Automatic registration of laser scanning data and colour images

Steven Alexander Sablerolle

TU Delft

Zoller+Fröhlich

DLR

Automatic registration of laser scanning data and colour images

Graduation thesis
in
Geomatics

Steven Alexander Sablerolle

Graduation Professor: Prof. Dr. D.G. Simons
Supervisor TU Delft: Alexander Bucksch
Supervisor Z+F: Thomas Abmayr

Optical and Laser Remote Sensing
Faculty of Aerospace Engineering
Technical University of Delft

Research by order of:
Zoller + Fröhlich GmbH
Wangen im Allgäu, Germany

And in cooperation with:
Deutsche Zentrum für Luft- und Raumfahrt
Munich, Germany

October 2006

Preface

This report is the result of my graduation research, entitled as “Automatic registration of laser scanning data and colour images”. This thesis is carried out for the completion of the Master Geomatics at the Technical University of Delft. The research is executed by order of Zoller + Fröhlich GmbH, Wangen im Allgäu, Germany.

The introduction in chapter 1 explains the reader the purpose of this research and the environment in which it was carried out. Chapters 2 and 3 concern the background information as the foundation of the theory. The chapters 4 and 5 go more into detail of the execution of the research. Chapter 6 describes all the results that are retrieved from the execution and finally in chapter 7 the reader can find the conclusions and recommendations.

The execution of this research has taken place partially in Munich (Germany) and partially in Delft (the Netherlands). Three months (April – June 2006) were spent in Munich at the site of DLR, “Deutsche Zentrum für Luft- und Raumfahrt” (Germany's Aerospace Research Centre and Space Agency). Here the practical part of the research was carried out. The report is written in Delft.

First of all I would like to thank Zoller + Fröhlich in the persons of Dr. Ing. Christoph Fröhlich and Dipl. Ing. Markus Mettenleiter for giving me the opportunity to do this research and to include my work in the cooperation. I thank the Institute of Robotics and Mechanics of DLR in the person of Prof. Dr. Ing. Gerd Hirzinger for having me at the DLR-site in Munich. From the Technical University of Delft I thank Prof. Dr. Dick Simons for his interest in my topic, the feedback and the nice collaboration. I thank Sisi Zlatanova as co-reader of the exam committee.
Many thanks go to Alexander Bucksch and Thomas Abmayr, who have supervised me very well. I could come to you for advice and help at any time, thank you. On top of that special thanks to Thomas who has shown me the great way to live in the city of Munich.
Most grateful I am to my parents, who have supported me all the way to this graduation; therefore I am not the only one graduating here!

Delft, September 2006
Steven Sablerolle

Summary

Laser scanning is a new technique for 3 dimensional point measurements. Over the last decades the number of applications in which these measurements take place, has grown significantly. In every application the surveyors are looking for the same result, namely a 3 dimensional point cloud of the object of interest. These point clouds are used for modelling and visualization.

The purpose of this research is to improve the quality of the visualization: the objective is to automatically register a laser scanner point cloud with colour images. This objective is based on the developed data acquisition of Zoller + Fröhlich GmbH, Wangen im Allgäu, Germany: a digital camera is fixed on the laser scanner as close to its centre as possible to avoid occlusion. The instrument produces a full scan and colour images of the band around the turning axis.

As the title already states this research strives for automation of the integration of 3D points and colour information. To accomplish this only feature based method are used in this research to find the connection between the intensity scan and every colour image. In these methods features from objects in the images determine the connection between the images. In intensity based methods the intensities in the images form the connection.

After a literature study on feature based methods one method is designed to integrate the laser and the colour data fully automatic. This method is implemented in the programming language C/C++ supported by the openCV library.

The implementation starts with improving the intensity and colour images on brightness and contrast in order to make the images ready for further processing. Then the program starts with selecting features from the images, called key points. With the Canny edge detector the edges of the objects are found. With morphological operation like pruning unusable noise and small edge segments are removed. The Harris corner and edge detector finds the points of interest on the corners of the objects. The Hough transform selects straight lines from the objects and then reduces the number of objects by constraining them on containing straight lines. With the Eigen values from the Harris detector the remaining edge points are removed and the highest corner response is set to be the key point location.

After the key points have been selected, they get a SIFT (Scale invariant Feature Transform) descriptor. These descriptors are used to match the key points from the intensity image with the points from the colour images. Because of the difference between laser and camera light source, the descriptors are used in different modes and the search for matches is restricted to a local area. The dataset of point pairs is the input for a non-linear rigid optimization. The output is optimized into a set of external parameters of the colour image. These parameters form the connection between the colour points and the laser 3D coordinates. Visualization can be made now by either plotting the colours onto the 3D points or plotting the range data onto the colour image.

The automatic registration of laser scanning data and colour images designed in this research is successfully implemented. During the implementation new ideas have been added to the design. For example the pruning algorithm is used to remove more noisy objects and line segments, the SIFT descriptor is used in different modes and different parameters have been researched.

For the future this research should be tested for different types of environments where the laser scanner can be in. The optimization can be improved by RANSAC and by adding point pairs of overlapping colour images. Moreover other types of descriptors can be tried to improve the results of the point pair selection. Finally the results should be compared with the results of the intensity based methods before selecting this method as a base for the software tool of automatic registration.

Table of contents

Introduction

Laser scanning is a technique to retrieve 3 dimensional coordinates based on angle and distance measurements. Over the last decades the number of applications in which these measurements take place, has grown significantly. In every application the surveyors are looking for the same result, namely a 3 dimensional point cloud of the object of interest.
The point clouds again can be the input for different ways of post processing; they can for example function as a base for a digital height model of the earth (DHM) or be part of a comparison with respect to the design model of an object like a building or an airplane.
In these examples the function of the laser scanner is 3D modelling. Another function that this instrument can fulfil is visualization. Examples of applications in visualization are product presentation or mapping of cultural heritage. This research takes place in the context of progressive improvement of this visualization.
The German company Zoller + Fröhlich GmbH ("**Z+F**") covers a wide spectrum in the field of laser measurement technology. One of the latest interests of Z+F is the integration of a digital camera and the laser scanner Imager 5003. The purpose of this integration is that the instrument is upgraded from a point cloud producer to an instant 3 dimensional visualizer. The colour information of the camera will give considerable improvement to the quality of the visualization.

As the title already states this research strives for automation of the process of integration. The goal here is to produce a program that automatically integrates the laser scanner data and colour information of the digital images of the camera. The complete process is based on feature based registration methods. Feature based methods retrieve the connection between 2 images by finding features on the objects in both images. Parallel to these methods the intensity based methods are researched by *Hannes Sahl* [14]. Intensity based methods compare the intensity values of both images to find the connection between them.

This research starts with a literature study on image processing and feature based methods. First the goal is to get familiar with the theory. Then the knowledge is used to produce a method design that is motivated to be able to produce the final objective. By programming in C/C++ this method is investigated, changed and improved. The conclusions of this research therefore consider the in-between and final results of the design method. The results retrieved throughout the process indicate the quality of the method. Problems and changes are described in order to bring out a number of recommendations.

The method that is designed and programmed during this work is based on the constraint that the images and scan are taken from roughly the same position. This reduces the problem complexity considerably, because occlusions are less of a problem. Additionally the base line between the camera and the laser scanner is approximately known from the data acquisition. This knowledge can improve the quality of the integration considerably.
The implementation of the proposed method requires programming, though a final software product is not the aim. The theoretical results from this research provide a base for further research and software. The proposed method is based on existing approaches, but innovations and new ideas are found and implemented.

2 Laser scanning

Laser scanning is a new technique for 3 dimensional ("**3D**") point measurements. It is developing quickly in proving itself as a popular instrument to replace the established techniques of photogrammetry and tachometry. Laser scanners are used more and more as surveying instruments for various applications, such as mining, archaeology, architecture, cultural heritage, automobile and mechanical engineering.
The technique of laser scanning has a central role in this report. Therefore it is important to learn its basics. By order of Zoller + Fröhlich ("**Z+F**") this research is directly linked with the Imager 5003, the laser scanner that they have developed. In paragraph 2.1 the general principle of laser scanning is shortly explained. Paragraph 2.2 introduces the important properties of the scanner Imager 5003. Finally paragraph 2.3 brings out the relevant theory about the point clouds retrieved by the laser scanner.

2.1 General principle

There are 3 types of laser scanners; the pulse, the triangulation and the phase scanner. Although these scanners have different principles of the distance measurement, the basic principle is the same. They send a laser pulse to the object of interest and retrieve the returning pulse. Figure 2.1 illustrates the general principle of the laser scanners.

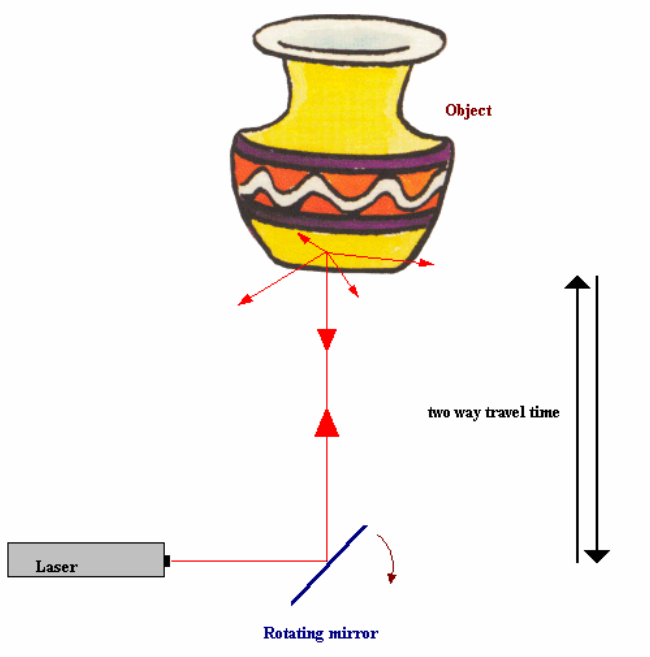
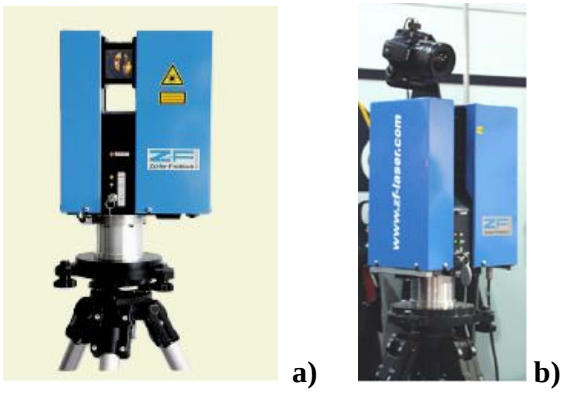


Figure 2.1: The general principle of laser scanners.

The pulse scanner simply measures the two way travel time of a short laser pulse. With the speed of light the scanner can compute the distance from the instrument to the object. The triangulation scanner makes use of a small mechanical baseline between the outgoing and the incoming pulse. By measuring the direction of the outgoing pulse the scanner can reconstruct the triangle and compute the distance. The phase scanner measures the phase angles of the outgoing and incoming laser signal. From the phase differences the scanner solves the problem of ambiguities by using a combination of different wavelengths and computes the distance.
The combination of the direction of the pulse and the distance determines the x, y and z-coordinates of the point on the object with respect to the reference frame of the instrument. Additionally also the intensity of the backscattered pulse is measured.
Mostly the instrument is placed on a tripod. The either oscillating or rotating mirror sends the pulses continuously in a large range of directions in the vertical plane. Together with a rotation around its z-axis, the laser is "scanning" its surroundings, measuring the complete 360 degrees of the sphere around the location of the instrument. This results in a short measurement time for a very large point cloud.
This paragraph gives the principles of laser scanning very shortly, because for this research it is only relevant to understand the basics of the measurement system. For a more detailed look into the subject of laser scanning see [13], *J.M. van Ree*.

2.2 Imager 5003

The laser scanner IMAGER 5003 (see figure 2.2.a) is developed and produced by Z+F (see [5] *Fröhlich*). This phase scanner is used for testing the research program during this Master thesis. The IMAGER 5003 is capable of measuring with an absolute precision of 19 millimetres till the range of 56.6 metres. At every point it also measures the intensity of the backscattered laser light. The instrument turns around its vertical axis, so that horizontally it is capable to scan 360 degrees. A rotating mirror sends the laser pulse in the vertical plane; the range here is 270 degrees, as it can not see its footprint beneath the tripod. The measurement speed reaches up to 625000 points per second. The total intensity image contains 36000 x 20000 (horizontal x vertical) pixels maximally.
For the purpose of the integration of the colour information and the laser point clouds, Z+F has produced the capability to fix a Canon 350D digital camera on the side of the scanner. Figure 2.2.b shows the camera fixed on the laser scanner. During the rotation of the laser scanner the camera can shoot the colour images.



a) **b)**

Figure 2.2: a) The Z+F IMAGER 2003 **b)** the CANON 350D fixed on the IMAGER 2003 (source: [22] *Zoller + Fröhlich*)

2.3 Laser point clouds

When a measurement with a laser scanner is performed, the result will be a point cloud of x, y and z-coordinates (including the intensity) with every point in sight of the scanner. One scan results in a 3 dimensional point cloud with intensity values. Most software packages have two ways to handle this point cloud and visualize its data. The first way is an intensity image that shows a panorama picture of intensity grey values. An example is shown in figure 2.3. The intensity values are represented in spherical coordinates (θ, λ, I), with horizontal coordinate θ, vertical coordinate λ and intensity value I. The pixels can also be valued by the range measurements instead of the intensity measurements. Then the points are in the spherical coordinate system (θ, λ, r), where r is the range value.

Figure 2.3: Example of an intensity image with artificial targets (pointed by the yellow arrows).

For the second visualization a 3D viewer is used. With such a viewer the software is capable of plotting all the points in a 3D environment in which the user can move around. The points are transformed into Cartesian coordinates and are coloured by there intensity. Figure 2.4 shows an example with the intensity from low to high in the order of red-green-blue. In this way the user can easily distinguish different objects and interpret their intensity property.
The laser scanner measures the angle and distance to every point. Thus the actual measurements have spherical coordinates. When the light rays of the laser (direction and range) are reconstructed into a 3D environment a transformation takes place from the spherical coordinate system to the Cartesian coordinate system.

One scan is never enough to see the complete objects that have to be modelled. Moreover the object can block the sight of other objects (this is called occlusion). One single scan gives a 3 dimensional point cloud. But if you use this cloud to produce a 3 dimensional model of the object of interest its properties will be 2.5 dimensional[1], what means that every direction includes one range value only. This causes unwanted occlusions. Another problem can be sometimes that the object of interest is too large for one single scan. To solve this, the scanner is placed at *n* different position. This results in *n* different point clouds and intensity images. In order to combine these measurements to a complete point cloud, corresponding points are searched in all scans. This can be done with pre-designed targets or original local features in the image. The yellow arrows in figure 2.3 point some pre-designed targets as an example. When enough (4 at minimum, but picking more is good for reliability, accuracy and stability) corresponding points are found in two sets they can be lined up and transformed into one coordinate system. With a least squares adjustment the best fit orientation and relative position of the point clouds with respect to each other is determined. If two clouds are combined, a third one is added and so on. When finally all point clouds are combined, the result will be a 3D point cloud containing all points of interest on the object with the retrieved intensity.

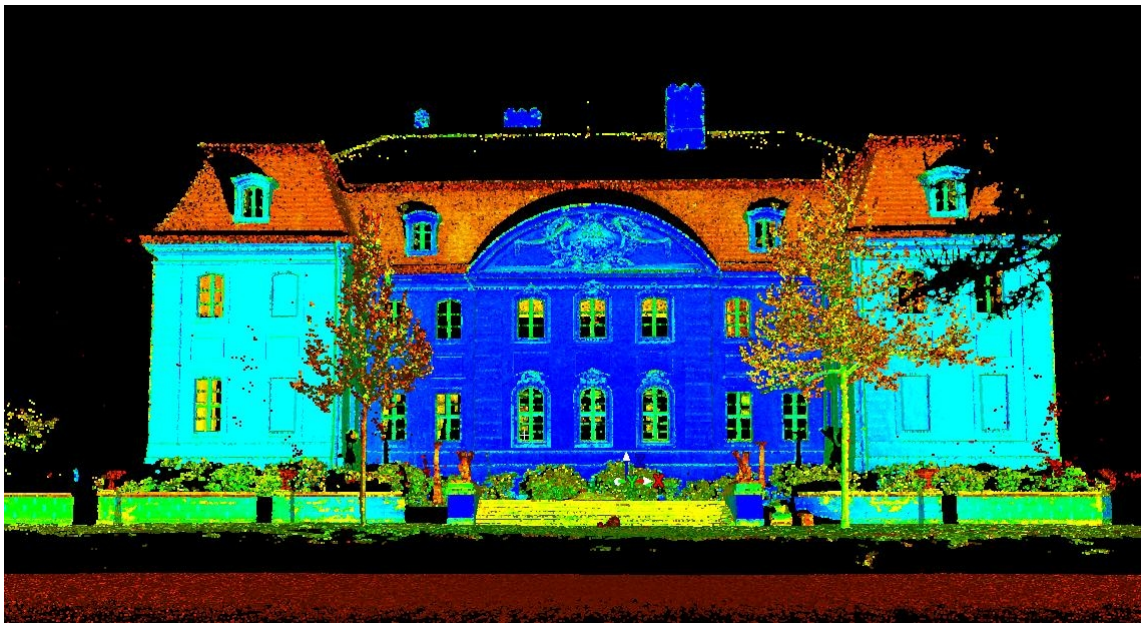


Figure 2.4: The 3D viewer shows the scanned points and gives colours to its intensity value (low-mid-high = red-green-blue).

[1] **2.5 dimensional**: the concept of 2.5 dimensions means that every x, y coordinate pair can have one z value only, (see [18] University of Stuttgart).

3 Digital image processing

The practical goal of this research is to produce a method design for the automatic registration of laser scan data and colour images. The word 'registration' actually means image registration. Image registration is the process of aligning two or more images at the same scene. The word 'automatic' means that this process is fully automated.
For the automatic registration of 3D and colour data in this research digital images (intensity and colour) are used. They contain the connection between the 3D data and the colour information. This brings the research directly to the subject of digital image processing. The basics of digital image processing, which are needed to design the algorithm for automatic registration, are presented in this chapter.
Before creating the program it is required to make the theory accessible and to design a method for the different algorithms that will be used. This chapter shows the results of the literature study performed to investigate the possibilities of using different existing algorithms. Paragraph 3.1 starts with a short introduction to the theory of digital image processing. In paragraph 3.2 three relevant principles are explained. Paragraph 3.3 – 3.7 contain the theory of different types of image operators, detectors and descriptors, which are used in this project.

3.1 Introduction to digital image processing

The history of digital image processing starts already in the 1960's, when many of the techniques of digital image processing were developed in relation with for example satellite imagery, medical imaging, videophone, character recognition and photo enhancement. When in the 1970's cheaper computers and dedicated hardware became available, digital image processing grew very fast. Images could then be processed in real time. The general-purpose computers became faster and they started to take over the role of the dedicated hardware except for the most specialized applications, which were concerned with very compute-intensive operations. From this moment digital image processing is generally used. It has developed into the most common form of general image processing, because computers are faster and more digital instruments are available. One of these new instruments is the laser scanner (presented in chapter 2).
The main goal of this research is to integrate the laser scanner 3D data and a digital camera colour data automatically. To be able to succeed in this automatic approach, which would replace a lot of time consuming post-processing, the laser intensity image and the colour intensity image are used.

In digital image processing two approaches exist to perform the registration between the images; the intensity-based and the feature-based methods. In the first approach of methods the correspondences are directly searched in the intensity profiles of the images. In the second approach first features are extracted from the images from which then the matching process is applied.
This research will be based on the possibilities of feature based methods, which means that the methods are aiming to feature extraction from the data. These features make the link between the 3D data and the colour data possible. The intensity based methods do not extract features, but use all pixel intensities instead to iteratively optimize the location and orientation of the images. Parallel to this research the intensity based methods are developed by *Hannes Sahl* [14].
In digital image processing feature based methods have been developed in order to automatically extract features from images and to determine the location and orientation of the images. In these methods operators, detectors and descriptors play an important role.
The operators can handle digital intensity images in different ways depending on their purpose. The Gaussian operator for example is used to smooth an image. The purpose of detectors is to find certain objects in an image. Detectors can use different operators, for example to get the gradient of the intensity of the image or to remove the noise. In this way detectors like Harris and Moravec are able to find edge points or corner points.
Many descriptors use different detectors to detect feature points. After detection they give a proper description to the feature points to make them distinguishable. The descriptors are used to match one feature in different images. Below some descriptors are explained. Examples are Scale Invariant Feature Transform (SIFT), Gradient location-orientation histogram (GLOH), Steerable filters, Cross correlation and Spin images.
In order to find the best strategy to perform the automatic integration for the Imager 5003 and a digital camera, this literature study to different types of detectors and local descriptors was carried out. The results of this study are described in chapter 4, where the designed strategy is presented.

3.2 Basics of image processing

This paragraph describes three basics of image processing, namely image gradients, spatial filtering and histogram spreading. First the important principles of image gradients and then secondly the basics of spatial filtering is explained. These two subjects are necessary to understand how different operators, detectors and descriptors function. At the end of this paragraph histogram spreading is shortly explained, because it can possibly improve the input images of the program prototype.

3.2.1 Intensity gradients

Edge detection is an important and common approach for detecting meaningful discontinuities in grey level. First and second-order derivatives are implemented in an image to find changes in the gradient. These changes point out the location of an edge. To understand this it has to be clear how a digital edge is defined.
Intuitively an edge is a set of connected pixels that lie on the local boundary of two different regions. Figure 3.1.a shows an ideal edge with its grey-level profile. In practice an edge is never that sharp. The model shown in figure 3.1.b is therefore more realistic. With the model for a blurred edge (b), also called a ramp edge, the thickness of the edge is determined by the length of the ramp. It moves from an initial to a final grey level (the intensities of the face left and right of the edge). So blurred edges tend to be thick and sharp edges tend to be thin.
The first derivative of the intensity of the pixels is zero in areas of constant grey level. It shows maxima (negative or positive) when the intensity changes. The second derivative contains maxima at the beginning and ending of the grey-level profile. From these observations (figure 3.1.c) it can be concluded that the magnitude of the first derivate determines the presence of an edge point (pixel). The second derivative produces the mid line points of the edge (zero crossings in between the maxima) and tells whether a point is on the light or dark site of the edge. When determining proper edge points the use of thresholds is important, because the derivates are very sensitive to noise. Besides handling the noise with filters (e.g. a median or a Gaussian filter) a threshold can remove the blurred part of the edge; only the derivatives above the threshold are defined as edge. Filters and the threshold together handle noise very well.
If derivatives have to be retrieved from digital images, discretization should take place. The derivatives can be implemented by making them discrete with so-called masks. These masks are windows that are shifted over the pixels. These windows are often called operators and many different operators exist. The Roberts, Prewitt and Sobel operator are examples of discrete masks that return the first derivative of the image signal. The Laplacian operators are examples of windows that return the second order derivative of the signal. In this chapter some of these examples are explained.

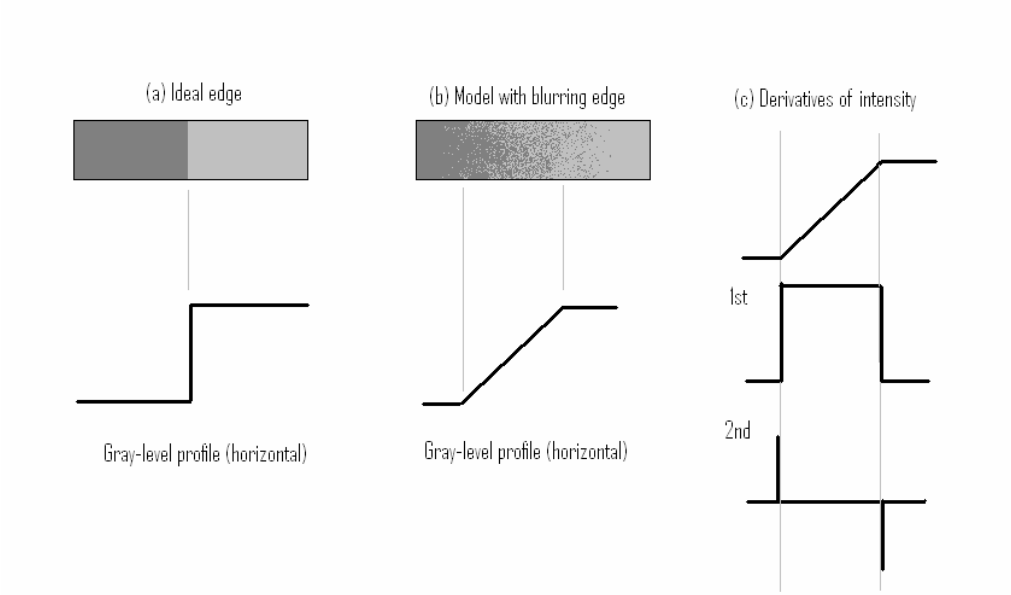


Figure 3.1: Gray-level profile of **a)** the ideal edge, and **b)** a blurred edge, **c)** Profiles of the 1st and 2nd derivatives (source: [6] *Gonzalez and Woods*).

3.2.2 Spatial filtering

Many image operations in automatic registration deal with the values of the image pixels. To be able to do this first the basics of spatial filtering needs to be explained. All the above mentioned operators work with sub-images implemented on the images. A sub-image can be called filter, mask, kernel, template or window. Such a window is built up by discrete numbers that represent the operator function. These are called the filter coefficients.
The mechanics of spatial filtering are summarized here in a simple process. The filter mask is moved over the image from point to point. In this way every pixel gets a response from the operator. The response is the linear result of the sum of products of the filter coefficients and the corresponding image pixel values within the filter window (see [6] *Gonzalez and Woods*). Figure 3.2 illustrates an example of the process for a 3x3 mask. The response derived with this mask is defined in the following formulas (3.1):

$$R = \sum_{u,v} w(u,v) f(x+u, y+v) \tag{3.1}$$
$$= w(-1,-1) f(x-1, y-1) + w(-1,0) f(x-1, y) + \quad \Lambda \quad + w(0,0) f(x,y) +$$
$$\Lambda \quad + w(1,0) f(x+1, y) + w(1,1) f(x+1, y+1)$$

with

w	*the mask coefficients,*
(u,v)	*the position in the mask,*
f	*the image value, and*
(x,y)	*the pixel coordinates in the image.*

In spatial filtering this basic process between an operator and an input image is called the convolution operation. It is usually indicated with the mathematical signs $*$ *or* $\otimes$.

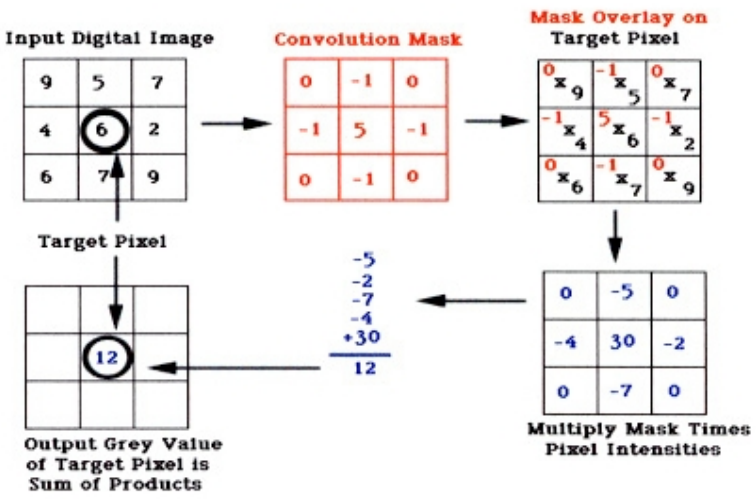


Figure 3.2: An example of the process of convolution (source: [19] *www.udel.edu*)

3.2.3 Histogram equalization

When processing images the contrast and the brightness of the images can be improved with histogram equalization. Figure 3.3 illustrates the principle idea of histogram equalization. The intensities of the pixels are organised in 256 bins. The left histogram shows the number of pixels that have the same grey value and are in the same bin. In this case most of the pixels are in the low intensity region. Like here images might not use the full available range of grey values $\{0,255\}$. Therefore an image can have a low brightness (many pixels in lower grey values) or a bad contrast (all pixels in a small range). When histogram spreading is used to improve these properties, the following steps are performed:

1. Calculate the histogram for the input image.
2. Normalize the histogram into **H**.
3. Compute **H'** the integral of the normalized histogram **H**.
4. Transform the image using **H'** as a look-up table: OutputImage(x,y) = H'(InputImage(x,y))

In step 1 all pixels are simply counted for every intensity bin of the image (256 grey values). Figure 3.3.a shows an example of the resulting histogram. Step 2 normalizes the resulting histogram of step 1. Normalizing means that the number of pixels in every bin is divided by the total number of pixels, so that the sum of all bins becomes 255:

Normalized histogram (**I**) = Total Number of Pixels of Intensity **I** / Total Number of Pixels (3.2)

In step 3 the integral of the normalized histogram is computed in formula (3.3):

$$H'(i) = \int_0^i H(j) \cdot dj \tag{3.3}$$

The discrete version of formula 3.3 (needed for digital images) is defined in formula 3.4:

$$H'(i) = \sum_{j=0}^{j=i} H(j) \tag{3.4}$$

Finally in step 4 the transformation takes shape as the new position of every bin is selected from the integral of the normalized histogram. The look-up table selects the bin, finds the value under the normalized histogram (integral) and places the bin at the position of this value in the output histogram. The histogram of the output image is shown in figure 3.3.b. Note from the cumulative frequency graph of this result (figure 3.4), that the grey level frequency is now a linear function of the grey level value; all values are represented equally in the image.

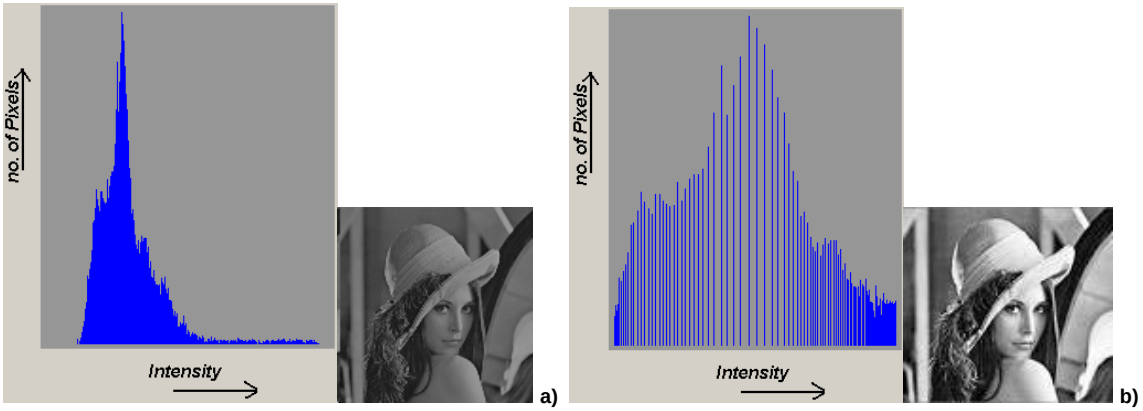


Figure 3.3: Intensity histogram **a)** before equalization and **b)** after equalization (source: [16] *www.codersource.net*).

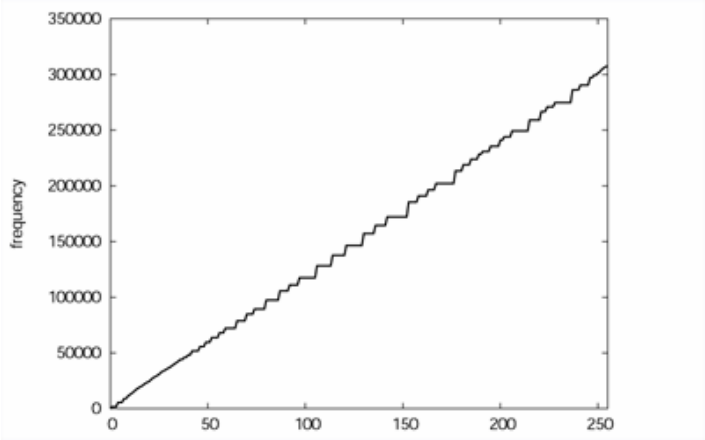


Figure 3.4: Cumulative frequency graph of an equalized image (source: [17] *www.generation5.org*).

3.3 Image operators

In the world of image processing a lot of different operators exist. This chapter explains a few of them that are important for the planned algorithm. The types of operators that are discussed here are respectively gradient operators, Laplacian operators and Gaussian operators.

3.3.1 Gradient operators

The derivatives explained above can be implemented for an entire image by using discrete masks or windows. Most of the masks are odd in their size. It is wise to give the window an odd size (3x3 or 5x5), because then the middle position of the window can be placed exactly on the pixel of interest. But nevertheless even sizes also exist. Most operators can also be enlarged and they can also differ in shape. But the main correspondence of all these operators is that they can return an approximation of the first derivates from discrete data. Figure 3.5 contain three examples of gradient operators.
The Sobel operator is an example of an odd sized window patch. The Sobel operator returns a value for the first directional derivative of the image intensity. The left window returns the horizontal derivative (x-direction) and the right window gives the vertical derivative (y-direction). The centre of the window is place on the pixel (indicated by the squares) and its 8 surrounding pixels are placed on the neighbours in the image. The content of the operator is then multiplied by the intensity of the corresponding pixel. Finally the summation of the pixels gives the directional derivative. The Prewitt (figure 3.5) operator works equally, but does not give double weight to the pixels that lie in the same direction. For the Roberts operators (2x2) the top left position is the location of the pixel of interest. Here the output of the operator loses accuracy on its location, because the derivative for every pixel is pulled from the pixels down right of the original pixel (the value properties shift a little in the direction of the top left corner).

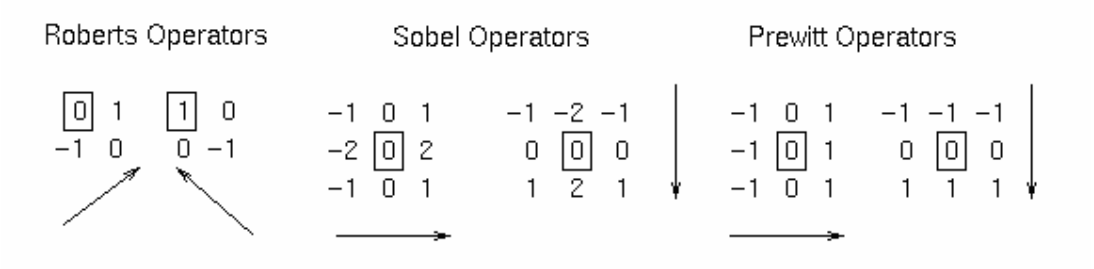


Figure 3.5: Examples of gradient operators.

3.3.2 Laplacian operators

The Laplacian operators are examples of image processors that return the second order derivative. Formula 3.5 contains the simplest isotropic way (only two directions) to describe the Laplacian function:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{3.5}$$

In order to be useful for digital image processing the Laplacian function must be expressed in discrete form. First the two directional partial second derivatives are defined in discrete form:

$$\begin{aligned} \frac{\partial^2 f}{\partial^2 x^2} &= f(x+1,y) + f(x-1,y) - 2f(x,y) \\ \frac{\partial^2 f}{\partial^2 y^2} &= f(x,y+1) + f(x,y-1) - 2f(x,y) \end{aligned} \tag{3.6}$$

Implementation in formula 3.5 is obtained by summation of the partial derivatives. The result is given by formula (3.7):

$$\nabla^2 f = \left[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \right]. \tag{3.7}$$

From this formula the left mask of figure 3.6 is created. The diagonal directions in the right mask can be incorporated in the definition of the Laplacian by adding the diagonal second derivatives to formula 3.6.

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 3.6: Two examples of discrete mask of the Laplacian function.

3.3.3 Gaussian operators

It is sometimes useful to apply a Gaussian smoothing filter to an image before performing edge detection. The filter can be used to make the edges softer and filter out the noise. It is constructed from the Gaussian function for 2 two dimensions and made discrete into a mask.

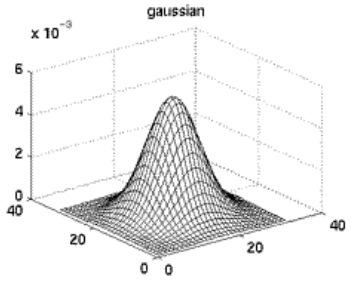
The Gaussian function (plotted in figure 3.7.a) is defined as

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{3.8}$$

with

x,y	*pixel coordinates*
σ	*standard deviation (Gaussian scale)*

By using a small Gaussian filter (for example 7x7 mask) and then applying an edge detector, a great deal of fine details remains. However, there will still be a lot of unwanted edge fragments appearing due to noise and fine texture. For a larger filter (of order – 31 x 31) there will be fewer unwanted edge fragments, but the disadvantage is that much of the detail in the edges will be lost. Figure 3.7.b shows the 7x7 Gaussian mask as an example.



a)

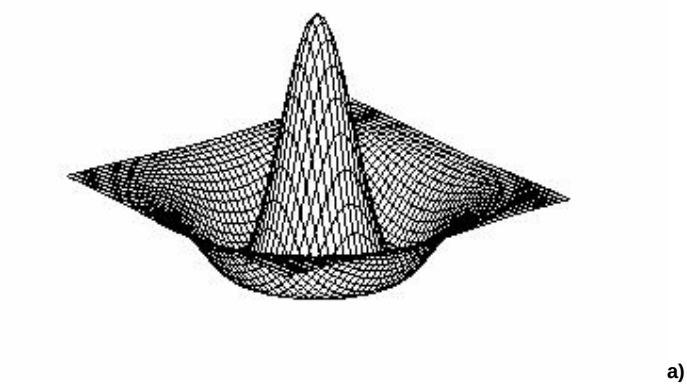
$$\begin{pmatrix} 1 & 1 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 2 & 4 & 8 & 16 & 8 & 4 & 2 \\ 2 & 2 & 4 & 8 & 4 & 2 & 2 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 1 & 1 \end{pmatrix}$$

b)

Figure 3.7: a) Gaussian function in two dimensions, **b)** Example of a 7x7 Gaussian mask.

The Gaussian operator is not only used by edge detectors. Also operators use its properties. The Laplace of Gaussian filter (LoG) for example combines the Laplace operator and the Gaussian filter, because the Laplace function tends to enhance the noise level in the images considerably. The result is the so-called 'Mexican hat' as shown in figure 3.8. The mask is a 5x5 discretion of the Mexican hat. The LoG is defined in formula 3.9. This formula is derived by taking the second derivative of the Gaussian function *G* (formula 3.8).

$$LoG(x, y, \sigma) = -\left[\frac{\left(x^2 + y^2\right) - \sigma^2}{\sigma^4}\right] \cdot e^{-\frac{\left(x^2+y^2\right)}{2\sigma^2}} \tag{3.9}$$



a)

$$\begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & -1 & -2 & -1 & 0 \end{pmatrix}$$

b)

Figure 3.8: a) Laplace of Gaussian function ("Mexican hat") and **b)** the discrete mask of the Laplace of Gaussian.

Another example of an operator that makes use of the Gaussian filter is the Difference-of-Gaussian (DoG). This function *DoG(x,y,σ)* is computed from the difference of two nearby scales separated by a constant multiplicative factor *k*:

$$DoG(x,y,\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) \otimes I(x,y) \tag{3.10}$$

with

G	*Gaussian function*
I	*Input image intensity*

This operator can actually be interpreted as a detector. It is removing the flat faces from the image, because their difference will be zero. The pixels with higher gradient in intensity (edges and corners) will remain. The advantage of this operator is its efficient computational effort.

3.4 Detectors

During the automatic registration of image features edge and corner detectors play an important role. The role of detectors is to extract features from images by recognizing their edges and corners. Discrete and reliable features are the basis for a successful registration. For this reason many different detectors have been developed. Some examples have been mentioned already in paragraph 3.1. This paragraph gives a more detailed description of how detectors work while making use of the important and much used detectors. Also their different properties are described. Edge detection on an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image.

3.4.1 Moravec detector

The corner detector of Moravec, developed by Moravec in 1977, is based on a local window patch that is moved through the image. It determines the changes of the image intensity caused by shifting the window a little bit in eight different directions (every 45 degrees).
The mathematical description of the Moravec operator outcome is

$$E_{x,y} = \sum_{u,v} w_{u,v} \otimes \left| I_{x+u,y+v} - I_{u,v} \right|^2 , \tag{3.11}$$

with

w	*the window patch,*
I	*the image intensity and*
E	*the output image produced by the shift.*

The window patch is just a square of ones and (x,y) defines the shift of this window. Before summation of all the patch pixels *w* is convoluted with the square of the absolute difference between the intensity of every position (u,v) before and after the shift, $\left| I_{x+u,y+v} - I_{u,v} \right|^2$. It points out different situations:

- If the window moves over a local flat intensity
  → the outcome is small.
- If the window moves over an edge
  → the outcome is large (perpendicular) or small (parallel) depending on the direction.
- If the window shifts over a corner or an isolated point
  → the outcome is large in both directions.

After deriving *E* for every pixel the Moravec detector looks for the local maxima in *E* above a defined threshold. The shortcomings are that the detector only handles 8 directions and that it has a noisy response, which focuses more on the edge pixels than the corner pixels. The Harris detector handles these problems that were initially not covered with the Moravec operator. The next section explains how this detector works.

3.4.2 Harris detector

The Harris detector was developed by Harris and Stephens in 1988 (see [7] *Harris and Stephens*). The Harris detector is the base of many new algorithms in the field of automatic registration. Most of the algorithms that exist now depend in someway on its basics.
The basic idea of Harris is that a corner and edge point should be easily recognized by looking through a small local window. Figure 3.9 states that shifting a window in any direction gives a large change in intensity, when looking at a local corner point.

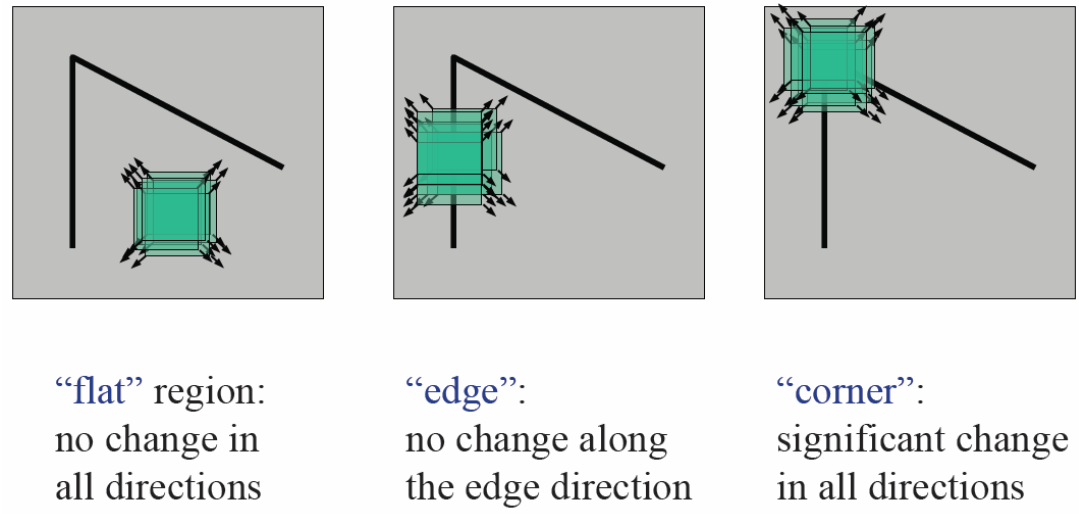


Figure 3.9: The Harris detector; shift a local window in all directions to find changes in intensity.

In the Moravec detector the change E is given by formula 3.11. But to quantify the change in intensity Harris makes use of the auto-correlation matrix M. This matrix, also called the Hessian or the second moment matrix, describes the gradient distribution in the local neighbourhood of any point picked from the image. By analytic expansion E can be written as the following formula (3.12):

$$\begin{aligned} E_{x,y} &= Ax^2 + By^2 + 2Cxy \\ &= (x,y) \cdot M(x,y) \cdot (x,y)^T, \end{aligned} \tag{3.12}$$

with

$$M(x,y) = \begin{bmatrix} A & C \\ C & B \end{bmatrix},$$

$$A = \sum_{u,v} w(u,v) \otimes \left(\frac{\partial I}{\partial x} \right)^2$$

$$B = \sum_{u,v} w(u,v) \otimes \left(\frac{\partial I}{\partial y} \right)^2$$

$$C = \sum_{u,v} w(u,v) \otimes \left(\frac{\partial I^2}{\partial x \partial y} \right)$$

The window patch w is convoluted with the square of the local derivative, which can be retrieved from the Sobel operator. Out of the Hessian matrix M Harris subtracts the squared *trace* from the *determinant* to compute the corner response R of the selected point:

$$R = Det(M) - k \cdot Trace(M)^2 \tag{3.13}$$

When completed for the whole image, all local maxima of R point out the locations of interest points. R is positive for corner regions, negative for edge regions and very small for flat regions. The flat regions are defined by a threshold and the parameter k. This parameter can be chosen by trying different numbers. The constant k has a direct influence on the number of points that will be returned by the detector, because R depends linearly on it. By changing k a maximum number of edge and corner points returned can be found and the behaviour of the parameter can be modelled. It is also possible to fix a suitable k and change the threshold value to influence the number of points detected.

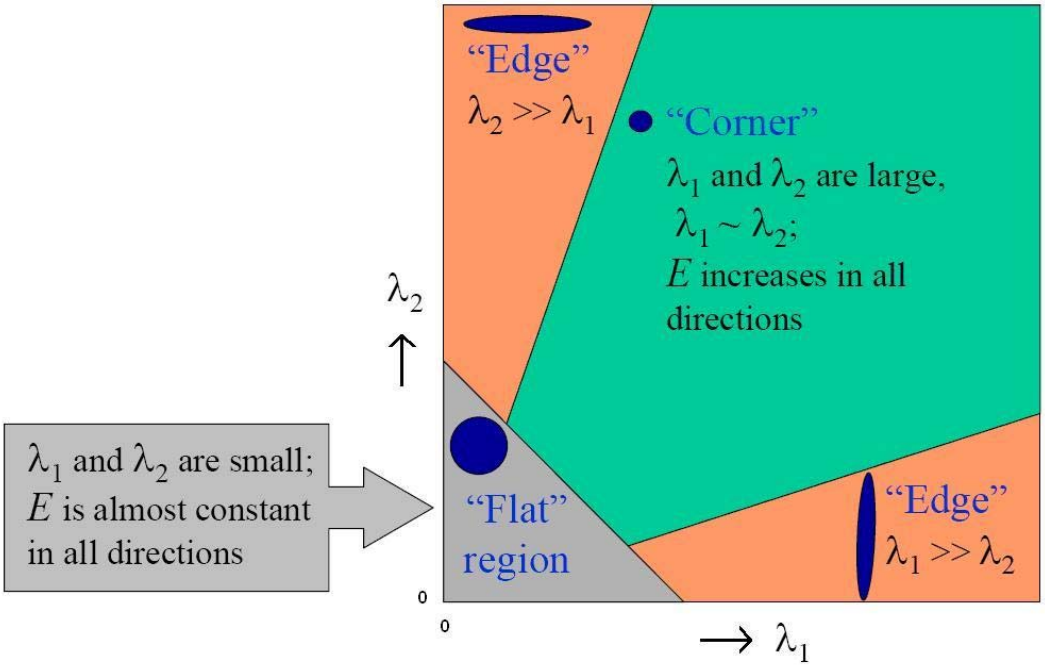


Figure 3.10: The Eigen values of the correlation matrix determine the type of the pixel: edge, corner or flat region.

The image pixels can also be classified by the Eigen values λ_1 and λ_2 of the correlation matrix *M*. The values of λ_1 and λ_2 depend on the factors *A, B* and *C* from formula 3.12. If these factors grow, the Eigen values grow and if the ratio between the factors changes, then the ratio λ_1: λ_2 changes too. Three cases are found here:

1. If both Eigen values are large, the gradients are large in both directions; a corner pixel is found.
2. If the ratio is large, the gradient is large in one direction; an edge pixel is found.
3. If both Eigen values are small, the gradients are small in both directions; a flat region pixel is found.

As demonstrated in figure 3.10 flat regions have small Eigen values and the corner regions have high Eigen values. If one of the two Eigen values is high we look at an edge point. The Eigen values are invariant to rotation, which means that one corner will give the same response in two rotated images. So the corner response is invariant to image rotation. The Harris detector is not invariant to image scale, because on a small scale a corner can be detected as a series of edges. So it does not provide a good basis for matching images of different sizes. The detector is partially invariant to affine[2] intensity change, because an intensity shift will not change the response but only add or remove a few points close to the threshold. These three properties are visualized in figure 3.11.

[2] **affine**: rotation and translation related.

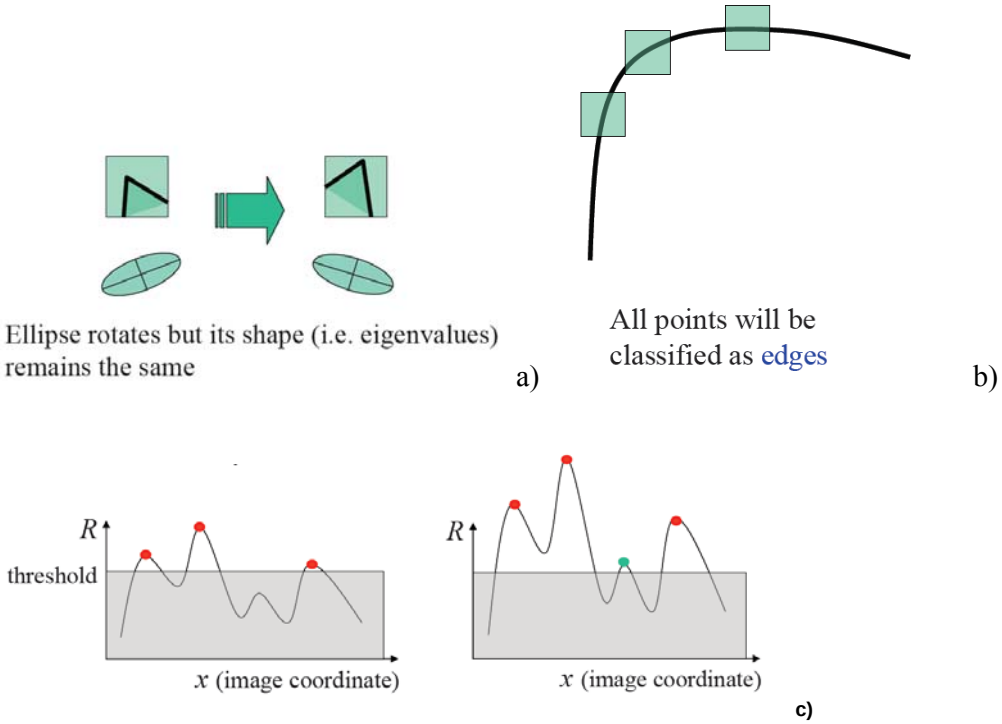


Figure 3.11: The response of Harris is **a)** rotation invariant **b)** not scale invariant and **c)** partially invariant to intensity change.

In [11] *Mikolajczyk and Schmid* a Harris-Affine detector is proposed as an improvement to the Harris detector. It is made more robust to affine deformations than the Harris detector, by making various shapes of elliptic windows to search for features. With the process of iteration the ellipses for every point of interest detected by Harris is constructed. With the ellipse parameters the response of the Harris-Affine detector is more invariant to affine deformations when used in feature based matching.

Another detector proposed by Mikolajczyk and Schmid is the Harris-Laplace detector. It uses the scale-adapted Harris function which makes the detector more robust to resolution differences. The scale adapted Harris function is just like the classical Harris based on the Hessian matrix M, but in this function a scale factor σ is taken into account. The response of the Harris-Laplace detector is more resistant to scale differences when used in feature based matching.

3.4.3 Canny edge detector

The Canny edge detector takes a greyscale image as input and produces as output an image showing the positions of the intensity discontinuities that were found. At the time he started his work Canny's intentions were to optimize the many edge detectors already existing. To achieve this he selected three criteria.

The first criterion is to have a low error rate, which means that edges occurring in images should not be missed as much as possible and that there should be no responses to non-edges. The second criterion is that the location of the edges must be determined accurately. The distance between the detected edge and the real edge should be minimal. Finally the third criterion is that multiple edge responses should be eliminated. This criterion was made because the first two do not exclude the possibility of multiple responses to one edge. The canny detector performs six steps:

1. Image smoothing
2. Determination of image gradient
3. Computation of gradient direction
4. Classification in main directions
5. Selection of maxima
6. Hysteresis

The first step is that the Canny smoothes the image to eliminate most of the noise. The Gaussian filter (section 3.3.3) is used in this step to smoothen the image. The Gaussian function is approximated in a discrete way by using a mask. It is shifted over the image as explained in section 3.2.2. The larger the mask the more noise will be removed. The counter-argument for a large mask is that details in the image are lost.
During the second and third step the detector finds the image gradient to highlight the regions with a high spatial derivative and its direction. Different operators can be used, for example the Sobel operator. In the middle of figure 3.5 the Sobel operator as discrete masks for the x ('**Gx**') and y ('**Gy**') direction is shown. The magnitude or edge strength of the gradient is then approximated by:

$$Grad = |Gx| + |Gy| . \tag{3.14}$$

The edge direction θ is computed by:

$$\vartheta = \tan^{-1}\left(\frac{Gy}{Gx}\right). \tag{3.15}$$

In the fourth and fifth step the algorithm tracks along these regions and suppresses any pixel that is not at the maximum. This phase is called non-maximum suppression.
First (in the fourth step) the computed directions θ are related to a main direction which can be traced in an image. From one pixel to its neighbour there are four possible directions:

- 0 degrees (in the horizontal direction),
- 45 degrees (along the positive diagonal),
- 90 degrees (in the vertical direction), or
- 135 degrees (along the negative diagonal).

For every direction θ the closest pixel direction is determined. Now (in the fifth step) it is possible for the algorithm of non-maximum suppression to trace along the main directions and remove (or set to zero) all points that don't have a gradient maximum. All the remaining pixels result in a thin line following the pixels with the highest gradient along the edge.
In the sixth step the gradient array is further reduced by the so-called Hysteresis process. The process of Hysteresis performs a double-threshold check along the remaining pixels that have not been suppressed in step 5. For the image two thresholds (T1 and T2) are defined. If the response is below the low threshold T2, it is set to zero, which means that it is set to be a non-edge. If the magnitude is above the high threshold T1, it is made an edge immediately. And if the magnitude is between the two thresholds, then it is set to zero unless there is a path from this pixel to another pixel with a gradient above the high threshold T1. In the algorithm process this means that an edge point (above threshold T1) is selected and from this point every neighbouring pixel in the edge direction is tested with threshold T2. With the hysteresis the Canny detector avoids removing edges that are partly bad and partly good, which means that they lie partly below the threshold T1. Thresholds can be defined by looking at the image statistics. It is possible to make a histogram and choose the threshold at for example a certain percentage or a local minimum.
The final result of the Canny edge detector is a number of sharp edge contours (see figure 3.12). For feature point matching this result is made usable by *Cottier* (94). The basic idea is to extract Harris corner points out of the Canny response edges. First thin edges are retrieved with the Canny edge detector, and then the Harris corner and edge detector defines the edges and corners on the Canny output image.

Figure 3.12: Example of Canny edge detection.

3.5 Hough transform

The Hough transform (see [12] *Rabbani*) is a general method for finding objects like straight lines or circles hidden in larger amounts of other data. It is an important and much used technique in image processing. In this case the Hough transform can be used to cluster the feature points that lie on the same straight line, which is again part of an object. When these clusters are found and they together point out the same pose of an object, the probability that this interpretation is correct, is much higher than for any single feature point. In other words pixels that are found in group of points on the same line have a higher chance to be part of an object than any randomly selected pixel. Because bad features like noise and bended objects will not be part of the lines, this will improve the dataset of key points. This improvement again will lead to a more reliable and stable non-linear optimization (paragraph 4.2), because this method performs poorly when it has to deal with a lot of outliers. It can for example be defined that at least 3 features need to correlate in Hough space to be selected as a reliable feature for feature based matching.
To understand the Hough Transform, it is important to know what the Hough space is. Each point (p, θ) in Hough space corresponds to a line with angle θ and distance p from the origin in the original data space. Each image coordinate couple (x_i , y_i) is mapped to the Hough space by the relation:

$$p = x\cos\theta + y\sin\theta \tag{3.16}$$

Figure 3.13 shows an example of mapping three points on a straight line in the image to the Hough space. The points which are collinear in the Cartesian image space become apparent as they show curves which intersect at a common (p, θ) Hough point. The Hough points that are intersected by a lot of lines can be interpreted as a line in the image space. The value of the outcome of the Hough transform gives the point density along a line in the data space. The main advantage of the Hough transform is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.

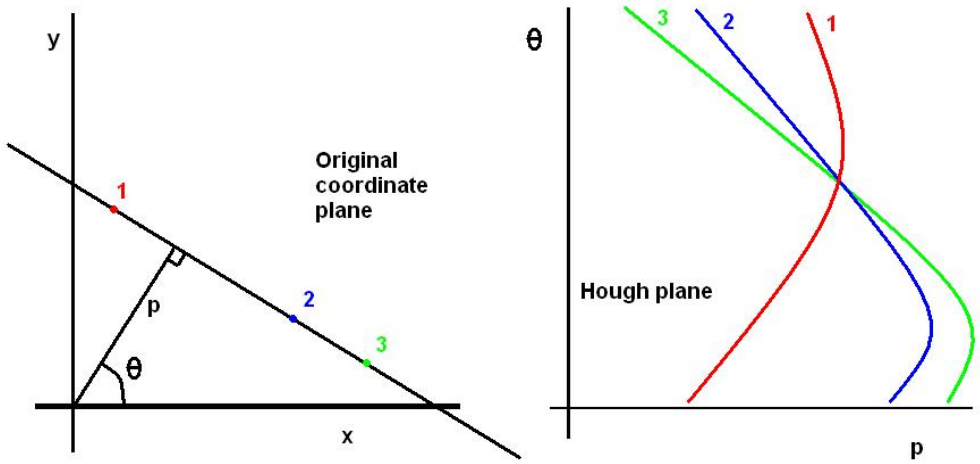


Figure 3.13: Hough transform: 3 points on 1 line represent 3 lines in the Hough space that intersect in 1 point.

The output of the Hough transform depends on finite intervals or *accumulator cells*. When the Hough space is created it can be divided into these cells. By looking at the intensity of their content, peaks are found which represents a line. To select these peaks a threshold is used. The size of the cells is defined by the angle resolution and the pixel resolution (respectively the vertical and the horizontal axis in the Hough space).

3.6 Local feature descriptors

For fully automatic registration of the laser scan data and the digital images local feature descriptors are very useful. They are able to match features in both datasets. The main idea of a descriptor is that it gives a discrete formulation (description) of the feature properties in such a way, that it can be compared with features in other images. For automatic registration a descriptor has to fulfil 3 demands:

1. It should be invariant to ordinary geometric transformations such as rotation, translation and uniform scaling.
2. It should be robust against noise.
3. It should capture the shape information sufficiently, so that a fast best matching procedure can retrieve the correspondence.

The length of the feature boundary for example is one of the simplest properties that descriptors can use. A more complicated property is the ratio, called the eccentricity of the boundary; an edge is placed in a box with the ratio of the major and minor axis as a useful description.

Besides these examples a large number of properties can be used to give good description of a key point. During the literature study of this research, different descriptors are studied. Many different methods have been designed to perform reliable matching of features and more than one descriptor may be used in one method. In the following a few descriptors from the literature study are described.

Shape context descriptor

The shape context descriptor is a method that uses the shape properties of the detected features like curvatures, corners and edges. The method computes a 3D histogram of location and orientation for the edge points. This descriptor has been successfully used in drawings in which edges are reliable features, but does not work well when matching shapes in a background clutter[3]. It is therefore typically used in uncluttered scenes in which a heavy majority of the points are known to belong to an object.

Steerable filters

The method of steerable filters basically *steers* the derivatives of the detected points and there neighbourhood in the direction of the local gradient. When looking at figure 3.14, it is seen that the four neighbouring pixel locations and their filter responses are steered to create a rotational invariant descriptor. This descriptor is not scale invariant.

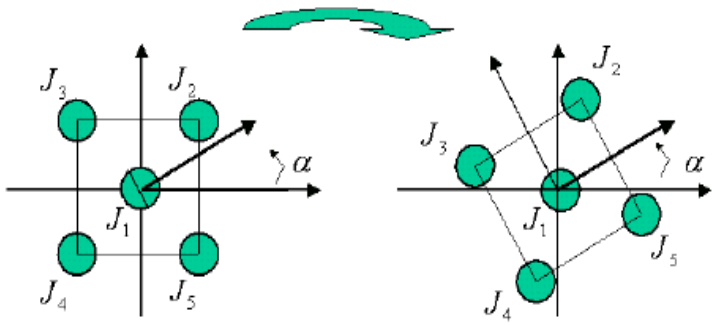


Figure 3.14: Steerable filters: the 4 neighbours are rotated to the direction of the local gradient.

Spin images

This descriptor is a histogram of pixel locations and intensity values in a cylindrical perspective. All detected points are projected in the cylindrical coordinates with respect to point of interest. The main axis of the cylinder is the normal of the point of interest. The spin image is created by unfolding the cylinder. This descriptor is only suitable in 3D scenes, because it is not possible to construct the normal in 2D images.

[3] **background clutter:** irregular objects in the background such as leaves or text on walls, that are not usable for matching two images.

Cross-correlation

When looking at the vector of image pixels of two edges, a very simple descriptor, cross-correlation can be computed and used to describe the similarity between the edges. The problem here is the large size of the vectors and the computational load.

SIFT

This method is called the scale invariant feature transform and is proposed by Lowe (see [9] *Lowe*). It combines a scale invariant region detector and a local descriptor which is based on the gradient distribution in the detected regions. Like the shape context descriptor it is represented by a 3D-histogram of gradient location and orientation. Paragraph 3.7 goes into detail about this descriptor.

PCA-SIFT

This method is recently developed by *Ke* and *Sukthankar* and is similar to SIFT. They have applied Principle Component Analysis to the image gradient patch derived in the SIFT method. PCA is a statistical method that extracts uncorrelated components from a normalized data set. Applying this to the gradient patch gives the detected point a good description. This method has proved to outperform the original SIFT on artificially generated data, but for real data SIFT gives better results.

GLOH

The Gradient location-orientation Histogram is an extension of the SIFT descriptor. It is designed to improve its robustness and distinctiveness by using a log-polar location grid instead of a Cartesian grid. Figure 3.15 shows an example of such a grid. In [10] *Mikolajczyk and Schmid* it is proven that GLOH performs slightly better than SIFT. Disadvantage is the programming complexity of the log-polar region.

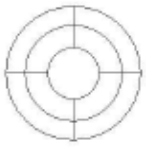


Figure 3.15: The log-polar location grid used by GLOH.

Not only many different descriptors have been proposed in the literature, also many have been compared and evaluated. A broad performance evaluation for example was carried out by Mikolajczyk and Schmid in (see [10] *Mikolajczyk and Schmid*). The research proved the SIFT-based methods to obtain the best results. Shape context descriptors show good results too, but they fail in scenes with many unreliable edges, which is very often the case when using a laser scanner (scenes with much background clutter result in incomplete and noisy edges). The research also states that region based descriptors perform better than point-wise descriptors.

3.7 Scale invariant feature transform (SIFT)

Because SIFT-based methods proved to be reliable and usable for automatic registration, this paragraph will go into more detail about this method. The algorithm is developed by [9] *Lowe* in 2004 in Canada. It includes not only the descriptor, but it also handles detection of good features. It returns a large number of features that densely cover the image at all scales and locations in the image. A typical image size of 500x500 pixels will give typically about 2000 stable features. This number off course still depends on the content of the image and on the choice of some parameters.
Basically the process of SIFT can be divided in four major computations, where step 4 contains the actual descriptor:

1. Scale-space extrema detection
2. Key point localization
3. Orientation assignment
4. Key point descriptor

In the first stage, scale-space extrema detection, the algorithm searches points of interest in all image locations and over all scales. For this the difference-of-Gaussian function is used (see formula 3.10). It identifies potential interest points that are invariant to scale and orientation.
The second stage looks at all interest points that are found. To all candidate locations a detailed model is fit to determine location and scale. With a measure of the quality of the fit the best key points are selected.
Then again the third stage is only looking at the points returned from the previous stage. By looking at the local image gradient directions every key point gets one or more orientations.

In the fourth stage the local image gradients of the region around every key point is measured at the scale of that key point. From there the algorithm is able to come up with a representation of the orientation and magnitudes of the image gradients around the location of every key point. Finally this representation is used for a valid matching of two points from different images. The scale at one point is used to select σ in the DoG function. When matching the features the computation becomes more rapid by using nearest neighbour algorithms to find candidate matching features. Because the descriptors are highly distinctive, it is possible to find the correct match in a large set of features with high probability.

Detection of scale-space extrema

As mentioned before, this is the first stage of the SIFT method. The aim is finding interest key points that can be assigned under different views of the same object. Identifying their location and scale can be accomplished by searching for stable features across all possible scales. For this a continuous function of scale (called a scale-space) is needed. For this scale-space function *Lowe* uses the Gaussian function in a discrete kernel. Thus the scale-space of an image is defined by $L(x,y,\sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) \otimes I(x, y), \tag{3.17}$$

where $*$ is the convolution operation in point (x, y), $I(x,y)$ the input image intensity at point (x, y), and $G(x,y,\sigma)$ the Gaussian function:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{3.18}$$

For an efficient detection of key point *Lowe* (1999) proposed to use the difference-of-Gaussian function $D(x,y,\sigma)$ to find scale-space extrema. $D(x,y,\sigma)$ is computed from the difference of two adjacent Gaussians $L(x,y,\sigma)$ separated by a constant factor k:

$$\begin{aligned} D(x, y, \sigma) &= L(x, y, k\sigma) - L(x, y, \sigma) \\ &= \left[G(x, y, k\sigma) - G(x, y, \sigma)\right] \otimes I(x, y). \end{aligned} \tag{3.19}$$

For construction of D each octave of the scale space (i.e. doubling of σ) is divided in s intervals, so that $k = 2^{1/s}$. The second and next octave (with two times the previous σ each step) is produced by taking every second pixel in each row and column. As shown in figure 3.17 all adjacent images $L(x,y,\sigma)$ are subtracted to produce the difference-of-Gaussian function $D(x,y,\sigma)$.

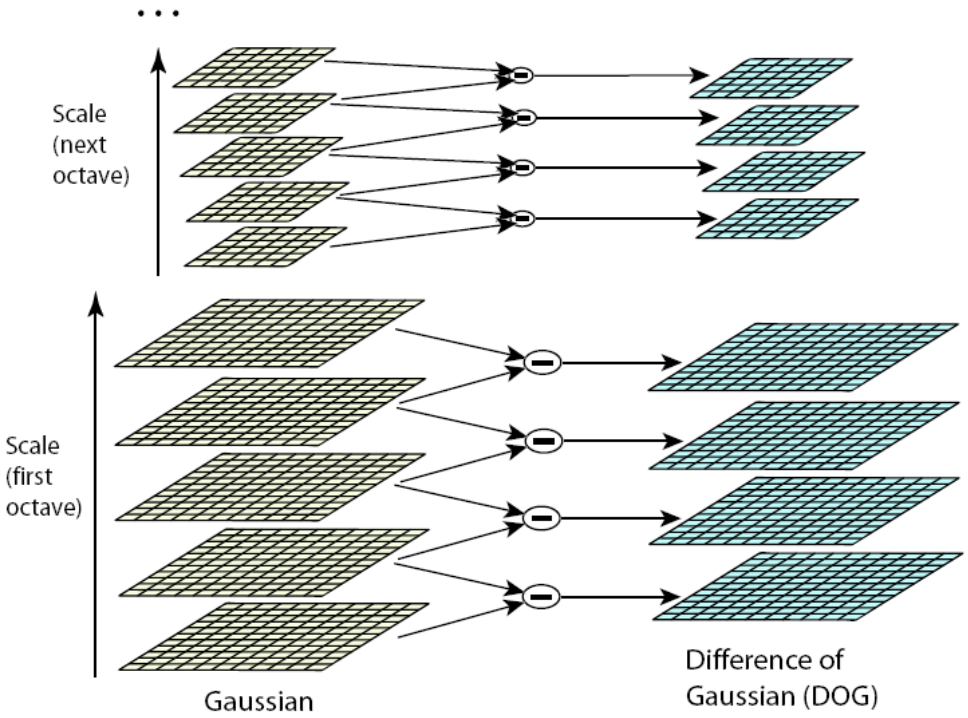


Figure 3.17: The construction of DoG at different octaves (Source: *Lowe* [9])

After the construction of *D* the local minima and maxima are detected by comparing every pixel with his 8 neighbours at the same octave and the 9 neighbours in the octaves below and under this scale (see figure 3.18). Only when all neighbours are either larger or smaller, the pixel is selected as a local extrema. This effort is largely reduced because most points will be eliminated in the first checks.

The most confronting problem of this detection process is that the reliability depends on the frequency of sampling in the image. The location of extrema is a continuous function of the image. When they are close to each other a higher sampling frequency is needed to get a stable result. The determination of a solution between efficiency and stability can be done experimentally, but that is a disadvantage with respect to the automation process. It is advisable to replace the type of detector (with Harris or Canny for example), when dealing with automatic registration.

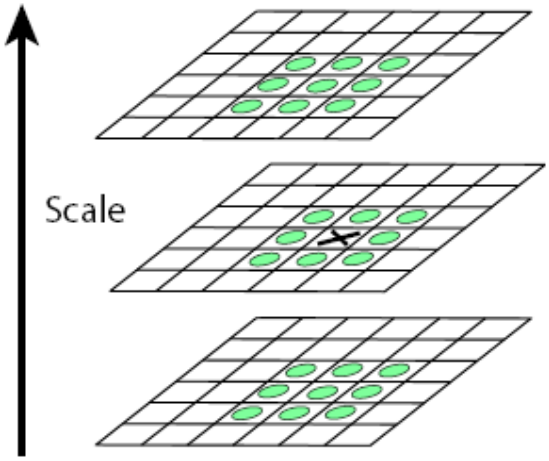


Figure 3.18: Every pixel is compared the direct neighbours: including one scale above and beneath the point.

Accurate key point localization

After the local extrema detection the next step is to make a more accurate localization of the detected key point. The aim of this step is the possibility to reject the points that have low contrast or are poorly localized along an edge. Initially the location and scale of the key point would be the ones on the central sample point from figure 3.18. But recently *Brown and Lowe* [3] have developed a method to interpolate the location of the extrema by using a shifted Taylor expansion of the scale-space function *D*. By setting a threshold the offset (determined by the shift) is used to reject key points with a low contrast. Removing the low contrast points is not sufficient. With the Eigen values of the Hessian matrix as derived with the Harris detector the edge responses can be found and eliminated. Unlike the Harris detector (using determinant minus trace^2), SIFT derives and uses the quantity $(r + 1)^2 / r$, where r is the ratio between the largest and the smallest Eigen value (the principle curvatures), so that $\lambda_1 = r \cdot \lambda_2$. Formula 3.20 derives how r is related to the Eigen values.

$$\frac{Tr(M)^2}{Det(M)} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} = \frac{(r\lambda_2 + \lambda_2)^2}{r\lambda_2^{\,2}} = \frac{(r+1)^2}{r} \cdot \frac{\lambda_2^{\,2}}{\lambda_2^{\,2}} = \frac{(r+1)^2}{r}. \tag{3.20}$$

In the experiments of *Lowe* r was set to 10 as a threshold, so that all key points that have a ratio between the principle curvatures bigger then 10, are selected as edge response and eliminated. This is simply done with the check in formula 3.21:

$$\frac{Tr(M)^2}{Det(M)} < \frac{(r+1)^2}{r} \tag{3.21}$$

Orientation assignment

In this step the aim is to assign a consistent orientation to every key point. First the scale of the remaining key points is used to select the Gaussian smoothed image *L* with the closest scale, so all computations are scale invariant. Then for each *L(x,y)* at this scale, the magnitude *m* and orientation θ of the gradient is computed by using a pixel shift:

$$m(x,y) = \sqrt{\left(L(x+1,y) - L(x-1,y)\right)^2 + \left(L(x,y+1) - L(x,y-1)\right)^2}$$
$$\theta(x,y) = \arctan\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \tag{3.22}$$

From *m* and θ within a region around the key point a histogram is made. The gradient magnitudes and a circular Gaussian window (σ is 1.5 times that of the used scale) are used for weighting each sample in the histogram. As a result one or more peaks can be detected in the region around the key point, so the dominant gradient directions are known. Only the peaks within 80 % of the highest peaks are assumed to be dominant. The points that come out assigned with multiple orientations contribute significantly to the stability of the matching. In addition a parabolic interpolation is used to find a more accurate peak position.

The local image descriptor

Finally the last step of the SIFT method contains the construction of the descriptors. Together all previous operations have resulted in an image location, scale and orientation to each key point. Now that the local image regions of the points are invariant to change in intensity and 3D viewpoint, they are ready to be represented for matching. Therefore the descriptor must be computed for every point.

Figure 3.19 illustrates the computation of the key point descriptor. On the left side the orientation assignment is shown. The circle illustrates the Gaussian weighting window. All orientations are put in one of the 8 main directions (orientation bins) as shown in the right part of the figure. The sum of the magnitudes determines the length of the vector in every main direction. A 2x2 descriptor is shown, but testing has shown that a 4x4 array of histograms with 8 orientation bins achieve the best results.

Now we end up with a 4x4x8=128 element feature vector, which is finally normalized to unit length to reduce the effect of intensity change. The normalization cancels out the possible difference in brightness. Because a brightness shift does not affect gradient values, the descriptor becomes invariant to affine changes in intensity.

Still non-linear intensity changes might occur due to camera saturation for example. These effects can cause large changes in the gradient magnitudes, but small changes in the gradient orientations. Therefore the unit vector elements are cut of at a minimum magnitude and then the vector is renormalized.

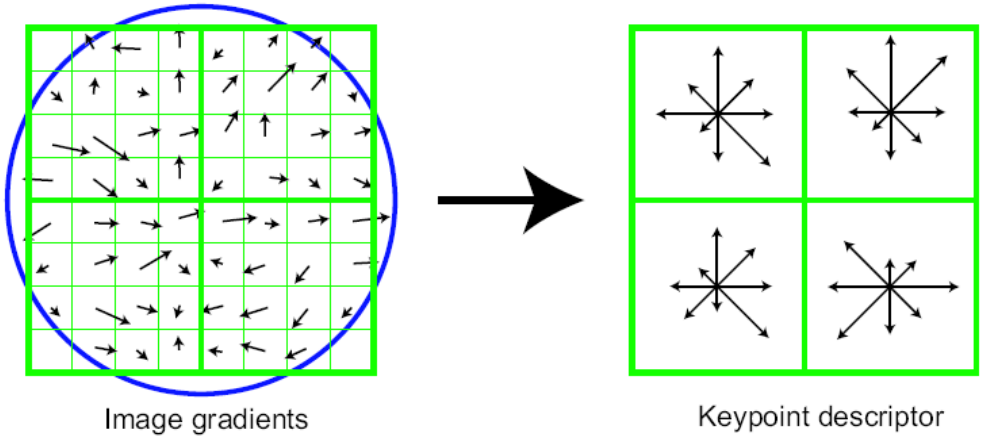


Figure 3.19: Construction of the SIFT descriptor.

Key point matching

When the descriptors are made of each key point in the images, the matching can be performed. A good match is found by identifying its nearest neighbour in key point database. The nearest neighbour is defined by the minimum Euclidian distance between two invariant descriptor vectors.

In practice the problem is that many key points will not have any correct match, because they arise from background clutter or were not detected in both images. To safeguard reliability in the matching these features must be discarded. This can be done by implementing a threshold. Just one threshold might not yield a good result, as some descriptors are much more distinctive than others. A better way to measure the quality of the match is by looking at the second nearest neighbour. Because a good match must be significantly closer than the closest incorrect match, this method is more distinctive. On top of that false matches will have a number of other false matches within similar distance due to the high dimension of the feature space.

Besides matching points on there descriptor values it is also possible to set a constraint on there location. Because the position of the camera with respect to the laser scanner is approximately known, the search for good matches can be reduced to a smaller area. The advantage is large, because the descriptors only need to be distinguishable with respect to their close environment. An additional advantage is that the algorithm becomes faster, because it does not need to search through the complete database of key points.

This chapter has not only explained the basics of digital image processing, but has also described operators, detectors and descriptors. The theory has been written into the direction of the method design, so that some subjects have been worked out into great detail. This design is the result of the literature study and presented in the next chapter. The detailed theory will be useful during the implementation of the design in chapter 5.

4 Method design for automatic registration

In this chapter a method design is proposed for automatic registration of the Imager 5003 laser scan and the colour images from a digital camera. The method is based on the literature study (exposed in chapter 2 and 3). According to this design the implementation in C++ will start and hold on, but during the research it is possible to make different choices that move away from this design.
Paragraph 4.1 works out the structure of the designed method and it also explains why different choices are made. In paragraph 4.2 additional theories are written on the non-linear rigid optimization, which is chosen to be the method to estimate the connection between the laser scan and the camera image.

4.1 Algorithm structure

Figure 4.1 illustrates a schematic overview of the algorithm process designed for this research. It starts with the input of the measurements and ends with the reconstruction of the point cloud including RGB colour values of the images. This structure is used and operational in [2] *Bendels*.
The camera is assumed to be fixed to the Imager 5003 in such a way, that only a small baseline exists between them and that they will have approximately the same relative position. This brings the advantage that the objects in both datasets concern only a small affine transformation and simplifies the choice of detectors and descriptors.
The input of this system is one laser scan (coordinates and intensity) and a number of digital coloured images. Because the resolution of the digital camera is higher than the resolution of the laser scanner, the matching has to deal with this difference in resolution (≈ factor 2). Note that this difference is a constant factor, while the small baseline between the camera and the scanner is not changing during the measurements.
Still because of the resolution difference the Harris detector (see section 3.4.2) might not be suitable for this application because it is not scale invariant. This design therefore contains the Harris-Laplace detector as an alternative. In [11] *Mikolajczyk and Schmid* prove that the Harris-Laplace detector performs better than the Harris-Affine detector in the presence of uniform scale changes. The Harris-Affine detector rejects many points with correct scale and location, but with a highly anisotropic shape. To overcome the scale dependency Mikolajczyk and Schmid propose the Harris-Laplace detector.

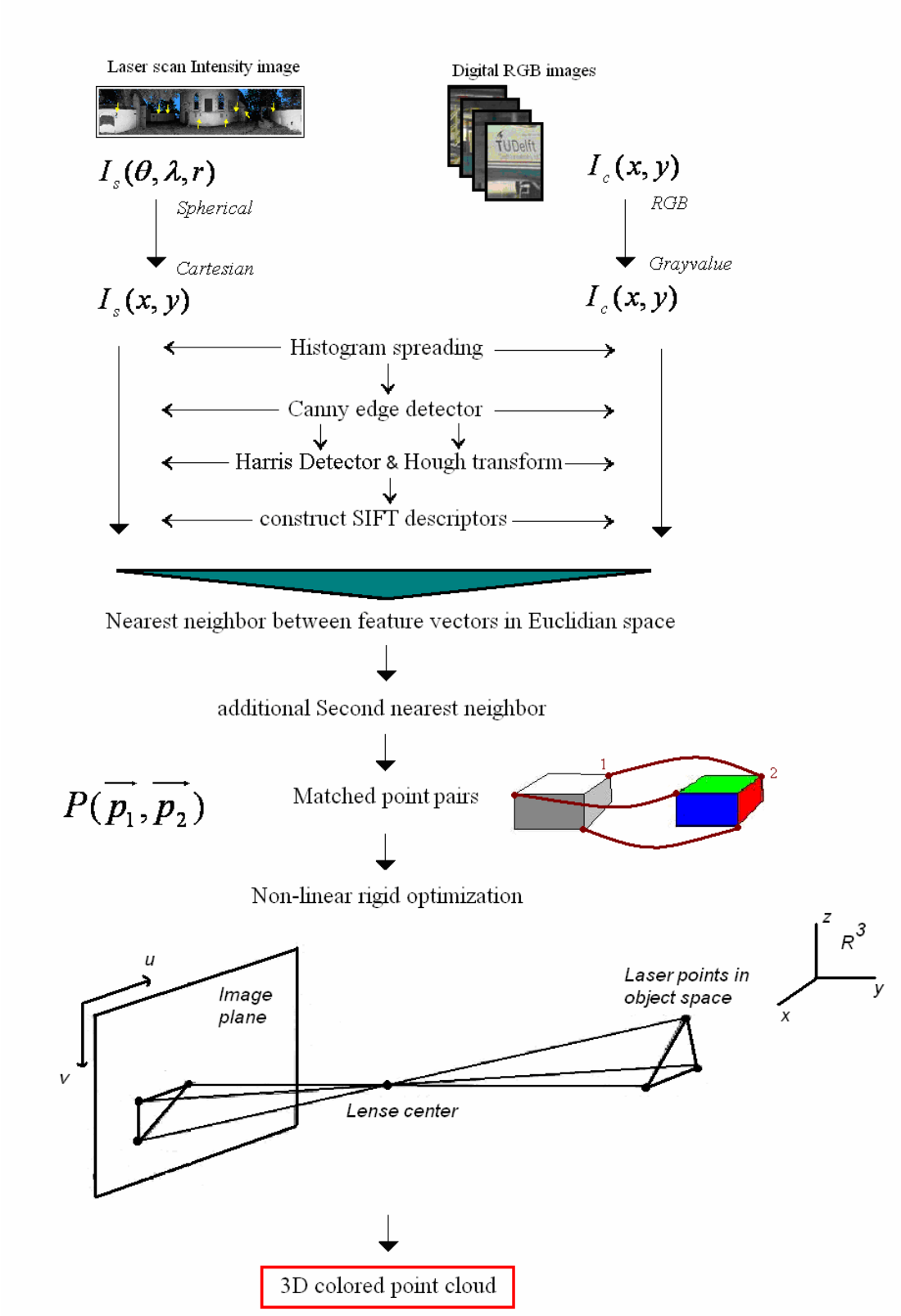


Figure 4.1: Schematic overview of the designed algorithm process.

The Canny edge detector is commonly used because of its improved criteria (see section 3.4.3), but the responses of this detector are only edges. The edges are also not guarantied to be scale invariant. To solve this and to find the best quality key points in an image, the method design makes use off an approach that is comparable to *Cottier* (a combination of Harris and Canny). Where *Cottier* uses the non-scale-invariant Harris detector, this method design implements the Harris-Laplace detector on the Canny edges.
After the Harris-Laplace detector has ensured the scale invariance, the Hough transform will be used to further improve the quality of the key point dataset. When clusters of feature points are found and they together point out the same pose of an object, the probability that this interpretation is correct is much higher than for any single feature. Unmatchable key points from the background clutter will hardly survive this cluster selection and thus be removed from the key point dataset.
All together the Canny, Harris-Laplace and the Hough transform will lead to the selection of key points. So at this point the collection of key points is finished. But before the key points are ready to be matched, they need a descriptor to become distinguishable. For the descriptor design the SIFT is used. According to *Mikolajczyk and Schmid* [10] the SIFT based descriptors have proved to give the best results. Because PCA-SIFT (Principle Component Analysis SIFT) is more suitable for artificial databases and GLOH (Gradient location-orientation histogram) with his more complex log polar region description does not outperform normal SIFT, Lowe's descriptor is planned to be used for matching the key points in the digital and scan images.
This method design uses a combination of Canny, Harris-Laplace, Hough and SIFT. The detection and localization of the key points is taken care off by the Canny and Harris detectors. The Hough transform is used for clustering the key points. Then for every selected key point the SIFT descriptor is constructed as in Lowe's algorithm to bring corresponding points together.
After the construction of the descriptors the key points are ready to be matched. Nearest neighbours of the descriptor's feature vectors are searched in Euclidian space. Additionally also the second nearest neighbour can be looked upon, but because this is mostly useful for noisy images it might be redundant for this application.
When the points are matched, all point pairs will be used as measurements in a non-linear optimization to estimate the external camera parameters in the 3D laser scan space. Then finally all the colour information in the digital images can be allocated to the 3D laser points. Because the resolution of the colour images is higher than the resolution of the scan, every laser point will be coloured by a number of pixels. The final colour will be made from the average of these pixels.

4.2 Non-linear rigid optimization

Until now all effort has been put into finding reliable distinctive features from both the digital images and the laser scan. When all the methods have succeeded into a dataset of matched key points, the next step is to use these point pairs to place the 2D digital images into the 3D laser scan space. To accomplish this, a non-linear rigid optimization is used. This method is chosen because it is most commonly used by the already existing Z+F software.

With this optimization it is possible to make a visual reconstruction, i.e. a camera pose is produced in which the external parameters (rigid orientation and relative position) are optimized. In other words the optimization derives the geometric constraints on the rigid transformation from the camera body system to the laser coordinate system. The rigid external parameters describe this transformation, in contrast with the internal (calibration) parameters that describe the transformation between the pixel coordinate system and the camera body system.

The word *optimization* in this case means that all parameter estimates are found by minimizing the model fitting error. The model fitting error is defined as the Euclidian distance in the camera coordinate system between the image point and the 3D laser point.

To formulate the problem accurately, a more detailed look to the basic equations is necessary. First of all the camera can be seen as a projection from the scan world coordinates $[X,Y,Z]^T$ to the image coordinates $[u,v]^T$ in the following formula 4.1:

$$[u,v]^T_{color} = K\left(R_{ort} \cdot [X,Y,Z]^T_{laser} + \bar{t}\right), \tag{4.1}$$

with K the matrix with the internal (calibration) parameters of the camera, R_{ort} the 3x3 orthonormal matrix representing the camera's rigid orientation and t the vector for the relative position of the camera with respect to the laser scanner reference frame.

In this project the camera is assumed to be already calibrated and so matrix K is well known. Thus the goal of the optimization is to find the external parameters; three angles in matrix R_{ort} and three translations in vector t. The minimizing problem requires good start values for these parameters, so that it will converge to the solution with the minimum error. Now the error is defined as the Euclidian distance in the image between the image point and the 3D laser point for every point pair i. To compute the Euclidian distance in the image, the 3D laser points are rotated, translated and finally projected into the image by matrix P. The optimization then iterates the sum of the squared distance to the minimum.

$$\min \sum_i \left[P\left([X_i,Y_i,Z_i]_{laser}\right) - [u_i,v_i]_{color}\right]^2 \tag{4.2}$$

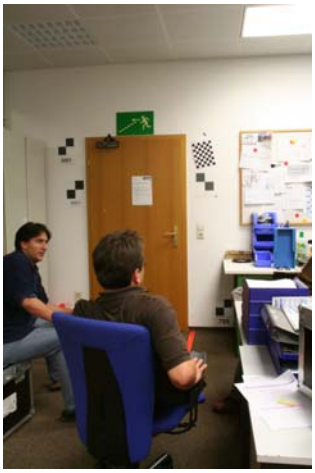
A non-linear rigid optimization as a stand alone does not need to be absolute. It will be precise, but in a relative way. When there are no absolute references involved the absolute accuracy is unknown. The 3D coordinate frame itself will be uncertain. All outcomes will follow this uncertainty. For this application this is not a problem, because only the relative relation between the images and the laser scan is searched for. If this relation is known precisely, it will result in a precise fusion of 3D coordinates and RGB information. The accuracy of the coordinates will depend on the laser scanner, digital camera and the key point locations of all point pairs.

5 Implementation in C/C++

The previous chapter describes the background theory and finally proposes a strategy to achieve the goal of this project. The aim of this chapter is to point out how the designed algorithm is implemented. The programming languages C and C++ are used. Microsoft Visual Studio 2005 is used as programming compiler for the implementation of all operations on the images.
The main reason of this choice is the availability of the openCV (computer vision) library. This C++ library contains many useful algorithms for image processing, such as histogram equalization, smoothing and edge detection. The implementation is performed step by step in the order of figure 4.1, starting with the input images and ending with the result of the optimization. During the implementation also other algorithms that are not in figure 4.1 are used and planned processes have changed to solve problems or to improve the results. All processes and problems are described in this chapter.
Paragraph 5.1 describes the procedure of the data acquisition. In paragraph 5.2 the implementation of the key point selection is reported. Paragraph 5.3 contains the process of point pair selection and finally paragraph 5.4 concerns the non-linear optimization.

5.1 Data Acquisition

During the implementation process test images are used, which were made at Z+F in Wangen. In an office chamber a set up was made containing numerous different optical elements like windows, plants, desks, artificial targets and calibration chessboards. Figure 5.1.a gives an idea of the set-up area.


a)

b)

Figure 5.1: a) Test set up area: office and **b)** the IMAGER 5003 with the Canon 350D

The acquisition is performed according to the Z+F fusion of the laser scanner and a digital camera (see [1] *Abmayr*). The Imager 5003 with the Canon 350D digital camera fixed on it (see figure 5.1.b), was placed in the middle of the room and one full scan was made at middle resolution (10000x5000 samples). The acquisition is performed in two steps. In the first step the laser scanner turns 360 degrees and scans the complete room. Then in the second step the scanner turns back 360 degrees while the camera on the scanner takes a picture after every 25 degrees. This procedure results in 14 pictures and 1 scan.
The resulting intensity image has a size of 10109 x 4973 pixels. The size of every picture is 2304 x 3456 pixels. Every picture overlaps about 40 % of his neighbouring picture. All pictures together overlap one third of the intensity image. The overlap contains horizontally the full range of 360 degrees and vertically the range of 60 to 120 degrees. This is shown in figure 5.2.

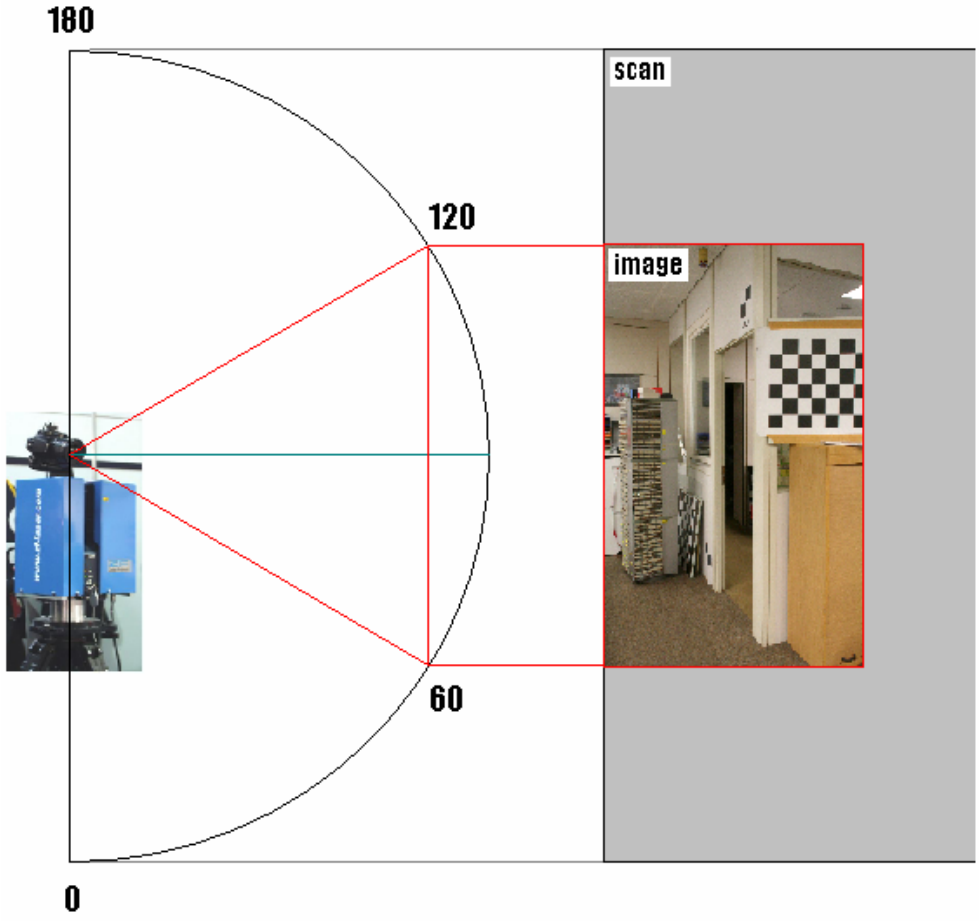


Figure 5.2: Illustration of the laser scan measurement and the image retrieval.

Before starting with the implementation a rectangle was cut out of the test space from both the laser scan and the corresponding image, because then the processing time becomes significantly smaller. Figure 5.3 shows an example of such a subset of the test data.

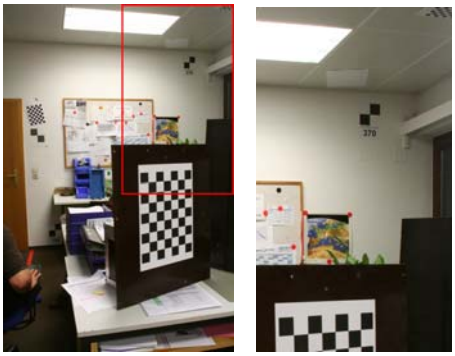


Figure 5.3: Working with a subset of the image data.

Furthermore the images are built up with pixel coordinates *(x,y)* in JPEG format. The origin is at the top left corner pixel. The 8-bit digital colour images contain 3 channels per pixel, respectively Red, Green and Blue intensity values: $I_c(x,y)_{RGB}$. The intensity image has one channel containing the returned pulse intensity: $I_s(x,y)$. With the function *cvLoadImage* both images are loaded as 1 channel greyscale images. The transformation from RGB to greyscale used in this case is:

$$\text{RGB} \rightarrow \text{Gray: } Y = 0.212671*Re + 0.715160*Gr + 0.072169*Bl, \tag{5.1}$$

With

Y	*the grey value,*
Re	*the red channel intensity,*
Gr	*the green channel intensity,*
Bl	*the blue channel intensity.*

The resolution of the colour image is assumed to be twice as high as the resolution of the intensity image. This is computed from the image sizes in table 5.1 and illustrated in figure 5.4. This assumption is used later on at the implementation of the descriptors.

Table 5.1: Calculation of the resolution factor between the intensity and colour image.

Intensity image:						
10000 x 5000 pixels (Colour overlap ≈ 33 %)				→	10000	x 1667 pixels
Every 25 degrees a picture				→	694	x 1667 pixels
Colour images:						
2304 x 3456 pixels (25 degrees overlap ≈ 60 %)				→	1382	x 3456 pixels
Pixel ratio	Laser : Colour	=	694 : 1382	≈	1 : 2	for x direction
	Laser : Colour	=	1667 : 3456	≈	1 : 2	for y direction

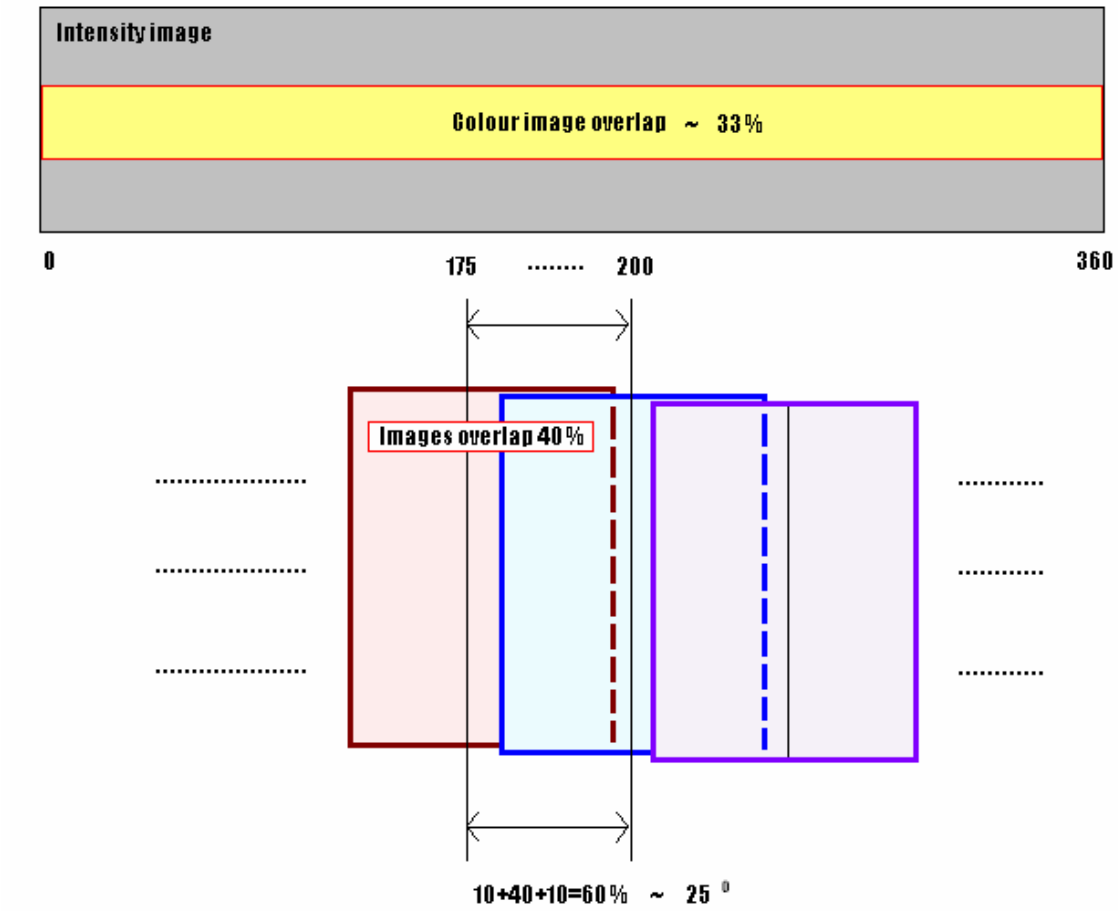


Figure 5.4: The colour images overlap 33 % of the intensity image and 40 % of the next image.

5.2 Process of key point selection

The first step in the algorithm procedure is the key point selection. From both the colour image and the intensity image distinguishable object points have to be found. The resulting sets of object points will be the input for matching point pairs in both images later on. This paragraph describes how all operations were tried and used in the process of key point selection.

5.2.1 Improving the images

When the images are loaded as greyscale images, it is seen that the intensity image is very dark and does not contain a lot of contrast. Before the search of features by Canny and Harris is started, it is wise to improve the images with respect to noise and contrast. To accomplish these improvements a Gaussian mask and histogram equalization are tried and investigated. The Gaussian mask removes the noise and the histogram equalization improves the contrast and the brightness.

The Gaussian mask is convoluted with the image with the function *cvSmooth*. This function takes the given input image and returns an output image depending on 4 parameters. The first parameter determines the type of smoothing, for which the Gaussian function is chosen because of his smooth properties. The second and third parameters define the size of the Gaussian kernel. The last parameter is used to define sigma, the standard deviation of the Gaussian function. If this parameter is set to zero, it is automatically calculated from the kernel size n by:

$$\sigma = \left(n/2 - 1\right) \cdot 0.3 + 0.8 \,. \tag{5.2}$$

The smoothing is investigated in combination with the histogram equalization. For the implementation the function *cvEqualizeHist* is used. This function simply asks for an input image and returns the equalized output image. There are no parameters involved.

Figure 5.5.b gives an example of the result of the noise and contrast enhancements on the intensity image by using *cvSmooth* and *cvEqualizeHist* (the original in figure 5.5.a). This result is not satisfying, because the geometry of the image changed. As a result of that Canny edge detector will react on the changed edges that occur and the Harris corner detector will react on the noisy surfaces. The corners are rounded too much due to the smoothing. The histogram equalization causes the noise on flat surfaces which the smoothing function is not able to remove without loosing too much detail. Some strong edges become weak and disappear from the Canny result. Additionally some weak edges (that do not contain good features) become strong and show up in the results.

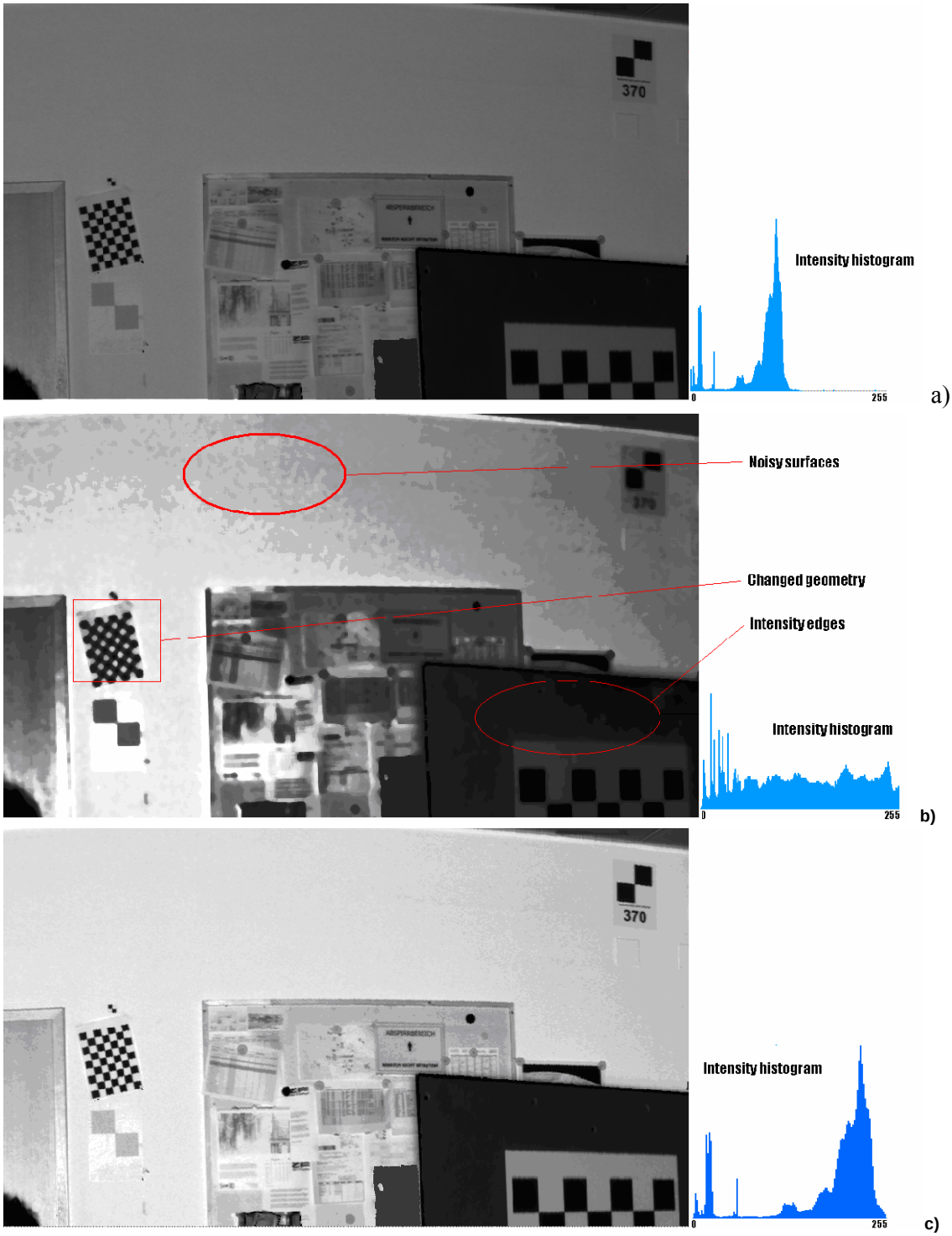


Figure 5.5: a) Original image and histogram **b)** Equalized image and histogram **c)** Stretched image and histogram.

Initially the smoothed and equalized image was the input for further processing. When the results turned out to be unsatisfying in the case of the intensity image, *cvSmooth* and *cvEqualizeHist* were neglected. Instead the smoothing is replaced by step 1 of the Canny algorithm (section 3.4.3), also a Gaussian operator. The histogram equalization is replaced by a histogram stretch (or spreading). This histogram stretch turns out to be sufficient as an enhancement of the input for Canny. The results of the equalization and the stretch are shown in figure 5.5.c. For the colour image the histogram equalization is used.

The stretch is done without an openCV function. It is a linear scaling over the complete range of 256 grey values as the intensities of the original image do not use the complete range (left side of figure 5.6). First the histogram of the original intensities is computed. Then the histogram is search through from left to right and from right to left, to determine the significant bins respectively on the left sight (*C_left*) and on the right sight (*C_right*). The horizontal red line in figure 5.6 is the threshold for this cut off. The threshold depends on the image size. Finally the new intensity is linearly scaled by multiplication with factor a_1 and addition of factor a_2:

$$I_{new}(x,y) = a_1 * I_{old}(x,y) + a_2, \tag{5.3}$$

with

$$a_1 = \frac{255}{(C\_right - C\_left)}$$

$$a_2 = -(C\_left).$$

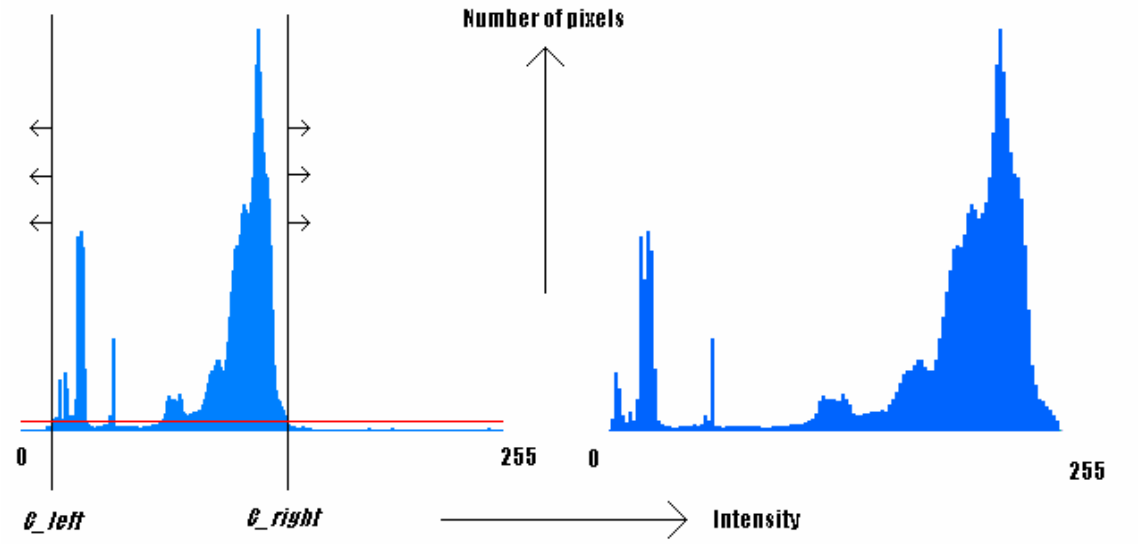


Figure 5.6: Histogram stretch.

5.2.2 Canny edge detection

For the implementation of the Canny edge detector the function *cvCanny* is used. It returns an image with edges depending on three parameters: the upper threshold, the lower threshold and the Sobel kernel size. The upper threshold determines the minimum strength of an edge that will be returned by Canny. An edge with strength above the lower threshold will be returned as an edge only if there is a connection with an edge pixel above the upper threshold. In this way, named hysteresis, Canny returns strong edges and avoids the removal of good edges with weaker parts. Hysteresis also helps to ensure that noisy edges are not broken up into multiple edge fragments. The maximum suppression (section 3.4.3) ensures the thin lines in the output image of Canny.

For the determination of the thresholds the intensity histogram of the input image is used. This histogram (figure 5.7) shows different peaks. Every peak contains a group of points with similar intensities. The thresholds are ideally placed in a local minimum of the histogram, because then it distinguishes different groups. A good way to automate this threshold selection is to compute a start value from a percentage of points above and below the upper and lower threshold respectively, because these percentages are more constant for different images than the intensity value itself. The start value is put on the percentage and then the closest local minimum is found by the following steps:

1. Select the start value.
2. Compute the sum of 10 bins left and right of the start value to see if the histogram is going up or down.
3. Follow the direction downwards (retrieved from step 2).
4. Compare bin after bin until one bin is higher than the last.
5. Set the threshold to the first last bin.

This minimum becomes the threshold value. This is repeated for both the upper and lower threshold. Figure 5.8 shows the result of the threshold selection on the colour image.
Because of the histogram stretch some of the bins in the histogram are empty. Therefore in step 4 the constraint is set that the selected bin contains at least 10 pixels. To prevent the algorithm stopping on possible noisy slopes of the histogram, a second constraint is set into step 4, namely that the last bin should be at least 0.2 % higher than the previous before going to step 5. In this way the algorithm can overcome minima that are too small to be a local minimum.

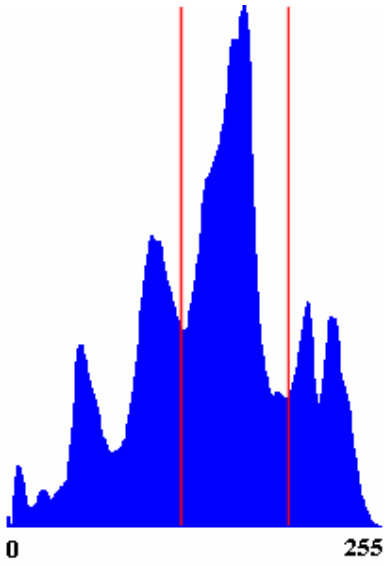


Figure 5.7: The thresholds are moved to the local minimum of the histogram.

At this stage the two thresholds are determined automatically. Still the Sobel kernel size remains. This parameter is set to 3. This value means that the kernel size is 3 x 3. This number was found by trying the possible inputs of the kernel size. Their outputs are plotted in figure 5.8. The result of kernel size 3 is the best, because the output contains much less background clutter than the outputs of 5 and 7. The Canny algorithm becomes more unstable when the kernel size grows. Therefore the smallest kernel is the best for distinguishing the sharp edges.

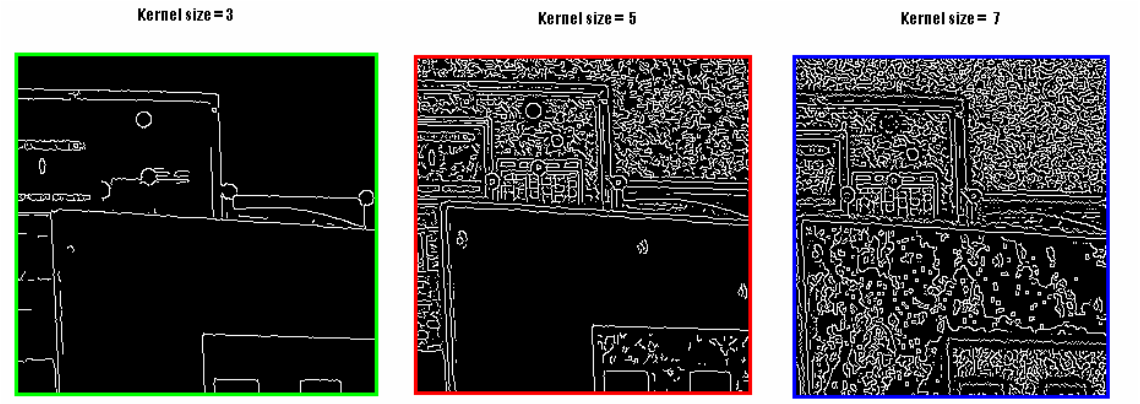


Figure 5.8: Results of Canny for different kernel sizes.

Now that all parameters are set and automated, the function *cvCanny* is called by the program. After that the output images are saved and released to prevent memory leaks.

5.2.3 Pruning

The result of canny is an image of object contours as can be seen in figure 5.8. A lot of these contours are very small or cut off halfway from an object. Some flat surfaces cause little noisy components. Pruning is a successful algorithm to remove these little line segments and parasitic components. This algorithm was not planned to be used, but it has shown to be a very useful way to improve the data.
The basic idea of pruning is that in a number of iterations the endpoints are shaved off or 'eaten'. When programming this algorithm there are two possibilities to find the end points of the lines. The first method is the morphologic convolution with eight different kernels. The kernels, shown in figure 5.9.b, are shifted over the image and due to the summation of the eight convolutions the end points at the eight different directions are found. By the convolution all end points are also removed. The number of iterations determines how much pixels are shaved off the lines.
In programming terms the second method for finding the end points is simpler. For all line pixels the number of neighbours is counted. The line pixel is selected as end point and deleted, if only one neighbour is found. Again if this action is iterated, the number of iterations determines how much pixels are eaten from the lines. Figure 5.9.c shows the result of this stage, which is the same for both methods.
Now after all the parasitic line components within the iteration length are deleted, the remaining line segments have to be restored to their original size. To accomplish this, the algorithm is turned around. Again all end points are found as described above (figure 5.9.d). Then these endpoints are dilated during the same number of iterations. The dilation is conditioned on the original line segments (figure 5.9.e). The parasitic elements that are completely deleted will not start to grow, because they do not deliver end points for the dilation. Finally the dilated end points are added to the 'eaten' segments (figure 5.9.c + 5.9.e) and so the remaining segments are restored in their original shape and size (figure 5.9.f).
The two methods described above, only differ in the way of selecting the end points. The main theory of eat and restore is in both approaches the same. During this research both tactics are programmed and tested, but the second method is finally used because it is simpler and faster.
When looking at the result of the pruning algorithm on the canny output, it is seen that a lot of noisy objects have been removed. The objects of interest remained and are ready for further processing. The next step is to find the corners of the objects with the Harris corner and edge detector.

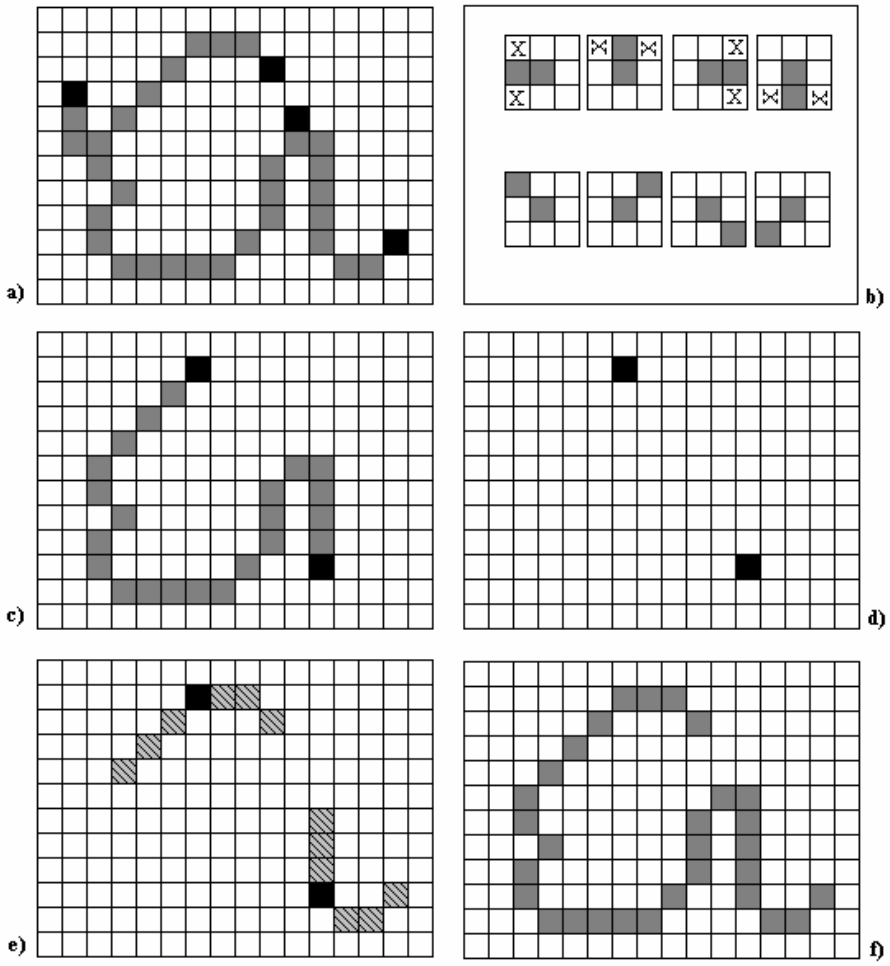


Figure 5.9: The pruning algorithm.

5.2.4 Harris corner detection

As computed in paragraph 5.1, the resolution of the colour image is about twice as high as the resolution of the intensity image. To solve this problem the Harris detector has to be either changed into the Harris-Laplace detector (Mikolajczyk and Schmid) as planned in the method design, either implemented for the two different scales. The Harris-Laplace detector can be used for every possible scale. But by enlarging the window for which Harris shifts the derivatives, it is also possible to overcome the doubled resolution. Figure 5.10 shows that a 3x3 kernel on the first resolution has the same size 5x5 kernel on the doubled resolution. Because the openCV library does not include the Harris-Laplace detector and because the resolution problem can be solved by this approximation, this last option is used.

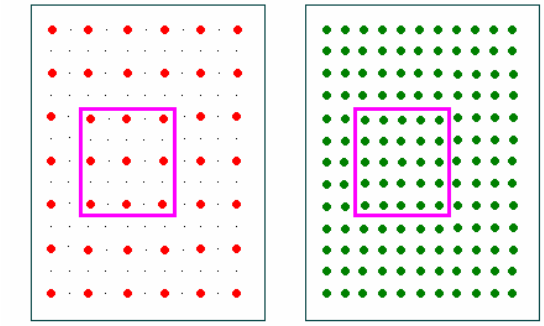


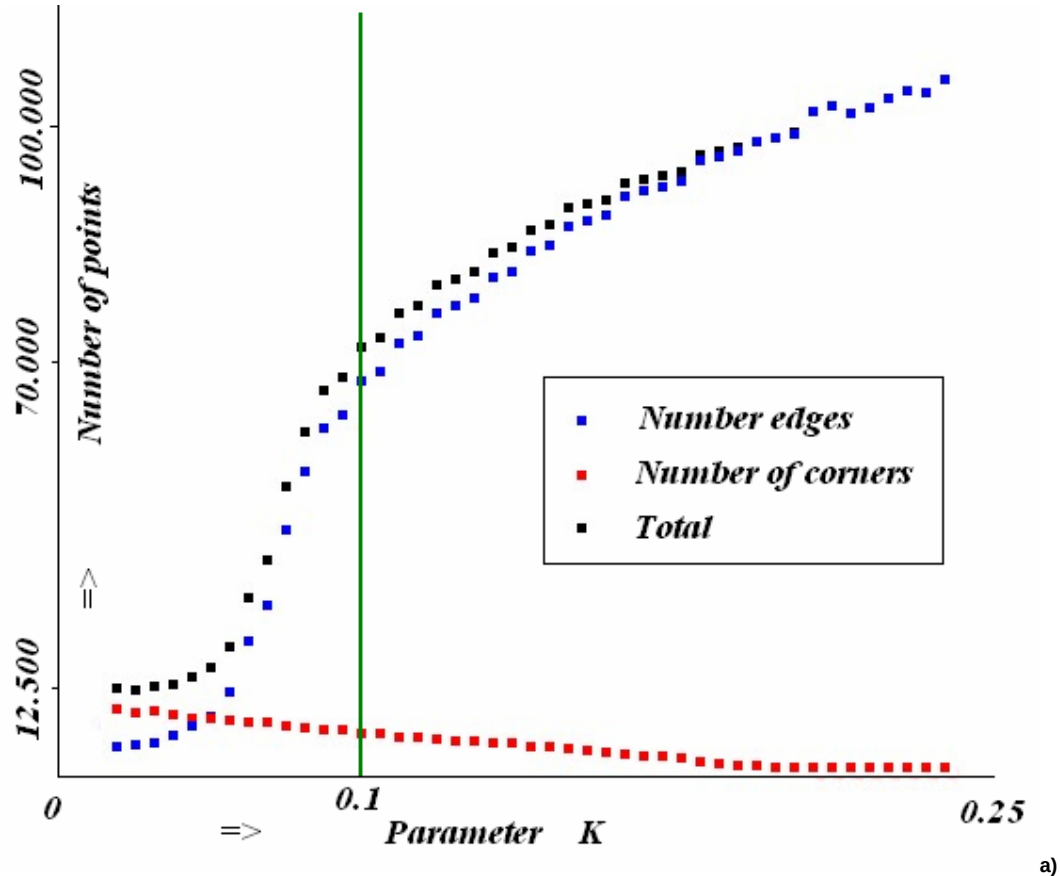
Figure 5.10: The resolution factor determines the kernel size.

For the implementation of the Harris corner and edge detector the function *cvCornerHarris* is used. This function has three parameters, namely the *kernel size*, the *Sobel kernel size* and parameter *k*.

The kernel size is the size of the neighbourhood, over which the Hessian matrix (covariance matrix of the derivatives) is computed. For the intensity image a 5 x 5 kernel and for the colour image a 9 x 9 kernel is implemented. The derivatives are computed with the Sobel operator. The size of the Sobel kernel is set to 3 x 3, the same size as in the Canny function. This choice is motivated in section 5.2.2.

At last the response of Harris is dependent on the parameter *k*. To see the behaviour of *k*, this parameter was plotted against the number of edges and corners returned by the function. Figure 5.11 shows the result for the colour image (**a**) and the intensity image (**b**) of the subset of the data (figure 5.3). The red points are the number of corners and blue points the number of edges. The black points give the total number of points returned by the Harris function.

The choice of *k* depends on the purpose of using the Harris detector. When *k* is too small the number of edges becomes very low, but many corners are found. For a large value of *k* Harris returns many edges and no corners. When only edges are searched for, it is recommendable to take a high value and when corners are searched for, a low value will give the best result. But in this case when both edges and corners are of interest, it becomes more difficult. For the colour *k* is optimal just after the number of edges steeply increased (at the green line), because at that point a lot of edges are won and still enough corners remain. The corners that are now lost, have a low response value. This means that they are weak corners and thus are not points of interest. The value for *k* at this point is *0.10*. For the intensity image there is an equal behaviour of *k*. But now there is a steeper decrease of corner points. Now the value of *k* is chosen just in front of this increase of edges, so that not too many corner points are lost. The value of *k* is *0.11* at this point.



a)

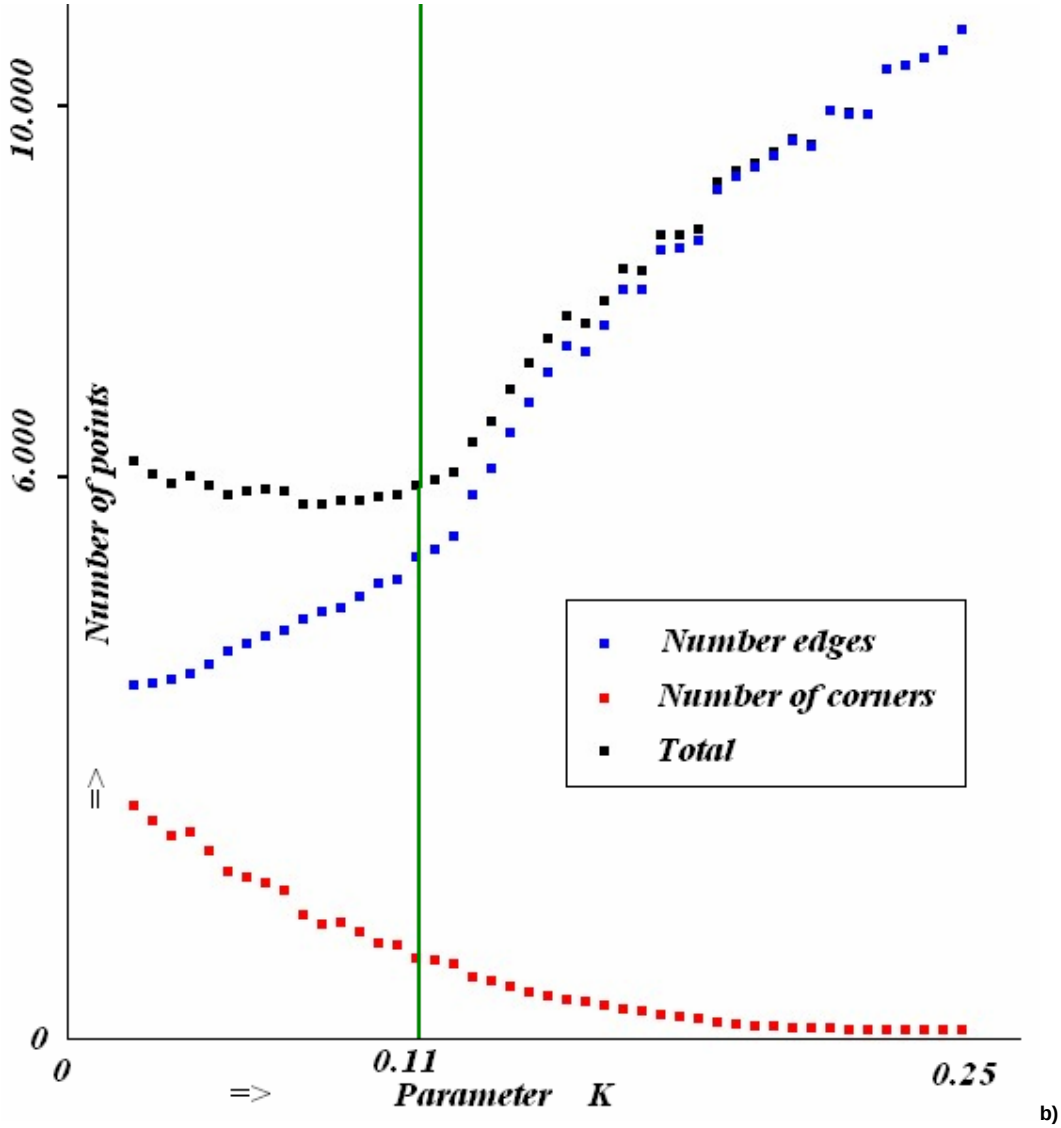


Figure 5.11: Parameter k plotted against the number of Harris responses for **a)** the colour image and **b)** the intensity image.

After the parameter *k* and the kernel sizes are determined the Harris function can be performed. The output of this *openCV* function is the response *R* for every pixel as defined in formula 3.13. The output values *R(x,y)* are scaled to the grey value domain { 0 , 255 } and plotted in an image (figure 5.12.a) and an intensity histogram (figure 5.12.b).

The feature points are filtered out with two thresholds; one for the edges and one for the corners. To automate the threshold selection the histogram is used. The high peak in the middle contains all 'flat region' pixels, which are the points in the image that are neither a corner neither a pixel. All the bins left of the peak contain the edge points and on the right side of the peak the corner points are found. At this point the thresholds are simply placed 2 bins left and right of the peak, because the neighbouring bins still contain a lot of pixels of the flat region. In figure 5.12.c the result is shown with the corners in red, the edges in blue and the flat region in black.

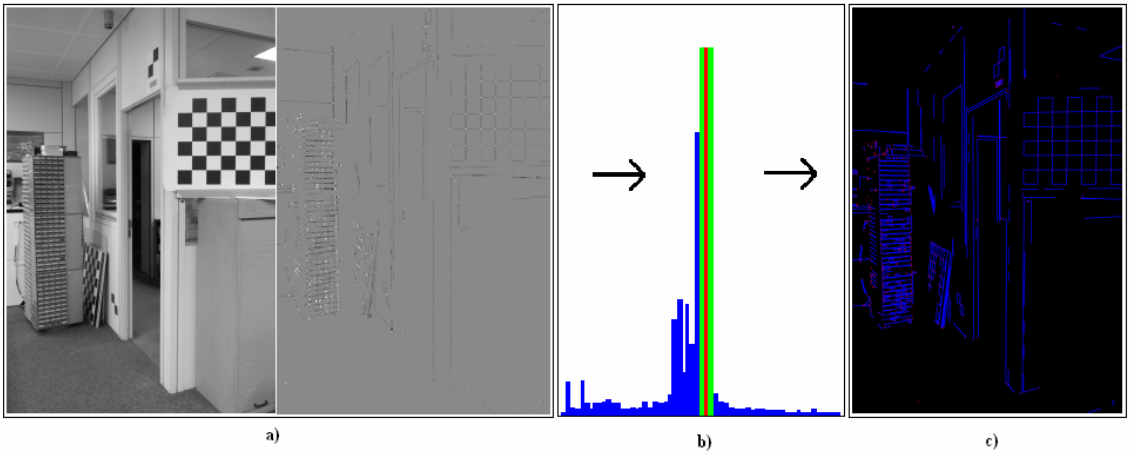


Figure 5.12: a) The original and the Harris output, **b)** histogram of the Harris output with thresholds, **c)** output after applying the thresholds

5.2.5 Hough transform

After the erosion of the little segments from the Canny output by the pruning algorithm, still a number of bad features remain, e.g. little curved lines from the background clutter. The objects of interest in the image should contain unique feature points such as corners or crossing edges. Because these objects of interest contain straight lines, they can be selected with the Hough transform for lines. All objects that are not clustered in lines are deleted. In this way the remaining background clutter and curved objects are removed from the image. This paragraph describes the implementation of the Hough transform for lines and its problems.

For the implementation of the Hough transform the openCV library contains the function *cvHoughlines2*. This function finds lines from the pixels in the input image. The input image for the Hough function is the Canny output (after pruning). Later on the Hough lines are combined with the Harris corners. The function is used in two modes:

- Classical Hough transform
- Probabilistic Hough transform

The classical Hough transform finds the lines in the Hough space (ρ, θ), like it is explained in paragraph 3.5. The probabilistic Hough transform was first introduced by *N. Kiryati , Y. Eldar and A.M. Bruckshtein* (see [8]). The fundamental difference of the probabilistic Hough transform with the classical Hough transform is that it randomly selects a number of pixels instead of looking at all of them (see [18] *Technion Institute of Technology*). In other words the main idea is that to be able to detect objects, it is sufficient to compute the Hough transform from only a proportion of the pixels in the image. The algorithm grasps a number of points and tests by means of the classical method if they represent a line segment. Besides being a faster algorithm the main advantage is that it is more efficient if the image contains a few long lines, because it returns line segments (pixel coordinates of the begin and end point) rather than whole lines (Hough coordinates (ρ, θ)).
The function cvHoughlines2 requires a set of parameters with some depending on the type of Hough transform:

- Distance resolution [pixel], determines the size of the accumulator cells (paragraph 3.5).

- Angle resolution [radians], determines the size of the accumulator cells.

- Threshold, a line is returned by the function if the corresponding accumulator value is greater than the threshold.

- The first method-dependent parameter:
  - For classical Hough transform it is not used (0).
  - For probabilistic Hough transform it is the minimum line length.

- The second method-dependent parameter:
  - For classical Hough transform it is not used (0).
  - For probabilistic Hough transform it is the maximum gap between line segments lying on the same line to treat them as the single line segment (i.e. to join them).

All parameters are dependent on the image resolution. They can standardize in case of a constant data acquisition. When this algorithm is used on a different acquisition, the parameters must be readjusted.
The output of Hough is a set of lines. They are plotted in figure 5.13. The red lines are derived from the standard and the green lines from the probabilistic Hough transform. The probabilistic method returns the lines by the pixel coordinates of two end points, so that they are easily plotted in the image. The standard method returns the Hough parameters (ρ, θ). In the image these parameters draw a line through the complete image without a beginning or an end.
The problem exists now that the line object is also containing pixels from objects or background clutter in other parts of the image. By searching for standard Hough lines in smaller areas this problem is shrinking. So the image is split up in rectangular sub images and after that the Hough transform is performed on the sub-images separately. Finally all sub-images are placed together again and the resulting set of lines is ready to be combined with the feature point candidates.

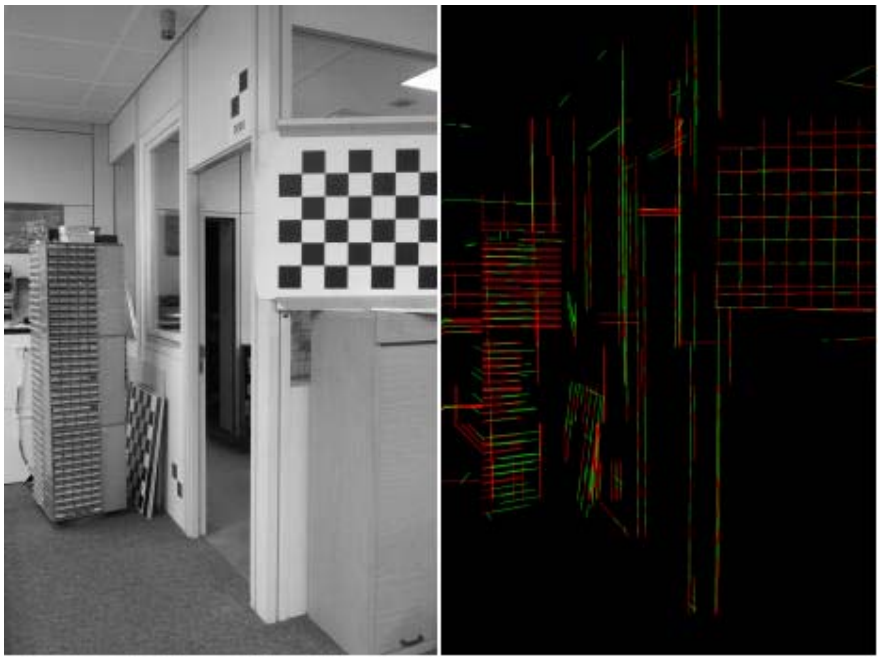


Figure 5.13: Result of the Hough transform with the classical (red) and the probabilistic (green) lines.

5.2.6 Selection of the Key points

The final stage is the actual selection of the key points. This selection is done by making use of pruned Canny, Harris corners and Hough lines. The main idea is to overlay these three data sets and select their intersection. All pixels that the intersection contains are selected:

$$Canny \cap Harris \cap Hough = key\ point\ dataset\ (u,v) \tag{5.4}$$

Figure 5.14: Result of the intersection of Canny, Harris and Hough.

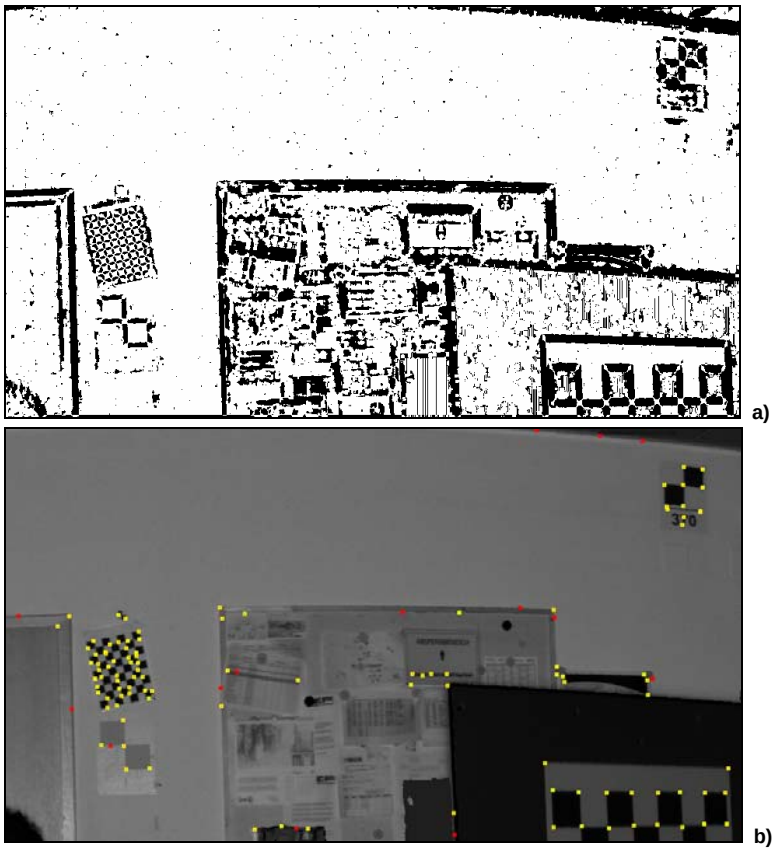
In figure 5.14 the red pixels show the result of the intersection plotted on the Canny edges (figure 5.8 with a kernel size of 3). Still two problems have to be solved in order to get the right dataset of key points. The first problem is that the points are still groups of pixels. As seen in the plot the intersection returns little lines segments on the corners. So from every segment the best pixel must be selected to represent the corner point. The second problem is that there are still some points left on the edges of the objects. The position of these points depends on where the thresholds in Canny have cut of the edges, so this position will never be the same for the laser and colour image. This can result in a false point pair during the matching of these points.

The first problem is solved by separating the different groups of pixels and search for the pixel with the strongest response from the Harris corner detector. The subprogram for this called *getMax*, scans through the image until it finds a red pixel. Then it draws a window around this pixel to isolate the pixel group from the image. The window size is multiplied by the resolution factor used before by the Harris detection. For the intensity image it is 20 x 20 pixels and for the colour image 40 x 40 pixels. From the Harris result the corner responses of the pixels in the window are selected and then the maximum is picked out. The coordinates of this pixel become the coordinate of a key point.

The second problem is solved by eliminating the edge responses from the resulting dataset of key points in the same way as SIFT (see [9] *Lowe*). The edge elimination is constructed from the Eigen values of the same Hessian matrix that Harris uses (see formulas 3.12). The Eigen values can be retrieved by the openCV function *cvCornerEigenValsAndVecs*. For this function the same kernel size and Sobel size as in Harris were used. After that it finds the Eigen vectors and Eigen values and stores them into a destination image. With the Eigen values and ratio r a new image, called the edge eliminator is constructed. The following threshold (derived in paragraph 3.7, formula 3.20) is performed:

$$\frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} < \frac{(r+1)^2}{r} \tag{5.5}$$

If the ratio between the principle curvatures r is greater than 10 the pixel is black and if not, then it is white. This binary image can be laid over the dataset of key points and eliminate all the point in the black areas as shown in figure 5.15.a.



a)

b)

Figure 5.15: a) The edge eliminating image and **b)** the result key point selection including the eliminated (red) edge points.

After the edge elimination the data set of key points is finished. Figure 5.15.b shows an example of the results. The yellow dots are plotted around the key points to show the final result. The red dots show the points that are removed from the data set by the edge eliminator image.

5.3 Process of point pair selection

The previous paragraph contains the complete process of finding good features and selecting a dataset of key points, which are suitable for connecting the colour image from the camera and the intensity image from the laser scanner. This connection is found by matching both datasets of key points. The method design proposes to match the points by finding the nearest neighbour. The nearest neighbours are distinguished by their SIFT descriptor. Section 5.3.1 describes how this descriptor is produced for every key point. In section 5.3.2 the descriptor is used as a constraint for the matching algorithm. Besides SIFT it is explained how a second constraint on the pixel location is added and how three modes are implemented to get a more stable dataset of point pairs.

5.3.1 Construction of the SIFT Descriptor

After finding all the key points the next step is to give every key point a descriptor, so that it will be distinguishable from the rest. In the literature study the SIFT descriptor is found as one of the best reliable descriptor. This section shows how it was implemented in the program. The implementation contains the following 5 steps:

- Derivation of the gradients
- Gaussian mask
- Computation of the magnitude and orientation
- Descriptor construction
- Normalization

As seen in paragraph 3.7 the basic quantities for the descriptor are the gradient magnitudes and orientation of the key points (the yellow squares in figure 5.15.b). Both the magnitudes and the orientations can be approximated from the x- and y-gradient of a pixel by formulas 3.22.

So the first step is to derive the x- and y-gradients of every pixel in the image. Like before the Sobel operator is used, because it is known to be better than the Roberts operator with respect to noise and better than the Prewitt operator with respect to isotropy, and because Sobel is easily used from the openCV library. The function *cvSobel* makes the convolution of the Sobel kernel with the image and puts in an output image. This output image is given 16 channels instead of 8 to prevent the loss of pixels that will give a higher response then 255, which is possible and a well known problem with the Sobel operator.
The second step is a preparation for the computation of the magnitude of the derivatives. A Gaussian mask will give a weight to all the pixels used in the descriptor, so that closer pixels will have a bigger influence than pixels on the edge of the window around the key point. The size of the mask is equal to the size of the descriptor window. The weights are computed by formula 3.1. In this discrete application it is implemented by the function

$$G_{x,y} = w\frac{1}{2\pi\sigma^2}e^{-((x-w/2)^2+(y-w/2)^2)/2\sigma^2}, \tag{5.6}$$

with

x, y	*the window pixel coordinates,*
σ	*the standard deviation and*
w	*the window size.*

The standard deviation is automatically set depending on the descriptor size by the empirical rule, which is that 68,2 % of the pixels are within 1 sigma away from the centre. This means that σ is computed by $\sigma = w \cdot 0.341$. Figure 5.16 shows the results of step 2.

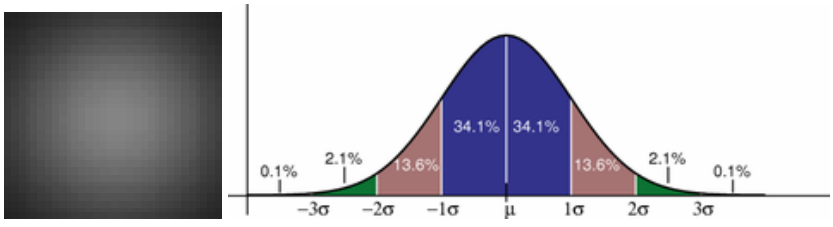


Figure 5.16: a) The Gaussian mask and **b)** the Gaussian curve (source: [21] Wikipedia encyclopaedia)

The third step is the computation of the magnitude and orientation. First the program scans the image until a key point is found. Then the window with size *w* is defined around that point. In this loop for every pixel the magnitude and orientation is computed by making use of the approximations in formulas 3.14 and 3.15. At the same time the magnitude is multiplied by the corresponding value from the Gaussian window. The results are stored in a magnitude vector and an orientation vector.
The fourth step is the construction of the descriptor. Paragraph 3.7 describes that the window is divided into 4 or even 16 quadrants. The pixels are placed in one of the 8 main directions by their orientation and the magnitudes are summed up to produce the length of every main direction. The program selects a pixel, searches the closest main direction and adds its magnitude to it. The result is stored in a large vector. Figure 5.17 shows the order in which the vectors of the SIFT descriptor are stored in the large vector. From left to right and downwards for every part the directions are stored clockwise.

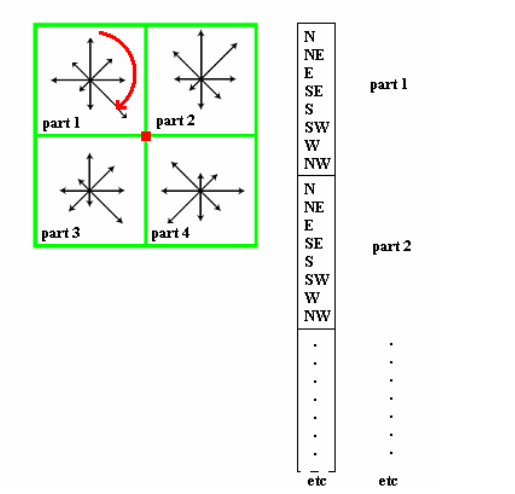


Figure 5.17: The order of storage of the SIFT descriptor vectors.

The fifth and the last step is the normalization of the SIFT descriptor. The normalization is repeated for every point, so that the descriptors are independent of the gradient magnitude range around the point. In this way they are more suitable for matching.

The normalization is performed by first computing the mean and the standard deviation of the vector and then applying formula 5.7:

$$V' = \frac{V - \mu}{\sigma}, \tag{5.7}$$

with
V' *the normalized value,*
V *the original value,*
μ *the mean and σ the standard deviation.*

This way the mean of the vector is set to 0 and the standard deviation to 1. To transform the vector to the range of [0;1] formula 5.8 is applied:

$$V'' = \frac{(V' - \min)}{(\max - \min)}, \tag{5.8}$$

with
V'' *the transformed value,*
V' *the normalized value,*
min *the minimum normalized value and*
max *the maximum normalized value.*

After the normalization the descriptors are finished and can function as a constraint for the matching of point pairs. But when in Matlab a little program was written to visualize the descriptors, the construction looked very unstable. This is caused by the placement of the key point in the descriptor window, which is right at the cross of the four descriptor parts. So with a little inaccuracy of the key point location, the vectors in the descriptor start wandering around the cross. To overcome this problem a buffer is introduced. Every part of the descriptor grips an extra buffer of the other parts as shown in figure 5.18. Now that the key point is in every quadrant, it can not move anymore.
Lowe proposed in his article not to use a 2x2 descriptor but a 4x4. According to his testing the descriptors are performing best, when implemented as a 4x4 array of histograms with 8 orientations. The better performance is confirmed during the implementation as the 4x4 array returned significantly more matches of point pairs. A few examples of resulting descriptors are shown in figure 5.19. The first example shows a corner point of a target. The dominant directions of the gradients are up and left. The second example shows the result for a vaguer corner; the vectors are more divided. The last example is the descriptor for a cross point, which shows a combination of two corner points. Note that it is a visualization in Matlab and that in the computer the descriptors are feature vectors with 128 dimensions (4x4x8).

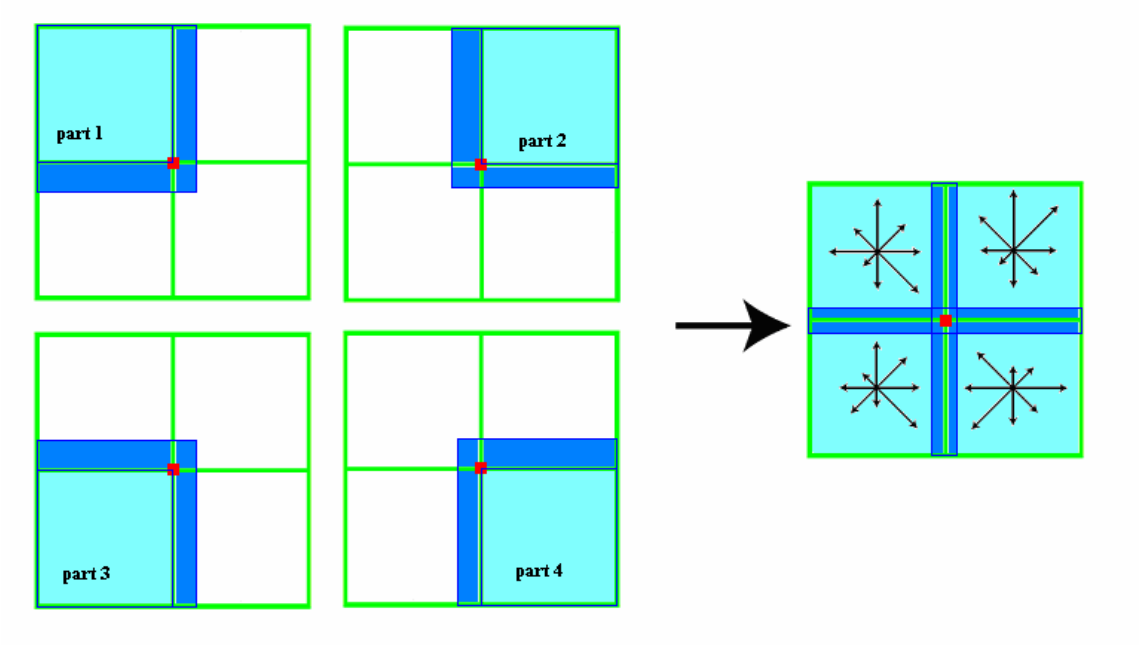


Figure 5.18: The SIFT descriptor includes a buffer zone.

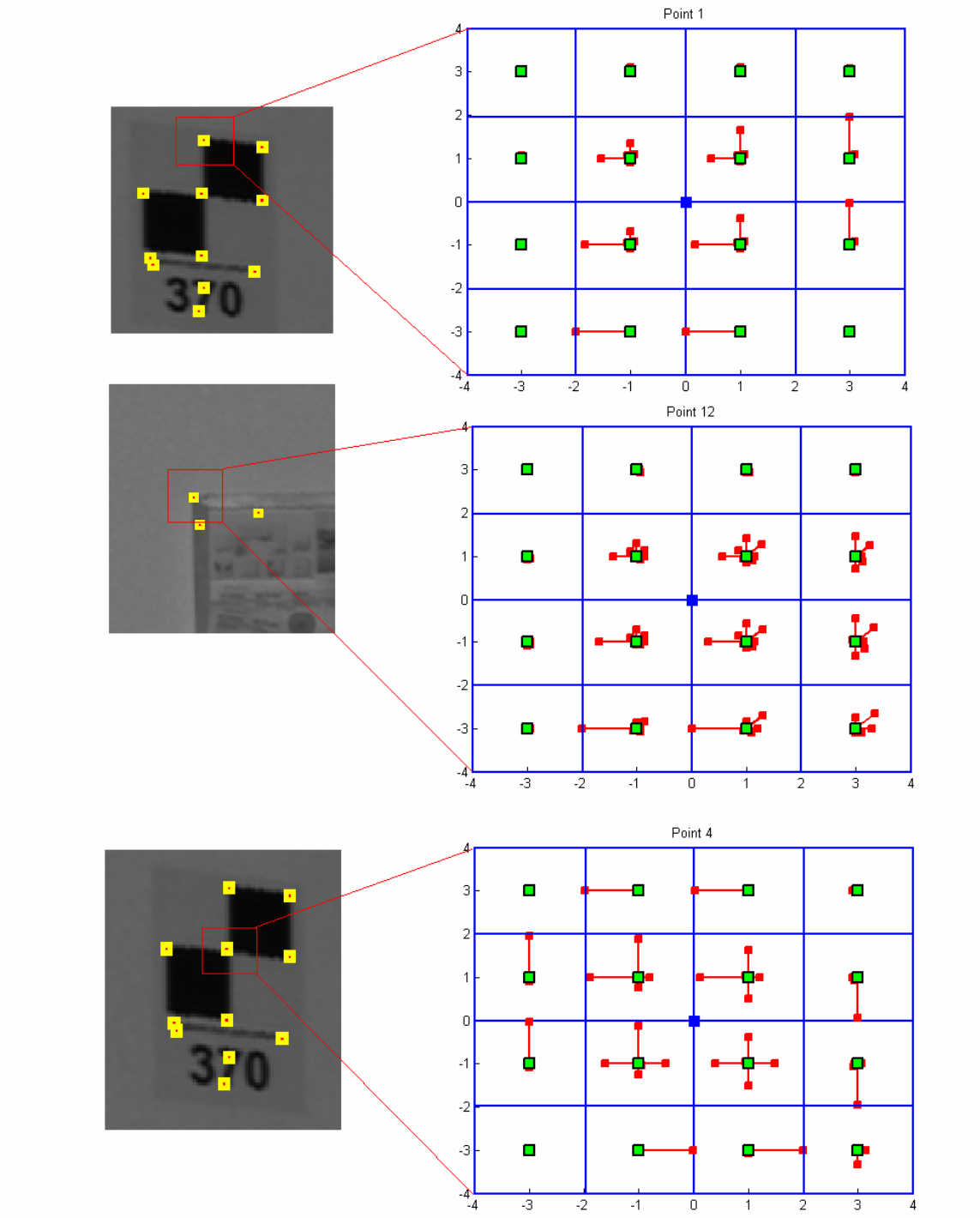


Figure 5.19: Three examples of the constructed SIFT descriptor.

During the implementation of the descriptor two alternatives are found. Together with the descriptor described above, they are set into three modes:

1 Standard mode,
2 Steered mode,
3 Mirrored mode.

The idea of the steered mode is to make the descriptor invariant to orientation differences. In this mode for every key point the mean gradient direction in the descriptor window is computed. Then before all the pixels are put into one of the 8 bins (of orientation), first this mean is added to their orientation value. In this way the descriptor is steered towards the mean direction of the key point. The expectation of this mode is to find a few more matches. But because the baseline between the laser scanner and the camera is very small and thus the orientation difference is minimal, it is expected that the results look similar to the standard mode. Still for points on the chessboard for example the steering can be a property that separates them from their neighbours that look very similar.
The idea of the mirrored mode is the ability to find matches between laser and camera key points that have a mirrored intensity gradient. The infra red light from the laser differs from the visual light spotted with the camera. A regular consequence of this difference is that an object in the intensity image might be darker than its surroundings, while it is brighter in the colour image. For example a white wall that makes a small angle with the light rays gives a bright intensity to the camera, but a dark intensity value to the laser scanner (see figure 5.20). To overcome this problem the mirrored mode mirrors the descriptors of the intensity image key points simply by multiplying with -1.

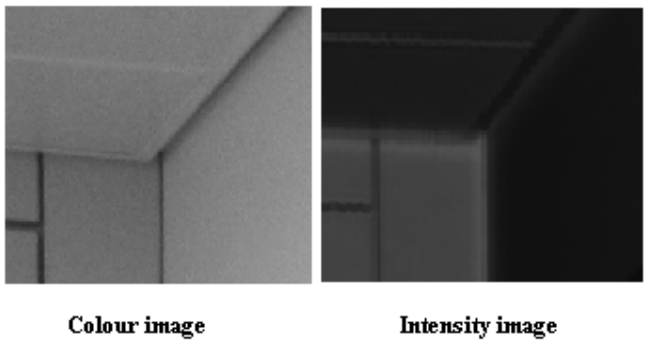


Figure 5.20: The problem of mirrored gradients.

The program stores the three descriptors of all key points together with their pixel coordinates in the class called *feature*.

5.3.2 Matching

The program is now ready to bring the intensity image and the colour image together. The goal of the matching algorithm is to find as much correct point pairs as possible without bringing to many mismatches that will disturb the optimization. To achieve this two constraints have been implemented; one constraint on the pixel location and one constraint on the descriptor content.
In first instance it is tried to match the point pairs only on the descriptor constraint. It turned out that this was practically impossible, because the image types are so different. The intensity image contains many irregularities with respect to the colour image. This is caused by the different type of light source. Besides this disadvantage also the resolution difference between the images makes the matching less accurate. To solve these problems a second constraint is added; the matching is constraint on the location and is only performed in the neighbourhood of the key point. This way the descriptor does not have to compete with all descriptors in the image. The change that a correct match is found has increased and moreover the algorithm becomes much faster.
The descriptors are available for all key points now, but for the location constraint more data is needed. During the data acquisition angle α is measured, which is the angle between the orientations of two pictures with respect to the spherical coordinate system of the laser scan. From this knowledge the program can compute the location in the intensity image, where the colour key point can find its correct match. Because this location is an approximation a window is constructed around it to overcome the inaccuracy. Within this window all key points are tested by computing the Euclidian distance between their descriptor and the one from the colour key point. The key point from the colour image is matched with its nearest neighbour in the intensity image. This procedure is repeated three times containing the three descriptor modes that are constructed in the previous step.

5.4 Non-linear optimization

Until now all effort has been put into finding reliable distinctive features from both the digital images and the laser scan. When all the methods have succeeded into a dataset of matched key points, the next step is to use these point pairs to place the 2D digital images into the 3D laser scan space. To accomplish this, the non-linear rigid optimization is used.

The program for the optimization is delivered by the Z+F library. The only issue for this research is to implement it after the matching of the point pairs. The optimization functions as it is explained in paragraph 4.2. Besides the point pairs (u_c, v_c and x_l, y_l, z_l) the input consists of the camera calibration parameters (internal parameters) and the approximate values for the external parameters (translation and rotation), which are known from the data acquisition. With the optimization the external parameters (rigid orientation and relative position) are optimized. In other words the optimization derives the geometric constraints on the rigid transformation from the camera coordinate system to the laser coordinate system. When there are enough correct matches found, the optimization program performs an iteration to find the minimum error (equation 4.2).

6 Result presentation

In the previous chapter the complete process for implementing the automatic registration in C/C++ is carried out. All modules are explained and their problems are treated. This chapter shows the results accomplished after the implementation. To visualize the results in a proper way one part of the images is cut out and the results for this part are shown in the figures.
From the beginning until the end of the algorithm numerous parameters have been used; thresholds, scale factors, resolutions and other factors influence the results during the process. This chapter shows the results of different moments in the process and explains the chosen values for the parameters. The main issue for this research is the possibility to automatically find representative values for each of them. Paragraph 6.1 contains the results of the key point selection in both the intensity image and the colour image. In paragraph 6.2 the results of the point pair selection are shown. For both processes different intermediate results are stepwise shown and the behaviour and values of different parameters are carried out.

6.1 Results of key point selection

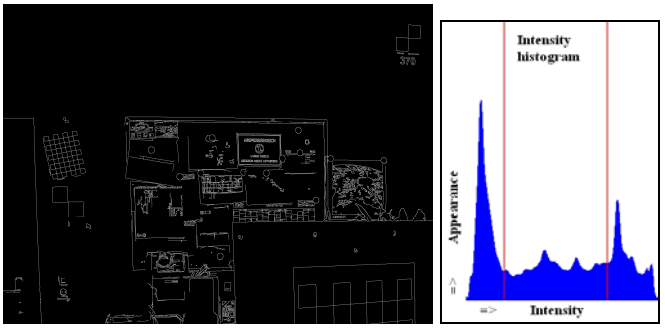
The selection of key points starts with the improvement of the images in order to make them usable in the further processing. As seen in section 5.2.1 especially the intensity image needs attention, because it does not use the full range of intensity bins [0 - 255]. This problem is solved by spreading the histogram of the image. There are no parameters involved in this process. The result is shown in figure 6.1 with a) the colour image and b) the intensity image.
Especially the intensity image has improved a lot. The image now uses the complete range of intensity bins. The result is that the image is much brighter and that the contrast has improved significantly. This has a positive effect on the further processing of the image. Still the resolution of the intensity is low, which causes the edges to be less sharp than in the colour image.
The colour image (loaded as greyscale intensity values) has not changed. This is not a problem because of the quality of the image. It uses the full range of intensities, the contrast between the objects is high and the high resolution gives sharp edges. The image is ready for further processing.

a)

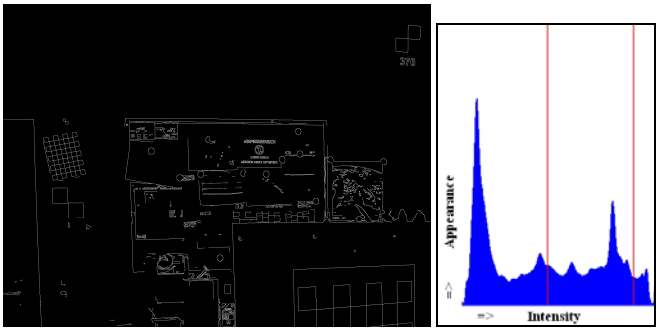
b)

Figure 6.1: a) The colour image and **b)** the intensity image after image improvement.

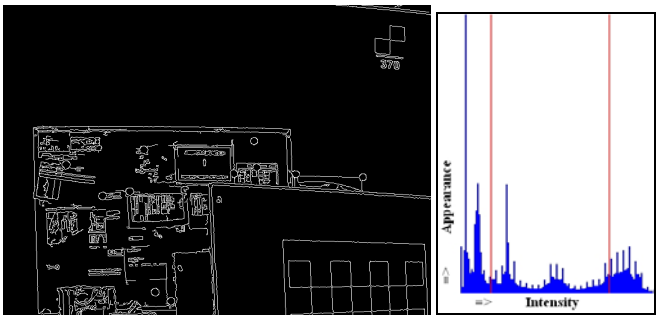
The next step in the algorithm is the Canny edge detection. This detection algorithm has two thresholds and a variable kernel size for the Sobel operator, which determines the first derivative of the image intensity. Section 5.2.2 proves that the kernel size of 3 gives the best results in this research. The thresholds are changed into percentages during the implementation and from these percentages the nearby minimum in the intensity histogram is found. In order to get the best values different pairs of percentages are tried.



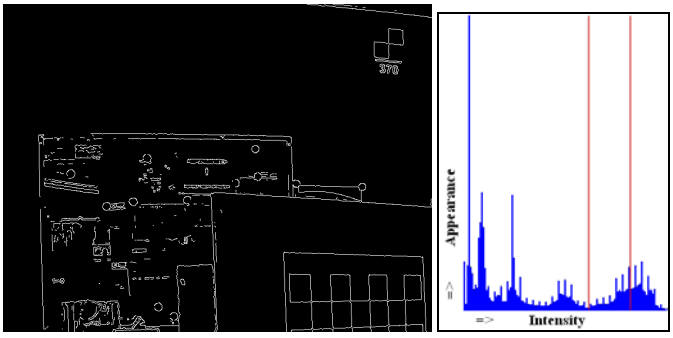
a) 30 % and 80 % (colour)



b) 50 % and 90 % (colour)



c) 40 % and 80 % (laser intensity)

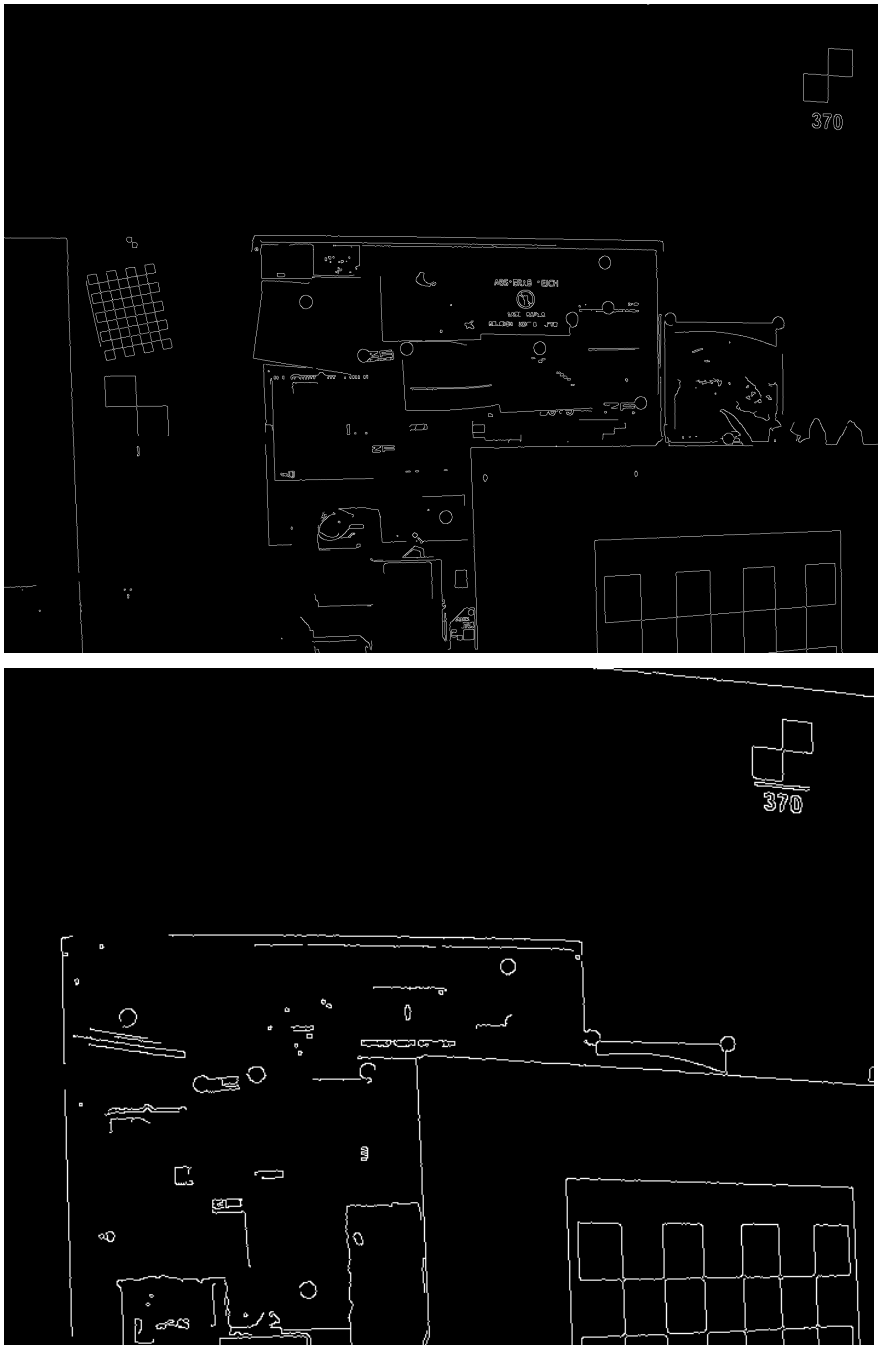


d) 70 % and 85 % (laser intensity)

Figure 6.2: Example of the Canny result for **a)** and **b)** the colour image and **c)** and **d)** the intensity image.

Through a series of experiments for both image types two satisfying thresholds are found. For the colour image the optimal percentages are 50 % for the lower threshold and 90 % for the upper threshold. For the intensity image the lower threshold is 70 % and the upper threshold is 85 %. In figure 6.2 two examples of the result with these thresholds are shown for both images.

The edges are successfully detected by Canny in both image types. The unwanted irregular objects have been removed as much as possible by adjusting the ideal thresholds. In figure 6.2 it can be seen that figures b) and d) contain less noise than figures a) and c). The figures b) and d) also show that with these thresholds a redundant amount of candidate key point locations on the objects of interest are detected; the chessboard for example is perfectly detected. The result on the colour image contains sharper edge, because its resolution is higher. The edges in the intensity image contain more irregularities.

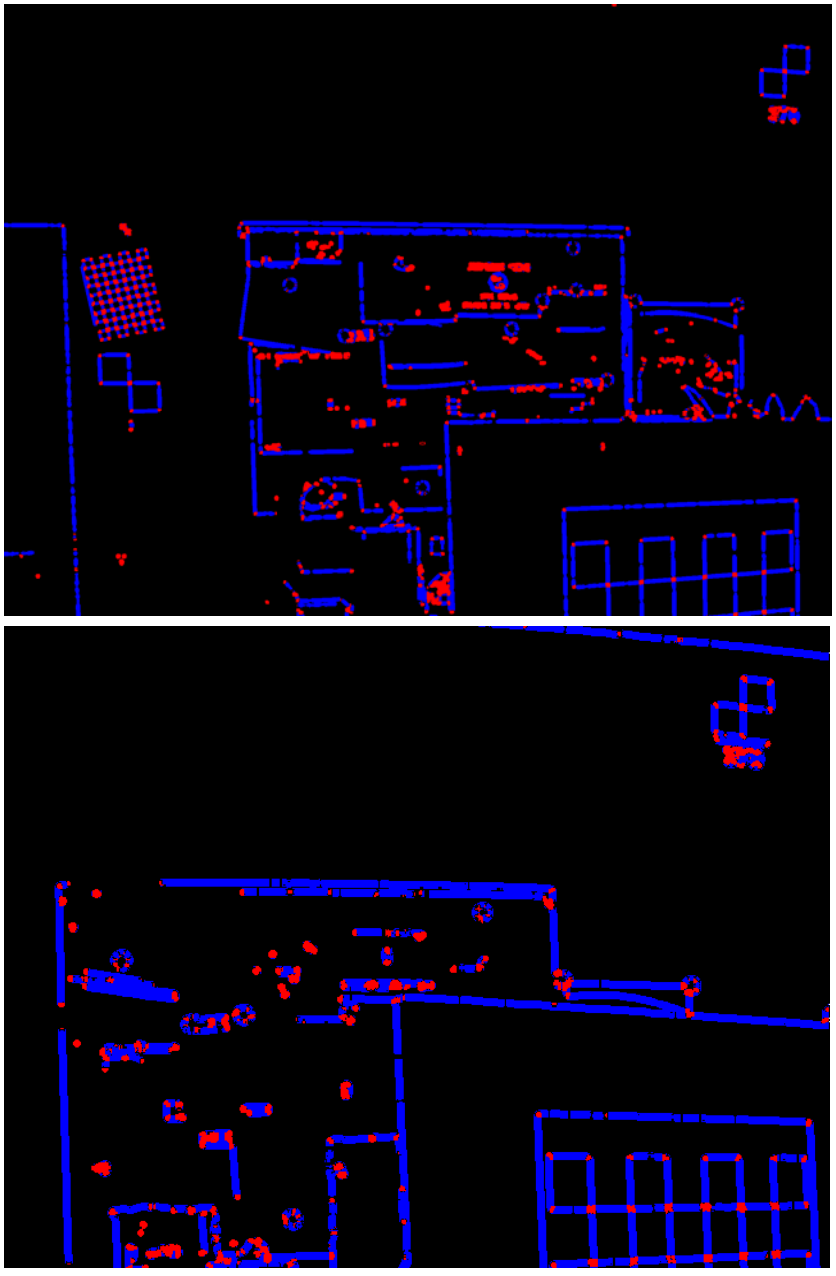


a)

b)

Figure 6.3: The pruning result using 40 iteration steps on **a)** the colour image and **b)** the intensity image.

After the Canny edge detection this result is improved with the pruning algorithm, because a lot of little edge components and noisy edges remained. The only variable parameter here is the number iterations (see section 5.2.3). If this number is too small, too many noisy objects remain in the image. If it is too large, then too much useful information will be deleted and lost. Again the best value for this parameter is found by a series of experiments. Figure 6.3 gives the results using 40 iterations in the pruning algorithm.
Compared to the results of the Canny edge detection now a lot of little edge segments have been removed. The set of remaining edges do not contain a lot of noisy and small edges. Note that the objects of interest that contain the candidate key points remain.



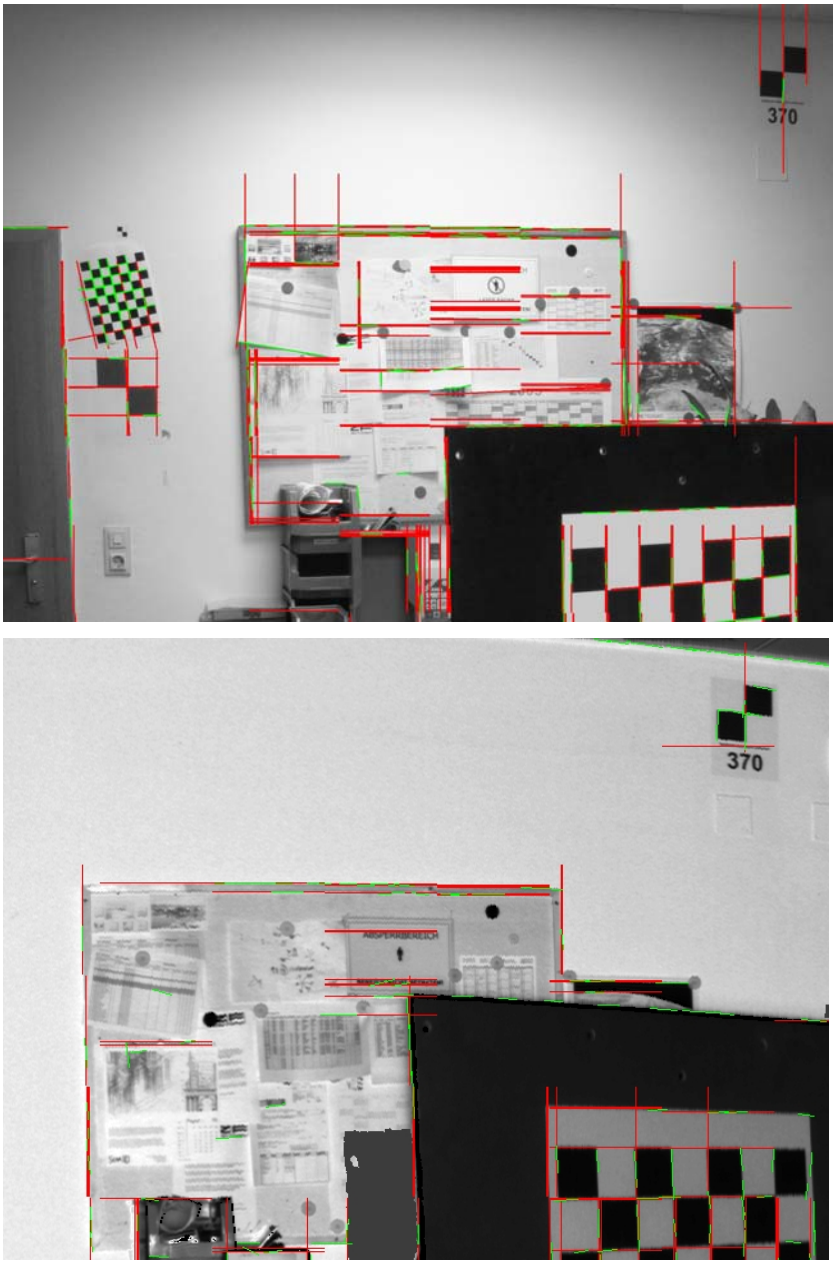
a)

b)

Figure 6.4: The Harris result using the optimized parameters on **a)** the colour image and **b)** the intensity image.

The Harris edge and corner detection is performed as explained in section 5.2.4. Three parameters are involved: the kernel size (size of window for determination of the Hessian matrix), the Sobel kernel size (first derivative) and parameter k. Figure 6.4 shows the result of the corner (red) and edge (blue) detection. The choice for the parameters is based on section 5.2.4.
The main purpose of the Harris function in this research is to detect the corners which contain the candidate points of interest. The result provides many corners on these candidate locations. There are no features lost due to the Harris operation. All the objects that were found with Canny and pruning are still in the image, and from this point the algorithm also knows the location of their corners.

In order to make the set of objects found by the algorithm more useful for matching, the Hough transform is applied to cluster the lines in the images. The Hough algorithm includes five parameters (see section 5.2.5), but the advantage here is that they are all dependent on the resolution of the image. The resolution is assumed to be constant, because of the standard data acquisition procedure of the IMAGER 5003 (see paragraph 5.1 and [1] *Abmayr*). Still if the parameters are not well adjusted to each other the results are not satisfying. By a series of experiments a good configuration is found for the test data. If in any application after this research the image resolution changes because other instruments are used, this configuration is not justified and needs to be adjusted again. For the data in this research the configuration gives a good result on the colour and the intensity image. Both examples are shown in figure 6.5. The combination of the standard (red) Hough lines and the probabilistic (green) Hough lines makes the set of lines complete. Both methods were implemented, because separately they do not provide enough lines. Because of the lower resolution the orientation of lines in the intensity image is less accurate than in the colour image. Still this result is able to fulfil the purpose of the Hough transform in this algorithm: to isolate the objects of interest.



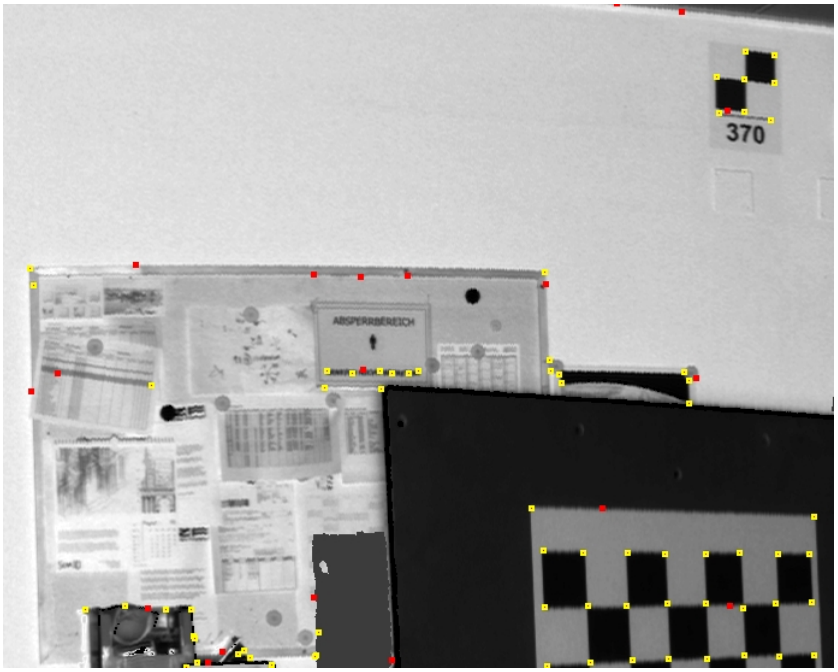
a)

b)

Figure 6.5: Examples of the Hough results for **a)** the colour image and **b)** the intensity image

The final stage of the key point selection is the intersection between the Canny, Harris and Hough result (equation 5.4). Additionally the edge elimination is performed, which deletes all key points that are actually edge points (section 5.2.6). The result is the final dataset of key points. In figure 6.6 some examples of this end result are plotted. The yellow squares are the selected key points and the red squares represent key points deleted by the edge elimination image. Many good key points are found, but especially in the intensity image some unwanted key points remain. The unwanted key points will mostly delay the algorithm and not disturb it, as they will not find a match with their descriptor.

a)



b)

Figure 6.6: Examples of the selected key points for **a)** the colour image and **b)** the intensity image

6.2 Results of point pair selection

The point pairs are found in two steps: the construction of SIFT descriptors and the matching. After the construction of the descriptors (section 5.3.1) the key points are ready to be matched with each other. This matching (see section 5.3.2) is performed on two constraints: the approximate locations of the key point and the correspondence of their descriptors.
The result of the matching process is a dataset of point pair combinations between the colour image and the intensity image. In figure 6.7 an example of this result is visualized onto the colour image. The laser points are transformed to image coordinates and then the point pairs are represented by the blue squares connected with a red line in the colour image.

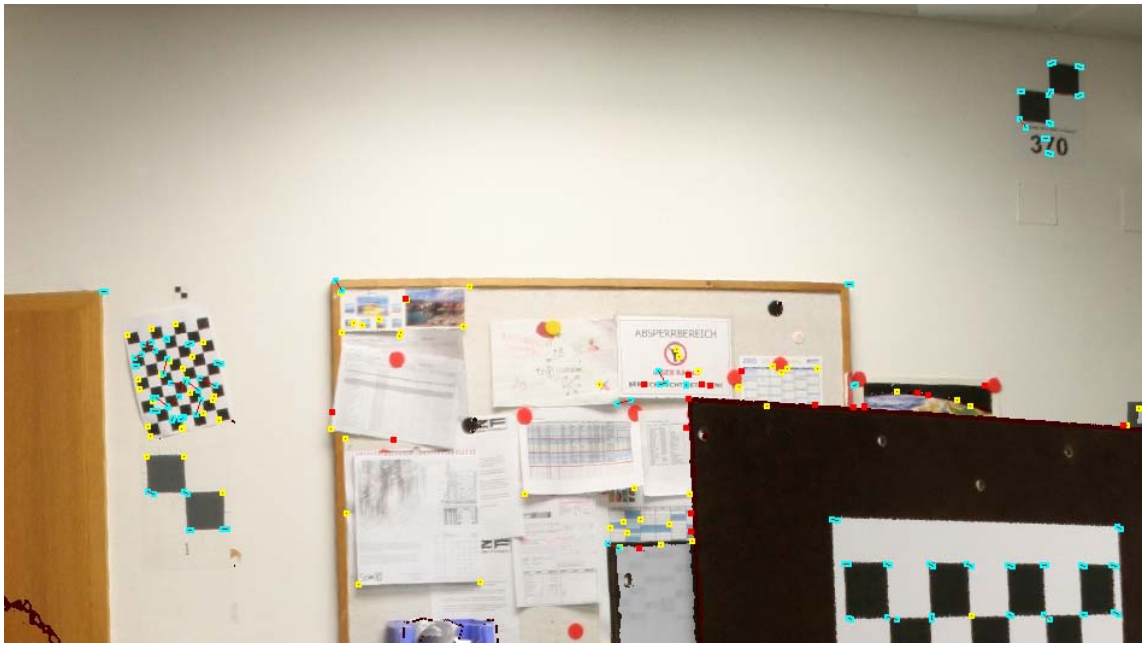


Figure 6.7: Example of the selected point pairs after matching.

Now that the point pairs are selected the input for the non-linear rigid optimization is complete. The calibration of the camera and the laser scan is known from the a priori camera calibration procedure and the a priori laser scanner calibration. The approximate values for the external parameters are retrieved from the data acquisition.
The results of the optimization are tested with the dataset that was cut out from the full laser scan. For this test an artificial colour image is constructed from a semi-automatic registration; in the semi-automatic procedure the point pairs are pointed out manually, followed by the optimization procedure. Therefore the artificial colour image has the same orientation and position as the laser scan itself.
The optimization from the Z+F software is implemented to determine the rotations angles (*pitch, roll* and *yaw*) and the translations (*Tx, Ty* and *Tz*). In the test case when the artificial colour image is used, the optimization finds all external parameters to be zero. This is expected because there is no base line and no rotation between the images. For the full scan the program succeeded to determine the external parameters for all images.
After the optimization has succeeded, the external parameters can be used to plot the RGB values of the colour image onto the 3D point cloud of the laser scan (see [4] *Fröhlich*). This final stage is not executed in this research; unfortunately the creation of the 3D colour image did not fit into the graduation time. Therefore the 3D colour image can not be shown in this report. The creation of the 3D colour image already exists in the semi-automatic approach of Z+F. The difference between the automatic approach in this research and the semi-automatic approach is that the point pairs are selected automatically. Therefore figure 6.7 can be seen as the end result.

During the implementation of the optimization the idea came up to use the Random Sample Consensus algorithm (RANSAC) in order to improve the stability. The RANSAC algorithm is an algorithm for robust fitting of models in the presence of many data outliers. This algorithm is able to handle the mismatches in the point pair data set. It iteratively tries different sets of point pairs and stochastically arguments a good model for the non-linear rigid optimization.

7. Conclusions and recommendations

This chapter carries out the conclusions and recommendations that are found during this research. Paragraph 7.1 contains the conclusions. The recommendations are written in paragraph 7.2.

7.1 Conclusions

The automatic registration of laser scanning data and colour images designed in this research is successfully implemented. In this chapter of conclusions the designed method is evaluated according to the results that are retrieved in the chronological order. The conclusions follow from the implementations (chapter 5) and the results (chapter 6). The algorithm contains different parameters. They are listed in table A3 (appendix A).

The histogram equalization introduces noise and decreases the contrasts in the intensity image of the laser scan. Therefore the histogram spreading (stretching only) instead of equalization is used and improves the intensity image significantly in order to make it ready for further processing.

The Canny edge detector is successful in finding the edges in both the intensity image and the colour image. Because of the lower resolution the edges in the intensity image have more irregularities. Besides finding all the edges, Canny always leaves noisy objects and little edge segments.
Canny contains three parameters. The Sobel kernel size is dependent on the image resolution and does not change for the data in this research. The upper and lower threshold both depend on the content of the image histogram. If the scene changes the ideal parameters change. By selecting the threshold on pixel percentages and move them to the local minimum in the histogram, this effect is minimized.

The pruning algorithm is effective in removing noisy objects and edge segments. The objects of interest are untouched by the algorithm that removes a significant number of line segments. The number of iterations used is dependent on the image resolution which is constant.

The Harris edge and corner detector is able to find corners and edges on both image types. In this research only the corner information is used, because the edges are already defined sharply by the Canny edge detector.

Parameter *k* is dependent on the content of the image. The ideal *k* might change if the scene changes (see recommendations). The kernel size and the Sobel kernel size are dependent on the image resolution.

The Hough transform is successfully used to select objects of interest from the images. Together with Harris and Canny the lines form a base for selecting the key points from the objects of interest only.
The parameters minimum line length, the maximum gap between the lines and the size of the accumulator cells are all resolution dependent.

The designed method succeeds in selecting key points from the colour image and the intensity image automatically. Many useful key points are found, but always a number of unusable object points remain.

It is possible to construct the SIFT descriptor vectors for all key points in the colour image and the intensity image. The SIFT descriptor alone is not able to find a proper set of point pairs. When the SIFT descriptor vectors of the colour image are compared (Euclidian distance) with the descriptor vectors at the approximate identical location in the intensity image a good set of point pairs can be found. In order to get the approximate location this method needs approximate values for the rotation of the camera in the laser scan system.

The set of point pairs functions properly as the input for the non-linear rigid optimization. The optimization demands the approximate values for the external parameters of the camera. The external parameters are optimized and can be used to put the image colours onto the 3 dimensional point cloud of the laser scanner.

7.2 Recommendations

For an optimal result of the automatic registration, the base line between the camera and the laser scanner should be minimal. This will minimize the influence of occlusions where the camera cannot see the laser points. For the acquisition of the data the existing method of Z+F is recommended. The approximated values for the external parameters are to be retrieved in this stage by measuring the rotation angle alpha in order to make the location constraint possible during the selection of the point pairs.

For the construction of the software tool for automatic registration the method described in this research is recommended. For stability of the optimization RANSAC is expected to have a positive influence, because it can handle large amounts of outliers. Therefore research on the implementation of RANSAC into the method of this research is advisable.

The input of the optimization contains point pairs from the colour image and the intensity image. Because the colour images overlap each other the optimization should be improved by adding point pairs from two colour images.

This research is performed on one scene only. In order to make the automatic registration useful in practice the behaviour of the parameters has to be investigated in different scenes. Many parameters depend on the image resolution; with a constant acquisition procedure of data they do not change. Therefore when this method is implemented as a prototype software tool, they can be programmed to change with respect to the input of the image resolution.

The thresholds of the Canny edge detector and parameter k of the Harris corner detector depend on the image scene. Further research on their behaviour is needed before implementing this method for different scenes.

This research has implemented the SIFT descriptor for the purpose of matching. Because it does not function as a stand alone, research to other types of descriptors is recommended to possibly improve the matching results. Paragraph 3.6 shortly describes different descriptor types that can be used.

Literature

[1] Abmayr T., Härtl F., Breitner M., Ehm M. and Fröhlich C. *Multimodale Sensorfusion auf Basis des Imager 5003*, Zoller + Fröhlich GmbH, Wangen im Allgäu, Germany, 2005.

[2] Bendels G.H., et al. *Image-Based Registration of 3D-Range Data Using Feature Surface Elements*, Institute for Computer Science II – Computer Graphics, University of Bonn, Germany, 2004.

[3] Brown M. and Lowe D.G. *Invariant features from interest point groups*, British Machine Vision Conference, Cardiff, Wales, pp. 656-665.

[4] Fröhlich C., et al. *Standardization and visualization of 2.5D scanning data and color information by inverse mapping*, Zoller + Fröhlich GmbH, Wangen im Allgäu, Germany, 2005.

[5] Fröhlich C., et al. *Vermessung und Modellierung von 3D-Umgebungen*, Zoller + Fröhlich GmbH, Wangen im Allgäu, Germany.

[6] Gonzalez R.C. and Woods R.E. *Digital Image Processing*, Second edition, Pearson Education Inc., Singapore, 2002

[7] Harris C. and Stephens M., *A combined corner and edge detector*, The Plessey Research Roke Manor, United Kingdom, 1988.

[8] Kiryati N. , Eldar Y. and Bruckstein A.M. *A Probabilistic Hough Transform*, Pattern Recognition vol.24, pp. 303-316, 1991

[9] Lowe D.G. *Distinctive Image Features from Scale-Invariant Keypoints*, Computer Science Department, University of British Columbia, Vancouver, Canada, 2004.

[10] Mikolajczyk K. and Schmid C. *A performance evaluation of local descriptors*, Dept. of Engineering Science, University of Oxford, Oxford, United Kingdom, 2004.

[11] Mikolajczyk K. and Schmid C. *Scale & Affine Invariant Interest Point Detectors*, INRIA Rhône-Alpes GRAVIR-CNRS, Montbonnot, France, 2004.

[12] Rabbani T., *Automatic reconstruction of industrial installations*, Netherlands Geodetic Commission, Delft, 2006

[13] Ree J.M. van. *How to test a terrestrial laser scanner,* Graduation thesis, Faculteit Lucht- en Ruimtevaarttechniek, Technische Universiteit Delft, 2006.

[14] Sahl H. *Intensitätsbasierte Registrierung von Farbbildern zu 3D-Laserscannerdaten*, Graduation thesis, Department Geomatik, HafenCity Universität Hamburg, 2006

[15] Seo J. K., Sharp G.C. and Lee S.W. *Range Data Registration Using Photometric Features*, Dept. of Media Technology, Sogang University, Seoul, Korea, 2005.

Visited websites

[16] CodeSource.net, Histogram equalization.
www.codersource.net/csharp_histogram_equalization.aspx, latest visit on 17-07-2006.

[17] Generation5, Picture of cumulative function.
www.generation5.org/content/2004/images/eqlCumulative.png, latest visit on 17-08-2006.

[18] Technion Institute of Technology, Probabilistic Hough Transform.
http://www.cs.technion.ac.il/Labs/Isl/Project/Projects_done/VisionClasses/Vision_1998/Hough/node6.html, latest visit on 11-07-2006.

[19] University of Delaware, Digital imaging college notes.
www.udel.edu/Biology/Wags/b617/digital/digital.htm, latest visit on 16-08-2006.

[20] University of Stuttgart, 2.5 dimension concept.
www.ifp.uni-stuttgart.de/publications/phowo95/Carosio.pdf, latest visit on 16-08-2006.

[21] Wikipedia encyclopaedia, Normal distribution.
http://en.wikipedia.org/wiki/Main_Page, latest visit on 26-07-2006.

[22] Zoller + Fröhlich, Hardware en software specifications.
www.zf-laser.com, latest visit on 06-09-2006.

Used software

Matlab®, Version 7.1.0. R2006 Service Pack 3. Copyright 2006, The MathWorks, Inc.

Microsoft Visual Studio 2005, © 2006 Microsoft Corporation.

Appendix A: Summary of the program in C/C++

This appendix gives an overview of the program that is made during this research. The main structure is shown in table A1. Table A2 gives an overview of all the functions that are used. The complete functions can be found on the attached cd-rom. In table A3 a list of parameters that are used in the program is shown with their suggested values.

Table A1: Program structure

```
main
{
      Image processing on intensity image:
      1 key point selection
      2 construction of the SIFT descriptors

      Loading the range values

      Start loop for all colour images

            Image processing on color image
            1 key point selection
            2 construction of the SIFT descriptors

            Matching

            Optimization

      End for loop

      return external parameters;
}
End of main
```

Table A2: List of functions

Name	Function
init_data( )	Initializes image and parameters
findFeaturePoints( )	Finding key points and SIFT construction
histSpreading( )	Performs histogram spreading
canny( )	Performs canny edge detection
moveToMinimum( )	Moves threshold to local minimum
pruning( )	Performs pruning algorithm
harris( )	Performs Harris corner detection
eliminateEdge( )	Eliminates edge responses
hough_lines( )	Finds Hough lines
selectKeypoints( )	Performs intersection of Canny, Harris and Hough
getMax( )	Selects the maximum Harris response locally
cloneImage( )	Copies a complete image
Sift( )	Constructs the SIFT descriptors
matching( )	Performs the matching
optimize( )	Starts the optimization
nl_opt_nViews_Minpack( )	Searches the minimum error in optimization
autoMap_world2image_Steven( )	Maps the world to image coordinates
autoMap_image2world_Steven( )	Maps the image to world coordinates
calcExternOfFixedCamera( )	Calculates the external parameters of the camera
loadingRange( )	Loads the range values from the range image
findLaser( )	Performs the key point selection on the intensity image in parts

Table A3: List of parameters used in the program and their suggested values (changeable in function init_data)

Parameter	Definition	Colour Image	Intensity Image	Unit
res_factor	resolution factor	**2**	**1**	-
percentage1	lower threshold [greyscale] at percentage [%] of pixels	**50**	**70**	[%]
percentage2	upper threshold [greyscale] at percentage [%] of pixels	**90**	**85**	[%]
sobel	kernel size of the Sobel operator	**3**	**3**	-
kernel	shift window size in Harris	**9**	**5**	-
k	response parameter k used by Harris	**0.10**	**0.11**	-
ite	number of iterations in pruning	**40**	**40**	-
min_length	minimum line length from Hough (probabilistic)	**40**	**20**	[pixel]
max_gap	maximum gap in a line from Hough (probabilistic)	**10**	**3**	[pixel]
thr_hough_std	threshold for the probabilistic Hough lines	**30**	**25**	-
pixel_res	pixel resolution for Hough during search	**0.25**	**0.125**	[pixel]
angle_res	angle resolution for Hough during search	**pi/180**	**pi/360**	[radian]
thr_hough_prb	threshold for the classical Hough lines	**1**	**1**	-

Appendix B: List of symbols and formulas

This appendix shows a list of symbols (table B1) and a list of formulas (table B2) that were used in this report.

Table B1: List of symbols

A	*x direction correlation*
a_1	*scale factor*
a_2	*shift*
B	*y direction correlation*
Bl	*blue channel intensity*
C	*x-y direction correlation*
Canny	*output of Canny*
Det	*determinant of matrix*
DoG	*difference of Gaussian*
e	*constant for natural logarithm*
E	*intensity change*
f	*image value*
G	*Gaussian function*
Gr	*green channel intensity*
Grad	*gradient magnitude*
Gx	*gradient in x direction*
Gy	*gradient in y direction*
H	*histogram value*
H'	*normalized histogram value*
Harris	*output of Harris*
Hough	*output of Hough*
i	*array position*
I	*image intensity*
j	*array position*
k	*Response parameter of Harris*
K	*calibration matrix with the internal parameters*
L	*image value after convolution with Gauss*
LoG	*Laplacian of Gaussian*
m	*magnitude*
M	*Hessian matrix (Harris)*
max	*maximum*
min	*minimum*
n	*kernel size*
p,θ	*Hough space coordinates*
P	*transformation matrix from Cartesian to image coordinates*

Table B1: List of symbols

r	*constant in threshold (edge eliminator)*
R	*response*
Re	*red channel intensity*
R_{ort}	*orthonormal orientation matrix*
t	*vector with translations*
Tr	*trace of matrix*
(u,v)	*position in the mask*
V	*original value*
V'	*normalized value*
V''	*transformed normalized value*
w	*window patch*
(x,y)	*image pixel coordinates*
[X,Y,Z]	*3 dimensional Cartesian coordinates*
Y	*Grey value*
μ	*mean*
σ	*standard deviation (Gaussian scale)*
$\angle$	*partial derivative*
∇	*Laplace operator*
π	*mathematical constant pi*
θ	*gradient direction*
λ	*Eigen value*
Σ	*sum*

Table B2: List of formulas

Number	Equation	Application
(3.1)	$R = \sum_{u,v} w(u,v) f(x+u, y+v)$	Spatial filter
(3.2)	Normalized histogram (I) = Total Number of Pixels of Intensity I / Total Number of Pixels	Histogram equalization
(3.3)	$H'(i) = \int_0^i H(j) \cdot dj$	Histogram equalization
(3.4)	$H'(i) = \sum_{j=0}^{j=i} H(j)$	Histogram equalization
(3.5)	$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$	Laplace Second derivative
(3.6)	$\frac{\partial^2 f}{\partial^2 x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$ $\frac{\partial^2 f}{\partial^2 y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$	Laplace Second derivative
(3.7)	$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)]$	Laplace
(3.17)	$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$	Gaussian
(3.8)	$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$	Gaussian
(3.9)	$LoG(x, y, \sigma) = -\left[\frac{(x^2 + y^2) - \sigma^2}{\sigma^4}\right] \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}}$	Laplacian of Gaussian
(3.10)	$DoG(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$	Difference of Gaussian
(3.11)	$E_{x,y} = \sum_{u,v} w_{u,v} \left\| I_{x+u, y+v} - I_{u,v} \right\|^2$	Intensity change (Moravec)
(3.12)	$E_{x,y} = Ax^2 + By^2 + 2Cxy$ $= (x, y) \cdot M(x, y) \cdot (x, y)^T,$	Intensity change (Harris, Hessian)
(3.12)	$M(x, y) = \begin{bmatrix} A & C \\ C & B \end{bmatrix},$ $A = \sum_{u,v} w(u,v) \otimes \left(\frac{\partial I}{\partial x}\right)^2$ $B = \sum_{u,v} w(u,v) \otimes \left(\frac{\partial I}{\partial y}\right)^2$ $C = \sum_{u,v} w(u,v) \otimes \left(\frac{\partial I^2}{\partial x \partial y}\right)$	Hessian matrix in Harris detector
(3.13)	$R = Det(M) - k \cdot Trace(M)^2$	Harris response

Table B2: List of formulas

(3.14)	$Grad = \lvert Gx \rvert + \lvert Gy \rvert$	Gradient magnitude
(3.15)	$\vartheta = \tan^{-1}\left(Gy / Gx\right)$	Gradient direction
(3.16)	$p = x\cos\theta + y\sin\theta$	Hough space
(3.20)	$\frac{Tr(M)^2}{Det(M)} = \frac{(\lambda_1+\lambda_2)^2}{\lambda_1\lambda_2} = \frac{(r\lambda_2+\lambda_2)^2}{r{\lambda_2}^2} = \frac{(r+1)^2}{r}\cdot\frac{{\lambda_2}^2}{{\lambda_2}^2} = \frac{(r+1)^2}{r}$	Edge elimination threshold
(3.21)	$\frac{Tr(M)^2}{Det(M)} < \frac{(r+1)^2}{r}$	Edge elimination threshold
(3.22)	$m(x,y) = \sqrt{(L(x+1,y)-L(x-1,y))^2 + (L(x,y+1)-L(x,y-1))^2}$ $\theta(x,y) = \arctan\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right)$	SIFT Gradient magnitude and direction
(4.1)	$[u,v]^T \sim K\left(R_{ort}\cdot[X,Y,Z]^T + \bar{t}\right)$	World to image transformation
(4.2)	$\min\sum_i \left[P(\bar{x}_{i,laser}) - \bar{x}_{i,color}\right]^2$	Optimization
(5.1)	Y = 0.212671*Re + 0.715160*Gr + 0.072169*Bl	Colour to grey value transform
(5.2)	$\sigma = \left(n/2 - 1\right)\cdot 0.3 + 0.8$	Standard deviation of Gaussian
(5.3)	$I_{new}(x,y) = a_1 * I_{old}(x,y) + a_2$	Histogram spreading
(5.3)	$a_1 = \frac{255}{(C\_right - C\_left)}$	Scale factor
(5.3)	$a_2 = -(C\_left)$	Shift
(5.4)	$Canny \cap Harris \cap Hough = key\ point\ dataset\ (u,v)$	Intersection for key point selection
(5.5)	$\frac{(\lambda_1+\lambda_2)^2}{\lambda_1\lambda_2} < \frac{(r+1)^2}{r}$	Eigen value threshold in edge eliminator
(5.6)	$G[x][y] = w\frac{1}{2\pi\sigma^2}e^{-((x-w/2)^2+(y-w/2)^2)/2\sigma^2}$	Discrete Gaussian function (2D)
(5.7)	$V' = \frac{V-\mu}{\sigma}$	Normalization
(5.8)	$V'' = \frac{(V'-\min)}{(\max-\min)}$	Shift after normalization