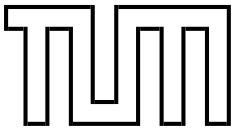
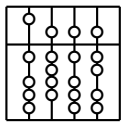


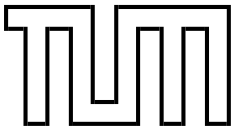
Technische Universität München
Fakultät für Informatik



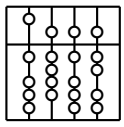
Diplomarbeit

Tracking Scissors Using Retro-Reflective Lines

Katharina Pentenrieder



Technische Universität München
Fakultät für Informatik



Diplomarbeit

Tracking Scissors Using Retro-Reflective Lines

Katharina Pentenrieder

Aufgabensteller:	Prof. Gudrun Klinker, Ph.D.
Betreuer:	Dipl.-Inf. Martin Bauer
Abgabedatum:	15. Januar 2005

Ich versichere, dass ich diese Diplomarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Garching, den 15. Januar 2005 Katharina Pentenrieder

Abstract

Tracking plays an important role in Augmented Reality (AR) applications. In medical augmented reality, tracking systems must allow to accurately determine the position of medical instruments without influencing or disturbing the surgeons working environment. Retro-reflective material combined with infrared light presents an interesting and promising method of optical tracking.
This thesis analyzes the special case of tracking medical scissors using retro-reflective lines. The particular structure of this instrument permits its identification in an image only through the position of the two legs. During this project the whole process of image generation, line detection, camera calibration and 3D reconstruction via line triangulation was examined. The used methods, problems and undertaken optimizations are presented here. Furthermore some ideas for the part of tracking the instrument over a series of images are briefly discussed.

Zusammenfassung

Tracking spielt eine wichtige Rolle in Augmented Reality (AR) Anwendungen. In medizinischer Augmented Reality müssen Tracking-Systeme die genaue Positionsbestimmung medizinischer Intrumente ermöglichen ohne dabei das Arbeitsumfeld des Arztes zu beeinflussen oder zu stören. Reflektorfolie in Kombination mit infrarotem Licht stellt eine interessante und vielverprechende Methode optischen Trackings dar.
In dieser Diplomarbeit wird ein spezieller Fall untersucht: das Tracking von medizinischen Scheren mit Hilfe von reflektierenden Linien. Die besondere Struktur dieses Instruments erlaubt seine Erkennung in einem Bild allein über die Lage der zwei Schenkel. In diesem Projekt wird der gesamte Prozess von Bildergenerierung, Linienfindung, Kamerakalibrierung und 3D Rekonstruktion über Linientriangulierung behandelt. Die verwendeten Methoden, Probleme und vorgenommenen Optimierungen werden hier vorgestellt. Weiterhin werden einige Ideen für das Tracking des Instruments in einer Bildfolge kurz behandelt.

Contents

1 Introduction

Motivation and Introduction to Approach and Utilities

1.1 Motivation

Tracking plays an important role in Augmented Reality (AR) applications. Various kinds of tracking methods can be used, for example magnetic, mechanic or optical tracking [5]. This project applies optical tracking. Here a pair of medical scissors shall be tracked using its legs. Two questions arise:
Firstly, why and where is it useful to track medical instruments? Looking at an operating room answers can be found very quickly. First it is important to know where the objects of interest are at any point in time. The knowledge of how medical instruments are used during an operation can be helpful for visualization. Simulations or even a robot could be controlled with that information. In general tracking can be used to optimize work procedures and protocols.
And secondly, what are the reasons to choose retro-reflective lines for the optical tracking process? One possibility to find and track an object is to attach a fixed arrangement of balls to it and use their constellation to determine the three-dimensional position of the object. Medical instruments like scissors could also be tracked that way but the ball arrangement can be disturbing while working with the instrument. Hence another approach to identify the position and orientation using only the legs of the instrument can be interesting to analyze.

1.2 Approach

The aim of the diploma thesis is to track medical scissors using retro-reflective lines. The legs of the instrument are covered by retro-reflective material and the pair of scissors is exposed to infrared light. Thus images can be generated where the legs are easily recognizable.
Figure 1.1 shows the principal steps in this project. First of all images are created. We used two different systems to do this, a stereo infrared camera system and the RAMP System

which are both described in detail in chapter 2. The resulting infrared images are then processed to retrieve the lines representing the legs of the pair of scissors. This is done using Hough Transformation and some additional methods for optimization (see chapter 3). The extracted information about the 2D lines in left and right images are then put together in the triangulation step where the 3D position of the medical instrument is determined. The methods and the approach can be found in chapter 5. But in order to calculate the 3D coordinates more than just the 2D image data is necessary. Information on the camera projections and the position of the right and left camera with respect to each other need to be known, too. These so called intrinsic and extrinsic camera parameters are computed in the camera calibration process which is specified in chapter 4 on stereo camera calibration. To compute the calibration parameters a stereo image series with known point correspondences is necessary, therefore additional non-infrared images have to be generated during the image acquisition step for calibration usage. This connection is displayed in figure 1.1 by the calibration branch on the left side. Finally the scissors shall be tracked over a series of images. Chapter 6 presents some ideas and methods which could be used for this part of work. An overall conclusion of the project is given in chapter 7.
This thesis presents several established and fully developed algorithms and methods for line detection, calibration, stereo triangulation and tracking. They are analyzed for their appropriateness and their quality in this special case. In addition this paper introduces modifications and extensions for a more exact and enhanced functioning.

1.3 Programming Language and Utilities

1.3.1 Matlab

The intention of this project is to find and test the main algorithms for the detection and 3D reconstruction of the pair of scissors. Matlab 6.5 [1] is used for the whole implementation. This programming language offers a good environment for image processing and numerical computations. It includes a lot of useful functions and allows an easy handling of matrix-vector-calculations. This makes Matlab ideal for this project because the focus can lie on the development of the algorithms while the basic computation possibilities and needed numerical functions are already given and ready for usage.
The disadvantage of Matlab is that it tends to use a significant amount of memory and it is relatively slow in executing if-statements and for- and while-loops. Hence the Matlab algorithms created in this project show the possibilities and the quality of the selected methods but they can not be representative concerning memory and time consumption (see section 7.2.3 on time comparison).

1.3.2 Matlab Camera Calibration Toolbox

For the determination of the intrinsic and extrinsic parameters special algorithms are necessary which use point correspondences to estimate these parameters. The ideas behind estimation are described in section 4.1.6. The Matlab Camera Calibration Toolbox was developed by Jean-Yves Bouguet (California Institute of Technology) and is a free contribution

offering Matlab functions for camera calibration [2]. In this project it is used to do the calibration of the stereo camera system. The usage of the toolbox and its functionality are specified in section 4.2.1.

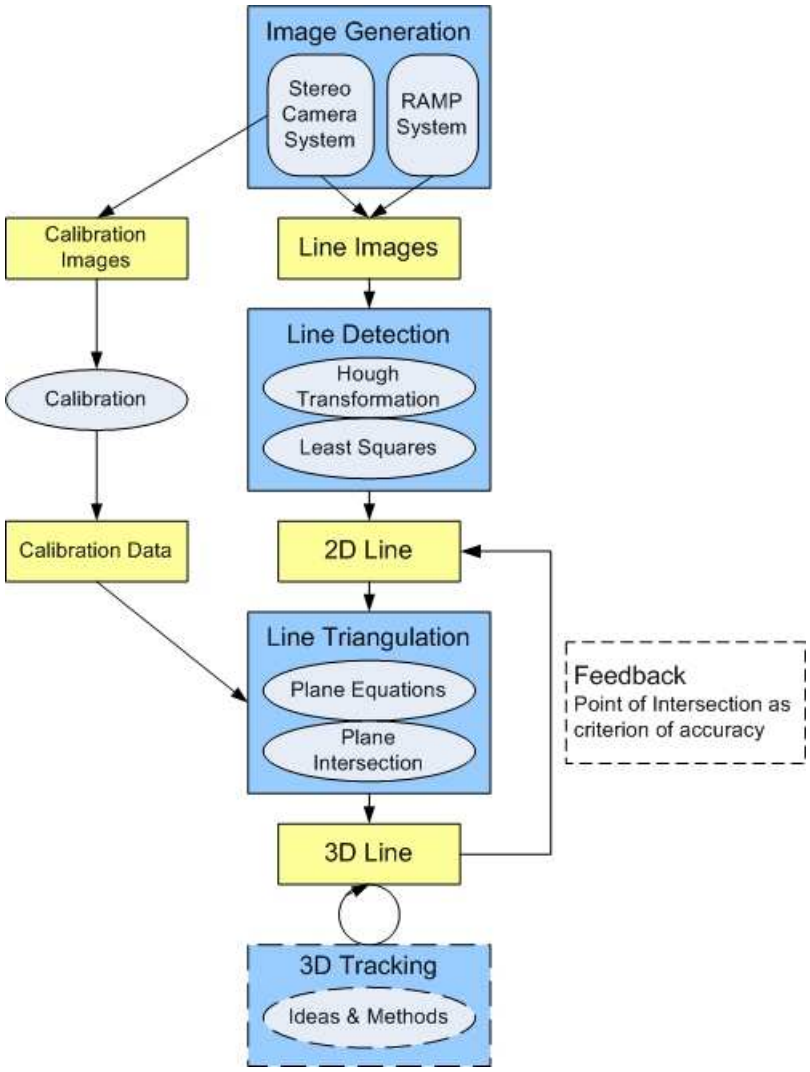


Figure 1.1: Graphical presentation of the approach

1.3.3 Hardware

In this project two systems are used for image generation: a stereo camera system and the RAMP System. Both approaches are described in detail in the next chapter.

2 Stereo Image Series

Experimental Setup and Generation of Images

For this project we used two different methods for image acquisition.
The first series were taken with a system of two stereo cameras with attachable infrared filters. Secondly the RAMP System was used to get images of a pre-calibrated camera. Both systems are described in detail in the following sections.

2.1 Stereo Camera System

To recover the position of the pair of scissors in 3D space from 2D image data stereo information is needed. So the first image series were generated with a stereo camera system as figure 2.1 shows.

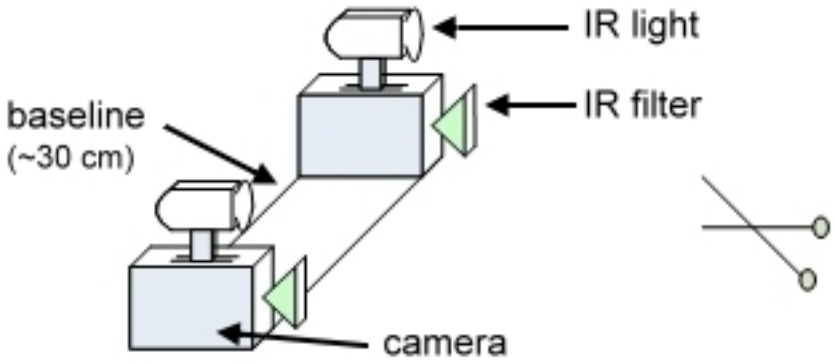


Figure 2.1: Stereo Camera System

2.1.1 Experimental Setup

Two SONY cameras of type DFW-VL500 are used for the experimental setup. The technology integrated in these digital cameras ensures high color accuracy square pixels and sharp, high resolution images even of fast moving objects. To visualize the retro-reflective lines we used Heliopan RC 830 filters which can be attached to both cameras. For an optimal setup infrared flashes should be used, resulting in an equal and adequate illumination of the scene. Unfortunately such flashes were not available during this project and simple infrared light heating lamps were taken instead. They produce a rather unequal illumination and therefore the generated images do not have the best achievable quality (compare figure 3.7 (b)). The scissors were covered by reflector foil (3M-Scotchlite high reflection film 8850) providing an optimal reflection of the visible spectrum. For the stereo system the two cameras were used with a baseline of about 30 cm.

2.1.2 Calibration Data

As this stereo system is uncalibrated the calibration parameters are retrieved by using the Matlab Camera Calibration Toolbox. This toolbox allows to compute the intrinsic and extrinsic parameters of a stereo system which encode the projection parameters for each camera and the relative position of the right and left camera with respect to each other in the stereo system. A detailed description of the parameters and their usage is given in chapter 4 on Stereo Calibration. Section 4.1 gives an overview on the theory behind projection and calibration and section 4.2 presents the calibration process with the Toolbox. For this process chessboard images are necessary (see figure 4.4). In contrast to the scissors images they are non-infrared.
Hence two stereo image series needed to be generated: a non-infrared stereo image series of a chessboard to gain data for the calibration process and the infrared image series of the pair of scissors for the further computation steps. Both series must be taken with the same fixed camera arrangement to have calibration data which fits the infrared images. Here the great advantage of the used system is the simple way of switching between infrared and non-infrared just by attaching or detaching the filters.

2.1.3 Image series

With an adequate software the retrieval of the image series is a very easy task. The computer program gets the data from the left and right camera and stores it as movies or image lists so that corresponding left and right data are recognizable. Unfortunately these tools for stereo acquisition were not available at the beginning of this project. Therefore at development time only single images with the pair of scissors at a fixed place could be generated which led to a very small number of test pairs (left-right). For the second series we could use a stereo image acquisition system implemented by Johannes Schäffner as a system development project (SEP) at the TU Munich. Given this system a large data set for calibration as well as line detection could be generated for testing and validation.

2.1.4 Problems

The first image series of the stereo camera system gave no convincing results. The unequal illumination and the difficulties of getting good calibration data were most likely the causes. A second data set of a more stable camera setup and thus more exact calibration parameters achieved better 3D reconstructions of the scissors. But still the outcome was not optimal as section 5.4 states.
To find out whether the bad results are a problem of calibration accuracy another approach was tried with a pre-calibrated system: RAMP.

2.2 RAMP System

The RAMP System (Real Time Augmentation for Medical Procedures) is a project of SCR, Siemens Corporate Research. It shall support doctors by imaging complex data in an easily understandable way. The system improves medical procedures in several different ways, for example 3D-Visualization, Image and Tool registration.
RAMP's major components are a video see-through Head Mounted Display (HMD) with two stereo color cameras and an infrared camera for optical tracking. The software consists of a RAMP server and a RAMP client. A short overview on these parts is given in the following paragraph. More details on RAMP can be found in [18] and [17].
We do not use the system in the sense which is described above, only the infrared tracking images and the given calibration data are important in our case.

2.2.1 RAMP's Components

The video see-through Head Mounted Display provides the visualization. It consists of two color cameras which copy the original field of view to the HMD. In addition a third infrared camera is used for optical tracking. It has a wide angle lens to extend the angle of view and is surrounded by a strong circular infrared flash to give an illumination which does not change the visible spectrum. Figure 2.2 shows the HMD from the side. For this project only the images taken by the infrared camera are used. The HMD and its multifarious possibilities are not of interest.
The RAMP server provides the position of the HMD relative to an absolute coordinate system. To determine the position a set of so called fiducials needs to be visible in the infrared camera images. These circular retro-reflective markers are illuminated by the infrared flash and allow RAMP to compute the relative position of the HMD to the absolute coordinate system whose origin is located in the marker set. Figure 2.3 shows the fiducial markers. In addition the RAMP server also provides the position of tools marked with fiducials, a feature which is not used in this project.
Finally the RAMP client takes the information from the server and reconstructs the scene with one view for each eye using the images of the two color cameras.

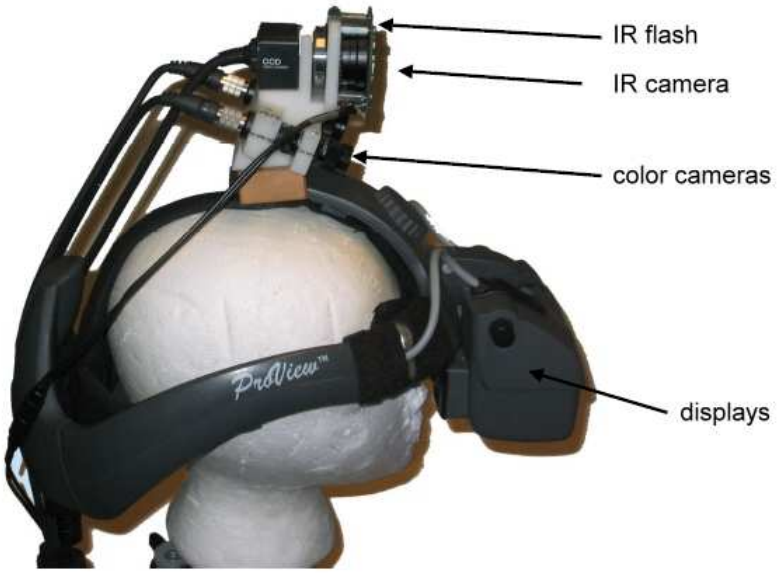


Figure 2.2: HMD from the side

2.2.2 Experimental Setup and Image Series

As RAMP provides only one infrared camera the stereo images need to be generated by using two different positions of the HMD with respect to the medical instrument. The scissors are placed at a fixed position, the HMD is moved around them and images are taken at divers positions resulting in several pairs of left and right images. In every case the majority of the fiducials must be visible in the infrared images to enable the computation of the extrinsic calibration parameters.

Figure 2.3: Fiducials without flash (left) and with flash (right)

2.2.3 Calibration Data

The introduction to this chapter already presented RAMP as a pre-calibrated system. The intrinsic calibration parameters for the infrared camera are already given and therefore no additional chessboard images need to be generated. The extrinsic information, the relative position of right and left images with respect to each other, can be derived from the single images. The fiducials allow the RAMP software to compute the position of the HMD in the absolute coordinate system. It is given as a rotation and a translation in the left upper corner of the images (see figure 4.6). With this information the position of the HMD for a left and a right image shot can be calculated as in section 4.3.

2.3 Resulting Data

Both systems store the images as RGB arrays of size 480×640. However, in the infrared spectrum, the color channels don't have significant meaning anymore. Hence for the further calculations only one of the three color arrays is used. Its cell values are between 0 and 255. Figure 2.4 shows a gray value image with a magnified part including some of the gray values.

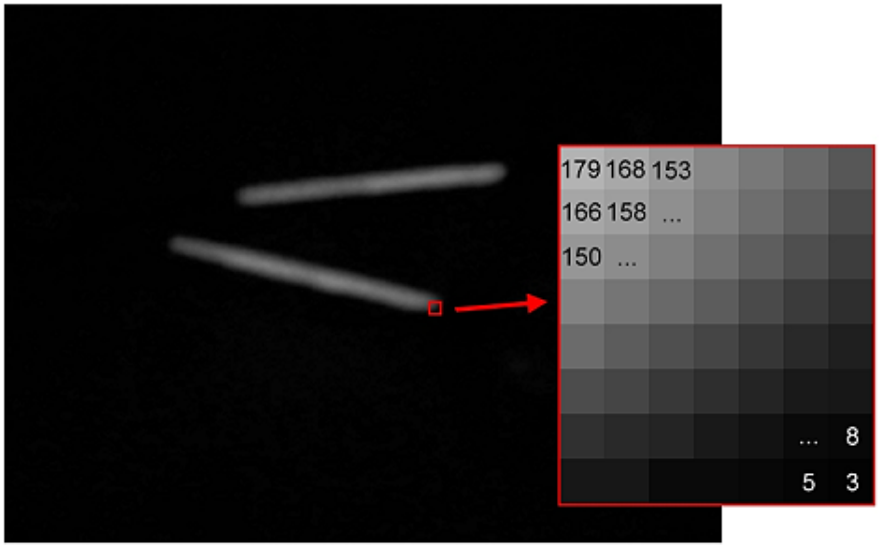


Figure 2.4: Image with partial gray value array

3 Line Detection

Methods, Problems and Optimizations

The obtained infrared images are now processed to detect the pair of scissors. This problem is reduced to simple line detection as the position of the instrument is determined by two intersecting lines.
The line retrieval is done via Hough Transformation which is a nice method with good computational complexity. Section 3.1 presents a theoretical introduction and then a plain realization of the Hough Transformation is described. As the results are not satisfying the process of detection is extended. The Hough Transformation is split to find each leg of the pair of scissors separately and for a more exact result a Least Squares Method is applied afterwards to achieve better line equation accuracy. These improvements are described in detail in section 3.2.
The chapter continues with a section on the problem of deinterlacing and some marginal cases which need to be catched (3.3). Finally the data which is passed on to the next processing step is presented in section 3.4.

3.1 Hough Transformation

Hough Transformation is a very efficient method for detecting lines and curves in images. A straightforward way to find lines could be to test all the lines formed by each pair of figure points for their likelihood. But this implies a lot of calculation effort. Hough Transformation chooses a more elegant way by transforming every figure point into a parameter space. There it votes for all the lines it might lie on. So if all possible lines were represented as buckets in parameter space each figure point would put a ball in every potentially matching bucket. Finally the actually present lines in the image should be the ones which gained a lot of votes, the buckets with a large number of balls.

3.1.1 Theoretical Basics

Hough Transformation is done in two steps. First the image points are transformed into a parameter space and then the votes are accumulated to find the maximum.

Transformation
Each line can be represented by a tuple (Θ, ρ) where Θ is the angle between the line and the x-axis and ρ is the distance of the line to the origin (see Figure 3.1). The transformation computes for every point (x, y) all possible pairs (Θ, ρ) representing all possible lines going through that point. The angle Θ is varied and ρ, the distance, is calculated with the following equation:

$$\rho = x \cdot cos(\theta) + y \cdot sin(\theta)$$

This transformation is continuous. Hence a first discretization is done, as the input figure points are discrete. Each of these points is transformed into a sinusoidal curve in the parameter plane. On the other side each point in parameter space refers to a straight line in the image plane. Figure 3.1 shows this exemplarily.

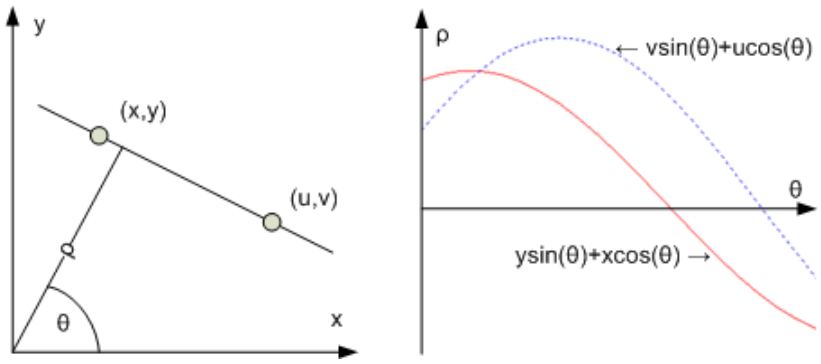


Figure 3.1: Hough Transformation

There the points (x, y) and (u, v) in the image plane on the left are transformed into the red and blue curve on the right side. As these points are colinear their parameter space curves have a common point of intersection, (Θ, ρ). This intersection point refers to the line through (x, y) and (u, v) in the original image coordinate system.

Accumulation
Transforming all the figure points into parameter space results in a large number of curves with multiple intersection points. Exactly colinear image points can be found by finding coincident intersection points in parameter space. Present lines in the picture correspond to large numbers of coinciding points (lots of votes) and therefore these intersections need to be counted. Unfortunately this plain approach of counting requires a lot of computational effort as n sinusoidal curves normally intersect in $\sum_{i=1}^{n} i = \frac{n*(n-1)}{2}$ points which means that the computation grows quadratically with the number of image points [7].

So for the accumulation of votes in parameter space the burden can be reduced by discretizing the parameter space. This second discretization quantizes the $(\Theta, \rho)-$ plane into a quad ruled grid and the counting can then be done with a so called accumulator array. Each cell of the array represents a pair (Θ, ρ) and collects the votes for this pair. The cells which gained a high number of votes in the array can easily be retrieved and the corresponding

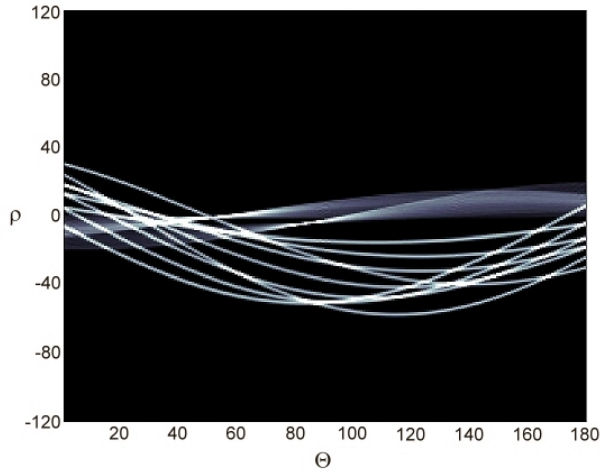


Figure 3.2: Accumulator Array

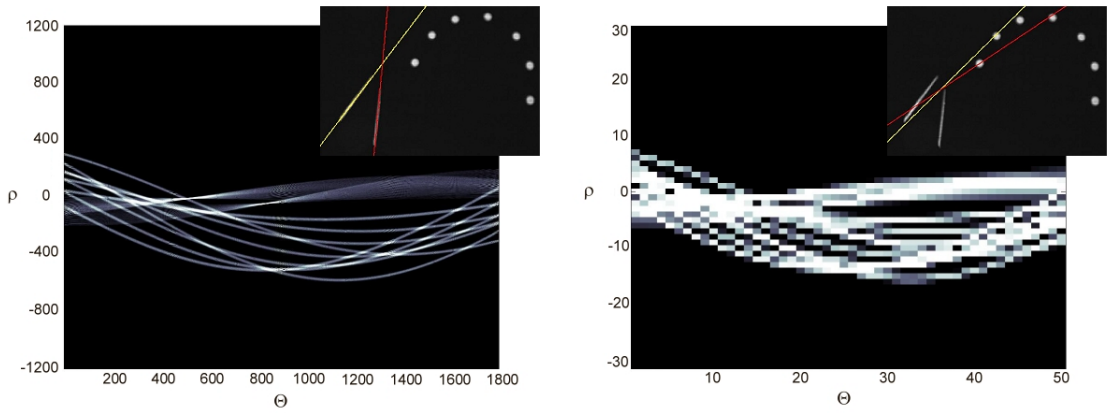
pairs (Θ, ρ) are the angle and distance parameters of the lines in the image. If a cell got k votes then precisely k figure points in the image lie on that line. Figure 3.2 shows an image representation of such an array. The number of votes for a cell is represented by its intensity.

3.1.2 Practical Problems

Hough Transformation was introduced as a nice and efficient way of detecting lines. Still a number of problems arise in practical usage, general ones as described in [7],[8] and some which occur particularly in our case. They shall be described here.

Discretization
In the theoretical introduction of the transformation the problem of discretization is already explained. In order to reduce the time consumption of the computation the values of Θ and ρ are quantized to allow the accumulation in an array of fixed size in both dimensions. But the question of grid size still needs to be answered. A very fine grid results in better resolution but higher computation times. And if points are not exactly colinear the transformation might put their votes in different buckets. Then the corresponding line might not be identified. But too coarse quantizations lead to falsely large votes because quite different lines refer to the same grid cell.
Figure 3.3 points out the difficulty of grid size. The accumulator array for fine and coarse quantization is displayed as well as the resulting hough lines in the right upper corners. Further explanations for example on the quantization parameters r and t will follow in section 3.1.3.

Difficulties with noise
An advantage of Hough Transformation is that it connects widely separated image points if they are nearly colinear. But this can also be a weakness. Phantom lines might be detected in regions with uniformly distributed figure points. Here this problem does not arise as the number of image points belonging to the legs of the scissors is a lot larger than any possible additional noise in the images.



(a) Fine grid (t=1200,r=800)

(b) Coarse grid (t=50,r=30)

Figure 3.3: Discretization: Accumulator Array and Hough Lines

Different pixel intensities
The infrared images are grey value matrices with values between 0 and 255. Feature points with values close to zero probably don't belong to the bright pixel fields representing the lines. Hence these image points must not influence the accumulation as they would disturb the calculation.

Different line intensities
First tests with infrared images revealed the problem of different line intensities. When the number of figure points belonging to the two legs of the instrument differ considerably, the Hough Transformation can only find the more intensive line. Taking more pairs (Θ, ρ) with high value will only give more lines through the intensive leg with slightly different angles and distances. An example is shown in figure 3.4. The upper leg is more intensive than the lower one and the back transformation of more points in parameter space results in several lines through the upper leg. Section 3.1.3 resumes this aspect and introduces the parameter btt, a back transformation threshold.

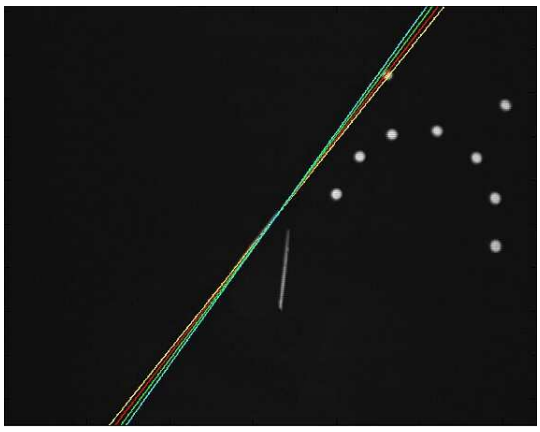
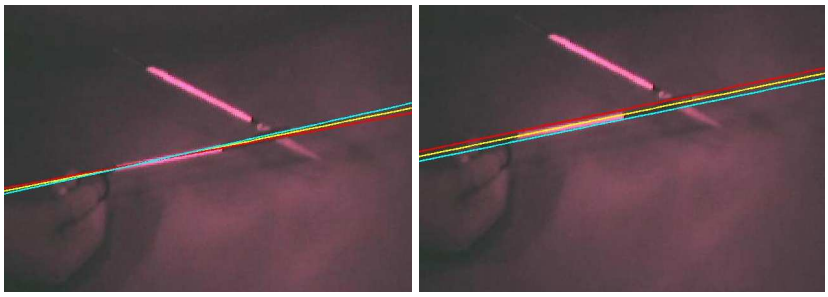


Figure 3.4: Different line intensities: btt=0.7

Exact line equation retrieval
When the maximum of the accumulator array is taken to define a line in the original image this line will very likely be a diagonal through the pixel field representing the line in the image (see figure 3.12 (a)). The reason for this behavior is that the pixel areas are mostly rectangular and longish and therefore the diagonal covers a maximum possible number of feature points. In addition the possible line positions depend on the discretization of ρ and Θ. Figure 3.5 shows three discretization steps for Θ (a) and ρ (b) which are closest to the searched line in the image. In both cases the middle line is the one which was detected by the Hough Transformation. Its parameter space representation is (Θ, ρ). Figure 3.5 (a) displays the lines $(\Theta - 1°, \rho)$, (Θ, ρ) and $(\Theta + 1°, \rho)$, figure 3.5 (b) shows the detected line (Θ, ρ) and the two nearest lines with fixed Θ, $(\Theta, \rho + \frac{640}{130})$ and $(\Theta, \rho - \frac{640}{130})$. The spacing for neighboring lines in Θ and ρ direction depends on the discretization parameters t and r (see section 3.1.3). In this case they are $t = 180$ which results in one degree steps and $r = 130$ resulting in $\frac{640}{130}$ steps.
So Hough Transformation allows to detect lines in an image but for an exact equation additional methods need to be considered.



(a) Θ discretization (b) ρ discretization

Figure 3.5: Discretization: Possible line results

In the following sections the realization of the line detection process will be described step by step. First a plain realization is presented which only deals with the problems of discretization and different pixel intensities (section 3.1.3). For the issues of different line intensities and the detection of two lines several approaches of clustering were examined. The results are given in section 3.1.4. The two major improvements for a satisfying calculation of both line equations are the splitting of the Hough Transformation to find each line separately (section 3.2.1) and the application of a Least Squares Method for an optimal line equation (section 3.2.2).

3.1.3 Plain Realization

The first approach for line detection implements the Hough Transformation using the equation of section 3.1.1 and an accumulator array as described. The function has four parameters: $hough1(r, t, gvt, btt)$. Their usage and the results for different settings are shown in this section.

Implementation
To face the problem of discretization the function allows variable grid cell sizes. The quantization is controlled with two parameters t and r indicating the number of partitions of the interval for the angle Θ and the distance ρ. Figure 3.6 shows the position of the $(\Theta, \rho)-$coordinate system in the image and the quantization depending on t and r.

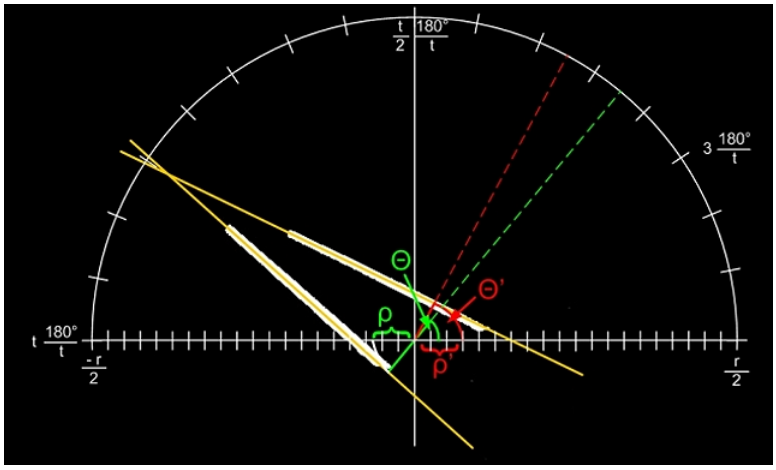


Figure 3.6: Position and quantization of the (Θ, ρ) coordinate system

The origin is placed in the middle of the image and the $\rho-$axis is the horizontal line which is discretized in r partitions. The labels range from $\frac{-r}{2}$ to $\frac{r}{2}$. Θ, the angle, can be found on the half circle indicated around the coordinate system which is quantized in t parts. For the transformation $\Theta \in [0; 180]$ therefore ρ can also be negative, for example in the case of the green colored parameter pair (Θ, ρ). Figure 3.6 also demonstrates the problem of grid size. The quantization of Θ is very coarse and therefore both lines have very inexact values for the angle as the rays meet the half circle in the middle of the $\Theta-$partitions.
The accumulation is slightly modified to consider the different pixel intensities. Instead of incrementing the array cells by 1 a weighted accumulation is done. For each pixel p its weight is calculated as $weight_p = \frac{pixelvalue}{255}$. In the process of accumulation the grid cells which get a vote from p are incremented by $weight_p$.
Besides the values for discretization two additional parameters need to be set in this first implementation: a natural grey value threshold $gvt \in [0; 255]$ and a real back transformation threshold $btt \in [0; 1]$. The different pixel intensities are already considered in the accumulation process but in order to decrease the computational effort feature points with very small values shall be excluded completely from the whole Hough Transformation process. It is not possible to determine a reasonable fixed value. The threshold for unimportant pixels is strongly varying because the image brightness and therefore the values are depending on the illumination and the distance of the pair of scissors to the camera. The parameter gvt offers a variable setting of this threshold to find an adequate value.
Finally the back transformation threshold btt determines the number of pairs (Θ, ρ) to be considered for back transformation into the image as lines. If $btt = 0.9$ then all the cells of the accumulator array with values greater or equal 90% of the maximum of the whole array are transformed back into the image (see figure 3.4). At the beginning this was supposed to allow both lines in the image to be discovered. But as mentioned in 3.1.2 the different line

intensities thwart this approach.

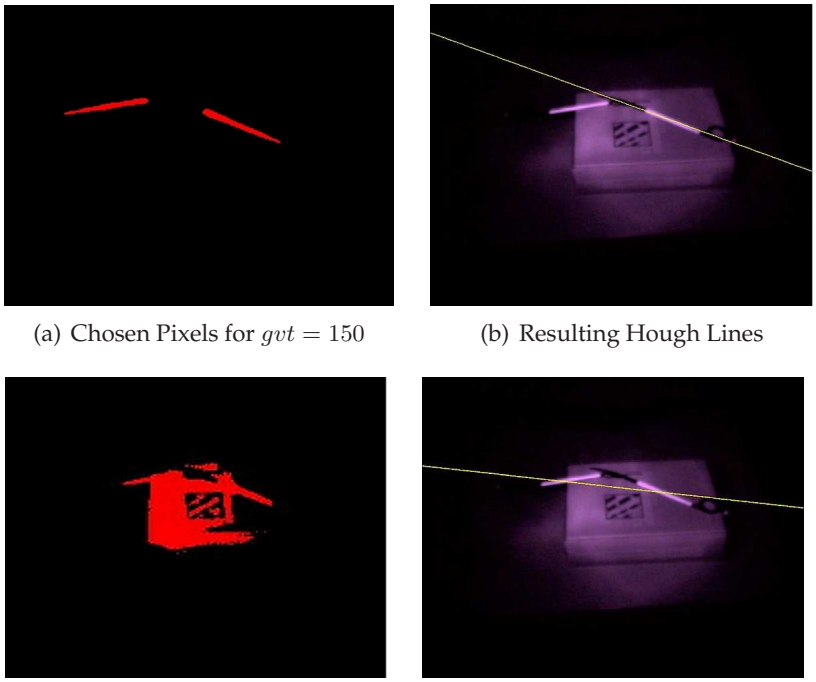
Results

After an introduction to the different parameters their influence on the resulting line detection shall now be examined. Additionally several figures are presented to give a better idea of the effects.

For the case of discretization the best choice for the cell sizes can hardly be determined. Figure 3.3 in section 3.1.2 reveals the two extremities for a very fine or a very coarse grid. Still there is a lot of room for variation left in-between these borders. More tests with images differing in illumination, line slope and quality allowed to determine reasonable values for r and t. In the further calculations the distance quantization is set to $r = 130$. This permits a sufficing exactness for the given images of 480×640 pixels while the computational complexity is still low. For the angle the discretization is $t = 180$. In addition to good results in computation this value is also intuitive as $180°$ divided by 180 result in a one degree quantization.

The gray value threshold cannot be set to a fixed value. Different intensities of illumination while generating the images and the variable distance of the pair of scissors to the camera result in highly differing pixel values. Therefore gvt has to be determined for each image separately by a set of tests. Alternatively histogram-based methods could be used to adaptively identify an adequate value for gvt.

Figure 3.7 presents resulting lines for different values of gvt. Here the image is disturbed



(a) Chosen Pixels for $gvt = 150$ (b) Resulting Hough Lines

(c) Chosen Pixels for $gvt = 100$ (d) Resulting Hough Lines

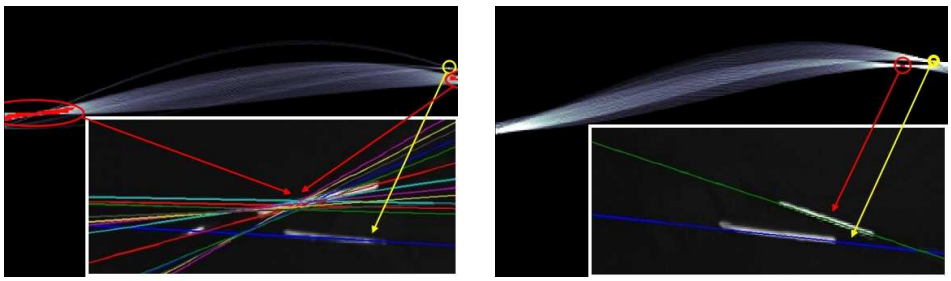
Figure 3.7: Comparison of different gray value thresholds

by noise originating from bad, non-uniform illumination. In order to extract the legs of the instrument the gray value threshold has to be very high. The left subfigures show the pixels

from the original image chosen for transformation. They have values $value_p \geq gvt$. In the right subfigures the resulting hough lines can be seen in the original image.
The effects of different values for the fourth and last parameter, the back transformation threshold, are demonstrated in figure 3.4. It is not possible to determine the equations of both lines by taking more grid cells for back transformation than just the maximum one. For very unequal intensities the additionally computed lines still refer to the same leg.
In any case if the back transformation threshold was dropped to a point where eventually both legs would be found, still a method would be necessary to determine which pairs (Θ, ρ) approximate the searched lines best. The next section discusses some efforts to at least identify the line bundles for each leg by using different parameters for clustering.

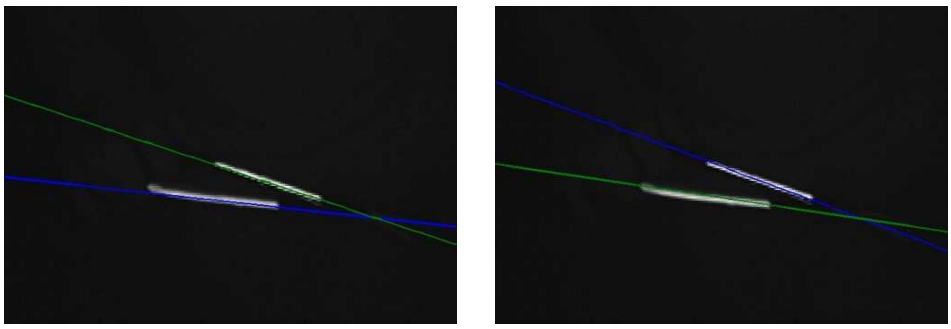
3.1.4 Clustering efforts

When several accumulator cells are used for back transformation the resulting lines mostly gather around an average line which identifies the searched line quite nicely (compare figure 3.4). Therefore this good approximation for the line could be retrieved by taking the whole line bundle of the leg and averaging it. The intention here is to identify the pairs (Θ, ρ) representing the same leg of the scissors in the image by examining similar values of Θ and ρ.
In both cases a certain number of accumulator cells is used for back transformation according to the threshold btt. The chosen parameter pairs are grouped either by values of Θ or ρ. For a ρ clustering all the pairs $(\Theta_1, \rho), ..., (\Theta_n, \rho)$ are used to calculate an average value for the angle $\bar{\Theta} = \frac{1}{n}\sum_{i=1}^{n}\Theta_i$. Analogously an average $\bar{\rho}$ is determined for similar values of Θ which are retrieved by rounding. But the resulting line parameters $(\bar{\Theta}, \rho_k)$ and $(\Theta_j, \bar{\rho})$ are not useful in determining the best approximation. It is very sensitive to different line intensities and a cluster for Θ can be broken in two parts around the change $0°$ to $180°$. In 3.8 the accumulator arrays of two images are displayed. The clusters for each line are marked. Here the sensitivity to line intensity is visible. The lines in figure 3.8 (a) have very different intensities, hence if both line cluster shall be found the cluster for the more intense line grows too big. A line equation retrieval is not possible. For case 3.8 (b) the lines are equally intense and the clustering is successful. The resulting lines are shown in 3.9.
For a general application this approach is not satisfying, hence other methods are necessary to improve the results.



(a) Different line intensities (b) Equal line intensities

Figure 3.8: Accumulator array and detected clusters



(a) Image with ρ clustering lines (b) Image with Θ clustering lines

Figure 3.9: Successful clustering efforts

One more idea could be to find clusters in the accumulator array which was not examined for this project. A quite different approach is implemented instead. The grey value threshold, if chosen conveniently, already allows to identify the two line clusters in the image as can be seen in figure 3.7 (a) and the final distinction of the two legs is done via splitting.

3.2 Improvements

In order to identify the pair of scissors in the infrared images both legs need to be detected and good approximations for the line equations must be calculated. The parameters which were introduced with the plain realization of the Hough Transformation are not able to achieve this goal. To get satisfying results further steps need to be undertaken. They will now be described in detail.

3.2.1 Split Hough Transformation

Section 3.1.3 states that Hough Transformation can easily detect the most intensive line in the regarded images if the parameters are chosen conveniently. Hence the first line is found by simply applying the transformation as described in the plain realization. To retrieve the second leg a modified image is needed in which the missing leg is more intensive than the one already determined. This image can be created by eliminating the bright pixels along the Hough line for the known leg. So the approach to detect both legs is realized by a split Hough Transformation, a separate transformation for each leg. The resulting improved function is $hough2(r, t, gvt, et)$.

Steps of $hough2(r, t, gvt, et)$

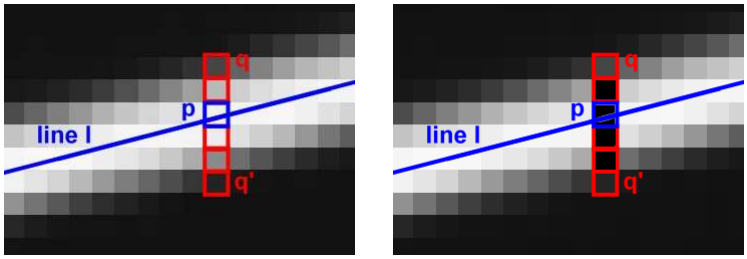
1. Apply $hough1(r, t, gvt, 1.0)$ to the given image I.
   Only the maximum in the accumulator array is taken to get the pair (Θ, ρ) for the most intensive line in the image, therefore $btt = 1.0$.

2. Calculate the equation for the line l represented by (Θ, ρ).

3. Eliminate bright pixels along the line l to create a new image I_{new}.
   Starting from each pixel on the line, $p \in l$, the pixels above and below p are examined for brightness. As long as their gray values are greater than the elimination threshold et the pixels are colored black, their values are set to 0 and the adjacent pixels are examined next. If darker pixels q, q' with gray value $\leq et$ are reached the elimination is stopped (compare figure 3.10). This proceeding results in an image I_{new} where the bright pixels of the more intensive scissors leg are blackened (see figure 3.11 (a)).

4. Apply $hough1(r, t, gvt, 1.0)$ to the new image I_{new}.
   The new accumulation maximum (Θ', ρ') represents the second leg which is now the brighter one.

5. Calculate the equation of the second line l' with the new parameters.



(a) Image before elimination

(b) Image after elimination of first column

Figure 3.10: Elimination process

Results

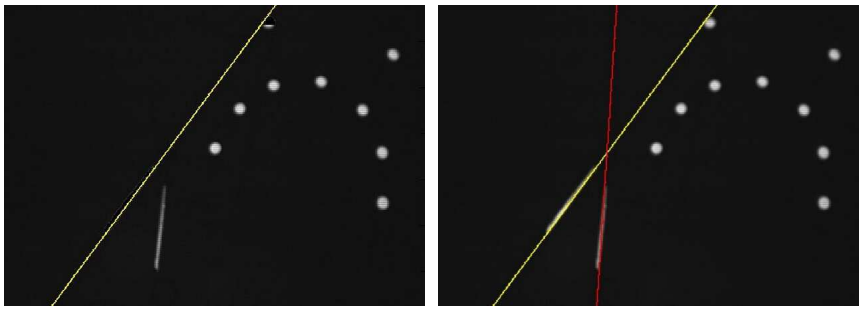
The success of the split Hough Transformation is displayed in figure 3.11. The left image presents the image I_{new} created through the elimination process and the Hough line along which the elimination was done. The intensive leg is nearly completely blackened. On the right side both computed Hough lines are shown in the original image.
Without separate line detection only one line is found (see figure 3.4). Now both legs are determined but the calculated equations are not optimal. The lines pass the pixel areas representing the legs rather diagonal than straight. This problem of exact line retrieval was already mentioned earlier (see 3.1.2) and the solution to it is described in the next section.

3.2.2 Least Squares Method

Least Squares is a fitting procedure that takes a number of points (x_i, y_i) and chooses the line that minimizes the distance between this line and each point. Here the Total Least Squares method is implemented which uses the perpendicular distance instead of vertical distance. This paragraph presents the algorithm for the procedure and describes its application in combination with the Hough Transformation to give a better approximation of the line equations for the scissors.

Algorithm

A line l is represented by the equation $ax + by + c = 0$. The perpendicular distance for an



(a) Image after Elimination along yellow Hough line

(b) Image with both Hough lines

Figure 3.11: Results of split Hough Transformation

arbitrary point (u, v) to l is given by $|(au + bv + c)|$ if $a^2 + b^2 = 1$. This can be proved by looking at the vectorial formular for the distance between a point **p** and a line given by a normal vector **n** with length one and an arbitrary point on the line **a** as $\mathbf{n} \circ (\mathbf{p} - \mathbf{a})$. For l and (u, v) this equation is

$$distance = \begin{pmatrix} a \\ b \end{pmatrix} \circ \begin{pmatrix} u - 0 \\ v - \frac{c}{b} \end{pmatrix} = au + bv + c$$

and $a^2 + b^2 = 1$ because **n** has length one.
In order to minimize the sum of all perpendicular distances between l and k points (x_i, y_i) the following sum has to be minimized:

$$E = \sum_{i=1}^{k} (ax_i + by_i + c)^2$$

Differentiating E with respect to c gives

$$\frac{\partial E}{\partial c} = 2 \sum_{i=1}^{k} (ax_i + by_i + c)$$

which must be zero for the minimum. Hence $c = -(a\overline{x} + b\overline{y})$ where $\overline{z} = \frac{\sum z_i}{k}$. Substituting this result back into the sum results in the homogeneous equation system

$$E = \sum_{i=1}^{k} (a(x_i - \overline{x}) + b(y_i - \overline{y}))^2 = \left|\left| \begin{pmatrix} x_1 - \overline{x} & y_1 - \overline{y} \\ x_2 - \overline{x} & y_2 - \overline{y} \\ \dots & \dots \\ x_n - \overline{x} & y_n - \overline{y} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \right|\right|^2 = ||A\mathbf{h}||^2 = 0$$

For the solution of this equation system some thoughts are helpful:
$E = ||A\mathbf{h}||^2$ shall be minimized, but the solution $\mathbf{h} = \mathbf{0}$ is not wanted. A non-zero solution is sought and hence a constraint is set: $||\mathbf{h}|| = 1$. Using this constraint the original minimization equation is enlarged to

$$E_\lambda = \mathbf{h}^T A^T A\mathbf{h} + \lambda(\mathbf{h}^T\mathbf{h} - 1)$$

where λ is called the Lagrange multiplier [8].
Looking at the derivative which shall be zero

$$\frac{\partial E_\lambda}{\partial \mathbf{h}} = 2A^T A\mathbf{h} + 2\lambda\mathbf{h} = 0$$

the relationship $A^T A\mathbf{h} = -\lambda\mathbf{h}$ is retrieved. Substituting this back into E_λ gives

$$E_\lambda = -\mathbf{h}^T\lambda\mathbf{h} + \lambda\mathbf{h}^T\mathbf{h} - \lambda = -\lambda$$

So for a minimal λ the whole equation is minimized. As the relationship $A^T A\mathbf{h} = -\lambda\mathbf{h}$ is an eigenvector problem the searched λ is the smallest eigenvalue of the equation and its corresponding eigenvector $\mathbf{h}$ is the solution for the minimization problem E.
The matrix $A^T A$ can be calculated as

$$\begin{aligned} A^T A &= \begin{pmatrix} x_1 - \overline{x} & \dots & x_n - \overline{x} \\ y_1 - \overline{y} & \dots & y_n - \overline{y} \end{pmatrix} \begin{pmatrix} x_1 - \overline{x} & y_1 - \overline{y} \\ \dots & \dots \\ x_n - \overline{x} & y_n - \overline{y} \end{pmatrix} = \\ &= \begin{pmatrix} \sum_{i=1}^n (x_i - \overline{x})^2 & \sum_{i=1}^n (x_i - \overline{x})(y_i - \overline{y}) \\ \sum_{i=1}^n (x_i - \overline{x})(y_i - \overline{y}) & \sum_{i=1}^n (y_i - \overline{y})^2 \end{pmatrix} = \\ &= n \cdot \begin{pmatrix} \overline{x^2} - \overline{x}^2 & \overline{xy} - \overline{x}\,\overline{y} \\ \overline{xy} - \overline{x}\,\overline{y} & \overline{y^2} - \overline{y}^2 \end{pmatrix} \end{aligned}$$

So the eigenvalue problem is

$$\begin{pmatrix} \overline{x^2} - \overline{x}\,\overline{x} & \overline{xy} - \overline{x}\,\overline{y} \\ \overline{xy} - \overline{x}\,\overline{y} & \overline{y^2} - \overline{y}\,\overline{y} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \mu \begin{pmatrix} a \\ b \end{pmatrix}$$

The eigenvector corresponding to the smallest eigenvalue is $(a, b)^T$ and c can be calculated using the equation $c = a\overline{x} + b\overline{y}$. With a, b, c the line is determined for which the perpendicular distances of all given points (x_i, y_i) are minimal.

Application in the line detection process
Least Squares finds the line that best fits a set of pixels in the sense of perpendicular distances. With split Hough Transformation only diagonal approximations for the scissors legs could be computed (compare section 3.2.1). Now a combination of these two methods is the final improvement for the line detection process. $hough3(r, t, gvt, et)$ proceeds as follows. The first five steps are similar to $hough2$ except a small modification in step three:

1. Apply $hough1(r, t, gvt, 1.0)$ to the given image I.
2. Calculate the equation for the line l represented by the accumulator maximum (Θ, ρ).
3. Eliminate bright pixels along the line l to create a new image I_{new}.
   The proceeding is the same as for $hough2$ but in addition to the simple elimination of the image pixels with values greater than et, these values are stored in a new image array I_{elim1}. This array has the size of I and if $(x, y) \in I$ is blackened then its old value val_{xy} is stored at position $(x, y) \in I_{elim1}$.
4. Apply $hough1(r, t, gvt, 1.0)$ to the modified image after elimination I_{new}.

5. Calculate the equation of the second line l' with the new parameter maxima (Θ', ρ').

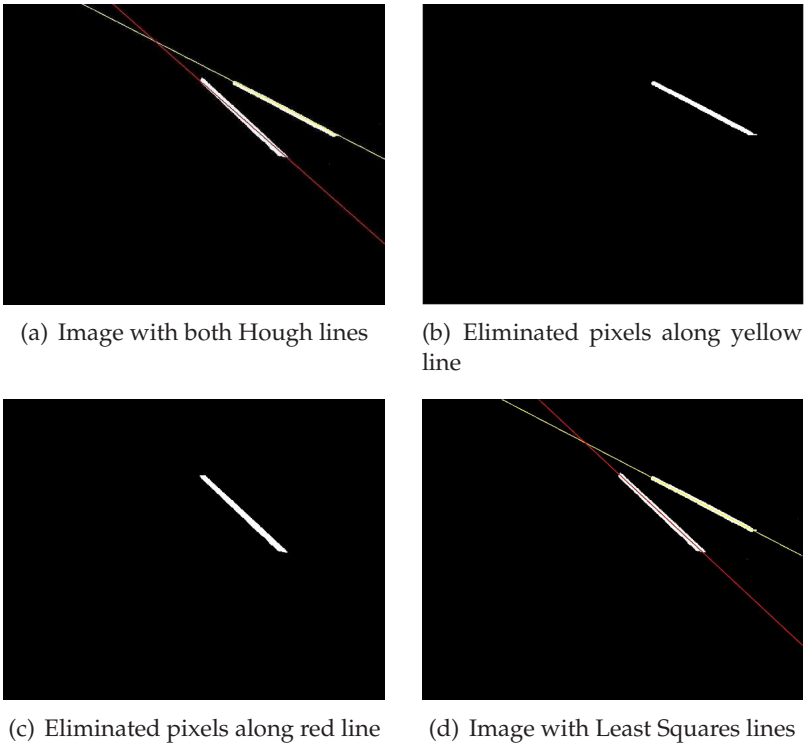
6. Eliminate the bright pixels along l' analogous to step three. Again the erased values are stored in an array I_{elim2}.

7. Apply Least Squares to I_{elim1} and I_{elim2}.
   Both image arrays contain only the pixels around the previously calculated Hough lines representing the scissors legs. I_{elim1} describes the first (more intensive) leg, I_{elim2} the second one. All other cells in the image arrays are 0. Obviously the number of pixels and the accuracy of the leg representation are strongly depending on the exactness of the calculated Hough lines and the value of et. If they are appropriate the Least Squares procedure computes a very good approximation for the two line equations.

For a better understanding of the proceeding some images of the intermediate arrays and the results are shown in the next paragraph.

Results

The previous sections presented the first results of plain Hough transformation (compare figure 3.4) and the improvements with a split approach (see figure 3.11). Now the amelioration through a final Least Squares smoothing are depicted. The calculated Hough lines are



(a) Image with both Hough lines

(b) Eliminated pixels along yellow line

(c) Eliminated pixels along red line

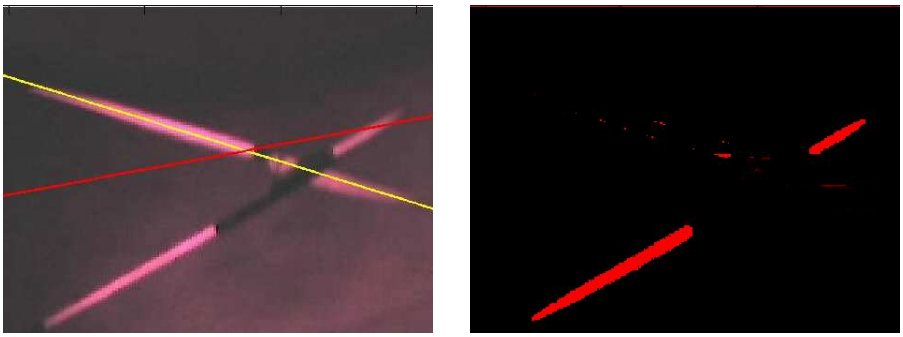
(d) Image with Least Squares lines

Figure 3.12: Least Squares Procedure

shown in figure 3.12 (a). During the process of elimination (steps three and six of $hough3$) the image arrays I_{elim1} and I_{elim2} are created (compare figure 3.12 (b) and (c)). These pixel

areas are now used to calculate Least Squares Lines as described in step seven of $hough3$. The resulting lines are shown in subimage 3.12 (d). The improvement achieved with the additional method is clearly visible although its performance depends on the initial data given by the Hough Transformation.

Problems
Least Squares offer a great amelioration potential for the computation of the line equations. Still it has a disadvantage which can cause severe problems in the case of line detection: it is very sensitive to noise. Figure 3.13 (a) shows Least Squares lines calculated with algorithm $hough2$. Although the Hough lines fit the pixel areas perfectly one Least Squares line is computed completely wrong. The reason for this can be found by looking at the elimination array for the line (see 3.13). A single bright pixel line in the top row of the image array causes this problem. The additional noisy pixels are also considered for the minimization of the perpendicular distances and hence the best fit is the resulting line equation. In our case this line resulted from a badly cut screen shot of the infrared image.



(a) Least Squares Lines (b) Eliminated pixels

Figure 3.13: Least Squares Sensitivity to Noise

3.3 Interlaced Images and Marginal Cases

In the process of line detection some special cases may appear. The initial data for the algorithm can be corrupted in case of interlaced images. Furthermore some marginal cases are possible during the line retrieval which interfere a correct computation.

3.3.1 Deinterlacing

Interlacing
Interlacing is a video display technique in which all odd-numbered scan lines are updated in one sweep of the screen and all even-numbered scan lines in the next. The idea of refreshing alternate lines halves the number of lines to update in one sweep. Additionally it takes advantage of the human eye's tendency to average subtle differences in light intensity and

reduces flickering because for a given update rate the top and bottom of the screen are redrawn twice as often as if the scan simply proceeded from top to bottom in a single vertical sweep.

Interlaced Images

When the scene taken by an interlacing camera moves very fast, the alternate update can lead to images where the interlacing technique becomes visible for the human eye. The difference between the generated even and odd lines is too large to be averaged. Figure 3.14 (a) shows a movie frame where the motion of the pair of scissors led to interlaced images.

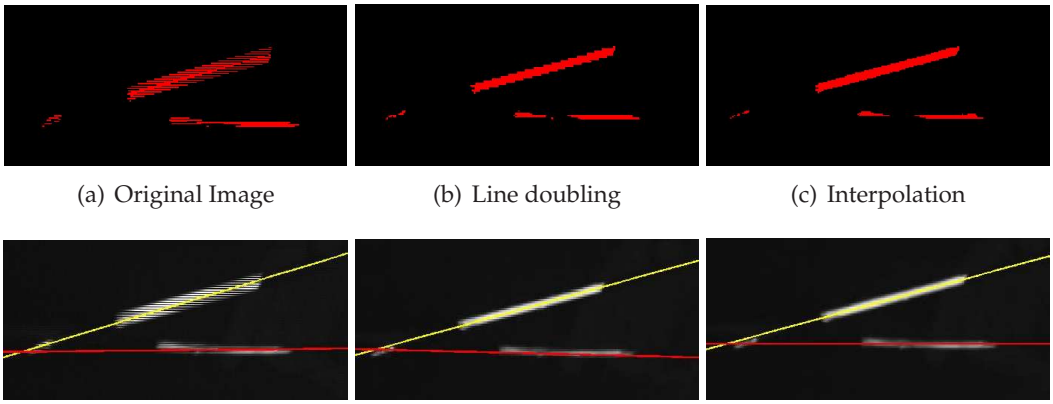
Deinterlacing Approaches

The easiest way of avoiding interlaced images is to prevent their origin. This can be done by either using a non-interlacing camera or by allowing only slow motion of the object so that alternate lines do not differ decidedly.
Images which are corrupted can be deinterlaced by trying one of the following approaches to achieve smoother line changes.

1. Line Doubling
   As all odd and all even image rows fit together two new images can be created by either doubling every odd line or every even line. So for the odd case lines one and two of the new image correspond to the first line in the original image. Line three and four to the third line, line five and six to the fifth one and so on.

2. Interpolation
   For this method only the odd rows are taken and the missing rows in the new image are filled by interpolating all the pixels around the actual one. The value of pixel (x, y) is calculated as $val_{x,y} = \frac{1}{6}(val_{x-1,y-1} + val_{x-1,y} + val_{x-1,y+1} + val_{x+1,y-1} + val_{x+1,y} + val_{x+1,y+1})$, taking the adjacent values of the row above and below.

Examples of images created with these methods and their effects on the quality of the line detection are displayed in figure 3.14.



(a) Original Image (b) Line doubling (c) Interpolation

(d) Original Image (e) Line doubling (f) Interpolation

Figure 3.14: Deinterlacing: chosen pixels and resulting lines

Results
The corruption of interlaced images can be smoothed by the deinterlacing approaches but the results are not always sufficient to allow a correct line detection. For the further calculations cameras are taken which do not generate interlacing problems.

3.3.2 Marginal Cases

There are several possible cases in which no (complete) line detection is possible. Hence a special treatment is needed and some kind of indicating information must be passed on to avoid further calculations with wrong data.

1. The image contains no feature points or only one feature point.
   This can happen when the gray value threshold is set too high so that no points are taken into account for the calculation. Or the image really contains no or just one point for example because of occlusion.

2. After the first elimination the resulting image contains no feature points or only one feature point.
   Again this might be a question of threshold or only one line is visible in the original image.

3. The elimination does not result in adequate data for the Least Squares calculation.
   The eliminated pixels are taken to compute the least squares line equations. If no pixels are eliminated the Least Squares method can not be applied. Reasons for this are a too high elimination threshold or wrongly calculated Hough lines. And if the image contains noise which becomes part of the elimination array it can corrupt the resulting equation (compare Least Squares Problems 3.2.2).

The ideal case for further calculations would be two calculated Least Squares lines giving an optimum of accuracy. Still the next computations could be executed if at least one line equation was available for each line. So both Hough equations are needed and additional Least Squares data would ameliorate the results. In case of less information, one or no lines, no triangulation is possible and the 3D reconstruction can not be done.
In order to catch all these cases an indicating variable is used to identify the situation and to decide whether further steps are possible or not.

3.4 Final Data for the next Step

After improving the plain Hough Transformation the two line equations can be calculated very accurately. Now for the next step, the triangulation, two points for each line are necessary to go on with the proceeding. For unambiguous data a scheme is needed which identifies line one and respectively line two in both images as the same leg of the scissors. Otherwise the stereo data would not fit together. Furthermore the marginal cases mentioned above (3.3.2) need to be treated.

Unambiguous leg identification
The method for leg numbering used in this project has three steps. First the actual position

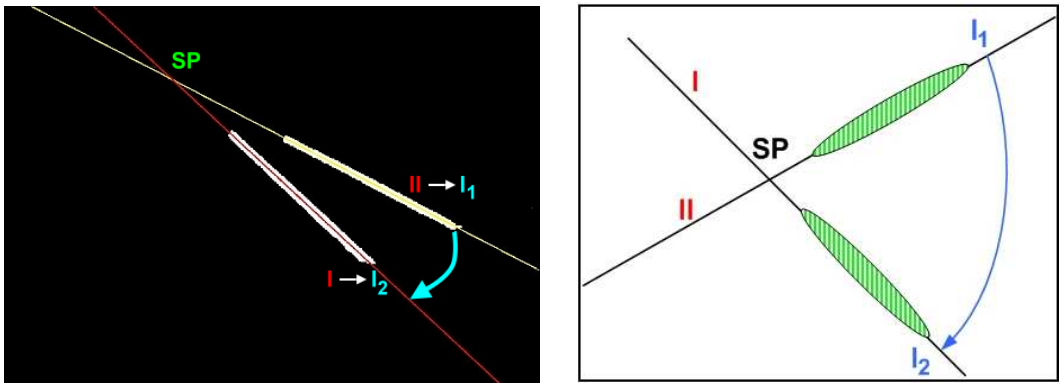
of the scissors legs on the detected lines is determined. Which two branches of the line cross represent the legs of the instrument? After that the upper line on the right side of the point of intersection is found. Is it the first detected line or the one which was found after the first elimination? This information suffices to get an idea of the principle structure of the scene and the legs can then be numbered.

1. Determining the leg positions
   For this step buckets are used to accumulate the gray values along each of the four parts of the line cross starting from the point of intersection. In the example (see figure 3.15 (a)) the buckets for the two right branches contain larger values than the ones for the left parts.

2. Determining the upper line
   Using the line equations the upper line on the right side of the point of intersection can be found by comparing the slopes of both lines. In figure 3.15 the red Latin numbers indicate the order in which the lines were detected, hence line II is the upper one on the right side of the point of intersection.

3. Principal structure of the scene
   With the buckets and the upper line number the principal structure of the scene is given as an arbitrary line cross in which the two branches of the legs and the line detection order is known. Figure 3.15 (b) displays the information derived above. Using this knowledge the unambiguous line numbering can now be done.

4. New numbering of the two lines
   The new numbering is done clock-wise using the smaller angle between the two scissor legs. In the example 3.15 (a) the angle is indicated and the new numbering is given through the blue Arabic numbers. In the example case the new ordering is inverse to the old one. Of course it could also be possible that this process does not change the line numbers.



(a) Image with old and new line numbering (b) Derived principal structure of the scene

Figure 3.15: Unambiguous Leg Numbering

Data which is passed on

Now that the lines are identified the data which shall be handed on to the triangulation computations can be determined. Two points per line are necessary to determine the plane passing through the camera center and the line. So for both lines l_1, l_2 the point of intersection **SP** and one additional point on each line $\mathbf{X}_1 \in l_1$, $\mathbf{X}_2 \in l_2$ are satisfactory. The unambiguous line numbering assures that line l_1 in one image corresponds to line l_1 in another image of the same scene, hence corresponding plane equations can be calculated. It is important to notice that the $\mathbf{X}_i$ in one image do not correspond to the $\mathbf{X}_i$ determined for another image. The only known point correspondence is given by the point of intersection.

For the point determination the marginal cases must be considered, too. If both Least Squares lines were successfully calculated the three points are chosen using the Least Squares equations. In case only the Hough lines could be found, they are used for the point determination. For any other case the three points cannot be identified and no further processing steps are executed.

4 Projective Geometry and Camera Calibration

Computing extrinsic and intrinsic Parameters for the two Camera Systems

In the process of 3D positioning of the pair of scissors the lines on the images have to be interpreted to get relationships between pixel data and real data on the one hand and right and left image data on the other hand. These relationships are expressed through intrinsic and extrinsic camera calibration data.
First an overview of the theory on projective geometry and calibration is given. Then the process of retrieving the intrinsic and extrinsic parameters is described both for the stereo camera system and the RAMP system. Their usage will then follow in the section on triangulation, in the proceeding part (5.3).

4.1 Projection and Calibration Theory

This section shall introduce the reader to projective geometry and the theory behind camera calibration. First the difference between projective and euclidian geometry is discussed. Then various coordinate systems and transformations between them are presented. The focus here lies on the camera model used in this project. Hence only the necessary coordinate systems, transformations and parameters are mentioned. Further information on these subjects can be found in [10].

4.1.1 Euclidian and Projective Geometry

Euclididan vs. Projective Geometry
Euclidian geometry deals with angles and shapes of objects. It is intuitive and familiar but for projections it is not sufficient. In Euclidian space only rotations and translations are allowed because a lot of measures must be invariant: length, angles, ratio of length, parallelism, incidence and cross ratio. A projection cannot hold all these constraints as length

and angles are not preserved and parallel lines may intersect. If a larger class of transformations shall be allowed the invariants must be reduced. Projective geometry is the most general geometry where only incidence and cross ratio stay invariant. Euclidian geometry is a subset of it.

2D Homogeneous Coordinates
A point in the Euclidian plane is given by (x, y). For a representation in the projective plane a third coordinate 1 is added: $(X, Y, 1)$. Scaling is unimportant here, so $(kX, kY, kW) = (X, Y, W)$ for any $k \neq 0$ which gives equivalence classes of coordinate triples, the so called homogeneous coordinates. Given an arbitrary triple the original point in the Euclidian plane can be retrieved by dividing the whole vector by the third coordinate.
Hartley and Zisserman [10] state that the Euclidian space is translated into the projective space by adding points at infinity. Looking at the homogeneous coordinates the points at infinity arise through the triples $(X, Y, 0)$ which have no corresponding points in Euclidian space. Dividing by the last coordinate gives $(X/0, Y/0)$ which is infinite.
Lines can also be written in homogeneous coordinates. A 3D line $ax + by + c = 0$ is represented by the vector $(a, b, c)^T$. Again $k(a, b, c)^T$ and $(a, b, c)^T$ represent the same line for any $k \neq 0$. Using this notation a point $\mathbf{X}$ lies on a line $\mathbf{l}$ if and only if $\mathbf{X}^T l = 0$, the intersection of two lines l, l' is the point $\mathbf{X} = l \times l'$ and the line given through the points $\mathbf{X}$ and $\mathbf{X'}$ is $l = \mathbf{X} \times \mathbf{X'}$.

Degrees of Freedom
The degrees of freedom specify the number of values which must be provided to specify a certain object. For a 2D point for example two values are necessary. For a line also two parameters are needed, the independent ratios $a : b$ and $b : c$. Hence both objects have two degrees of freedom.

2D Projective Transformations
Planar projective transformations on homogeneous vectors are represented by a non-singular 3×3 matrix H. The transformation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

does not change if H is multiplied by an arbitrary non-zero scale factor. In consequence H is called a homogeneous matrix. For lines the transformation is $l' = H^{-1T} l$.
The important specializations of a projective transformation and their properties are listed in the following:

1. **Isometries**

$$H = \begin{pmatrix} \epsilon cos\theta & -sin\theta & t_x \\ \epsilon sin\theta & cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

   with $\epsilon = \pm 1$. Here Euclidian transformations are modeled with three degrees of freedom: one for the rotation R and two for the translation T. If $\epsilon = -1$ then the orientation of the object is reversed otherwise it is preserved.

2. **Similarities**

$$H = \begin{pmatrix} s\,cos\theta & -s\,sin\theta & t_x \\ s\,sin\theta & s\,cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

The scalar s represents the isotropic scaling and gives the fourth degree of freedom for this planar similarity transformation.

3. **Affinities**

$$H = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{T} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

Here objects are no longer shape-preserved but deformed. The resulting affine transformation has six degrees of freedom.

4. **Projectivities**

$$H = \begin{pmatrix} A & t_x \\ \mathbf{v}^T & 1 \end{pmatrix}$$

The most general form of a projective transformation has eight degrees of freedom. Here only the two invariants cross-ratio and incidence are left. With the vector $\mathbf{v}^T$ in the third column a point at infinity can now be mapped to a finite point.

4.1.2 Coordinate Systems

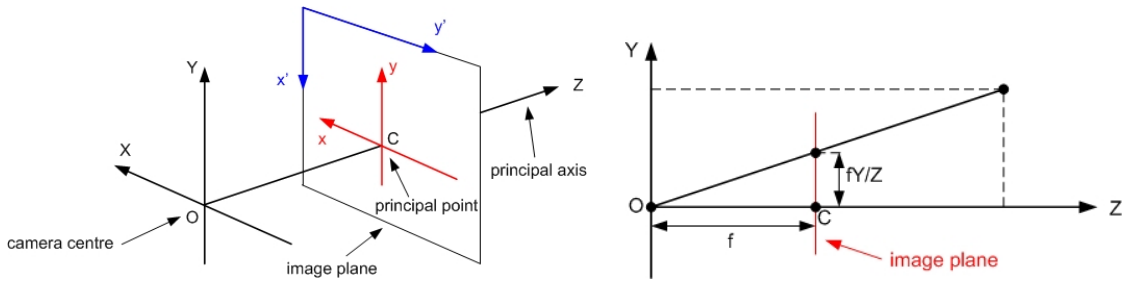
In the process of projecting a point in 3D space onto a 2D plane and displaying it as a pixel image three different coordinate systems are used. Figure 4.1 (a) presents them all together.

Camera Coordinate System
First the point is given in the 3D camera coordinate system (X, Y, Z). Its origin is the camera center and the three coordinate axes form a right handed system with the Z-axis, also called the principal axis, pointing to the field of view. The base unit of the coordinate system is millimeters (mm).

Image Coordinate System
The projection of the 3D point onto the image plane is represented in the 2D image coordinate system (x, y). The directions of the $x-$ and $y-$axis are the same as in the camera coordinate system. The origin, the so called **principal point C**, lies on the intersection of the principal axis with the image plane. Again the base unit is mm.

Computer Coordinate System
Finally the image plane is represented as a 2D pixel array in the computer coordinate system (x', y'). The origin is now the upper left corner of the image and the $x'-$ and $y'-$axis point down and right. Therefore a point (r, c) in the computer coordinate system indicates the row and column of the pixel it describes in the image array. Here the unit is one pixel.

For a complete listing of coordinate systems the so called world coordinate system is missing. Sometimes the coordinate system of the objects in the world is different from the camera coordinate system and an extra transformation between them is necessary. Here no additional world coordinate system is given.



(a) Three coordinate systems (b) Pinhole projection

Figure 4.1: Pinhole camera geometry

To find the relationships between the lines in the camera images and the scissors in 3D space several transformations are necessary. They allow to move a point representation through the different coordinate systems.

4.1.3 Transformation between two Camera Coordinate Systems

To find the pair of scissors in space at least two camera images are needed. In order to use the stereo data the relative position of the two camera coordinate systems to each other must be given. Such a transformation is represented by a 3D rotation matrix R and a translation T. Hence a point $\mathbf{X} = (X, Y, Z)^T$ in the first camera coordinate system O_1 can be transformed into the second coordinate system O_2 by

$$\mathbf{X}_{trans} = R\mathbf{X} + T$$

Rotation Matrices and Euler angles

There are different ways of expressing rotations in space. Rotations about the x-, y- and z-axis in a clockwise direction when looking towards the origin result in three matrices [20].

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & sin\alpha \\ 0 & -sin\alpha & cos\alpha \end{pmatrix}$$

$$R_y(\beta) = \begin{pmatrix} cos\beta & 0 & -sin\beta \\ 0 & 1 & 0 \\ sin\beta & 0 & cos\beta \end{pmatrix}$$

$$R_z(\gamma) = \begin{pmatrix} cos\gamma & sin\gamma & 0 \\ -sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

According to Euler's rotation theorem any rotation can be described as a composition of rotations about three axes. Thus the 3×3 matrix

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha)$$

with so called Euler angles $\alpha,\ \beta,\ \gamma$ covers all possible rotations in 3D space.
An efficient method for computing rotation matrices is given by Rodrigues' rotation formula [19]. R corresponds to a rotation by an angle $\theta \in \mathbb{R}$ about a fixed axis given by the unit vector $\omega = (\omega_x, \omega_y, \omega_z)^T \in \mathbb{R}^3$.

$$R = e^{\tilde{\omega}\theta} = \mathbf{I} + \tilde{\omega} sin\theta + \tilde{\omega}^2(1 - cos\theta)$$

with the identity matrix **I** and the antisymmetric matrix

$$\tilde{\omega} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & \omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

The Matlab Camera Calibration Toolbox uses this formula for the extrinsic rotation parameter (see section 4.2.1).

Quaternions
Finally quaternions shall be introduced as another possibility for the description of rotations [14], [15]. Quaternions are an example of the class of hypercomplex numbers discovered by William Rowan Hamilton. They are representable as a sum of real and imaginary parts

$$H = a \cdot 1 + bi + cj + dk$$

where *i, j, k* satisfy the fundamental equation of quaternion algebra

$$i^2 = j^2 = k^2 = ijk = -1$$

Quaternions can also be interpreted as a scalar plus a vector:

$$a = a_1 + a_2 i + a_3 j + a_4 k = (a_1, \mathbf{a})$$

where $\mathbf{a} = (a_2, a_3, a_4)$. This leads to the expression of a rotation about the unit vector ω by the angle θ as

$$R = (s, \mathbf{v}) = (cos(\frac{1}{2}\theta), \omega sin(\frac{1}{2}\theta))$$

Using quaternions to represent rotations has several advantages over the Euler angle representation [15]. Euler angles define a rotation as a composition of three independent rotations about coordinate axes. This causes an ambiguity of representation, different combinations are possible for the same 3D rotation. In addition the rotations can be carried out in the base coordinate system or in the already rotated one. Further, the independence characteristic of Euler angles breaks down when the second Euler angle becomes $90°$, resulting in a loss of one degree of freedom. This phenomenon is called the gimbal lock. The quaternion representation involves only the angle and the vector of rotation hence the mentioned problems do not occur here.

4.1.4 Transformation between Camera and Image Coordinates

Through the pinhole projection a point in 3D space $(X, Y, Z)^T$ is projected onto the image plane as $(fX/Z, fY/Z, f)^T$. So the x and y coordinates of the point in the image coordinate system are

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} fX/Z \\ fY/Z \end{pmatrix}$$

where f, the so called focal length, is the distance of the image plane to the projection center. Figure 4.1 emphasizes this mapping.
Using homogeneous coordinates the upper projection can be written as

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

In compact notation this is $\mathbf{x} = P\mathbf{X}$ where $\mathbf{X}$ is the homogeneous 4-vector in the camera coordinate system, $\mathbf{x}$ is its projection in the image plane and $P = diag(f, f, 1)[I|\mathbf{0}]$. To get the image plane coordinates $(x, y)^T$ from above $\mathbf{x}$ has to be homogenized to a 3D vector with last entry 1. This gives the same result $\mathbf{x}/Z = (fX/Z, fY/Z, 1)^T$.

Image Distortion
The assumption above has been that a linear model is an accurate one for the imaging process. For real (non-pinhole) lenses this assumption will not hold. Improper lens and camera assembly lead to positional errors, so called distortion. Radial distortion is the most important error in this case which is caused by imperfect lens shape. To face it, the image measurements have to be corrected. As the distortion takes place during the initial projection onto the image plane the correction must be done in that phase of the computation. The actual projected point is related to the ideal point on the image plane by a radial displacement which can be modelled as

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \dot{(}1 + \sum_{i=1}^{3} \kappa_i r^{2i})$$

with distorted coordinates $(x_d, y_d)^T$ and ideal pinhole projection coordinates $(x, y)^T$. The radial distortion is expressed through the coefficients κ_i and $r^2 = (x - x_c)^2 + (y - y_c)^2$, the distance of the point from the radial distortion center $(x_c, y_c)^T$. In case not only the lens shape is imperfect the positional error consists of a radial and a tangential error (see Figure 4.2).

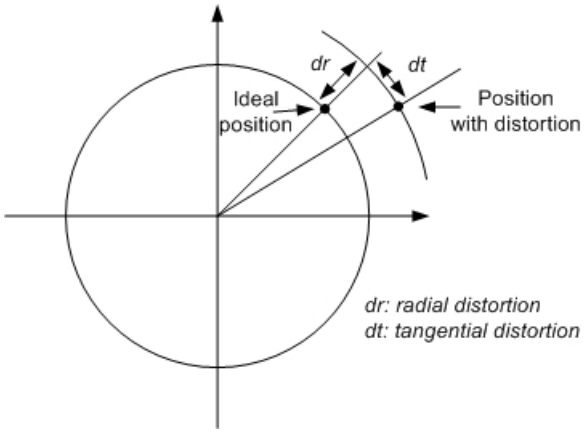


Figure 4.2: Radial and tangential distortion

The Matlab Camera Calibration toolbox models the additional tangential deformation as

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} x_d \\ y_d \end{pmatrix} + \Delta_{tangential} \text{ with } \Delta_{tangential} = \begin{pmatrix} \kappa_4(2xy) + \kappa_5(r^2 + 2x^2) \\ \kappa_5(2xy) + \kappa_4(r^2 + 2y^2) \end{pmatrix}$$

4.1.5 Transformation between Image and Computer Coordinates

For the final transformation to computer coordinates several parameters need to be taken into account. The computation for homogenous vectors is given as

$$\begin{pmatrix} x_{pixel} \\ y_{pixel} \\ 1 \end{pmatrix} = K \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} = \begin{pmatrix} sx/dx & alpha & c_x \\ & 1/dy & c_y \\ & & 1 \end{pmatrix} \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix}$$

with the calibration matrix K. dx and dy express the distance in millimeters between two adjacent pixels in x and y direction. sx is the scale factor and $C = (c_x, c_y)^T$ is the principal point. $alpha$ is called a skew parameter which is zero in most cases. $alpha \neq 0$ would only occur if the $x-$ and $y-$axes in the image were not perpendicular.
So the upper matrix-vector-multiplication can be interpreted as a conversion of millimeter values to pixel values (using dx and dy), a scaling in $x-$direction according to the factor sx and a translation of the resulting point so that the new coordinate system has its origin in the upper left corner of the image. To achieve this the principal point is added to the negative values of the converted and scaled distorted point.
For a correct representation in the $x' - y'-$coordinate system the two vector entries must be switched and multiplied by -1 as the axes of the image coordinate system and the computer coordinate system are vice versa and have opposite directions (see figure 4.1 (a)).

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -y_{pixel} \\ -x_{pixel} \end{pmatrix}$$

4.1.6 Parameter Estimation and Reprojection Error

Parameter Estimation
Throughout this section a number of projective transformations are used. The problem of parameter estimation refers to the computation of theses transformations, for example the matrix which maps points in 3D space onto the image plane.
For a general approach a set of point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x'}_i$ between two images shall be considered. The problem is to compute a 3×3 matrix H, a 2D projective transformation, which satisfies $H\mathbf{x}_i = \mathbf{x'}_i$ for each i (see also [10]). H has 9 entries but is defined only up to scale. Thus the total number of degrees of freedom here is 8. In order to fully specify H at least four point correspondences are necessary as each of them has two degrees of freedom, corresponding to the $x-$ and $y-$coordinates. So for four points an exact solution for H is possible.
In general only approximate solutions can be computed as the points are measured with noise. Hence the best possible transformation shall be found, a matrix which minimizes some cost function. These functions either try to minimize an algebraic error or a geometrical distance.

Direct Linear Transformation Algorithm (DLT)
The DLT is a very simple algorithm for determining H and is given as a basis for further improvements.
The equation $H\mathbf{x}_i = \mathbf{x'}_i$ is expressed as the vector cross product $\mathbf{x'}_i \times H\mathbf{x}_i = 0$. Writing $\mathbf{x'}_i = (x'_i, y'_i, w'_i)^T$ and the j-th row of H as $\mathbf{h}^{jT}$, the cross product is given as

$$\mathbf{x'}_i \times H\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}^{3T}\mathbf{x}_i - w'_i\mathbf{h}^{2T}\mathbf{x}_i \\ w'_i\mathbf{h}^{1T}\mathbf{x}_i - x'_i\mathbf{h}^{3T}\mathbf{x}_i \\ x'_i\mathbf{h}^{2T}\mathbf{x}_i - y'_i\mathbf{h}^{1T}\mathbf{x}_i \end{pmatrix}$$

With $\mathbf{h}^{jT}\mathbf{x}_i = \mathbf{x}_i^T\mathbf{h}^j$ this gives a set of three equations

$$\begin{pmatrix} \mathbf{0}^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & \mathbf{0}^T & -x'_i\mathbf{x}_i^T \\ -y'_i\mathbf{x}_i^T & x'_i\mathbf{x}_i^T & \mathbf{0}^T \end{pmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

Only two of the three equations are independent, hence the set is reduced to

$$\begin{pmatrix} \mathbf{0}^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & \mathbf{0}^T & -x'_i\mathbf{x}_i^T \end{pmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

For i point correspondences this are i equations of the form $A_i\mathbf{h} = \mathbf{0}$ with A_i 2×9 matrices. In general the system is overdetermined and more than four point correspondences are given. In addition the solution $\mathbf{h} = \mathbf{0}$ is of no interest and must be avoided. So a minimal solution to $||A\mathbf{h}||$ is searched with $||\mathbf{h}|| = 1$ as a second constraint. As shown in section 3.2.2 the unit eigenvector of A^TA corresponding to the smallest eigenvalue gives the solution.

Cost functions
A number of cost functions can be used to be minimized in order to compute H for an overdetermined system. Some examples are given here. The used notation is $\mathbf{x}$ for the measured coordinates, $\hat{\mathbf{x}}$ the estimated ones and $\bar{\mathbf{x}}$ the true values of the points.

1. **Algebraic distance**
   The DLT algorithm minimizes $||A\mathbf{h}||$. $\epsilon = A\mathbf{h}$ is called the residual vector. Each point correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x'}_i$ contributes a partial error vector ϵ_i to ϵ. The norm of the algebraic error vector ϵ_i is called the algebraic distance:

   $$d_{alg}(\mathbf{x'}_i, H\mathbf{x}_i)^2 = ||\epsilon_i||^2 = \left|\left| \begin{pmatrix} \mathbf{0}^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & \mathbf{0}^T & -x'_i\mathbf{x}_i^T \end{pmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} \right|\right|^2$$

   The algebraic error for the complete set is

   $$\sum_i d_{alg}(\mathbf{x'}_i, H\mathbf{x}_i)^2 = \sum_i ||\epsilon_i||^2 = ||\epsilon||^2$$

2. **Geometric distance**
   The disadvantage of the algebraic distance is that it has no geometrical or statistical

meaning. The geometric distance minimizes the difference between the measured and the estimated image coordinates. Now two error considerations are possible. Either only the measurements of one image are considered to be noisy. Then the transfer error is

$$\sum_i d(\mathbf{x'}_i, H\bar{\mathbf{x}}_i)^2$$

Or measurement errors occur in both images. Hence they should also be minimized in both images. This can be done using a forward (H) and backward (H^{-1}) transformation and sum the geometric errors:

$$\sum_i d(\mathbf{x}_i, H^{-1}\mathbf{x'}_i)^2 + d(\mathbf{x'}_i, H\mathbf{x}_i)^2$$

3. **Reprojection error**
   The two previous functions tried to minimize an error function. For the reprojection error not only $\hat{H}$ is estimated but in addition a correction for each point correspondence is sought to get perfectly matching points $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}'_i$ which minimize the total error function

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2$$

   subject to $\hat{\mathbf{x}}'_i = \hat{H}\hat{\mathbf{x}}_i$ for all i.

The difference between symmetric error and reprojection error is displayed in figure 4.3. In the upper case of symmetric error the points $\mathbf{x'}$ and $H\mathbf{x}$ do not correspond perfectly but for the reprojection error the newly estimated points $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ do fit consummately.

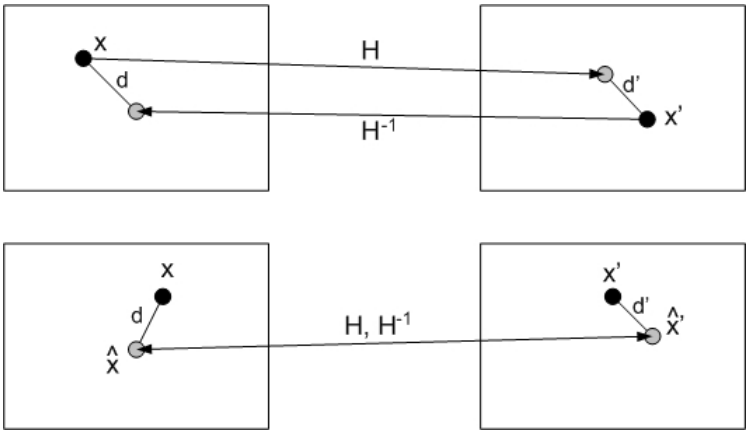


Figure 4.3: Comparison of symmetric error (upper) and reprojection error (lower) for estimation

The Matlab Camera Calibration Toolbox minimizes the reprojection error for the computation of the projection of 3D world points onto the image plane.

4.2 Calibration Data of the Stereo System

For this project two different systems were used for image acquisition. For the calibration of the stereo camera system separate non-infrared image series are necessary which are

processed by the Matlab Camera Calibration Toolbox to compute the intrinsic and extrinsic parameters of the system.

4.2.1 Matlab Camera Calibration Toolbox

Camera Calibration
The Matlab Camera Calibration Toolbox allows camera calibration using images of a chessboard (see figure 4.4). The toolbox offers Matlab functions to load the images, to extract the grid corners from each image and to do the calibration with the collected data.

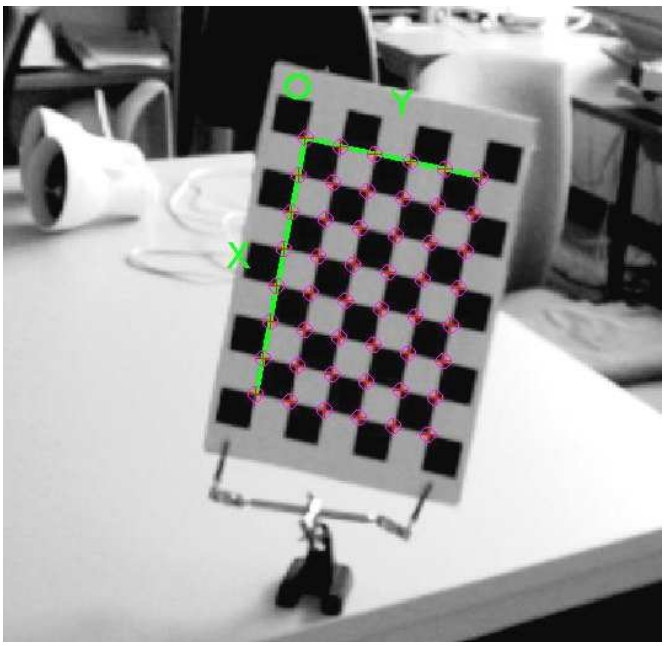


Figure 4.4: Chessboard for Calibration: + extracted image points, ∘ reprojected grid points

To extract the grid corners the user has to click on the four extreme corners of the chessboard. The inner corners are then computed automatically. If the computation does not fit the real corners the image is distorted. This problem can be handled by giving an initial guess for distortion which will then be used by the system to ameliorate the data.
After the corner detection has been done for all images the calibration can be started. It is done in two steps: first initialization and then non-linear optimization. The initialization computes a closed-form solution for the calibration parameters not including any lens distortion. The non-linear optimization step minimizes the total reprojection error over all the calibration parameters (see theory on estimation and reprojection error 4.1.6).
This process results in the intrinsic parameters of the camera: focal length $fc(1), fc(2)$, principal point $cc(1), cc(2)$, skew coefficient $alpha_c$ and distortion coefficients $kc(1)..kc(5)$. Hence all the information is available for the transformation of a 3D point in space to its corresponding pixel in the image. The back transformation can also be computed except for the scaling factor. The correspondence between these parameters and the calibration theory notation is shown in table 4.1.

Toolbox notation	Theory notation (4.1.4)
$fc(1)$	$\frac{fsx}{dx}$
$fc(2)$	$\frac{f}{dy}$
$cc(1)$	cx
$cc(2)$	cy
$alpha_c$	0
$kc(1), kc(2), kc(5)$	radial distortion: $\kappa_1, \kappa_2, \kappa_3$
$kc(3), kc(4)$	tangential distortion: κ_4, κ_5

Table 4.1: Correspondences in notation

Stereo Calibration
The toolbox also offers a function for stereo calibration. It takes the intrinsic data of the left and right camera and starts by estimating the extrinsic parameters R and T which characterize the relative location of the right camera with respect to the left. Given a point in space $\mathbf{X} = (X, Y, Z)^T$ R and T are defined such that the coordinate vectors of $\mathbf{X}$ in the left and right coordinate system $\mathbf{X}_L$ and $\mathbf{X}_R$ are related to each other through the transformation $\mathbf{X}_R = R\,\mathbf{X}_L + T$.
These initial values are then optimized together with the extrinsic parameters of both cameras as to minimize the reprojection error on both cameras for all calibration grid locations.
The resulting data consists of two 3D vectors T and om. The rotation matrix is $R = rodrigues(om)$ (compare Rodrigues' rotation formula 4.1.3).

4.2.2 Calibration Results

The results of this work step are the extrinsic and intrinsic camera parameters. They allow to completely describe the stereo scene. The relative position of the two cameras , and therefore of the two given images, is expressed by the transformation (R, T) consisting of rotation and translation. Figure 4.5 displays this scene. Using (R, T) the left camera coordinate frame can be transformed into the right one. The three colored planes indicate the positions of the chessboard used for camera calibration. The intrinsic parameters enable the computation of the 3D metric coordinates of a pixel point on the image. So a pixel point $(x_{pixel}, y_{pixel})^T$ is transformed to metric coordinates (image coordinate system) as $(x_d, y_d)^T$ and the distortion is removed to give $(x, y)^T$ using the equations in section 4.1.4 and 4.1.5. $(x, y)^T$ is on the image plane in camera coordinates therefore it can be expanded to a 3D-vector with third coordinate f, the focal length, the distance of the image plane to the camera center.
With this information the next task of triangulation can be faced.

4.3 Calibration Data of the RAMP System

4.3.1 Intrinsic Parameters

For the RAMP System we used the existing calibration data. The intrinsic parameters for the calibration matrix and the distortion coefficients are known (see table 4.1). The notation

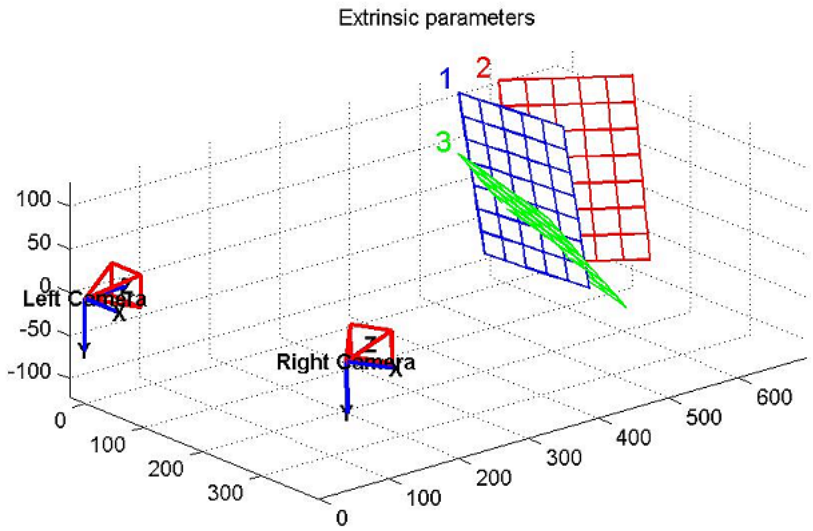


Figure 4.5: Visualization of the extrinsic parameters (plotted by Matlab Camera Calibration Toolbox)

matches the one used in the section on calibration theory (4.1.4). The only difference to the right column of the correspondence table is that in contrast to the Calibration Toolbox the RAMP System does only consider radial distortion $\kappa_1, \kappa_2, \kappa_3$, the tangential one is not taken into account.

4.3.2 Extrinsic Parameters

The RAMP System has only one camera whose relative position to a Zero-Coordinate-System located in the ring of fiducials is computed in real time constantly while using the system. The transformation from the actual camera coordinate system to the Zero-Coordinate-System is displayed as rotation and translation in the given images (see 4.6) and also sent to the RAMP client. To get a stereo system two images at different positions of the camera are taken, while the position of the scissors is fixed relative to the world coordinate system. The relative position of the two camera images to each other can be calculated using the computed transformations with respect to the Zero-Coordinate-System. If R_1 and T_1 are the transformation data for the first scene shot and R_2 and T_2 for the second one then the transformation of a point $\mathbf{X}_1$ from the first camera coordinate system into the second can be expressed as

$$\mathbf{X}_2 = R_2\dot{(}R_1^{-1}\dot{(}\mathbf{X}_1 - T_1)) + T_2 = R_2\dot{(}R_1^{T}\dot{(}X_1 - T_1)) + T_2$$

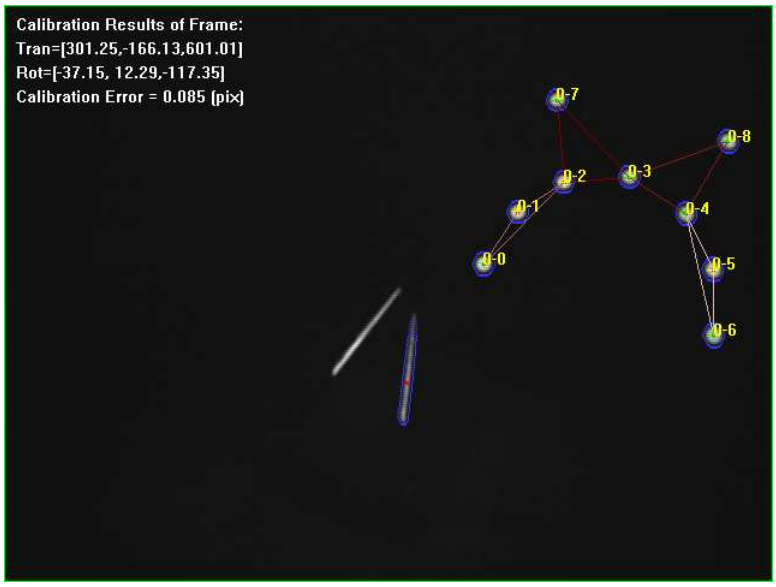


Figure 4.6: Image from the RAMP System with transformation data

5 Line Triangulation

Calculation of the 3D Line Position from Image and Calibration Data

---

This chapters covers the process of determining the actual 3D position of the pair of scissors starting from 3D points on the stereo image planes. The set of points of one coordinate system will be moved into the second frame according to the transformation encoded in the extrinsic parameters. Then plane equations are determined for each line on each image. The corresponding planes are then intersected to give the equation of the 3D lines. Finally the intersection of the two resulting 3D lines is calculated to serve as a criterion for quality and accuracy of the calculation.
Again the first section starts with a theoretical introduction, then the implemented approach is described in detail. The chapter finishes with an overview on the triangulation results.

5.1 Projective Geometry in 3D

Section 4.1.1 already introduced the basics of planar projective geometry. Now these ideas are expanded to the 3D space.

3D Homogeneous Coordinates
A point in space (X, Y, Z) is represented in homogeneous coordinates - similar to the 2D case - by the 4-vector $(X_1, X_2, X_3, X_4)^T$ with $X_4 \neq 0$. As in the 2D case the inhomogeneous coordinates can be retrieved by dividing by the last coordinate:

$$X = \frac{X_1}{X_4},\ Y = \frac{X_2}{X_4},\ Z = \frac{X_3}{X_4}$$

Again the points at infinity are given by the 4-vectors with last coordinate zero.
In 2D points and lines are dual, they have analogous representations. In 3D this analogy is given for points and planes. A plane $\pi : \pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0$ can be written in homogeneous coordinates as $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)^T$. Hence $\pi^T \mathbf{X} = 0$ if $\mathbf{X}$ lies on the plane. Both planes and points in 3D have three degrees of freedom.

3D Projective Transformations
Points in space are transformed similar to the 2D case as $\mathbf{X}' = H\mathbf{X}$ and for planes it follows $\pi' = H^{-1T}\pi$.

Line representation
Lines in 3D can be represented by two points defining the line or by one point on the line and an orientation vector. Another point-independent representation of lines is given by the Plücker coordinates. It is described in section 5.2.4.

5.2 Triangulation Theory

Stereo triangulation is the process of reconstructing a 3D scene given their left and right image projections. The most common way is to use point correspondences for the calculation. But in this project the only correspondence known in the stereo images is the point of intersection of the two legs. This single pair does not suffice for a point triangulation. Therefore a different approach for reconstruction is taken: line triangulation.

5.2.1 Line triangulation

If a line is projected to lines in two 2D views $\mathbf{l}, \mathbf{l}'$ the original line in 3D-space can be reconstructed by back-projecting $\mathbf{l}$ and $\mathbf{l}'$ to give two planes and intersecting the planes as figure 5.1 shows. Each line in the two images is represented by two points. This is necessary as the

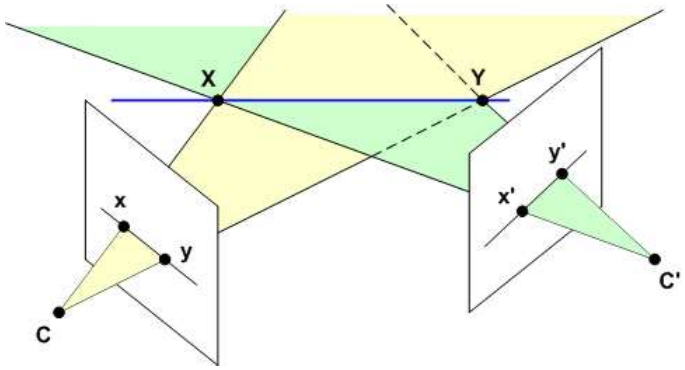


Figure 5.1: Line triangulation

transformation between the pixel computer coordinate system and the camera coordinate system is given for points. So in order to get the line representations in camera coordinates these points are transformed according to the equations in section 4.1.4 and 4.1.5.

5.2.2 Plane intersection

After that these points are used to determine the plane equations:

$$\begin{aligned} n_1x + n_2y + n_3z &= d \\ \mathbf{n}^T \cdot \mathbf{x} &= d \end{aligned}$$

The normal vector to the plane $\mathbf{n}$ is given through the cross product of two vectors. With $\mathbf{X}_1$, $\mathbf{X}_2$ in the image and $\mathbf{C}$ the camera center the normal vector is $\mathbf{n} = \overrightarrow{\mathbf{CX}_1} \times \overrightarrow{\mathbf{CX}_2}$. The right

side can be determined by inserting a point of the plane $\mathbf{X}_1$ in the equation, $d = \mathbf{n} \cdot \mathbf{X}_1$.
To get the 3D line equation the planes of corresponding lines in the left and right images are intersected. This leads to the following system of equations:

$$\begin{pmatrix} \mathbf{n}_{left} \\ \mathbf{n}_{right} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} d_{left} \\ d_{right} \end{pmatrix}$$

The solution to this underdetermined system is the searched line equation.

5.2.3 Line intersection

As a pair of scissors shall be reconstructed in 3D space there are two line equations which can be calculated in the way described above. To have a criterion for the accuracy of the computation these two lines can be intersected to approximate the original point of intersection of the scissors legs. Again an equation system needs to be solved.

$$\begin{pmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ -\lambda_2 \end{pmatrix} = \mathbf{P}_2 - \mathbf{P}_1$$

with $\mathbf{X} = \mathbf{P}_i + \lambda_i \mathbf{z}_i \ , \ i = 1, 2$ the two line equations.

5.2.4 Plücker line representation

In the paragraph above a line is represented by $l = \mathbf{P} + \lambda \mathbf{z}$ where $\mathbf{P}$ is an arbitrary point on the line and $\mathbf{z}$ is its orientation in space. This kind of representation is very intuitive but has the disadvantage of being dependant on the choice of $\mathbf{P}$ out of an infinite set of possible points which makes the representation arbitrary. With Plücker matrices a line can be represented independently without any arbitrary factor.

Plücker Matrices
The Plücker matrix for a 3D line defined by two points with homogeneous coordinates $\mathbf{A} = (a_1, a_2, a_3, 1)^T$, $\mathbf{B} = (b_1, b_2, b_3, 1)^T$ is the skew-symmetric matrix

$$\begin{aligned} L &= \mathbf{A}\mathbf{B}^T - \mathbf{B}\mathbf{A}^T = \\ &= \begin{pmatrix} a_1b_1 & a_1b_2 & a_1b_3 & a_1 \\ a_2b_1 & a_2b_2 & a_2b_3 & a_2 \\ a_3b_1 & a_3b_2 & a_3b_3 & a_3 \\ b_1 & b_2 & b_3 & 1 \end{pmatrix} - \begin{pmatrix} b_1a_1 & b_1a_2 & b_1a_3 & b_1 \\ b_2a_1 & b_2a_2 & b_2a_3 & b_2 \\ b_3a_1 & b_3a_2 & b_3a_3 & b_3 \\ a_1 & a_2 & a_3 & 1 \end{pmatrix} = \\ &= \begin{pmatrix} 0 & a_1b_2 - b_1a_2 & a_1b_3 - b_1a_3 & a_1 - b_1 \\ & 0 & a_2b_3 - b_2a_3 & a_2 - b_2 \\ & & 0 & a_3 - b_3 \\ & \iddots & & 0 \end{pmatrix} \end{aligned}$$

The diagonal elements of L are zero, the last column (and respectively the last row) contain the line orientation and the other three elements above and below the diagonal represent the cross product $\mathbf{A} \times \mathbf{B} = (L_{2,3}, -L_{1,3}, L_{1,2}, 1)^T$. Hence the Plücker matrix has 6 parameters

which would imply five degrees of freedom. But the determinant $det(L) = 0$ and therefore the degrees of freedom are reduced to four which is the required number for a line in 3D space. Further properties of L are $rank(L) = 2$, it is independent of the choice of the line points **A** and **B** and a transformation H on L results in $L' = HLH^T$.
A line can also be defined by the intersection of two planes represented by the homogeneous 4-vectors **P** and **Q** which gives the so called dual Plücker matrix

$$L^* = \mathbf{P}\mathbf{Q}^T - \mathbf{Q}\mathbf{P}^T$$

Its properties are similar to the ones of L and the correspondence between L and its dual $L*$ is given through the ratios $l_{12} : l_{13} : l_{14} : l_{23} : l_{42} : l_{34} = l^*_{34} : l^*_{42} : l^*_{23} : l^*_{14} : l^*_{13} : l^*_{12}$.

Plücker Coordinates
The Plücker line coordinates are the six non-zero elements of the Plücker matrix $L = [l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}]$ which hold the equation $l_{12}l_{34} + l_{13}l_{42} + l_{14}l_{23} = 0$. A 6-vector only corresponds to a line in space if this equation is valid.

Join and Incidence
With the Plücker representation relations between lines, planes and points can be expressed very elegantly:
A plane through a point **X** and a line L is defined by $\pi = L^*\mathbf{X}$.
A point **X** lies on a line L if and only if $L^*\mathbf{X} = 0$. The intersection of a plane and a line is the point $\mathbf{X} = L\pi$. And two lines which are the joints of the points **A**, **B** and **A'**, **B'** are coplanar if and only if $det[\mathbf{A}, \mathbf{B}, \mathbf{A'}, \mathbf{B'}] = 0$. This determinant expands as

$$det[\mathbf{A}, \mathbf{B}, \mathbf{A'}, \mathbf{B'}] = l_{12}l'_{34} + l_{13}l'_{42} + l_{14}l'_{23} + l'_{12}l_{34} + l'_{13}l_{42} + l'_{14}l_{23} = (L|L')$$

In case the two lines are the intersections of the planes **P**, **Q** and **P'**, **Q'** then the lines intersect if and only if $det[\mathbf{P}, \mathbf{Q}, \mathbf{P'}, \mathbf{Q'}] = 0$ and $det[\mathbf{P}, \mathbf{Q}, \mathbf{P'}, \mathbf{Q'}] = (L|L')$.

5.2.5 Accuracy criteria

Now that the principal procedure of 3D reconstruction is given, criteria are needed to measure the accuracy of the calculation.
The four planes which are intersected to find the 3D position of the scissors result in two 3D line equations. In general these two lines will not have a common point of intersection. The reason lies in computation and measurement errors throughout the whole process from 2D line detection to 3D line intersection. Hence only a least squares solution $SP_{linetria}$ of the 3D point of intersection can be computed. Regarding the 2D images the point of intersection gives the only point correspondence in the left and right image of the scene. So these points **sp** and **sp'** can be used for a point triangulation resulting in another 3D point of intersection $SP_{pointtria}$. Again the solution will be a least squares one because of the problem of errors.
In the ideal case a 3D point **X** and its 2D representations **x** and **x'** form a plane in space going through the base line of the stereo system $\mathbf{C} - \mathbf{C'}$ (compare figure 5.2). For the scissors stereo system the computations and measurements would therefore be exact if the rays from the camera centers through the 2D points of intersection $\overrightarrow{\mathbf{Csp}}$ and $\overrightarrow{\mathbf{Csp'}}$ were coplanar and had a common point of intersection, the ideal $\mathbf{SP}_{3D}$. The closer the least squares solutions $SP_{pointtria}$, $SP_{linetria}$ and the rays $\overrightarrow{\mathbf{Csp}}$ and $\overrightarrow{\mathbf{Csp'}}$ are, the better is the computed approximation for the real scene.

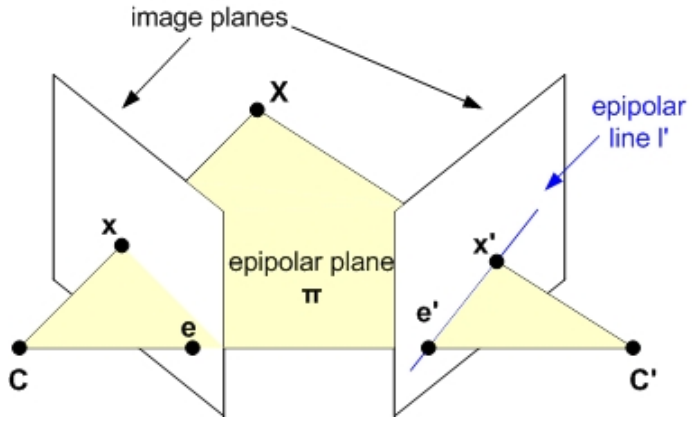


Figure 5.2: Epipolar plane

5.3 Practical Approach

For the practical approach of the triangulation the two different systems which were used for image acquisition must be taken into account. They have influence on the process of calculating the metric representation of the pixel points. After the metric data is determined the plane and line calculations can be done as described above.

5.3.1 Computing the camera coordinates of image pixel points

The Matlab Camera Calibration Toolbox offers a function for the transformation of image coordinates to camera coordinates: $normalize(\mathbf{X}, fc, cc, kc, alpha_c)$.
The pixel point **X** is transformed into metric camera coordinates by taking the inverse equations of section 4.1.5. First the distorted metric point $\mathbf{X}_d$ is calculated using the focal length fc and the principal point cc. Then the skew is undone ($alpha_c$). Finally the radial and tangential distortion is removed with the given coefficients kc.
For the RAMP System this transformation process was not yet specified as a function and therefore needed to be implemented. The procedure $ramp\_normalize(\mathbf{X}, extrinsic\ Ramp\ Data)$ runs through the same steps as its Toolbox equivalent. It calculates the metric coordinates using the resolution parameters dx, dy and the scale factor sx. As the RAMP System does not consider skew this step is skipped. And the final undistortion process does only take into account radial distortion.

5.3.2 Plane Equations and Intersections

To get the plane equations the two lines in each image are represented by three points the point of intersection $\mathbf{SP}_{pixel}$ and an arbitrary point on each line $\mathbf{X1}_{pixel}$, $\mathbf{X2}_{pixel}$. These points are then transformed into the camera coordinate frame resulting in the 3D-vectors **SP**, **X1**, **X2**. Now four plane equations can be determined:

$$p1_{left}: \ \mathbf{n1}_{left} \cdot \mathbf{X} = d1_{left}$$

$$p2_{left}: \ \mathbf{n2}_{left} \cdot \mathbf{X} = d2_{left}$$
$$p1_{right}: \ \mathbf{n1}_{right} \cdot \mathbf{X} = d1_{right}$$
$$p2_{right}: \ \mathbf{n2}_{right} \cdot \mathbf{X} = d2_{right}$$

with

$$\mathbf{ni}_{left} = \overrightarrow{\mathbf{C}_{left}\mathbf{SP}_{left}} \times \overrightarrow{\mathbf{SP}_{left}\mathbf{Xi}_{left}}$$
$$\mathbf{ni}_{right} = \overrightarrow{\mathbf{C}_{right}\mathbf{SP}_{right}} \times \overrightarrow{\mathbf{SP}_{right}\mathbf{Xi}_{right}}$$

and $i = 1, 2$ for line one and line two in the left and right image plane.
The intersection of the planes from the left and right image of each leg give two 3D line equations, l_1 and l_2 which satisfy the following equation systems:

$$l_1: \begin{pmatrix} \mathbf{n1}_{left} \\ \mathbf{n1}_{right} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} d1_{left} \\ d1_{right} \end{pmatrix}$$

$$l_2: \begin{pmatrix} \mathbf{n2}_{left} \\ \mathbf{n2}_{right} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} d2_{left} \\ d2_{right} \end{pmatrix}$$

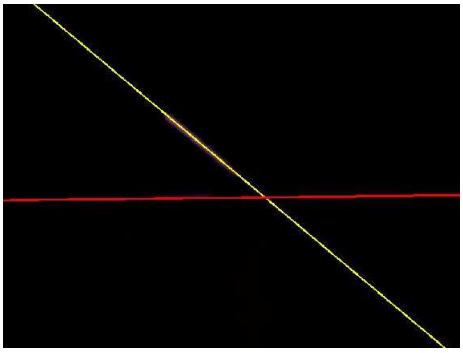
The right side of the equation system is $di_{left} = \mathbf{ni}_{left} \cdot \mathbf{SP}_{left}$ and $di_{right} = \mathbf{ni}_{right} \cdot \mathbf{SP}_{right}$, for $i = 1, 2$.
l_1 and l_2 are intersected to determine the 3D coordinate vector $\mathbf{SP}_{3D}$, the original point in space which is projected onto the image planes as $\mathbf{SP}_{left}$ and $\mathbf{SP}_{right}$. The equation system is given in section 5.2.3.

5.3.3 Verification and Accuracy

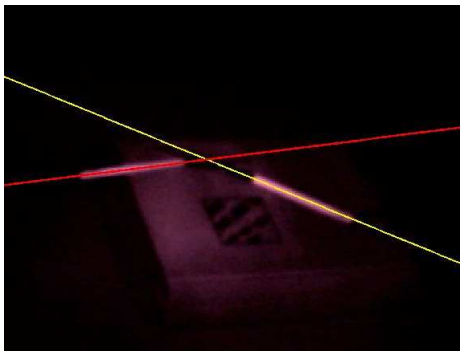
For the verification of the calculations the ideas of section 5.2.5 are used. The Matlab Camera Calibration Toolbox offers a function for point triangulation $[SP3D_{left}, SP3D_{right}] = stereo\_triangulation(SP_{left}, SP_{right}, intrinsic\ and\ extrinsic\ calibration\ parameters)$. The resulting 3D points represent the point of intersection in space in the left and right camera frame. A comparison with the point computed by plane and line intersection allows to evaluate the quality of the data and the result. For the RAMP System a similar function was implemented to provide the same possibilities for evaluation.
In addition the resulting points were visualized to give an idea of the complete 3D stereo scene, the ratios of distances between the cameras and the scissors. Exemplary images are shown in the following section.

5.4 Triangulation Results

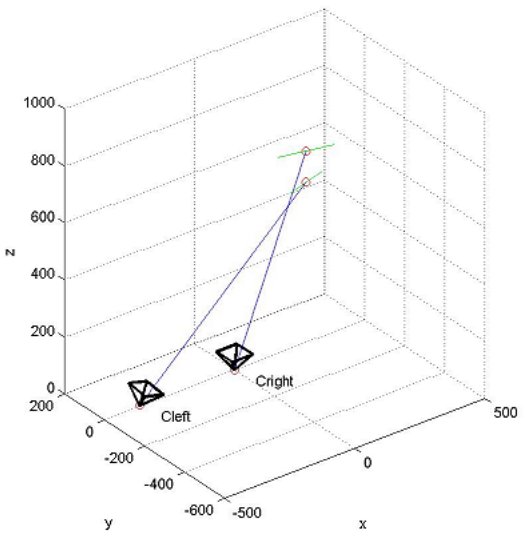
The results for triangulation express the problem of accuracy already mentioned in section 5.2.5 and 5.3.3. The general error in measurements leads to inexact line equations and calibration parameters. Hence a solution for the 3D reconstruction can only be approximated. The sensitivity of the result to errors is visible in figure 5.3. Here two example sets are shown.



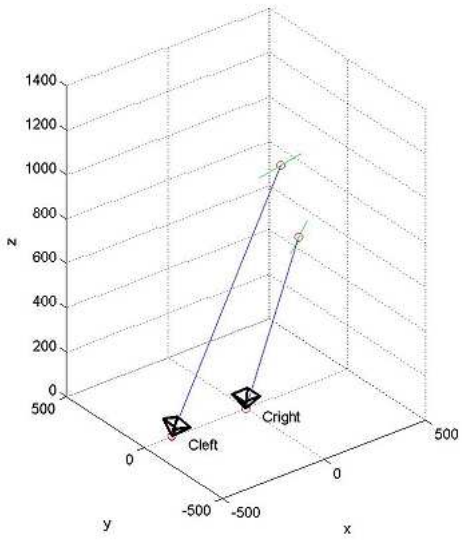
(a) Least Squares Lines



(b) Least Squares Lines



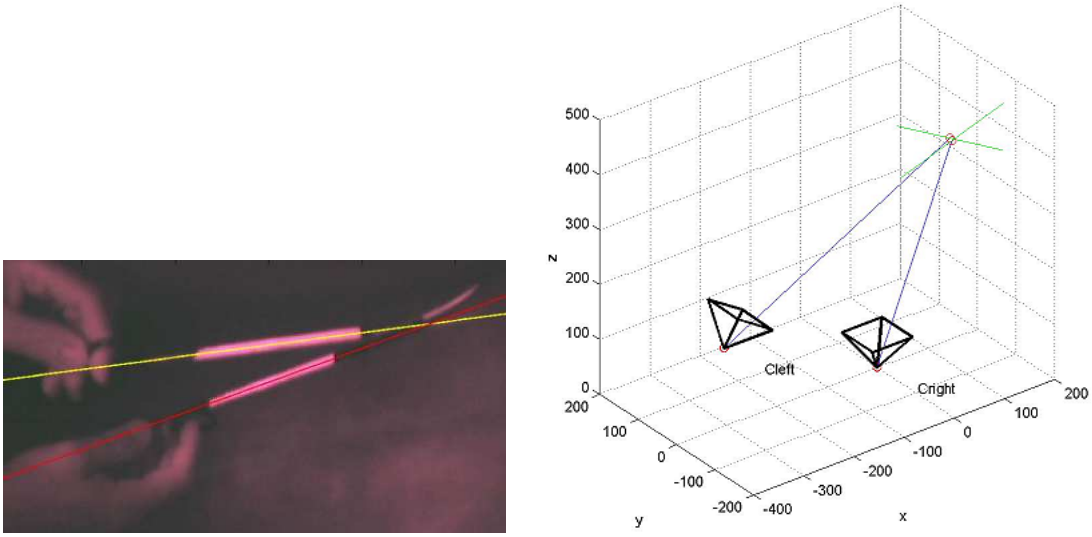
(c) Triangulation



(d) Triangulation

Figure 5.3: Bad Triangulation Results

Each pair displays the calculated Least Squares Lines (a),(b) for one of the stereo images and the 3D scene computed via triangulation (c),(d). In the first case (a) the pair of scissors is hardly visible in the infrared image. Thus the line equations are very imprecise approximations of the actual legs and the computed 3D reconstruction (c) can represent the real scene only roughly. In contrast the detected lines in image (b) meet the scissors quite accurately. Still the scene reconstruction (d) is insufficient. Reasons for this are the unequal illumination for this image series and an inaccurate camera calibration. Therefore the triangulation results can vary strongly.



(a) Least Squares Lines

(b) Triangulation

Figure 5.4: Good Triangulation Results

To verify the bad light conditions and the calibration exactness as cause for the errors another image series was generated. Here the focus lay on better calibration data, images where the chessboard was clearly visible. In addition the scissors were positioned closer to the camera to get larger views of the instrument. That way more pixels in the infrared images correspond to the legs and more information is available for an accurate line detection. The new series resulted in better 3D scene reconstructions for all tested images. An example is given in figure 5.4.

Altogether the various tests revealed several criteria for a good scene reconstruction. Good line visibility and accurate calibration data are essential for a satisfying triangulation result. Hence the scene illumination and the size of the scissors in the infrared images must be sufficient. Furthermore the position of the stereo cameras with respect to each other can influence the accuracy of the result. A rectangular viewing angle would be optimal but this implies a restricted tracking area [5]. For the case of scissors tracking in an operating room this does not cause any problems as here the desired working volume is rather small.

6 Tracking Possibilities

Tracking the Lines in a Series of Images

This chapter gives an overview on possible methods for optical tracking. 'Tracking' can express the process of object detection in single images as well as its tracing through a series of images. In addition a distinction can be drawn whether the object shall be tracked in 2D - hence on the image planes - or in 3D space.
During our project we did only implement the detection of an object in single images, in 2D. The following sections give an idea of what can be used for an efficient tracking in single images as well as in sequences, considering the 2D and the 3D case. First a brief introduction on 2D visual tracking is given. General questions are presented and the suitability for the tracking of scissors is discussed. After that the problem of noise is considered and the Kalman Filter will be introduced as a robust tool which is extensively used for 2D and 3D tracking in Computer Graphics [4].

6.1 Ideas for 2D Visual Tracking

Visual tracking can be described as the following problem: Use a previously identified target in an image with an initial configuration to estimate the state of the target in a sequence of subsequent images [9].
Now given this problem some questions should be considered in advance in order to gather more information on the situation and thus achieve good tracking results.

6.1.1 General questions

How can the target be identified?
For the tracking process it is helpful to know how many targets shall be tracked and how they can be distinguished. Possible properties could be the color, appearance or shape of the object(s). Obviously there is no unique global answer for the choice on these properties but often good approximations are possible.
In our case of scissors tracking the identification of the target is handled in the process of

image generation. Through the infrared filters and the retro-reflective foil only the legs of the scissors are (clearly) visible in the resulting images. Hence the color and the shape of the object allow its identification. We know that the legs are represented by line segments and that these areas of interest have decidedly larger gray values than the rest. In addition the angle between the legs could also hold as a constraint. When the instrument is in usage only a restricted range of angles is possible. This information is used in section 3.4 to find an unambiguous numbering of the legs. Finally the coplanarity of the two lines is a useful property which we use as an accuracy criterion (see section 5.2.5).

What are the viewing conditions?
Secondly details on changes in object pose, lighting or camera position can be useful for the design of the tracking system. Previously given information can be considered in the tracking process and allow to reach better state estimation. On the other hand the viewing conditions can influence the quality of the results. Hence the lighting and the camera position could be varied in such a way to optimize the overall tracking conditions.
The influence of lighting and geometrical configuration of the tracking setup can also be seen in this project. Bad illumination causes problems in the line detection process (see 3.1.3 and 5.4) and for the stereo triangulation the position of the cameras with respect to the scissors is of great importance to the quality of the reconstruction result. When the cameras are close together compared to the distance to the scissors, the planes which are intersected for triangulation are nearly parallel. This results in poor positional accuracy which is even aggravated by measurement uncertainty. Therefore for an optimal geometrical configuration the cameras should be far away from each other and have a nearly perpendicular view to the medical instrument [5].

6.1.2 Different approaches in Tracking

This section gives a short overview on several methods for 2D object tracking. The principle ideas are presented and the suitability for the case of scissors tracking is discussed.

Blob Tracking
Blob tracking tries to optimize the pixel selection based on some properties, for example intensity, color, texture or motion. It could also be referred to as segmentation-based tracking. The basic approach is to identify a segmentation function σ and an initial region of interest R_0 in the first image I_0. Then for every subsequent image the new region of interest R_i is computed by placing R_{i-1} in the new image I_i and applying $R_i = \sigma(R_{i-1})$ or by directly computing the segmentation function for the whole image $R_i = \sigma(I_i)$.
We use some kind of blob tracking for the initial line detection process in section 3.1.3. The gray value threshold gvt is the segmentation function σ. The region of interest in every image is given by all the pixels with gray values $\geq gvt$. The line equation retrieval is done via Hough transformation and Least Squares is used for optimization. As this segmentation-based tracking is too time consuming for application in image series it should only be used to identify the initial region of interest. For tracking in subsequent images other methods are more adequate.

Template-Based Region Tracking
When template-based region tracking is used, the direct appearance of the target is matched from image to image, pixel by pixel. Again an initial region of interest R_0 with location c_0

in the first image I_0 is used. For the image sequence some variation or correlation is used to determine u_i with the best match for the template R_{i-1} in I_i. The new template R_i is sampled around c_i in the actual image I_i.
This kind of tracking assumes a roughly constant appearance of the target from image to image. For small motions of the pair of scissors and a small range of variation for the angle between the legs these conditions could be met. The template could then consist of two line segments or two infinite intersecting lines. In case the leg angle was nearly constant one could even reduce the template to only the symmetry line of the two legs.

Snake or Spline-based Tracking
Parametrized snakes or spline-curves are the basis of mathematical curve description in most computer graphic applications. A spline curve in the parameter s consists of two curves, $x(s)$, $y(s)$, so called splines. Splines are piece-wise polynomial functions built of several spline segments (spans). Simple curves can be described with very few spans and for more complex curves the degree of the underlying polynoms can be increased [6].
Snake or spline tracking processes as follows: An initial contour C_0 is identified in the first image I_0. For every subsequent image in the series the contour is traversed along a set of discrete points $s_1, s_2, ...s_n$. At each s_j an edge is searched orthogonal to the contour. The normal distance to the contour is stored: d_j. Using the information $(s_1, d_1), (s_2, d_2), ..., (s_n, d_n)$ the new contour C_i can be estimated.
Snakes or splines are not necessary for the case of scissors tracking as they are far too complex structures. For the scissors description only two lines are needed. Still the concept of tracking can be applied in our case. The principal configuration could be as shown in figure 6.1. In order to determine d_j at each point s_j a template vector of gray values, representing

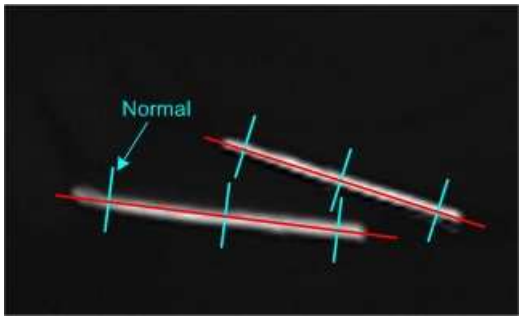


Figure 6.1: Configuration for line tracking

the profile of the line, can be moved along the line normals until an optimum of conformance with the image data is found. The template vector can be calculated from the first image in the series. For a template T and an image I the point of maximum conformance can be determined by minimizing the mathematical correlation (**correlation matching**):

$$\int_{x=0}^{\delta} I(x - x')T(x)dx \text{ with } 0 \leq \mathrm{x} \leq \delta$$

6.2 Stochastic Tracking: Considering Noise

Tracking an object requires the estimation of the object pose using its current position and measurements. No matter which sensors are used - mechanical, optical, acoustical or magnetic - the measurement data will be corrupted with noise which is mostly statistic in nature. Hence the problem of pose estimation is a stochastic one.
The Kalman Filter is one of the most-used and well-known mathematical tools for stochastic pose estimation [5]. It shall therefore be presented here in a brief way. For the understanding of the Kalman concepts some basic knowledge in stochastics is necessary. An introduction on this subject and some more details on the derivation of the formulas can be found in [4].

6.2.1 Kalman Filter

Essentially the Kalman Filter is a set of mathematical equations implementing prediction and correction steps. It is optimal in the sense that it minimizes the estimated error covariance when some presumed conditions are met. Because of its simplicity and robustness the Kalman Filter is subject of extensive research and application, for example in Computer Graphics. The filter even works well if the conditions necessary for optimal estimation are not met.

Estimation Process
The state $x \in \mathcal{R}^n$ of a discrete time-controlled process shall be estimated. The process model is given by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}$$

and the measurement model is

$$z_k = Hx_k + v_k$$

Hence the next process state x_k is related to the previous state x_{k-1} by an $n \times n$ matrix A. u_k is an optional control input related to x_k by the $n \times l$ matrix B. The $m \times n$ matrix H relates the state x_k to the measurement z_k. w_k and v_k represent the process and measurement noise. They are assumed to be independent and normally distributed: $p(w) \sim \mathcal{N}(0, Q)$ and $p(v) \sim \mathcal{N}(0, R)$ with process and measurement covariances Q and R.
For the estimation *a priori* and *a posteriori* state estimates $\hat{x}_k^-$ and $\hat{x}_k$ are defined. $\hat{x}_k^-$ is estimated with information given prior to step k and $\hat{x}_k$ is the estimate at step k given measurements z_k. Now *a priori* and *a posteriori* estimate errors and covariances can be defined as

$$\begin{aligned} \text{estimate errors:} \quad & e_k^- = x_k - \hat{x}_k^- \quad e_k = x_k - \hat{x}_k \\ \text{estimate covariances:} \quad & P_k^- = E[e_k^- e_k^{-T}] \quad P_k = E[e_k e_k^T] \end{aligned}$$

with $E[e]$ the expected value of e.

Discrete Kalman Filter Algorithm
The Kalman Filter algorithm estimates a process by using a form of feedback control. The process state at some time is estimated and then a feedback in form of measurements is obtained. This results in the "prediction"and "correction"cycle shown in figure 6.2. In the time update step the current state estimate is used to obtain the *a priori* state estimate for the

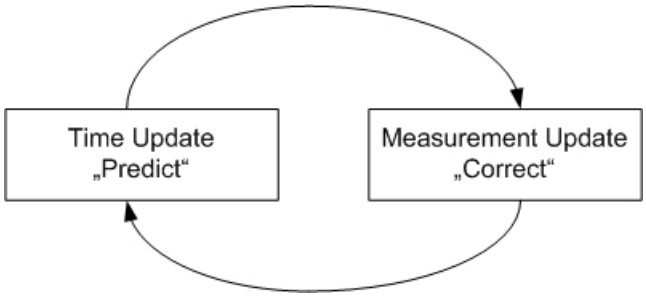


Figure 6.2: Discrete Kalman Filter Cycle

next process step. In addition the *a priori* estimate error covariance is updated.
Time update equations

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned}$$

The measurement update equations are responsible for the feedback. New measurements are used to obtain an improved *a posteriori* estimate.
Measurement update equations

$$\begin{aligned} K_k &= P_k^- H^T (HP_k^- H^T + R)^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \\ P_k &= (1 - K_k H)P_k^- \end{aligned}$$

First the so called Kalman gain K_k is computed. This $n \times m$ matrix minimizes the *a posteriori* error covariance equation $P_k = E[e_k e_k^T]$. Then an *a posteriori* state estimate $\hat{x}_k$ is computed as a linear combination of the *a priori* estimate $\hat{x}_k^-$ and a weighted difference between the measurement z_k and a measurement prediction $H\hat{x}_k^-$. This difference is called the innovation or the **residual**. It reflects the discrepancy between the predicted and the actual measurement. Finally the error covariance is updated using the Kalman gain and the *a priori* estimate error covariance P_k^-.

Extended Kalman Filter
The discrete Kalman Filter algorithm presented above works on linear stochastic difference equations for the process state and the measurements. In many situations these relationships might be non-linear. Here the so called Extended-Kalman-Filter (EKF), a Kalman filter which linearizes about the current mean and covariance, can be applied.
The process and measurement relations are expressed through non-linear functions: $x_k = f(x_{k-1}, u_k, w_{k-1})$ and $z_k = h(x_k, v_k)$. The estimation is linearized using partial derivatives of the process and measurement functions f and h.

6.2.2 Possible Application for this Project

In the current implementation Hough Transformation combined with a least squares method is used to detect the lines in the infrared images. This approach is suitable for the detection in single images but for the tracking over sequences of images it is too time-consuming (see

section 7.2.2). Hence the current line detection algorithm could be used for the initial computation of the line equations and for the further tracking of the scissors other approaches should be sought. For the case of stochastic state estimation the Kalman Filter was presented as a very robust and often-used tool. The principal ideas and formulas were presented in the previous section. The basic formulas of the filter are the process model and the measurement model. These models must be determined: necessary parameters must be included and adequate representations should be chosen which are favorable for the computation.
For the representation of the lines several approaches are described in section 5.2. Plücker coordinates for example provide an independent line representation without any arbitrary factors. For rotations in space different representations are presented in this thesis, too (see 4.1.3). Here quaternions are favorable because of their advantages compared to Euler angles and thus often used, for example in [13].

7 Conclusion

Overall Results and Possibilities for Optimization and further Research

The last chapter shall resume the results of the different steps in the project: which methods were successful, what we could achieve and where there is room for optimization. Additionally, we present some statistics on time and parameter choice for the implemented algorithms.

7.1 Overall Results

For this project the process of tracking scissors was divided in five major steps. Each of them is presented in a single chapter of this thesis. Here every part shall be summed up briefly to give an idea of the results, the problems which were met and the possibilities for optimization and further research.

7.1.1 Image Generation

Two systems were used for image generation, a stereo camera system and the RAMP system. Both created acceptable images for the further processing steps. The acquisition and storage of the images of the stereo camera system was optimized by using a stereo image acquisition software (2.1.3). Thus a larger data set of image pairs was available for testing and validating the implemented algorithms. Still the setup can be improved as for the illumination of the scene only conventional heating lamps were given. Infrared flash lights would provide a more evenly illumination and allow the generation of images with better quality.
The RAMP system uses such infrared flashes. Here the quality of the images is optimal but the original sense of usage of this system is a different one (2.2). It has only one infrared camera which is used for optical tracking. Therefore stereo infrared image pairs had to be created by moving the head-mounted display of the system around the pair of scissors and taking images at different positions. Because of the cumbersome image acquisition RAMP is rather unsuitable for our case.

Overall the part of image generation could be optimized by ameliorating the scene illumination of the stereo system. In addition the position of the cameras in the setup influences the quality of the triangulation results (see 5.4 and [5]) and could motivate further tests.

7.1.2 Line Detection

The basis for line detection is provided by the Hough Transformation algorithm (3.1.1). Several changes and additions were implemented in order to overcome some practical problems (3.1.2) and to optimize the results.
A gray value threshold was introduced to choose the pixels which shall be considered for the Hough Transformation and to reduce the time consumption of the calculation. Furthermore a weighted accumulation of the image pixels is used so that brighter pixels have more influence on the resulting Hough line than dark ones (3.1.3). Finally a Least Squares method is applied to optimize the line equation (3.2).
The line detection process gives quite nice and exact line equations for the legs of the scissors. Still, for good results the parameters for the detection function have to be chosen manually. The optimal gray value threshold is strongly varying because of unequal scene illumination (see 3.7). An optimized setup for image acquisition would solve this problem. Alternatively histogram-based methods could be used for the determination of adequate gray value thresholds. Furthermore the sensitivity to noise of the Least Squares method can cause problems which are not considered yet (3.13). When only the legs are visible in the infrared images no problems arise but otherwise additional algorithms are necessary to detect the noisy areas and discard them in further computations. RANSAC for example is such an algorithm which is able to cope with a large portion of outliers [10].
Finally the Hough Transformation is very time-consuming for usage in image series (see section 7.2.2). It is a suitable method for the first detection of the scissors in the infrared images. But for tracking the lines over a sequence of images other approaches are more adequate.

7.1.3 Camera Calibration

For camera calibration the Matlab Camera Calibration Toolbox was used [2]. The accuracy of the resulting intrinsic and extrinsic camera parameters is depending on the quality of the chessboard images. A good resolution and sharp images lead to exact corner approximation and therefore to satisfying data for the parameter estimation.
Still there will always be measurement errors and the computed values can only be an approximation of the real parameters. Hence the input images should be as high quality as possible and the more calibration data is available the better the estimation can be done.

7.1.4 Line Triangulation

The theory behind line triangulation is very simple but the actual results are strongly depending on the quality of the data. For the 3D reconstruction of the pair of scissors plane equations are computed whose intersection in space results in the desired 3D lines. In general the two lines representing the scissors legs won't intersect, hence only a least squares

solution is possible. The reason for that are measurement errors. Even with optimal image data the line detection and the camera calibration results will only approximate the real scene (5.4).
In addition the triangulation results are sensitive to the camera setup. If the object is far away from the cameras and the baseline between the two cameras is small the planes which are intersected for 3D reconstruction are nearly parallel. So the positional accuracy is even worst (6.1.1). A perpendicular view would be optimal for the camera setup but this results in a reduced viewing area. For tracking scissors in a medical environment this would not cause problems. The tracking region is limited and a good accuracy of 3D reconstruction and positioning of the instrument in the 3D world would only be necessary in a close environment around the area of operation. Whenever the surgeon does not use the scissors the accuracy demand decreases. So the cameras could focus on the surgery itself and allow clear images with sufficient instrument size.
Overall the previous steps need to be optimized in order to achieve good triangulation results, especially the image generation part.

7.1.5 Tracking

Tracking includes the detection and the tracing of an object. The part of line detection is already discussed above. For object tracking over a series of images various approaches are possible. Chapter 6 presents some ideas and methods which give a lot of material for further tests. Their time consumption and the accuracy of the results could be analyzed in order to identify convenient approaches for tracking in this special case.

7.2 Statistics

At the end, the implemented methods shall be analyzed for their time consumption, robustness and quality. With the second image series of the stereo camera system a large number of left and right images could be acquired. About 150 of them were used for the generation of the following statistics.

7.2.1 Analysis of the algorithms for a fixed set of parameters

To test the robustness of the algorithms, the line detection process was tried for the fixed parameters

$$(r, t, gvt, et) = (130, 180, 130, 130)$$

During all the line detection tests $(r, t) = (130, 180)$ emerged to be reasonable values for the Θ and ρ discretization. The gray value threshold and the elimination threshold were identified through trial detection runs with two images. Further details on these parameters and their influence on the computation can be found in section 3.1.3.

Results of the line detection
The line detection for a set of 144 infrared images gave the following results:
In two thirds of the cases the lines were detected correctly, giving a recall of 66.7 %. For about 20% of the images the gray value threshold was either too small or too high. When then gvt

is too low, the number of pixels which are considered for the transformation is too high. Noises are included in the transformation and the results useless. In the other case the lines cannot be detected because the amount of pixels is too small. The pixel areas representing the legs of the scissors cannot be identified. Figure 7.1 presents images for both cases of inappropriate gray value thresholds.
The precision of the line detection is given by the ratio of images with two correct detected lines to all images with two detected lines. For the test set it is 71.4 %.
A detailed listing of the test results is given in table 7.1 and table 7.2.

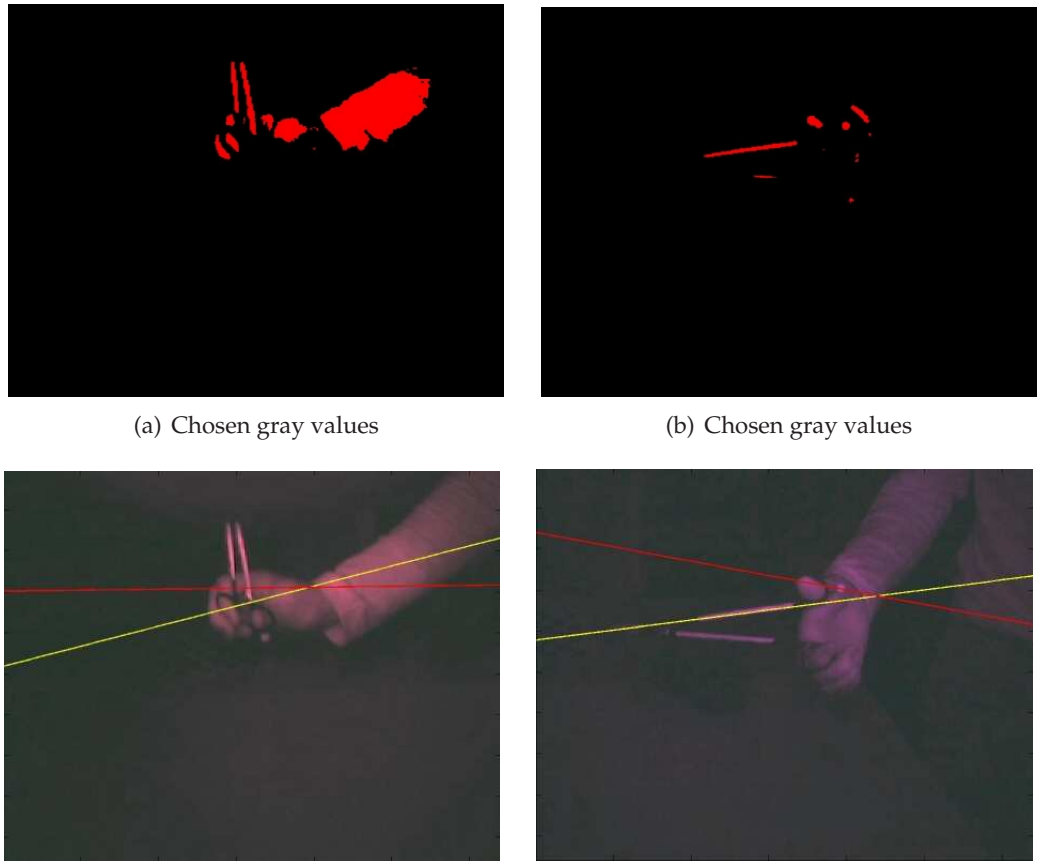
	successful line detection	gvt too high	gvt to small	leg angle too small
right cam	47	5	10	10
left cam	49	9	5	9
overall	96	14	15	19
percentage	**66.7 %**	9.7 %	10.4 %	13.2 %

Table 7.1: Recall of the line detection for fixed parameters

	2 of 2 lines correct	1 of 2 lines correct	0 of 2 lines correct
right cam	45	14	7
left cam	50	12	5
overall	95	26	12
percentage	**71.4 %**	19.6 %	9.0 %

Table 7.2: Precision of the line detection for fixed parameters

Triangulation
To do the 3D reconstruction stereo data is needed. Hence the line detection results must be satisfying for the left and the right images. For the set of test images only 35 pairs of left and right images achieved adequate line detection results for the line triangulation. A criterion for the quality of the triangulation result is the distance between the 3D point of intersection computed via line triangulation and the point of intersection resulting from point triangulation (compare section 5.2.5). Furthermore the two 3D lines computed via plane intersection should not be too far apart from each other. For 11 of the 35 image pairs the distances of the two computed 3D lines and the distance between the two points of intersection (derived via line triangulation and point triangulation) were less than 1 centimeter. Hence more than 30 percent of the test data resulted in millimeter accuracy for a camera setup of 30 centimeters baseline and approximately 50 to 100 centimeters distance of the pair of scissors to the cameras. Figure 7.2 shows the best and the worst case of 3D reconstruction for the 35 image pairs. The two distances are specified: d_1, the distance between the two points of intersection computed by line and point triangulation and d_2, the distance between the two reconstructed scissors legs.



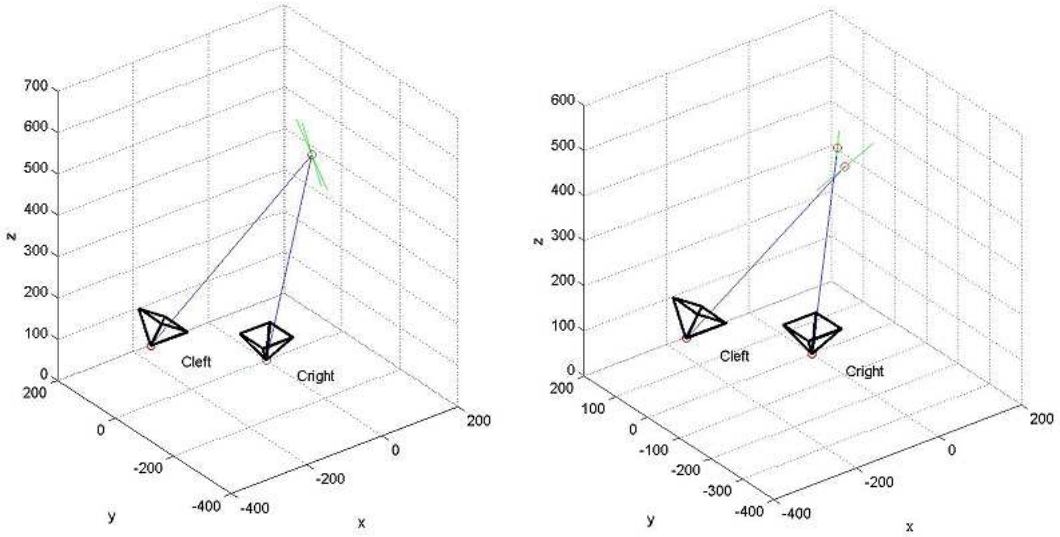
(a) Chosen gray values

(b) Chosen gray values

(c) Resulting Lines

(d) Resulting Lines

Figure 7.1: Inappropriate gray value thresholds



(a) Best case: $d_1 = 0.12\ mm$, $d_2 = 0.04\ mm$ (b) Worst case: $d_1 = 115.90\ mm$, $d_2 = 40.58\ mm$

Figure 7.2: Triangulation Result

7.2.2 Time consumption of the different steps

Many processing steps are necessary in order to reconstruct the 3D position of the pair of scissors starting from two infrared images. Figure 7.3 shows the ratio of time consumption of the different steps in the computation process. The whole process takes about two seconds.

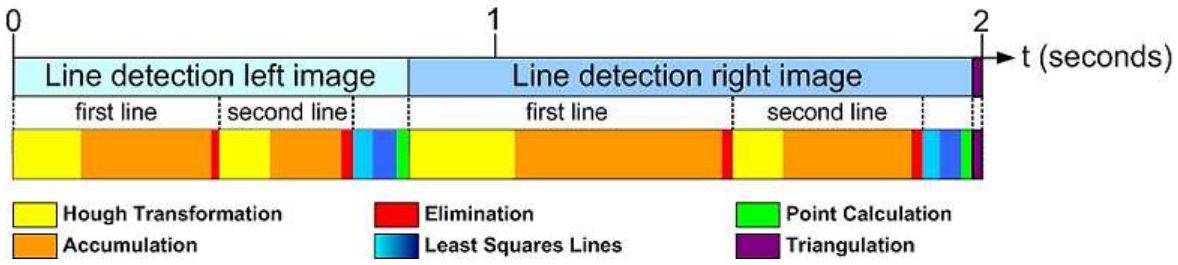


Figure 7.3: Time Consumption of the different steps

For the detection of the lines in the two images much more time is necessary than for the triangulation. Only about one percent of the time is consumed by the 3D reconstruction. Within the detection process most of the time is spent on the Hough Transformation and the accumulation of the votes for the Θ-ρ-pairs (red and orange bars). This effort could be reduced by choosing a coarser grid in parameter space. But then the accuracy of the result would decrease. The duration of these two steps is also depending on the number of pixels which are considered for the transformation. This amount can be controlled by the gray value threshold gvt (see section 3.1.3).
For the generation of the statistics the parameters for line detection were set to fixed values. The line detection in the right images was more time consuming than in the left ones because the right images had a brighter illumination. Hence for the same gray value threshold more

pixels were considered for the further computations than in the left images. Still the ratios of the different steps within one line detection process are about the same.

7.2.3 Time comparison: Matlab & C++

In the introduction Matlab is presented as an ideal tool for image processing and numerical computations which has some disadvantages concerning time and memory consumption (see 1.3). For this project the focus lay on the development of the algorithms, the duration of the different steps was not so important. Nevertheless we wanted to have an idea of the time consumption for an implementation in a more time and memory efficient programming language. Hence the most time consuming steps, Hough Transformation and accumulation (compare figure 7.3), were additionally implemented in **C++**. Table 7.3 gives an overview of the time consumption for the C++ implementation and two Matlab implementations. In the first one the whole line detection process is implemented within one file, for the second one four sub steps were put in separate files: hough transformation and accumulation, elimination, least squares calculation and point calculation. The first Matlab implementation shows

Process step	Matlab (all in one)	Matlab (several files)	C++
1st hough transformation	0.110 s	0.141 s	0.190 s
1st accumulation (4954 pixels)	26.469 s	0.453 s	0.160 s
1st elimination	0.016 s	0.015 s	–
2nd hough transformation	0.078 s	0.110 s	0.140 s
2nd accumulation (3400 pixels)	18.547 s	0.312 s	0.110 s
2nd elimination	0.016 s	0.015 s	–
1st least squares calculation	0.063 s	0.032 s	–
2nd least squares calculation	0.047 s	0.031 s	–
point calculation	0.026 s	0.031 s	–

Table 7.3: Time comparison: Matlab, C++

the problem of time and memory consumption very clearly. The for-loop in the accumulation process takes between 5.3 and 5.4 seconds per 1000 pixels. This slow execution was the reason for the additional C++ implementation. Later the sub steps were put in separate Matlab files and the time consumption decreased significantly for the accumulation part. Furthermore the memory occupation was ameliorated with this partitioning. Matlab keeps all variables in working space until the end of a function. Hence when all the statements are executed within one function the memory consumption is much larger than necessary.

Bibliography

[1] *Matlab from Mathworks*. http://www.mathworks.com/.

[2] *Matlab Camera Calibration Toolbox*.
http://www.vision.caltech.edu/bouguetj/calib_doc/.

[3] S. BIRCHFIELD, *An Introduction to Projective Geometry*.
http://robotics.stanford.edu/~birch/projective.

[4] G. BISHOP and G. WELCH, *An Introduction to the Kalman Filter*. SIGGRAPH 2001, Course Notes http://www.cs.unc.edu/~welch/kalman/.

[5] G. BISHOP, G. WELCH, and B. D. ALLEN, *Tracking: Beyond 15 Minutes of Thought*. SIGGRAPH 2001, Course Notes http://www.cs.unc.edu/~welch/kalman/.

[6] A. BLAKE and M. ISARD, *Active Contours*, Springer Verlag, 1998.

[7] R. O. DUDA and P. E. HART, *Use of the Hough Transformation to detect lines and curves in pictures*, in Communication of the ACM (Volume 15, Issue 1), January 1972.

[8] D. FORSYTH and J. PONCE, *Computer Vision - A modern Approach*, Prentice Hall, 2003.

[9] G. HAGER, *A Brief Reading Guide to Dynamic Vision*. A Guide to Visual Tracking
http://www.cs.jhu.edu/~hager/tutorial/shortlist.pdf.

[10] R. HARTLEY and A. ZISSERMAN, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2003.

[11] J. HEIKKILÄ and O. SILVÉN, *A Four-step Camera Calibration Procedure with Implicit Image Correction*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97, p. 1106-1112), San Juan, Puerto Rico, 1997.

[12] J. ILLINGWORTH and J. KITTLER, *A Survey of the Hough Transform*, in Computer Vision, Graphics and Image Processing (CVGIP, Volume 44, p. 87-116), April 1988.

[13] B. JIANG and U. NEUMANN, *Extendible Tracking by Line Auto-Calibration*, in IEEE and ACM International Symposium on Augmented Reality (ISAR'01), New York, October 2001.

[14] J. KUIPERS, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*, Princeton University Press, 2002.

[15] R. MUKUNDAN, *Quaternions: From Classical Mechanics to Computer Graphics and Beyond*, in Proceedings of the 7th Asian Technology Conference in Mathematics (ATCM'02,p. 97-106), Malaysia, December 2002.

[16] N. NAVAB, *Visual Motion of Lines and Cooperation between Motion and Stereo*, PhD thesis, Université de Paris XI Orsay, U.F.R. d'Informatique, January 1993.

[17] F. SAUER, A. KHAMENE, and S. VOGT, *An Augmented Reality Navigation System with a Single-Camera Tracker: System Design and Needle Biopsy Phantom Trial*, in Proceedings of the Second International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2002.

[18] T. SIELHORST, *High Accuray Tracking For Medical Augmented Reality*, Master's thesis, Technical University Munich, Department of Computer Science, October 2003.

[19] E. W. WEISSTEIN, *Rodrigues' Rotation Formula*. From Mathworld – A Wolfram Web Resource `http://mathworld.wolfram.com/RodriguesRotationFormula.html`.

[20] E. W. WEISSTEIN, *Rotation Matrix*. From Mathworld – A Wolfram Web Resource `http://mathworld.wolfram.com/RotationMatrix.html`.

[21] J. WENG, P. COHEN, and M. HERNIOU, *Camera Calibration with Distortion Models and Accuracy Evaluation*, in IEEE Transaction on Patterns and Machine Intelligence (PAMI'92, Vol. 14, No. 10, p. 965-980), 1992.