Vehicle detection and classification in video sequences

Examensarbete utfört i Bildbehandling
vid Tekniska Högskolan i Linköping
av

Andreas Böckert

Reg nr: LiTH-ISY-EX-3270-2002
Vehicle detection and classification in video sequences

Examensarbete utfört i Bildbehandling vid Tekniska Högskolan i Linköping

av

Andreas Böckert

Reg nr: LiTH-ISY-EX-3270-2002

Supervisor: Per-Erik Forssén

Examiner: Klas Nordberg

Linköping, August 21, 2002.
Upptäckt och klassificering av fordon i videosequenser.  
Vehicle detection and classification in video sequences.

Author: Andreas Böckert

The purpose of this thesis is to investigate the applicability of a certain model based classification algorithm. The algorithm is centered around a flexible wireframe prototype that can instantiate a number of different vehicle classes such as a hatchback, pickup or a bus to mention a few. The parameter fitting is based on Newton minimization of errors between instantiated model segments and observed data segments. Furthermore a number of methods for object detection based on motion are described and evaluated. Results from both experimental and real world data will be presented.

Keywords  
Object detection, object classification, model based tracking, model based classification
Abstract

The purpose of this thesis is to investigate the applicability of a certain model based classification algorithm. The algorithm is centered around a flexible wire-frame prototype that can instantiate a number of different vehicle classes such as a hatchback, pickup or a bus to mention a few. The parameters of the model are fitted using Newton minimization of errors between model line segments and observed line segments. Furthermore a number of methods for object detection based on motion are described and evaluated. Results from both experimental and real world data is presented.

Keywords:

Object detection, object classification, model based tracking, model based classification.
Acknowledgments

I would like thank my coworkers at the computer vision laboratory at the department of electrical engineering at the Linköping university, especially my examiner Klas Nordberg and my supervisor Per-Erik Forssén, for helpful input and guidance. I would also like to thank Andreas Sigfridsson for excellent feedback and mild pressure to complete this work. Finally I would like to thank my friends for putting up with my all of my whining during the darker hours of my project.
Contents

1 Introduction ................................. 1
   1.1 WITAS .................................. 1
   1.2 Problem description ....................... 1
   1.3 Problem conditions ....................... 2
      1.3.1 Domain description ................. 2
      1.3.2 Platform description ............... 2
   1.4 Report structure ......................... 3

2 Significant features ....................... 5
   2.1 Shape .................................. 5
   2.2 Colour .................................. 5
   2.3 Motion .................................. 6
   2.4 Deformability ......................... 6

3 Related work ................................ 7
   3.1 Traffic monitoring from moving platform .... 7
   3.2 WITAS .................................. 7
   3.3 Cornell .................................. 7
   3.4 Carnegie Mellon ......................... 8
   3.5 University of Southern California ....... 8
   3.6 Traffic monitoring from static platform ... 8

4 Object detection .......................... 9
   4.1 Method outline .......................... 9
   4.2 Locating moving objects ................. 9
   4.3 Feature tracking ........................ 11
      4.3.1 Kanade-Lucas-Tomasi tracker ....... 11
   4.4 Image registration ....................... 13
      4.4.1 Translational transformation ....... 14
      4.4.2 Affine transformation ............... 14
      4.4.3 Quadratic transformation .......... 15
      4.4.4 Homographic transformation ....... 16
      4.4.5 Outlier detection and handling .... 19
4.5 Image warping ........................................... 20

5 Object classification ........................................ 23
  5.1 Proposed method ........................................ 23
  5.2 Line extraction ......................................... 25
  5.3 Vehicle prototype ....................................... 25
  5.4 Introducing the prototype vector ....................... 25
  5.5 Model instantiation ..................................... 28
  5.6 Initial estimate ......................................... 29
  5.7 Parameter fitting ....................................... 30
    5.7.1 Error measure ...................................... 30
    5.7.2 Calculation of the Jacobian ......................... 32
    5.7.3 Newton iteration .................................... 37
    5.7.4 Levenberg-Marquardt iteration ..................... 37
  5.8 Keeping fixed lines .................................... 38
  5.9 Segment matching ...................................... 39
    5.9.1 Midpoint distance .................................. 40
    5.9.2 Centered midpoint distance ......................... 40
    5.9.3 Normalized midpoint distance ....................... 41
    5.9.4 Mahalanobis distance ............................... 42
    5.9.5 Mahalanobis ratio .................................. 43
  5.10 Fitting polices ....................................... 43
    5.10.1 Fitting all lines .................................. 43
    5.10.2 Minimal fitting .................................... 44
    5.10.3 Overdetermined fitting ............................. 44
    5.10.4 Close fitting ...................................... 44
    5.10.5 Progressive fitting ............................... 44
    5.10.6 Random fitting .................................... 45

6 Evaluation of detection techniques ......................... 47
  6.1 Tracker performance ................................... 47
  6.2 Resampling ............................................. 47
  6.3 Evaluation of the different transformations .......... 49

7 Evaluation of classification techniques .................... 53
  7.1 Synthetic test setup ................................... 53
    7.1.1 Perfect test setup .................................. 54
    7.1.2 Noisy test setup ................................... 54
    7.1.3 Incomplete test setup .............................. 55
    7.1.4 Garbage test setup .................................. 55
    7.1.5 Nightmare test setup ............................... 55
  7.2 Evaluation of different line matching methods .......... 55
    7.2.1 Perfect test results ................................ 58
    7.2.2 Noisy test results .................................. 58
### Notations

<table>
<thead>
<tr>
<th>Term</th>
<th>Symbol(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector</td>
<td>$\mathbf{x}$, $\mathbf{x}_i$</td>
</tr>
<tr>
<td>Vector component $i$</td>
<td>$x_i$, $x_{it}$</td>
</tr>
<tr>
<td>Matrix</td>
<td>$A$</td>
</tr>
<tr>
<td>Matrix element at row $i$, column $j$ in $A$</td>
<td>$a_{ij}$</td>
</tr>
<tr>
<td>Expected value of $\mathbf{x}$</td>
<td>$E{\mathbf{x}}$</td>
</tr>
<tr>
<td>Normally distributed variable $\mathbf{x}$ with mean $\mu$ and covariance $C$</td>
<td>$\mathbf{x} \in \mathcal{N}(\mu, C)$</td>
</tr>
<tr>
<td>Two norm, $(\sqrt{\mathbf{x}^T \mathbf{x}})$</td>
<td>$|\mathbf{x}|_2$</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In this chapter will give a short introduction to the problem that is to be addressed. A description of the WITAS project will be given followed by a outline of the problem. Finally a section is dedicated to the structure of the report.

1.1 WITAS

The Wallenberg laboratory for research on Information Technology and Autonomous Systems, WITAS, is formed by four research groups at the Linköping University, Sweden. Three of the research groups are at the Department of Computer and Information Systems and the last one is from the Department of Electrical Engineering (Computer Vision Lab). An overview of the current work at CVL can be found in [9].

Currently the main project at WITAS is the development of a small autonomous helicopter. The helicopter should make rational decisions based on on-board sensors, such as visual information, known geographical data and information sent to it via radio.

The intended application for the helicopter is traffic monitoring. A number of different tasks are executed, such as queue detection and tracking of individual vehicles. To achieve this the helicopter currently relies solely on visual information attained by an on board video camera. This camera is an active sensor, even more so than the traditional active cameras with zoom, pan and tilt control, since the helicopter can be positioned at an arbitrary location. It delivers a video sequence at a frame rate of 25 frames per second with colour information given in a RGB colour space. The colour information is given at a resolution of 8 bits per channel.

For more information about WITAS see their home page [24].

1.2 Problem description

The problem that is addressed in this thesis is the detection and classification of vehicles in monocular video sequences. This problem can be split into two relatively
separate subproblems, namely detection and classification. Detection techniques are aimed at locating interesting regions within the image that will be classified. It would be possible to classify every available region in the image but since classification techniques generally are fairly sophisticated this task would be computationally infeasible. The aim of the classification is to determine which class a certain vehicle belongs to, for instance a hatchback or a truck.

1.3 Problem conditions

There are a number of difficulties that need to be addressed in order to solve the problem. Some difficulties are inherent to the domain and some are inflicted by the target platform.

1.3.1 Domain description

The data that will be processed are aerial video sequences of daytime traffic scenes. Some of the problems that are imposed by this domain are:

- Varying lighting conditions such as clouds and shadows.
- Varying weather conditions, for instance snow and rain.
- Occlusion by objects, debris and buildings.

Some assumptions about the scenes can be made in order to reduce the complexity of the problem:

- All scenes are outdoor.
- Scenes are relatively open, i.e. no city scenes.
- Video sequences are captured in such a way that the vehicles that are to be detected and tracked have a size of at least $30 \times 30$ pixels.
- The interframe motion is relatively small.

1.3.2 Platform description

The destination platform is the autonomous helicopter developed in the WITAS project. Problems imposed by the destination platform:

- Camera ego motion.
- Need for real-time performance.

Assumptions about the video sequences:

- The camera can be panned, tilted and zoomed.
1.4. REPORT STRUCTURE

• The camera can be moved.
• The camera will deliver data at a maximum rate of 25 frames per second.
• Video data will be available in RGB colour.
• Produced images has low noise.
• Video sequences are captured at an altitude of at least 20m.

1.4 Report structure

The chapters are:

• Introduction - A brief introduction to WITAS and the problem that is to be addressed.
• Significant features - Describes the different features in images that may be used in the process of detection and classification
• Related Work - Other research in this area is presented here.
• Object detection - A method for object detection is proposed and the theoretical aspects are described.
• Object classification - A method for object classification is proposed and described.
• Evaluation of detection techniques - The proposed detection method is evaluated using a number of different approaches.
• Evaluation of classification techniques - The proposed classification method is evaluated.
• Discussion - Contains a small discussion on the relevance of this work and possible future research.
• Summary - A brief summary of the report.
Chapter 2

Significant features

In order to perform classification and detection we need to establish and characterize a number of features. These features should in some way reflect the nature of the objects that we are interested in and hopefully distinguish them from other objects.

What features that are to be considered significant is dependent on both the domain and the platform. For instance, colour features would be totally useless if the platform did not have sensors that retrieve this information. For a description of the domain and the platform see section 1.3.1 and section 1.3.2 respectively.

2.1 Shape

Vehicles has an articulated shape. In a “natural” environment a vehicle would stand out by its sharp corners and straight lines. Shape is a powerful cue in outdoor environments where the vehicle itself is the only man-made object. However, to a vision system the shape of a car is not that different to that of a building. In environments that are cluttered with man-made objects, such as buildings and roads, the shape of an object is not as powerful as in a “natural” environment. Furthermore, if we have a high degree of occlusion the shape cue is harder to interpret.

Shape is a strong cue only if we can assume the absence of other man-made objects of similar shape and no or low occlusion.

2.2 Colour

It is known fact that we humans like to decorate our surroundings in pretty colours and our preference when it comes to vehicles it is not different. This being the case we can expect cars to have colours that vary a great deal from those of our natural surroundings. In many cases this is true but not always, there will of course always be gray and green cars that blend well with the background of our images. To further complicate things there are surroundings that would make even the brightest
car seem bland. Imagine the problem of classifying anything by colour on “The Strip” in Las Vegas.

None the less, colours are important cues if they can be distinguished accurately.

### 2.3 Motion

Perhaps the most characteristic property of vehicles is that they are often moving. Furthermore their movement is fairly predictable. Vehicles generally move along their principal axis which may rotate if the vehicle is turning. Obviously this cue can only be established if the vehicle is moving, but in many scenes the vehicle itself is the only object with significant movement. Also, in many applications we are only interested in moving vehicles.

There are a number of difficulties when estimating movement. Since we are working with a single camera the only information we have available is a projection of the world. The perspective introduced by this projection results in different motion in different parts of the image. This is the so called parallax effect, objects close to the camera undergo a larger translation than objects far from the camera for the same camera movement.

In the absence of other moving objects the movement cue is a very strong one.

### 2.4 Deformability

Most vehicles are rigid objects. Trailers are not strictly rigid but a number of rigid objects connected by joints that are constrained to rotation about a single axis. While this fact in itself is does not provide any means to identify vehicles it lets us make some simplifications in the identification procedure.
Chapter 3

Related work

A lot of research has been devoted to object tracking and classification. This is a topic that has a wide area of application, from surveillance and traffic monitoring to medical imaging. This thesis will mainly focus on work that handles the problem at hand, namely traffic monitoring. Within this domain two different classes exist, monitoring from a static camera and monitoring from a moving camera.

3.1 Traffic monitoring from moving platform

There has only been a limited amount of research that is directly related to this field. Besides the WITAS project only three other groups conducting similar research has been found.

3.2 WITAS

For a brief introduction to the WITAS project see section 1.1.

3.3 Cornell

A project on “Detection and Long Term Tracking of Moving Objects in Aerial Video” is being developed at the Computer Science department at the Cornell University [23]. They use affine image-registration and local motion estimation to detect patches that are moving differently from the background. An adaptive model-based tracker then follows the object in subsequent frames.

Their work shows very promising results and it partially overlaps this report. However, the real time performance of their system is questionable within this project. They state a frame rate of 6 frames per second on a small cluster of Pentium II computers. Since this project is to be implemented in on board hardware on an autonomous helicopter, using a cluster of computers is not feasible.
3.4 Carnegie Mellon

The well renowned robotics lab at the Carnegie Mellon University is developing an autonomous helicopter [16]. Information on their project is scarce. From their published material it shows that their applications are not related to traffic monitoring.

For the “1997 International Aerial Robotics Entry” they used a colour discriminating system implemented in analogue hardware for object detection. This was followed by template matching using eigen images described in [22].

3.5 University of Southern California

Another interesting project is being developed at the University of Southern California. Their focus is in aerial images taken from a fairly high altitude. They use a hierarchical, feature based method for inter frame stabilization. Detection of moving objects is done by examination of the motion components of the residual motion. See [5] for further information.

3.6 Traffic monitoring from static platform

There are a number of projects that perform traffic monitoring from a static platform. They often use cameras that are mounted either overhead or at the side of the road. For some interesting examples see [8], [15], [11] or [1].
Chapter 4

Object detection

For the purpose of object detection this thesis will only deal with motion based techniques. The reason behind this is that research regarding detection using colour has already been performed at CVL, see [17]. The motion cue is fairly robust if one can compensate for the ego motion of the camera. One approach to this will be described in the following sections.

4.1 Method outline

This is a rough outline on the method used to detect moving objects. A more thorough treatment of each subtask will be given below.

- Track a number of points in two consequetive images.
- Using the tracked points, find the transformation that the image has undergone.
- Resample one image to fit the other.
- Calculate the difference between the resampled image and the target image.

This method should hopefully indicate moving objects with a high magnitude in the difference image. A graphical illustration of the proposed method is shown in figure 4.1.

4.2 Locating moving objects

In order to distinguish objects from the background we need to be able to estimate what movement is due to camera motion and what is due to object motion. The purpose of this is to in a later stage undo the movement and examine the residual which can be considered to be moving objects.

This can be achieved by using image registration which will be described later. The image registration techniques used in this work is based on tracked points so first the process of feature tracking will be described.
Figure 4.1: A graphical outline of the proposed detection method.
4.3 Feature tracking

There has been extensive studies within this area and it would be cumbersome to cover all methods. This section will focus on one well established method that has been successfully applied to the problem of feature tracking in aerial image sequences.

4.3.1 Kanade-Lucas-Tomasi tracker

This section will briefly introduce the Kanade-Lucas-Tomasi (KLT) [20] tracker. The KLT tracker works with affine motion. They introduce an affine motion field:

\[ \delta = D\mathbf{x} + \mathbf{d} \]  

where

\[ D = \begin{pmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{pmatrix} \]  

is a deformation matrix and \( \mathbf{d} \) is a translation vector. For convenience we introduce an affine transformation matrix:

\[ A = D + I \]  

where \( I \) is the \( 2 \times 2 \) identity matrix. Given a reference image \( I_r \) and a target image \( I_t \), we ideally would like the following equality to be true:

\[ I_r(A\mathbf{x} + \mathbf{d}) = I_t(\mathbf{x}) \]  

At a global scope this might not be the case but in small regions it is a fair assumption. The process of tracking is achieved by determining six parameters, namely the elements of the deformation matrix \( D \) and the components of the translation vector \( \mathbf{d} \). It is argued in [20] that even when the affine motion model is good, accurate estimation of these parameters is hard to achieve. They also argue that for most tracking applications inter frame motion is small so reducing the affine motion model to a model using only translation is reasonable. This reduction yields:

\[ I_r(\mathbf{x} + \mathbf{d}) = I_t(\mathbf{x}) \]  

The process of tracking is achieved by determining the components of the translation vector \( \mathbf{d} \). The equality in equation (4.5) is rarely satisfied, instead we formulate the problem to finding \( \mathbf{d} \) that minimize the squared difference residual:

\[ \varepsilon = \int \int_W [I_r(\mathbf{x} + \mathbf{d}) - I_t(\mathbf{x})]^2 w(\mathbf{x}) d\mathbf{x} \]  

where \( W \) is the region and \( w(\mathbf{x}) \) is a windowing function. Windowing functions can be chosen rather arbitrarily, if one wishes to emphasize the central parts of the
region a Gaussian shaped window could be chosen, if all parts should be equally important one would choose an identity window where \( w(x) = 1 \).

In order to find the minimum of \( \varepsilon \) we differentiate \( \varepsilon \) with respect to \( \mathbf{d} \) and set it to zero and solve the resulting equation:

\[
\frac{\partial \varepsilon}{\partial \mathbf{d}} = 0
\]  

(4.7)

In [2] it is shown that solving equation (4.7) is approximately equivalent to solving:

\[
Z \mathbf{d} = \mathbf{e}
\]  

(4.8)

where \( Z \) is the \( 2 \times 2 \) matrix given by:

\[
Z = \int \int_W g(x) g^T(x) w(x) dx
\]  

(4.9)

and \( \mathbf{e} \) is the \( 2 \times 1 \) vector:

\[
\mathbf{e} = 2 \left( \int \int_W (I_r(x) - I_t(x)) w(x) dx \right) \mathbf{g}
\]  

(4.10)

where \( \mathbf{g} \) is:

\[
\mathbf{g} = \left( \frac{\partial}{\partial x} (I_r + I_t) \right)
\]  

\[
\left( \frac{\partial}{\partial y} (I_r + I_t) \right)
\]  

(4.11)

The criterion for selecting good features to track is given implicitly by the tracker definition. In order to have an accurate estimation of \( \mathbf{d} \) it is required that \( Z \) in the equation (4.8) is well conditioned. It is also important that \( Z \) is insensitive to noise. The well conditioned requirement implies that both eigenvalues of \( Z \) need to be of roughly the same magnitude. The noise insensitivity requirement means that both eigenvalues need to be fairly large. If the two eigenvalues of \( Z \) are called \( \lambda_1 \) and \( \lambda_2 \) we can according to [20] use the selection criterion:

\[
\min(\lambda_1, \lambda_2) > \lambda
\]  

(4.12)

where \( \lambda \) is some predefined threshold. Since the largest eigenvalue \( \lambda_1 \) is bounded by the intensity values of the input images a sufficiently large \( \lambda_2 \) will ensure that the matrix \( Z \) is well conditioned.

There are some methods to achieve subpixel accuracy. In [18] a quadratic polynomial is fitted to the error function, \( \varepsilon \), in a \( 3 \times 3 \) neighborhood around the integer position with the minimum residual. The resulting surface is then solved for the minimum value. As shown in [18] the results achieved by this method are not great.

For a more extensive discussion and experimental results of the Kanade-Lucas-Tomasi tracker see [20]. Extensions to allow the KLT tracker to more robustly handle outliers are proposed in [21]. The KLT tracker has been implemented with good results within the “Detection and Long Term Tracking of Moving Objects in Aerial Video”-project developed at the Cornell university [23].
4.4 Image registration

The purpose of image registration is to find and undo any movement that has occurred between two frames. This includes translational, rotational and parallax movement. Typically one searches some transform space to find a transform that transforms the reference image to the target image.

Among the techniques available there are basically two different types, intensity based and feature based. Intensity based techniques consider all pixels in the image while feature based only considers certain key points. Using the hardware of today, good real time performance is infeasible with most, if not all, intensity based techniques. Since real time performance is an issue here only feature based techniques will be given further attention. For a comprehensive survey of image registration techniques see [4].

Given the fact that we have successfully managed to track a number of feature points in the image sequence we now need to address the task of finding the mapping between two images.

The mapping between a reference image $I_r$ and a target image $I_t$ can be expressed as:

$$I_t(x) = g(I_r(f(x)))$$

(4.13)

where $f$ is a coordinate transformation and $g$ is an intensity transformation [4]. If we assume that no intensity changes are present, i.e. $g(x) = x$, equation (4.13) can be simplified to:

$$I_t(x) = I_r(f(x))$$

(4.14)

This equation gives us a mapping of the points in the two images, namely:

$$y = f(x)$$

(4.15)

where $x$ is a point in the image $I_r$ and $y$ is the corresponding point in $I_t$. Ideally the relation (4.15) is true for all points in the two images and obviously it will then also be true for our set of tracked feature points.

Let $X$ be the set of reference positions:

$$X = \{x_1, x_2, \ldots, x_n\}$$

(4.16)

and $Y$ be the set of target positions:

$$Y = \{y_1, y_2, \ldots, y_n\}$$

(4.17)

the relation (4.15) should be true for all points in these sets:

$$y_k = f(x_k), \forall k \in \{1, \ldots, n\}$$

(4.18)

To determine the transformation between two images we need to find the point mapping $f$. Finding the exact mapping is an impossible task in all but the most trivial cases. To still be able to achieve some reasonable processing we need to introduce some model of the mapping.
4.4.1 Translational transformation

The simplest case is to state that the image has only undergone a translational transformation. This model is fairly good for frames that only have a small temporal difference, especially for small regions. In a translational model we assume the mapping:

\[ f(x) = x + d \]  

(4.19)

We assume that a number of feature points has been tracked successfully between two images. Let \( X \) and \( Y \) be the sets of reference and target positions respectively. The relationship between the elements of the sets \( X \) and \( Y \) is thus given by:

\[ y_k = x_k + d, \forall k \in \{1, \ldots, n\} \]  

(4.20)

In order to solve this equation we only need to know one feature point in the two images. However, since we will have errors in the positions of our feature points we will use an over determined equation system and solve it using least-squares. The least-squares solution to equation (4.20) is simply the mean of the difference between the positions in the sets:

\[ d = \frac{1}{n} \sum_{k=1}^{n} y_k - x_k \]  

(4.21)

4.4.2 Affine transformation

A logical extension to the translational model is to allow the image to undergo an affine transformation. An affine transformation will not only capture translation but also rotation, mirroring, scaling and skewing. With an affine transformation model we get the point mapping:

\[ f(x) = Ax + d \]  

(4.22)

where \( A \) is a \( 2 \times 2 \) matrix.

Let \( X \) and \( Y \) be the set of reference positions and the set of target positions respectively. The relationship between the elements of the sets \( X \) and \( Y \) is given by:

\[ y_k = Ax_k + d, \forall k \in \{1, \ldots, n\} \]  

(4.23)

In order to solve equation (4.23) we need to know the position of at least 3 feature points. Furthermore the motion of the feature points needs to be such that equation (4.23) has a non singular solution.

As in the translational case we use an over determined system to get more accurate solutions. The components of a given feature point can be expressed by:

\[ y_1 = a_{11}x_1 + a_{12}x_2 + d_1 \]  

(4.24)

\[ y_2 = a_{21}x_1 + a_{22}x_2 + d_2 \]  

(4.25)
where \( a_{ij} \) are the components of \( A \) and \( d_i \) are the components of \( d \). When we consider all the feature points we get a linear equation system with \( 2n \) equations. Once again we’ll use least squares to solve the over determined equation system. We need to reformulate equation (4.23) somewhat to make it suitable for solving using least-squares:

\[
S \mathbf{p} = \mathbf{y}
\]  

(4.26)

with:

\[
S_k = \begin{pmatrix}
x_{1k} & x_{2k} & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_{1k} & x_{2k} & 1
\end{pmatrix}
\]  

(4.27)

and:

\[
\mathbf{p} = \begin{pmatrix}
a_{11} & a_{12} & d_1 & a_{21} & a_{22} & d_2
\end{pmatrix}^T
\]  

(4.28)

The least-squares solution to equation (4.26) is given by:

\[
\mathbf{p} = \left( \sum_k S_k^T S_k \right)^{-1} \left( \sum_k S_k^T \mathbf{y}_k \right)
\]  

(4.29)

### 4.4.3 Quadratic transformation

Further refinement of our model can be achieved if we include quadratic terms. The choice of a quadratic model is not an obvious one since the transformations involved in a camera system are ideally linear. However, the image transformations between frames are not affine and we do in some way hope that the quadratic terms will capture this behavior. One advantage of a quadratic model over a linear one is that the effects of lens distortion can hopefully be reduced.

We extend our feature positions to include quadratic terms:

\[
\mathbf{x}_q = \begin{pmatrix}
x_1^2 & x_2^2 & x_1 x_2 & x_1 & x_2
\end{pmatrix}^T
\]  

(4.30)

The point mapping can then be expressed as:

\[
f(x) = Q \mathbf{x}_q + \mathbf{d}
\]  

(4.31)

where \( Q \) is a \( 2 \times 5 \) matrix. And the correspondence between our sets \( X \) and \( Y \) of feature positions is:

\[
\mathbf{y} = Q \mathbf{x}_q + \mathbf{d}
\]  

(4.32)

In order to solve this system we need to know the position of at least 6 feature points. The manner in which it is solved is similar to the Affine case. We first rewrite it in a manner that is suitable for least-squares solution:

\[
S_k \mathbf{p} = \mathbf{y}_k
\]  

(4.33)
with:

\[ S_k = \begin{pmatrix} x_{1k}^2 & x_{2k}^2 & x_{1k}x_{2k} & x_{1k} & x_{2k} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & x_{1k}^2 & x_{2k}^2 & x_{1k}x_{2k} & x_{1k} & x_{2k} & 1 \ \end{pmatrix} \]  

(4.34)

and:

\[ p = \begin{pmatrix} q_{11} & \cdots & q_{15} & d_1 & q_{21} & \cdots & q_{25} & d_2 \end{pmatrix}^T \]  

(4.35)

The least-squares solution to equation (4.33) is given by:

\[ p = \left( \sum_k S_k^T S_k \right)^{-1} \left( \sum_k S_k^T y_k \right) \]  

(4.36)

### 4.4.4 Homographic transformation

Our previous models have only considered transformations to be restricted to the image plane, distortions introduced by perspective projection are not considered. When a 3D scene is projected to a 2D image we lose one dimension. In order to be able to perform meaningful reasoning about the scene we therefore need to introduce some sort of constraint. One popular and well motivated choice is that of a flat ground, that is, all points in the scenes are located on a single plane. Obviously this is not the case for entire 3D scenes but if we are careful when selecting feature points it can be true for those. The process of finding the mapping between two images under the flat ground constraint is known as homography estimation.

Consider the generic transformation from world coordinates to camera coordinates, this can in homogeneous coordinates be expressed as:

\[
    x_c = \begin{pmatrix} x_{c1} \\ x_{c2} \\ x_{c3} \\ 1 \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{w1} \\ x_{w2} \\ x_{w3} \\ 1 \end{pmatrix} = C x_w \]  

(4.37)

where \( x_c \) are camera space coordinates, \( C \) is a \( 4 \times 4 \) matrix expressing the mapping from world space coordinates to camera space coordinates and \( x_w \) are world space coordinates. As shown in appendix A the mapping from camera coordinates to the image coordinates is:

\[
    x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \end{pmatrix} = \frac{f}{x_{c3}} \begin{pmatrix} x_{c1} \\ x_{c2} \end{pmatrix} \]  

(4.38)

where \( f \) is the camera focal length. Using homogeneous coordinate relation (4.38) can be expressed in matrix form:

\[
    x_h = P x_c \]  

(4.39)

where:

\[
    P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \]  

(4.40)
Homogeneous image coordinates relate to normalized image coordinates by:

$$
\begin{pmatrix}
  x_{d1} \\
  x_{d2}
\end{pmatrix}
= \frac{1}{x_{d3}}
\begin{pmatrix}
  x_{h1} \\
  x_{h2}
\end{pmatrix}
$$

(4.41)

The compound mapping from world coordinates to homogeneous image coordinates can be expressed as:

$$
x_h = PCx_c
$$

(4.42)

The compound matrix $PC$ is $3 \times 4$ which means that the mapping cannot be inverted properly. If we introduce the constraint of flat ground, i.e. $x_{w3} = 0$, we can eliminate one column from $PC$ and it can then be inverted. This column is removed by introducing the matrix:

$$
F = \begin{pmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 0 \\
  0 & 0 & 1
\end{pmatrix}
$$

(4.43)

Giving us the final compound mapping:

$$
x_h = PCFx_w
$$

(4.44)

From equation (4.44) it is obvious that:

$$
x_w = (PCF)^{-1}x_h
$$

(4.45)

If we once again examine our two sets of feature positions $X$ and $Y$ we see that for the points in $X$ we have the relationship:

$$
x_h = P_xC_xFx_w
$$

(4.46)

and similarly for the points in $Y$:

$$
y_h = P_yC_yFy_w
$$

(4.47)

Since our tracked features hopefully correspond to the same feature in both images they should have the same world space coordinates and we get:

$$
x_w = y_w
$$

(4.48)

and furthermore:

$$
y_h = P_yC_yF(P_xC_xF)^{-1}x_h
$$

(4.49)

In order to find the transformation that the reference image has undergone we once again have a linear equation system to solve. However, the homogeneous image coordinates $x_h$ are not known, only the normalized image coordinates. For convenience we use:

$$
H = P_yC_yF(P_xC_xF)^{-1}
$$

(4.50)
Using equation (4.41) we get:

\[
\frac{1}{x_{h3}} y_h = H x_i \tag{4.51}
\]

Even though we multiply \( y_h \) with an unknown factor this imposes no problem. Since we’re really interested in normalized image coordinates \( y_i \) the unknown factor will be eliminated in the normalization process in equation (4.41). One case that causes problem is \( x_{h3} = 0 \), which correspond to the point \( x_w \) being very close to the camera, in this paper it is assumed that feature points are chosen in a manner that eliminate this problem. Let:

\[
y'_h = \frac{1}{x_{h3}} y_h
\]

We then get a linear equation system to solve, namely:

\[
y'_h = H x_i \tag{4.52}
\]

If we examine equation (4.52) more closely we see that:

\[
y'_{h1} = h_{11} x_{i1} + h_{12} x_{i2} + h_{13}
\]

\[
y'_{h2} = h_{21} x_{i1} + h_{22} x_{i2} + h_{23}
\]

\[
y'_{h3} = h_{31} x_{i1} + h_{32} x_{i2} + h_{33}
\]

where \( h_{ij} \) are the components of \( H \). However, in this equation system the \( y_{h3} \) components of the homogeneous image coordinates are unknown. But since we know that equation (4.41) is valid we can express the equation system using normalized image coordinates as:

\[
y_{i1} = \frac{h_{11} x_{i1} + h_{12} x_{i2} + h_{13}}{h_{31} x_{i1} + h_{32} x_{i2} + h_{33}} \tag{4.56}
\]

\[
y_{i2} = \frac{h_{21} x_{i1} + h_{22} x_{i2} + h_{23}}{h_{31} x_{i1} + h_{32} x_{i2} + h_{33}} \tag{4.57}
\]

or equivalent:

\[
h_{33} y_{i1} = h_{11} x_{i1} + h_{12} x_{i2} + h_{13} - h_{31} x_{i1} y_{i1} - h_{32} x_{i2} y_{i1} \tag{4.58}
\]

\[
h_{33} y_{i2} = h_{21} x_{i1} + h_{22} x_{i2} + h_{23} - h_{31} x_{i1} y_{i2} - h_{32} x_{i2} y_{i2} \tag{4.59}
\]

This equation system is solvable except for an unknown scale factor, \( h_{33} \). In practice this scale factor is not interesting for image warping due to the normalization process in equation (4.41). If we set \( h_{33} = 1 \) we can now solve the problem using least-squares. We thus need to solve 8 unknown parameters. In order to do this uniquely at least 4 feature points has to be tracked. Using:
4.4. IMAGE REGISTRATION

\[ S_k \mathbf{p} = \mathbf{y}_i \]  \hspace{1cm} (4.60)

with:

\[ S_k = \begin{pmatrix} x_{1i} & x_{2i} & 1 & 0 & 0 & 0 & -x_{1i}y_{1i} & -x_{2i}y_{1i} \\ 0 & 0 & 0 & x_{4i} & x_{5i} & 1 & -x_{1i}y_{2i} & -x_{2i}y_{2i} \end{pmatrix} \]  \hspace{1cm} (4.61)

and:

\[ \mathbf{p} = ( h_{11} \ h_{21} \ h_{31} \ h_{12} \ h_{22} \ h_{32} \ h_{13} \ h_{23} )^T \]  \hspace{1cm} (4.62)

The least-squares solution to equation (4.60) is given by:

\[ \mathbf{p} = \left( \sum_k S_k^T S_k \right)^{-1} \left( \sum_k S_k^T \mathbf{y}_i \right) \]  \hspace{1cm} (4.63)

4.4.5 Outlier detection and handling

Due to the rather unintelligent nature of feature tracking it is unavoidable that we sometimes will be tracking points that for some reason are bad for our purpose. In the current application we do not want to track points in the image that for instance is on a car since we are only trying to estimate the motion of the static background. To further complicate things feature points may be lost due to occlusion or other factors. The method for detection and handling such points, called outliers, is the same as the one used in [6].

Given our motion model, that has been estimated using one of the methods above, we have determined the relation between our tracked points using equation (4.15). For convenience it will be repeated here. The relation between the points in our reference image, \( \mathbf{x} \), and the points in our target image, \( \mathbf{y} \), is:

\[ \mathbf{y} = f(\mathbf{x}) \]  \hspace{1cm} (4.64)

Outlier detection is based on the assumption that any outlier points will not fulfill this relation. Recall that we have two sets of tracked points, \( X \) and \( Y \), for each point in these sets we now calculate the distance between the tracked and the predicted point:

\[ d_k = \| \mathbf{y}_k - f(\mathbf{x}_k) \|_2 \]  \hspace{1cm} (4.65)

It is now tempting to simply state that all \( d_k \) exceeding some threshold is to be considered an outlier and thus it should be ignored. Indeed this approach might be work well. However, since the outliers themselves have been used when estimating the motion field it might be so that the presence of one outlier has affected the motion field so much that another point which actually isn’t an outlier is considered as such. The algorithm used in [6] and in this work is iterative.

1. Estimate motion field.
2. Calculate \( d_k \).
3. Find \( l \) such that \( d_l \geq d_k \forall k \).

4. If \( d_l > \text{thr} \) consider \( x_l \) to be an outlier, remove it and restart from step 1.

5. Otherwise we are done.

Admittedly this algorithm is computationally expensive but most of the outliers will be detected fairly fast and it should not require many iterations to arrive at a stable solution. There are many ways to improve this algorithm but that is beyond the scope of this thesis.

### 4.5 Image warping

Performing image warping is the task of applying a known transformation to an image. In the transformations studied above this is no more than resampling. Most resampling approaches uses the Sampling Theorem as a base for their arguments. One might argue that images are not really constructed as a linear combination of sinc shaped functions and that reasoning about them in terms of Fourier transforms is invalid. A better approach would be to properly analyse the process that is actually performed when capturing images with a camera. However, to fully examine this process is beyond the scope of this document and in practice the concepts from Fourier transforms works pretty well even when dealing with images.

Recall from section 4.4 the equation (4.14):

\[
I_t(x) = I_r(f(x))
\]

which imposed the point mapping:

\[
y = f(x)
\]

Where \( y \) are coordinates in the target image and \( x \) are coordinates in the reference image. However, since we are interested in finding the source image for every point in our target image we need the inverse mapping of equation (4.67) given by:

\[
x = f^{-1}(y)
\]

In many of the cases above inversion is straightforward, for instance in the translational case described in section (4.4.1) setting \( d_t = -d \) gives us the inverse transform. Other cases have a more ambiguous inversion, for example the quadratic case. A convenient way of solving this is mearly to solve the least-squares systems introduced for the inverse mapping. This process is simple, just switch all references to target with reference and vice versa.

The traditional way to treat resampling is defined by the Sampling Theorem. The Sampling Theorem states that: If we sample a signal at a sampling frequency
greater than twice the maximum frequency of the signal we can reconstruct it perfectly. Let \( x_k \) be a set of points located on integer positions in \( \mathbb{R}^2 \). Sampling can be expressed as:

\[
\hat{I}_k = \int_{x \in \mathbb{R}^2} I(x) \delta(x - x_k) \, dx = I(x_k)
\]  

(4.69)

If the input signal is band limited the original signal can then be perfectly reconstructed by:

\[
I(x) = \sum_{k \in \mathbb{Z}} \hat{I}_k \text{sinc}(x_1 - x_{k1}) \text{sinc}(x_2 - x_{k2})
\]

(4.70)

Consider a reference image \( I_r \) that we wish to resample to an image \( I_t \), according to equation (4.69) we get:

\[
\hat{I}_{kt} = I_t(x_k)
\]

(4.71)

Since we know the point mapping between \( I_t \) and \( I_r \) we get:

\[
\hat{I}_{kt} = I_r(f^{-1}(x_k))
\]

(4.72)

However, the true reference function \( I_r \) is not known. But using equation (4.70) we get:

\[
\hat{I}_{kt} = \sum_{l \in \mathbb{Z}} \hat{I}_k \text{sinc}(f^{-1}(x_{k1}) - x_{l1}) \text{sinc}(f^{-1}(x_{k2}) - x_{l2})
\]

(4.73)

The \( \text{sinc} \) function is not really suitable for practical use due to its infinite distribution. A number of approximations have been proposed, most popular are cubic spline and bilinear. When real time performance is an issue it is advantageous to have a small distribution of the interpolation function. Linear interpolation has computational advantage over the cubic spline and still gives us fair quality. The distribution of the different interpolation kernels are shown in figure 4.2. Results from a small resampling experiment are presented in section 6.2. For a more thorough discussion on resampling see [10].
Figure 4.2: Illustration of different interpolation kernels.
Chapter 5

Object classification

This chapter will describe a method to classify vehicles. The method is based on a flexible wireframe model introduced by D. Koller in [12]. In order to determine the class of a vehicle a number of parameters describing vehicle shape needs to be determined, this is done using by modifying the parameters of an initial guess so that the model will fit the image.

The different parts of this method are described in more detail below.

5.1 Proposed method

An initial guess of the pose and shape of a vehicle is instantiated. The resulting model line segments is then matched to data line segments extracted from video data. The initial guess is then modified to minimize the error between the model and the data line segments. A illustration of the proposed method is shown in 5.1.

A rough outline for an algorithm is:

1. Capture a video frame and extract line segments.
2. Instantiate the model using initial pose and shape estimate.
3. Match model lines to extracted lines.
4. Modify the initial guess to minimize the error between the matched lines.
5. If necessary, repeat from 2.
6. Classify the estimated parameters using a suitable classifier.

The big difference between this method and the one proposed by Koller in [12] is the error minimization step. While Koller uses an iterated extended Kalman filter to perform the minimization this work will choose the more simple approach of using Newton iteration for minimization. The error measure used is also different.
CHAPTER 5. OBJECT CLASSIFICATION

Figure 5.1: Illustration of the proposed classification method.
5.2 Line extraction

The code for extracting the line segments was implemented by Peter Kovesi at The University of Western Australia. No related publications were found, unless one considers the code itself documentation. The matlab source code is available at [13].

5.3 Vehicle prototype

In order to be able to distinguish between different types of vehicles a flexible vehicle prototype is used. The model used is the one introduced by D. Koller in [12]. It consists of 12 parameters describing the vehicle shape. The influence of the different parameters is shown in figure 5.2. Several different classes of vehicles can be modeled as seen in figure 5.3. Using the same notation as in [12] we thus get a vector with the 12 shape parameters describing a certain vehicle.

\[
\mathbf{s} = ( b_1, b_2, b_3, d_1, d_2, d_3, f_1, f_2, h_1, h_2, a_1, a_2 )^T \quad (5.1)
\]

Definition of different vehicle classes is also given in [12]. The expected value \( \mu_i \) and covariance \( \gamma_i \) of the different shape parameters for the different classes can be found in figure 5.3.

5.4 Introducing the prototype vector

The 12 shape parameters are combined with translation and rotation of the prototype to form the final vehicle model. Since the operating platform is assumed to be a PTZ (pan, tilt and zoom) camera, rotation is limited to two axes.

Combining the 12 shape parameters with 3 translational and 2 rotational parameters yields a 17 dimensional parameter vector:

\[
\mathbf{x} = 
\begin{pmatrix}
  t_x \\
  t_y \\
  t_z \\
  \alpha \\
  \beta \\
  \mathbf{s}
\end{pmatrix} \quad (5.2)
\]

where \( t_x, t_y \) and \( t_z \) are translations along the x-, y- and z-axis respectively. \( \alpha \) is rotation about the x-axis and \( \beta \) is rotation about the y-axis. As mentioned earlier we do not consider rotation about the z-axis since we are using a PTZ-camera.
Figure 5.2: The vehicle model used.
5.4. INTRODUCING THE PROTOTYPE VECTOR

![Vehicle classes and their expected parameter values and variances.](image)

<table>
<thead>
<tr>
<th>s</th>
<th>Hatchback</th>
<th>Sedan</th>
<th>Stationwagon</th>
<th>Mini-van</th>
<th>Pick-up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu_s$</td>
<td>$\sigma_s$</td>
<td>$\mu_s$</td>
<td>$\sigma_s$</td>
<td>$\mu_s$</td>
</tr>
<tr>
<td>bl</td>
<td>3.80</td>
<td>0.80</td>
<td>4.40</td>
<td>1.00</td>
<td>4.40</td>
</tr>
<tr>
<td>bb</td>
<td>1.50</td>
<td>0.10</td>
<td>1.60</td>
<td>0.10</td>
<td>1.60</td>
</tr>
<tr>
<td>bh</td>
<td>0.30</td>
<td>0.10</td>
<td>0.30</td>
<td>0.10</td>
<td>0.30</td>
</tr>
<tr>
<td>dl</td>
<td>1.50</td>
<td>0.30</td>
<td>1.60</td>
<td>0.30</td>
<td>2.60</td>
</tr>
<tr>
<td>db</td>
<td>1.30</td>
<td>0.10</td>
<td>1.40</td>
<td>0.10</td>
<td>1.40</td>
</tr>
<tr>
<td>dh</td>
<td>1.40</td>
<td>0.10</td>
<td>1.40</td>
<td>0.10</td>
<td>1.40</td>
</tr>
<tr>
<td>dk</td>
<td>1.60</td>
<td>0.30</td>
<td>1.60</td>
<td>0.30</td>
<td>1.60</td>
</tr>
<tr>
<td>fl</td>
<td>1.00</td>
<td>0.30</td>
<td>1.10</td>
<td>0.30</td>
<td>1.10</td>
</tr>
<tr>
<td>fh</td>
<td>0.50</td>
<td>0.10</td>
<td>0.50</td>
<td>0.10</td>
<td>0.50</td>
</tr>
<tr>
<td>hl</td>
<td>0.00</td>
<td>0.20</td>
<td>0.70</td>
<td>0.10</td>
<td>0.00</td>
</tr>
<tr>
<td>hh</td>
<td>0.60</td>
<td>0.10</td>
<td>0.55</td>
<td>0.10</td>
<td>0.60</td>
</tr>
<tr>
<td>ah</td>
<td>0.60</td>
<td>0.10</td>
<td>0.60</td>
<td>0.10</td>
<td>0.60</td>
</tr>
</tbody>
</table>

Figure 5.3: Vehicle classes and their expected parameter values and variances.
5.5 Model instantiation

The instantiation of the model is a straightforward projection of the 3d-lines onto the image plane. The common pin-hole camera model is used, a brief description of this model can be found in appendix A.

Since we will be working a lot with homogeneous coordinates later on it is convenient to transform the shape vector into homogeneous coordinates. This is achieved by:

$$s_h = \begin{pmatrix} s \\ 1 \end{pmatrix}$$

(5.3)

Using simple linear expressions we can calculate the vertices of the vehicle model in object space coordinates. Using the homogeneous shape vector $s_h$ the vertex $i$ is:

$$o = M_t s_h$$

(5.4)

The matrix $M_t$ is an matrix that describes the relationship between the different parameters in the shapevector and the the vertex $i$. The coordinates of the resulting point $o$ are in homogeneous coordinates:

$$o = \begin{pmatrix} o_1 \\ o_2 \\ o_3 \\ 1 \end{pmatrix}$$

(5.5)

Transforming object space coordinates into camera space coordinates is done by applying the translational matrix $T$ and the rotational matrices $R_\beta$ and $R_\alpha$:

$$c = TR_\beta R_\alpha o$$

(5.6)

Where $R_\alpha$ is:

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

(5.7)

and $R_\beta$ is:

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

(5.8)
5.6. INITIAL ESTIMATE

and finally $T$:

$$
T = \begin{pmatrix}
1 & 0 & 0 & t_x \\
0 & 1 & 0 & t_y \\
0 & 0 & 1 & t_z \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

(5.9)

Projection from camera space coordinates to image plane coordinates is done using a projective matrix:

$$
q = Pc
$$

(5.10)

with the projection matrix:

$$
P = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & \frac{1}{f}
\end{pmatrix}
$$

(5.11)

where $f$ is the focal length of our camera model.

Since graphic displays normally has the origin in the upper left corner and we want it to be at the center of the image we apply a translation place our origin there:

$$
h = Dq
$$

(5.12)

where $D$ is:

$$
D = \begin{pmatrix}
1 & 0 & 0 & x_m \\
0 & 1 & 0 & y_m \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

(5.13)

$x_m$ and $y_m$ are the coordinates of the image center on the screen.

The last step of the instantiation is to normalize the homogeneous coordinates:

$$
p = \frac{1}{h_4} h
$$

(5.14)

where $h_4$ is the fourth component of $h$, the homogeneous coordinate.

5.6 Initial estimate

From the image sequence the detection algorithm will give us a rough indication of the position within the image of any moving objects. Using this information one should make an educated guess about the location and orientation of the object. The vehicle parameters could be instantiated using some a priori knowledge. However, in this thesis all initial estimates will be handcrafted since the main purpose is to determine the applicability of the proposed method. An promising approach for obtaining an initial estimate is described in [12].
5.7 Parameter fitting

The process of parameter fitting can be described as the process of finding the parameters which minimize a specific error. In the current case the parameters to be fitted are the pose- and shape-parameters of the vehicle prototype. The error measure is based on the difference between observed line segments and the line segments instantiated from the vehicle model.

5.7.1 Error measure

In order to minimize the error between our model instantiation and our observed data segments we first need to define this error measure. The error measure used here is closely related to the one used in [14]. The big difference is that an error measure of segment midpoints is also included. In [14] it is argued that the midpoint measure is unreliable due to uncertainties of the end positions of the extracted line. Occlusion and other difficulties in the extraction process will effect the precision of the end positions. However, when only a few line matches can be established the inclusion of the midpoint distance measure should increase performance.

In [12] D. Koller uses an error measure based on the midpoints, orientations and lengths of line segments. While this is a natural, and probably well suited, way to define the error the differentiation of the errors needed later will be fairly complex.

The error is thus composed by three components. Two error components are defined as the perpendicular distance from endpoints of a model segment to a line defined by an observed segment. The third error component is defined as the distance between the midpoints of the two segments projected onto the observed segment. For an illustration see figure 5.4. In this figure and in the following section \( \mathbf{s} \) and \( \mathbf{e} \) refers to the start- and end-points of a model line segment. \( \mathbf{m} \) is the midpoint of that segment defined as:

\[
\mathbf{m} = \frac{1}{2}(\mathbf{s} + \mathbf{e}) \quad (5.15)
\]

The point \( \mathbf{m}^* \) is the midpoint of an observed line segment and is defined the same way.

Given the vector \( \mathbf{n} \), that is a normalized vector pointing in the normal direction of the observed line segment it is easy to construct a vector that is orthonormal to \( \mathbf{n} \). This can be achieved by:

\[
\mathbf{o} = \begin{pmatrix} \hat{n}_2 \\ -\hat{n}_1 \end{pmatrix} \quad (5.16)
\]

It is obvious that \( \mathbf{o} \) is orthonormal to \( \mathbf{n} \).

Let us study the calculation of an endpoint error. Using notations from figure 5.4 we see that:

\[
\varepsilon_s = \mathbf{n}(\mathbf{s} - \mathbf{m}^*) = \hat{n}\mathbf{s} - \hat{n}\mathbf{m}^* \quad (5.17)
\]
Figure 5.4: Illustration of the error measure.
Similarly, for the end point of the line the error is:

$$\epsilon_e = \hat{n}e - \hat{n}m^* \quad (5.18)$$

The midpoint error is the projected distance between the midpoints projected onto the observed line. More specifically:

$$\epsilon_m = \hat{o}(m - m^*) = \hat{o}m - \hat{o}m^* \quad (5.19)$$

using equation (5.15) we get:

$$\epsilon_m = \frac{1}{2} \hat{0}e + \frac{1}{2} \hat{0}e - \hat{o}m^* \quad (5.20)$$

The three errors are combined in an error vector $\epsilon$, defined as:

$$\epsilon = \begin{pmatrix} \epsilon_s \\ \epsilon_e \\ \epsilon_m \end{pmatrix} \quad (5.21)$$

where $\epsilon_s$ and $\epsilon_e$ are the perpendicular distances at the start- and end-points of a segment respectively. $\epsilon_m$ is the parallel distance between the midpoints of the segments.

### 5.7.2 Calculation of the Jacobian

Many techniques for error minimization require computation of the Jacobian. In section 5.7.3 Newton minimization will be introduced and the Jacobian will then be necessary.

The Jacobian matrix is defined as:

$$J = \begin{pmatrix} \frac{\partial \epsilon_1}{\partial x_1} & \cdots & \frac{\partial \epsilon_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial \epsilon_m}{\partial x_1} & \cdots & \frac{\partial \epsilon_m}{\partial x_m} \end{pmatrix} \quad (5.22)$$

Where $\epsilon_i$ are the errors and $x_j$ are the components of the parameter vector $\mathbf{x}$ defined in equation (5.2). The different kinds of parameters (translational, rotational and shape) in the parameter vector requires different approaches so we’ll study each of them separately. However, some results are needed in order to do this in a simple manner.

First we’ll observe that the error $\epsilon$ is a function of the start- and end-points of the model segment. That is:

$$\epsilon = \epsilon(s, e) \quad (5.23)$$
5.7. PARAMETER FITTING

Consequently, if we differentiate \( \varepsilon \) we get:

\[
\begin{align*}
\frac{\partial \varepsilon}{\partial x_j} &= \frac{\partial \varepsilon}{\partial s_1} \frac{\partial s_1}{\partial x_j} + \frac{\partial \varepsilon}{\partial s_2} \frac{\partial s_2}{\partial x_j} + \frac{\partial \varepsilon}{\partial e_1} \frac{\partial e_1}{\partial x_j} + \frac{\partial \varepsilon}{\partial e_2} \frac{\partial e_2}{\partial x_j} \\
\end{align*}
\]  
(5.24)

where \( s_1 \) and \( s_2 \) are the coordinates of the starting point \( s \) and \( e_1 \) and \( e_2 \) are the coordinates of the point \( e \).

Let us study the differentiation of the different components of \( \varepsilon \). First we will examine \( \varepsilon_s \):

\[
\frac{\partial \varepsilon_s}{\partial x_j} = \frac{\partial \varepsilon_s}{\partial s_1} \frac{\partial s_1}{\partial x_j} + \frac{\partial \varepsilon_s}{\partial s_2} \frac{\partial s_2}{\partial x_j} + \frac{\partial \varepsilon_s}{\partial e_1} \frac{\partial e_1}{\partial x_j} + \frac{\partial \varepsilon_s}{\partial e_2} \frac{\partial e_2}{\partial x_j} \\
\]  
(5.25)

since \( \varepsilon_s \) is a function of only \( s_1 \) and \( s_2 \) the third and fourth terms will be zero. Examining one part of the first term and using equation (5.17) we get:

\[
\frac{\partial \varepsilon_s}{\partial s_1} = \frac{\partial}{\partial s_1} (\hat{n}s - \hat{n}m^*) = \hat{n}_1 \\
\]  
(5.26)

Similarly we get:

\[
\frac{\partial \varepsilon_s}{\partial s_2} = \hat{n}_2 \\
\]  
(5.27)

Where \( \hat{n}_1 \) and \( \hat{n}_2 \) are the components of \( \hat{n} \). Thus equation (5.25) can be written as:

\[
\frac{\partial \varepsilon_s}{\partial x_j} = \begin{pmatrix} \hat{n}_1 & \hat{n}_2 & 0 & 0 \end{pmatrix} \\
\]  
(5.28)

If we apply the same process to \( \varepsilon_c \) we will derive the expression:

\[
\frac{\partial \varepsilon_c}{\partial x_j} = \begin{pmatrix} 0 & 0 & \hat{n}_1 & \hat{n}_2 \end{pmatrix} \\
\]  
(5.29)

Let us now examine the differentiation of \( \varepsilon_m \):

\[
\frac{\partial \varepsilon_m}{\partial x_j} = \frac{\partial \varepsilon_m}{\partial s_1} \frac{\partial s_1}{\partial x_j} + \frac{\partial \varepsilon_m}{\partial s_2} \frac{\partial s_2}{\partial x_j} + \frac{\partial \varepsilon_m}{\partial e_1} \frac{\partial e_1}{\partial x_j} + \frac{\partial \varepsilon_m}{\partial e_2} \frac{\partial e_2}{\partial x_j} \\
\]  
(5.30)

In the same manner as before we only examine a part of this expression, that is:

\[
\frac{\partial \varepsilon_m}{\partial s_1} = \frac{\partial}{\partial s_1} (\frac{1}{2} \hat{\varepsilon}s + \frac{1}{2} \hat{\varepsilon}e - \hat{\varepsilon}m^*) = \frac{1}{2} \hat{\varepsilon}_1 \\
\]  
(5.31)
The other parts of the expression is calculated in the same way, this yields:

\[
\frac{\partial \varepsilon_m}{\partial s_2} = \frac{1}{2} \dot{\theta}_2 \tag{5.32}
\]
\[
\frac{\partial \varepsilon_m}{\partial e_1} = \frac{1}{2} \dot{\theta}_1 \tag{5.33}
\]
\[
\frac{\partial \varepsilon_m}{\partial e_2} = \frac{1}{2} \dot{\theta}_2 \tag{5.34}
\]

So, the differentiation of \( \varepsilon_m \) can be written as:

\[
\frac{\partial \varepsilon_m}{\partial x_j} = \frac{1}{2} \begin{pmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_1 & \dot{\theta}_2 \end{pmatrix} \begin{pmatrix} \frac{\partial s_1}{\partial x_j} \\ \frac{\partial s_1}{\partial x_j} \\ \frac{\partial s_1}{\partial x_j} \\ \frac{\partial s_1}{\partial x_j} \end{pmatrix} \tag{5.35}
\]

Recall from equation (5.14) the relationship between normalized image coordinates and homogeneous image coordinates. Given an image point \( s \) and the corresponding unnormalized point \( h_s \), differentiation of one of the coordinates \( s \) is:

\[
\frac{\partial s_1}{\partial h_1} \frac{\partial h_1}{\partial x_j} + \frac{\partial s_1}{\partial h_2} \frac{\partial h_2}{\partial x_j} + \frac{\partial s_1}{\partial h_3} \frac{\partial h_3}{\partial x_j} + \frac{\partial s_1}{\partial h_4} \frac{\partial h_4}{\partial x_j} \tag{5.36}
\]

Examination of the different parts of this expression yields:

\[
\frac{\partial s_1}{\partial h_1} = \frac{\partial}{\partial h_1} \left( \frac{1}{h_{4s}} h_{1s} \right) = \frac{1}{h_{4s}} \tag{5.37}
\]
\[
\frac{\partial s_1}{\partial h_2} = 0 \tag{5.38}
\]
\[
\frac{\partial s_1}{\partial h_3} = 0 \tag{5.39}
\]
\[
\frac{\partial s_1}{\partial h_4} = \frac{\partial}{\partial h_1} \left( \frac{1}{h_{4s}} h_{1s} \right) = -\frac{1}{h_{4s}^2} h_{1s} \tag{5.40}
\]

Thus equation (5.36) becomes:

\[
\frac{\partial s_1}{\partial x_j} = \frac{1}{h_{4s}} \frac{\partial h_{1s}}{\partial x_j} - \frac{1}{h_{4s}^2} h_{1s} \frac{\partial h_{4s}}{\partial x_j} \tag{5.41}
\]

Differentiation of \( s_2 \) is done in a similar way. Using matrix notation we get:

\[
\frac{\partial s}{\partial x_j} = \begin{pmatrix} \frac{1}{h_{4s}} & 0 & -\frac{1}{h_{4s}^2} h_{1s} & 0 \\ 0 & \frac{1}{h_{4s}} & -\frac{1}{h_{4s}} h_{2s} & 0 \end{pmatrix} \frac{\partial h_s}{\partial x_j} \tag{5.42}
\]
5.7. PARAMETER FITTING

Doing the same for $e$ yields:

\[
\frac{\partial s}{\partial x_j} = \begin{pmatrix}
\frac{1}{h_{4e}} & 0 & -\frac{1}{h_{4e}}h_{1e} & 0 \\
0 & \frac{1}{h_{4e}} & -\frac{1}{h_{4e}}h_{2e} & 0
\end{pmatrix}
\frac{\partial h_s}{\partial x_j} \tag{5.43}
\]

The relationship between homogeneous image coordinates and camera space coordinates is given by equation (5.10) and equation (5.12). This is a simple matrix multiplication:

\[
h_s = DPc_s \tag{5.44}
\]

Differentiation is then particularly simple

\[
\frac{\partial h_s}{\partial x_j} = DP\frac{\partial c_s}{\partial x_j} \tag{5.45}
\]

We are now ready to study the differentiation of camera coordinates, $c$, with respect to the different parameters, $x_j$.

**Differentiation of the translational parameters**

Recall from equation (5.6) the expression:

\[
c = TR_{\beta}R_\alpha o \tag{5.46}
\]

Differentiation of this expression with respect to $t_x$ yields:

\[
\frac{\partial c}{\partial t_x} = \frac{\partial}{\partial t_x}TR_{\beta}R_\alpha o \tag{5.47}
\]

Since the only part of this expression that depends on $t_x$ is $T$ it holds true that:

\[
\frac{\partial c}{\partial t_x} = \frac{\partial T}{\partial t_x}R_{\beta}R_\alpha o \tag{5.48}
\]

Further inspection of \( \frac{\partial T}{\partial t_x} \) yields:

\[
\frac{\partial T}{\partial t_x} = \frac{\partial}{\partial t_x} \begin{pmatrix}
1 & 0 & 0 & t_x \\
0 & 1 & 0 & t_y \\
0 & 0 & 1 & t_z \\
0 & 0 & 0 & 1
\end{pmatrix} = \begin{pmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix} \tag{5.49}
\]

Differentiation of $e$ with respect to $t_y$ and $t_z$ is calculated in a similar fashion.
CHAPTER 5. OBJECT CLASSIFICATION

Differentiation of the rotational parameters

Starting with equation (5.6):

\[ c = TR_\beta R_\alpha o \]  (5.50)

Differentiation with respect to \( \alpha \) yields:

\[ \frac{\partial c}{\partial \alpha} = \frac{\partial}{\partial \alpha} TR_\beta R_\alpha o \]  (5.51)

As with the translational differentiation the only part of this expression that depends on \( \alpha \) is \( R_\alpha \). Thus:

\[ \frac{\partial c}{\partial \alpha} = TR_\beta \frac{\partial R_\alpha}{\partial \alpha} o \]  (5.52)

If we study \( \frac{\partial R_\alpha}{\partial \alpha} \) further we see:

\[ \frac{\partial R_\alpha}{\partial \alpha} = \frac{\partial}{\partial \alpha} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -\sin(\alpha) & -\cos(\alpha) & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \]  (5.53)

Differentiation with respect to \( \beta \) is done in a similar fashion.

Differentiation of shape parameters

The relationship between object space coordinates and the shape vector is described in equation (5.4). The transformation from object coordinates to camera coordinates is shown in equation (5.6). Differentiation of this expression with respect to the different shape parameters is simple. A given camera space point \( p_c \) is:

\[ c = TR_\beta R_\alpha M_i s \]  (5.54)

Using a vector \( v_j \) defined as:

\[ v_k = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{else.} \end{cases} \]  (5.55)

Differentiation of equation (5.54) is:

\[ \frac{\partial c}{\partial s_j} = \frac{\partial}{\partial s_j} (TR_\beta R_\alpha M_i s) = TR_\beta R_\alpha M_i v_j \]  (5.56)
5.7. PARAMETER FITTING

5.7.3 Newton iteration

Given a vector valued function $f$ that we wish to minimize we can introduce an error function $\varepsilon$:

$$\varepsilon(x) = f(x) - f(x^*)$$

(5.57)

where the $x^*$ is the location of our local minimum. When using Newton minimization we approximate the error as being locally linear. Using Taylor expansion we get the following approximation:

$$\varepsilon(x_0 + \Delta) \approx \varepsilon(x_0) + J\Delta$$

(5.58)

with $J$ being the Jacobian matrix evaluated at $x_0$. Per definition our local minimum is in $x^*$ and $\varepsilon(x^*) = 0$. Thus setting equation (5.58) to zero yields:

$$\varepsilon(x_0) + J\Delta = 0$$

(5.59)

In general, we will have a lot of error estimates. The equation system in (5.59) is thus over determined. Using least-squares to solve (5.59) we get:

$$\Delta = -(J^T J)^{-1} J^T \varepsilon(x_0)$$

(5.60)

Using $\Delta$ to update $x_n$ we get the starting point for our next iteration:

$$x_{n+1} = x_n + \Delta$$

(5.61)

Newton iteration will converge fast when the approximation of local linearity is valid. However, the method has serious stability issues when this requirement is not met. However, for small changes the transformations that occurs in this system are approximately linear.

5.7.4 Levenberg-Marquardt iteration

L-M (Levenberg-Marquardt) minimization tries to combine the fast convergence of Newton minimization but at the same time reduce instability issues. With $A$ defined as:

$$A = J^T J$$

(5.62)

we introduce $A'$ defined as:

$$A'_{ij} = (1 + \lambda)A_{ij}, \text{ if } i = j$$

(5.63)

$$A'_{ij} = A_{ij}, \text{ if } i \neq j$$

(5.64)

For small $\lambda$ values the matrix $A'$ is almost to the same as the one used in the Newton method. For large $\lambda$ values $A'$ will be dominated by the values in the diagonal. This will result in behavior close to that of a steepest decent algorithm. A complete L-M iteration is:
Calculate $A'.$

- Solve $A' \Delta = -J^T \varepsilon.$
- Evaluate $\varepsilon^* = \varepsilon(x_n + \Delta)$
- If $\varepsilon^* \geq \varepsilon$ then $\lambda = 10 \lambda$ and $x_{n+1} = x_n$
- If $\varepsilon^* < \varepsilon$ then $\lambda = 0.1 \lambda$ and $x_{n+1} = x_n + \Delta$

### 5.8 Keeping fixed lines

The process of instantiating our model can be described as a vector valued function:

$$v = f(x) \quad (5.65)$$

where each row of $v$ corresponds to one distance measure as described in section 5.7.1. The error is then described by:

$$e = f(x_i) - f(x) \quad (5.66)$$

which is minimized using Newton or Levenberg-Marquardt minimization. The trouble with the Newton method is that it is notoriously unstable. To handle some of this it is sometimes useful to impose some restrictions on the solutions. Recall the standard equation system that is solved:

$$J(x_i - x) = J \Delta_x = e \quad (5.67)$$

where $J$ is the Jacobian matrix of $f.$ We wish to impose a restriction in the form:

$$W^T \Delta_x = 0 \quad (5.68)$$

where $W$ is a matrix with columns $w_i:$

$$W = \begin{pmatrix} w_1 & w_2 & \ldots & w_m \end{pmatrix} \quad (5.69)$$

If the columns of $W$ are chosen from the rows in $J$ this restriction expresses that the errors for rows should not change. What this implies is that the chosen lines should remain static. The only thing that may vary is the length of the segment since the errors do not tell us anything about the length.

One solution to this problem is to find some basis where the columns of $W$ are all contained in the space spanned within the first $r$ basis vectors, where $r$ is the rank of $W.$ This way we know that the first $r$ elements must be zero if the restriction in equation (5.68) is to be fulfilled. Let $U$ be a matrix containing orthonormal basis vectors that spans $x:$

$$U = \begin{pmatrix} e_1 & e_2 & \ldots & e_n \end{pmatrix} \quad (5.70)$$
chosen so that:
\[
\mathbf{w}_k^T \mathbf{e}_l = 0, \text{ for } k = 1, \ldots, m \text{ and } l = r + 1, \ldots, n \tag{5.71}
\]
where \( r \) is the rank of \( \mathbf{W} \). Obtaining the basis matrix \( \mathbf{U} \) is easily done using the singular value decomposition (SVD) of \( \mathbf{W} \).

We can now performing a basis change:
\[
\mathbf{\Delta}_y = \mathbf{U}^{-1} \mathbf{\Delta}_x \tag{5.72}
\]
and since \( \mathbf{U} \) is orthonormal we get:
\[
\mathbf{\Delta}_y = \mathbf{U}^T \mathbf{\Delta}_x \tag{5.73}
\]
The inverse basis change is:
\[
\mathbf{\Delta}_x = \mathbf{U} \mathbf{\Delta}_y \tag{5.74}
\]
The equation in (5.71) then implies that any \( \mathbf{\Delta}_y \) which has zeros in the first \( r \) components will satisfy the restriction in (5.68). Solving (5.67) is thus reduced to determining the last \( n - r \) components of \( \mathbf{\Delta}_y \). Performing the basis change in (5.67) yields:
\[
J_x \mathbf{U} \mathbf{\Delta}_y = J_y \mathbf{\Delta}_y = \mathbf{e} \tag{5.75}
\]
The submatrix needed to solve equation (5.67) is then:
\[
J'_y = \begin{pmatrix} j_{r+1} & \cdots & j_m \end{pmatrix} \tag{5.76}
\]
where \( j_k \) are the last \( n - r \) columns of \( J_y \). The last components of \( \mathbf{\Delta}_y \) are then computed by the following equation solving in a suitable fashion:
\[
J'_y \mathbf{\Delta}'_y = \mathbf{e} \tag{5.77}
\]
Thus:
\[
\mathbf{\Delta}_y = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{\Delta}'_y \end{pmatrix} \tag{5.78}
\]
Finally we get the desired solution by transforming \( \mathbf{\Delta}_y \) back to the original basis using equation (5.74).

### 5.9 Segment matching

Obtaining the match between an instantiated segment from the vehicle model and an observed line segment is a difficult task. Due to the limitations of this work only cover a few fairly simple ones will be covered.
5.9.1 Midpoint distance

Given two sets of lines \( X \) and \( Y \) we first calculate the midpoint of the lines:

\[
m_{ix} = \frac{x_{is} + x_{ie}}{2} \quad (5.79)
\]

where \( x_{is} \) and \( x_{ie} \) are the start- and end-points of line \( i \) in the set \( X \). The same is done for \( Y \) and we get:

\[
m_{iy} = \frac{y_{js} + y_{je}}{2} \quad (5.80)
\]

The distance between two lines \( i \) and \( j \) is then:

\[
d_{ij} = \| m_{ix} - m_{iy} \|_2 \quad (5.81)
\]

An observed line \( y_j \) is considered to match a model line \( y_i \) if:

\[
d_{ij} \leq d_{ik}, \text{ for all } k \quad (5.82)
\]

Obviously this is a very naive approach that will only work if the differences between the model segments and the observed segments are small.

5.9.2 Centered midpoint distance

In order to improve the performance of the Midpoint distance algorithm one can remove the mean of the line sets \( X \) and \( Y \). Calculation of the mean is done by:

\[
m_X = \frac{1}{n} \sum_{i=1}^{n} m_{ix} \quad (5.83)
\]

and:

\[
m_Y = \frac{1}{m} \sum_{j=1}^{m} m_{ix} \quad (5.84)
\]

The new distance measure is then:

\[
d_{ij} = \| (m_{ix} - m_X) - (m_{iy} - m_Y) \|_2 \quad (5.85)
\]

The selection criterion is the same as for the naive midpoint distance, that is:

An observed line \( y_j \) is considered to match a model line \( y_i \) if:

\[
d_{ij} \leq d_{ik}, \text{ for all } k \quad (5.86)
\]
5.9. SEGMENT MATCHING

5.9.3 Normalized midpoint distance

A natural extension is to not only remove the mean of the sets but also try to normalize the direction and extension of them. If we consider the midpoints \( m_{ix} \) to be of normal distribution, more specifically:

\[
\mathbf{m}_{ix} \in N(\mathbf{m}_X, \mathbf{C}_X)
\]  

(5.87)

Similarly the distribution for \( y_{iy} \) is:

\[
\mathbf{m}_{iy} \in N(\mathbf{m}_Y, \mathbf{C}_Y)
\]  

(5.88)

The next step is to normalize these distributions. That is, remove the mean and normalize the covariance to the identity matrix. Removing the mean is trivial and in the following text any reference to \( \mathbf{m}_{ix} \) assumes that the mean has been removed.

Furthermore it is assumed that the relation between the normalized points and the current points is linear. That is:

\[
\mathbf{m}_{ix} = A\mathbf{m}^*_{ix}
\]  

(5.89)

where \( \mathbf{m}^*_{ix} \) has the distribution:

\[
\mathbf{m}^*_{ix} \in N(\mathbf{0}, \mathbf{I})
\]  

(5.90)

where \( \mathbf{I} \) is the identity matrix.

Using eigenvalue decomposition of the covariance matrix \( \mathbf{C}_X \) we get:

\[
\mathbf{C}_X = \mathbf{U}\Sigma\mathbf{U}^T
\]  

(5.91)

Where \( \mathbf{U} \) is an orthonormal matrix and \( \Sigma \) is a diagonal matrix with the squared eigenvalues of \( \mathbf{C}_X \) as diagonal elements. Since \( \mathbf{C}_X \) is symmetric and positive definite it is also true that the eigenvalues will be real.

We are now ready to introduce the normalizing matrix:

\[
A^\dagger = \Sigma^{-\frac{1}{2}}\mathbf{U}^T
\]  

(5.92)

Where \( \Sigma^{-\frac{1}{2}} \) is:

\[
\Sigma^{-\frac{1}{2}} = \begin{pmatrix}
\frac{1}{\sqrt{\sigma_1^2}} & 0 \\
0 & \frac{1}{\sqrt{\sigma_2^2}}
\end{pmatrix}
\]  

(5.93)

It will be shown that the variable \( \mathbf{m}^*_{ix} \) will have a distribution normalized to \( N(\mathbf{0}, \mathbf{I}) \) if:

\[
\mathbf{m}^*_{ix} = A^\dagger\mathbf{m}_{ix}
\]  

(5.94)

Since the mean of \( \mathbf{m}_{ix} \) has been removed the mean of \( \mathbf{m}^*_{ix} \) will also be zero. If we examine the covariance of \( \mathbf{m}^*_{ix} \) we see:

\[
E\{\mathbf{m}^*_{ix}(\mathbf{m}^*_{ix})^T\} = A^\dagger E\{\mathbf{m}_{ix}\mathbf{m}_{ix}^T\}(A^\dagger)^T = A^\dagger\mathbf{C}_X(A^\dagger)^T
\]  

(5.95)
Using equation (5.91) and equation (5.92) we see that:
\[ A^T C X (A^T)^T = \Sigma^{-\frac{1}{2}} U^T U \Sigma U^{T} U \Sigma^{-\frac{1}{2}} = \Sigma^{-\frac{1}{2}} \Sigma \Sigma^{-\frac{1}{2}} = I \] (5.96)

Hence, \( m_x^* \) has a distribution of \( N(0, I) \).

Performing the above transformation for both \( X \) and \( Y \) yields:
\[ m_x^* \in N(0, I) \] (5.97)

and:
\[ m_y^* \in N(0, I) \] (5.98)

The distance measure between two lines is then:
\[ d_{ij} = \| m_x^* - m_y^* \|_2 \] (5.99)

The selection criterion remains the same.

**5.9.4 Mahalanobis distance**

The Mahalanobis distance takes into account the different covariances of different variables in a vector. For instance, an angle does not vary in the same way that a coordinate does. In order to have a useful distance measure this should be considered. For more information on the Mahalanobis distance see [3]. Given two random vectors \( x \) and \( y \) defined as:
\[ x \in N(0, C) \] (5.100)

and:
\[ y \in N(0, C) \] (5.101)

respectively. The Mahalanobis distance between them is:
\[ d^2 = (x - y)^T C^{-1} (x - y) \] (5.102)

For the purpose of segment matching we need to define a suitable parameter vector. In this work the vector used is defined as:
\[ x = \begin{pmatrix} m_1 \\ m_2 \\ l \\ \Theta \end{pmatrix} \] (5.103)

Where \( m_1 \) and \( m_2 \) are the coordinates of the segments midpoint. \( l \) is the length of the segment and \( \Theta \) is the orientation of the segment.

Using a number of extracted line segments, the distance between them was measured and the covariance was estimated. The resulting covariance matrix is:
This matrix indicates that difference in orientation should have a bigger influence then difference in position or length. This should be the case since a big orientation difference might have the magnitude 1 and a large position difference is in the magnitude 100.

Admittedly, the samples used for this estimation might have been biased and this may effect the performance.

Selection is the done in the same manner as for the midpoint methods.

### 5.9.5 Mahalanobis ratio

While this measure will not result in any different line matches than the Mahalanobis it will give a different magnitude in the measure. The motivation behind this measure is that it should in some way capture the unambiguity of the match. If a line is matched to two others with almost the same small distance it is ambiguous which one that should be chosen. On the other hand, if a line is matched, even poorly, to another line but the only competing matches are very poor it should be fairly certain that this match is valid.

The Mahalanobis ratio is defined as:

\[
q = \frac{d_{\text{min}}}{\lambda + d_{\text{min2}}}
\]

where \(d_{\text{min}}\) is the minimum distance between two segment, \(d_{\text{min2}}\) is the second smallest distance and \(\lambda\) is a constant that can be chosen arbitrarily to finetune the function. In this work \(\lambda\) is set to zero.

### 5.10 Fitting polices

To achieve convergence on data that is erroneous we can manipulate the minimization process to only minimize with respect to certain lines. This section could be expanded very much but due to the limitations of this work only a few very naive approaches will be described.

#### 5.10.1 Fitting all lines

This policy is to try to establish matches for all instantiated lines and minimize the residual errors. If the data is incomplete or incorrect matches are obtained this policy will probably yield bad results.
5.10.2 Minimal fitting

The extreme opposite of the “fit all” policy is to use only as many lines that are needed to determine the parameters uniquely. Since each line gives us three equations we only use $\lceil n/3 \rceil$ lines in the minimization process. The lines are chosen are the ones with the least distance between the instantiated and the extracted line. This approach should handle incomplete data better than the “fit all” policy. If the data is erroneous or if matches are incorrect it will give very large errors in the resulting parameter estimate since we have no redundancy in our data.

5.10.3 Overdetermined fitting

In an effort to combine the good points from the “all” and “minimal” policies one could choose to fit using a higher number of lines then required and thus have an overdetermined equation system. This will give us some redundancy in our data while not being effected as much by missing data.

5.10.4 Close fitting

This policy attempts to use the distance measures obtained while matching to determine which lines to use in the fitting process. The motivation for this policy is that small errors should yield small parameter adjustments. This way we will hopefully prevent the minimization process from diverging too much. Given a set of distance measures $d$ obtained in the matching process we include a line in the fitting process if:

$$d_i < \lambda$$  \hspace{1cm} (5.105)

where $\lambda$ is a threshold. The choice of threshold is not obvious. In this work the following choice is used:

$$\lambda = \max(2\min(d_i), \sigma)$$  \hspace{1cm} (5.106)

where $\sigma^2$ is an estimate of the covariance of our distance measures. This choice is rather ad hoc. The thought behind this definition is that the $2\min(d_i)$ threshold will select a suitable subset of all lines to be used in the fitting. However, if this threshold is lower than the error covariance $\sigma$ all lines with lower variance than this will be used since they will most likely be good enough to be included in the minimization process.

5.10.5 Progressive fitting

This simple policy tries to use few lines to get a rough improvement of our initial estimate. Once a rough improvement has been obtained more lines will be included to further refine our estimate. This policy simply chooses the $k$ lines with the least
error measure obtained from matching. The variable $k$ is a function of the iteration number. In this work a very simple method is employed, namely:

$$k = 2i$$

where $i$ is the iteration number.

### 5.10.6 Random fitting

Mostly for testing purposes a random policy is also included. This policy simply chooses the lines to use when fitting at random. Each line is included with the probability $p = 0.5$. 
Chapter 6

Evaluation of detection techniques

This chapter will cover the results of the detection method. First the tracker will be discussed even though it is not evaluated in the strict sense. A small experiment showing the results of different resampling techniques is then shown. Finally, image warping is performed using the different transformation models described in section 4.4.

6.1 Tracker performance

The tracker used has been developed at CVL at Linköping university. Though not strictly a KLT tracker it uses the same criterion for selecting features. The tracking is then done in a brute force manner, that is, search a small neighbourhood for the position that minimizes the SSD (sum of squared difference). It has a decent performance that certainly is enough for the different estimation techniques that will be evaluated. If real-time performance was to become an issue a tracker based on a sparse representation using gradients has been developed. For more information see [7].

6.2 Resampling

In order to determine which resampling approach to use a small experiment was conducted in matlab. A simple image shown in figure 6.2 was resampled using four different methods, nearest neighbour, linear interpolation, cubic spline and ideal sinc. For a theoretical background see section 4.5. The results of this experiment are shown in figure 6.2

The nearest neighbour approach is very fast but shows some heavy aliasing and the quality is too low. Linear interpolation is a bit more computationally expensive than the nearest neighbour method but the result is much better. Using linear in-
CHAPTER 6. EVALUATION OF DETECTION TECHNIQUES

Figure 6.1: Image used in resampling experiment.

Figure 6.2: Top Left: Resampled using nearest neighbour. Top Right: Resampled using linear interpolation. Bottom Left: Resampled using cubic spline. Bottom Right: Resampled using ideal sinc function.
6.3 Evaluation of the different transformations

In order to evaluate the different transformations proposed in section 4.4.1 a number of features points were tracked on real video data that what captured from an aerial platform. Using the tracked feature points different transformations were estimated. Finally one image was warped to fit the other and the magnitude of the difference image was calculated. Ideally we would only see two white blobs in the final differential image, one where the car was at the first frame and one where the car is in the final frame. Any other contributions to the differential image should be cancelled out due to the image warping process.

Figure 6.3 shows the results of one test run. Over the 20 frames 100 feature points were tracked and the difference between the first and the last image was calculated. The differences between the first and the last image is quite large, notice that the entire scene has been translated and rotated. As expected the method only using translation performs very poorly, giving large responses where the road has rotated. The quadratic method performs better but it still gives some unwanted contributions. Since this method is a generalization of the affine one it should give us at least as good results. The lack of performance of the quadratic method is probably due to numerical issues, small errors in the quadratic coefficients will yield large errors in the warping stage. Both the affine and the homographic methods work well with homographic being the superior of the two.

The background in the sequence used in figure 6.4 is changing more slowly
6.3. EVALUATION OF THE DIFFERENT TRANSFORMATIONS

then the previous one. This allow us to track the feature points for a longer duration. If we study the figure we see, much as in the previous experiment, that the affine and homographic methods yield superior results and that the difference between the two is small.

The two previous tests have shown that the difference between homographic and affine transformation estimation is small. While this is the case for the sequences in question it is not generally true. Sequences where the camera undergo large tilts the homographic estimation will prove superior. However, one might argue that the sequences used are typical for the WITAS platform.

For a comparison of the computational efficiency of the different methods a small example is shown in table 6.2. The numbers are based on the assumption that 50 feature points are tracked. Outlier detection has not been taken in to account when calculating the figures, roughly the cost of each iteration in the outlier algorithm is the same as the cost stated in the table. If many outliers are present in one frame, the computational cost will be then increase proportionally.

<table>
<thead>
<tr>
<th>Method</th>
<th>KFlops (excluding inversion)</th>
<th>Inversion size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td>0.2</td>
<td>N/A</td>
</tr>
<tr>
<td>Affine</td>
<td>4.8</td>
<td>$6 \times 6$</td>
</tr>
<tr>
<td>Quadratic</td>
<td>16.8</td>
<td>$12 \times 12$</td>
</tr>
<tr>
<td>Homographic</td>
<td>8.0</td>
<td>$8 \times 8$</td>
</tr>
</tbody>
</table>

Table 6.2: Cost of estimating different transformations from 50 feature points.

Obviously the most efficient method is the one using pure translational movement. However this method yields unacceptable results. The most expensive method is the one using quadratic transformation, but as we have seen it gives worse results than both the homographic and the affine methods. The choice between the homographic and the affine method is close. Given the relatively small computational difference the homographic method should probably be preferred.
Chapter 7

Evaluation of classification techniques

This chapter contains evaluation of the classification techniques. First the a number of synthetic test setups will be described. Using these tests the different segment matching algorithms will be evaluated. Two of the matching algorithms will then be used in conjunction with the different fitting policies to determine the best combination with the best performance. Finally, the algorithm is tested on real data. That is, lines extracted from a rendered sequence.

7.1 Synthetic test setup

Due to the lack of suitable test data the algorithms will first be evaluated using a synthetic test. The test data is generated using the vehicle model. An initial set of parameters are used as the initial estimate in the algorithm. The parameters controlling the model pose are then modified to generate the target parameters. It is the minimization algorithms task to minimize the errors so the initial estimate converges to the target parameters. It is also possible to corrupt the target data by adding noise to the lines, adding non existent lines or removing lines.

Let $\mathbf{U}$ be our initial estimate, defined in equation (5.2). The target parameter vector is then calculated by:

$$\mathbf{U} = \hat{\mathbf{U}} + \mathbf{\Delta}_U$$

where $\mathbf{\Delta}_U$ is a random vector with a given covariance. That is:

$$\mathbf{\Delta}_U \sim N(0, C_U)$$

The parameter vector $\mathbf{x}_t$ is then used to generate the lines that will be the target data in the minimization process. After minimization the estimate is considered to have converged if:

$$\| \mathbf{x}_t - \mathbf{x}^* \|_2 \leq \epsilon$$

where $\epsilon$ is a predefined threshold and $\mathbf{x}^*$ are the minimized parameters.
CHAPTER 7. EVALUATION OF CLASSIFICATION TECHNIQUES

There is also a possibility to introduce noise to the start and end positions of the line segments. This is done by updating positions according to:

\[ \mathbf{p}' = \mathbf{p} + \Delta_p \]  

(7.4)

where \( \mathbf{p} \) is the start or endpoint of a line segment. \( \Delta_p \) is a random variable defined by:

\[ \Delta_p \sim N(0, C_p) \]  

(7.5)

To further simulate “real” conditions there is also the possibility to remove lines. A percentage of the instantiated lines are removed at random.

The testing environment also includes the possibility to add garbage lines to the simulation. Garbage lines are constructed using copies of the existing lines with noise added to the start and end points:

\[ \mathbf{p}_g' = \mathbf{p} + \Delta_g \]  

(7.6)

The distribution of the noise is given:

\[ \Delta_g \sim N(0, C_g) \]  

(7.7)

7.1.1 Perfect test setup

The first test only modifies the position and rotation of the model. No noise is added to the lines, no lines are removed and no garbage lines are added. The convergence threshold is \( \varepsilon = 0.1 \). This threshold is fairly low but since there is no noise present a correct minimization should yield very low errors. The model is varied by adding Gaussian noise with the covariance:

\[ C_x = \begin{pmatrix} 0.3 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0.3 \end{pmatrix} \]  

(7.8)

This test only updates the pose of the model and this is why the covariance matrix presented is only \( 5 \times 5 \). If the shape of the vehicle was to be modified a full \( 17 \times 17 \) covariance matrix would be used. A number of different instantiations from this setup is shown in figure 7.1.

7.1.2 Noisy test setup

The noisy test is similar to the perfect test with the difference that the lines have noise added to them. The noise is Gaussian and has the covariance:

\[ C_p = \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix} \]  

(7.9)
7.2. Evaluation of different line matching methods.

Since there is noise present we cannot expect perfect convergence we will thus be a bit more relaxed when judging this. The convergence threshold in this experiment is set to $\varepsilon = 0.2$. Figure 7.2 shows a couple of examples from this setup.

7.1.3 Incomplete test setup

In the next test 40% of the lines are removed, no noise is added. The convergence threshold remains the same at $\varepsilon = 0.2$. In figure 7.3 a number examples from this setup are shown.

7.1.4 Garbage test setup

The garbage test is made to judge how the different methods perform in the presence of irrelevant lines. The test is simply a model with updated pose which has had a number of garbage lines added. In this setup 9 garbage lines are added. As in the previous two tests the convergence threshold is $\varepsilon = 0.2$. The garbage lines are copies of the original lines with strong Gaussian noise added to the endpoints. This Gaussian noise has zero mean and the covariance:

$$C_g = \begin{pmatrix} 30 & 0 \\ 0 & 30 \end{pmatrix}$$ (7.10)

Figure 7.4 shows some examples of instantiations from this setup.

7.1.5 Nightmare test setup

The last test setup is a worst case scenario. It combines the difficulties of previous three tests. That is, noise is added, lines are removed and garbage lines are added. The parameters for these modifications are the same as in the previous tests. As before, the convergence threshold is $\varepsilon = 0.2$. In figure 7.5 some variations from this tests is shown.

7.2 Evaluation of different line matching methods.

In order to evaluate the different line matching methods the fitting policy remains the same throughout the tests. The different line matching methods are described in section 5.9. For testing purposes there is also a random picking algorithm. It picks line matches totally at random. The non minimized results are also presented, that is, when the initial guess is not modified at all. The evaluation is done using the overdetermined fitting policy, without a proper evaluation of the different policies this one seems to be a strong contender on paper.
CHAPTER 7. EVALUATION OF CLASSIFICATION TECHNIQUES

Figure 7.1: Perfect test variations. Top Left: Initial estimate. Others: Target variations.

Figure 7.2: Noisy test variations. Top Left: Initial estimate. Others: Target variations.
7.2. EVALUATION OF DIFFERENT LINE MATCHING METHODS.

Figure 7.3: Incomplete test variations. Top Left: Initial estimate. Others: Target variations.

Figure 7.4: Garbage test variations. Top Left: Initial estimate. Others: Target variations.
CHAPTER 7. EVALUATION OF CLASSIFICATION TECHNIQUES

7.2.1 Perfect test results

The minimization process was run 100 times and the result were obtained by allowing the different methods to work on the exact same data. In table 7.1 the results are shown. Recall that the convergence was considered to have occurred when the magnitude of the difference between the minimized parameters and the true parameters were less then a certain threshold. This experiment has the convergence threshold set to $\varepsilon = 0.1$.

The results show that two methods outperform the rest, namely the Mahalanobis ratio and normalized midpoint distance. However, further testing is needed to ensure that they indeed are the best of the pack.

<table>
<thead>
<tr>
<th>Method</th>
<th>Convergence rate</th>
<th>Mean parameter error</th>
</tr>
</thead>
<tbody>
<tr>
<td>No minimization</td>
<td>0.09</td>
<td>0.59</td>
</tr>
<tr>
<td>Random picks</td>
<td>0.00</td>
<td>2.15</td>
</tr>
<tr>
<td>Midpoint distance</td>
<td>0.22</td>
<td>0.88</td>
</tr>
<tr>
<td>Centered midpoint distance</td>
<td>0.62</td>
<td>0.38</td>
</tr>
<tr>
<td>Normalized midpoint distance</td>
<td>0.90</td>
<td>0.13</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>0.57</td>
<td>0.30</td>
</tr>
<tr>
<td>Mahalanobis ratio</td>
<td>0.94</td>
<td>0.054</td>
</tr>
</tbody>
</table>

Table 7.1: Comparison of picking methods in the perfect test.

7.2.2 Noisy test results

The results of 100 runs of the noisy test are shown in table 7.2. In this test the convergence threshold was set to $\varepsilon = 0.2$.

From this test we see that even in the presence of fairly strong noise normalized midpoint distance and Mahalanobis ratio performs well. Some of more naive approaches actually performs worse then not optimizing at all.

<table>
<thead>
<tr>
<th>Method</th>
<th>Convergence rate</th>
<th>Mean parameter error</th>
</tr>
</thead>
<tbody>
<tr>
<td>No minimization</td>
<td>0.20</td>
<td>0.60</td>
</tr>
<tr>
<td>Random picks</td>
<td>0.00</td>
<td>2.10</td>
</tr>
<tr>
<td>Midpoint distance</td>
<td>0.11</td>
<td>0.98</td>
</tr>
<tr>
<td>Centered midpoint distance</td>
<td>0.47</td>
<td>0.47</td>
</tr>
<tr>
<td>Normalized midpoint distance</td>
<td>0.75</td>
<td>0.31</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>0.37</td>
<td>0.48</td>
</tr>
<tr>
<td>Mahalanobis ratio</td>
<td>0.71</td>
<td>0.28</td>
</tr>
</tbody>
</table>

Table 7.2: Comparison of picking methods in the noisy test.

The results of 100 runs of the noisy test are shown in table 7.2. In this test the convergence threshold was set to $\varepsilon = 0.2$.

From this test we see that even in the presence of fairly strong noise normalized midpoint distance and Mahalanobis ratio performs well. Some of more naive approaches actually performs worse then not optimizing at all.
7.2. EVALUATION OF DIFFERENT LINE MATCHING METHODS.

Figure 7.5: Nightmare test variations. Top Left: Initial estimate. Others: Target variations.

<table>
<thead>
<tr>
<th>Method</th>
<th>Convergence rate</th>
<th>Mean parameter error</th>
</tr>
</thead>
<tbody>
<tr>
<td>No minimization</td>
<td>0.23</td>
<td>0.54</td>
</tr>
<tr>
<td>Random</td>
<td>0.00</td>
<td>2.17</td>
</tr>
<tr>
<td>Midpoint distance</td>
<td>0.20</td>
<td>0.71</td>
</tr>
<tr>
<td>Centered midpoint distance</td>
<td>0.21</td>
<td>0.76</td>
</tr>
<tr>
<td>Normalized midpoint distance</td>
<td>0.29</td>
<td>0.72</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>0.63</td>
<td>0.27</td>
</tr>
<tr>
<td>Mahalanobis ratio</td>
<td>0.67</td>
<td>0.36</td>
</tr>
</tbody>
</table>

Table 7.3: Comparison of picking methods in the incomplete test.
7.2.3 Incomplete test results

As before, the test was run 100 times with the convergence threshold set to $\varepsilon = 0.2$. The results of the incomplete test in table 7.3 show that the normalization process used in normalized midpoint distance and centered midpoint distance is heavily penalized when lines are removed. For example, calculation of the mean will be erroneous if not all lines are present. This is a major drawback of all the midpoint distance algorithms. Most methods perform poorly, only the Mahalanobis methods has acceptable performance. If this performance hit can be overcomed with the use of a different fitting policy remains to be seen.

7.2.4 Garbage test results

<table>
<thead>
<tr>
<th>Method</th>
<th>Convergence rate</th>
<th>Mean parameter error</th>
</tr>
</thead>
<tbody>
<tr>
<td>No minimization</td>
<td>0.31</td>
<td>0.43</td>
</tr>
<tr>
<td>Random</td>
<td>0.00</td>
<td>2.15</td>
</tr>
<tr>
<td>Midpoint distance</td>
<td>0.15</td>
<td>0.85</td>
</tr>
<tr>
<td>Centered midpoint distance</td>
<td>0.38</td>
<td>0.57</td>
</tr>
<tr>
<td>Normalized midpoint distance</td>
<td>0.52</td>
<td>0.38</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>0.58</td>
<td>0.33</td>
</tr>
<tr>
<td>Mahalanobis ratio</td>
<td>0.91</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Table 7.4: Comparison of picking methods in the garbage test.

Results from the garbage test is shown in table 7.4. Once again, most algorithms performed poorly. The big exception is the Mahalanobis ratio that still had a high level of convergence.

7.2.5 Nightmare test results

The results from 100 runs of the nightmare test is shown in table 7.5. Not surprisingly most pick methods perform poorly. The Mahalanobis ratio is still the best performing method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Convergence rate</th>
<th>Mean parameter error</th>
</tr>
</thead>
<tbody>
<tr>
<td>No minimization</td>
<td>0.25</td>
<td>0.50</td>
</tr>
<tr>
<td>Random</td>
<td>0.00</td>
<td>2.12</td>
</tr>
<tr>
<td>Midpoint distance</td>
<td>0.08</td>
<td>0.88</td>
</tr>
<tr>
<td>Centered midpoint distance</td>
<td>0.15</td>
<td>0.82</td>
</tr>
<tr>
<td>Normalized midpoint distance</td>
<td>0.17</td>
<td>0.69</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td>0.43</td>
<td>0.41</td>
</tr>
<tr>
<td>Mahalanobis ratio</td>
<td>0.58</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Table 7.5: Comparison of picking methods in the nightmare test.
7.2.6 Conclusions

The different tests has shown that the picking algorithms themselves make some difference in the convergence of the method. The two pick methods that performed best were Mahalanobis ratio and Mahalanobis distance. However, since both are based on the Mahalanobis distance it is more interesting to perform further testing on one of the other algorithms. The algorithm that performed best except the Mahalanobis based ones was midpoint distance with normalized covariance. Further testing will thus only be done using normalized midpoint distance and Mahalanobis ratio.

7.3 Evaluation of different fitting methods

In the previous section we evaluated the different methods available for segment matching. In this section two of the matching methods will be used in conjunction with the different fitting policies to evaluate performance. The two picking methods used are Mahalanobis ratio and Normalized midpoint distance.

7.3.1 Perfect test results

The perfect test setup was run 100 times. The convergence threshold was set to \( \varepsilon = 0.1 \) which is fairly strict, but as stated previously, if convergence occurs the error will be very small. The results of this test are shown in table 7.6. All algorithms perform well and so they should considering the relatively easy test.

<table>
<thead>
<tr>
<th>Method</th>
<th>Normalized midpoint</th>
<th>Mahalanobis ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Convergence</td>
<td>Mean error</td>
</tr>
<tr>
<td>No minimization</td>
<td>0.14</td>
<td>0.54</td>
</tr>
<tr>
<td>All</td>
<td>0.83</td>
<td>0.16</td>
</tr>
<tr>
<td>Random</td>
<td>0.89</td>
<td>0.21</td>
</tr>
<tr>
<td>Close</td>
<td>0.91</td>
<td>0.15</td>
</tr>
<tr>
<td>Progressive</td>
<td>0.91</td>
<td>0.14</td>
</tr>
<tr>
<td>Minimal</td>
<td>0.78</td>
<td>0.44</td>
</tr>
<tr>
<td>Overdetermined</td>
<td>0.88</td>
<td>0.16</td>
</tr>
</tbody>
</table>

Table 7.6: Comparison of different fitting methods in the perfect test.

7.3.2 Noisy test results

The results of the noisy test are shown in table 7.2. This test was run 100 times and had the convergence threshold \( \varepsilon = 0.2 \). The minimal policy performs very poorly due to its lack of error redundancy. Surprisingly the random policy performs fairly well. However, this test is a bit to easy to truly judge the real performance of the policies.
7.3.3 Incomplete test results

Table 7.8 show the results of the incomplete test. As before, the test was run 100 times and with the convergence threshold set to \( \varepsilon = 0.2 \). This test shows that the previous performances of the random and all policies were dependent on the fact that all lines were present. In fact, the methods that considers all of the lines perform rather poorly.

7.3.4 Garbage test results

From the results of the garbage test shown in table 7.9 we see that the presence of garbage is mostly a challenge for the picking algorithms. Most policies perform well when matching is done with the Mahalanobis ratio method.

7.3.5 Nightmare test results

The results from the nightmare test in table 7.10 are somewhat disappointing. The only algorithms that performed better then no fitting were Mahalanobis ratio combined with either the close policy or the overdetermined policy.

7.3.6 Conclusions

During the testing the different policies showed different strengths and weaknesses. The policies that are dependent on the presence of all the lines (all, random and progressive) performed poorly in tests where some lines were removed. The methods that only operate on a subset of the lines performed poorly when noise added to the noise, in particular the minimal policy. The best choices seem to be either the close policy combined with Mahalanobis ratio matching or the overdetermined policy combined with Mahalanobis ratio matching.

<table>
<thead>
<tr>
<th>Method</th>
<th>Normalized midpoint</th>
<th>Mahalanobis ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Convergence</td>
<td>Mean error</td>
</tr>
<tr>
<td>No minimization</td>
<td>0.26</td>
<td>0.57</td>
</tr>
<tr>
<td>All</td>
<td>0.86</td>
<td>0.22</td>
</tr>
<tr>
<td>Random</td>
<td>0.83</td>
<td>0.25</td>
</tr>
<tr>
<td>Close</td>
<td>0.64</td>
<td>0.48</td>
</tr>
<tr>
<td>Progressive</td>
<td>0.70</td>
<td>0.46</td>
</tr>
<tr>
<td>Minimal</td>
<td>0.12</td>
<td>1.40</td>
</tr>
<tr>
<td>Overdetermined</td>
<td>0.63</td>
<td>0.37</td>
</tr>
</tbody>
</table>

Table 7.7: Comparison of different fitting methods in the noisy test.
### 7.3. Evaluation of Different Fitting Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Normalized midpoint</th>
<th>Mahalanobis ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Convergence</td>
<td>Mean error</td>
</tr>
<tr>
<td>No minimization</td>
<td>0.31</td>
<td>0.52</td>
</tr>
<tr>
<td>All</td>
<td>0.20</td>
<td>0.68</td>
</tr>
<tr>
<td>Random</td>
<td>0.09</td>
<td>0.85</td>
</tr>
<tr>
<td>Close</td>
<td>0.31</td>
<td>1.02</td>
</tr>
<tr>
<td>Progressive</td>
<td>0.21</td>
<td>0.80</td>
</tr>
<tr>
<td>Minimal</td>
<td>0.27</td>
<td>1.36</td>
</tr>
<tr>
<td>Overdetermined</td>
<td>0.28</td>
<td>0.82</td>
</tr>
</tbody>
</table>

Table 7.8: Comparison of different fitting methods in the incomplete test.

<table>
<thead>
<tr>
<th>Method</th>
<th>Normalized midpoint</th>
<th>Mahalanobis ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Convergence</td>
<td>Mean error</td>
</tr>
<tr>
<td>No minimization</td>
<td>0.31</td>
<td>0.49</td>
</tr>
<tr>
<td>All</td>
<td>0.85</td>
<td>0.13</td>
</tr>
<tr>
<td>Random</td>
<td>0.74</td>
<td>0.28</td>
</tr>
<tr>
<td>Close</td>
<td>0.45</td>
<td>0.62</td>
</tr>
<tr>
<td>Progressive</td>
<td>0.68</td>
<td>0.33</td>
</tr>
<tr>
<td>Minimal</td>
<td>0.28</td>
<td>1.45</td>
</tr>
<tr>
<td>Overdetermined</td>
<td>0.44</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Table 7.9: Comparison of different fitting methods in the garbage test.

<table>
<thead>
<tr>
<th>Method</th>
<th>Normalized midpoint</th>
<th>Mahalanobis ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Convergence</td>
<td>Mean error</td>
</tr>
<tr>
<td>No minimization</td>
<td>0.22</td>
<td>0.57</td>
</tr>
<tr>
<td>All</td>
<td>0.13</td>
<td>0.68</td>
</tr>
<tr>
<td>Random</td>
<td>0.08</td>
<td>0.83</td>
</tr>
<tr>
<td>Close</td>
<td>0.05</td>
<td>1.30</td>
</tr>
<tr>
<td>Progressive</td>
<td>0.12</td>
<td>0.90</td>
</tr>
<tr>
<td>Minimal</td>
<td>0.01</td>
<td>1.83</td>
</tr>
<tr>
<td>Overdetermined</td>
<td>0.11</td>
<td>0.86</td>
</tr>
</tbody>
</table>

Table 7.10: Comparison of different fitting methods in the nightmare test.
7.4 Evaluation on real data

The data used for evaluation is an image sequence rendered with 3DStudio MAX. This sequence shows a Golf model car that is being rotated. The sequence was made in order to evaluate the performance of a pose estimation algorithm used in [19]. A couple of frames from this sequence are shown in figure 7.6. Lines extracted from these frames are shown in figure 7.7. This image sequence should be fairly easy to track since there is no other objects present in the picture except the vehicle. If tracking cannot be done on this sequence there is no way tracking will be successful in real video data captured from the WITAS platform.

![Figure 7.6: Example frames from the golf sequence used in evaluation.](image)

The tracker was given a initial estimate that was estimated by hand. The position and shape of the initial estimate is shown in figure 7.8. This estimate is fairly accurate, probably more accurate than what can be expected from an automated guess. After minimization in one frame the new estimate is used as an initial guess in the next frame.

Results from the attempted tracking are shown in figure 7.9. As shown, the tracker diverges very fast. In the very first frame it has diverged to a pose that is nothing like the true pose of the vehicle. It does not recover and is hopelessly lost throughout the rest of the tracking process.
7.4. EVALUATION ON REAL DATA

Figure 7.7: Lines extracted from the golf sequence used in tracking evaluation.

Figure 7.8: Illustration of the pose and shape of the initial estimate used in the tracking test.
Figure 7.9: Result of tracking experiment. The black lines are constructed using the estimated pose at each frame.
7.4.1 Conclusions

As shown in the previous section the current performance of the tracker is not enough to perform successful tracking on even relatively clean image sequences. There is absolutely no hope to achieve successful tracking in image sequences captured from the WITAS platform.
Chapter 8

Discussions

The different detection methods covered turned out to work fairly well. Even though the specific framework for tracking the resulting residual “blobs” in the difference image has not been investigated it should be a fairly simple task to develop this.

In this work there is a noticeable gap between the detection and the classification phases. The cause of this is time related. A lot of time was spent in an effort to get the tracker working properly and integration of the two parts was to be investigated after had been achieved. However, since the tracker never became practically usable the integration of the two part has not been investigated.

The tracker/classifier worked very poorly. If this is the fault of the method itself or if it is the rather naive implementations of some parts is not entirely obvious. Admittedly, there is a lot of work to be done with the segment matching and fitting policies but it might be the case that the input data will never be refined enough to perform adequate tracking nor classification.

It is stated in the beginning of this work that real time performance is an issue. While the detection algorithm should be able to run reasonably well in real time it is highly doubtful that the tracker/classifier will. There are a lot of time consuming steps in this algorithm.

The fitting of vehicle shape has not been investigated fully in this work. A big problem with increasing the number of free parameters in the minimization process is that the quality of the data need to be sufficient. While it might be possible to fit the vehicle shape if the matching and fitting processes are refined it is not possible in the current state of this work.

8.1 Future work

There is a lot of topics that could be investigated further in this work. This section will name a few.
8.1.1 Feature point replenishing

When tracking feature points it is inevitable that they will be lost after a while. This occurs for instance when the feature point has moved out of the picture. Once enough feature points have been lost the estimation process will become unstable or even impossible. The process of replenishing is to somehow seamlessly pick new feature points and incorporate them into the set of points currently being tracked. This is not as trivial as it may seem at first since the estimation of transform is done over several frames.

8.1.2 Refined detection algorithm

The detection algorithm works fairly well. One improvement that could be made is to instead of using just the reference image when computing a difference image one could use some mean image. The mean image could be computed continually using the frames before the target image. This way the fake response seen in the difference image could be reduced.

8.1.3 Model instantiation

Implementing hidden line removal in the model instantiation algorithm should improve the performance of both the matching and minimization algorithms. Currently, only hidden surfaces are removed, any lines that are only partially occluded will still be considered. There are even cases where lines that should be totally hidden are included in the instantiation.

8.1.4 Segment matching

As stated this work only covers a few very basic approaches too this. A number of extensions are possible. Using relational graphs for reasoning about which segments match is a possibility.

8.1.5 Data refining

One problem with the extracted segments is that they may have been split into several parts. As an example think about the lower edges of the side windows, ideally they should be one line to suit the vehicle model but they are most likely going to be extracted as two segments. Another possibility would be to filter out irrelevant lines in order not to confuse the minimization nor matching algorithms.

8.1.6 Fitting policies

The policies used for fitting in this work are very simple. None of them utilizes the possibility to keep certain lines static during the fitting process. More sophisticated polices might also include knowledge from the model instantiation. For instance
that a surface is about to appear should influence the fitting since it is hard to establish line matches correctly when the surface is almost parallel to the viewport.

8.1.7 Minimization process

In this work only Newton minimization has been used. The Levenberg-Marquardt minimization technique should be fully investigated. This has been omitted due to practical reasons.

8.1.8 Explicit time integration

The algorithms described in this work has no concept of time. It may be possible to refine the input data by utilizing the information from a number of frames and to fuse them somehow.

8.1.9 Adding external constraints

A lot of external constraints could be added. For instance, the size of the residual blob in the detection image could be used to restrict the possible size of the instantiated vehicle model. Further constraints could be added with the use of GIS (Geographic Information System). For instance, vehicle locations could be restricting to reside on a road.

8.1.10 Classification

Since the pure tracking performance of the proposed method was so poor any fitting of vehicle shape parameters was omitted. This should be included and a proper classifier should be investigated. Using Bayes or Maximum Likelihood classifiers with the parameter vectors comes to mind.
CHAPTER 8. DISCUSSIONS
Chapter 9

Summary

This thesis has examined two different topics, detection and classification of vehicles. Due to project limitations the topics had to be examined separately and no fully functional system has been developed. The integration of the two should be fairly straightforward. The detection part detects interesting regions to be classified.

For the purpose of vehicle detection a method based on feature point tracking and image registration has been proposed and evaluated. The performance of this method is satisfactory even though the real-time performance is somewhat questionable. Further work is needed to transform it into a fully useful object tracker but the current method should work as a foundation.

The method used for vehicle classification is based on model-based tracking using a flexible vehicle model. This model is described by 12 parameters that control the shape of the vehicle. The model is instantiated and matched to extracted lines from the image. Any errors between the instantiated model and the extracted lines is minimized using controlled Newton minimization. The results of this method is questionable. Not only is Newton minimization highly unstable but it is also dependent on obtaining correct segment matches. The topic of segment matching has not been investigated as much as needed. A couple of approaches to stabilize the Newton minimization has been investigated and show promising results. The real time performance of the classification method is also very questionable.

To summarize, the method for vehicle detection works well but is fairly computationally expensive. The method for vehicle classification has very poor performance and a lot of further research is needed to make it of any practical use. Even if convergence issues can be solved the real-time performance might be too poor.
Appendix A

The pin-hole camera model

A widely used model of the perspective projection that occurs in a camera system is the pin-hole camera model. The model is inspired by the way a real camera works. Figure A.1 shows how a point is projected onto the image plane. From this figure we see that:

\[ \frac{x'}{f} = \frac{x}{z} \quad (A.1) \]

where \( f \) is the focal distance of the camera model. \( x \) and \( z \) are coordinates of the point that is to be projected and \( x' \) is the projection of this point. If we apply the same process to the y-coordinate and collect the result in a vector we get:

\[ \begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x \\ y \end{pmatrix} \quad (A.2) \]

Figure A.1: The pin-hole camera model.
A convenient representation when working with the pin-hole model is homogeneous coordinates. We represent the original point using homogeneous coordinates:

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]  
(A.3)

The projection can now be achieved using a simple matrix multiplication:

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  \frac{1}{f}
\end{pmatrix}
= \begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & \frac{1}{f} & 0
\end{pmatrix}
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]  
(A.4)

The projection matrix can be scaled by an arbitrary factor, this scaling will be removed in the normalization process. Finally we see that when the homogeneous coordinates are normalized we get:

\[
\begin{pmatrix}
  x' \\
  y' \\
  f
\end{pmatrix}
= f
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\]  
(A.5)

where \(x'\) and \(y'\) are the same coordinates as in equation (A.2).
Bibliography


Index

Abstract, iii
Acknowledgments, v
Affine transformation, 14
All lines policy, 43
Camera model, 75
Centered midpoint distance, 40
Classes, 25
Classification evaluation, 53
Classification method, 23
Classification test setup, 53
Close policy, 44
Colour feature, description, 5
Conclusions, fitting evaluation, 62
Conclusions, matching evaluation, 61
Conclusions, real data, 67
Cubic spline interpolation, 21
Deformability feature, description, 6
Detection method, 9
Differentiation, errors, 32
Differentiation, image coordinates, 34
Differentiation, rotation, 36
Differentiation, shape, 36
Differentiation, translation, 35
Discussions, 69
Dynamic monitoring, 7
Error measure, 30
Error, parallel, 32
Error, perpendicular, 30
Evaluation, classification, 53
Evaluation, cost of image transformations, 52
Evaluation, cost of resampling, 49
Evaluation, detection, 47
Evaluation, fitting policies, 61
Evaluation, image transformations, 49
Evaluation, image warping, 49
Evaluation, matching methods, 55
Evaluation, real data, 64
Evaluation, resampling, 47
Evaluation, tracker, 47
Feature tracking, 11
Features, significant, 5
Fit policy, all lines, 43
Fit policy, close, 44
Fit policy, minimal, 44
Fit policy, Overdetermined, 44
Fit policy, progressive, 44
Fit policy, random, 45
Fitting parameters, 30
Fitting polices, 43
Fitting policies, evaluation, 61
Fixed lines, 38
Future work, 69
Garbage test results, for fitting evaluation, 62
Garbage test results, for matching evaluation, 60
Garbage test setup, 55
Homogeneous coordinates, 28, 29
Homographic transformation, 16
Homography estimation, 16
Ideal interpolation, 21
Image registration, 13
Image warping, 20
Image warping evaluation, 49
Incomplete test results, for fitting evaluation, 62

79
Incomplete test setup, 55
Initial estimate, 29
Instantiation, 28
Interpolation, 21
Interpolation kernels, 21
Introduction, 1
Inverse transformation, 20

Jacobian, calculation of, 32
Jacobian, definition, 32

Kanade-Lucas-Tomasi tracker, 11
Keywords, iii
KLT tracker, 11

L-M minimization, 37
Linear interpolation, 21
Lines, fixed, 38

Mahalanobis distance, 42
Mahalanobis ratio, 43
Matching, 39
Matching methods, evaluation, 55
Midpoint distance, 40
Minimal policy, 44
Minimization, L-M, 37
Minimization, Newton, 37
Model instantiation, 28
Model transformations, 28
Monitoring, 7
Monitoring, dynamic, 7
Monitoring, static, 8
Motion feature, description, 6

Newton minimization, 37
Nightmare test results, for fitting evaluation, 62
Nightmare test results, for matching evaluation, 60
Nightmare test setup, 55
Noisy test results, for fitting evaluation, 61
Noisy test results, for matching evaluation, 58, 60
Noisy test setup, 54

Normalized midpoint distance, 41
Notations, xi

Object detection, 9
Outlier detection, 19
Outlier handling, 19
Overdetermined policy, 44

Parallel error, 32
Parameter fitting, 30
Perfect test results, for fitting evaluation, 61
Perfect test results, for matching evaluation, 58
Perfect test setup, 54
Perpendicular error, 30
Perspective projection, 75
Pin-hole camera, 75
Platform description, 2
Problem conditions, 2
Problem description, 1
Problem domain, description, 2
Progressive policy, 44
Prototype, 25
Prototype vector, 25

Quadratic transformation, 15

Random policy, 45
Real data evaluation, 64
Related work, 7
Related work, Carnegie Mellon, 8
Related work, Cornell, 7
Related work, USC, 8
Report structure, 3
Resampling, 20
Resampling evaluation, 47
Results, detection, 47

Sampling theorem, 20
Segment matching, 39
Shape feature, description, 5
Significant features, 5
Sinc interpolation, 21
Static monitoring, 8
INDEX

Summary, 73
Synthetic test setup, 53

Test setup, 53
Test setup, garbage test, 55
Test setup, incomplete test, 55
Test setup, nightmare test, 55
Test setup, noisy test, 54
Test setup, perfect test, 54
Tracker evaluation, 47
Transformation, affine, 14
Transformation, homographic, 16
Transformation, quadratic, 15
Transformation, translational, 14
Translational transformation, 14

Vehicle classes, 25
Vehicle prototype, 25

WITAS, 1
Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

© [Andreas Böckert]