Linköping Studies in Science and Technology
Thesis No. 840

Vehicle Size and Orientation Estimation Using Geometric Fitting

Christina Carlsson

Division of Automatic Control
Department of Electrical Engineering
Linköpings universitet, SE–581 83 Linköping, Sweden
WWW: `http://www.control.isy.liu.se`
Email: `stina@lin.foa.se`

Linköping 2000

Vehicle Size and Orientation Estimation Using Geometric Fitting

Department of Electrical Engineering,
Linköpings universitet,
SE–581 83 Linköping,
Sweden.

ISBN 91-7219-790-0
ISSN 0280-7971
LiU-TEK-LIC-2000:36

Printed by UniTryck, Linköping, Sweden 2000

To Thomas and A

Abstract

Over the years imaging laser radar systems have been developed for both military and civilian applications. Among the applications we note collection of 3D data for terrain modelling and object recognition. One part of the object recognition process is to estimate the size and orientation of the object.

This thesis concerns a vehicle size and orientation estimation process based on scanning laser radar data. Methods for estimation of length and width of vehicles are proposed. The work is based on the assumption that from a top view most vehicles' edges are approximately of rectangular shape. Thus, we have a rectangle fitting problem.

The first step in the process is sorting of data into different lists containing object data and data from the ground closest to the object. Then a rectangle with minimal area is estimated based on object data only. We propose an algorithm for estimation of the minimum rectangle area containing the convex hull of the object data. From the rectangle estimate, estimates of the length and width of the object can be retrieved.

The first rectangle estimate is then improved using least squares methods based on both object and ground data. Both linear and nonlinear least squares methods are described. These improved estimates of the length and width are less biased compared to the initial estimates.

Three algorithms are evaluated; a minimum rectangle estimator proposed by the author, the mixed LS-TLS algorithm and a quasi-Newton algorithm. The algorithms are applied to both simulated and real laser radar data.

The thesis ends with a discussion of the assumptions that this work is based on and some suggestions of future work.

Acknowledgments

First of all, I would like to thank my supervisor Prof. Mille Millnert for his guidance in this work and into the world of research in general. I appreciate his patient during the periods when there were less progress in this work.
Second, I would like to thank the Swedish Defence Research Establishment (FOA) for letting me do this work, especially Dr. Ove Steinvall, head of the Laser Systems department. No matter how tight his schedule is, he always has time to discuss present and future laser radar systems, both regarding sensors and signal processing. I would also like to thank the head of the Sensor Division, Prof. Svante Ödman, for letting me do this work and supporting me financially so that it became possible.
Most of the work have been done in projects run by Prof. Erland Jungert, with whom I have stimulating discussions concerning object detection and classification and on geometric fitting. I am very grateful to him and to the other project members.
I would also like to thank Prof. Lennart Ljung and the Automatic Control group, Linköping University, for providing me the opportunity to join the group. Several of you have over the years helped me in theoretical discussions, with LaTeXquestions and other practical matters, thank you all!
The developers and manufacturers of the TopEye system, Saab Dynamics AB, Saab Survey Systems AB and TopEye AB have provided FOA with laser radar data over the years. I appreciate their interest in discussing the system and our wild ideas of applying extra sensors.
I would also like to thank Prof. Chi-Lun Cheng, Institute of Statistical Science, Academia Sinica, Taipei, Taiwan, for discussions that have deepened my knowledge in the asymptotic distributions of problems with measurement errors.
Those who read earlier versions of the manuscript are also acknowledged: Dr. Fredrik Gunnarsson, Division of Automatic Control, Linköping University, and Dr. Dietmar Letalick, Dr. Christer Karlsson, Dr. Thomas Kaiser, Tech. Lic. Michael Tulldahl, MSc Simon Ahlberg and MSc Jonas Svensson, FOA.
At FOA the network of female researchers (Brynja) has also been a great source of fun and support.
Finally, I would like to thank my family for their support over the years and maybe this thesis can encourage my sister Elisabeth to complete hers? And of course, I would like to thank Thomas for his support, love and patience all nights and weekends when I used our time to complete this thesis. The thesis is dedicated to him and our coming child.

Contents

Notation

Symbols

A	Matrix
I	Identity matrix
b	Vector
θ	Parameter vector
θ_0	True parameters (a vector)
$\hat{\theta}$	Estimated parameters (a vector)
$f(\theta)$	Criterion function to be minimized (by nonlinear LS methods)
$J(\theta)$	Jacobian of $f(\theta)$ (a matrix)
$H(\theta)$	Hessian of $f(\theta)$ (a matrix)
M	Length of the parameter vector θ
N	Number of samples
P_i	A data point (or vertex), $P_i = (x_i, y_i), i = 1, \ldots, N$
e_x	Noise in $x_i, i = 1, \ldots, N$
e_y	Noise in $y_i, i = 1, \ldots, N$
$\sigma^2_{e_x}$	Variance of e_x
n_i	Normal vector of the line (or edge) between points P_i and P_{i+1}
ϕ_i	Angle corresponding to the normal vector n_i
P^i	Vertex point belonging to the minimal rectangle, $i = 1, 2, 3, 4$
$\mathbb{R}^d$	Euclidian d-dimensional space
`foo.m`	Matlab commands are written in typewriter style

Operators and Functions

A^T	Matrix transpose
A^{-1}	Matrix inversion
$\left\|\cdot\right\|$	Euclidian norm (of a vector), e.g., $\left\|b\right\| = \sqrt{b_1^2 + \ldots + b_N^2}$
$\left\|\cdot\right\|_F$	Frobenius norm (of a matrix), e.g., $\left\|A\right\|_F = \sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J}\left\|a_{ij}\right\|^2}$
$E\left(x\right)$	Expectation value of $x = \left(x_1, \ldots, x_N\right)$
$\nabla f\left(\theta\right)$	Gradient of $f\left(\theta\right)$, i.e., $\nabla f\left(\theta\right) = \frac{\partial}{\partial\theta} f\left(\theta\right)$
$\nabla^2 f\left(\theta\right)$	Second gradient of $f\left(\theta\right)$, i.e., $\nabla^2 f\left(\theta\right) = \frac{\partial^2}{\partial\theta^2} f\left(\theta\right)$
$\mathrm{O}\left(\cdot\right)$	Computational complexity of an algorithm, e.g., an algorithm with a running time linearly proportional to the number of samples is denoted $\mathrm{O}\left(N\right)$
$\mathcal{R}\left(\cdot\right)$	Range operator
$\mathrm{sign}\left(\cdot\right)$	Sign operator
$\log_2\left(\cdot\right)$	Logarithm with base 2

Abbreviations

2D	Two dimensional
3D	Three dimensional
DTM	Digital terrain model
EKF	Extended Kalman filter
IR	Infra red
KF	Kalman filter
LS	Least squares (in general)
ML	Maximum likelihood
MSE	Mean square error
OLS	Ordinary least squares
PCA	Principal component analysis
PDF	Power density function
QSD	Qualitative slope descriptor
SCKF	Smoothly constrained Kalman filter
SVD	Singular value decomposition
TLS	Total least squares

1

Introduction

1.1 Laser radar systems

Laser radar systems have been investigated over several decades primarily for military applications, see for instance [14]. Laser radars are, just as conventional radars[1], mainly used for remote sensing. As in microwave radar technology, the range to object and background is often obtained by measuring the time of flight for a modulated laser beam from the transmitter to the object and back to the receiver. Some unique features in laser radar systems are high angular, range and velocity resolution. The high range resolution makes 3D imaging possible and due to the short wavelength, in general 0.5-10 μm, detailed range images of objects and background can be obtained. The high resolution also makes laser radars useful for velocimetry, range finding, obstacle avoidance, remote analysis of vibrations and wind sensing. Laser radar is sometimes called ladar[2] or lidar[3].

The laser radar system used, TopEye[4], is carried by a helicopter. It contains a vertical scanning direct detection laser radar operating at a wavelength of 1.06 μm. The pulse repetition frequency is 7 kHz and the emitted energy about 0.1 mJ per pulse. The accuracy is approximately 0.1 m in all directions of the 3D space. The laser beam is swept from side to side and when the helicopter flies this results in a zigzag-shaped scanning pattern, see Figure 1.1. An example of data from the system is shown in Figure 1.2. Each sample data point contains x, y and

[1] radar: radio detection and ranging
[2] ladar: laser detection and ranging
[3] lidar: light detection and ranging
[4] TopEye is a civilian system developed for terrain profiling. It is a product of TopEye AB, Sweden, see URL: http://www.topeye.com.

z coordinates where z is the altitude. Thus, the set of image points corresponds to a 3D mapping of the terrain. For an introduction to airborne scanning laser radar, we refer to [1], [23] and [25]. The TopEye system is further described in [13].

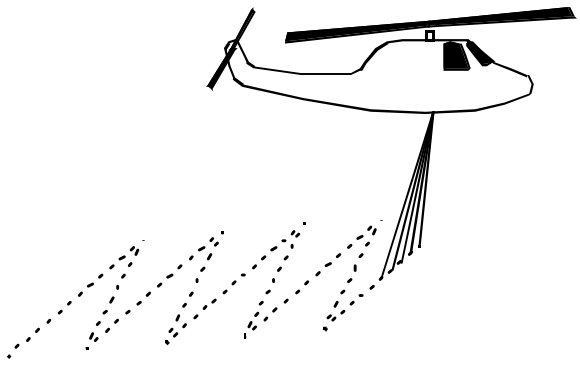

Figure 1.1: The scanning pattern of the TopEye laser radar system.

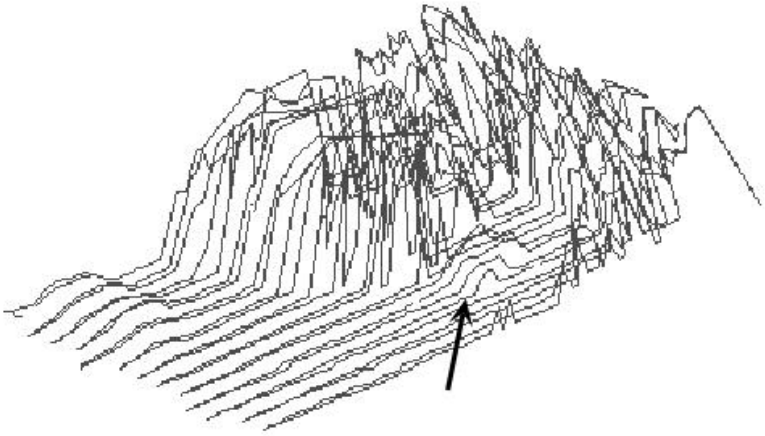

Figure 1.2: Example of the data from the TopEye laser radar system. The arrow marks a vehicle placed in an opening of a forest.

1.2 The object classification process

An object classification (or recognition) process can usually be divided into a detection phase, where data are analyzed to find if something interesting is present, and a classification phase, where the objects are roughly grouped (personal car, truck, tank etc.). The classification can be followed by an identification step, where the objects are separated within their group (Volvo, Saab etc.). Wellfare et al. [26] have developed a set of detection and vehicle identification algorithms based on data from (scanning) laser radar systems.

Another classification process for objects, especially vehicles, using imaging laser radar has been proposed in [16]. The process starts with search and detection of vehicles and ends with the result of the matching of the object with reference objects stored in a library database. This process can be divided into six steps:

1. Identification of regions of interest.
2. Segmentation of those regions.
3. Edge estimation of detected objects.
4. Construction of qualitative object descriptions.
5. Matching of the unknown object with reference objects (stored in a database).
6. Determination of a possibility value for a correct match.

In the first two steps the object detection is performed and the following steps perform the object classification. Some of these steps are fairly straight-forward while others are more complicated. However, a main goal in this work has been to design and implement a technique that can be run in real time or close to real time.

The first step is to find the regions of interest. This is performed, using differentiation in z direction, in the same order as the laser radar system generates the image elements, i.e., by following the zigzag pattern of the laser scanner. This means that detection is performed during sampling. Differentiation gives a simple but fast method to handle large data sets. Regions of man-made objects are, in principle, identified by finding pairs of derivatives that correspond to reasonable lengths and heights of interesting objects. In the second step, regions of reasonable length and width are grouped to form sets of segments for further investigation. The segmentation process is described further in [2] and [3].

The result of the detection is a list of segments, each represented by the (x, y, z) coordinates of the data points. Those segments are considered to be likely objects. A result of a segmentation is shown in Figure 1.3. In detail, the result of the segmentation process is a list containing detected points on the object and on the ground surrounding the object. Due to data uncertainties these points will not correspond to the actual positions of the edge of the object, see Figure 1.4. To overcome this the edges can be estimated with straight lines, which can be considered a lowpass filtering.

Our assumption is that most vehicles seen from above look like rectangles. The rectangle that approximate the object should fit the edge points as close as possible. In this case errors will occur both in x and y directions. Therefore, an estimation method that can handle errors in both coordinates will be used to fit a rectangle as close as possible to the edge data, by minimizing the residual. The rectangle estimate gives an approximation of the length, width and orientation of the object. The resulting minimized residual is a measure of how good the rectangle fits. This "quality measure" is needed in the next step of the object classification process. It must be possible to handle all kinds of rotations of the object independently of the scan pattern, as their directions are arbitrary relative to each other.

When features of the object are estimated, among them the length and width, it is possible to estimate its class (e.g., personal car, truck or tank) in a matching process. The construction of qualitative object descriptions, the matching process and the determination of the possibility value are described in [16].

Figure 1.3: Example of the object detection process. The grey- and black-marked points are the remaining data points after the region of interest search. The grey points remain after the segmentation step. The true objects (vehicles) are encircled. The covered area is approximately 160×30 m.

1.3 Problem description

During the object detection all three dimensions in data is concerned. In the edge estimation process the problem is reduced to $\mathbb{R}^2$, as the z values will not be used. The z values are ignored as they have approximately the same value (otherwise they would not be grouped into the same segment). In the middle of the scan the data points are about equally distributed between consecutive points within a scan line but also between scan lines. Close to the turning points of the scan the image point are not equally distributed, due to the zigzag pattern. All in all, output data of the laser radar system are distributed in an uneven, non-grid format. Data can be considered under-sampled, i.e., there are gaps between the samples. This means

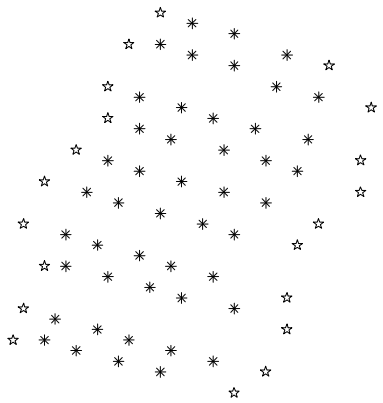

Figure 1.4: Example of a measurement of an object (a terrain vehicle). Stars ($*$) represent data points on the object and pentagrams ($\star$) data points on the ground nearest the object.

that the object's edge is placed between two samples and that the exact position of the object's edge is not known. Further, there is a very small data set describing the object, sometimes only 3-6 data points describe one edge on the object. According to the manufacturer, there are uncertainties in the positioning in both coordinates (x and y) of the same size. From this we assume that the position error in each data point has a Gaussian power density function (PDF) with equal variance in x and y, respectively, and zero mean.

We will study methods for fitting a rectangle to the data set of the object. The "best fit" is defined as a fit with minimal residual. The methods must take care of the fact that there are errors in both coordinates. Further, the methods must be rotation invariant, as we do not know how the object is placed relative to the scan direction.

This study is based on some assumptions. First, we assume that the data collection and the object detection have been successful, i.e., that no outliers are present. Second, we assume that the complete object is visible, i.e., parts of the object is not hidden under trees, camouflage etc. and that the object is not located close to another object, resulting in merging of the two objects in data.

1.4 Outline and contributions

In this thesis we propose a method for estimation of size and orientation of an object. This method is proposed as step 3 in the classification process described on page 3. The size and orientation estimation can be divided into several steps:

Given: A list of object points and ground points closest to the object, ordered in scan lines.

1. Sort all data points, both from the object and from the ground, in an order following the object edge clockwise or counter-clockwise.

2. Estimate the smallest rectangle that includes all data points from the object.

3. Associate each data point to one side of the rectangle (from step 2).

4. Improve the rectangle estimate using least squares methods based on both object and ground data.

5. Calculate the residual of the final rectangle estimate and the estimated length and width, respectively.

Returned: Estimates of length, width and orientation of the object. The residual of the rectangle estimate is also returned.

The outline of the thesis is as follows; in Chapter 2 we review some work on geometric fitting, both some general methods and some methods applied to data from scanning laser radar systems. The following chapters go through the estimation method sequentially. Chapter 3 describes the sorting process. Chapter 4 describes a method for calculation of the minimum rectangle that includes all data points on the object and ends with some tests (simulations) of the method. In Chapter 5 we improve the rectangle estimate by including both object and ground data. First we propose a method of association of the data points to the different sides of the rectangle. Then some linear and nonlinear least squares (LS) methods are described and tested. In Chapter 6 we perform tests using the best linear and nonlinear LS methods on both simulated realistic data and on data from the laser radar system. We end the thesis in Chapter 7 with conclusions and a discussion of the assumptions in this work and some suggestions of further work.

The proposed solution is based on the assumption that a rectangle, with minimum area, that contains all object points gives a good first estimate of size and orientation. Further, we believe that the estimates can be improved by including information of the statistics of the data set.

The main contributions of the thesis can be summarized as follows:

- A method for data sorting that requires $\mathrm{O}(N)$ time (Chapter 3).
- A method for minimum rectangle estimation that requires $\mathrm{O}(N)$ time (Chapter 4).
- A method for data association (Chapter 5).
- A method for object edge estimation based on both object data and data from the ground closest to the object (Chapter 5).

2

Geometric fitting

2.1 Introduction

In this chapter some work on geometric fitting is reviewed. First, we describe what is generally meant by geometric fitting and then some of the references on a very common fitting problem, fitting of ellipses, are described. Then follow two proposed ways of handling the rectangle fitting problem. Finally, some approaches for fitting of buildings and vehicles from scanning laser radar data are described.

For a geometric shape, e.g., an ellipse or a rectangle, the minimization criterion in *geometric fitting* is to minimize the squared sum of orthogonal distance between the data points and the shape. Let $\theta = (\theta_1, ..., \theta_M)^T$ be a vector of unknown parameters and consider the (linear or nonlinear) system of N equations $f_i(\theta) = 0, i = 1, ..., N$. If $N > M$, then we want to minimize

$$\min_{\theta \in \mathbb{R}^M} f(\theta) = \min_{\theta \in \mathbb{R}^M} \sum_{i=1}^{N} f_i(\theta) = \min_{\theta \in \mathbb{R}^M} \sum_{i=1}^{N} r_i^2(\theta),$$

where r_i is the orthogonal distance between data point i and the geometric shape. An advantage with the orthogonal distance is that it is invariant under transformations in Euclidean space. Geometric fitting is also called orthogonal distance regressions, orthogonal regressions, data fitting and errors-in-variables regression in the literature.

2.2 Fitting of ellipses

In [27], Zhang gives an overview of several parameter estimation techniques that can be used for conic fitting. Conic fitting is a general term that contains fitting of several expressions, e.g., circles and ellipses. A general conic can be described by the equation

$$Q(x, y) = ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0,$$

where $\theta = (a; b; c; d; e; f)$ are the parameters that are to be estimated. The techniques suitable for geometric fitting in [27] are:

1. **Gradient weighted least-squares fitting**. Both Zhang [27] and Van Huffel and Vandewalle [12] have shown that the ordinary least squares method does not return consistent estimates for the geometric fitting problem. This can be overcome by gradient weighting of the covariance matrix. The minimization expression usually gets very complicated and non-linear and iterative optimization algorithms must be used, e.g., Gauss-Newton.

2. **Bias-corrected renormalization fitting**. This method uses weighted least-squares minimization. The weights are positive and can be chosen to the inverse proportion of the variance. This method is optimal only in the sense of unbiasness. Further, the method is based on statistical analysis of the data points and is therefore only useful if the data set is large.

3. **Kalman filtering techniques**. In this approach the Kalman filter (KF) is applied to a spatial sequence instead of the usual temporal sequence. Furthermore, the observation function $f_i(\theta)$ is nonlinear:

   $$f_i(x_i, y_i, \theta) = \left(x_i^2 - y_i^2\right) a + 2x_i y_i b + 2x_i d + 2y_i e + f + y_i^2.$$

   If this expression is expanded in a Taylor series and second order and higher terms are ignored, the extended Kalman filter (EKF) can be applied. For nonlinear problems the EKF will yield different results depending on the order of processing the measurements and can be trapped in a local minimum. This is because KF and EKF are designed to handle one measurement at a time and if all data are available at the same time Zhang recommends bias-corrected renormalization.

In [6], Gander et al. describe several algorithms for geometric fitting of circles and ellipses. The curves are represented in parametric form:

$$Q(x, y) = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + 2 \begin{pmatrix} d & e \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + f = 0,$$

which is well suited for minimizing the sum of the squares of the distance. In the minimization of the geometric distance a nonlinear LS problem needs to be

solved. This is solved iteratively by a sequence of linear least squares problems. The authors perform a comparison of some nonlinear LS algorithms, among them Newton, Gauss-Newton, Gauss-Newton with Levenberg-Marquardt modification (these methods will be described in Section 5.4), with respect to stability and efficiency. Simulations show that it is crucial to obtain good starting values of the parameters for the iteration process. If the problem is well posed and the accuracy of the result should be high, the Newton method is the most efficient. Calculation of the geometric distance is relatively expensive, concerning the number of computations, and convergence is not guaranteed. A pragmatic way to limit the cost, suggested in [6], is to perform a fixed number of iterations.

2.3 Fitting of rectangles

There is a vast difference between fitting of rectangles and fitting of ellipses. In fitting of ellipses the function $f_i(\theta)$ describes the whole shape in one, continuous function. For rectangles each side of the rectangle can be described by one, continuous function, but the shape of the rectangle consists of four such functions and there are constraints between the functions. Altogether, the function $f_i(\theta)$ describing the shape of a rectangle is a discontinuous, constrained function. Two methods for geometric fitting of rectangles are described below. The first is a linear, constrained function description and the second is a nonlinear, constrained function description.

In [7], Gander and von Matt describe a linear, constrained least squares technique for geometric fitting of straight lines, rectangles and some other common shapes. In $\mathbb{R}^2$ a straight line can be uniquely represented by the equation

$$c + n_1 x + n_2 y = 0, \quad n_1^2 + n_2^2 = 1, \tag{2.1}$$

where the (unit) normal vector $(n_1; n_2)$ is orthogonal to the line and c is the perpendicular distance from the line to the origin. The constraint $n_1^2 + n_2^2 = 1$ guarantees uniqueness. A point $P_i = (x_i, y_i)$ is on the line if it satisfies (2.1). If P_i is not on the line we compute

$$c + n_1 x_i + n_2 y_i = r_i,$$

where r_i is the orthogonal distance between the line and P_i. Using the fact that $|r|$ and $|r|^2$ will have the same minimum, the geometric fitting problem can now be formulated as

$$\min_{c,n_1,n_2} |r|^2 = \min_{c,n_1,n_2} \sum_{i=1}^{N} r_i^2$$

subject to

$$\begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{pmatrix} \begin{pmatrix} c \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{pmatrix} \text{ and } n_1^2 + n_2^2 = 1.$$

Using the description of straight lines according to (2.1) the equations for the four sides of a rectangle can be achieved. Two parallel sides of the rectangle will have identical normal vector $(n_1; n_2)$ but different c parameters. The other two sides will be perpendicular to those sides and have the normal vector $(-n_2; n_1)$. Thus, the four sides of a rectangle will have the equations

$$\text{Side 1} : c_1 + n_1 x + n_2 y = 0 \tag{2.2a}$$
$$\text{Side 2} : c_2 - n_2 x + n_1 y = 0 \tag{2.2b}$$
$$\text{Side 3} : c_3 + n_1 x + n_2 y = 0 \tag{2.2c}$$
$$\text{Side 4} : c_4 - n_2 x + n_1 y = 0 \tag{2.2d}$$
$$\text{and} : n_1^2 + n_2^2 = 1. \tag{2.2e}$$

This model will be called *Rectangle model 1*. We will come back to this model and an algorithm for solving the problem in Chapter 5.

De Geeter et al. [8] present another method for estimation of a rectangle's edge. They parameterize the model of the rectangle as

$$\text{Side 1} \quad : \quad -(x_1 - x_c)\sin\alpha_1 + (y_1 - y_c)\cos\alpha_1 - d_1 = 0 \tag{2.3a}$$
$$\text{Side 2} \quad : \quad -(x_2 - x_c)\sin\alpha_2 + (y_2 - y_c)\cos\alpha_2 - d_2 = 0 \tag{2.3b}$$
$$\text{Side 3} \quad : \quad -(x_3 - x_c)\sin\alpha_3 + (y_3 - y_c)\cos\alpha_3 - d_3 = 0 \tag{2.3c}$$
$$\text{Side 4} \quad : \quad -(x_4 - x_c)\sin\alpha_4 + (y_4 - y_c)\cos\alpha_4 - d_4 = 0, \tag{2.3d}$$

where (x_c, y_c) is a reference point close to the edge, α_1 is the angle of side 1 with the x-axis and d_1 is the signed distance from side 1 to the point (x_c, y_c), see Figure 2.1. The parameters are $\theta = (\alpha_1; \alpha_2; \alpha_3; \alpha_4; l; w)$ and the following constraints between the parameters are used:

$$\text{Constraint 1} \quad : \quad \alpha_1 - \alpha_2 = \frac{\pi}{2} \tag{2.3e}$$
$$\text{Constraint 2} \quad : \quad \alpha_2 - \alpha_3 = \frac{\pi}{2} \tag{2.3f}$$
$$\text{Constraint 3} \quad : \quad \alpha_3 - \alpha_4 = \frac{\pi}{2} \tag{2.3g}$$
$$\text{Constraint 4} \quad : \quad |d_{13}| = l \tag{2.3h}$$
$$\text{Constraint 5} \quad : \quad |d_{24}| = w, \tag{2.3i}$$

where l and w is the length and width, respectively, of the rectangle. d_{13} is obtained by substituting a point on side 1 in the equation for side 3 and d_{24} is obtained

analogously. Constraints 1-3 are linear, and represent a clockwise listing of the lines while constraints 4-5 are nonlinear. This model will be called *Rectangle model 2*.

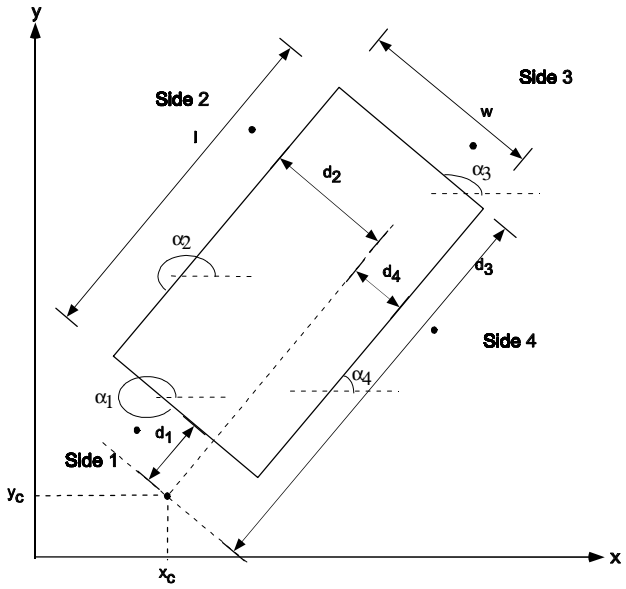

Figure 2.1: Rectangle parametrization according to De Geeter et al. [8].

Rectangle model 2 is used to estimate the position and orientation of a rectangular plate. The method used and proposed in [8], is a smoothly constrained Kalman filter (SCKF), which uses a smooth application of nonlinear constraints. In the rectangle estimation it is known on which side the measurement is taken and thus, there is no data association problem to solve. Measurements on the sides are added in consecutive iterations; in iteration 1 data from side 1 are added, in iteration 2 data from side 2 are added etc. The constraints are added when there are enough data from a side present so that the constraints are applicable.

2.4 Fitting of buildings and vehicles

When generating digital terrain models (DTM) based on laser radar data, one of the problems is separation of the object from the background. In building reconstruction top view data of the buildings are available. The first step in the building reconstruction process is to separate the building from the surrounding ground. When the DTM is generated for an urban area this step can be worked-around using ground plan data, i.e., a detailed 2D map of the area. This approach

is used by Haala and Brenner [11]. When the ground plan is matched with laser radar data, the edge of the building is given.

Maas and Vosselman [19] suggest two methods of building reconstruction from laser radar data that does not need ground plan information. The first approach is based on the analysis of invariant moments of the data points of the building, closed solutions for the parameters of standard gable roof house types are derived from 0th, 1st and 2nd order moments. Using the moments, seven house model parameters are calculated. Assuming a rectangular ground shape of a building, its ground dimensions (length and width) can be obtained directly from the formulation of 2nd order rotation invariant moments. After the determination of these house parameters, a goodness of fit is determined by projecting the house model into the segmented data set and computing height residual for every point. This allows for rejection of the computed house model in case of bad fit and for elimination of outliers in the data points.

The second approach is a data driven technique based on the intersection of planar faces in triangulated points. First the data points of the roof are clustered into different areas using (Delaunay) triangulation, see e.g. [20]. From the triangulation the laser points that belong to the edge of the roof are extracted. The roof edge is assumed to be equal to the edge of the building. The triangulation will give a very irregular description of the roof edge. Since the building (in 3D) can be described in polyhedral models, the roof edge can be described as connected straight lines. The straight lines are estimated such that the least square sum of the distances of the points to the edges is minimized, i.e., a geometric fit is performed. In the straight line estimation the lines are enforced to be either parallel or perpendicular to the main building orientation. As only roof data, and no ground data, is used in the straight line estimation the result is an underestimation of the size of the roof. This is compensated by constraining the estimation such that most data points (80%) are placed on the roof-side of the straight line.

Both methods were quite successful in tests. The second method had some problems to determine the outline of the building, especially when trees are located close to the building. On the other hand, using the second method more complex buildings can be determined.

In the reconstruction of vehicles the task is a bit more complicated than in building reconstruction. First, we cannot assume that there is a detailed ground plan of the area where the vehicle is placed. Second, the amount of measurements on the object is smaller, as vehicles normally are smaller than buildings. In [26] the object edge estimation is made using a modified Hough transform, see e.g. [10]. According to Zhang [27], the Hough transform is applicable when the amount of data is much larger than the number of parameters. In our case the amount of data is limited, and less than in [26], and the Hough transform will not be applicable.

Two different approaches for object edge estimation from raw laser data have been tested at FOA, both methods are working on object data only. In the first method, developed by Jungert, an object is described using a formal structure, see [15] and [16]. The formal structure that describes an atomic element of the

object is called a qualitative slope descriptor (QSD). Each QSD describes the orientation of a straight line, called line segment, between two points of the object. Complete objects are described in terms of sets of sequences of QSDs and their inter-relationships. The general representation of a QSD is defined as

$$(\text{sign}\,(\Delta x)\,,\text{sign}\,(\Delta y)\,,\langle\text{qualitative slope indicator}\rangle)\,,$$

where Δx and Δy are used to determine the slope coefficient of the line segment. The third parameter is the qualitative slope indicator which is equivalent to an angle interval. In the QSD processing of an object, all data points of the object's edge are connected with straight lines. For each line the QSD is applied and if two line segments belong to the same angle interval they are replaced by one line segment (and one QSD). The structure of the QSD allows determination of several local parameters, e.g., whether convexities or concavities exist and whether the angles between QSDs are acute or obtuse. Using a relative large angle interval, the irregularities in data of a vehicle's edge can be overcome. For a vehicle the final description usually is quite similar to a rectangle, see example in Figure 2.2a.

From the idea of QSD, Svensson suggests another approach [24]. For all data points on the object the major and minor axes of data are calculated using principal component analysis (PCA), see e.g. [10]. Then a rectangle is fitted such that all data points on the object are included and the sides of the rectangle are parallel to the major and minor axes, respectively. An example of the method is shown in Figure 2.2b.

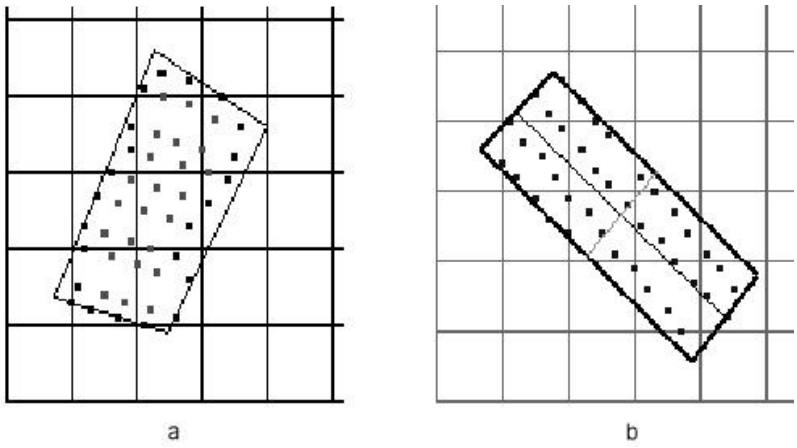

Figure 2.2: Edge estimation methods developed earlier at FOA.
a: QSD with angle intervals of $\pi/8$, b: rectangle placement using PCA.

3

Sorting data

The first step in the size and orientation estimation process is sorting of data. Data will be sorted so that object data and ground data follow the edge of the object clockwise or counter clockwise. In this chapter an algorithm for sorting of laser radar data is proposed.

3.1 Method

Typical results of the segmentation step are shown in Figure 3.1. The zigzag pattern shows the scan pattern of the laser system and the arrow shows the flight direction of the helicopter. The rectangle (dashed) represents the true border of the object and the circles represent measurements saved in the segmentation. The time indices, t_i, $i = 1, 2, ..., N$, represent the registration order of the samples. Each time index corresponds to a spatial coordinate (x_i, y_i). As can be seen in the figure, the measurements of the objects differ depending on the position of the object relative to the flight (and scan) direction. In each scan line not only the measurements on the object, but also the previous and following ground points are saved. We can also note that in each scan line the true object edge is somewhere between the first (or last) ground point and the first (or last) object point. The first and last object points in each scan line have special importance and will be called *outer object points*.

The edge estimation process assumes that data are ordered in such a way that the edge of the object can be followed clockwise or counter clockwise. Sorting data to that format is rather straightforward as the scan pattern is known. The sorting process will result in three lists for each object, see Figure 3.2; one with

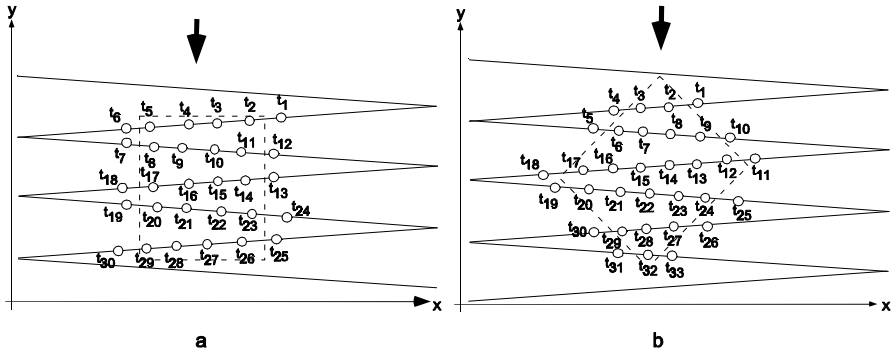

Figure 3.1: Example of data from the segmentation step (arbitrary data). The zigzag pattern shows the scan pattern of the laser and the arrow shows the flight direction of the helicopter. The dashed rectangle represents the true border of the object and the circles represent measurements saved in the segmentation process.

ground points, one with outer object points and one with inner object points. The ground points will be saved as $g_i = (x_i, y_i, \text{pair}_{\text{outer}})$, $i = 1, 2, ..., N_{\text{ground}}$, where $\text{pair}_{\text{outer}}$ refers to the nearest (previous or following) outer object point in the same scan line. The outer object points will be saved in the same manner; $o_i = \left(x_j, y_j, \text{pair}_{\text{ground}}\right)$, $j = 1, 2, ..., N_{\text{outer}}$, where $\text{pair}_{\text{ground}}$ refers to the nearest (previous or following) ground point in the same scan line. For example, in Figure 3.2a ground point g_1 will refer to outer object point o_1, and vice versa, but outer object point o_2 will not refer to any ground point ($\text{pair}_{\text{ground}} = \text{nil}$). The inner object points will be saved in the same order as in the original data, (x_k, y_k), $k = 1, 2, ..., N_{\text{inner}}$, $N = N_{\text{ground}} + N_{\text{outer}} + N_{\text{inner}}$. Data will be ordered clockwise or counter clockwise depending on the order of data in the first scan line. If the first scan line on the object goes from right to left (see Figure 3.1) ground points will be ordered clockwise and outer object points counter clockwise. If the first scan line on the object goes from left to right ground points will be ordered counter clockwise and outer object points clockwise. Results of sorting data in Figure 3.1 are shown in Figure 3.2.

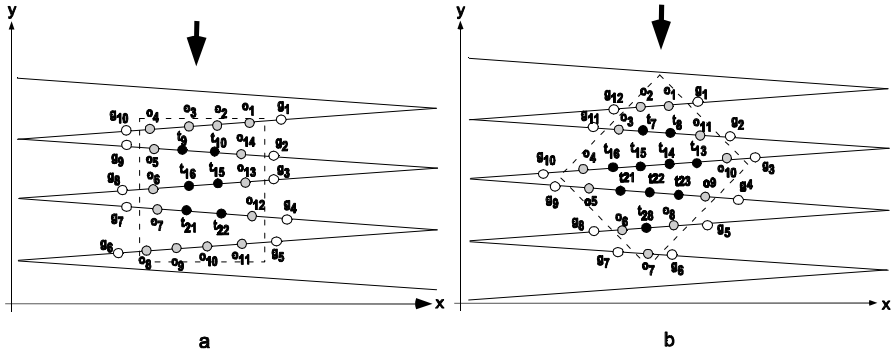

Figure 3.2: Result of sorting of data in Figure 3.1a and b, respectively. White points: ground data, grey: outer object data and black: inner object data.

3.2 Algorithm

The sorting process can be described in an algorithm that reads the input file line by line, where each line contains information of one sample, t_i.

Algorithm 1 (Data sorting)

Step 1: *Calculate the number of scan lines, S, by traversing data once.*
Step 2: $S = 1$:
Add first ground point, with $\text{pair}_{\text{outer}} = 1$, *in a ground point list G.*
Store the first outer object point, with $\text{pair}_{\text{ground}} = 1$, *in an outer object list O.*
Add the following outer object points, with $\text{pair}_{\text{ground}} = \text{nil}$, *last in O.*
Add the last outer object point, with $\text{pair}_{\text{ground}} = 2$, *last in O.*
Store the last ground point, with $\text{pair}_{\text{outer}} = 2$, *in another ground point list G'.*
Step 3: *For* $s = 2, ..., S-1$
If s is even
Add first ground point, with $\text{pair}_{\text{outer}} = 3, 7, 11$ *etc., last in G'.*
Add first outer object point, with $\text{pair}_{\text{ground}} = 3, 7, 11$ *etc., last in O.*
Store inner object points in time order in an inner point list I.
Store last outer object point, with $\text{pair}_{\text{ground}} = 4, 8, 12$ *etc., in another outer object list O'.*
Add last ground point, with $\text{pair}_{\text{outer}} = 4, 8, 12$ *etc., last in G.*
else
Add first ground point, with $\text{pair}_{\text{outer}} = 5, 9, 13$ *etc., last in G.*
Add first outer object point, with $\text{pair}_{\text{ground}} = 5, 9, 13$ *etc., last in O'.*
Add inner object points in time order last in I.
Add last outer object point, with $\text{pair}_{\text{ground}} = 6, 10, 14$ *etc., in O.*

Add last ground point, with $\text{pair}_{\text{outer}} = 6, 10, 14$ *etc., last in* G'.

Step 4: $s = S$*:*

If s *is even*

Add first ground point, with $\text{pair}_{\text{outer}} = N_{\text{outer}} - 1$*, last in* G'.
Add first outer object point, with $\text{pair}_{\text{ground}} = N_{\text{outer}} - 1$*, last in* O.
Add the following outer object points, with $\text{pair}_{\text{ground}} = \text{nil}$*, last in* O.
Add last outer object point, with $\text{pair}_{\text{ground}} = N_{\text{outer}}$*, last in* O.
Add last ground point, with $\text{pair}_{\text{outer}} = N_{\text{outer}}$*, last in* G.

else

Add first ground point, with $\text{pair}_{\text{outer}} = N_{\text{outer}} - 1$*, last in* G.
Add first outer object point, with $\text{pair}_{\text{ground}} = N_{\text{outer}} - 1$*, last in* O'.
Add the following outer object points, with $\text{pair}_{\text{ground}} = \text{nil}$*, last in* O'.
Add last outer object point, with $\text{pair}_{\text{ground}} = N_{\text{outer}}$*, last in* O'.
Add last ground point, with $\text{pair}_{\text{outer}} = N_{\text{outer}}$*, last in* G'.

Step 5: *Append* O' *last in* O *in reverse order.*

Step 6: *Append* G' *last in* G *in reverse order.*

3.3 Execution time

When there is no knowledge of the structure of data in the input file available the fastest sorting algorithms require $\mathrm{O}(N \log_2 N)$ time, where $\log_2$ is the logarithm function with base 2, see for example [21]. In this case data can be considered as partly sorted, as we have knowledge of the data structure, and the algorithm will require less time. In the algorithm above, step 1 requires $\mathrm{O}(N)$ time as input data is only scanned once. In steps 2-4 input data is only scanned once and therefore, they require $\mathrm{O}(N)$ time. The simple operations in steps 5-6 will also require $\mathrm{O}(N)$ time. Altogether, the sorting algorithm for this data format requires $\mathrm{O}(N)$ time.

4

Calculation of the smallest rectangle

4.1 Introduction

When data have been sorted, the next step is to estimate the edge of the object. It is important to be careful in this step, otherwise unnecessary errors are transferred, and maybe even amplified, to the matching process. Due to the data format and the uncertainties, the object's edge will not look like a rectangle, see Figure 4.1. Therefore, we need to estimate the edge using the knowledge of the data format and the sampling procedure.

In this step we will fit a rectangle as good as possible to object data by minimizing the rectangle's area. We assumed in the beginning of this thesis that the "true" object edge lies between the measurements of the object and the measurements of the ground. If we assume that data is error free and calculate the rectangle with smallest possible area that includes all object data, we will get an estimate of the minimum area of the object. Our hypothesis is that the minimum rectangle area that includes all object points is a good, first estimate of the size and orientation of the object.

A straight line is completely described by the coordinates of a point on the line and a slope. For four lines we need the coordinates of a point and a slope for each line. In a rectangle the slopes of the lines (or sides) are linked - two lines are parallel and two are perpendicular to the other. Consequently, a rectangle can be completely described by four points and an orientation (slope) parameter. The question is how those four points and the orientation shall be selected to find the rectangle with the smallest area possible.

We will propose a method that finds the rectangle with smallest area that contains all object points. Let us place a rectangle of arbitrary size placed in an arbitrary orientation so that all object points are on the interior of the rectangle. When we minimize the rectangle's area (with that specific orientation) the area is minimal when all sides hit one of the vertices in the convex hull. If we perform this minimization in all possible orientations, we will find the rectangle with minimal area.

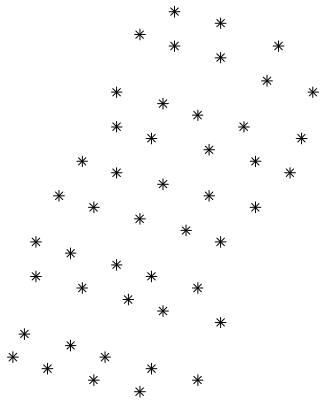

Figure 4.1: Example of (real) object data. Each star (∗) represents a data point on the object.

4.2 Calculation of the convex hull

4.2.1 Theory

To formalize the reasoning in the previous section, we need some definitions. In [20], page 18, the following is defined:

Definition 1 ($\mathbb{R}^d$)
Let $\mathbb{R}^d$ denote the d-dimensional Euclidean space, i.e., the space of the d-tuples $(x_1, \ldots, x_d)$ of real numbers $x_i = 1, \ldots, d$ with metric $\left(\sum_{i=1}^{d} x_i^2\right)^{1/2}$.

Definition 2 (Convex set)
A domain D in $\mathbb{R}^d$ is convex if, for any two points q_1 and q_2 in D, the segment $\overline{q_1 q_2}$ is entirely contained in D.

Definition 3 (Convex hull)
The convex hull of a set of points S in $\mathbb{R}^d$ is the boundary of the smallest convex domain in $\mathbb{R}^d$ containing S.

Definition 4 (Simple polygon, $\mathbb{R}^2$)
A polygon is simple if there is no pair of nonconsecutive edges sharing a point. A simple polygon partitions the plane into two disjoint regions, the interior (bounded) and the exterior (unbounded) that are separated by the polygon.

Two steps are required to find the convex hull; identify the extreme points and order these points so that they form a convex polygon. A basic convex hull algorithm is Graham's scan [20]. In the algorithm, first an internal point is found and data are transformed so that the internal point is at the origin. The N points are then sorted by polar angle and if two points have the same polar angle the squared distances to the origin are compared. In the next step the list is traversed (or scanned) and all internal points are eliminated. The points that remain in the list are the convex hull vertices in the required order. During the sorting a simple but important "trick" is used. The sorting is performed by pair-wise comparisons and it is only needed to determine which of the angles is greater, the numerical value is not required. Given two points P_1 and P_2 in the plane, P_2 forms a strictly smaller polar angle with the real axis than P_1 if and only if the triangle (O, P_1, P_2) has strictly positive signed area[1], see Figure 4.2.

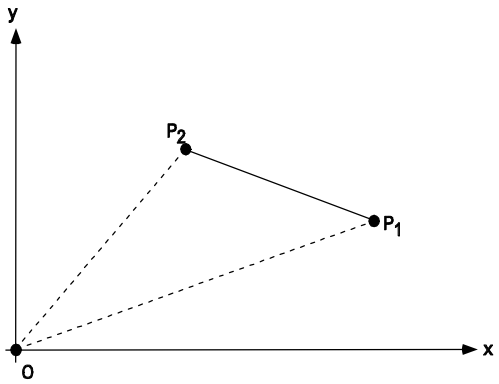

Figure 4.2: Comparing polar angles by means of the signed area of triangle (O, P_1, P_2), copy of Figure 3.5 in [20].

The convex hull of N points in the plane can be found, using Graham's scan, in $\mathrm{O}(N \log_2 N)$ time using only arithmetic operations and comparisons. The time limiting operation in the algorithm is the sorting, which requires $\mathrm{O}(N \log_2 N)$ time. The other operations, finding the internal point, transforming data and scanning

[1] The sign of the area is evaluated by a 3×3 determinant in the points' coordinates. The determinant of (O, P_1, P_2) is
$\det \begin{pmatrix} 0 & 0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix} = x_1 y_2 - x_2 y_1$. The area of the triangle (O, P_1, P_2) is positive if the determinant is positive.

the sorted list, require $\mathrm{O}(N)$ time. It can be shown that Graham's scan is optimal, but it has some limitations:

1. It will always use $\mathrm{O}(N \log_2 N)$ time, regardless of the data, because its first step is to sort the input.

2. Is cannot be generalized to higher dimensions, it is only applicable in $\mathbb{R}^2$.

3. It is an off-line algorithm since all points must be available before the processing begins.

4. It does not support a parallel environment as it cannot be split into smaller subproblems.

A special case of the convex hull problem is to find the convex hull for a simple polygon. For a simple polygon data can be considered sorted and the convex hull can be constructed in linear time:

Theorem 1 (Convex hull of a simple polygon)
The convex hull of an N-vertex simple polygon can be constructed in optimal $\mathrm{O}(N)$ time and $\mathrm{O}(N)$ space.

For a proof see [20], Theorem 4.12.

4.2.2 Application of the theory

In our application, data that can belong to the convex hull originates from the first and last object points in each scan line, the outer object points, and therefore, it is sufficient to calculate the convex hull using those data only. In the sorting process (Chapter 3) outer object points were sorted to follow the object clockwise or counter clockwise, i.e., into a simple polygon (compare with Definition 4). Compared to Graham's scan, the sorting is already performed and the remaining step in the convex hull calculation is to eliminate internal points. The internal points can be eliminated in $\mathrm{O}(N)$ expected time, which is in agreement with Theorem 1. A method like Graham's scan is sufficient as we are working in $\mathbb{R}^2$ and in off-line mode (all data are available from the segmentation step). The convex hull for data depicted in Figure 4.1 is shown in Figure 4.3.

In [21], Sedgewick notes that a fundamental property of the convex hull is that any line outside the hull, when it is moved in any direction towards the hull, hits one of its vertex points. Figure 4.4 shows all measurements of the object from Figure 4.1 and a rectangle of arbitrary size placed in an arbitrary orientation. In Figure 4.5 it is shown that when we minimize the rectangle's area (in that specific orientation) the area is minimized when all sides hit one of the vertices in the convex hull. Compare the marked points in the figure with the convex hull in Figure 4.3.

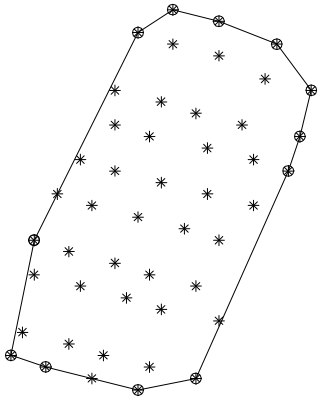

Figure 4.3: The convex hull for data depicted in Figure 4.1. Points belonging to the convex hull are marked by circles.

4.3 Description of the rectangle's area

For each edge of the convex hull a normal vector can be calculated using basic linear algebra. The normal vector, n_i, to the edge between the vertices P_i and P_{i+1}, $P_i = (x_i, y_i)^T$, $i = 1, 2, ..., N$, is

$$n_i = \frac{1}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \begin{pmatrix} -(y_{i+1} - y_i) \\ x_{i+1} - x_i \end{pmatrix} = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}_i .$$

This definition of the normal vector means that if data are listed counter clockwise all normal vectors will be directed into the convex hull and if data are listed clockwise all normal vectors will be directed out from the convex hull. The direction in itself does not matter, what is important is that all normal vectors around the convex hull uses the same definition. The normal vectors corresponding to the convex hull depicted in Figure 4.3 are shown in Figure 4.6. Let us now draw a straight line that touches a vertex point but not the interior of the convex hull. Consider a vertex of the convex hull, see Figure 4.7, we can see that a line that is *not* going through the interior of the convex hull has a normal vector that points into the convex hull and whose value is limited by the normal vectors of the closest edges. When the line's normal vector has a direction that is not bounded by the normal vectors of the closest edges the line passes the interior of the convex hull and thus, it is invalid. This means that for each vertex P_i there is a normal vector interval, with a minimum vector $n_{i,\min}$ and a maximum vector $n_{i,\max}$, see Figure 4.7, that a line's normal vector must belong to, i.e., $n_{i,\min} \leq n_{\text{line}} \leq n_{i,\max}$. Each edge normal n_i can also be represented as an angle ϕ_i and the line's normal vector can be represented as an angle ϕ_{line}. Thus, for each vertex P_i there is an angle interval, with a minimum angle $\phi_{i,\min} = \phi_{i-1}$ and a maximum angle $\phi_{i,\max} = \phi_i$ (if data

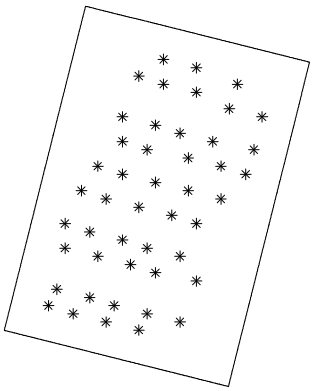

Figure 4.4: Object data and a rectangle in arbitrary orientation.

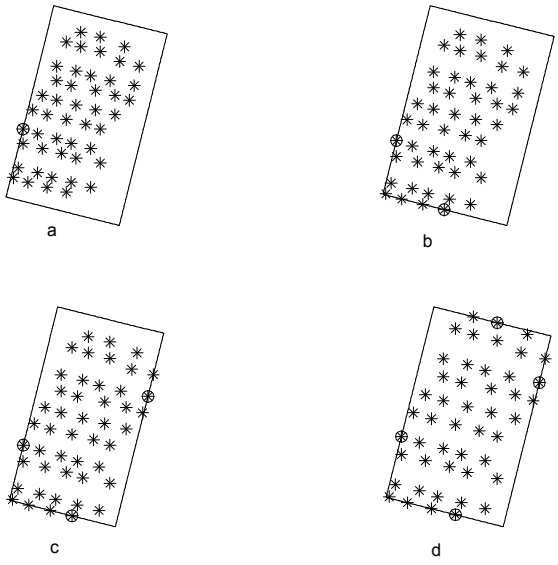

Figure 4.5: Shrinking the rectangle in Figure 4.4, side by side, until it hits the convex hull. The points that are hit by the rectangle's sides are marked by circles.

are listed counter clockwise), that ϕ_{line} must belong to; $\phi_{i,\min} \leq \phi_{\text{line}} \leq \phi_{i,\max}$. As can been seen in Figure 4.6, the angle intervals form a non-overlapping inclusive decomposition of the interval $[0, 2\pi]$.

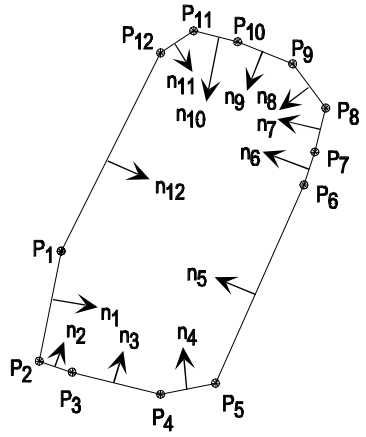

Figure 4.6: Convex hull with vertices, edges and normal vectors.

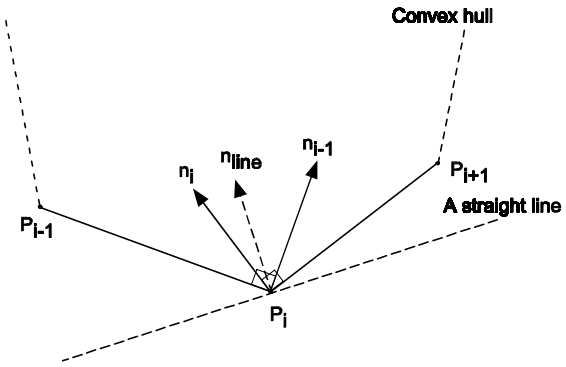

Figure 4.7: A straight line going through vertex P_i. The valid interval for the line's normal vector is $n_{i-1} \leq n_{\text{line}} \leq n_i$.

Assume that there exist four vertices P^1, P^2, P^3 and P^4, and a normal vector $n = \begin{pmatrix} n_1, & n_2 \end{pmatrix}^T$ for which a rectangle that includes the convex hull can be defined. The sides of the rectangle can be described by

$$\text{Side 1} \quad : \qquad P^1 n + c_1 = (x_1, y_1) \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} + c_1 = 0 \tag{4.1a}$$

$$\text{Side 2} \quad : \qquad P^2 \widetilde{n} + c_2 = (x_2, y_2) \begin{pmatrix} -n_2 \\ n_1 \end{pmatrix} + c_2 = 0 \tag{4.1b}$$

$$\text{Side 3} \quad : \qquad P^3 n + c_3 = (x_3, y_3) \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} + c_3 = 0 \tag{4.1c}$$

$$\text{Side 4} \quad : \qquad P^4 \widetilde{n} + c_4 = (x_4, y_4) \begin{pmatrix} -n_2 \\ n_1 \end{pmatrix} + c_4 = 0, \tag{4.1d}$$

where c_1, c_2, c_3, and c_4 are the perpendicular distances between the line and the origin, $\widetilde{n} = Rn$ and $R = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ is the rotation matrix (rotation $\frac{\pi}{2}$ radians counter clockwise), see Figure 4.8. The normal vector corresponds to an orientation of the rectangle.

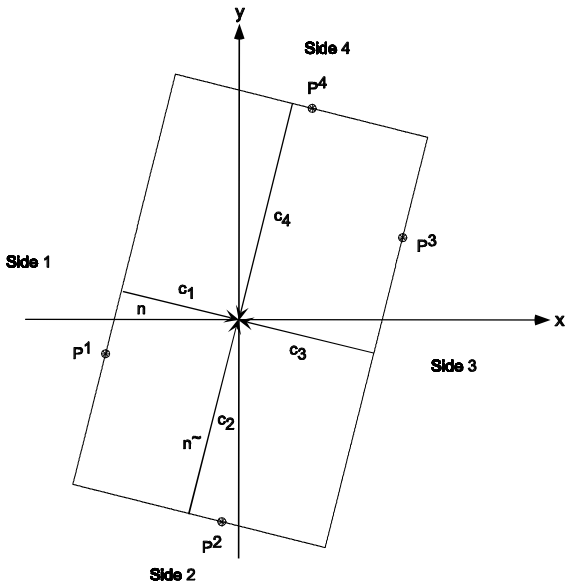

Figure 4.8: A rectangle with four points and a normal vector n (and $\widetilde{n}$). The lengths c_1, c_2, c_3 and c_4 are marked.

The rectangle's area can be calculated from

$$A = (c_1 - c_3)(c_2 - c_4).$$

Inserting the expressions of c_1, c_2, c_3, and c_4 from (4.1a)-(4.1d) gives

$$\begin{aligned} A &= \left(\left(P^3 - P^1\right) n\right)^T \left(P^4 - P^2\right) Rn \\ &= n^T \left(P^3 - P^1\right)^T \left(P^4 - P^2\right) Rn. \end{aligned} \tag{4.2}$$

The normal vector can be considered as a function of ϕ, i.e., $n = n(\phi)$. The normal vector n, and ϕ, do only belong to the interior of the convex hull for a certain angle interval $[\phi_{\min}, \phi_{\max}]$. Within this angle interval (4.2) can be interpreted as the area of a rectangle that includes the convex hull:

$$A(\phi) = n(\phi)^T \left(P^3 - P^1\right)^T \left(P^4 - P^2\right) Rn(\phi), \ \phi_{\min} \leq \phi \leq \phi_{\max}. \tag{4.3}$$

4.4 Finding feasible vertices

One problem is how to select the four feasible vertices P^1, P^2, P^3 and P^4. A complete search, which for example can be implemented by a hypothesis tree, will contain $(N-1)^4$ possible combinations of data points. This will in many cases be too many alternatives to be able to handle in real time or in near real time (for $N = 11$ there will be about 10.000 possible combinations).

Instead we propose a method that uses the allowed angle intervals to find the feasible vertices. For each of P^1, P^2, P^3 and P^4 there is an angle interval that has to be fulfilled if the vertex shall be part of the rectangle, where

$$\begin{aligned} &\text{Side 1} \ : && P^1 \text{ has an angle interval } \left(\phi_{1,\min}, \phi_{1,\max}\right) \\ &\text{Side 2} \ : && P^2 \text{ has an angle interval } \left(\phi_{2,\min}, \phi_{2,\max}\right) \\ &\text{Side 3} \ : && P^3 \text{ has an angle interval } \left(\phi_{3,\min}, \phi_{3,\max}\right) \\ &\text{Side 4} \ : && P^4 \text{ has an angle interval } \left(\phi_{4,\min}, \phi_{4,\max}\right). \end{aligned}$$

To represent the four sides of the rectangle the sides shall be placed with an angular difference of $\frac{\pi}{2}$. This can be interpreted as that the angle intervals of the vertices must be compatible, i.e., $\left(\phi_{1,\min}, \phi_{1,\max}\right)$, $\left(\phi_{2,\min}, \phi_{2,\max}\right) - \frac{\pi}{2}$, $\left(\phi_{3,\min}, \phi_{3,\max}\right) - \pi$ and $\left(\phi_{4,\min}, \phi_{4,\max}\right) - \frac{3\pi}{2}$ shall have a non-empty intersection. That intersection forms a smaller angle interval $(\phi_{\min}, \phi_{\max})$, which is the valid angle interval for the four vertices when describing the area as in (4.3).

4.5 Minimizing the area

Our goal is to find a rectangle with *minimum* area that contains the convex hull. Let us now study how to minimize the area, starting with (4.3):

$$\min_{\phi_{\min}\leq\phi\leq\phi_{\max}} A(\phi) = \min_{\phi_{\min}\leq\phi\leq\phi_{\max}} n(\phi)^T \left(P^3 - P^1\right)^T \left(P^4 - P^2\right) Rn(\phi). \quad (4.4)$$

Setting $\bar{Q} = \left(P^3 - P^1\right)^T \left(P^4 - P^2\right) R$ gives

$$A(\phi) = n(\phi)^T \bar{Q} n(\phi). \quad (4.5)$$

By construction the columns of $\bar{Q}$ are linearly dependent, i.e., $\operatorname{rank}\left(\bar{Q}\right) = 1$. This gives that one of the eigenvalues of $\bar{Q}$ is zero, i.e., $\lambda_{\min} = 0$, which is the lower limit of $A(\phi)$;

$$0 \leq n(\phi)^T \bar{Q} n(\phi) \leq \lambda_{\max}.$$

This means that $\bar{Q}$ is a positive, semidefinite, symmetric matrix that can be written

$$Q = qq^T,$$

where $q = \begin{bmatrix} q_1 & q_2 \end{bmatrix}^T$ is a real-valued vector (see for example [17]). Inserting this expression of Q into (4.5) gives

$$A(\phi) = \left(q^T n(\phi)\right)^2,$$

or

$$A(\phi) = \left(q_1 \cos(\phi) + q_2 \sin(\phi)\right)^2,$$

as $n(\phi)$ is a normalized vector that can be written $n(\phi) = \begin{bmatrix} \cos(\phi) & \sin(\phi) \end{bmatrix}^T$.

Finally, the minimization criteria (4.4) can be written

$$\min_{\phi_{\min}\leq\phi\leq\phi_{\max}} A(\phi) = \min_{\phi_{\min}\leq\phi\leq\phi_{\max}} \left(q_1 \cos(\phi) + q_2 \sin(\phi)\right)^2. \quad (4.6)$$

$A(\phi)$ is a continuous function, see example in Figure 4.9. We also know that $A(\phi) = 0$ is not a valid solution to (4.6). Further, $A(\phi) \neq 0$ is a monotonic function. Consequently, (4.6) will have its minimum either for $\phi = \phi_{\min}$ or for $\phi = \phi_{\max}$. The searched normal vector n, i.e., the one representing the minimum area, corresponds to the ϕ value that solves (4.6).

4.5.1 Algorithm

The algorithm for finding the feasible vertices and the normal vector that minimizes the rectangle's area can be motivated as follows. Consider the angle intervals of the convex hull in Figure 4.6 on a line axis, see Figure 4.10. A rectangle, with four

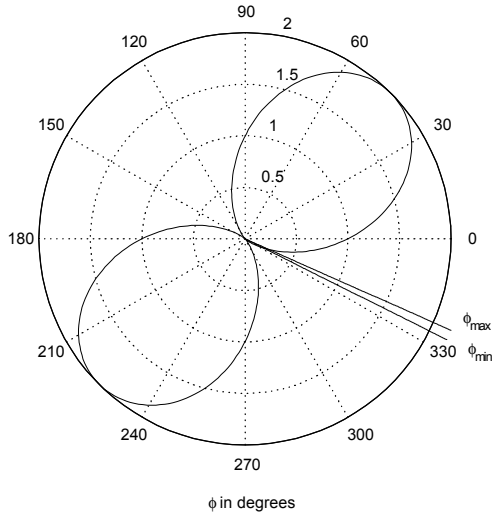

Figure 4.9: $A(\phi)$ as a function of ϕ for $q_1 > 0$ and $q_2 > 0$, $\phi_{\min}$ and $\phi_{\max}$ are marked. The values correspond to the estimate of the smallest rectangle of data depicted in Figure 4.1.

perpendicular sides, can be viewed as there need to be $\frac{\pi}{2}$ radians between the angle intervals $\left(\phi_{i,\min}, \phi_{i,\max}\right)$ of interest. Thus, one can place four pointers, ρ_1, ρ_2, ρ_3 and ρ_4, on the axis with a separation of $\frac{\pi}{2}$. The angle intervals that they point into represent a combination of four vertices. Then calculate the smallest increment Δ (for all the pointers) that is needed for one of the pointers to point into a new angel interval. Then move all the pointers the smallest step, Δ, to get a new combination of vertices. Remember that the angle intervals form a non-overlapping inclusive decomposition of the interval $[0, 2\pi]$. Further, the algorithm is working cyclic so that 0 radians equals 2π radians. When the pointers have been moved a quarter of a circle, $\frac{\pi}{2}$ radians, all possible combinations of vertices have been found.

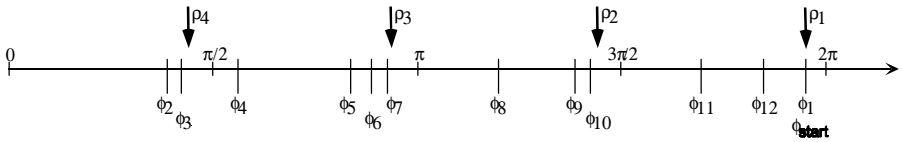

Figure 4.10: The angle intervals of Figure 4.6 depicted on a line axis.

Algorithm 2 (Minimum rectangle estimator)

Step 1: *For each* P_i *calculate* n_i *and* $\left(\phi_{i,\min}, \phi_{i,\max}\right)$, $i = 1, 2, ..., N$.
Step 2: *Set* $\phi_{start} = \max\left(\phi_i\right), i = 1, 2, ..., N, \rho_1 = \phi_{start}, \rho_2 = \rho_1 - \frac{\pi}{2}$, $\rho_3 = \rho_1 - \pi$ *and* $\rho_4 = \rho_1 - \frac{3\pi}{2}$.
Step 3: *While* $\rho_2 < \phi_{start}$*:*
Select the vertices P^1, P^2, P^3 *and* P^4 *corresponding to* ρ_1, ρ_2, ρ_3 *and* ρ_4*, respectively.*
Calculate $(\phi_{\min}, \phi_{\max})$ *using*
$\phi_{\min} = \max\left(\phi_{1,\min}, \phi_{2,\min} - \frac{\pi}{2}, \phi_{3,\min} - \pi, \phi_{4,\min} - \frac{3\pi}{2}\right)$ *and*
$\phi_{\max} = \min\left(\phi_{1,\max}, \phi_{2,\max} - \frac{\pi}{2}, \phi_{3,\max} - \pi, \phi_{4,\max} - \frac{3\pi}{2}\right)$.
Calculate $A\left(\phi_{\min}\right)$ *and* $A\left(\phi_{\max}\right)$ *using* (4.3).
Select the smallest area of $A\left(\phi_{\min}\right)$ *and* $A\left(\phi_{\max}\right)$.
Save P^1, P^2, P^3, P^4, ϕ *and* $A\left(\phi\right)$ *corresponding to the smallest area in a list.*
Find the smallest step Δ*, for* ρ_1, ρ_2, ρ_3 *and* ρ_4*, to the next angle interval* ϕ_i.
Increment ρ_1, ρ_2, ρ_3 *and* ρ_4 *by* Δ.
Step 4: *Find the set of vertices and normal vector in the list that is subject to the smallest area. This is the searched set of vertices and normal vector.*
Step 5: *Retrieve the length and width estimates from*
length $= \max\left(c_1 - c_3, c_2 - c_4\right)$ *and* width $= \min\left(c_1 - c_3, c_2 - c_4\right)$,
where c_1, c_2, c_3 *and* c_4 *are calculated by* (4.1*a*)-(4.1*d*).

Results of the three best rectangle estimates of data depicted in Figure 4.1 are shown in Figure 4.11. In the example there were 47 data points on the object, of whom 23 were outer object points and the convex hull contained 12 points. 10 point combinations were qualified, but only the three best (i.e., with smallest area) are shown.

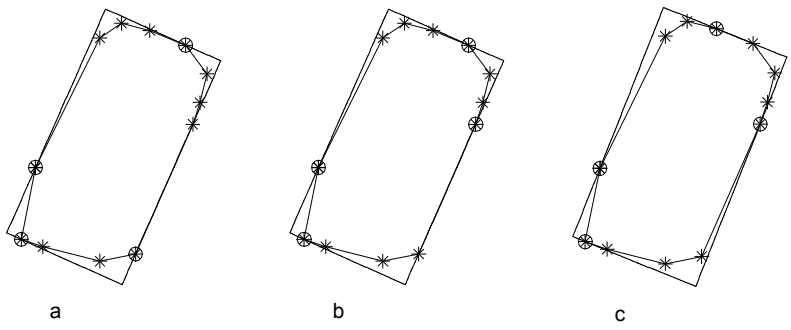

Figure 4.11: The three smallest rectangle estimates for data depicted in Figure 4.1. The data points of the convex hull are marked with stars, points marked with circles represent P^1, P^2, P^3 and P^4. (a: smallest area, b: second smallest area, c: third smallest area).

4.5.2 Execution time

Steps 1, 2, 4 and 5 in the algorithm are simple operations that can be performed in linear time. The loop in step 3 might be more time consuming, but it traverses data only once and can also be performed in linear time. Totally, the execution time of the algorithm is linear, i.e., $\mathrm{O}(N)$.

4.6 Performance

The performance of this rectangle estimation method will be investigated in some Monte Carlo simulations. The performance will be evaluated in terms of correctness in length and width estimates.

We start with an exact (error free) description of the rectangle, the test rectangles are viewed in Figure 4.12. Random errors, normal distributed with zero mean and variance σ_v^2 are added to the coordinates x and y, respectively. The noise is generated separately for each side and for each side separately in the x and y coordinates. The noise can therefore be considered independently and identically distributed with $\sigma_{e_x}^2 = \sigma_{e_y}^2 = \sigma_v^2$. Then length and width parameters are estimated using the minimum rectangle method on the perturbed data set. The simulations are repeated for increasing noise variance σ_v^2. The statistical properties of the length and width estimates are studied by the mean square error (MSE), defined as (see [12], page 244)

$$MSE\left(\hat{\theta}\right) = E\left\{\left(\hat{\theta} - \theta_0\right)^T \left(\hat{\theta} - \theta_0\right)\right\} \tag{4.7}$$

and estimated by

$$MSE\left(\hat{\theta}\right) = \frac{1}{J}\sum_{j=1}^{J}\left(\hat{\theta}_j - \theta_0\right)^2.$$

The MSE is averaged over 100 sets (i.e., $J = 100$). Each side of the rectangle is represented by 1000 data points. The test was performed in Matlab on a Windows NT computer (single processor).

The simulations show that the estimates of length and width have equal behavior concerning MSE. Results of length and width estimates for rectangle4 in Figure 4.12 are shown in Figure 4.13.

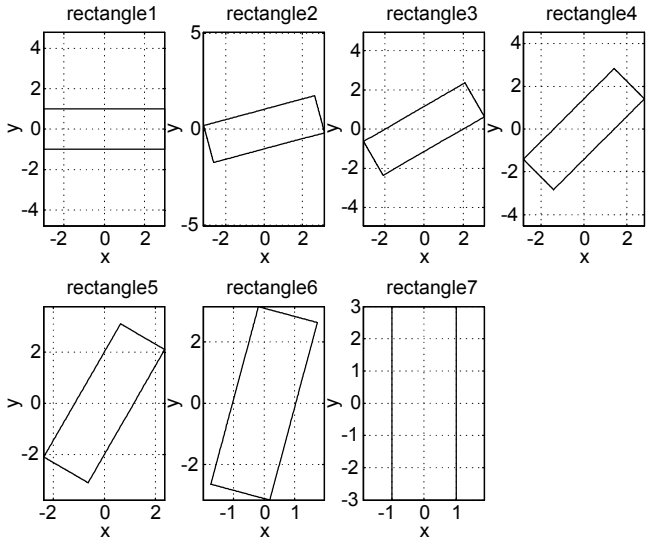

Figure 4.12: The test rectangles, without noise.

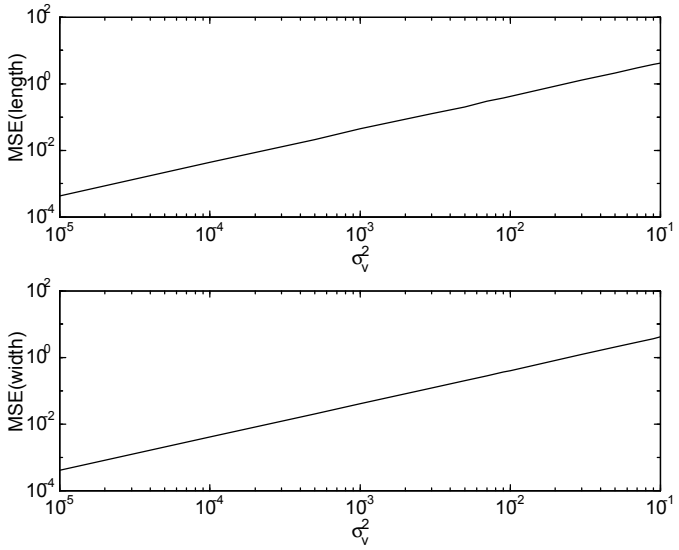

Figure 4.13: MSE of length (upper) and width (lower) as a function of error variance σ_v^2 for rectangle4 in Figure 4.12.

4.7 Conclusions

We have proposed a method for fitting a rectangle as good as possible to object data, assuming that data is error free. As only object data, and no ground data, are used we get an estimate of the minimum area of the object. The method finds a set of four vertices and a normal vector, i.e., an orientation, that minimizes the area of a rectangle that contains the convex hull of the object's data points. From the area estimate it is possible to receive estimates of the length and width of the object. Simulations showed that the mean square errors of the length and width estimates, respectively, were similar. The minimum rectangle estimation process consists of calculation of the convex hull and a rectangle minimization. Each step can be performed in linear time and thus, the total execution time requires $\mathrm{O}(N)$.

It can be noted that also a fifth point is given in the minimum rectangle estimation. As the minimum is found for either $\phi = \phi_{\min}$ or for $\phi = \phi_{\max}$, one side of the rectangle will be parallel with an edge of the convex hull. For that edge only one of the vertices is P^1, P^2, P^3 or P^4, while the other vertex is the extra fifth point.

The present algorithm does not check if two different sets of P^1, P^2, P^3, P^4 and ϕ represent the same area. For example, the rectangles in Figure 4.11a and b have almost identical area (differs in the 14th decimal). The rectangle in Figure 4.11a has orientation $\phi = \phi_{5,\max}$ and the rectangle in Figure 4.11b has orientation $\phi = \phi_{6,\min}$. The angle intervals form a non-overlapping inclusive decomposition of the interval $[0, 2\pi]$, which gives that $\phi_{5,\max} = \phi_{6,\min}$. This kind of duplicates can be avoided by adding an extra check in step 3 of Algorithm 2, before the set of points and orientation is saved in the list. This may, however, extend the execution time.

5

Improving the rectangle estimate

5.1 Introduction

In the previous chapter we proposed a method for fitting a rectangle as good as possible to object data. The best fit was defined as the minimum area containing the convex hull of the outer object points and moreover, this definition includes the assumption that data is error free. In the calculation only some of the available data points were used; the objects points belonging to the convex hull. The available ground points were not used at all. An estimate based on object points only will always under-estimate the rectangle. This can be seen in Figure 5.1, the area estimate is a bit too small when ground data also is taken into account.

We will now improve the area estimate using least squares methods based on all outer object and ground points. The least squares methods are designed to minimize the orthogonal distance between a geometric shape, in this case a rectangle, and the data set. By including the ground points in the minimization we will receive a less biased estimate of the rectangle. There are both linear and nonlinear least squares methods, with the difference whether the equations for a rectangle describes a linear or a nonlinear relation between the parameters. It turns out that a rectangle can be described either by linear equations with a constraint or by nonlinear equations. Furthermore, both methods model each side of the rectangle as a submodel. This means that before we use those methods we need to associate each data point to one of the rectangle's sides.

In this chapter we start by describing the proposed data association method. Then some linear and nonlinear least squares methods are described and the rectangle estimation problem is modelled both linearly and nonlinearly. The chapter

finishes with some simulations to evaluate the selected least squares methods.

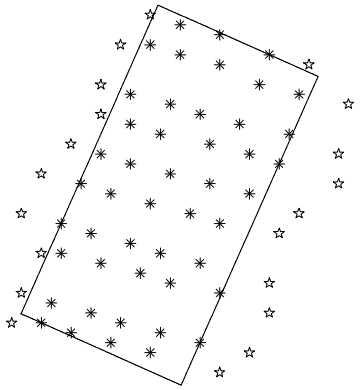

Figure 5.1: The smallest rectangle containing the convex hull, compare with Figure 4.11a. The stars ($*$) represent object data points and the pentagrams ($\star$) ground data points.

5.2 Data association

In the data association we will associate the outer object points and the ground points to one side of the rectangle, as we know that the true object edge is between the outer object points and the ground points. Inner object points will be ignored. If we consider the outer object data we see that there are two kinds: points paired with ground points and points that are not paired with ground points, see Chapter 3 for a description of the data sorting. The data association will differ depending on the type of outer object data. For pairs of outer object and ground points we know that the true object edge lies somewhere between those points. This can be interpreted as that the true edge point lies somewhere on a straight line between the outer object point and the ground point. This line will intersect with one of the rectangle's sides, which is the side that the point pair will be associated with.

For single outer object points, that are *not* paired with ground points, we will associate the points with the side to which they have the smallest orthogonal distance. Those points lie on a certain distance from the sides described by (4.1a)-(4.1d). For a data point $P_i = (x_i, y_i)$ the perpendicular distance to each side can be calculated from

$$\text{Side 1} \quad : \qquad P_i n - c_1 = r_1 \tag{5.1a}$$

$$\text{Side 2} \quad : \qquad P_i \widetilde{n} - c_2 = r_2 \tag{5.1b}$$

$$\text{Side 3} \quad : \qquad P_i n - c_3 = r_3 \tag{5.1c}$$

$$\text{Side 4} \quad : \qquad P_i \widetilde{n} - c_4 = r_4, \tag{5.1d}$$

where n, c_1, c_2, c_3 and c_4 represent the minimum rectangle from (4.6) and $\widetilde{n} = Rn$ is the rotated normal vector in (4.1a)-(4.1d). Each point is associated to the side to which it has the shortest orthogonal distance $|r_j|$, $j = 1, 2, 3, 4$. Results of data association for data in Figure 4.11a is shown in Figure 5.2.

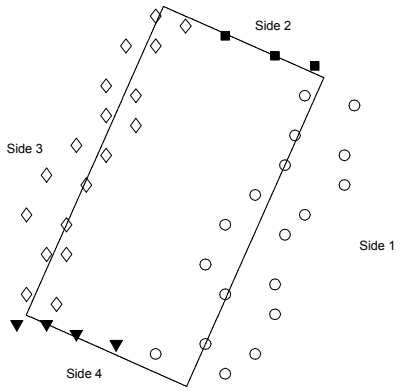

Figure 5.2: Data association based on the rectangle in Figure 4.11a. Outer object and ground points are viewed. Points associated with side 1 are marked with circles, side 2 with filled squares, side 3 with diamonds and side 4 with filled triangles.

5.3 Some linear least squares methods

We will now describe some linear least squares methods; the ordinary least squares (OLS), the total least squares (TLS) and the mixed LS-TLS. Note that in this thesis the traditional/ordinary least squares methods will be called OLS instead of LS, the term LS is used for least squares methods in general. The properties of OLS, TLS and mixed LS-TLS in the presence of errors in both coordinates will be studied.

The general formulation of the problem is to solve the overdetermined system of linear equations

$$A\theta \approx b, \tag{5.2}$$

where A has dimensions $(N \times M)$, b $(N \times 1)$ and θ $(M \times 1)$, $N \geq M + 1$. The estimate of θ, $\hat{\theta}$, is computed differently in the different methods. The difference lies in assumptions of noise contents in (parts of) A and/or b. If A contains errors the matrix will be modelled as an error free part and as a part that contain errors, i.e., $A = A_0 + e_A$, where A_0 is the true but unobservable variable and e_A is the error. If b contains errors this will be modelled as $b = b_0 + e_b$, where b_0 is the true but unobservable variable and e_b is the error.

Consider the case of straight line estimation, see Figure 5.3. If we compare OLS, which assumes that A is error free, and TLS, which assumes that A and b contain the same amount of noise, we see that the methods minimize different distances. In OLS the distance between the points and the line are minimized only in $b = y$ while in TLS (and also in mixed LS-TLS) the orthogonal distance between the points and the line is minimized.

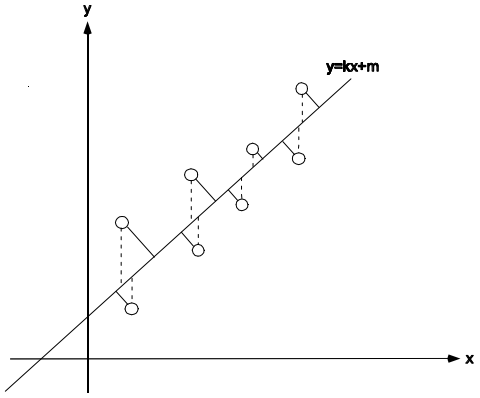

Figure 5.3: The distances that are minimized in the OLS method (dashed) and in the TLS and mixed LS-TLS methods (solid), respectively.

5.3.1 Ordinary least squares

In the ordinary least squares (OLS) model, only b is subject to error and the problem is modelled as

$$b_0 + e_b = A\theta. \tag{5.3}$$

In [12], Section 2.3, the following definition of the ordinary least squares problem is given:

Definition 5 (Ordinary least squares problem)
Given an overdetermined set of N linear equations $A\theta \approx b$ in M unknown θ, the ordinary least squares (OLS) problem seeks to

$$\min_{\hat{b} \in \mathbb{R}^N} \left| b - \hat{b} \right|$$

$$\textit{subject to } \hat{b} \in \mathcal{R}(A).$$

Once a minimizing $\hat{b}$ is found, then any θ satisfying

$$A\theta = \hat{b}$$

is called an OLS solution and $e_b = b - \hat{b}$ the corresponding OLS correction.

When A has full column rank there exists a unique OLS solution that solves the normal equation

$$A^T A\widehat{\theta}_{\text{OLS}} = A^T b.$$

In the OLS model only b is subject to error, i.e., $e_A = 0$. In the case of geometric fitting, however, there will be error also in A ($e_A \neq 0$). It is shown in [12], Theorem 8.4, and in [27] that the OLS parameter estimate is not consistent when applied to geometric fitting, i.e., when the amount of data N increases $\widehat{\theta}_{\text{OLS}}$ will not approach the true value θ_0.

5.3.2 Total least squares

In the total least squares (TLS) model both A and b are observed with errors, i.e., $A = A_0 + e_A$ and $b = b_0 + e_b$, where A_0 and b_0 are the true but unobservable variables and e_A and e_b, respectively, is noise. The problem is now modelled as

$$b_0 + e_b = (A_0 + e_A)\,\theta. \tag{5.4}$$

In [12], Section 2.3, the following definition of the total least squares problem is given:

Definition 6 (Total least squares problem)
Given an overdetermined set of N linear equations $A\theta \approx b$ in M unknown θ, the total least squares (TLS) problem seeks to

$$\min_{[\widehat{A};\hat{b}]\in\mathbb{R}^{N\times(M+1)}} \left\| [A;b] - \left[\widehat{A};\hat{b}\right] \right\|_F$$

$$\textit{subject to } \hat{b} \in \mathcal{R}\left(\widehat{A}\right).$$

Once a minimizing $\left[\widehat{A};\hat{b}\right]$ is found, then any θ satisfying

$$\widehat{A}\theta = \hat{b}$$

is called a TLS solution and $[e_A; e_b] = [A;b] - \left[\widehat{A};\hat{b}\right]$ the corresponding TLS correction.

In this formulation all columns in $[A;b]$ contain errors. If some of the columns in A are known to be error free the LS and TLS methods can be mixed, see next subsection.

5.3.3 Mixed LS-TLS

In the mixed LS-TLS model some columns of A are subject to errors, while some are known exactly. A is described as $A = [A_1; A_2]$, where A_1 is error free and $A_2 = A_{2,0} + e_{A_2}$. The problem is modelled as

$$b_0 + e_b = A_1\theta_1 + (A_{2,0} + e_{A_2})\,\theta_2. \tag{5.5}$$

In [12], Section 3.5, the following definition of the mixed LS-TLS problem is given:

Definition 7 (Mixed LS-TLS problem)
Given a set of N linear equations in M unknown θ:

$$A\theta \approx b, \qquad A \in \mathbb{R}^{N\times M},\ b \in \mathbb{R}^{N\times 1},\ \theta \in \mathbb{R}^{M\times 1}$$

Partition

$$\begin{aligned} A &= [A_1; A_2], & A_1 &\in \mathbb{R}^{N\times M_1},\ A_2 \in \mathbb{R}^{N\times M_2} \\ \theta &= [\theta_1; \theta_2]^T, & \theta_1 &\in \mathbb{R}^{M_1\times 1},\ \theta_2 \in \mathbb{R}^{M_2\times 1} \end{aligned}$$

and assume that the columns of A_1 are known exactly and $M = M_1 + M_2$. Then, the mixed LS-TLS problem seeks to

$$\min_{\left[\hat{A}_2;\hat{b}\right]\in\mathbb{R}^{N\times(M_2+1)}} \left\| [A_2; b] - \left[\widehat{A}_2; \hat{b}\right] \right\|_F$$

$$\textit{subject to } \mathcal{R}\left(\hat{b}\right) \subseteq \mathcal{R}\left(\widehat{A}\right) = \mathcal{R}\left(\left[A_1; \widehat{A}_2\right]\right).$$

Once a minimizing $\left[\widehat{A}_2; \hat{b}\right]$ is found, then any $\theta = [\theta_1; \theta_2]^T$ satisfying

$$\widehat{A}\theta = A_1\theta_1 + \widehat{A}_2\theta_2 = \hat{b},$$

is called a mixed LS-TLS solution and $[e_{A_2}; e_b] = [A_2; b] - \left[\widehat{A}_2; \hat{b}\right]$ is the corresponding mixed LS-TLS correction.

When the solution is not unique, the minimum norm solution can be singled out, see [12], Chapter 3. This is a general form of the linear least squares problem. If all columns in A are known exactly, i.e., $A = A_1$, the mixed LS-TLS problem reduces to OLS. If no columns in A are known exactly, i.e., $A = A_2$, it reduces to TLS. A mixed LS-TLS algorithm is described in [12], Section 3.6, and [7]:

Algorithm 3 (Mixed LS-TLS for $d = 1$)

Given:

An $N \times M$ data matrix A with M_1 exactly known linearly independent columns and $0 < M_1 < M$,
an $N \times 1$ observation vector b,
an M-dimensional vector t indicating whether the i:th column of A is known exactly ($t_i = 0$) or not ($t_i = 1$), $i = 1, ..., M$.

Step 1: Column permutations in A

If $0 < M_1 < M$, permute the columns A of such that

$$AP = [A_1; A_2], \qquad M = M_1 + M_2,$$

where P is an $M \times M$ permutation matrix and A_1 contains the M_1 exactly known columns in A.

Step 2: QR factorization of $[A_1; A_2; b]$

If $M_1 > 0$ then compute the QR factorization:

$$[A_1; A_2; b] = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix},$$

where Q is orthogonal and R_{11} $(M_1 \times M_1)$ is upper triangular,
$R_{12} : M_1 \times (M_2 + 1)$, $R_{22} : (N - M_1) \times (M_2 + 1)$.
If $M_1 = M$ then calculate the OLS solution $\hat{\theta}$
solve $R_{11}\hat{\theta} = R_{12}$ by back-substitution.

else

$R_{22} \leftarrow [A_1; A_2; b]$

end

Step 3: Singular value decomposition (SVD) of R_{22}

$U^T R_{22} V = \mathrm{diag}(\sigma_1, \ldots, \sigma_s), \qquad s = \min(N - M_1, M_2 + 1),$
with U, $V = [v_1, \ldots, v_{M_2+1}]$ orthonormal, $\sigma_{i-1} \geq \sigma_i$ for $i = 2, .., s$ and $\sigma_j = 0$ for $j > s$.

Step 4: Rank determination

If not user-determined, compute the numerical rank r $(\leq M_2)$ of $[A_2; b]$:

$$\sigma_1 \geq ... \geq \sigma_r > R_\nu \geq \sigma_{r+1} \geq \sigma_{M_2+1},$$

with R_ν an appropriate rank determinator.

Step 5: Solution space V_2

If $M_1 > 0$ then

$V_{22} \leftarrow [v_{r+1}, \cdots, v_{M_2+1}]$
solve $R_{11}V_{12} = -R_{12}V_{22}$ by back-substitution

$$V_2 \leftarrow \begin{bmatrix} V_{12} \\ V_{22} \end{bmatrix}, \qquad V_{12} : M_1 \times 1, V_{22} : M_2 + 1 \times 1.$$

else

$V_2 \leftarrow [v_{r+1}, \cdots, v_{M_2+1}]$

end

Step 6: TLS solution $\hat{\theta}$

Compute with Householder transformations the orthogonal matrix Q such that

$$V_2 Q = \begin{bmatrix} Y & Z \\ 0 & \Gamma \end{bmatrix},$$

$Y : M \times M_2 - r$, $Z : M \times 1$, $\Gamma : 1 \times 1$
If $\Gamma = 0$ then
{TLS solution is nongeneric}
lower the rank r with the multiplicity of σ_r
go back to step 5.
else
solve by forward elimination: $\hat{\theta}\Gamma = -Z$.
end

Step 7: Inverse row permutation in $\hat{\theta}$

If $0 < M_1 < M$, undo the permutations P: $\hat{\theta} \leftarrow P\hat{\theta}$.

5.3.4 Selected method

In the estimation of a rectangle only some columns in A are subject to errors, see Rectangle model 1 on page 10. Thus, the only suitable linear LS method is the mixed LS-TLS method. In this method it cannot be guaranteed that a unique solution always is calculated, but then the minimum norm can be singled out.

It can be shown that in the mixed LS-TLS method the parameter estimates are consistent if both e_{A_2} and e_b are known, or if the relation between e_{A_2} and e_b is known [4], [12]. Further, if $A_{2,0}$, e_{A_2} and e_b are jointly normal distributed the parameter estimates are maximum likelihood (ML) estimates [4].

5.4 Some nonlinear least squares methods

This description of nonlinear least squares methods follows Dennis and Schnabel [5], Chapter 9-10. The nonlinear least squares problem is

$$\min_{\theta \in \mathbb{R}^M} f(\theta) = \frac{1}{2} \sum_{i=1}^{N} r_i^2(\theta) = \frac{1}{2} R^T(\theta) R(\theta),$$

where $N > M$, the residual function $R : \mathbb{R}^M \rightarrow \mathbb{R}^N$ is nonlinear in θ and $r_i(\theta)$ denotes the i:th component function of $R(\theta)$. In geometric fitting $r_i(\theta)$ represents the orthogonal distance between the data point $(x_i; y_i)$ and the rectangle. The criterion function $f(\theta)$ is minimized iteratively and this can be performed using different methods. When deriving the nonlinear least squares methods below, we will need the derivatives of $R(\theta)$ and $f(\theta)$. We start with deriving them before we describe the specific methods. The first derivative matrix of $R(\theta)$ is the Jacobian matrix $J(\theta) \in \mathbb{R}^{N \times M}$, where

$$J(\theta)_{ij} = \frac{\partial r_i}{\partial \theta_j}.$$

The first derivative (or gradient) of $f(\theta)$ is

$$\nabla f(\theta) = \sum_{i=1}^{N} r_i(\theta) \cdot \nabla r_i(\theta) = J^T(\theta) R(\theta)$$

and the second derivative (or Hessian) is

$$\nabla^2 f(\theta) = \sum_{i=1}^{N} \left(\nabla r_i(\theta) \cdot \nabla r_i^T(\theta) + r_i(\theta) \cdot \nabla^2 r_i(\theta) \right) = J^T(\theta) J(\theta) + S(\theta),$$

where

$$S(\theta) = \sum_{i=1}^{N} r_i(\theta) \cdot \nabla^2 r_i(\theta).$$

This can be summarized in the iterative Newton method

$$\begin{aligned} \theta_{t+1} &= \theta_t - \Delta\theta \\ &= \theta_t - \left(\nabla^2 f(\theta) \right)^{-1} \nabla f(\theta) \\ &= \theta_t - \left(J^T(\theta_t) J(\theta_t) + S(\theta_t) \right)^{-1} J^T(\theta_t) R(\theta_t), \\ t &= 1, 2, \ldots \end{aligned}$$

The problem is that $S(\theta)$ usually is unavailable, inconvenient to obtain or too expensive to approximate by finite differences. We will therefore study three approximations of the Newton method: the Gauss-Newton method, the Gauss-Newton method with regularization and the quasi-Newton method.

5.4.1 Gauss-Newton

In the Gauss-Newton method it is assumed that $J^T(\theta_t) J(\theta_t) \gg S(\theta_t)$ or that $\nabla^2 f(\theta) \approx J^T(\theta) J(\theta)$. Let us assume that $J(\theta_t)$ have full column rank, we then have the iterative Gauss-Newton method

$$\theta_{t+1} = \theta_t - \left(J^T(\theta_t) J(\theta_t) \right)^{-1} J^T(\theta_t) R(\theta_t).$$

The Gauss-Newton method have some advantages and disadvantages:

Advantages

1. Locally quadratically convergent on zero-residual problems.
2. Quickly locally linearly convergent on problems that are not too nonlinear and have reasonably small residuals.
3. Solves linear least-squares problems in one iteration.

Disadvantages

1. Slowly locally linearly convergent on problems that are sufficiently nonlinear or have reasonably large residuals.
2. Not locally convergent on problems that are very nonlinear or have very large residuals.
3. Not well defined if $J(\theta_t)$ does not have full column rank.
4. Not necessarily globally convergent.

5.4.2 Regularized Gauss-Newton

The step size $\Delta\theta$ is well-defined and in the descent direction if $J(\theta_t)$ has full column rank and if $J^T(\theta_t)J(\theta_t)$ is non-singular and positive definite. This may not always be the case during an iteration process. To overcome this problem some kind of regularization technique can be used. A common improvement of the Gauss-Newton algorithm is to choose θ_{t+1} by a trust region approach (also called Levenberg-Marquardt regularization)

$$\theta_{t+1} = \theta_t - \left(J^T(\theta_t)J(\theta_t) + \mu_t I\right)^{-1} J^T(\theta_t)R(\theta_t),$$

where $\mu_t = 0$ if $\delta_t \geq \left|\left(J^T(\theta_t)J(\theta_t)\right)^{-1} J^T(\theta_t)R(\theta_t)\right|$ and $\mu_t > 0$ otherwise. In this method $S(\theta)$ is approximated by $\mu_t I$. Different strategies of selecting values on μ_t and δ_t is described in [5]. The regularized Gauss-Newton method may be slowly locally convergent on large residual or very nonlinear problems.

5.4.3 Quasi-Newton

The quasi-Newton method is more complex than the Gauss-Newton method, but can on the other hand handle problems that contain more nonlinearities and larger residuals. The difference lies in that $S(\theta)$ is approximated with a more complex function than in the Gauss-Newton methods. In one implementation of the quasi-Newton method $\nabla^2 f(\theta)$ is approximated by

$$\begin{aligned} H_{t+1} &= H_t + \frac{y_t y_t^T}{y_t^T s_t} - \frac{H_t s_t s_t^T H_t}{s_t^T H_t s_t}, \\ \text{where } y_t &= \nabla f(\theta_{t+1}) - \nabla f(\theta_t), \\ s_t &= x_{t+1} - x_t, \\ x_{t+1} &= x_t - H_t^{-1}\nabla f(\theta_t). \end{aligned}$$

This is the positive definite secant method (also called BFGS after its discoverer Broyden, Fletcher, Goldfarb and Shanno), but it will be called the quasi-Newton method in this thesis. According to the authors [5], the positive definite secant method is only a little slower than Newton's method and it can be superior to the Gauss-Newton method on medium- and large-residual problems. It may be quickly locally convergent on examples for which the Gauss-Newton method do not converge at all. This method have proven strong convergence results when it is applied to a strictly convex function, see Theorem 9.5.1 in [5].

5.4.4 Initial values of the parameters

As noted in [6], the initial value of θ is crucial for convergence to the true values (and not to a local minimum) and that when possible, linear least squares techniques can be used for retrieving initial values of the parameters.

Ljung [18] notes that it is worth some effort on producing good initial values of θ. Both the Gauss-Newton and the quasi-Newton methods have good local convergence rates, but not necessarily fast convergence far from the minimum. Good initial values, i.e., that are reasonable close to the true values, usually pays off in fewer iterations and shorter total computing time. Ljung suggests that for physically parametrized models physical insight shall be used to provide reasonable initial values.

5.4.5 Selected methods

We have shortly described three different methods for iterative minimization of a nonlinear least squares problem; the Gauss-Newton method, the regularized Gauss-Newton method and the quasi-Newton method. The simulations in [6] showed that if the initial values of the parameters are reasonable, all three methods converge when fitting a circle or an ellipse.

The equation of a rectangle, however, contains discontinuities and more nonlinear elements compared to the equation of a conic. It is hard to say whether the rectangle estimation problem will be too nonlinear or if the residuals are too large for the Gauss-Newton and regularized Gauss-Newton methods. If the Gauss-Newton method shall be applicable $J^T(\theta)J(\theta)$ must be non-singular. When we have evaluated the equation of the rectangle we will check whether $J^T(\theta)J(\theta)$ is non-singular or singular. If it is possible to calculate $H(\theta)$ analytically it is also interesting to test the (complete) Newton method.

None of the methods can guarantee global convergence. The convergence and speed of convergence depend on the nonlinearities in $f(\theta)$, the size of the residuals and how close the initial parameters are to the minimizing parameters. Therefore, we will perform tests on all four methods, i.e., the Gauss-Newton method, the regularized Gauss-Newton method, the quasi-Newton method and the Newton method.

5.5 Rectangle fitting using least squares methods

5.5.1 Constrained linear least squares

In Section 2.3 Rectangle model 1 was described, see page 10:

$$\begin{aligned} \text{side 1} &: c_1 + n_1 x + n_2 y = 0 \\ \text{side 2} &: c_2 - n_2 x + n_1 y = 0 \\ \text{side 3} &: c_3 + n_1 x + n_2 y = 0 \\ \text{side 4} &: c_4 - n_2 x + n_1 y = 0 \\ \text{and} &: n_1^2 + n_2^2 = 1, \end{aligned}$$

where the normal vector $(n_1; n_2)$ is orthogonal to side 1 and side 3 of the rectangle, the normal vector $(-n_2; n_1)$ is orthogonal to side 2 and side 4 of the rectangle and c_i is the perpendicular distance between side i and the origin, $i = 1, 2, 3, 4$.

Divide the data set $(x_i; y_i), i = 1, ..., N$, between the four sides of the rectangle so that $(x_{1,s}; y_{1,s}), s = 1, ..., N_1$, belongs to side 1, $(x_{2,t}; y_{2,t}), t = 1, ..., N_2$, belongs to side 2, $(x_{3,u}; y_{3,u}), u = 1, ..., N_3$, belongs to side 3 and $(x_{4,v}; y_{4,v}), v = 1, ..., N_4$, belongs to side 4, respectively. In this case the data set consists of the outer object and ground points and the division into the different sides of the rectangle will be made using the data association approach described in Section 5.2. We now have the following constrained least squares problem:

$$\min_{\theta \in \mathbb{R}^M} |r|^2 = \min_{\theta \in \mathbb{R}^M} \sum_{i=1}^{N} r_i^2$$

subject to

$$\begin{pmatrix} 1 & 0 & 0 & 0 & x_{1,1} & y_{1,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & x_{1,N_1} & y_{1,N_1} \\ 0 & 1 & 0 & 0 & y_{2,1} & -x_{2,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & y_{2,N_2} & -x_{2,N_2} \\ 0 & 0 & 1 & 0 & x_{3,1} & y_{3,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & x_{3,N_3} & y_{3,N_3} \\ 0 & 0 & 0 & 1 & y_{4,1} & -x_{4,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & y_{4,N_4} & -x_{4,N_4} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{N_1+N_2+N_3+N_4} \end{pmatrix}$$

$$\text{and } n_1^2 + n_2^2 = 1,$$
$$N_1 + N_2 + N_3 + N_4 = N,$$

where $\theta = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & n_1 & n_2 \end{bmatrix}^T$. The suitable algorithm for this problem is the mixed LS-TLS algorithm, see Section 5.3.3, where A_1 consists of the four first columns of the large matrix, A_2 consists of the fifth column of the large matrix and b consists of the sixth column of the large matrix, $\theta_1 = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 \end{bmatrix}^T$ and $\theta_2 = \begin{bmatrix} n_1 & n_2 \end{bmatrix}^T$. An explanation of how the constraint $n_1^2 + n_2^2 = 1$ is included in the minimization is given in Appendix A.

5.5.2 Nonlinear least squares

Let us study Rectangle model 2, see page 10. The four angles $\alpha_1, \alpha_2, \alpha_3$ and α_4 can be replaced with one angle α and (x_c, y_c) can be replaced with a point (x_0, y_0), which is the center of gravity of the rectangle. Then the equations of the rectangle

becomes

$$\text{side 1} \quad : \quad r_s = -(x_s - x_0)\sin\alpha + (y_s - y_0)\cos\alpha - \frac{W}{2} \tag{5.6a}$$

$$\text{side 2} \quad : \quad r_t = -(x_t - x_0)\cos\alpha - (y_t - y_0)\sin\alpha - \frac{L}{2} \tag{5.6b}$$

$$\text{side 3} \quad : \quad r_u = (x_u - x_0)\sin\alpha - (y_u - y_0)\cos\alpha - \frac{W}{2} \tag{5.6c}$$

$$\text{side 4} \quad : \quad r_v = (x_v - x_0)\cos\alpha + (y_v - y_0)\sin\alpha - \frac{L}{2}, \tag{5.6d}$$

where α is the angle between the x'- and x-axis, (x_0, y_0) is the center of gravity the rectangle, and the length and width of the rectangle is L and W, respectively, see Figure 5.4. This parametrization is called *Rectangle model 3*.

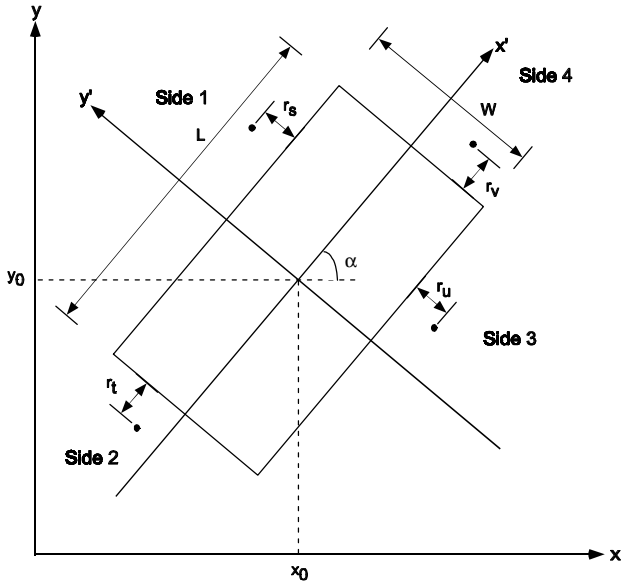

Figure 5.4: Overview of the nonlinear LS minimization problem.

The data set $(x_i; y_i), i = 1, ..., N$, is divided between the four sides of the rectangle so that $(x_s; y_s), s = 1, ..., N_1$, belongs to side 1, $(x_t; y_t), t = 1, ..., N_2$, belongs to side 2, $(x_u; y_u), u = 1, ..., N_3$, belongs to side 3 and $(x_v; y_v), v = 1, ..., N_4$, belongs to side 4, respectively, $N_1+N_2+N_3+N_4 = N$. r_s represents the orthogonal distance between a data point and side 1 of the rectangle, r_t the orthogonal distance between a data point and side 2, r_u the orthogonal distance between a data point and side 3 and r_v the orthogonal distance between a data point and side 4. In this case the data set consists of the outer object and ground points and the division into the different sides of the rectangle is made using the data association approach described in Section 5.2.

For a counter clockwise listing of the sides the residual for each side are expressed as in (5.6*a*)-(5.6*d*). For a clockwise listing of the sides the residual of side 2 is expressed as (5.6*d*) and the residual of side 4 is expressed as (5.6*b*). If equations (5.6*a*)-(5.6*d*) are used anyway for describing a rectangle where the sides are listed clockwise, the result will be that either L or W will have a negative value. Thus, if we study the absolute values of the estimates of L and W the last two constraints in Rectangle model 2, (2.3*h*)-(2.3*i*), are also included in this formulation.

Expressions of $f(\theta), \nabla f(\theta)$ and $J(\theta)$

As each side of the rectangle is described in a separate submodel, each side will have a separate minimization function. Hence, the criterion function $f(\theta)$ will be

$$\min_{\theta\in\mathbb{R}^M} f(\theta) = \min_{\theta\in\mathbb{R}^M} \left[\frac{1}{2}\sum_{s=1}^{N_1} r_s^2(\theta) + \frac{1}{2}\sum_{t=1}^{N_2} r_t^2(\theta) + \frac{1}{2}\sum_{u=1}^{N_3} r_u^2(\theta) + \frac{1}{2}\sum_{v=1}^{N_4} r_v^2(\theta) \right], \quad (5.7)$$

where $\theta = (x_0, y_0, L, W, \alpha)^T$ and the expressions of r_s, r_t, r_u and r_m are given in (5.6*a*)-(5.6*d*). The expressions of $\nabla f(\theta)$, $J^T(\theta)J(\theta)$ and $H(\theta)$ are calculated in Appendix B. The calculations show that $J^T(\theta)J(\theta)$ is singular only when the data set is error free and that $H(\theta)$ can be calculated analytically without too much effort. Thus, both the Gauss-Newton method and the Newton method are worth to be tested on the rectangle estimation problem.

Initial values

The initial values of θ are retrieved from the minimum rectangle estimation, see Section 4.5, where $x_0^{\text{start}} = y_0^{\text{start}} = 0$, L^{start} and W^{start} is the estimated length and width, respectively, and α^{start} equals the ϕ that corresponds to the minimal area.

Returned estimates

The returned estimates will be $\hat{\theta} = \left(\hat{x}_0, \hat{y}_0, \hat{L}, \hat{W}, \hat{\alpha}\right)^T$ if data are listed counter clockwise. When the sides are listed clockwise either $\hat{L}$ or $\hat{W}$ will be negative. To be able to handle both clockwise and counter clockwise listing of the sides the final estimate of the parameters is returned as $\hat{\theta} = \left(\hat{x}_0, \hat{y}_0, \left|\hat{L}\right|, \left|\hat{W}\right|, \hat{\alpha}\right)^T$.

5.6 The size and orientation estimation algorithm

The complete rectangle estimation process, i.e., the size and orientation estimation process, can be described as an algorithm:

Algorithm 4 (Rectangle estimation process)
Given: *Data of an object (delivered by the previously performed segmentation process).*
Step 1: *Sort data using Algorithm 1.*
Step 2: *Estimate the minimum rectangle using Algorithm 2 and outer object data.*
Step 3: *Associate all outer object and ground data to one of the sides of the rectangle using the method described in Section 5.2.*
Step 4: *Improve the rectangle estimate using an LS algorithm.*

So far, we cannot say whether the appropriate LS algorithm is the linear, constrained LS-TLS algorithm or one of the nonlinear LS algorithms. If the appropriate algorithm is one of the Newton methods (with or without approximations), the size and orientation estimates calculated in step 3 will be used as initial parameter values.

5.7 Performance

We will now investigate the performance of these LS rectangle estimation methods in some Monte Carlo simulations. The data association is performed using the method described in Section 5.2 and the initial values of the nonlinear LS methods are the estimates of the minimum rectangle estimator. We will in principle perform Algorithm 4, but on outer object data only (i.e., $\text{pair}_{\text{ground}} = \text{nil}$). The tests are performed on the same data set as in Section 4.6. Each side of the rectangle is represented by 1000 data points, i.e., $N_1 = N_2 = N_3 = N_4 = 1000$. The performance is studied with respect to correctness in length and width estimates, use of CPU time and number of iterations (for the nonlinear LS methods).

The used LS-TLS algorithm is from [7] (`clsq.m`). That algorithm differs from Algorithm 3 in this thesis in that the normalization in step 6 is not performed, because the result of step 5, V_2, fulfils

$$V_2 = \begin{bmatrix} V_{12} \\ V_{22} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & n_1 & n_2 \end{bmatrix}^T.$$

The nonlinear algorithms used are the Matlab functions for the Gauss-Newton method (`lsqnonlin.m` with $J(\theta)$ given), regularized Gauss-Newton method (`lsqnonlin.m` with the Levenberg-Marquardt method selected and $J(\theta)$ given), quasi-Newton (`fminunc.m` with Hessian update method equal to BFGS, line search procedure equal to cubic polynomial and $\nabla f(\theta)$ given) and Newton (`fminunc.m` with line search procedure equal to cubic polynomial and $\nabla f(\theta)$ and $H(\theta)$ given).

For all methods the termination tolerance on the function value and the termination tolerance on θ were set to 10^{-4}.

The linear and nonlinear LS methods will be compared in their computational efficiency. The goal is to study if there is a trend that one of the methods in general is faster then the others, i.e., a comparison of relative CPU usage will be performed. The absolute values of CPU time usage depend on the computer and is not interesting. The computational efficiency will be studied by logging the CPU time used by Matlab to perform the calculations (command `cputime.m`). For the linear LS method the CPU time for the call to `clsq.m` will be logged and for the nonlinear LS methods the CPU time for the call to `lsqnonlin.m` and `fminunc.m`, respectively, will be logged.

5.7.1 Results

During the tests the Gauss-Newton algorithm often warned that $J^T(\theta)J(\theta)$ was near singular and that the result may be inaccurate. The results of MSE of length and width estimates confirms the problem with the Hessian approximation of the Gauss-Newton method, see Figure 5.5. The rectangle estimation problem is either too nonlinear or the residuals too large (or both) for this method. The Gauss-Newton method is classified as not applicable to this problem.

During the tests of the regularized Gauss-Newton method the algorithm warned a few times that $J^T(\theta)J(\theta)$ was near singular, but the estimates of length and width are, in MSE sense, almost as good as for the remaining methods, see Figure 5.5.

The best estimates, in MSE sense, of length and width are given by the LS-TLS, quasi-Newton and Newton algorithms. Those three methods perform about equal. The MSE for the estimates of length and width are smaller, about 10^4 times smaller, than those of the minimum rectangle estimator, see Section 4.6. Results of length and width estimates for these three algorithms are also shown in Figure 5.5.

For the nonlinear LS methods it is also interesting to study how many iterations, on the average, each method needs to find a minimum, see Figure 5.6. The mean number of iterations decreases with the knowledge of the Hessian. The regularized Gauss-Newton method, that have the simplest approximation of the Hessian, needs about 20-60 iterations to reach the minimum. The quasi-Newton method, that have a more complex approximation of the Hessian, needs about 13-20 iterations to reach the minimum. Finally, the Newton method, that uses the analytical expression of the Hessian, needs 6-5 iterations to reach the minimum.

A drawback of the Newton method is that there are many calculations involved to calculate the Hessian. In Figure 5.7 the mean of CPU usage for the LS-TLS, regularized Gauss-Newton, quasi-Newton and Newton algorithms are shown. The cost of the calculation of the Hessian is remarkable, as the quasi-Newton and Newton methods needs about the same amount of CPU time to reach the minimum. The fastest algorithm is the LS-TLS algorithm, which uses about 10 times less CPU time than the best nonlinear LS algorithm!

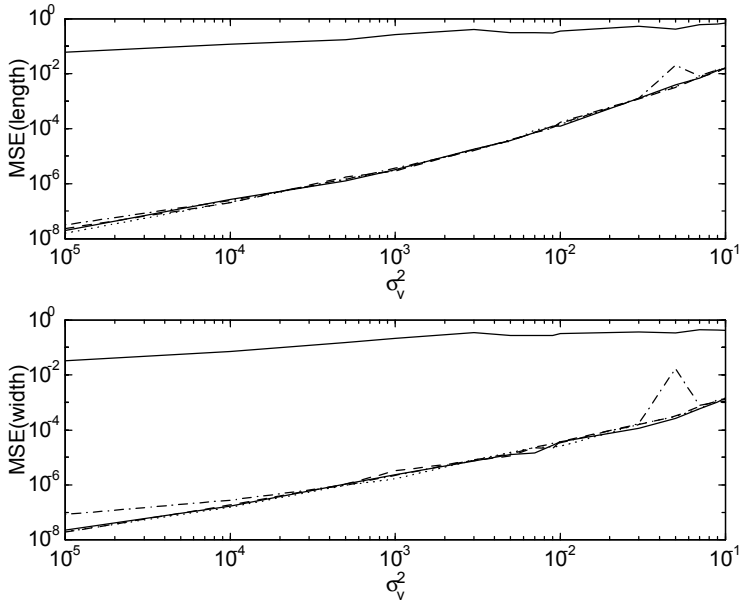

Figure 5.5: MSE of length (upper) and width (lower) as a function of error variance σ_v^2 for rectangle4 in Figure 4.12. Lower solid: LS-TLS, dashed: Newton, dotted: quasi-Newton, dash-dotted: regularized Gauss-Newton, upper solid: Gauss-Newton. The lines of LS-TLS, Newton and quasi-Newton are very close to each other.

5.8 Conclusions

In this chapter we proposed a method for association of outer object and ground data to the different sides of the rectangle. Further, we proposed two ways of improving the rectangle estimate; a linear, constrained LS method (mixed LS-TLS) and a nonlinear LS method (quasi-Newton).

Several LS methods were tested in simulations. The fastest method was the linear, constrained LS method which needed 10 times less CPU usage than any of the nonlinear LS methods. Of the nonlinear LS methods, the quasi-Newton and Newton methods performed about equal in sense of CPU usage. All three methods, i.e. the mixed LS-TLS, the quasi-Newton and Newton methods, returned estimates that, concerning MSE, were about 10^4 times smaller than those of the minimum rectangle estimator.

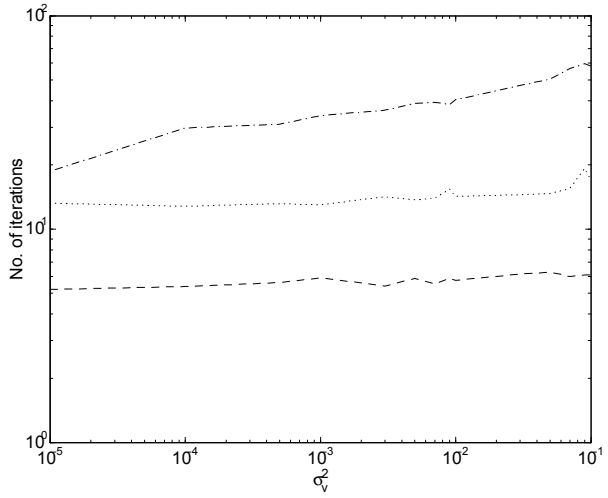

Figure 5.6: Mean number of iterations of the nonlinear LS methods as a function of error variance σ_v^2 for rectangle4 in Figure 4.12. Dashed: Newton, dotted: quasi-Newton, dash-dotted: regularized Gauss-Newton.

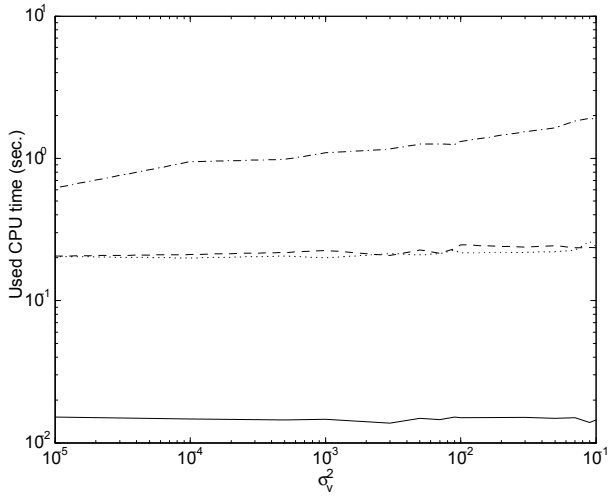

Figure 5.7: Mean CPU usage as a function of error variance σ_v^2 for rectangle4 in Figure 4.12. Solid: LS-TLS, dashed: Newton, dotted: quasi-Newton, dash-dotted: regularized Gauss-Newton.

6

Final tests

In the previous chapters only parts of the rectangle estimation process (Algorithm 4) have been evaluated. It is now time to test the complete rectangle estimation process. We will do tests both on simulated, realistic data and on real data collected with the laser radar system.

6.1 Tests on simulated, realistic data

The performance of the rectangle estimation process will be investigated in some Monte Carlo simulations. We will perform Algorithm 4 but in step 4 both the LS-TLS algorithm and the quasi-Newton algorithm will be tested. The Newton algorithm will not be studied further, as it is similar to the quasi-Newton algorithm in terms of CPU time usage. The performance of the algorithms (minimum rectangle, LS-TLS and quasi-Newton) will, again, be tested in terms of correctness in length and width estimates and CPU usage. We will also check whether the LS methods return less biased estimates than the minimum rectangle estimator. The tests are performed in the same manner as the earlier tests, see Section 4.6 and Section 5.7, but on a more realistic data set, see Figure 6.1. The outer objects points are placed with equal distance (0.4) around the object, which means that there are more data on the longer sides than what there are on the shorter sides of the rectangle. The ground points are placed with equal distance (0.4), both relative to the outer object points and to the other ground points, along the longer sides of the object.

We start with an exact (error free) description of the rectangle. Random errors, normally distributed with zero mean and variance σ_v^2 are added to the coordinates x

and y, respectively. The noise is generated separately for the x and y coordinates. The noise can therefore be considered independently and identically distributed with $\sigma^2_{e_x} = \sigma^2_{e_y} = \sigma^2_v$. The lengths and widths are estimated using the minimum rectangle, mixed LS-TLS and quasi-Newton algorithms on the perturbed data set. The simulations are repeated for increasing noise variance σ^2_v. The properties of the length and width estimates are studied in MSE, see (4.7), and in squared bias sense. The squared bias is defined as (see [12], page 244)

$$\text{squared bias}\left(\hat{\theta}\right) = \left(E\left(\hat{\theta}\right) - \theta_0\right)^T \left(E\left(\hat{\theta}\right) - \theta_0\right)$$

and estimated by

$$\text{squared bias}\left(\hat{\theta}\right) = \left\{\left(\frac{1}{J}\sum_{j=1}^{J}\hat{\theta}_j\right) - \theta_0\right\}^2 .$$

The MSE and squared bias are averaged over 100 sets (i.e., $J = 100$).

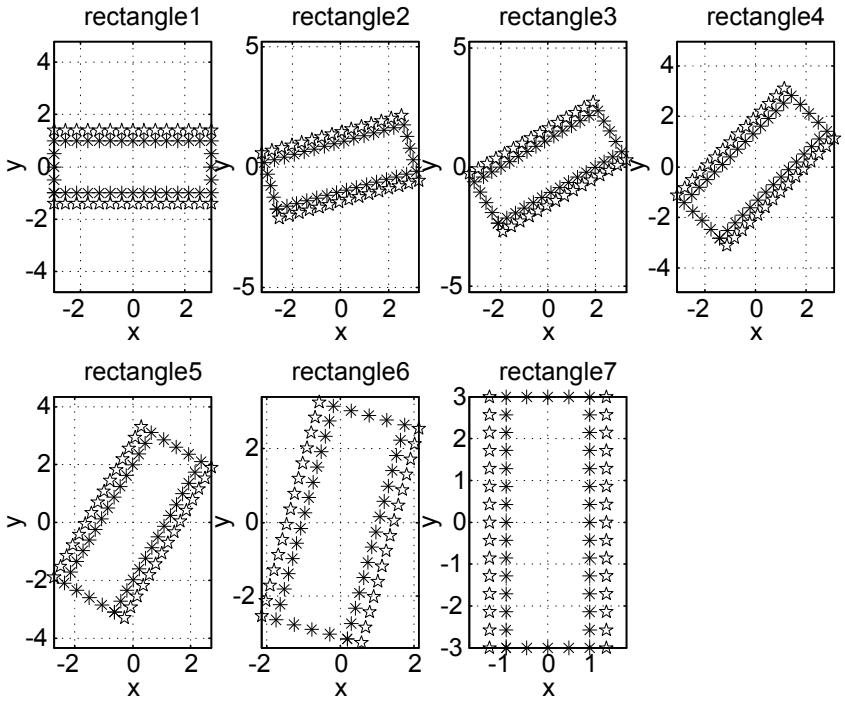

Figure 6.1: The second set of simulated rectangle data, without noise. Stars ($*$) represent outer object points and pentagrams ($\star$) ground data points.

6.1.1 Results

For this type of data the improvement in MSE for using a LS method instead of the minimum rectangle estimator is about $10 - 100$ times. The improvement is smaller than in the previous test, see Section 5.7, as the number of samples is smaller ($N = 70$ instead of $N = 4000$). The different LS methods are, however, comparable even in this simulation. In the beginning of Chapter 5 we claimed that the bias in the estimates would decrease if we used LS methods. The simulations show that the bias decreases when the LS methods are used after of the minimum rectangle estimator. Both LS methods performs similar also in squared bias sense. The MSE and squared bias of the estimates of length and width for rectangle4 in Figure 6.1 are shown in Figure 6.2 and Figure 6.3, respectively. If we study the mean CPU usage of the LS-TLS and quasi-Newton algorithms, see Figure 6.4, we see that the difference between the methods have increased to about 100 times!

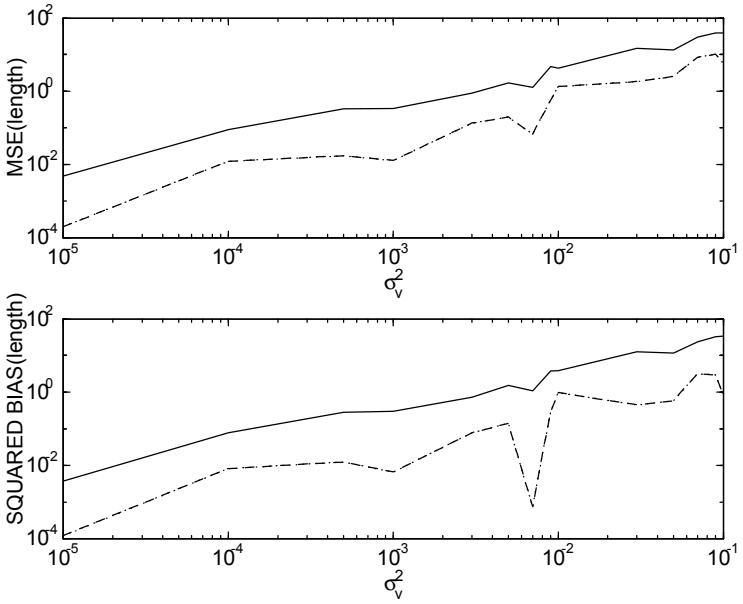

Figure 6.2: MSE (upper) and squared bias (lower) of length as a function of error variance σ_v^2 for rectangle4 in Figure 6.1. Solid: minimum rectangle estimator; dashed: mixed LS-TLS; dotted: quasi-Newton. The LS-TLS and quasi-Newton curves are overlapping.

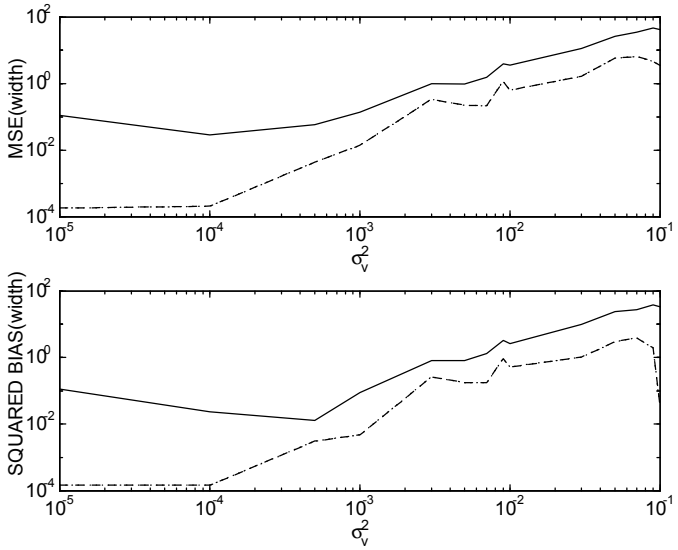

Figure 6.3: MSE (upper) and squared bias (lower) of width as a function of error variance σ_v^2 for rectangle4 in Figure 6.1. Solid: minimum rectangle estimator; dashed: mixed LS-TLS; dotted: quasi-Newton. The LS-TLS and quasi-Newton curves are overlapping.

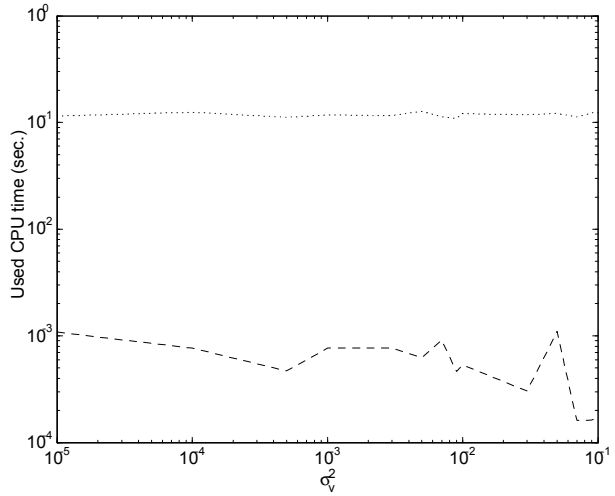

Figure 6.4: Mean CPU usage as a function of error variance σ_v^2 for rectangle4 in Figure 6.1. Dashed: mixed LS-TLS; dotted: quasi-Newton.

6.2 Tests on real data

We also tested the rectangle estimation process (Algorithm 4) on some vehicles measured by the laser radar system. In this case we do not know the true length and width exactly and therefore, we show the estimated rectangles overloaded on the data sets. However, to have some sense of the true lengths and widths of the vehicles we also measured them using a measuring-tape.

In Figure 6.5 and Figure 6.6 two different measurements on a military terrain vehicle are shown, hereafter called terrain vehicle 1 and 2, respectively, and in Figure 6.7 a measurement of a truck is shown. These examples show three different scan directions. For terrain vehicle 1 the scan direction was parallel to the shorter sides and consequently, all ground measurements are placed along the longer sides. For terrain vehicle 2 the scan direction was diagonal to the shorter sides and the ground measurements are placed along both the longer and shorter sides. For the truck the scan direction was parallel to the longer sides and all ground measurements are placed along the shorter sides.

6.2.1 Results

For terrain vehicle 1, see Figure 6.5, there were totally 67 data points on the object and the ground; 20 on the ground, 24 inner object and 23 outer object points. Of the outer object points, 12 belonged to the convex hull. The minimum rectangle estimation was performed on 10 different combinations of vertices and normal vectors. The quasi-Newton algorithm terminated after 18 iterations.

For terrain vehicle 2, see Figure 6.6, there were totally 59 data points on the object and the ground; 12 on the ground, 32 inner object and 15 outer object points. Of the outer object points, 10 belonged to the convex hull. The minimum rectangle estimation was performed on 9 different combinations of vertices and normal vectors. The quasi-Newton algorithm terminated after 13 iterations.

For the truck, see Figure 6.7, there were totally 85 data points on the object and the ground; 8 on the ground, 38 inner object and 39 outer object points. Of the outer objects point, 11 belonged to the convex hull. The minimum rectangle estimation was performed on 10 different combinations of vertices and normal vectors. The quasi-Newton algorithm terminated after 22 iterations.

The estimated lengths and widths of the vehicles are found in Table 6.1 and Table 6.2, respectively. The residuals of the LS methods are listed in Table 6.3 and in Table 6.4 the CPU times used by the LS algorithms are listed. The measurement of CPU time used are performed as described in Section 5.7 and the CPU values are averaged over 100 tests.

Let us first study the rectangles estimated by the LS methods. For all vehicles there are either ground points placed on the inside of the rectangle and/or outer object points placed on the outside of the rectangle. This is in order, as both coordinates of the data set are contaminated with errors.

If we compare the length and width estimates for terrain vehicle 1 and 2, we can see that the estimates for terrain vehicle 1 are closer to the true values than

	Min. rectangle	LS-TLS	Quasi-Newton	True value
Terrain vehicle 1	3.40	3.44	3.44	3.45
Terrain vehicle 2	3.82	4.00	4.00	3.45
Truck	7.52	7.57	7.57	7.50

Table 6.1: The length estimates of the vehicles. True value is a tape-measure of the vehicle. All values in meters.

	Min. rectangle	LS-TLS	Quasi-Newton	True value
Terrain vehicle 1	1.77	1.94	1.94	1.85
Terrain vehicle 2	1.92	1.92	1.92	1.85
Truck	2.15	2.04	2.04	2.09

Table 6.2: The width estimates of the vehicles. True value is a tape-measure of the vehicle. All values in meters.

the estimates for terrain vehicle 2. This is probably due to that for terrain vehicle 1 there are 43 ground and outer object points available while there are only 27 ground and outer object points available for terrain vehicle 2.

In the target classification process, Section 1.2, we proposed that the residual can be used as a "quality measure" of how well data fit the rectangle. The usage of the residual as a quality measure is probably useful when fitting the data set to different kinds of shapes, e.g., rectangles, ellipses etc. There is also a connection between the length and width estimates and the residual, although, it also includes the data point values. For example, the length and width estimates of Rectangle model 1 (see page 2.2e) are:

$$\begin{aligned}\hat{L} &= \hat{c}_1 - \hat{c}_3 \\ &= -\hat{n}_1 \left(\begin{pmatrix} x_{1,1} \\ \vdots \\ x_{1,N_1} \end{pmatrix} + \begin{pmatrix} x_{3,1} \\ \vdots \\ x_{3,N_3} \end{pmatrix} \right) - \hat{n}_2 \left(\begin{pmatrix} y_{1,1} \\ \vdots \\ y_{1,N_1} \end{pmatrix} + \begin{pmatrix} y_{3,1} \\ \vdots \\ y_{3,N_3} \end{pmatrix} \right) + \\ & \begin{pmatrix} r_{1,1} \\ \vdots \\ r_{1,N_1} \end{pmatrix} + \begin{pmatrix} r_{3,1} \\ \vdots \\ r_{3,N_3} \end{pmatrix}\end{aligned}$$

	LS-TLS	Quasi-Newton
Terrain vehicle 1	1.25	1.25
Terrain vehicle 2	.32	.32
Truck	.48	.48

Table 6.3: The residuals of the estimates for the LS algorithms. All values in m^2.

	LS-TLS	Quasi-Newton
Terrain vehicle 1	0.0009	0.1220
Terrain vehicle 2	0.00022	0.1073
Truck	0.0003	0.1369

Table 6.4: Mean of CPU time used, averaged over 100 tests, for the LS algorithms. All values in seconds.

and

$$\begin{aligned}\hat{W} &= \hat{c}_2 - \hat{c}_4 = \\ &= -\hat{n}_1 \left(\begin{pmatrix} y_{2,1} \\ \vdots \\ y_{2,N_2} \end{pmatrix} + \begin{pmatrix} y_{4,1} \\ \vdots \\ y_{4,N_4} \end{pmatrix} \right) + \hat{n}_2 \left(\begin{pmatrix} x_{2,1} \\ \vdots \\ x_{2,N_2} \end{pmatrix} + \begin{pmatrix} x_{4,1} \\ \vdots \\ x_{4,N_4} \end{pmatrix} \right) + \\ &\quad \begin{pmatrix} r_{2,1} \\ \vdots \\ r_{2,N_2} \end{pmatrix} + \begin{pmatrix} r_{4,1} \\ \vdots \\ r_{4,N_4} \end{pmatrix}.\end{aligned}$$

Thus, the residual cannot be used directly as a measurement of how well the estimates corresponds to the true (unknown) values. This can be seen clearly if we compare the residuals for terrain vehicle 1 and 2. The estimate of terrain vehicle 2 returns a smaller residual than the estimate of terrain vehicle 1, but the estimates of length and width are worse, compared to the true values. The correctness in the length and width estimates seems to be more dependent on the scan direction and the number of data points (N).

For this small set of examples the CPU time usage is some ten times smaller for the mixed LS-TLS algorithm than for the quasi-Newton algorithm. The difference between the algorithms seems to be larger for smaller data sets than for the larger data set used in Section 5.7.

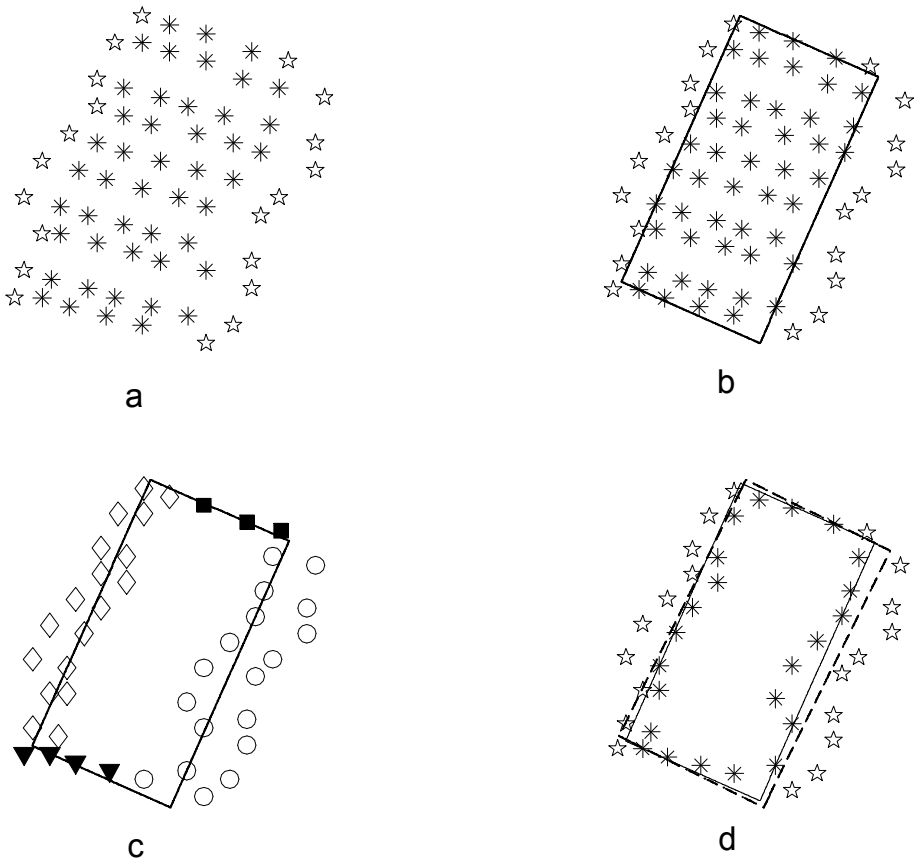

Figure 6.5: The rectangle estimation process for terrain vehicle 1. a: all data points on the object (∗) and ground (⋆), b: the minimum rectangle estimate, c: data association (same notation as in Figure 5.2), d: LS rectangle estimates. The LS-TLS and quasi-Newton curves are overlapping. Solid: minimum rectangle estimator; dashed: mixed LS-TLS; dotted: quasi-Newton.

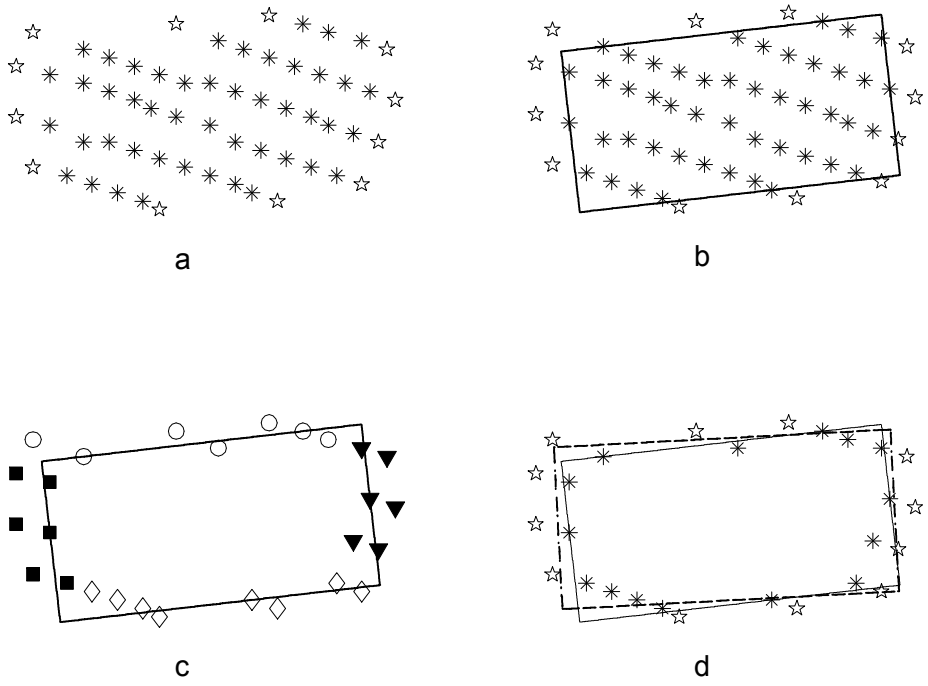

Figure 6.6: The rectangle estimation process for terrain vehicle 2. a: all data points on the object (∗) and ground (⋆), b: the minimum rectangle estimate, c: data association (same notation as in Figure 5.2), d: LS rectangle estimates. The LS-TLS and quasi-Newton curves are overlapping. Solid: minimum rectangle estimator; dashed: mixed LS-TLS; dotted: quasi-Newton.

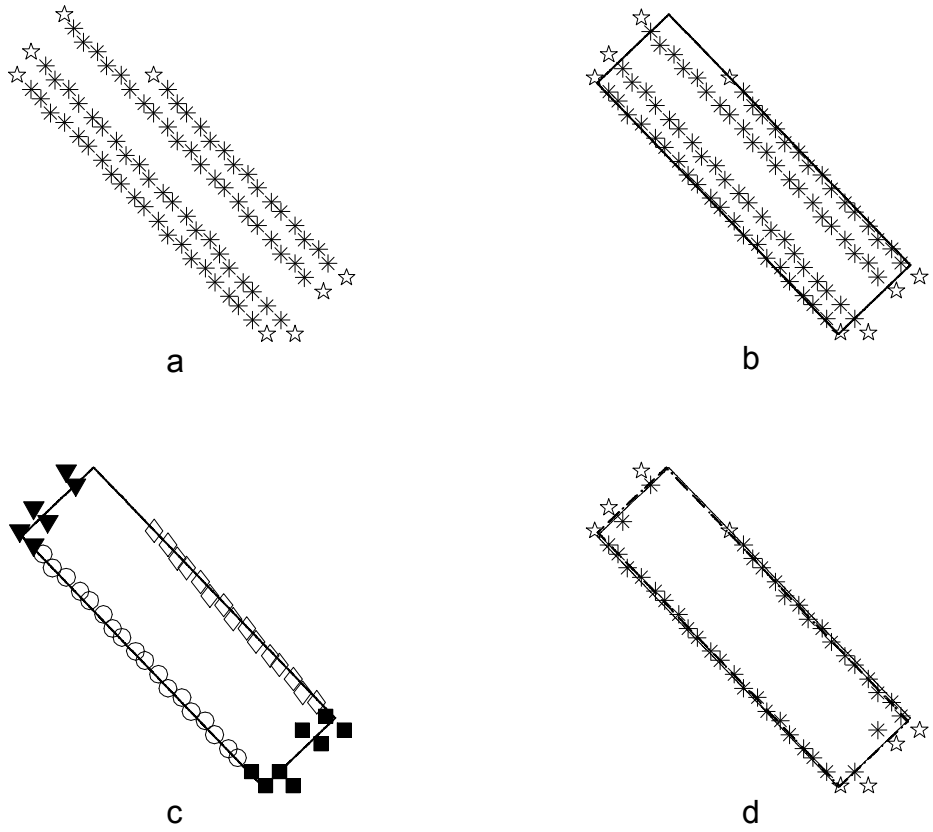

Figure 6.7: The rectangle estimation process for the truck. a: all data points on the object ($*$) and ground ($\star$), b: the minimum rectangle estimate, c: data association (same notation as in Figure 5.2), d: LS rectangle estimates. The LS-TLS and quasi-Newton curves are overlapping. Solid: minimum rectangle estimator; dashed: mixed LS-TLS; dotted: quasi-Newton.

6.3 Conclusions

In this chapter we tested the complete rectangle estimation process on both simulated, realistic data and on real laser radar data.

The tests on simulated data showed that the MSE and squared bias of the length and width estimates decreased when LS methods (LS-TLS or quasi-Newton) were used. For this type of data the improvement in MSE and squared bias for using a LS method after the minimum rectangle estimator was about $10-1000$ times. The improvement was less than in the previous test, see Section 5.7, as the number of samples, N, was smaller. The difference in mean CPU usage of the LS-TLS and quasi-Newton algorithms increased when the data set was smaller. The difference between the methods was about 100 times, compared to 10 times in the earlier simulations in Section 5.7.

The tests on real data showed that the number of samples, N, and the scan direction is vital for retrieving length and width estimates that are close to the true values. In the target classification process, see page 3, we suggested that the residual could be used as a "quality measure" of how well data fit the rectangle. The usage of the residual as a quality measure is probably useful when fitting the data set to different kinds of shapes, e.g., rectangles, ellipses etc. There is also a connection between the length and width estimates and the residual, but it also includes the data point values. Thus, the residual cannot be used directly as a measurement of how well the estimates corresponds to the true (unknown) values. For this small set of examples the CPU time usage was some ten times smaller for the mixed LS-TLS algorithm than for the quasi-Newton algorithm.

7

Discussion

7.1 Conclusions

We have proposed a method for estimation of the size and orientation of an object. The estimation process can be divided into several steps:

Given: A list of object points and ground points closest to the object, ordered in scan lines.

1. Sort all data points, both from the object and from the ground, in an order following the object edge clockwise or counter-clockwise.

2. Estimate the smallest rectangle that includes all data points from the object.

3. Associate each data point to one side of the rectangle (from step 2).

4. Improve the rectangle estimate using least squares methods based on both object and ground data.

5. Calculate the residual of the final rectangle estimate and the estimated length and width, respectively.

Returned: Estimates of length, width and orientation of the object. The residual of the rectangle estimate is also returned.

The data sorting algorithm proposed requires $\mathrm{O}(N)$ time, as knowledge of the data structure is available. We have proposed a method for fitting a rectangle as good as possible to object data, by minimizing the rectangle area that contains the convex hull. The method assumes that data are error free. As only object data, and no ground data, are used we get an estimate of the *minimum* area of the object. Thus, the length and width estimates of the vehicle will be biased. The method finds a set of four vertex points and a normal vector, i.e., an orientation, that minimizes the area of a rectangle. From the area estimate it is possible to receive estimates of the length and width of the object. The minimum rectangle estimation process consists of calculation of the convex hull and a rectangle minimization. Each step can be performed in linear time and thus, the total execution time requires $\mathrm{O}(N)$.

A method for association of outer object and ground data to the different sides of the rectangle have been proposed. Further, we proposed two ways of improving the rectangle estimate; a linear, constrained LS method (mixed LS-TLS, TLS: total least squares) and a nonlinear LS method (quasi-Newton). The fastest method, concerning CPU time usage, is the linear, constrained LS method which needs 10 times less CPU usage than any of the nonlinear LS methods.

The thesis ends with tests of the complete rectangle estimation process on both simulated, realistic, data and on real laser radar data. The tests on simulated data showed that the mean square error and squared bias of the length and width estimates decrease when LS methods (LS-TLS or quasi-Newton) are used. For this type of data the improvement in mean square error and squared bias when using a LS method after the minimum rectangle estimator was about 10-1000 times. The difference in mean CPU usage of the LS-TLS and quasi-Newton algorithms increased when the data set is smaller and is about 100 times.

The tests on real data showed that the number of samples, N, and the scan direction is vital for retrieving length and width estimates that are close to the true values. The usage of the residual as a quality measure is probably useful when fitting the data set to different kinds of shapes, e.g., rectangles, ellipses etc. There is also a connection between the length and width estimates and the residual, but it also includes the data point values. Thus, the residual cannot be used directly as a measurement of how well the estimates corresponds to the true (unknown) values.

7.2 Future work

This study is based on a number of assumptions. First, we assumed that the data collection and the segmentation process have been successful, i.e., that no outliers are present and that the complete object is visible. Second, we assumed that all vehicles in top view look like rectangles. Third, we have only used position data, i.e., x and y values.

7.2.1 Robustness to outliers

In many cases it is probably not possible to assume that all outliers are removed in the object detection process and then the rectangle estimation process must be able to handle them. In [20], Section 4.2.1, a method of removing outliers in the convex hull calculation is described. The method weights the vertices such that points that are too far away from the center of gravity of the data set are less trusted to be part of the convex hull.

Another way to handle outliers is to use another minimizing criterion than the squared residual during the improvement of the rectangle estimate. In [27], Zhang describes two interesting methods for conic fitting in the presence of outliers, M-estimators and least median of squares. In the M-estimator method the effect of outliers are reduced by replacing the squared residuals r_i^2 by another function of the residuals:

$$\min_{\theta \in \mathbb{R}^M} \sum_{i=1}^{N} \rho\left(r_i\right),$$

where ρ is a symmetric, positive-definite function with a unique minimum at zero and is chosen to increase less than square (e.g., $\rho\left(r_i\right) = r_i$). Several functions for ρ are described in [27]. If we can make a good estimate of the standard deviation of the errors of good data, then data whose error is larger than a certain number of standard deviations can be considered as outliers. In the least median of squares (LMedS) method the parameters are estimated by minimization of the median of the squared residuals, i.e.,

$$\min_{\theta \in \mathbb{R}^M} \underset{i}{\text{median}}\ r_i^2.$$

This estimator approach the smallest value for the median of squared residuals computed for the entire data set. According to [27], this method is very robust to false matches as well as outliers.

7.2.2 Other shapes than rectangles

In the 2D domain it is interesting to handle data sets including concavities. Then more general model (or models) must be applied. An idea of how to indicate where there are concavities in the data set is to perform the minimum rectangle estimation as described in Chapter 4. If ground data points are (significantly) placed in the interior of the convex hull there is probably a concavity in the object's shape. This improved shape estimator must be able to handle more complex shapes then a rectangle.

It is also interesting to extend the geometric fitting to 3D. Then the rectangle cannot be replaced by a cubic but with a more complex shape. Rectangle model 3 can probably be extended to shapes with concavities and/or to 3D. However, the problem will be a nonlinear LS problem, which has shown to need more CPU time than linear LS problems to reach a solution.

7.2.3 Including more information in the classification process

The laser radar system can also measure the intensity in the returned pulse. The intensity has shown to be dependent on the material in the object's surface [22]. This can be used in a data fusion process to improve the vehicle classification by adding information of what kind of material that is measured. Moreover, the laser radar system used in this thesis uses an older sensor technique. Never sensors with higher accuracy and resolution have been developed [23].

During the object detection process many imaging laser radar systems are supported by imaging systems working in infrared (IR) wavelengths. The detection can also be done by the IR system alone, i.e., the IR sensor is a cueing sensor. The IR signature of an object is also valuable to fuse with the laser radar information in a vehicle classification process.

A

Inclusion of the constraint in the linear least squares problem

In the linear constrained LS minimization problem, see Section 5.5.1, we have

$$\min_{\theta \in \mathbb{R}^M} |r|^2 = \min_{\theta \in \mathbb{R}^M} \sum_{i=1}^{N} r_i^2$$

subject to

$$\begin{pmatrix} 1 & 0 & 0 & 0 & x_{1,1} & y_{1,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & x_{1,N_1} & y_{1,N_1} \\ 0 & 1 & 0 & 0 & y_{2,1} & -x_{2,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & y_{2,N_2} & -x_{2,N_2} \\ 0 & 0 & 1 & 0 & x_{3,1} & y_{3,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & x_{3,N_3} & y_{3,N_3} \\ 0 & 0 & 0 & 1 & y_{4,1} & -x_{4,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & y_{4,N_4} & -x_{4,N_4} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{N_1+N_2+N_3+N_4} \end{pmatrix}$$

$$\text{and } n_1^2 + n_2^2 = 1,$$

where $\theta = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & n_1 & n_2 \end{bmatrix}^T$ and $N_1 + N_2 + N_3 + N_4 = N$.

The basic steps of Algorithm 3 will be explained in an example. The example will show how the constraint $n_1^2+n_2^2=1$ is included in the minimization. Consider the equation system

$$A\theta = 0,$$

which is estimated by

$$A\hat{\theta} = r,$$

using the minimization

$$\min_{\theta\in\mathbb{R}^M} |r|^2 = \min_{\theta\in\mathbb{R}^M} \sum_{i=1}^{N} r_i^2$$

subject to

$$n_1^2 + n_2^2 = 1.$$

Let us study the expression of the residual $|r|^2$. In Algorithm 3, step 2, a QR factorization of A is performed such that $A = QR$, where $Q^TQ = I$ and R is upper triangular. The 2-norm is invariant under orthogonal transformation (see [9], Section 2.5.2) and we have

$$|r|^2 = |A\theta|^2 = |QR\theta|^2 = |R\theta|^2,$$

where

$$|R\theta|^2 = \left| \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} \\ 0 & r_{22} & r_{23} & r_{24} & r_{25} & r_{26} \\ 0 & 0 & r_{33} & r_{34} & r_{35} & r_{36} \\ 0 & 0 & 0 & r_{44} & r_{45} & r_{46} \\ 0 & 0 & 0 & 0 & r_{55} & r_{56} \\ 0 & 0 & 0 & 0 & 0 & r_{66} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ n_1 \\ n_2 \end{pmatrix} \right|^2.$$

Thus, the minimization problem can be expressed as

$$\begin{gathered} \min_{\theta\in\mathbb{R}^M} \left(r_{11}c_1 + r_{12}c_2 + r_{13}c_3 + r_{14}c_4 + r_{15}n_1 + r_{16}n_2\right)^2 + \\ \left(r_{22}c_2 + r_{23}c_3 + r_{24}c_4 + r_{25}n_1 + r_{26}n_2\right)^2 + \\ \left(r_{33}c_3 + r_{34}c_4 + r_{35}n_1 + r_{36}n_2\right)^2 + \\ \left(r_{44}c_4 + r_{45}n_1 + r_{46}n_2\right)^2 + \\ \left(r_{55}n_1 + r_{56}n_2\right)^2 + \\ \left(r_{66}n_2\right)^2 = 0 \\ \text{subject to} \\ n_1^2 + n_2^2 = 1. \end{gathered}$$

The "real" minimization problem is a minimization over the unit circle:

$$\min_{n_1,n_2} \left| \begin{pmatrix} r_{55} & r_{56} \\ 0 & r_{66} \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} \right|^2$$
$$\text{subject to}$$
$$n_1^2 + n_2^2 = 1.$$

This problem is solved using SVD (step 3 in Algorithm 3). The values of $\begin{pmatrix} c_1 & c_2 & c_3 & c_4 \end{pmatrix}^T$ are given by back-substitution (step 5 in Algorithm 3).

B

Calculation of $\nabla f(\theta)$, $J^T(\theta) J(\theta)$ and $H(\theta)$

As described in (5.7), page 48, when the sides of the rectangle are listed counter clockwise the criterion function is

$$f(\theta) = \frac{1}{2}\sum_{s=1}^{N_1} r_s^2(\theta) + \frac{1}{2}\sum_{t=1}^{N_2} r_t^2(\theta) + \frac{1}{2}\sum_{u=1}^{N_3} r_u^2(\theta) + \frac{1}{2}\sum_{v=1}^{N_4} r_v^2(\theta),$$

where

$$\begin{aligned}
\theta &= (x_0, y_0, L, W, \alpha)^T, \\
r_s(\theta) &= -(x_s - x_0)\sin\alpha + (y_s - y_0)\cos\alpha - \frac{W}{2}, \\
r_t(\theta) &= -(x_t - x_0)\cos\alpha - (y_t - y_0)\sin\alpha - \frac{L}{2}, \\
r_u(\theta) &= (x_u - x_0)\sin\alpha - (y_u - y_0)\cos\alpha - \frac{W}{2}, \\
r_v(\theta) &= (x_v - x_0)\cos\alpha + (y_v - y_0)\sin\alpha - \frac{L}{2}, \\
N &= N_1 + N_2 + N_3 + N_4.
\end{aligned}$$

B.1 The gradient $\nabla f(\theta)$

The expression of $\nabla f(\theta)$ is

$$\nabla f(\theta) = \left[\begin{array}{ccccc} \frac{\partial}{\partial x_0} f(\theta) & \frac{\partial}{\partial y_0} f(\theta) & \frac{\partial}{\partial L} f(\theta) & \frac{\partial}{\partial W} f(\theta) & \frac{\partial}{\partial \alpha} f(\theta) \end{array} \right]^T,$$

where

$$\begin{aligned}
\frac{\partial}{\partial x_0} f(\theta) &= \sin\alpha \sum_{s=1}^{N_1} r_s(\theta) + \cos\alpha \sum_{t=1}^{N_2} r_t(\theta) \\
&\quad - \sin\alpha \sum_{u=1}^{N_3} r_u(\theta) - \cos\alpha \sum_{v=1}^{N_4} r_v(\theta), \\
\frac{\partial}{\partial y_0} f(\theta) &= -\cos\alpha \sum_{s=1}^{N_1} r_s(\theta) + \sin\alpha \sum_{t=1}^{N_2} r_t(\theta) \\
&\quad + \cos\alpha \sum_{u=1}^{N_3} r_u(\theta) - \sin\alpha \sum_{v=1}^{N_4} r_v(\theta), \\
\frac{\partial}{\partial L} f(\theta) &= -\frac{1}{2} \sum_{t=1}^{N_2} r_t(\theta) - \frac{1}{2} \sum_{v=1}^{N_4} r_v(\theta), \\
\frac{\partial}{\partial W} f(\theta) &= -\frac{1}{2} \sum_{s=1}^{N_1} r_s(\theta) - \frac{1}{2} \sum_{u=1}^{N_3} r_u(\theta), \\
\frac{\partial}{\partial \alpha} f(\theta) &= \sum_{s=1}^{N_1} \left[\frac{\partial}{\partial \alpha} r_s(\theta) \cdot r_s(\theta) \right] + \sum_{t=1}^{N_2} \left[\frac{\partial}{\partial \alpha} r_t(\theta) \cdot r_t(\theta) \right] + \\
&\quad \sum_{u=1}^{N_3} \left[\frac{\partial}{\partial \alpha} r_u(\theta) \cdot r_u(\theta) \right] + \sum_{v=1}^{N_4} \left[\frac{\partial}{\partial \alpha} r_v(\theta) \cdot r_v(\theta) \right],
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial}{\partial \alpha} r_s(\theta) &= -(x_s - x_0)\cos\alpha - (y_s - y_0)\sin\alpha, \\
\frac{\partial}{\partial \alpha} r_t(\theta) &= (x_t - x_0)\sin\alpha - (y_t - y_0)\cos\alpha, \\
\frac{\partial}{\partial \alpha} r_u(\theta) &= (x_u - x_0)\cos\alpha + (y_u - y_0)\sin\alpha, \\
\frac{\partial}{\partial \alpha} r_v(\theta) &= -(x_v - x_0)\sin\alpha + (y_v - y_0)\cos\alpha.
\end{aligned}$$

B.2 The Jacobian $J(\theta)$ and $J^T(\theta)J(\theta)$

To find the expression of the Jacobian, $J(\theta), J(\theta)_{ij} = \frac{\partial r_i}{\partial \theta_j}$, we must first identify $R(\theta)$:

$$\begin{aligned} f(\theta) &= \frac{1}{2}\sum_{s=1}^{N_1} r_s^2(\theta) + \frac{1}{2}\sum_{t=1}^{N_2} r_t^2(\theta) + \frac{1}{2}\sum_{u=1}^{N_3} r_u^2(\theta) + \frac{1}{2}\sum_{v=1}^{N_4} r_v^2(\theta) \\ &= \frac{1}{2} R^T(\theta) R(\theta), \end{aligned}$$

where

$$R(\theta) = \begin{pmatrix} R_s(\theta) & R_t(\theta) & R_u(\theta) & R_v(\theta) \end{pmatrix}^T.$$

Let us now return to the Jacobian:

$$\begin{aligned} J(\theta) &= \frac{\partial}{\partial \theta} R(\theta) \\ &= \begin{pmatrix} \frac{\partial}{\partial x_0} R_s(\theta) & \frac{\partial}{\partial y_0} R_s(\theta) & \frac{\partial}{\partial L} R_s(\theta) & \frac{\partial}{\partial W} R_s(\theta) & \frac{\partial}{\partial \alpha} R_s(\theta) \\ \frac{\partial}{\partial x_0} R_t(\theta) & \frac{\partial}{\partial y_0} R_t(\theta) & \frac{\partial}{\partial L} R_t(\theta) & \frac{\partial}{\partial W} R_t(\theta) & \frac{\partial}{\partial \alpha} R_t(\theta) \\ \frac{\partial}{\partial x_0} R_u(\theta) & \frac{\partial}{\partial y_0} R_u(\theta) & \frac{\partial}{\partial L} R_u(\theta) & \frac{\partial}{\partial W} R_u(\theta) & \frac{\partial}{\partial \alpha} R_u(\theta) \\ \frac{\partial}{\partial x_0} R_v(\theta) & \frac{\partial}{\partial y_0} R_v(\theta) & \frac{\partial}{\partial L} R_v(\theta) & \frac{\partial}{\partial W} R_v(\theta) & \frac{\partial}{\partial \alpha} R_v(\theta) \end{pmatrix}, \end{aligned}$$

where

$$\begin{aligned} R_s(\theta) &= r_s, \quad s = 1, ..., N_1 \\ R_t(\theta) &= r_t, \quad t = 1, ..., N_2 \\ R_u(\theta) &= r_u, \quad u = 1, ..., N_3 \\ R_v(\theta) &= r_v, \quad v = 1, ..., N_4. \end{aligned}$$

The Jacobian have dimension $(N_1 + N_2 + N_3 + N_4) \times M$. Thus,

$$J(\theta) = \begin{pmatrix} \sin\alpha & -\cos\alpha & 0 & -\frac{1}{2} & \frac{\delta}{\delta\alpha} r_{s=1}(\theta) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sin\alpha & -\cos\alpha & 0 & -\frac{1}{2} & \frac{\delta}{\delta\alpha} r_{s=N_1}(\theta) \\ \cos\alpha & \sin\alpha & -\frac{1}{2} & 0 & \frac{\delta}{\delta\alpha} r_{t=1}(\theta) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cos\alpha & \sin\alpha & -\frac{1}{2} & 0 & \frac{\delta}{\delta\alpha} r_{t=N_2}(\theta) \\ -\sin\alpha & \cos\alpha & 0 & -\frac{1}{2} & \frac{\delta}{\delta\alpha} r_{u=1}(\theta) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\sin\alpha & \cos\alpha & 0 & -\frac{1}{2} & \frac{\delta}{\delta\alpha} r_{u=N_3}(\theta) \\ -\cos\alpha & -\sin\alpha & -\frac{1}{2} & 0 & \frac{\delta}{\delta\alpha} r_{v=1}(\theta) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\cos\alpha & -\sin\alpha & -\frac{1}{2} & 0 & \frac{\delta}{\delta\alpha} r_{v=N_4}(\theta) \end{pmatrix}.$$

This gives the following expression of $J^T(\theta) J(\theta)$:

$$J^T(\theta) J(\theta) = \begin{pmatrix} j^{1,1} & j^{1,2} & j^{1,3} & j^{1,4} & j^{1,5} \\ j^{1,2} & j^{2,2} & j^{2,3} & j^{2,4} & j^{2,5} \\ j^{1,3} & j^{2,3} & j^{3,3} & j^{3,4} & j^{3,5} \\ j^{1,4} & j^{2,4} & j^{3,4} & j^{4,4} & j^{4,5} \\ j^{1,5} & j^{2,5} & j^{3,5} & j^{4,5} & j^{5,5} \end{pmatrix},$$

where

$$\begin{aligned}
j^{1,1} &= N_1 \sin^2\alpha + N_2 \cos^2\alpha + N_3 \sin^2\alpha + N_4 \cos^2\alpha, \\
j^{1,2} &= -N_1 \sin\alpha\cos\alpha + N_2 \sin\alpha\cos\alpha - N_3 \sin\alpha\cos\alpha + N_4 \sin\alpha\cos\alpha, \\
j^{2,2} &= N_1 \cos^2\alpha + N_2 \sin^2\alpha + N_3 \cos^2\alpha + N_4 \sin^2\alpha, \\
j^{1,3} &= -\frac{1}{2} N_2 \cos\alpha + \frac{1}{2} N_4 \cos\alpha, \\
j^{2,3} &= -\frac{1}{2} N_2 \sin\alpha + \frac{1}{2} N_4 \sin\alpha, \\
j^{3,3} &= \frac{1}{4} N_2 + \frac{1}{4} N_4, \\
j^{1,4} &= -\frac{1}{2} N_1 \sin\alpha + \frac{1}{2} N_3 \sin\alpha, \\
j^{2,4} &= +\frac{1}{2} N_1 \cos\alpha - \frac{1}{2} N_3 \cos\alpha, \\
j^{3,4} &= 0, \\
j^{4,4} &= \frac{1}{4} N_1 + \frac{1}{4} N_3, \\
j^{1,5} &= \sin\alpha \sum_{s=1}^{N_1} \frac{\partial}{\partial\alpha} r_s(\theta) + \cos\alpha \sum_{t=1}^{N_2} \frac{\partial}{\partial\alpha} r_t(\theta), \\
&\quad - \sin\alpha \sum_{u=1}^{N_3} \frac{\partial}{\partial\alpha} r_u(\theta) - \cos\alpha \sum_{v=1}^{N_4} \frac{\partial}{\partial\alpha} r_v(\theta), \\
j^{2,5} &= -\cos\alpha \sum_{s=1}^{N_1} \frac{\partial}{\partial\alpha} r_s(\theta) + \sin\alpha \sum_{t=1}^{N_2} \frac{\partial}{\partial\alpha} r_t(\theta), \\
&\quad + \cos\alpha \sum_{u=1}^{N_3} \frac{\partial}{\partial\alpha} r_u(\theta) - \sin\alpha \sum_{t=1}^{N_4} \frac{\partial}{\partial\alpha} r_v(\theta), \\
j^{3,5} &= -\frac{1}{2} \sum_{t=1}^{N_2} \frac{\partial}{\partial\alpha} r_t(\theta) - \frac{1}{2} \sum_{t=1}^{N_4} \frac{\partial}{\partial\alpha} r_v(\theta), \\
j^{4,5} &= -\frac{1}{2} \sum_{s=1}^{N_1} \frac{\partial}{\partial\alpha} r_s(\theta) - \frac{1}{2} \sum_{u=1}^{N_3} \frac{\partial}{\partial\alpha} r_u(\theta),
\end{aligned}$$

and

$$\begin{aligned} j^{5,5} &= \sum_{s=1}^{N_1}\left(\frac{\partial}{\partial\alpha}r_s(\theta)\right)^2+\sum_{t=1}^{N_2}\left(\frac{\partial}{\partial\alpha}r_t(\theta)\right)^2 \\ &\quad +\sum_{u=1}^{N_3}\left(\frac{\partial}{\partial\alpha}r_u(\theta)\right)^2+\sum_{t=1}^{N_4}\left(\frac{\partial}{\partial\alpha}r_v(\theta)\right)^2. \end{aligned}$$

If $N_1 = N_2 = N_3 = N_4$ we have

$$J^T(\theta)J(\theta) = \begin{pmatrix} 2N_1 & 0 & 0 & 0 & j^{1,5} \\ 0 & 2N_1 & 0 & 0 & j^{2,5} \\ 0 & 0 & \frac{1}{2}N_1 & 0 & j^{3,5} \\ 0 & 0 & 0 & \frac{1}{2}N_1 & j^{4,5} \\ j^{1,5} & j^{2,5} & j^{3,5} & j^{4,5} & j^{5,5} \end{pmatrix},$$

where

$$\begin{aligned} j^{1,5} &= \sin\alpha\sum_{s=1}^{N_1}\frac{\partial}{\partial\alpha}r_s(\theta)-\sin\alpha\sum_{u=1}^{N_1}\frac{\partial}{\partial\alpha}r_u(\theta) \\ &\quad +\cos\alpha\sum_{t=1}^{N_1}\frac{\partial}{\partial\alpha}r_t(\theta)-\cos\alpha\sum_{v=1}^{N_1}\frac{\partial}{\partial\alpha}r_v(\theta), \\ j^{2,5} &= -\cos\alpha\sum_{s=1}^{N_1}\frac{\partial}{\partial\alpha}r_s(\theta)+\cos\alpha\sum_{u=1}^{N_1}\frac{\partial}{\partial\alpha}r_u(\theta) \\ &\quad +\sin\alpha\sum_{t=1}^{N_1}\frac{\partial}{\partial\alpha}r_t(\theta)-\sin\alpha\sum_{t=1}^{N_1}\frac{\partial}{\partial\alpha}r_v(\theta), \\ j^{3,5} &= -\frac{1}{2}\sum_{t=1}^{N_1}\frac{\partial}{\partial\alpha}r_t(\theta)-\frac{1}{2}\sum_{t=1}^{N_1}\frac{\partial}{\partial\alpha}r_v(\theta) \\ j^{4,5} &= -\frac{1}{2}\sum_{s=1}^{N_1}\frac{\partial}{\partial\alpha}r_s(\theta)-\frac{1}{2}\sum_{u=1}^{N_1}\frac{\partial}{\partial\alpha}r_u(\theta), \\ j^{5,5} &= \sum_{s=1}^{N_1}\left(\frac{\partial}{\partial\alpha}r_s(\theta)\right)^2+\sum_{t=1}^{N_1}\left(\frac{\partial}{\partial\alpha}r_t(\theta)\right)^2 \\ &\quad +\sum_{u=1}^{N_1}\left(\frac{\partial}{\partial\alpha}r_u(\theta)\right)^2+\sum_{t=1}^{N_1}\left(\frac{\partial}{\partial\alpha}r_v(\theta)\right)^2. \end{aligned}$$

In this case $J^T(\theta)J(\theta)$ will be singular if

$$j^{1,5} = j^{2,5} = j^{3,5} = j^{4,5} = j^{5,5} = 0,$$

which will occur for an error free data set.

B.3 The Hessian $H(\theta)$

The Hessian is a symmetric matrix defined as

$$H(\theta) = \nabla^2 f(\theta) = \frac{\partial^2}{\partial \theta^2} f(\theta).$$

In this case the expression of the Hessian is

$$H(\theta) = \begin{pmatrix} \frac{\partial^2}{\partial x_0^2} f(\theta) & \frac{\partial}{\partial x_0}\frac{\partial}{\partial y_0} f(\theta) & \frac{\partial}{\partial x_0}\frac{\partial}{\partial L} f(\theta) & \frac{\partial}{\partial x_0}\frac{\partial}{\partial W} f(\theta) & \frac{\partial}{\partial x_0}\frac{\partial}{\partial \alpha} f(\theta) \\ \frac{\partial}{\partial x_0}\frac{\partial}{\partial y_0} f(\theta) & \frac{\partial^2}{\partial y_0^2} f(\theta) & \frac{\partial}{\partial y_0}\frac{\partial}{\partial L} f(\theta) & \frac{\partial}{\partial y_0}\frac{\partial}{\partial W} f(\theta) & \frac{\partial}{\partial y_0}\frac{\partial}{\partial \alpha} f(\theta) \\ \frac{\partial}{\partial x_0}\frac{\partial}{\partial L} f(\theta) & \frac{\partial}{\partial y_0}\frac{\partial}{\partial L} f(\theta) & \frac{\partial^2}{\partial L^2} f(\theta) & \frac{\partial}{\partial L}\frac{\partial}{\partial W} f(\theta) & \frac{\partial}{\partial L}\frac{\partial}{\partial \alpha} f(\theta) \\ \frac{\partial}{\partial x_0}\frac{\partial}{\partial W} f(\theta) & \frac{\partial}{\partial y_0}\frac{\partial}{\partial W} f(\theta) & \frac{\partial}{\partial L}\frac{\partial}{\partial W} f(\theta) & \frac{\partial^2}{\partial W^2} f(\theta) & \frac{\partial}{\partial W}\frac{\partial}{\partial \alpha} f(\theta) \\ \frac{\partial}{\partial x_0}\frac{\partial}{\partial \alpha} f(\theta) & \frac{\partial}{\partial y_0}\frac{\partial}{\partial \alpha} f(\theta) & \frac{\partial}{\partial L}\frac{\partial}{\partial \alpha} f(\theta) & \frac{\partial}{\partial W}\frac{\partial}{\partial \alpha} f(\theta) & \frac{\partial^2}{\partial \alpha^2} f(\theta) \end{pmatrix}$$

where

$$\begin{aligned}
\frac{\partial^2}{\partial x_0^2} f(\theta) &= N_1 \sin^2\alpha + N_2 \cos^2\alpha + N_3 \sin^2\alpha + N_4 \cos^2\alpha, \\
\frac{\partial}{\partial x_0}\frac{\partial}{\partial y_0} f(\theta) &= -N_1 \sin\alpha\cos\alpha + N_2 \sin\alpha\cos\alpha \\
&\quad -N_3 \sin\alpha\cos\alpha + N_4 \sin\alpha\cos\alpha, \\
\frac{\partial^2}{\partial y_0^2} f(\theta) &= N_1 \cos^2\alpha + N_2 \sin^2\alpha + N_3 \cos^2\alpha + N_4 \sin^2\alpha, \\
\frac{\partial}{\partial x_0}\frac{\partial}{\partial L} f(\theta) &= -\frac{1}{2} N_2 \cos\alpha + \frac{1}{2} N_4 \cos\alpha, \\
\frac{\partial}{\partial y_0}\frac{\partial}{\partial L} f(\theta) &= -\frac{1}{2} N_2 \sin\alpha + \frac{1}{2} N_4 \sin\alpha, \\
\frac{\partial^2}{\partial L^2} f(\theta) &= \frac{1}{4} N_2 + \frac{1}{4} N_4, \\
\frac{\partial}{\partial x_0}\frac{\partial}{\partial W} f(\theta) &= -\frac{1}{2} N_1 \sin\alpha + \frac{1}{2} N_3 \sin\alpha, \\
\frac{\partial}{\partial y_0}\frac{\partial}{\partial W} f(\theta) &= \frac{1}{2} N_1 \cos\alpha - \frac{1}{2} N_3 \cos\alpha, \\
\frac{\partial}{\partial L}\frac{\partial}{\partial W} f(\theta) &= 0, \\
\frac{\partial^2}{\partial W^2} f(\theta) &= \frac{1}{4} N_1 + \frac{1}{4} N_3,
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial}{\partial x_0}\frac{\partial}{\partial \alpha}f(\theta) &= \cos\alpha\sum_{s=1}^{N_1} r_s(\theta) - \sin\alpha\sum_{t=1}^{N_2} r_t(\theta) \\
&\quad - \cos\alpha\sum_{u=1}^{N_3} r_u(\theta) + \sin\alpha\sum_{v=1}^{N_4} r_v(\theta) \\
&\quad + \sin\alpha\sum_{s=1}^{N_1}\frac{\delta}{\delta\alpha} r_s(\theta) + \cos\alpha\sum_{t=1}^{N_2}\frac{\delta}{\delta\alpha} r_t(\theta) \\
&\quad - \sin\alpha\sum_{u=1}^{N_3}\frac{\delta}{\delta\alpha} r_u(\theta) - \cos\alpha\sum_{v=1}^{N_4}\frac{\delta}{\delta\alpha} r_v(\theta), \\
\frac{\partial}{\partial y_0}\frac{\partial}{\partial \alpha}f(\theta) &= \sin\alpha\sum_{s=1}^{N_1} r_s(\theta) + \cos\alpha\sum_{t=1}^{N_2} r_t(\theta) \\
&\quad - \sin\alpha\sum_{u=1}^{N_3} r_u(\theta) - \cos\alpha\sum_{v=1}^{N_4} r_v(\theta) \\
&\quad - \cos\alpha\sum_{s=1}^{N_1}\frac{\delta}{\delta\alpha} r_s(\theta) + \sin\alpha\sum_{t=1}^{N_2}\frac{\delta}{\delta\alpha} r_t(\theta) \\
&\quad + \cos\alpha\sum_{u=1}^{N_3}\frac{\delta}{\delta\alpha} r_u(\theta) - \sin\alpha\sum_{v=1}^{N_4}\frac{\delta}{\delta\alpha} r_v(\theta), \\
\frac{\partial}{\partial L}\frac{\partial}{\partial \alpha}f(\theta) &= -\frac{1}{2}\sum_{t=1}^{N_2}\frac{\delta}{\delta\alpha} r_t(\theta) - \frac{1}{2}\sum_{v=1}^{N_4}\frac{\delta}{\delta\alpha} r_v(\theta), \\
\frac{\partial}{\partial W}\frac{\partial}{\partial \alpha}f(\theta) &= -\frac{1}{2}\sum_{s=1}^{N_1}\frac{\delta}{\delta\alpha} r_s(\theta) - \frac{1}{2}\sum_{u=1}^{N_3}\frac{\delta}{\delta\alpha} r_u(\theta), \\
\frac{\partial^2}{\partial \alpha^2}f(\theta) &= \sum_{s=1}^{N_1}\left[\frac{\partial^2}{\partial\alpha^2} r_s(\theta)\cdot r_s(\theta)\right] + \sum_{s=1}^{N_1}\left[\frac{\delta}{\delta\alpha} r_s(\theta)\cdot\frac{\delta}{\delta\alpha} r_s(\theta)\right] + \\
&\quad + \sum_{t=1}^{N_2}\left[\frac{\partial^2}{\partial\alpha^2} r_t(\theta)\cdot r_t(\theta)\right] + \sum_{t=1}^{N_2}\left[\frac{\delta}{\delta\alpha} r_t(\theta)\cdot\frac{\delta}{\delta\alpha} r_t(\theta)\right] \\
&\quad + \sum_{u=1}^{N_3}\left[\frac{\partial^2}{\partial\alpha^2} r_u(\theta)\cdot r_u(\theta)\right] + \sum_{u=1}^{N_3}\left[\frac{\delta}{\delta\alpha} r_u(\theta)\cdot\frac{\delta}{\delta\alpha} r_u(\theta)\right] \\
&\quad + \sum_{v=1}^{N_4}\left[\frac{\partial^2}{\partial\alpha^2} r_v(\theta)\cdot r_v(\theta)\right] + \sum_{v=1}^{N_4}\left[\frac{\delta}{\delta\alpha} r_v(\theta)\cdot\frac{\delta}{\delta\alpha} r_v(\theta)\right],
\end{aligned}
$$

and

$$\begin{aligned}
\frac{\partial^2}{\partial \alpha^2} r_s(\theta) &= +(x_s - x_0)\sin\alpha - (y_s - y_0)\cos\alpha, \\
\frac{\partial^2}{\partial \alpha^2} r_t(\theta) &= +(x_t - x_0)\cos\alpha + (y_t - y_0)\sin\alpha, \\
\frac{\partial^2}{\partial \alpha^2} r_u(\theta) &= -(x_u - x_0)\sin\alpha + (y_u - y_0)\cos\alpha, \\
\frac{\partial^2}{\partial \alpha^2} r_v(\theta) &= -(x_v - x_0)\cos\alpha - (y_v - y_0)\sin\alpha.
\end{aligned}$$

Bibliography

[1] E. P. Baltsavias. Airborne laser scanning: Basic relations and formulas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:199–214, 1999.

[2] C. Carlsson, E. Jungert, C. Leuhusen, D. Letalick, and O. Steinvall. A comparative study between two target detection methods applied to ladar data. In *Proceeding of the Ninth Conference on Coherent Laser Radar*, pages 220–223, 23-27 June 1997.

[3] C. Carlsson, E. Jungert, C. Leuhusen, D. Letalick, and O. Steinvall. Target detection using data from a terrain profiling laser radar. In *Proceedings of the Third International Airborne Remote Sensing Conference and Exhibition*, pages I–431 – I–438. ERIM International, July 1997.

[4] C.-L. Cheng and J. W. Van Ness. *Statistical Regression with Measurment Error*. Kendall's Library of Statistics. Arnold, London, 1999.

[5] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall, Englewoods Cliffs, NJ, 1983.

[6] W. Gander, G. H. Golub, and R. Strebel. Least-squares fitting of circles and ellipses. *BIT*, 34:558–578, 1994.

[7] W. Gander and U. Von Matt. *Solving Problems in Scientific Computing Using Maple and Matlab*, chapter 6: Some Least Squares Problems, pages 83–101. Springer, Berlin, 3rd edition, 1997.

[8] J. De Geeter, H. Van Brussel, J. De Schutter, and M. Decréton. A smoothly constrained kalman filter. *IEEE PAMI*, 19(10):1171–1177, 1997.

[9] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins, Baltimore, MD, 3rd edition, 1996.

[10] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, 1993.

[11] N. Haala and C. Brenner. Virtual city models from laser altimeter and 2d map data. *Photogrammetric Engineering and Remote Sensing*, 65(7):787–795, 1999.

[12] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem. Computational Aspects and Analysis*. SIAM, Philadelphia, 1991.

[13] E. J. Huising and L. M. Gomes Pereira. Errors and accuracy estimates of laser data acquired by various laser scanning systems for topographic applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53:245–261, 1998.

[14] A. V. Jelalian. *Laser Radar Systems*. Artech House, Norwood, MA, 1992.

[15] E. Jungert. A qualitative approach ro recognition of man-made objects in laser-radar images. In *Proceedings of the 7th International Symposium on Spatial Data Handling, SDH '96*, pages A–15–27, Delft, The Netherlands, 12-16 August 1996.

[16] E. Jungert, C. Carlsson, and C. Leuhusen. A qualitative matching technique for handling uncertainties in laser radar images. In Firooz A. Sadjadi, editor, *Automatic Target Recognition VIII*, pages 62–71. SPIE, 13-17 April 1998.

[17] T. Kailath. *Linear Systems*, chapter Appendix A. Prentice Hall, Englewood Cliffs, NJ, 1980.

[18] L. Ljung. *System Identification. Theory for the User*. Prentice Hall, NJ, 2nd edition, 1999.

[19] H.-G. Maas and G. Vosselman. Two algorithms for extracting building models from raw laser altimetry data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:153–163, 1999.

[20] F. P. Preparata and M. I. Shamos. *Computational Geometry, An Introduction*. Springer, New York, 1985.

[21] R. Sedgewick. *Algorithms*. Addison-Wesley, Reading, MA, 2nd edition, 1988.

[22] O. Steinvall. Theory for laser systems performance modelling. Scientific Report FOA-R–97-00599-612-SE, Division of Sensor Technology, Defence Research Establishment of Sweden, Linköping, Sweden, October 1997.

[23] O. Steinvall, U. Söderman, S. Ahlberg, M. Sandberg, D. Letalick, and E. Jungert. Airborne laser radar: Systems and methods for reconnaissance and terrain modelling. In *SPIE Conference on Laser Radar Technology and Applications IV*, volume 3707, pages 12–26. SPIE, April 1999.

[24] J. Svensson. Matching vehicles from laser radar images in the target recognition process. Master's thesis LiTH-IDA-Ex-00/45, Department of Computer and Information Science, Linköping University, Sweden, 2000.

[25] A. Wehr and U. Lohr. Airborne laser scanning - an introduction and overview. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:68–82, 1999.

[26] M. Wellfare, T. Holmes, S. Pohlman, D. Geci, K. Norris-Zachery, and R. Patton. Identification of vehicle targets from low-cost ladar seeker imagery. In Gary W Kamerman, editor, *Conference on Laser Radar Technology and Applications*, volume 2748, pages 272–282. SPIE, April 1996.

[27] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. Technical Report 2676, Institut national de recherche an informatique et en automatique, Unité de recherche INRIA, Sophia-Antipolis, France, October 1995.