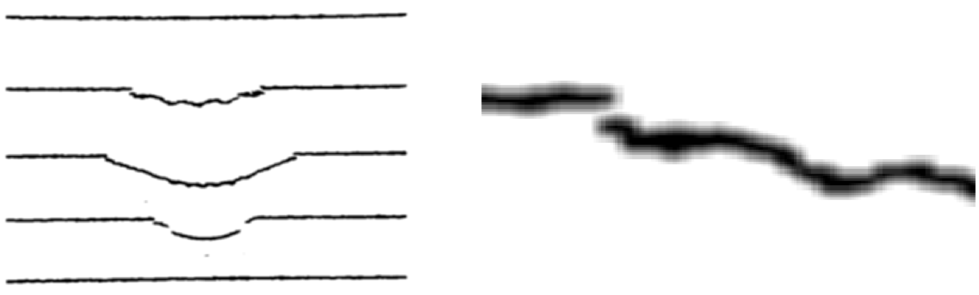


Figure 5.7: Result after first threshold and blurring. The blurring ensures that holes within the lines do not occur. (left) Entire image – (right) Zoom.

Figure 5.7 shows the image after the first threshold pass and the following blurring. Now a final threshold pass concludes the detection the lines.

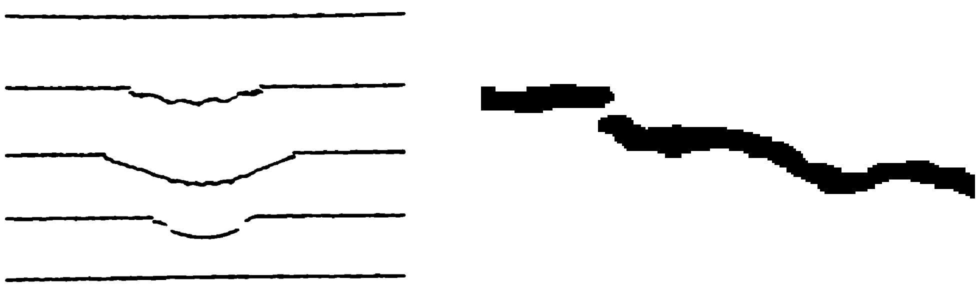


Figure 5.8: The detected lines represented by a binary image. The lines are several pixel thick but no holes occur within the lines. (left) Entire image – (right) Zoom.

Figure 5.8 shows the result of the final threshold pass which gives a binary image containing the detected lines. For the mathematical representation that we seek only the center of the line is needed. In order to obtain this a modified erosion of the lines are performed. This erosion work similar to the standard erosion, see [5] and [9], but operates on the columns of the image only. The technique is applied recursively until a one pixel thick line remains.

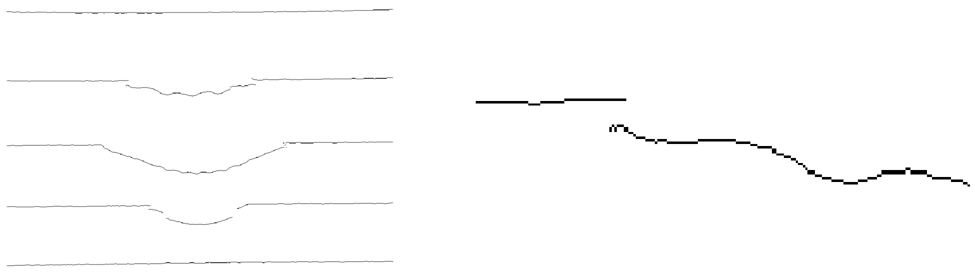


Figure 5.9 The result of eroding the binary image using a modified erosion technique that operates on the columns of the image. (left) Entire image – (right) Zoom.

Figure 5.9 shows the resulting binary image containing thin binary lines. This concludes the final step in detecting the lines and estimating the center roughly.

5.6 Optimizing the center estimate

So far we have obtained the center of the lines by means of pixel coordinate from a binary image, thus the center is represented by an integer value. Now we would like to improve this estimate by the use of optimization.

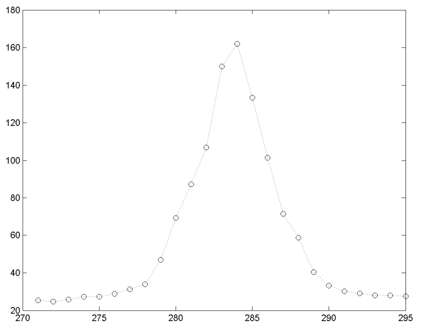


Figure 5.10: Cross section of a laser line subsequent to the horizontal blurring of the line detection section.

Figure 5.10 shows a section of a single line – the data is obtained after the horizontal blurring step. As previously mentioned the intensities of the laser orthogonal to the line should resemble a gauss bell and the figure seems to support this suggestion. A model of the data would therefore merely be a gauss bell plus an offset value representing the average background intensity, see (5.1) – where **x** is a vector containing the model parameters: x_1 the height of the curve, x_2 the center with respect to *t*, x_3 the width (or standard deviation) of the curve and finally x_4 is the offset.

$$M(\mathbf{x},t) = x_1 \cdot e^{\left(\frac{-(t+x_2)^2}{2x_3^2}\right)} + x_4 \tag{5.1}$$

Figure 5.11 shows the model and data. The parameters used here are just manually plugged in, the resulting model doesn't fit the data very well but we get feel of the capabilities of the model and it seems to able to fit the data at least.

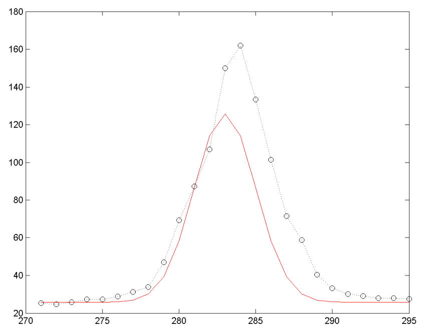


Figure 5.11: Cross section of laser line (data) and the corresponding model. Data: black circles connected with dotted line. Model: red continues line.

Obviously the parameters need an adjustment before the model will be of any use. For that purpose an optimization algorithm is used. Before the algorithm can be applied we need to state the problem in a way that the algorithm can understand – namely as an optimization problem. Therefore a function which calculates the residuals and the gradient of the residual vector with respect the model parameter vector **x**, namely the Jacobian, is created - see (5.2). The residuals are the element wise differences between the data and the model.

$$
\begin{gathered}
r = y - M(\mathbf{x},t) = y - x_1 \cdot e \frac{-(t+x_2)^2}{2{x_2}^3} + x_4 \\
J(\mathbf{x}) = \begin{bmatrix} -e^{\left(\frac{-(t-x_2)^2}{2{x_3}^2}\right)} \\ -x_1 \dfrac{t-x_2}{{x_3}^2} \cdot e^{\left(\frac{-(t-x_2)^2}{2{x_3}^2}\right)} \\ -x_1 \dfrac{(t-x_2)^2}{{x_3}^3} \cdot e^{\left(\frac{-(t-x_2)^2}{2{x_3}^2}\right)} \\ -1 \end{bmatrix}
\end{gathered}
\tag{5.2}
$$

Using this function the optimization algorithm can now be applied. The used algorithm is the so-called Levenberg-Marquardt method, see [19]. Apart from the model the Leverberg-Marquardt algorithm also requires a starting guess of the model parameters. Given the amount of data to process these cannot be filled in manually as above. Fortunately they can easily be obtained however: x_1 – the intensity value at the center is used, x_2 the integer valued center already found is used, x_3 – a constant value of five is suitable, x_4 – an estimate of the background intensity is used.

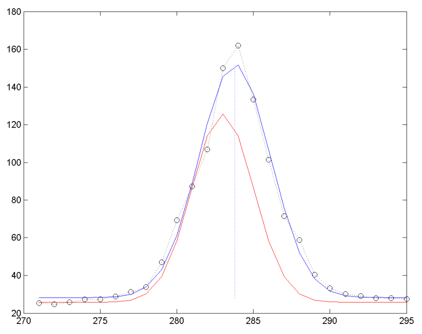


Figure 5.12: Data and optimized model. The optimized model (blue solid line) is very close to the data (black circular markers). The initial model is also shown (red solid line). The center of the optimized model is indicated by the vertical black dotted line. Notice that the estimated center is slightly to the left of the data point with the largest value.

Figure 5.12 shows the data and the model after the model parameters have been optimized. As can be seen the optimization works intentionally and thus provides us with a subpixel estimate of the center of the line.
When working with optimization it's always good practice to have a look at the residuals subsequent to the optimization.

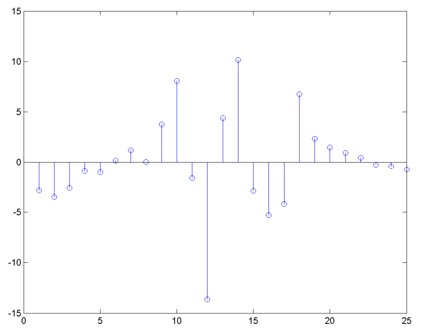


Figure 5.13: The residuals subsequent to optimization. The residuals seems mainly to consist of noise but there are also some trends present.

Figure 5.13 shows the residuals. While they primarily just are noisy they do also show some trends which means that the model doesn't exactly fit the data. There are two possible reasons for this: either the applied model is incapable of attaining the shape of the underlying (true) model or the algorithm is incapable of estimating the true parameters. The latter can happen if large residuals pull the model away from the correct solution. A combination of these two is naturally also a possibility and this is very likely to be the case here. However the model does seem to fit the data very well and therefore there is no need to further pursue this matter.

5.7 Optimizing entire lines

The optimization technique described in the above section is fully automated and can therefore be applied every single point constituting an entire line.

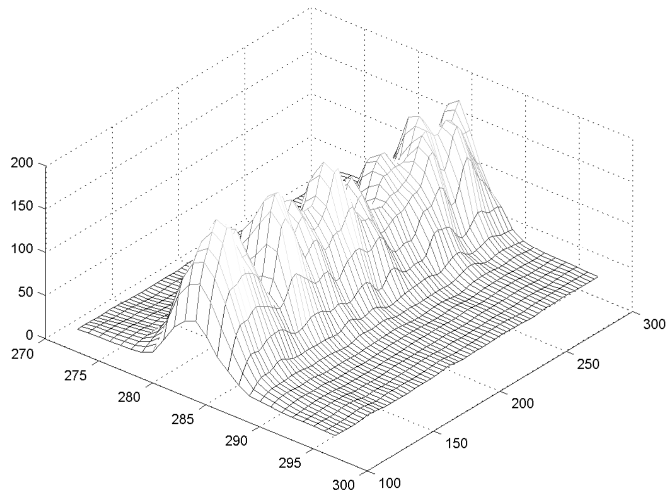


Figure 5.14: 3D surface mesh of a part of a line based on the intensity values from the horizontal blurred image that constitutes the data for the optimization.

Figure 5.14 shows a surface mesh of the intensity values representing part of a line. As can be seen the surface is rather hump-shaped, despite the attempts made to even these out. The humps originate from the speckle phenomenon earlier mentioned. The variations that occur along the ridge don't matter much – it is the peak of the humps in the direction orthogonal to the direction of the ridge that is of concern. Unfortunately these humps don't necessarily lie right at the true center of the line – they seem to move up and down a bit. This particular line segment is obtained at a flat surface so it should be relatively straight. It is not possible to smooth the line further since that would start to degrade the actual information which is pursued.

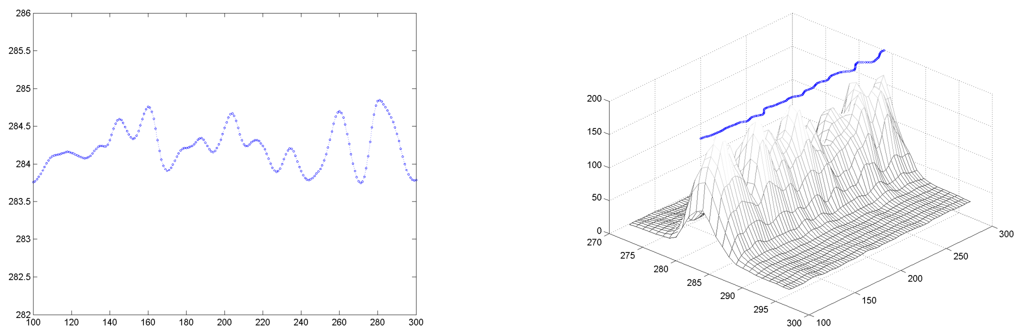


Figure 5.15: (left) Center of the line after optimization. The center is influenced by the speckle phenomenon and fluctuates as a result. (right) The surface mesh that representing the line and the estimated center shown on top.

Figure 5.15 shows the optimized center of the line. The center fluctuates as a consequence of the speckle phenomenon. The fluctuations of the center appears to have a magnitude of half a pixel or so. Thus, the aim of getting sub-pixel accuracy has been achieved but just barely as the estimate of the line center is very sensitive to the speckle noise.

5.8 Summary

In this section a method for detecting and estimation the position of the lines was presented. The algorithm is based on known image analysis techniques although some has been subject to slight modifications. The method have been tested on hundred of frames and works very well. Some of the threshold values used by the algorithm requires manual adjustment if significant changes to the light conditions are med. These can however relatively easily be set adaptively. Special effort for the algorithms ability to estimate the line position with sub-pixels accuracy has been made. It is believed that this accuracy can be further improved if the speckle noise can be more successfully reduced.

6 Line Segment Correspondence

6.1 Introduction

A line projected onto an object does from the lasers point of view appear as continues line. From the cameras point of view the line appear to fluctuate vertically according to the geometry of the objects being scanned. It is these fluctuations of the line that eventually give us height information about the scanned object. If the geometry of the object include step vertical inclinations the fluctuations of the line may be so abrupt that the line becomes discontinuous from the cameras point of view. Thus the line may brake into line segments. Since the acquired images contain multiple lines the line segments found in a single image originate from different lines. So, how can we tell which line segment belong together? This question is the topic of this section.

Line segment correspondence in general is a broad concept, in [7] David et al. proposes an algorithm for establishing the correspondences of model lines to image lines. Chun Lee et al. [17] presents an algorithm for correspondence between 2D and 3D line segments in order to estimate the position and orientation of a camera. Brown et al. [4] presents a method for tracking line segments through an image sequence. Corresponding line segments are matched by formulating the problem as linear least squares problem.

6.2 Overview

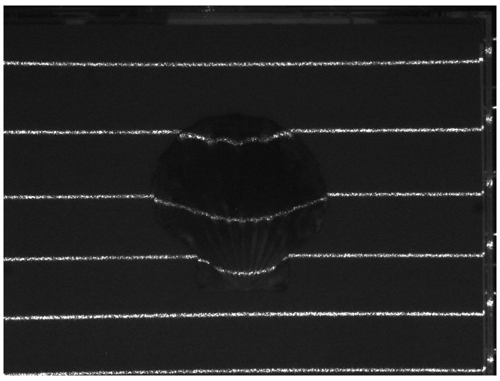
In the following section two approaches to solve the line segment correspondence is presented. In the first method the analysis is made on a single image basis whereas the second approach bases the analysis on the entire image sequence. The latter uses a line segment matching technique the track line segment across frames. This part is inspired of the line tracking technique of Brown et al. [4]. The sequence based analysis also uses a labeling technique similar to the standard labeling approach, see [5], [9] and [15], used for binary images.

After the correspondence problem has been solved there is one last issue that must be dealt with. The acquired image sequence contains only about have the line emitted by the laser. It must therefore be determined which of the 15 lines the camera sees. The section line identification looks at this subject.

6.3 Illustration of the correspondence problem

The line detection part generates a list of the line segments within an image (LSa – Line Segment array). Such a list is generated for each frame in a sequence of images. The task is now to identify the lines in the given image from the list

of line segments. Each single line segment may represent an entire line or only a tiny fraction of a line. This depends on the geometry of the object being scanned.
Let's first look a concrete example, Figure 6.1 shows an image of the laser lines projected onto a shell and the corresponding extracted line segments. In this case most of the lines are unbroken, but one line is broken two places and another line is broken at one point (this way be difficult to see on print). The object used here is neither very high nor does it have any very steep vertical surfaces. As a result the projected lines don't travel very far from their corresponding baselines (the lines projected onto the ground). For this reason it's very intuitive to see which line segment constitute a complete line.



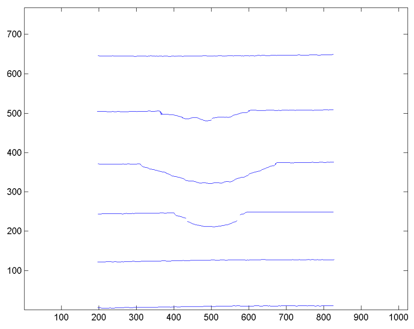
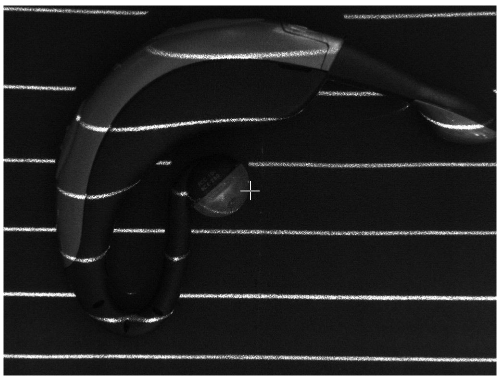


Figure 6.1: (left) Laser lines projected onto a shell. (right) The corresponding extracted line segments.

Naturally we should be able to handle objects a bit more complicated than this. Figure 6.2 shows an image of the laser lines projected onto a wireless headset and the corresponding extracted line segments.



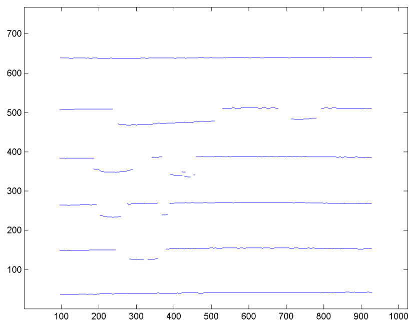


Figure 6.2: (left) Laser lines projected onto a wireless headset. (right) The corresponding extracted line segments.

This object is also not very high and therefore we again observe that the elevated line segments appear close the corresponding baseline. However this object has got some steep vertical surfaces which produce the discontinuities that we observe in the projected laser lines. Although it is still very intuitive to see which line segments constitute a line, this object illustrates some of the phenomena that we run into when projecting laser lines onto different objects. Due to the position of the laser all elevated line segment have moved downwards in the image. We can observe that a part of the line is generally missing around discontinuities. It turns out that this is most usually the case, but it can however also happen that a discontinuity results in an overlap, see Appendix A.
It is crucial that we are able to identify the line segments that constitute a line. If a line segment is mistakenly connected to a neighboring line it will yield a false 3D reconstruction. Generally there are two types of information that can be utilized to figure out which line segments constitute a line. One is spatial information related to the line segments within a single image. The second is the information about how line segments move through a sequence of images.

6.4 Analysis based on single images

In the following section we will explore the possibilities of solving the line segments correspondence problem on a single image basis. The following describes the proposed algorithm.

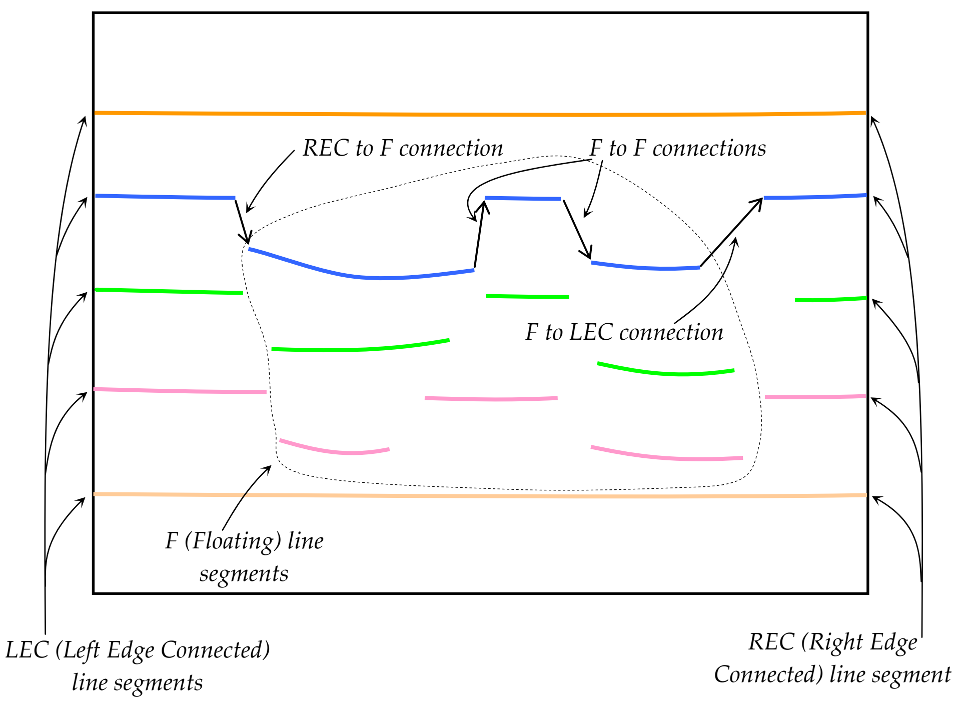

REC to F connection
F to F connections
F to LEC connection
F (Floating) line segments
LEC (Left Edge Connected) line segments
REC (Right Edge Connected) line segment

Figure 6.3: Illustration of line segment correspondence in a single frame. Line segments constituting a line a given the same color.

Figure 6.3 illustrates the problem at hand. The algorithm should be able to figure out which line segments constitute a complete line.
We first identify line segments connected to the left edge of the image. Such line segments are referred to as LEC (Left Edge Connected) line segments. Line segments connected to the right edge of the image are referred to as REC line segments. A line segment can be classified as both LEC and REC – this simply means that the line segment touches both the left and right edge of the image and that the line segment therefore constitutes a complete line by it self. However two LEC line segments can't be part of the same line – neither can two REC line segments. Line segments that aren't connected to any of the edges is referred to as floating or merely F line segments. A line can contain multiple F line segments.
We adopt that lines begin at the left edge and end at the right edge of the image. Hence each line starts with a LEC line segment and ends with a REC line segment. Therefore for each LEC line segment in the list of line segments (LSa) we can initialize a new line and assign the LEC line segment to that line. As line segments are assigned to a given line they are also removed from the list of line segment (LSa). Thus the LSa list contains only the unprocessed line segments. When the list is empty, meaning that every line segment have been assigned to a line, the analysis is done.

After assigning each LEC line segment to a line we now also have a list of lines (L). This list now contains the beginning of all the lines in the image. When we assign further line segments to a given line, this is done from the left to the right. Therefore once a LEC and a REC line segment have been assigned to the same line, the line is considered complete and is disregarded henceforward in the analysis. Some of the LEC line segments added to the list of lines (L), may also be REC line segments. These lines are therefore already complete, even before adding additional line segments to these lines. To sum up: we now have a list of lines (L) all containing one LEC line segment. Some of these lines are complete and some are only partial lines. We also have a list of line segments (LSa) containing the remaining line segments to process - see Figure 6.4. So far the analysis has been straightforward. Now comes the difficult part: assigning (or connecting) the remaining line segment to the proper lines.

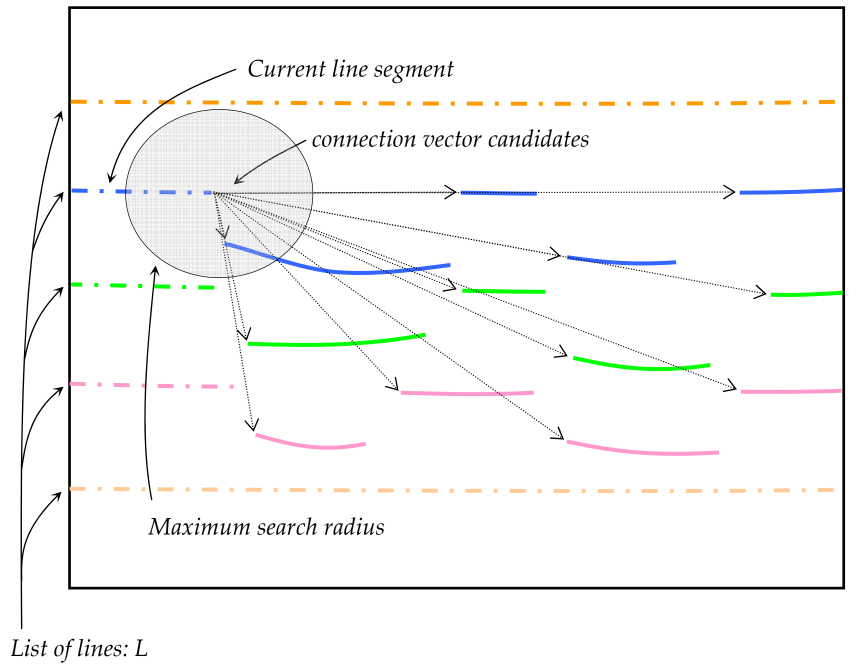


Figure 6.4: The REC line segments are utilized to initialize a list of lines L (the dash-dotted lines). Next the remaining line segments (solid colored) must be connected to the proper lines. A list of connection vectors is generated, corresponding to each of the candidates, and the proper line segment is chosen based on this list.

Taking starting point in a line segment from the list of lines, the line segment which constitutes the continuation of the line must be chosen from the list of line segment. This choice is based on the connection vector, defined from the end of the current line segment to the beginning of the next line segment - see

Figure 6.4, but also on the type of connection taking place. Since we initially only have LEC line segment in the list (L) there are only two possibilities: we can connect a LEC line segment to a REC line segment or a F line segment. We write this as LEC»REC and LEC»F respectively. Thus, initially the only possible connection type is LEC»? As we get F line segments into L there arise two more possibilities: F»F and F»REC. These constitute the second connection type F»? In Figure 6.3 page 48 three of the mentioned connections are illustrated. Knowledge about which type of line segments are about to connect can be used to impose difference rules on the given connection or to aid the selection of line segments.

In the case of LEC»? the selection is: shortest connection vector (**CV**) with positive y-component. Positive y-component ensures that the vector points downwards. Once this selection has been made a number of further requirements must be met, which are:

$$\mathbf{CV}_y \leq \mathbf{CV}_y \max \wedge \mathbf{CV}_x \leq \mathbf{CV}_x \max \tag{6.1}$$

$$\mathbf{SCV}_y \leq \mathbf{SCV}_y \max \wedge \mathbf{SCV}_x \leq \mathbf{SCV}_x \max \tag{6.2}$$

$$\neg\, ?\text{»}REC \vee \left| LEC_y(1) - REC_y(end) \right| \leq VEA \max \tag{6.3}$$

The first requirement ensures that the x- and y-components of the connection vector are less than or equal to some predefined threshold values. The purpose of this is to make sure that connections can be made within a certain local region only. The second requirement is similar except the vector in question is the SCV vector, which is the Sum of Connection Vectors along the route so far. Here the $\mathbf{SCV}_y$max threshold is chosen much smaller than the $\mathbf{SCV}_x$max threshold – the meaning being that the line should stay within a certain vertical range. The last requirement means that: in case a REC line segment is being connected (which would complete the line), the y-values at the two edges should be similar - thus there difference should be less that or equal to a predefined threshold value (VEA – Vertical Edge Alignment). The purpose of the last requirement is to make sure that the line begins and end at the same (or nearly the same) vertical position within the image.

If the requirements are not met the selected connection vector is rejected and a new is selected instead. If all possible connection vectors are rejected the algorithm is unable to connect further line segments to the current line and it then continues to the next line.

In case the current connection type is F»? Then the connection vector yielding the shortest SCV is selected. Hereafter the three above mentioned requirements are applied.

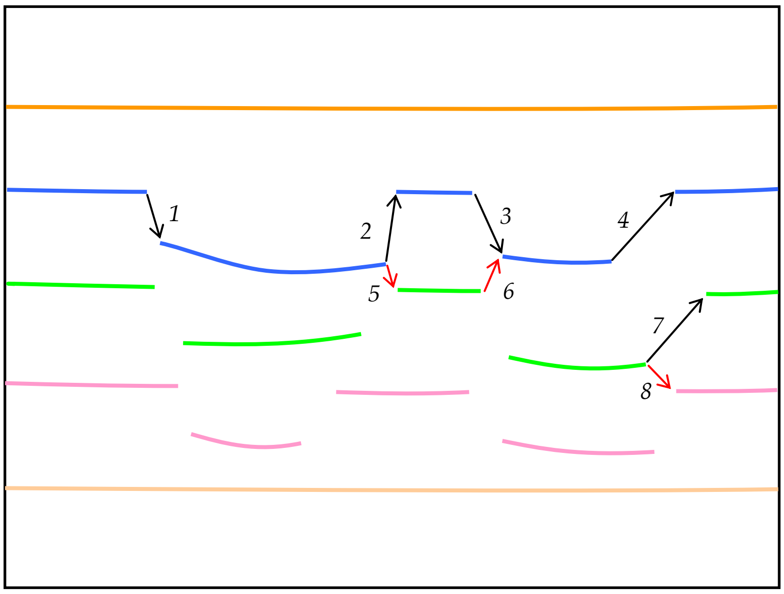


Figure 6.5: Illustration of analysis scenario. The numbered connection vector show different possible routes for the algorithm to chose from.

Figure 6.5 shows an illustration of an analysis scenario. Starting from the top and working our way down the left edge of the image we first encounter the orange line. This line segment is classified both as LEC and REC and is therefore added to the list of lines L along with the other LEC line segment when L is initialized. Now L(1) (the orange line segment) contains both a LEC and a REC line segment and is therefore considered complete and is thus disregarded henceforward. The next line encountered is the blue. The first line segment is already in L. The line is not considered complete and the algorithm will therefore try to connect additional line segments. The connection vector numbered 1 is the shortest of all possible connection vectors (not shown, see Figure 6.4 page 49) and is therefore selected. Furthermore it passes all the requirements previously stated and is therefore accepted. The next connection vector selected is the one numbered 5. This CV is very short and therefore clearly fulfills the first requirement. However not the second – the y-component of the SCV vector is larger that the $\mathbf{SCV}_y$max value, which should always be chosen less than the distance between two consecutive baselines. This means that CV number 5 is rejected. The next possibility is CV number 2 – it also fulfills the first requirement, but what about the second? Notice that the y-component of CV 1 and CV 2 roughly cancel each other out and hence the second requirement is also fulfilled. In a similar manner the CV 3 and CV 4 are selected and accepted. Notice that at CV 4 the third requirement comes into play, however it should be clear from the illustration that this requirement is fulfilled in this case. This then completes the blue line. Now imagine that we have worked our way through the green line in a similar manner and are now facing the choice of CV 7 or CV 8. Again the second requirement should prevent the algorithm from choosing CV 8 but let's say that it doesn't then the third requirement clearly should. So in this case requirement three would save the algorithm from making a terrible mistake. Let's for a moment return to the blue line and imagine that the algorithm had chosen the route dictated by CV 5 and CV 6. This could occur if the $\mathbf{SCV}_y$max value is chosen too large. In this case the third requirement wouldn't have been able to catch the mistake as the algorithm return to the correct line before it reaches the edge of the image. Thus there is no guarantee that the algorithm always selects the correct route, which of course is a serious shortcoming.

6.5 Analysis based on image sequence

Another approach to the line segment correspondence problem is to base the analysis on the entire sequence of images. This means that when the algorithm runs through the image sequence it only gathers information and first when all images have been analyzed the information will be put together. The idea for the purposed algorithm is to simply track line segments throughout the image sequence. So for the moment we don't worry about which line segments constitute a line – we only worry about the correspondence between single line segments from one frame to the next. The key to tracking line segments is a priori knowledge about how the line segments are expected to move. Basically the frame to frame movement of the line segments seen by the camera depends on two things: the speed of the intersection curve and the geometrical properties of the object. By the speed of the intersection curve is meant how far the intersection curve travels in between consecutive frames. Thus if we have an idea about the above properties we can predict where a line segment should appear in the following image. However we know that the intersection curves travels in tiny steps in order to get a good coverage of the object. This means that we can simplify things ever further – we can simply assume that line segment doesn't move from one frame to the next. This assumption is of course wrong and will therefore introduce an error but as we shall see this error is very small in the relationship in which it's been used.

6.5.1 Line segment matching

For the algorithm to work, a method is needed that can compare two line segments in two consecutive frames and make a decision weather the line segments are the same of not. A line segment consists of a number of discrete points originating from pixel positions. The x-values are integer values while the y-values are floating point values subsequent to the optimization. The proposed measure of how well such line segments match is simply the sum of the pixel wise difference between corresponding pixels relative to the number pixels that the two lines have in common. This match is referred to as the Relative Match (*RM*) and is expressed in (6.4).

$$RM = \frac{\sum_{i_c}^{i_c \in C\{n,k\}} \left| Line_n(i_c, y) - Line_k(i_c, y) \right|}{\# C\{n,k\}} \tag{6.4}$$

where $C\{n,k\}$ is the list of x indexes that the lines n and k have in commen and $\# C\{n,k\}$ is referred to as the Absolute Match Basis.

Figure 6.6 shows an illustration of how two line segments are compared. The idea is that a single line segment in frame n is compared to all line segments in frame n+1 using (6.4) to calculate a measure of how well the current match is. Now, for all line segment that lie far away from the line segment that they are compared against, the measure *RM* will yield a large number, while for all line segments close by, and especially the true match, *RM* will be small. Hence a suiting threshold value will eliminate almost all non-matches. There may also exist line segments where there is no horizontal overlap, meaning that $C\{n,k\} \in \varnothing$. For such line segments *RM* is not defined, and these line segments are disregarded as they surely aren't matches.

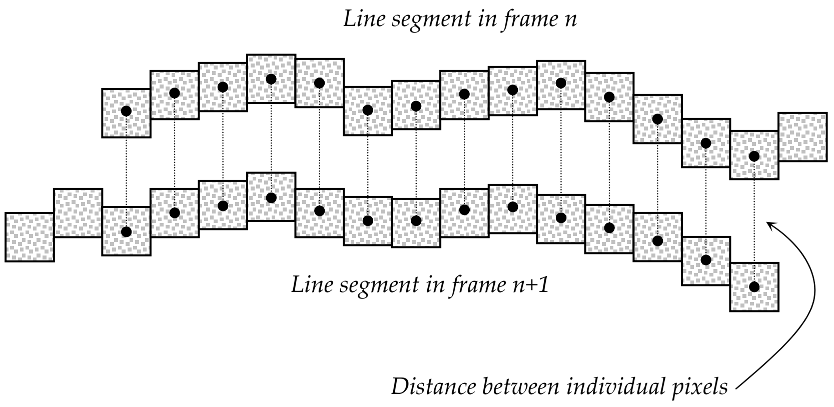


Figure 6.6: Comparison of line segments in two consecutive frames. As a measure of how well the lines match is based on the individual distances between corresponding pixels.

It's natural to think that one line segment should yield one and only one match, but in fact one line segment could yield multiple matches. Let's say, as an example, that at frame n a given line segment is 500 pixels long. In frame n+1 this line segment has broken into three smaller pieces. Now, all three pieces are correct matches for the line segment at frame n. Figure 6.7 shows an illustration of this.

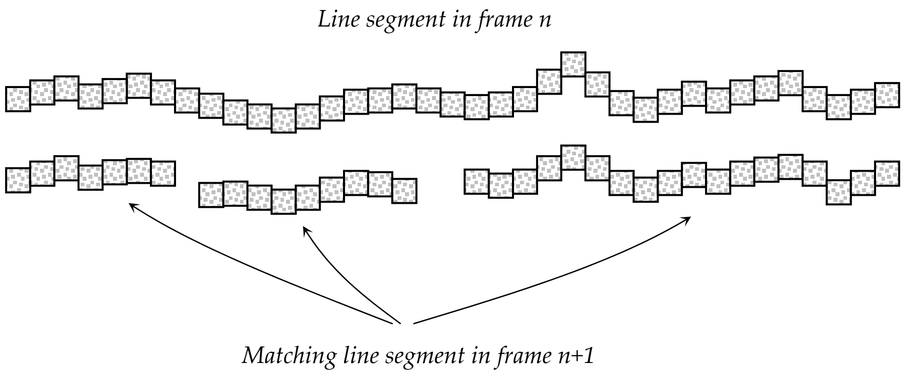


Figure 6.7: A single line segment breaks into three line segment between subsequent frames, thus generating multiple matches.

The above described situation may also happen in reverse order, meaning that three line segments in frame *n* join into one single line segment in frame *n*+1. In this case each of the three line segments will match the merged line segment in frame *n*+1.
In order to further sort the list of possible matches it's helpful to extract more information about the individual matches. Another proposed match measure is the Relative Match Basis expressed in (6.5).

$$RMB = \frac{\#C\{n,k\}}{2\cdot\#line_n} + \frac{\#C\{n,k\}}{2\cdot\#line_k} \tag{6.5}$$

The Relative Match Basis simply tells us how much two line segments overlap horizontally relative the there length. Returning to Figure 6.7, the line segment in frame *n* consist of 37 pixel and the three line segments in frame *n*+1 consist of 7,9 and 18 pixels respectively. For the 7 pixel line segment the Relative Match Basis is calculated as follows:

$$RMB = \frac{\#C\{n,k\}}{2\cdot\#line_n} + \frac{\#C\{n,k\}}{2\cdot\#line_k} = \frac{7}{2\cdot 7} + \frac{7}{2\cdot 37} \cong 0.5946 \tag{6.6}$$

The *RMB* for the other two line segments are 0.62 and 0.74 respectively. Notice that the expression for calculating the *RMB* is constructed such that any *RMB* ∈ [0;1]. While the *RM* in general give a very consistent and reliable measure to weather two line segments match, it can in some rare situations give some misleading result. Imagine that two relatively long line segment only have a tiny horizontal overlap. If the line segments are close where the overlap occurs the *MR* measure will yield a small value. However we don't necessarily want to declare these line segments as matching based on this tiny overlap. This is the motivation for the *RMB* measure which will reveal the lack of overlap. Thus after first eliminating match candidates by setting a *MR* threshold, we can (possibly) eliminate further candidates by also setting a *RMB* threshold. Finally a third measure, the Absolute Match Basis (*AMB*), is proposed. The *AMB* measure is applied in order to guarantee a certain minimal length of the line segments being matches. It is worth mentioning that *MR* threshold perform the majority of the work while the *RMB* and *AMB* threshold seldom make any difference, but they do however prevent the algorithm from making false matches.

Absolute Match Basis	**Relative Match**	**Relative Match Basis**
58	4.2160	0.6146
199	4.9951	0.8908
4	9.6630	0.0209
3	10.0427	0.5059
249	129.6073	0.8033
253	140.2215	0.8131
253	255.0363	0.6524
253	286.6429	0.6526

Table 6.1: List of Absolute Match Basis, Relative Match and Relative Match Basis measures generated by matching a single line segment against all line segments of the next frame. Line segments with no horizontal overlap are not included.

Absolute Match Basis	**Relative Match**	**Relative Match Basis**
58	4.2160	0.6146
199	4.9951	0.8908
4	9.6630	0.0209

Table 6.2: List of Absolute Match Basis, Relative Match and Relative Match Basis measures subsequent to the *RM* threshold step using the threshold value $RM < 10$.

Absolute Match Basis	Relative Match	Relative Match Basis
58	4.2160	0.6146
199	4.9951	0.8908

Table 6.3: List of Absolute Match Basis, Relative Match and Relative Match Basis measures subsequent to the *RBM* threshold using the threshold value *RBM* > 0.55.

Table 6.1,Table 6.2 and Table 6.3 shows how a list of match candidates are gradually reduced imposing two of the three threshold steps. The *AMB* threshold step is not shown here as it doesn't change the list, which is mostly the case. The resulting list of matches contains two line segments. We can't tell there exact length but we can conclude that they are at least 58 and 199 pixels long respectively. The line segment that they were matches against is at least 253 pixels long. This is a very careful selected example due to the fact that we end up with two matches. Normally there would be one or none at all. The mentioned possibilities can be categorized as follows:

- No matches: this happens when the algorithm can't match the current line segment to any of the line segment in the subsequent frame. It can happen that a line segment disappears from one frame to the next. This is causes by large vertical changes in the geometry of the object being scanned.
- One match: this is the case most likely to happen, it simply means that a line segment have been successfully tracked from one frame to the next.
- Two or more matches: this is as mentioned very rare but also very important since this only happen when two or more line segments merge into one. This event is very important when solving the line correspondence problem.

Using the above described procedure ensures a very robust way of tracking the line segments across frames.

6.5.2 Utilizing the match information

Another aspect of the image sequence based analysis is that the correspondence problem is treated as a labeling problem – very similar to labeling pixels in a binary image, see [1], [7] and [1]. From the beginning each line segment in the first frame is given its own unique label. As the analysis progress, the objective is to transfer the labels to the corresponding line segments in the subsequent frames. This correspondence is determined by the matching algorithm described in the previous section. In this manner the line segments are tracked through the image sequence. In case new line segments appear they are just given a new label.

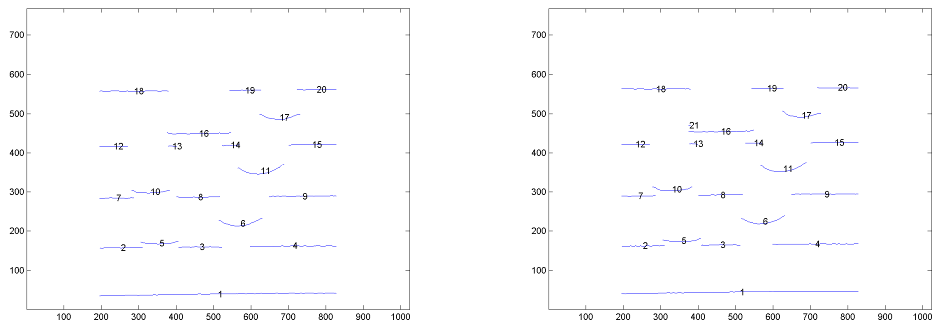


Figure 6.8: Initial labeling of the first two frames in an image sequence. (left) Frame one (right) Frame two. In frame one the line segments have been sequential numbered. All of the labels have been transferred to frame two, meaning that all line segments have been successfully tracked between the two frames.

Figure 6.8 shows the initial labeling of the first two frames in a sequence. Initially the line segments are merely sequentially numbered. As the line segments of frame two are matches against the line segments of frame one, they inherit the corresponding labels. In frame two a new tiny line segment appears – this segment have been labeled 21.
As mentioned line segments tend to break or merge. In case a line segment breaks into two segments, both these will make a positive match to the one segment from the previous frame and they will both inherit the label from this segment. An example of this is shown in Figure 6.9.

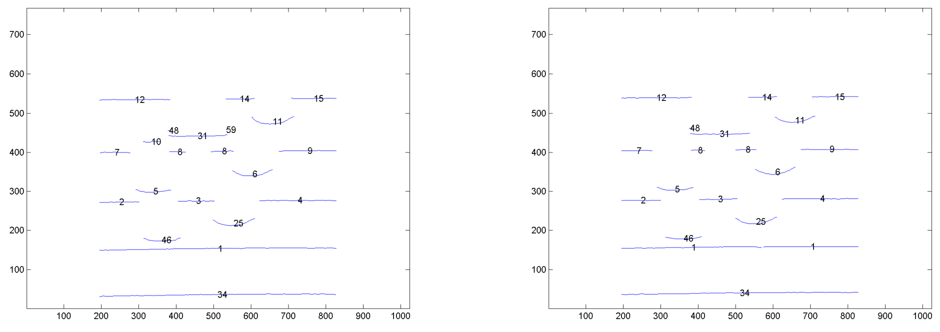


Figure 6.9: A line segment breaking into two segments. (left) Frame 40, the line segment labeled 1 is in one piece. (right) Frame 41, the line segment has broken into two segments. Both have inherited the same label.

In case two line segments merge, they will probably have two different labels and thus the two labels collide. The merged segment will inherit the label from one of the two line segments and a note is made that these two labels correspond to the same line.

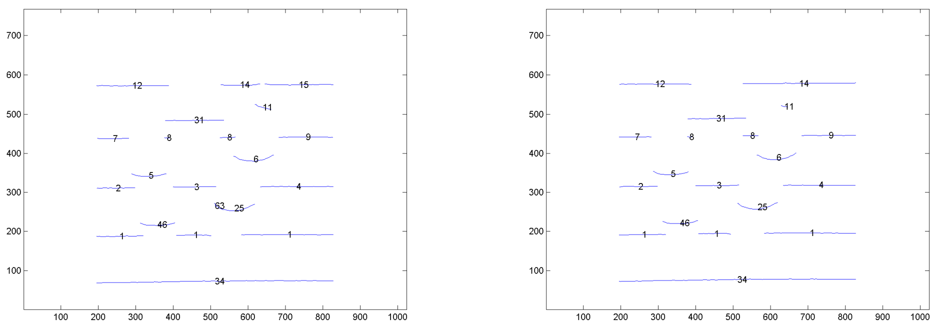


Figure 6.10: Two line segments merging into one. (left: frame 49) The gap between the line segments labeled 14 and 15 have been decreasing for a while and is now very small. (right: frame 50) The line segments have finally merged into one segment which is labeled 14.

Figure 6.10 shows an example of this. The labels 14 and 15 are put on a list and saved for post processing. When there is a gap between two segments the missing part is usually to be found. In this case the missing segment is labeled 11 and is located just below the two segments. Notice a funny phenomenon – when the two segments merge at frame 50 the small line segment labeled 11 is still present (it does however disappear at the next frame). This means that this part of the laser line is detected twice. When the projected line jumps from an elevated surface and onto the ground this usually happens in one frame. Sometimes however the image is taken at the exact time where half the line is present at the elevated surface and half is present at the ground, meaning that the line have been split along its length. The above mentioned phenomenon occurs when this happen, see Appendix A.

The idea of the labeling approach is that all of the line segment constituting a line will have made contact at some point within the sequence of frames. As a consequence, when the line segments have been re-labeled according to the detected label collisions, all the line segments belonging to the same line will have the same label. This in turn means that the line segment correspondence problem has been solved for this particular line. This does sound pretty easy, but we must remember that the assumption was that all line segments hade made contact, directly or indirectly. Therefore it's interesting to examine under which circumstances it can be guaranteed that this will occur. The answer is to this is actually straightforward: if the line in question at some point is projected onto the ground only, then all line segments will be connected through this point. Thus if the line is tracked from a point before it reaches the object or if it's tracked until it have passed the objects, then the line is projected onto the ground only. Lines that are projected onto the object throughout the entire sequence don't meet this requirement. For those lines there is therefore no guarantee that the correspondence problem is solved, but it may well be nonetheless.

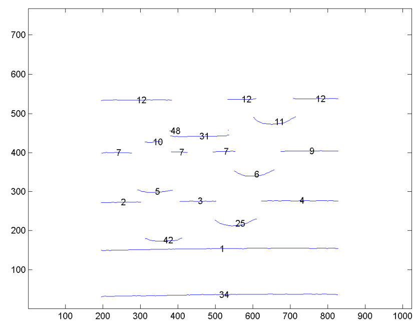


Figure 6.11: Frame 40 subsequent to post processing of labels. The correspondence problem has been solved for the baseline part of the lines labeled 1, 12 and 34.

Figure 6.11 shows frame 40 in the sequence after the post processing of the labels has been performed. Here the line correspondence has been solved for the baseline part of the lines labeled 1, 12 and 34. Of these the line labeled 12 is the most interesting. At this point the line consists of three segments which will merge later in the sequence. The rightmost merge at frame 50 as we saw in Figure 6.10 page 59. The leftmost merges with these at a latter point. There are three segments labeled 7 which almost, except the segment labeled 9, constitute the base line the second line from the top. This illustrates an example where the correspondence problem of the baseline part almost is solved. If the sequence had run for a few more frames the last line segment would also have been connected as where the case for the topmost line.

6.5.3 Tracking line segments indirectly

There is one type of line segments that can not be tracked directly. These occur when the object contain step vertical discontinuities. In the transition between ground and object the resulting projected line jumps a huge amount of pixels within the image. The line tracking algorithm is incapable of tracking the line segment as the jump occurs. As a consequence the label from the baseline segment is not transferred to the corresponding elevated segment. In Figure

6.11 these line segments are the ones labeled 5, 6, 10, 11, 25, 31, 42 and 48. Fortunately the link between these segments and there corresponding baseline can relatively easy be established using another approach. The elevated line segments appear when the laser intersects the elevated surface and disappear when the intersection leaves the surface. The appearance and disappearance of these line segments naturally happens in connection with splitting and merging of there corresponding baselines – this is illustrated in Figure 6.12.

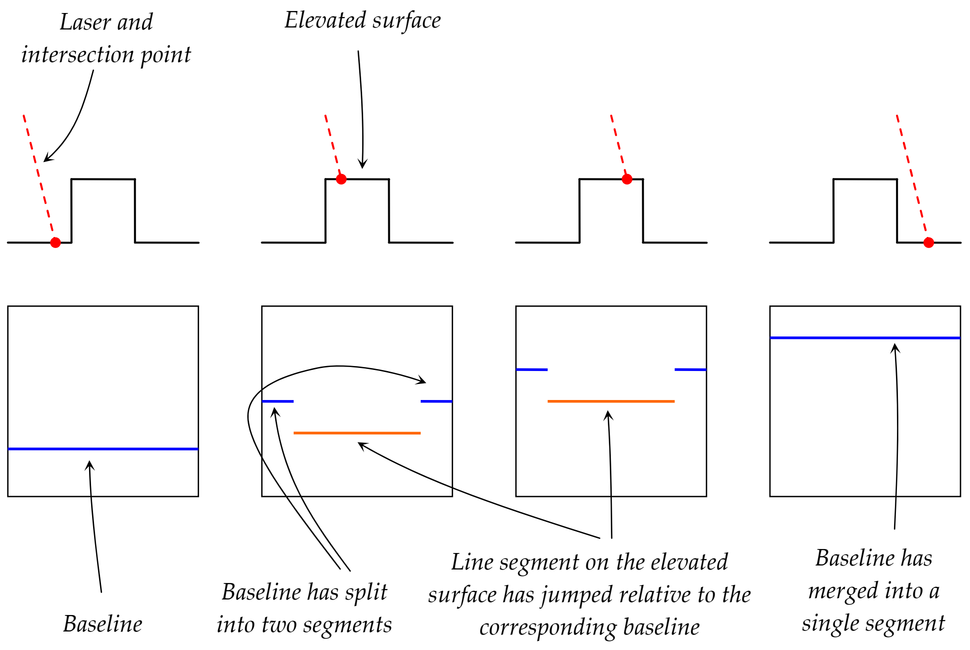

Laser and intersection point
Elevated surface
Baseline
Baseline has split into two segments
Line segment on the elevated surface has jumped relative to the corresponding baseline
Baseline has merged into a single segment

Figure 6.12: Illustration of an image sequence where the laser sweeps across an object with step vertical discontinuities. (above) Side view of the laser sweeping across the object. (below) Corresponding image sequence. The illustration shows how the line segment on the elevated surface jumps relative to the corresponding baseline and how it appears and disappears relative to the splitting and merging of the baseline. It is assumed that the elevated surface doesn't fill the entire width of the image, thus leaving one baseline segment on either side.

Consequently the "jumping" line segments can be linked to there corresponding baselines simply by noting the frame number of there appearance and disappearance and comparing these against the frame numbers where line merging and splits occurs. In order not to make false links it's a good idea also to compare the spatial location of where these events occur.
Line merging and splitting can be detected by monitoring the number of matches produces by the individual line segments. If more that one match is produced a split or a merge has occurred depending on the match direction.

6.6 Comparison of the two approaches

Figure 6.13 shows an example of a successful line segment analysis based on the single image approach. The pig's nose is elevated so far of the ground that the line segment projected on this surface appears slightly below its neighboring baseline. Despite this the algorithm is able to connect the line segment to the right baseline. At the moment where the "nose" line segment is chosen by the algorithm the y-component of the SCV vector is just below the allowed maximum. The intermediate line segment between the baseline and the "nose" line segment makes the resulting SCV vector a little less that it would have been had the intermediate line segment not been there. Thus the algorithm is able to handle objects that are so high that the projected line segments appear below neighboring baselines as long as this is not in connection with large discontinuities. In general the algorithm is not very good at handling large vertical discontinuities. In these cases the algorithm is not very robust - with similar images of the same object it failed to make a complete analysis (meaning that some of the line segments where not assigned to a line).

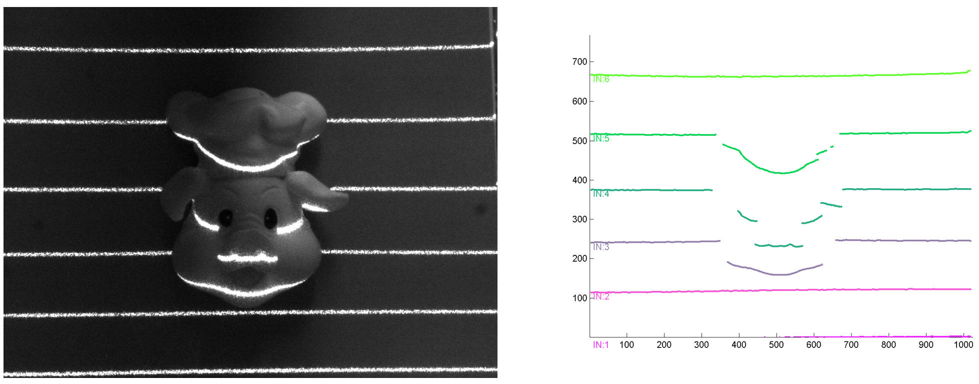


Figure 6.13: Successful line segment correspondence based on a single image. The object is a fridge magnet representing a pig. (left) The acquired image. (right) The result of the line segment analysis – line segments with the same color belong to the same line.

If these discontinuities are larger than the distance between consecutive baselines the algorithm will surely fail, since the **SCV**$_{y}$max value is chosen such that the algorithm can't connect two neighboring baselines directly.

The sequence based algorithm has as completely different way of handling discontinuities. The "jumping" line segment can't be directly connected with there corresponding baselines, but this can however be done indirectly as described earlier. How far the lines jump isn't accentually an issue, but it is however required that the corresponding baseline is present in the image. The sequence based analyses is by far the most powerful of the two algorithms. It is very robust and is able to handle much more complicated objects than the single frame based algorithm. This is mainly due to its ability to track line segments across frames. However it does require a substantial number of frames to sort out the correspondence issues. It is not possible to state exactly how many frames are required to solve the correspondence issues but in general the more frames the better the change of success.

Object class	**Registered line segments**	**Coverage**	**Single image correspondence algorithm success**	**Image sequence correspondence algorithm success**
Low objects with moderate or steep vertical inclinations.	Some broken lines. Elevated line segments appear close to the corresponding baseline.	Nearly full	yes	yes
Medium size objects with moderate or steep vertical inclinations.	More broken lines. Elevated line segments appear further away from the corresponding baseline.	Partial – a small part of the object is not scanned due to shadowing.	yes	yes
High objects with slow vertical variations.	Mostly continuous line segments. Elevated line segments may appear below the neighboring baseline.	Partial - a more significant part of the object is not scanned due to shadowing.	yes	yes
High objects with steep vertical inclinations.	More individual line segments. Elevated line segments may appear below the neighboring baseline.	limited	no – the large distance between corresponding line segment make the algorithm fail.	yes – provided that splitting and merging of lines occur within the cameras field of view.
Very high objects.		very limited	no	no

Table 6.4: Objects classified based on their height and rate of vertical variation and the two algorithms ability to analyze them.

The two line segment correspondence algorithms are capable of analyzing a number of objects successfully while for others the algorithm partially or fully fails to complete the analysis. Weather a given object can be successfully analyzed or not mainly depends on the height and the rate of the vertical variation of the object. Table 6.4 shows a classification of objects based on these features. In general both algorithms are most happy about low object with slow vertical variations. For such objects the single image based algorithm actually has an advantage simply because it can solve the correspondence problem on a single image basis. The sequence based algorithm does need a sequence of a certain length in order for the correspondence issues to be resolved. The sequence based algorithm does however have an advantage when it comes to objects with more complex geometry.

6.7 Line Identification

The two correspondence algorithms above sort out the correspondence of line segments and ensure that all segments originating from the same line are given the same number. But this number is just a relative number it doesn't tell us which of the fifteen laser lines we are dealing with.

Below we see an illustration of what a single frame captured by the camera might look like. A box like object is placed under the camera and a number of lines are visible on the ground and on the box. In this particular case we have seven lines within the cameras field of view. Before we can retrieve any kind of height information we must figure out which of the 15 lines we are dealing with. Since we only see about half the lines the laser is emitting at any one time this can pose a problem.

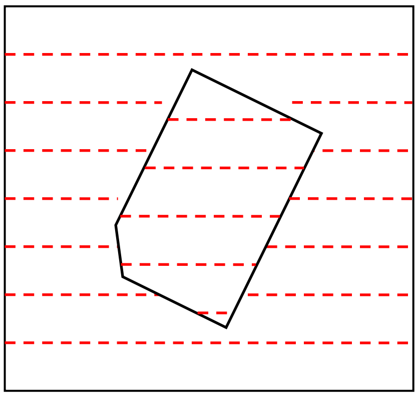


Figure 6.14: Illustration of laser lines projected onto an object. The cameras field of view only allows it to half the lines emitted by the laser.

The only thing that varies from line to line is the angle of incidence so it is obviously this fact we must exploit to identify the lines. One thing that comes to mind is that each line moves at different speed through the series of images due to their different incidence angle. (Lines with steeper incident angle moves slower.)

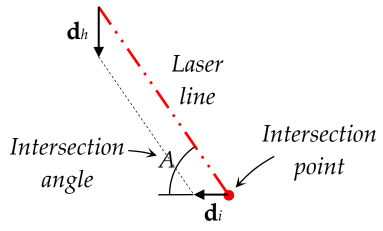


Figure 6.15: Illustration of the laser line, the intersection point on flat horizontal surface and the corresponding angle of incidence.

Figure 6.15 shows an illustration of the laser line and the intersection point on a horizontal surface. The vertical speed of the laser is illustrated by the vector $\mathbf{d}_h$ and the corresponding horizontal speed of the intersection point by the vector $\mathbf{d}_i$. The relationship between $\mathbf{d}_h$ and $\mathbf{d}_i$ is given by (6.7).

$$|\mathbf{d}_h| = \tan(A) \cdot |\mathbf{d}_i| \Leftrightarrow |\mathbf{d}_i| = \frac{\tan(A)}{|\mathbf{d}_h|} \tag{6.7}$$

The approximate incident angles of the central laser lines are { 55°, 57°, 59°, 61°, 63°, 65° }. The corresponding normalized speed of the intersection points, defined in (6.8), are {0.70, 0.64, 0.60, 0.55, 0.50, 0.46}. The speed of one line relative to the speed of the subsequent line gives the following sequence of numbers {1.07, 1.08, 1.08, 1.08, 1.09}. Thus, given the geometry in this setup the difference in speed of two subsequent lines is about 8%, which should be detectable. However the camera doesn't see the horizontal movement of the lines directly. What the camera does see is a combination of the horizontal movement of the lines and the vertical movement of the camera and laser. As a result the observed speed of a line varies depending on where in the image the line is located. Figure 6.16 illustrates this phenomenon.

$$\frac{|\mathbf{d}_h|}{|\mathbf{d}_i|} = \tan(A) \tag{6.8}$$

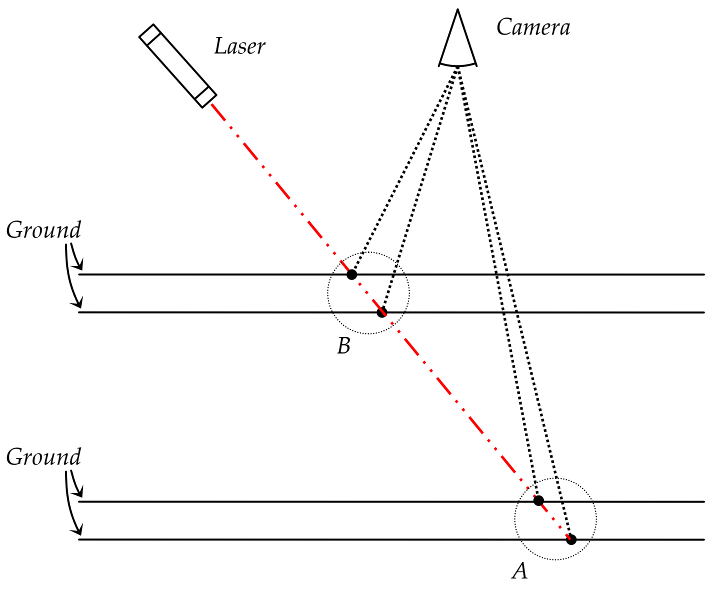


Figure 6.16: Illustration of how the speed of the line observed by the camera changes relative to the lines position within the images.

The horizontal movement of the lines intersection with the ground is the same at both position *A* and *B* but the movement observed by the camera at position *A* is much less than the movement observed at position *B*. Due to this phenomena it is not possible to identify the lines based on observed movement. The problem is that we can't obtain 3D information before we identify the lines and we can't identify the lines unless we can extract 3D information related to the intersections. However we do only have a limited number of the lines to choose from. So, given two line positions at two reasonably spaced frames we can make a 3D reconstruction of these two points of based on all the possible laser lines. Naturally only one if these 3D reconstructions will be true and the others will be false since they are based on a wrong angle of incidence. Now comes the big question – can we tell the true reconstruction from the false? With no further assumption then no, but if we know roughly how far the camera and laser have moved between the two frames in question we then have something to compare against the reconstructed geometry.

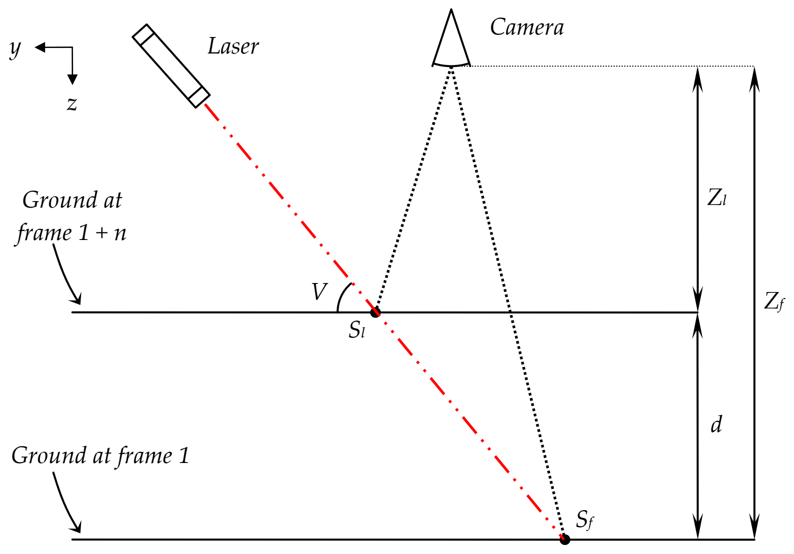


Figure 6.17: Reconstruction of 3D points.

Figure 6.17 illustrates the situation – for a given line observed through a sequence of images we calculate the intersection point S with the ground at the first and last frame (S_f and S_l). From these Z_f and Z_l can be derived which in turn gives d_c. Since we don't know which line the observed line represent, we repeat the calculation of d_c for each of the lines described by the model. Thus d_c becomes a vector $\mathbf{d}_c$ with one true d_c value and k-1 false values, where k is the number of surfaces in the model. From this vector the predicted heights, which is based on a assumption about how far the laser and camera travels between consecutive frames, is subtracted.

observed line number	first frame	last frame	Model surface number					
			1	2	3	4	5	6
1	1	8	0.7730	2.1358	4.1335	6.8297	9.9184	14.9338
2	1	38	2.2669	8.8175	18.3020	30.9056	45.1015	67.6789
3	1	74	-9.0626	1.1734	15.8538	35.1365	56.5866	90.1999
4	1	103	-25.5386	-13.8655	2.6976	24.1687	47.7182	84.0032
5	1	103	-35.8226	-26.4954	-13.4974	2.9805	20.6214	47.0205
6	4	103	-43.9026	-36.6328	-26.6422	-14.1914	-1.1032	18.0625
7	28	103	-39.4033	-34.8840	-28.7232	-21.1205	-13.2126	-1.7763
8	56	103	-27.4135	-25.0102	-21.7558	-17.7722	-13.6641	-7.7821
9	88	103	-10.5443	-10.0135	-9.2988	-8.4297	-7.5397	-6.2755

Table 6.5 Difference between the predicted and calculated vertical movement of the laser in millimeters. The gray shaded cells represent the smallest difference in each row.

Table 6.5 show the $\mathbf{d}_c$ vector for all the observed lines put together in a matrix where the dc vector constitutes the rows. For each $\mathbf{d}_c$ vector the smallest element is marked by gray shaded background. Clearly a pattern emerges, we see that the line labeled 2 corresponds to line 1 of the model, 3 to 2 and so on. Thus using this method the observed lines can be linked to the lines of the model.

6.8 Summary

In this section two methods for solving the line segment correspondence problem have been proposed. Both algorithms are able to solve the correspondence problem for a large amount of objects with simple geometry. The single image based algorithm works best on low objects with simple geometry while it becomes unstable for objects with steep vertical inclinations. The sequence based algorithm is robust and can handle object with more complex geometry. It does however need a long sequence of frames to solve the correspondence problem.
The line identification method presented gives a way of linking the observed lines to the physical lines emitted by the laser.

7 3D Reconstruction

7.1 Introduction

3D scanning of objects is found in numerous applications and the objects being scanned vary from tiny teeth to large statues and even buildings. The design of the scanner naturally reflects the nature of the intended application. The two most widespread techniques of 3D reconstruction are laser triangulation and structured light. The latter is often used in connection with stereo vision – a third technique for 3D reconstruction. The resulting 3D reconstructions produced by this technique are often referred to as depth maps. The underlying geometry and mathematics of these three techniques are basically the same. A point is seen from two or more viewpoints and coordinates of the point is calculated by intersection of the line of sight corresponding to each viewpoint. Scharstein et al. [21] uses structured light in combination with stereo vision to generat depth maps. In [11] Gao et al. uses structured light to reconstruct the surface of a bioprosthetic heart valve leaflet. Bernardini et al. [2] present a system based structured light to build a digital model of Michelangelo's *Florentine Pietà*. Tsalakanidoua et al. [23] uses color coded structured light for real-time acquisition of depth maps. Sadlo et al. [20] presents a 3D reconstruction system using gray-coded structured light. Objects are placed on a turn-able hence resembling type D of Figure 3.4 page 20. In [13] Hamette et al. presents a method for 3D sampling of a human hand using laser triangulation with a multipoint laser. David et al. [7] presents a system where an arrangement of mirrors produces two virtual viewpoints using a single camera. The system has been designed with emphasis on low calibration complexity.
In the following section the process of 3D reconstruction is described included calibration of the camera and laser. Initially the sheets-of-light emitted by the multibeam are assumed to flat planes. As this turns out to be a pour approximation a more advanced model is applied.

7.2 Overview

In the 3D reconstruction step the 2D data obtained earlier is converted into 3D points. The 2D data that we have actually corresponds to physical positions on the CCD chip of the camera. Such a position corresponds to a line of sight from the CCD chip and through the lens (pinhole) of the camera. An intersection between this line and the laser plane yields the 3D point that we are pursuing. In order to calculate this intersection two things are needed: Firstly, in order to obtain the line of sight corresponding to a physical position on the CCD chip a camera calibration must be performed. Secondly, a mathematical description of the laser plane in the camera coordinate frame must be available.

7.3 Camera calibration

The camera calibration is performed via a calibration tool called: Camera Calibration Toolbox for Matlab, see [3]. The objective of the calibration is to estimate the internal parameters which include: the focal length, the principal point, the skew coefficient and the distortion coefficients.

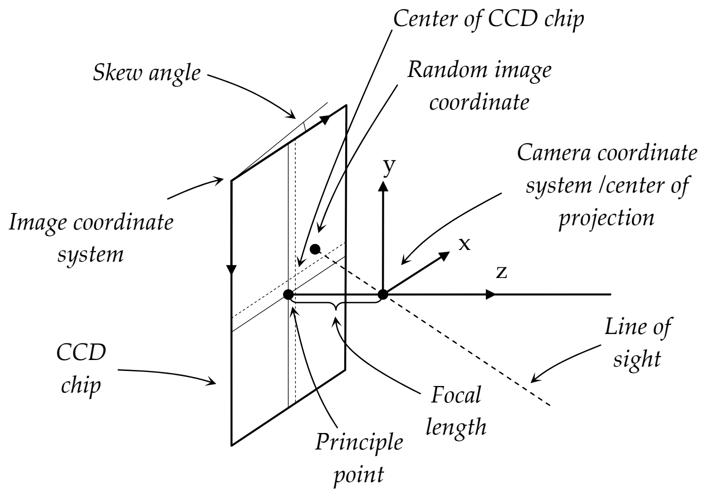


Figure 7.1: Illustration of some of the cameras internal parameters, namely the focal length, the principal point and the skew coefficient. The focal length is the distance from the CCD chip to the projection center of the lens. The principal point is the point on the CCD chip directly beneath the center of projection which may not exactly coincide with the center of the CCD chip. The skew angle describes the angle between the x- and y-axis of the CCD chip.

Figure 7.1 shows an illustration of the focal length, the principal point and the skew coefficient. The distortion coefficients describe how the lens differs from the pinhole model.
The normalized pinhole image projection is described by $\mathbf{x}_n$:

$$\mathbf{x}_n = \begin{bmatrix} X_c / Z_c \\ Y_c / Z_c \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{7.1}$$

The corrections made to this point in order to model the radial and tangential distortions are given in (7.2).

$$\mathbf{x}_d = (1 + kc_1 r^2 + kc_2 r^4 + kc_5 r^6)\mathbf{x}_n + \begin{bmatrix} 2kc_3 xy + kc_4(r^2 + 2x^2) \\ kc_3(r^2 + 2y^2) + 2kc_4 xy \end{bmatrix} \tag{7.2}$$

where $r = \sqrt{x^2 + y^2}$ and kc_1 to kc_5 are the radial and tangential distortion coefficients.

These corrections are not strictly necessary but they do provide a better estimation of the line of sight corresponding to a given pixel coordinate.

7.3.1 Data set for optimization

The camera calibration toolbox estimates the parameter via an optimization. For this purpose a data set is required. The toolbox algorithm is designed to extract points from images of checkerboards, meaning that once the corner points have been manually annotated the algorithm will extract the interior points automatically. This significantly speeds up the calibration process as we only need to annotate a tiny fraction of the final set of points. However a series of images of the checkerboard seen from different angle and distances are required for a successful calibration.

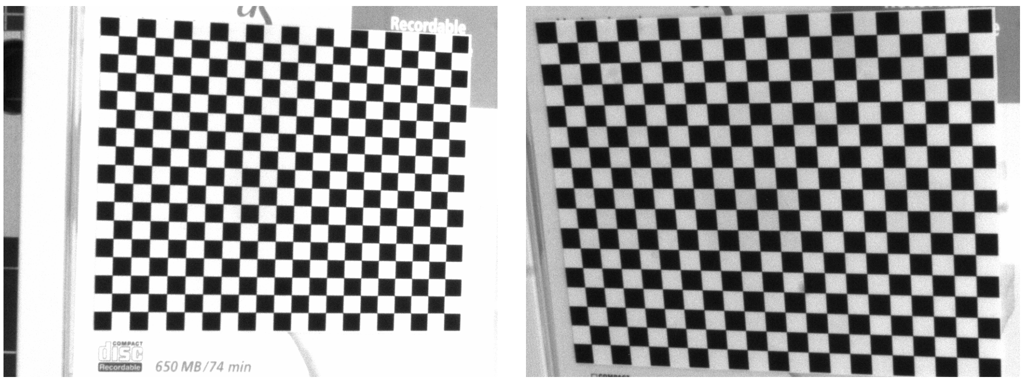
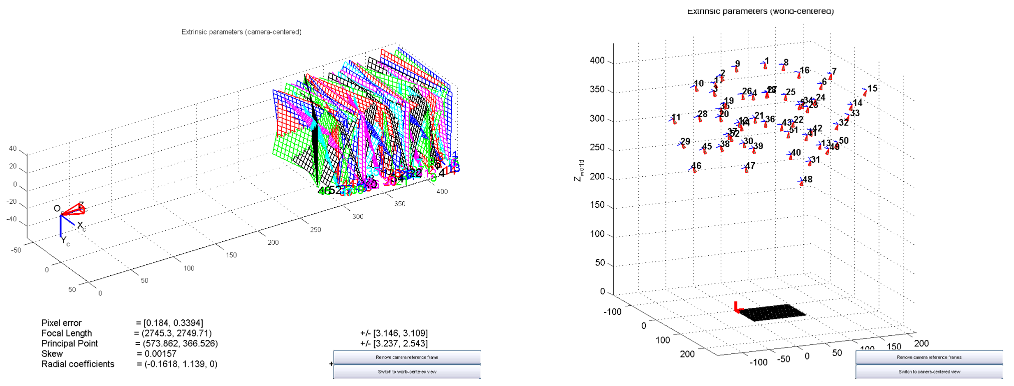


Figure 7.2: Checkerboard images for camera calibration. The checkerboard are printed on a 600 dpi printer a attached to a cd-cover using a glue stick.

Figure 7.2 shows two checkerboard images used for the camera calibration. In total 52 images have been used for the calibration, see [30]. The checkerboard has systematically been placed at different angles and distances relative to the camera. This is important since it ensures that the obtained points are well distributed in 3D space. If the data points are poorly distributed the optimization problem will yield an ill-posed problem and as a consequence the camera parameters are badly estimated. Figure 7.3 shows the distribution of the checker boards in space – as seen the checkerboards are well distributed within the limitations of the cameras field of view. The checker size is also an important parameter. It's advantageous to have as many checkers as possible as this will yield more data point for the algorithm to extract, and more points yields a better optimization. As a consequence the checker size will have to be as small as possible. However at a certain point the algorithm will experience difficulties estimating the corners of the checkers so a reasonable compromise must be made. The chosen checker size is believed to be such a compromise. Initially the larger checkers were used but that yielded a less successful calibration.


Extrinsic parameters (camera-centered)
Extrinsic parameters (world-centered)
Remove camera reference frame
Switch to world-centered view
Remove camera reference frames
Switch to camera-centered view

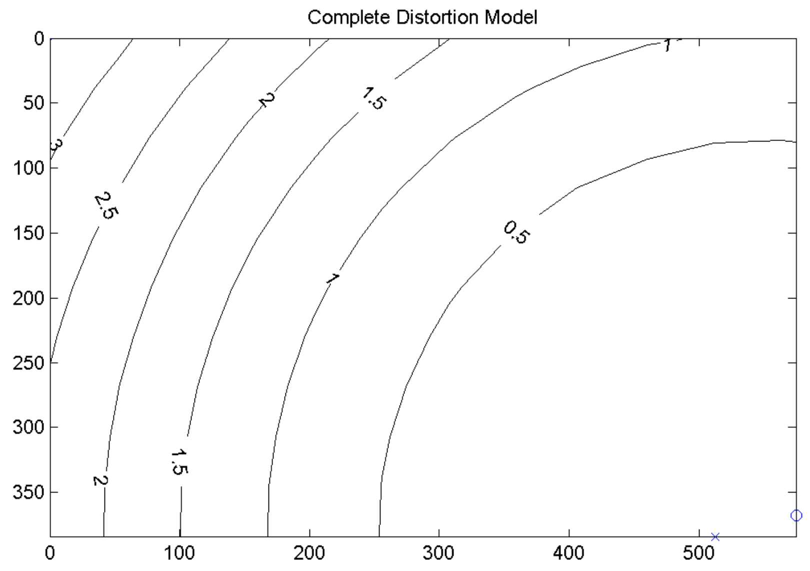
```
Pixel error            = [0.184, 0.3394]
Focal Length           = (2745.3, 2749.71)        +/- [3.146, 3.109]
Principal Point        = (573.862, 366.526)       +/- [3.237, 2.543]
Skew                   = 0.00157
Radial coefficients    = (-0.1618, 1.139, 0)
```

Figure 7.3: The extracted checkerboard data used for the calibration. (left) Camera centered view. (right) World centered view.

The algorithm presumes that the checkerboard is entirely flat and it is therefore vital that it actually is. To begin with the checkerboard was attached using sticky tape but it turned out that the checkerboard wouldn't lie flat using this technique. Instead a glue stick was used to attach the checkerboard which yielded a much better result.

7.3.2 Calibration results

Figure 7.4 shows the resulting camera calibration. The level curves show the magnitude of the distortion. We can observe that the worst possible distortion is a little more that 3 pixels in magnitude. Both the focal length and the principal point are estimated within an uncertainty of +/- 3 pixels approximately. The uncertainties are specified as three times there standard deviation.



```
Pixel error              = [0.184, 0.3394]
Focal Length             = (2745.3, 2749.71)                    +/- [3.146, 3.109]
Principal Point          = (573.862, 366.526)                   +/- [3.237, 2.543]
Skew                     = 0.00157                              +/- 9.649e-005
Radial coefficients      = (-0.1618, 1.139, 0)                  +/- [0.008797, 0.2068, 0]
Tangential coefficients  = (0.0003149, -0.001229)               +/- [0.0002629, 0.0003131]
```

Figure 7.4: The calibration result. (above) The level curves show the magnitude of the distortion or correction throughout the image. The blue cross specifies the center of the image while the blue circle specifies the principal point. (below) The model parameters along with there uncertainties.

Figure 7.5 shows the distortion model separated into the tangential and radial components respectively. We see that the radial component is responsible for the majority of the distortion.

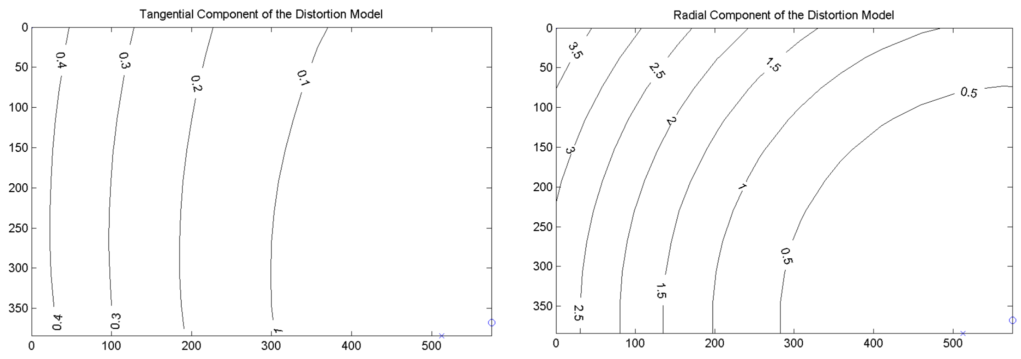


Figure 7.5: Distortion model separated into the radial and tangential components. (left) Tangential component. (right) Radial component.

The resulting parameters are plugged into (7.2) in order to apply the corrections. The normalized pinhole projection is obtained by dividing the found focal length into the current image coordinates.

7.4 A mathematical description of the laser plane

The most straightforward way to model the light planes emitted by the laser is to assume that they a flat planes. As it shall later be evident this assumption is far from true when using a multi beam laser but in this setup it will yield a fairly good approximation. Furthermore we assume that all the planes are orthogonal to the yz-plane, thus projected onto the yz-plane the intersection curves will merely be appear as points.
In order to establish a reference between the camera and laser we record a sequence of images where the laser is projected onto a flat surface at a series of known distances to the camera. At each distance the location of the laser lines are extracted. Since we know the distance to the camera at each image all the extracted locations can be converted to 3D points directly.

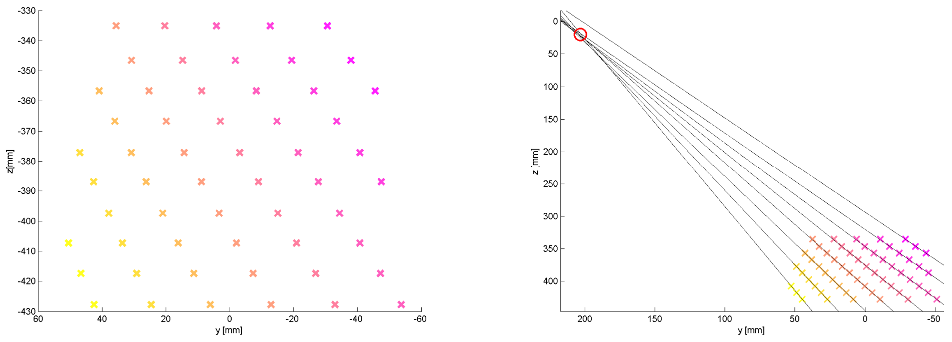


Figure 7.6: (left) Extracted laser plane intersections at known heights. Same color illustrates same line. (right) The extracted intersections are used to estimate the location and angle of the laser planes.

Figure 7.6 shows the laser plane intersections extracted from the image sequence. The height difference between each image is approximately 10 mm. Only three of the eight lines are tracked throughout the entire sequence. Based on the extracted intersections the angle and location of each laser plane can be estimated simply by fitting a line through the extracted points. This is done on an individual basis so there is no guarantee these lines will intersect at the same point. However, as we can see most of the lines do actually intersect at roughly the same point. The top most line is slightly off, but it is only based on three intersection points so that is understandable. The fitted lines give us an estimate of the laser position plus a mathematical description of each of the emitted laser planes in terms of the slope of each plane. Based on this it is now actually possible to reconstruct data into 3D points.

7.4.1 Reconstruction of 3D points

Reconstruction the 3D points are done by calculating the intersection between the line of sight from the CCD chip through the cameras lens and the laser plane.

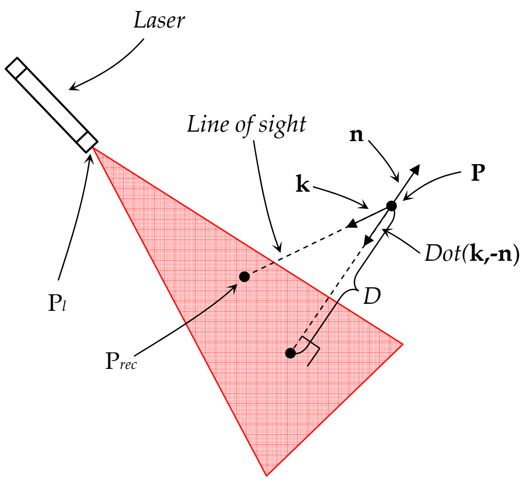


Figure 7.7: Illustration of the intersection between the line of sight and the light plane emitted by the laser.

The line of sight from the camera is given by (7.3):

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{P} + t \begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix} \tag{7.3}$$

Since the projection center of the camera is used as a reference for the coordinate system: **P** = **0**. The coordinates (k_x,k_y) corresponds to the image coordinates on the CCD chip relative to the principal point. Last k_z is the focal length measured in pixels.

The plane is defined by the point $\mathbf{P}_l = (x_l\ y_l\ z_l)^T$ and vectors **p** and **q** which lie in the plane (not shown in the illustration). The normal to the plane **n** is calculated as the cross product of the vectors **p** and **q** (it is required that **p** and **q** are chosen such that $\mathbf{p} \neq \pm\mathbf{q}$). The plane is defined orthogonal to the yz-plane – therefore **p** can be chosen $\mathbf{p} = (1\ 0\ 0)^T$ and **q** is determined by the slope of the plane.

The distance from the point P to a plane can be calculates as (7.4):

$$D = \frac{\left|\mathbf{n}^T \cdot \mathbf{P} + d\right|}{\sqrt{\mathbf{n}^T \cdot \mathbf{n}}}, \text{ where } d = -n_x x_l - n_y y_l - n_z z_l \tag{7.4}$$

The reconstructed point $\mathbf{P}_{rec}$ can then be calculated as:

$$\mathbf{P}_{rec} = \mathbf{k} \cdot \frac{D}{\left|\mathbf{k} \bullet \mathbf{n}\right|} \tag{7.5}$$

7.4.2 Reference height

All the obtained 2D image data can be converted to 3D points using the above approach. However as the 2D image data is acquired the camera continuously changes distance to the object. As the points are reconstructed relative to the coordinate frame of the camera this fact gives the undesirable effect of stretching the scanned object. In order to counter this effect points from different frames aligned to a mutual height. Thus for each frame a reference height must be determined and points extracted at a given frame must be reconstructed relative to the reference height of that particular frame. The reference height is calculated based on the baseline part of the projected line. It is assumed that the object doesn't fill the entire image and therefore a small part of the line is always guarantied to be projected onto the ground. The height of the ground at a particular frame is simply subtracted from all other points extracted at this frame.

7.4.3 Results

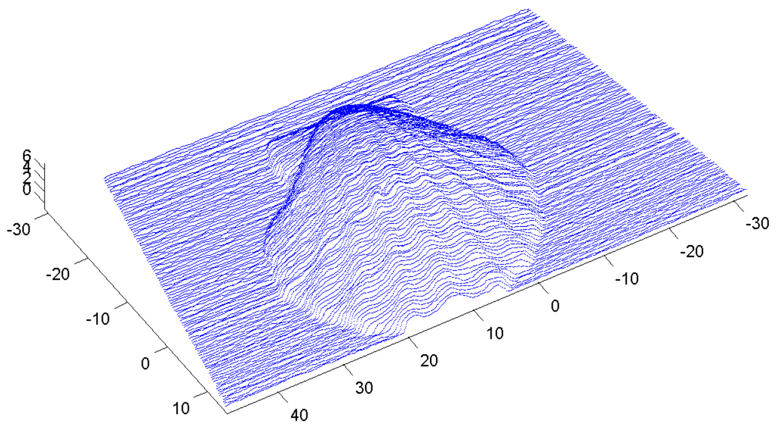


Figure 7.8: Reconstructed shell. The 3D points are all based on the same line tracked over a hundred frames. Notice that the scanned area is not rectangular shaped but shaped as a trapezoid. For this reason the ground appears to be lifted toward the backside but this merely a perspective illusion. It actually does lie in the xy-plane.

Figure 7.8 shows the reconstructed shell based on the data obtained from a single line over a hundred frames. Each line contains approximately a thousand points thus the point cloud consist of approximately 100.000 points. The scanned area span roughly 80 mm in the direction along the lines and about half in the direction orthogonal to the lines. These directions correspond to the x- and y-axis respectively, in the camera coordinate system. Consequently the point density is roughly five times higher along the x-axis than along the y-axis. If we insist on having the same point density along the y-axis it would require five times as many frames.

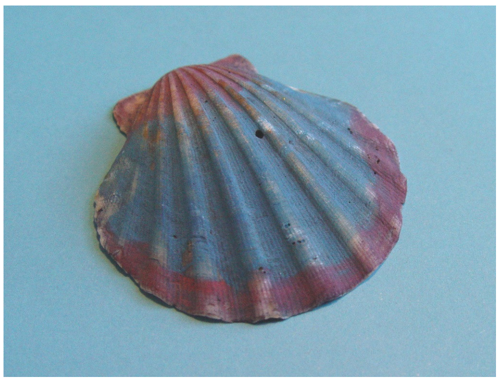


Figure 7.9: Image of the scanned shell seen from approximately the same angle.

Figure 7.9 shows a picture of the scanned shell for comparison. The reconstructed shell demonstrates that the system works – however we are only utilizing information form a single line. Clearly the next objective is to combine information from several lines.

7.5 Applying a more advanced model

So far we have assumed that the light emitted from the laser could be described as flat planar surfaces. This assumption is not entirely correct – while it is true for the center plane all other planes actually bend away from the center. Figure 7.10 shows an image of this phenomenon.

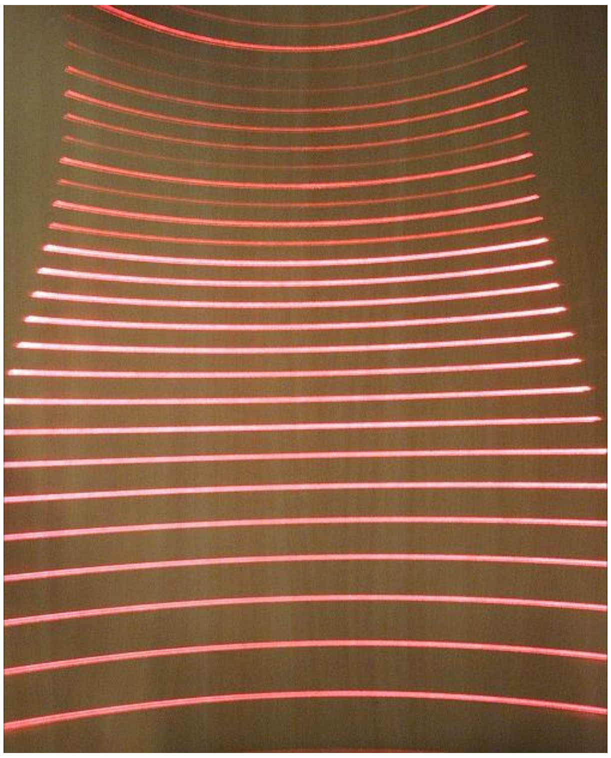


Figure 7.10: Image of the emitted laser planes projected on to a flat surface slightly angled. The laser creates 15 planes but some additional weaker planes also occur. The "true" planes are the 15 first starting from the bottom. Only the center plane is completely flat – all other planes bend away from the center.

The obvious problem here is that if the light planes are modeled as flat planar surfaces a considerable error will be introduced. As a consequence the mathematical description of the light planes must be revised. The intersecting curves look like parabolas, so it's natural to model them as such. If we imagine the flat surface which the light planes are projected onto is moved towards or away from the camera, the intersecting parabolas would be scaled proportionally to the translation of the intersecting surface. Thus the model should reflect this. Another way to describe this is that the model must reflect the fact that the light travels in straight lines. Thus if a random point on the surface is chosen and this point is connected via a straight line to the center of propagation, the modeled surface should coincide with this line.

$$y = f(x,z) = \frac{ax^2}{z} + bz \tag{7.6}$$

The surface given in (7.6) has these properties – however this is given in the lasers coordinate frame. In (7.7) the surface is modified to suit the cameras coordinate frame.

$$y - y_L = \frac{a(x-x_L)^2}{(z-z_L)} + b(z-z_L) \Leftrightarrow y = \frac{a(x-x_L)^2}{(z-z_L)} + b(z-z_L) + y_L \tag{7.7}$$

$$\text{where } (x_L, y_L, z_L) \text{ is the position of the laser.}$$

It should be mentioned that the *a*- and *b*-parameters in (7.6) does not correspond directly to the same parameters of (7.7). The reason is that only the translation of the laser relative to the camera is modeled by (7.7) – the rotation is modeled by a change of the parameters *a* and *b*.
The above surface is a model of a single laser plane. The model contains five parameters x_L, y_L, z_L, *a*, and *b*, which all must be adjusted in order for the model to fit the actual surface. For this purpose the Levenberg-Marquardt optimization algorithm is used. This requires that the optimization problem is stated as a function that returns the residuals along with the gradient with respect to the model parameter **m**:

$$r = y - M(\mathbf{m}, x, z) = y - \frac{a(x-x_L)^2}{(z-z_L)} - b(z-z_L) + y_L$$

$$J(\mathbf{m}) = \begin{bmatrix} \dfrac{2a(x-x_L)}{z-z_L} \\ 1 \\ -a\dfrac{(x-x_L)^2}{(z-z_L)^2} + b \\ -\dfrac{(x-x_L)^2}{(z-z_L)} \\ -z+z_L \end{bmatrix} \tag{7.8}$$

$$\text{where } \mathbf{m} = \begin{bmatrix} x_L & y_L & z_L & a & b \end{bmatrix}$$

In (7.8) this function is given. The residuals calculated in (7.8) do not correspond to the Euclidian distance from the data points to the surface, but this distance projected into the yx-plane. This is merely chosen for simplicity. The surfaces might as well be optimized together as apposed individually. All of the surfaces do have one thing in common, namely the position of the laser. This fact can be utilized by optimizing the surfaces together. Actually it would not be possible to estimate the position of the laser using the information from a single surface alone. Given the still relatively flat nature of the surface estimating the position of the laser would yield an ill-posed optimization problem. Using the information from multiple surfaces however the position of the laser is pined down more precisely. The x-coordinate of the position remains difficult to estimate.

We have obtained data from a total of eight laser planes, thus the combined model contain 19 parameters: x_L, y_L, z_L, $a_1 \ldots a_8$, and $b_1 \ldots b_8$.

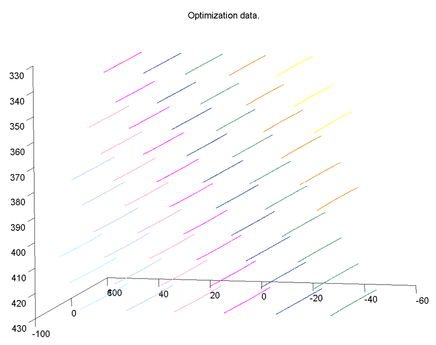


Figure 7.11: Extracted laser plane intersection at known heights. The same color illustrates the same line.

Figure 7.11 shows the optimization data which consist of line data extracted at ten different heights 10 millimeters apart. This figure corresponds to Figure 7.6, except here the entire line is extracted whereas earlier only data from the yz-plane was used. The lines don't bend as much as one should expect keeping Figure 7.10 in mind, but this is because the cameras field of view limits the portion of the laser line seen to about a quarter of the entire line where the bending is not yet so distinct. Moreover it's actually difficult to see the lines bend on a fixed print but it is visible if you take a close look.
Before the optimization process can start a set of initial parameters must be estimated. This process has been carried out manually changing the parameters while comparing the model and data.

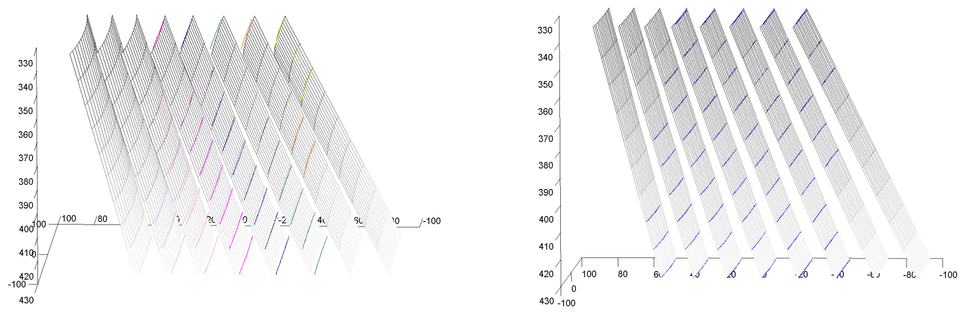


Figure 7.12: (left) The optimization data (colored lines) and the initial model (gray surfaces). (right) The optimization data (colored lines) and the optimized model (gray surfaces).

Figure 7.12 shows the data and model before and after the optimization. Subsequent to optimization the model follows the data much more precisely.

Table 7.1 shows a list of the model parameters prior and subsequent to the optimization.

Parameter Name	Initial Parameter			Optimized Parameter		
x_L	0			37.8824		
y_L	203.0000			206.7577		
z_L	0			14.1028		
a_1 a_2 a_3	-0.1000	-0.0700	-0.0300	0.0043	0.0071	0.0098
a_4 a_5 a_6	0.0100	0.0600	0.1200	0.0146	0.0181	0.0200
a_7 a_8	0.1500	0.2000		0.0224	0.0242	
b_1 b_2 b_3	-0.6900	-0.6400	-0.5900	-0.7374	-0.6815	-0.6290
b_4 b_5 b_6	-0.5400	-0.4950	-0.4500	-0.5789	-0.5309	-0.4848
b_7 b_8	-0.4100	-0.3700		-0.4403	-0.3972	

Table 7.1: Initial and optimized model parameters. The x_L, y_L and z_L parameters are measured in millimeters.

One of the best methods for evaluation an optimization process is to have a look at the resulting residuals.

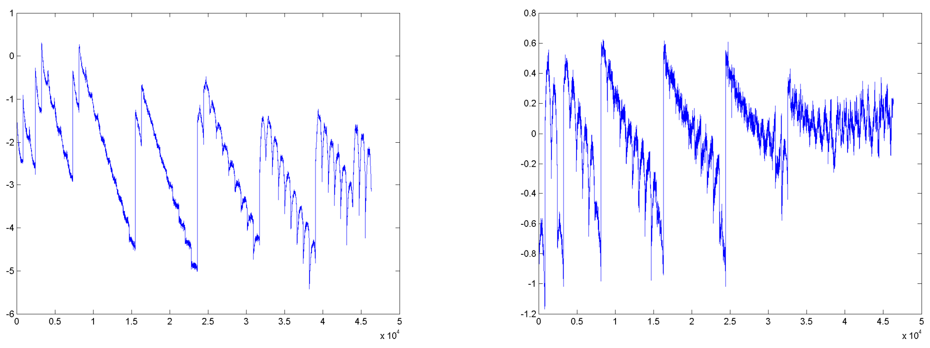


Figure 7.13: (left) Residuals before optimization. (right) Residuals after optimization. The optimization process has significantly reduces the residuals and the maximum deviation has been reduced from about 5 millimeters to roughly 1 millimeter.

Figure 7.13 shows the residuals before and after the optimization. Clearly the optimized model fits the data much better – thus the optimization works. The average distance *AD* from the data to model, defined in (7.9), has been reduced from 2.41 to 0.22 millimeters.

$$AD = mean(|F|), \text{ where } F \text{ is the residual vector.} \qquad (7.9)$$

However the resulting residuals are far from noise like which implies that the model is incapable of presuming the exact shape of the data. A closer examination of the residuals reveals that a periodic series of residual are significantly larger than the rest. It turns out apparently, that the height information related to one of the ten known heights is erroneous.

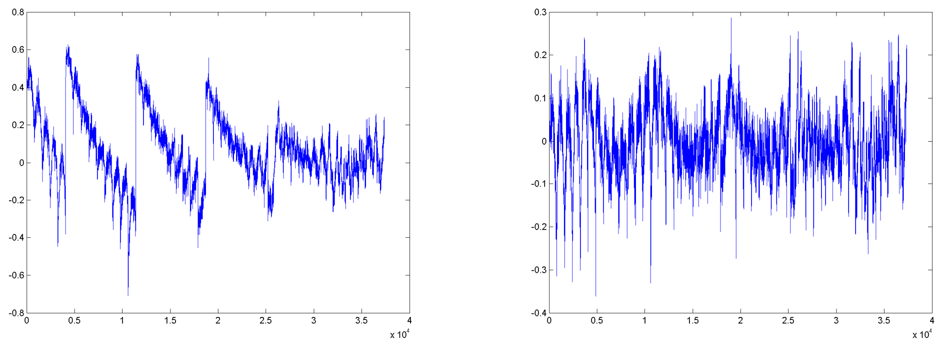


Figure 7.14: Optimization with erroneous data removed (left) Residuals prior to optimization. The optimized parameters from the previous optimization is used as the initial guess, hence the relatively small residuals. (right) Residuals subsequent to optimization. While the residuals still show some trends they are much more noise like which indicate a more successful optimization.

Figure 7.14 shows the residual prior and subsequent to a new optimization where the erroneous data have been removed. The resulting residuals have been further reduced and the severe trends observed earlier have also been considerable decreased. It is clear now that the erroneous data sabotaged the previous optimization the some degree. Normally erroneous data can be handled by using a robust optimization technique such as the huber estimator, see [19]. This technique is preferable if the erroneous data is scattered throughout the data. In this case however the erroneous data were grouped in sequences and were relatively easy identifiable once noticed. Therefore it was a better solution to remove the faulty data altogether. Also the data from the two outermost surfaces, which were only present in three images, has been removed from the data set. The dataset has been decimated from roughly 44000 points to about 37000 points. The reduced model now contains six surfaces which all are based on data that is present for a minimum of six frames. The average distance from model to data defined in (7.9) is now reduced to 0.062 millimeters and the maximum distance *MD*, defined in (7.10), yields a distance of 0.36 millimeters.

$$MD = \max(|F|), \text{ where } F \text{ is the residual vector.} \tag{7.10}$$

Thus the calibrated model now fits the data to a satisfactory precision.

7.5.1 Reconstruction of 3D points using the advanced model

The reconstruction of 3D points take starting point in (7.7) which describe the surface in the camera coordinate frame. However the calculations are easier transforming the expression to the laser coordinate frame. In (7.11) the camera- and laser coordinate frames are defined as well as the relationship between the two.

$$\left.\begin{array}{l}\hat{x} \quad \hat{y} \quad \hat{z} \rightarrow \text{camera coordinate frame} \\ \mathrm{x} \quad y \quad z \rightarrow \text{laser coordinate frame}\end{array}\right\} \Rightarrow \begin{array}{l} x = \hat{x} - x_L \\ y = \hat{y} - y_L\,, \\ z = \hat{z} - z_L \end{array} \tag{7.11}$$
$$\text{where } x_L, y_L \text{ and } z_L \text{ are the coordinatesof the laser.}$$

Using this (7.7) can be rewritten:

$$\begin{aligned} &S : \hat{y} = a\frac{(\hat{x}-x_L)^2}{(\hat{z}-z_L)} + b(\hat{z}-z_L) + y_L \Leftrightarrow \\ &\hat{y} - y_L = a\frac{(\hat{x}-x_L)^2}{(\hat{z}-z_L)} + b(\hat{z}-z_L) \Leftrightarrow \\ &y = a\frac{x^2}{z} + bz \Leftrightarrow bz^2 + ax - yz = 0 \end{aligned} \tag{7.12}$$

The line of sight from the CCD chip through the pinhole of the camera lens is given by (7.13):

$$\begin{aligned} &L : \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} + t\begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \\ &\text{where:} \\ &P_x = -x_L, P_y = -y_L \text{ and } P_z = z_L \\ &(k_x, k_y) \text{ are the image coordinates relative} \\ &\text{to the principal point.} \\ &\mathrm{k}_z \text{ is the focal length in pixels.} \end{aligned} \tag{7.13}$$

Inserting (7.13) into (7.12) gives:

$$\begin{gathered}
b(P_z+tk_z)^2+a(P_x+tk_x)^2-(P_y+tk_y)(P_z+tk_z)=0 \Leftrightarrow \\
b(P_z{}^2+t^2k_z{}^2+2P_ztk_z)+a(P_x{}^2+t^2k_x{}^2+2P_xtk_x) \\
-P_yP_z-P_ytk_z-tk_yP_z-k_yk_zt^2=0 \Leftrightarrow \\
t^2(bk_z{}^2+ak_x{}^2-k_yk_z)+t(2bP_zk_z+2aP_xk_x-P_yk_z-P_zk_y) \\
+bP_z{}^2+aP_x{}^2-P_yP_z=0
\end{gathered} \tag{7.14}$$

Thus (7.14) given a second order equation in *t* which has the following solutions:

$$\begin{aligned}
&t^2 : bk_z{}^2+ak_x{}^2-k_yk_z=e_2 \\
&t : 2bP_zk_z+2aP_xk_x-P_yk_z-P_zk_y=e_1 \\
&t^0 : bP_z{}^2+aP_x{}^2-P_yP_z=e_0 \\
&t=\frac{1}{2e_2}\left(-e_1\pm\sqrt{e_1{}^2-4e_2e_0}\right)
\end{aligned} \tag{7.15}$$

Since the surface describing the laser planes is second order surface, the intersection point between this and the line of sight yields two solutions. However the working range of the system rules out the one solution leaving us with one single 3D point which is obtained by (7.16):

$$P3D=t\begin{pmatrix}k_x\\k_y\\k_z\end{pmatrix} \tag{7.16}$$

7.6 Results

The more advanced model should provide a more correct reconstruction of objects. This should in turn enable us to combine the information obtained from the different lines.

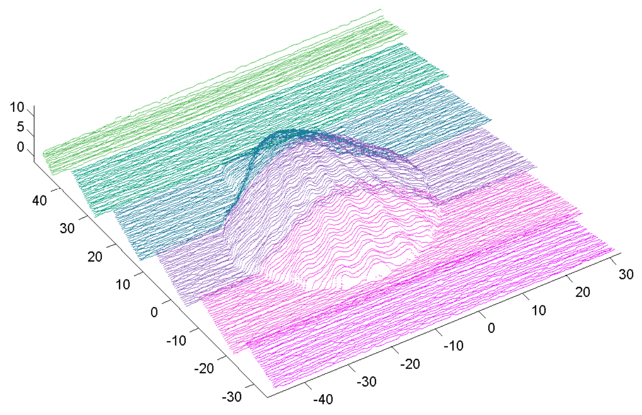


Figure 7.15: Reconstruction of the shell using information obtained from six consecutive lines tracked over 30 frames. The amount of overlap between sections varies as a consequence of the surfaces different incident angels. The axis measures are in millimeters.

Figure 7.15 shows the reconstructed shell based on data from multiple lines. The immediate impression is that the individual sections match very well together. It's possible to resample the data on to regular grid, hereby combining the data from the individual sections. The grid points are calculated as an average of the surrounding points in a suitable distance. As a result the points in the overlapping sections will also be an average between the data obtained from the two consecutive lines. Figure 7.16 shows the resulting surface.

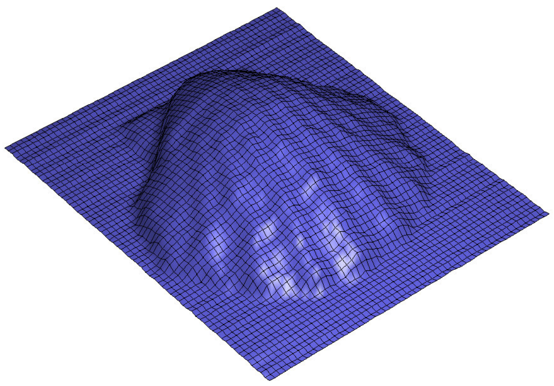


Figure 7.16: Reconstructed shell with data re-sampled to 1 by 1 millimeter grid. Shown as shaded surface with grid overlaid.

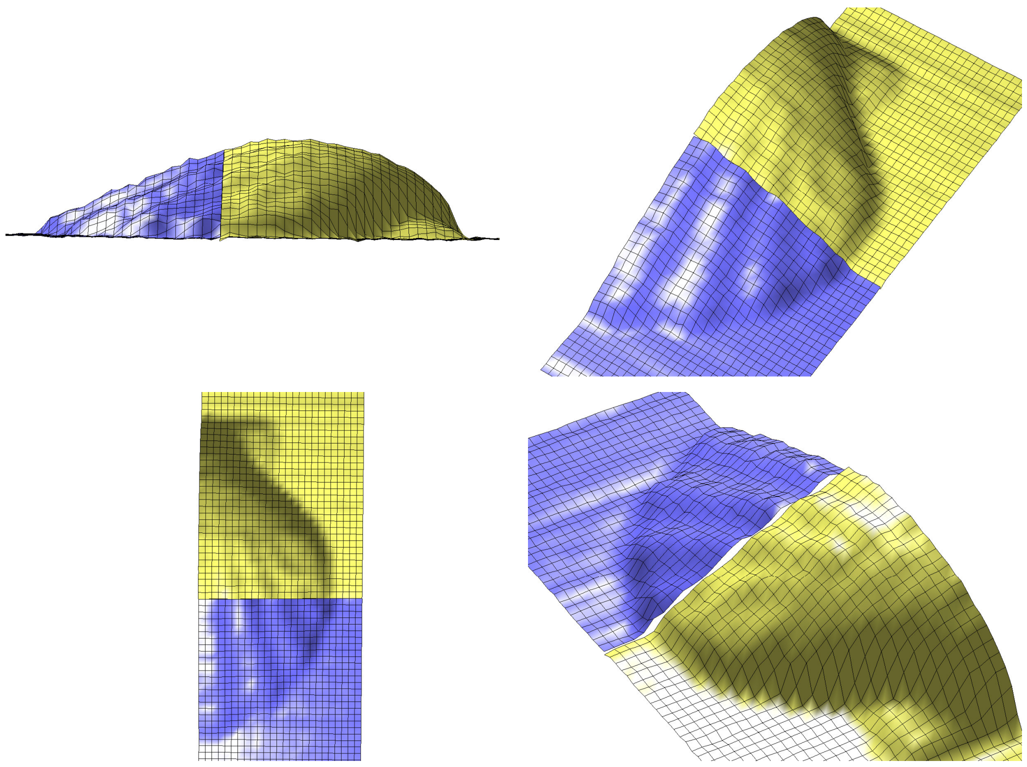


Figure 7.17: Shell reconstruction seen from four different views. The blue and yellow part is based on data from two consecutive lines. The data has been kept separate as it was resampled to a regular grid. The resulting surfaces illustrate how well or bad the data align.

Figure 7.17 illustrates how well the data obtained from two consecutive lines align. Overall the alignment seems fairly good however the blue surface is pushed slightly to the side relative to the yellow surface. This is best seen on the bottom right of the four views. The misalignment has been estimated to be just below a millimeter. This may not sound as much put relative to the size of the object it is slightly more than hoped fore. A plausible explanation for this misalignment is that the estimated x-coordinate of the laser simply doesn't correspond to the physical position. The intersection between the six model surfaces defines the y- and z-coordinates of the position very well. But this is not the case with the x-coordinate. As a result the x-coordinate may wander away from the true position during optimization. In other words the x-coordinate doesn't converge toward the true position or the convergence is very weak and sensitive to noise. A 3D reconstruction based on an erroneous x-coordinate of the laser doesn't show any problems visually as long as the data from a single line only is used. Only when data from several points are used is it revealed that the reconstructed data deviates from the actual physical position.

One possible way to go about this is to estimate the x-coordinate by other means and the simply fix it during the optimization process.

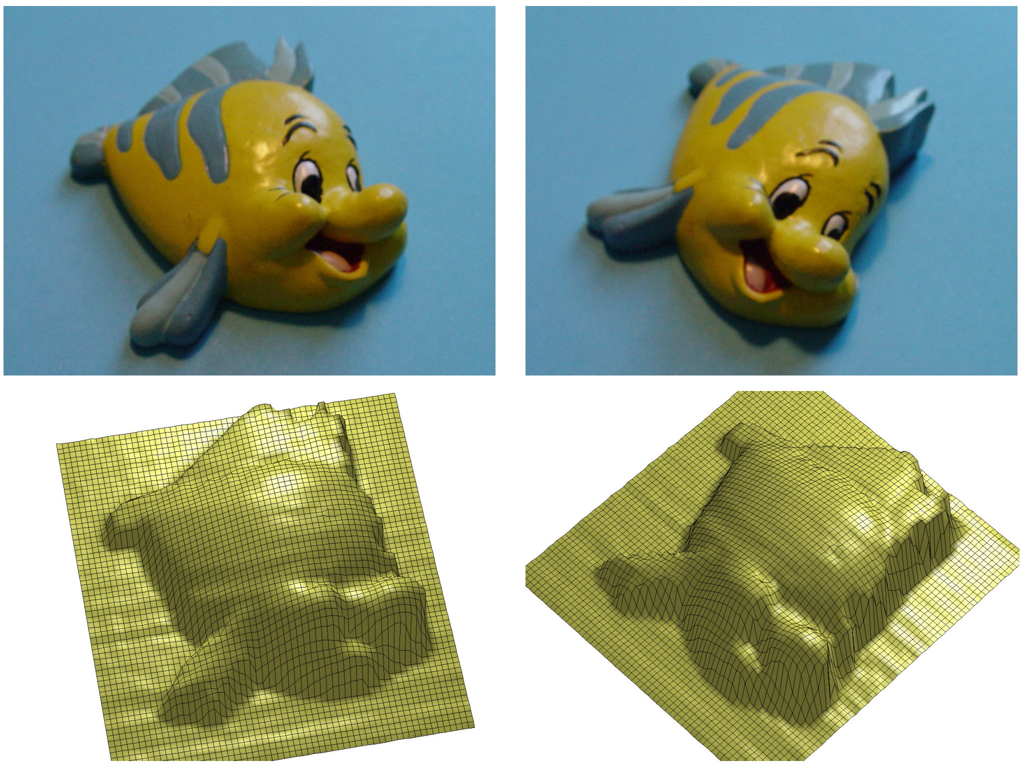


Figure 7.18: 3D reconstruction of a fridge magnet fish. The reconstructed 3D shape is generated from six lines tracked over 30 frames and resampled to a regular grid.

Figure 7.18 shows a fridge magnet fish from two different angles and the reconstructed 3D shape from approximately the same angles. The reconstructed shape is generated from six lines tracked over 30 frames and the raw data is resampled to 1 by 1 mm grid. Notice the right side of the reconstructed fish (in front of the fish eyes) – here we see a very steep vertical surface. On the object this surface actually curves inward but this shape cannot be captured in the scanning process due to the shadow phenomenon described is section 3.2.4. Apart from the artifacts due to shadowing the digital model seems to resemble the actual object very closely. The ground around the fish which is supposed to be entirely flat looks a bit like slightly creased tinfoil. These tiny variations give an idea about the noise level of the estimated line position. It should also be mentioned that an error on the reference height propagate throughout the rest of the line. This fact is also strongly related to the creased tinfoil effect.

7.6.1 Estimation of the precision

The digital models of the shell and fridge magnet fish shows that the laser scanner works intentionally. However it is interesting to know how precise the scanner actually is. There are two things that influence the precision of the scanner. Firstly the line position can be estimated to an accuracy of roughly 0.5 pixels. At a distance of 430 millimeters from the camera, which corresponds to the maximum possible distance between the camera and object, one pixel corresponds to 0.16 millimeters. The steepest of the laser planes have a slope of 2.56, thus an uncertainty of 0.5 pixels results in a height error of 0.20 millimeters. Secondly the mathematical function used to model the laser surface also contributes an error that influences the final precision. At the end of section 7.5 it was found that the maximum horizontal distance between the data and model was 0.36 millimeter. Again multiplying this number with the slope of the steepest laser surface gives a height error of 0.92 millimeters.

7.6.2 Object and Model Comparison

The precision of the system can be evaluated using a very straight forward method. A distinct point on the object is measured using a slide gauge and the resulting height is then compared to the model. For this purpose a short Matlab script that allowed a small patch to be moved around on the surface was used. The script continually displays the height of the patch and using this tool allowed for points on the model to be measured. Despite this it is actually somewhat difficult to define distinct point that can be measured both on the model and object.

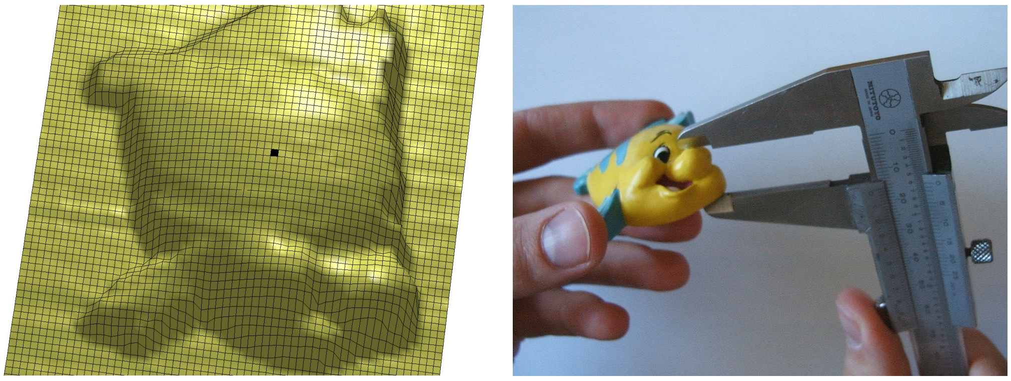


Figure 7.19: (left) The reconstructed fridge magnet fish with a single surface patch marked. The patch is moveable and average height of the patch is continually displayed. (right) Distinct points on the fish is measured using a slide gauge.

Figure 7.19 shows how distinct points on the fridge magnet fish can be measured on the digital model and on the actual object respectively.

Location	Model height	Object height	Difference
“nose”	14.63 mm	15.70 mm	-1.07 mm (-6.82%)
“cheek”	15.54 mm	16.58 mm	-1,04 mm (-6.27%)
“forehead”	14.88 mm	15.78 mm	-0.90 mm (-5.70%)

Table 7.2: Comparison of object and model heights for distinctly chosen points.

Table 7.2 shows a comparison of the three most distinct points, measurable by the slide gauge, on fridge magnet fish. Using the slide gauge it’s only possible to measure heights where there is a local maximum. This is the reason for the rather short list of points. We can see that the model heights are approximately 1 millimeter shy of the measured object height which corresponds to roughly 6% of the total height. Assuming that these three points gives a general picture, we can conclude that the generated model heights in general are slightly smaller that the actual heights.

7.7 Summary

In this section it has been described how the calibration of the camera was preformed using a camera calibration toolbox for Matlab. The calibration of the camera makes it possible to take distortions of the lens into account when working with image coordinates. It also calculates the focal length and the principal point which are vital to know when converting an image coordinate into a line of sight.
The sheets-of-light emitted by the laser has been approximated first by a simple model assuming flat planes and later by a more advanced model allowing the curvature of sheets to be modeled. The latter ensures a much more precise reconstruction of the object and makes it possible to assemble the digital model using data from multiple consecutive lines. The 3D data from consecutive lines doesn’t align perfectly, but the method en general has proven to work well.
Comparison of the object and digital model shows height error in the magnitude of 1 millimeter. It does seem to be a general trend that the digital model is lower that the scanned object. A plausible explanation is that the applied model is incapable of attaining the exact curvature of the sheets-of-light generated by the laser. Therefore the reconstructed model may curve slightly downward in the middle of the cameras field of view. Recall that the point used for the reference height is extracted at the extreme left of the cameras field of view.

8 Future Work

8.1 Line Detection

In general the proposed line detection method works very well. A few threshold values do however need manual adjustment in order to cope with changing light conditions. An automated adaptive adjustment of these parameters would make the algorithm more user-friendly. Apart from that the only thing that can be improved is the precision of the estimated center of the line. This estimate is very sensitive to the intensity variations of the line due to the speckle noise generated by the laser. If the speckle noise could be further reduced the precision of the estimated center would probably increase.

8.2 Line segment correspondence

The two line correspondence algorithms each have their advantages and drawbacks. It difficult to see how either of the algorithms in their current form can be pushed to perform significantly better. However combining elements from the two algorithms is very much a promising possibility. If the labeling approach of the sequence based algorithm was adapted to the single image based algorithm, this could be used to detect label collisions on a single image basis. These collisions could then be added to the existing list of label collisions found by the sequence based analysis. Since the single image based analysis now wouldn't be responsible for the entire analysis the algorithm could be made more robust by only labeling line segment correspondences that where very certain. Thus by combining the best features of the two algorithms it should hopefully result in an algorithm that had the robustness of the sequence based algorithm but at the same time is capable of solving the correspondence problem on short sequences.

8.3 3D reconstruction

The mathematical model approximating the sheets-of-light emitted by the laser significantly increases the precision of the reconstructed 3D points relative to the initial approximation assuming flat planes. However the applied model does have weaknesses - the x-coordinate of the estimated laser position does not seem to converge very well toward the true position. A modification of the applied model that fixes this issue would presumably increase the overall precision of the scanner.

Although the applied model is able to model the curvature of the sheets-of-light emitted by the laser to some degree it is not able to attain the exact same curvature. Applying a higher order surface might resolve this issue and thereby increasing the precision of the scanner.

9 Conclusion

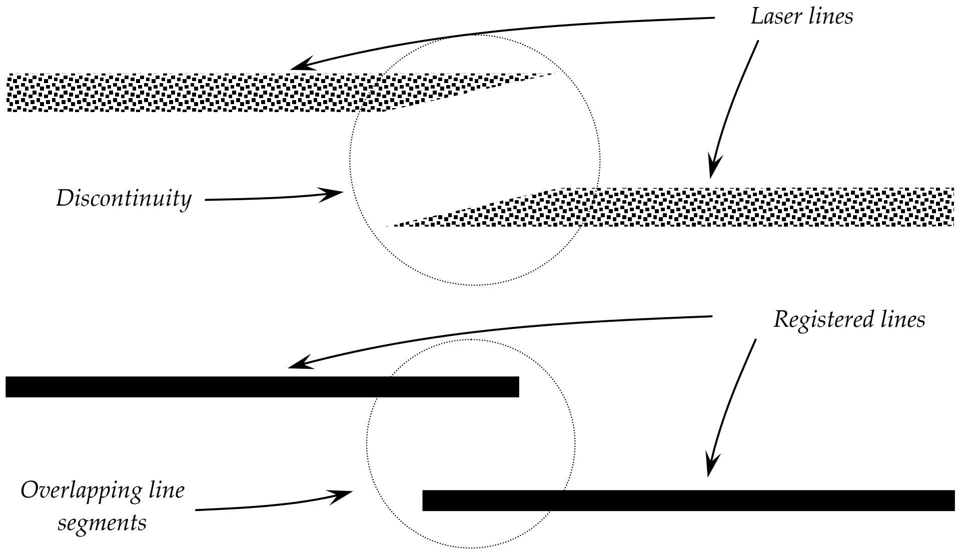
In this project a complete laser scanner system have been build from scratch. The detection of lines and line segments is based on a combination of standard image analysis and segmentation techniques or slight modifications of such. Two techniques for handling multiple lines have been developed and tested. The single image based correspondence technique work best for objects with a "simple" geometry and becomes unstable for objects with "complex" geometry. The force of this technique is that it can work on very short image sequences. The sequence based analysis is more robust and can handle more complex objects. It does however require a long image sequence in order to solve the correspondence issues.

The mathematical model applied to describe the sheets-of-light generated by the laser provides a much more precise reconstruction of 3D points than the one based on approximating the surfaces by flat planes. This along with the system calibration as a whole makes it possible to assemble the reconstructed points from different lines into a combined surface. Naturally there is still room for improvements. Modifying the model for the laser surfaces in order to obtain a better alignment between 3D points originating from different lines would improve the overall performance of the system.

In conclusion the main objectives of this project, namely building a laser scanner and make it utilize a multibeam laser, have been achieved. The generated digital model looks very convincing and resembles the scanned objects in great detail.

Appendix A – Detecting lines at discontinuities

At discontinuities the line is cut into two peaces. If the cut is more or less along the length of the line, then the line is present at two placed along the same vertical line. As a consequence the registered lines will overlap horizontally as illustrated below.



References

[1] Martin E. Anderson. *A beginner's guide to speckle*. http://dukemil.egr.duke.edu/Ultrasound/k-space/node5.htm

[2] Fausto Bernardini, Holly Rushmeier, Ioana M. Martin, Joshua Mittleman, and Gabriel Taubin. IBM T. J. Watson Research Center. *Building a Digital Model of Michelangelo's Florentine Pietà*.

[3] Jean-Yves Bouguet. *Camera calibration toolbox for Matlab,* http://www.vision.caltech.edu/bouguetj/calib_doc/

[4] M. Brown, T. Drummond and R. Cipolla. *3D Model Acquisition by Tracking 2DWireframes*. Department of Engineering University of Cambridge Cambridge CB2 1PZ, UK.

[5] Jens Michael Carstensen. Image analysis, vision and computer graphics. Technical University of Denmark, Lyngby, 2. edition 2002.

[6] Ivan Christov, *Multiscale Image Edge Detection*, May 12, 2004.

[7] Philip David, Daniel DeMenthon, Ramani Duraiswami, and Hanan Samet. *Simultaneous Pose and Correspondence Determination using Line Features.*

[8] James Davis, Xing Chen. *A Laser Range Scanner Designed for Minimum Calibration Complexity.* Proceedings of the Third International Conference on 3D Digital Imaging and Modeling, 3DIM 2001.

[9] Rafael C. Gonzalez & Richard E. Woods. *Digtital Image Processing.* 1993.

[10] E.R. Davies. *Truncating the Hough transform parameter space can be beneficial.* Machine Vision Group, Department of Physics, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK. 2002.

[11] BRUCE Z. GAO, OLAF SCHULZ, SAMIR PANDYA, and NED H. C. HWANG. *A Structured Light Technique for Monitoring Bioprosthetic Heart Valve Leaflet Surface Contours.* Cardiovascular Engineering, Vol. 1, No. 1, March 2001.

[12] Bill Green. Canny Edge Detection Tutorial http://www.pages.drexel.edu/~weg22/can_tut.html (2002)

[13] Patrick de la Hamette, Marc von Waldkirch, and Gerhard Tröster. *Laser Triangulation as a means of robust Visual Input for Wearable Computers.* International Symposium on Wearable Computer, October 2004.

[14] Janne Heikkilä. *Camera calibration toolbox for Matlab* http://www.ee.oulu.fi/~jth/calibr/

[15] Berthold Klaus Paul Horn, Robot Vision. The MIT Press, Cambridge, Massachusetts – London, England, 1998.

[16] Scott Konishi, Alan L. Yuille, James M. Coughlan, and Song Chun Zhu. *Statistical Edge Detection: Learning and Evaluating Edge Cues.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 1, January 2003.

[17] Sung Chun Lee, Soon Ki Jung, and Ram Nevatia. *Automatic Pose Estimation of Complex 3D Building Models*.

[18] Tony Lindeberg. Scale-space: A framework for handling image structures at multiple scales. Stockholm, Sweden, 1996.

[19] Hans Brunn Nielsen. Algorithms for Linear Optimization. Technical University of Denmark, Lyngby, 2. edition 1999.

[20] Filip Sadlo, Tim Weyrich, Ronald Peikert, Markus Gross. *A Practical Structured Light Acquisition System for Point-Based Geometry and Texture.* Eurographics Symposium on Point-Based Graphics 2005.

[21] Daniel Scharstein. Middlebury College & Richard Szeliski. Microsoft Research. *High-Accuracy Stereo Depth Maps Using Structured Light.*

[22] Guido M. Schtrster & Aggelos K. Katsaggelos. *Robust line detection using a weighted MSE estimator.*

[23] Filareti Tsalakanidoua, Frank Forsterc, Sotiris Malassiotisb, Michael G. Strintzisa. *Real-time acquisition of depthand color images using structured light and its application to 3D face recognition.* Real-Time Imaging 11 (2005).

[24] Alan Watt & Mark Watt. *Advanced Animation and Rendering Techniques*. 1992.

[25] Zhiyong Wang, Jixian Zhang, Tongxiao Wang. *The Contrast Research of the Methods of Restraining the Speckle Noise of SAR Images*. Shandong University of Science and Technology, Tai'an, China. Chinese Academy of Surveying and Mapping, Beijing, China.

[26] Mark A Schulze, Qing X Wu. *Nonlinear Edge-Preserving Smoothing of Synthetic Aperture Radar Image*. Procedings of the New Zealand Image and Vision Computing '95 Workshop, pp. 65-70. (Christchurch, New Zealand, August 28-29 1995.)

[27] ShapeGrapper 3D laserscanners. www.shapegrapper.com

[28] Laser Design. http://www2.laserdesign.com

[29] 3Shape - 3D laser scanners. www.3shape.com

References to contents on the enclosed DVD

[30] The checkerboard images used for the camera calibration can be found in the 'Matlab - work\TOOLBOX_calib\CalibrationImages' directory on the enclosed DVD.