Online Calibration of Camera Roll Angle

Examensarbete utfört i Datorseende
vid Tekniska högskolan vid Linköpings universitet
av

Astrid de Laval

LiTH-ISY-EX--13/4688--SE

Handledare: **Hannes Ovrén**
ISY, Linköpings universitet
Emil Nilsson
Autoliv Electronics

Examinator: **Klas Nordberg**
ISY, Linköpings universitet

Linköping, 18 juni 2013

Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Online Calibration of Camera Roll Angle

Examensarbete utfört i Datorseende
vid Tekniska högskolan vid Linköpings universitet
av

Astrid de Laval

LiTH-ISY-EX--13/4688--SE

Linköping 2013

Linköpings universitet
TEKNISKA HÖGSKOLAN

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

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Avdelning, Institution
Division, Department

Avdelningen för systemteknik
Department of Electrical Engineering
SE-581 83 Linköping

Datum
Date

2013-06-18

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ _____

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ _____

ISBN
__

ISRN
LiTH-ISY-EX--13/4688--SE

Serietitel och serienummer
Title of series, numbering

ISSN
__

URL för elektronisk version
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-XXXXX

Titel
Title

Dynamisk kalibrering av kamerarollvinkeln
Online Calibration of Camera Roll Angle

Författare
Author

Astrid de Laval

Sammanfattning
Abstract

Dagens moderna bilar är ofta utrustade med ett vision-system som samlar in information om bilen och dess omgivning. Kamerakalibrering är extremt viktig för att bibehålla en hög precision i en bils säkerhetssystem. Kamerorna kalibreras offline vid montering i fabriken, men kamerans rotation kan ändras med tiden. Om vinklarna för den faktiska monteringen av kameran kan skattas kan dessa kompenseras för i mjukvaran. Alltså behövs en kalibrering av kameran under drift.

Detta examensarbete beskriver hur man under drift kan skatta kamerans rollvinkel. Två olika metoder har implementerats och jämförts. Den första detekterar vertikala kanter i bilden, såsom på hus och gatlyktor. Den andra metoden beräknar rollvinkeln genom att detektera registreringsskyltar på andra bilar framför kameran.

De två metoderna utvärderas och resultaten diskuteras. Resultaten av metoderna är väldigt varierande, och den metod som visade sig ge bäst resultat är den som detekterar vertikala kanter.

Nyckelord
Keywords

online kalibrering, rollvinkel, kamerakalibrering

Sammanfattning

Dagens moderna bilar är ofta utrustade med ett vision-system som samlar in information om bilen och dess omgivning. Kamerakalibrering är extremt viktig för att bibehålla en hög precision i en bils säkerhetssystem. Kamerorna kalibreras offline vid montering i fabriken, men kamerans rotation kan ändras med tiden. Om vinklarna för den faktiska monteringen av kameran kan skattas kan dessa kompenseras för i mjukvaran. Alltså behövs en kalibrering av kameran under drift.

Detta examensarbete beskriver hur man under drift kan skatta kamerans rollvinkel. Två olika metoder har implementerats och jämförts. Den första detekterar vertikala kanter i bilden, såsom på hus och gatlyktor. Den andra metoden beräknar rollvinkeln genom att detektera registreringsskyltar på andra bilar framför kameran.

De två metoderna utvärderas och resultaten diskuteras. Resultaten av metoderna är väldigt varierande, och den metod som visade sig ge bäst resultat är den som detekterar vertikala kanter.

Abstract

Modern day cars are often equipped with a vision system that collects information about the car and its surroundings. Camera calibration is extremely important in order to maintain high accuracy in an automotive safety applications. The cameras are calibrated offline in the factory, however the mounting of the camera may change slowly over time. If the angles of the actual mounting of the camera are known compensation for the angles can be done in software. Therefore, online calibration is desirable.

This master's thesis describes how to dynamically calibrate the roll angle. Two different methods have been implemented and compared.The first detects vertical edges in the image, such as houses and lamp posts. The second one method detects license plates on other cars in front of the camera in order to calculate the roll angle.

The two methods are evaluated and the results are discussed. The results of the methods are very varied, and the method that turned out to give the best results was the one that detects vertical edges.

Acknowledgments

Primarily, I would like to thank everyone at Autoliv for all the help during this thesis. Both for your expertise and for supplying all materials that was needed to carry out the project. A special thanks to my supervisor, Emil Nilsson, for quickly and cheerfully sharing your good ideas and knowledge.

Lastly, I would like to thank all my friends and family for your love and support. Special thanks to my parents, Maria and Peter, for always being there and for helping me through seventeen years of school.

Linköping, June 2013
Astrid de Laval

Contents

1 Introduction **1**
- 1.1 Impact of Roll Angle Error . . . 2
- 1.2 Related Work . . . 3
- 1.3 Autoliv . . . 3
- 1.4 Outline . . . 3

2 Theory **5**
- 2.1 Coordinate Systems . . . 5
- 2.2 Camera Model and Camera Rotation . . . 6
- 2.3 Canny Edge Detection . . . 7
- 2.4 Hough Transform . . . 7
- 2.5 Optimal Global Thresholding Using Otsu's Method . . . 8

3 Vertical Edge Method **11**
- 3.1 Overview . . . 12
- 3.2 Pre-processing . . . 13
- 3.3 Canny Edge Detection . . . 14
- 3.4 Hough Transform . . . 14
- 3.5 Lens Distortion Correction . . . 14
- 3.6 Weigthing and Roll Angle Calculation . . . 15

4 License Plate Detection Method **17**
- 4.1 Overview . . . 18
- 4.2 License Plate Location . . . 19
- 4.3 Canny Edge Detection with Automatic Thresholding . . . 20
- 4.4 Rectangle Detection Based on Hough Transform . . . 20

5 Experimental Results **23**
- 5.1 Comparison . . . 23
  - 5.1.1 Dataset 1 . . . 24
  - 5.1.2 Dataset 2 . . . 25
  - 5.1.3 Dataset 3 . . . 26
- 5.2 Vertical Edge Method Evaluation . . . 28

5.3 License Plate Method Evaluation 28

6 Concluding Remarks **31**
6.1 Conclusions 31
6.2 Future Work 32

A Camera Extrinsic Error **33**
A.1 Impact of Camera Extrinsic Error on Target Position Estimation . . 33
A.2 Theoretical Limits of Vision-Based Camera Extrinsic Calibration . 36

B Datasets **37**
B.1 Dataset 1 38
B.2 Dataset 2 39
B.3 Dataset 3 40

Bibliography **41**

1

Introduction

Modern day cars are often equipped with a vision system that collects information about the car and its surroundings. A camera-based system can help prevent many traffic accidents, for example by warning the driver about pedestrians and other cars. In order to maintain an automotive safety system with good performance an online calibration of the cameras in the system is needed.

Figure 1.1: *Autoliv's mono-vision camera mounted on the inside of the vehicle's wind shield.*

The extrinsic parameters include both the camera's translation and rotation, which describe the camera's position relative to the car it is mounted on. Calibration of the parameters can be done at the factory, when the camera is mounted on the car. Once these parameters are determined, any point in the world coordinate system can be transformed into a point in the image coordinate system.

The camera translation are not likely to change, to an extent which will affect the performance of the system, after the installation. But camera rotation may change because of vehicle vibration or accidentally by the vehicle user. Camera rotation has significant impact on vision-based target detection and position esti-

mation. If the angles of the actual mounting of the camera are known, compensation for the camera rotation can be done in software. Therefore, it is necessary for the vision system to be able to calibrate the camera online in order to always have up-to-date angle-information.

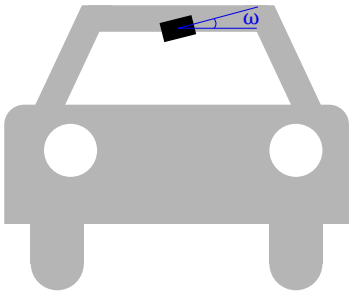

Figure 1.2: *A car with a mounted camera. ω is the roll angle between the vehicle and the camera.*

The idea behind this master's thesis is to:

- develop different methods for online calibration of the roll angle, between the camera and the car it is mounted on, using the image from the camera. See Figure 1.2.
- compare and evaluate these methods in order to determine which one that shows best performance.

1.1 Impact of Roll Angle Error

An online extrinsic camera calibration is very important in order to maintain a system with good performance. A small camera rotation error can cause large errors, in for example, the target position estimation which in turn can cause errors in other parts of the vision system. As seen in Table 1.1, a small roll angle error of 1 ° can cause estimation errors of several meters on off-center targets. A small roll rotation error can also cause large errors in the lane width estimation. See Appendix A.1 for more details about the camera and all calculations.

Distance	**Target at center of FOV**		**Target at edge of FOV**	
to target	Longitudinal Error	Lateral Error	Longitudinal Error	Lateral Error
10 m	0 m	0 m	0.86 m	0.63 m
30 m	0 m	0 m	7.78 m	5.66 m

Table 1.1: *Errors of target position estimation caused by* 1 ° *of camera roll rotation error for a 1000 pixel camera. The distance is the longitudinal distance to the target.*

The minimum detectable roll angle that can be found by this system can be calculated by making a few assumptions about the camera and its surroundings. This have been done in Appendix A.2 where the minimum detectable roll is estimated to approximately 0.094 °. Therefore, the desired precision of the online calibration system of the roll angle is set to 0.1 °.

1.2 Related Work

Using vertical lines or horizontal lines in the camera's surroundings in order to calibrate the camera have been developed in previous projects, for example by Lv et al. [2002].

Some work of estimating the roll angle using lane markers on the road have previously been developed, for example by Catalá-Prat et al. [2006]. However, there was no time to implement this method during the master's thesis.

1.3 Autoliv

Autoliv was founded in Vårgårda in 1953 by the two brothers Lennart and Stig Lindblad as a dealership and repair shop for motor vehicles called Lindblads Autoservice. Today Autoliv Inc. is one of the leading companies in automotive safety and they produce a variety of safety products such as airbags, seatbelts and vision systems.

Autoliv Electronics is a division of Autoliv. In Linköping they develop systems for night vision, mono vision, stereo vision and also some radar-vision fusion. The vision systems features are for example lane departure warning, speed sign recognition, vehicle collision mitigation, pedestrian warning and high beam automation.

1.4 Outline

Chapter 2 describes some of the basic theory that will be used throughout the thesis, such as a short introduction to basic edge detection algorithms and the Hough transform. The coordinate systems used are also presented here.

Each implemented method has its own chapter describing both how the algorithms are outlined as well as how they were implemented in finer detail. The results of the methods are presented in Chapter 5. Chapter 6 includes the conclusions that were made throughout the thesis and possible future development.

2
Theory

This chapter describes some basic theory that is used in the implemented methods, such as basic edge detection algorithms and the Hough transform. The chapter also present the coordinate systems used throughout the thesis.

2.1 Coordinate Systems

In the thesis three different coordinate systems have been used, an image coordinate system, a camera coordinate system and a world coordinate system. The coordinate system used for the camera is a right-handed system with the x-axis pointing in the direction of travel and the z-axis pointing upwards. The camera coordinate system is illustrated in Figure 2.1.

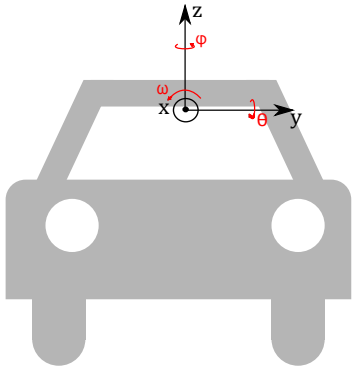

Figure 2.1: *Coordinate system of the camera with the angles roll (ω), pitch (θ) and yaw (ϕ).*

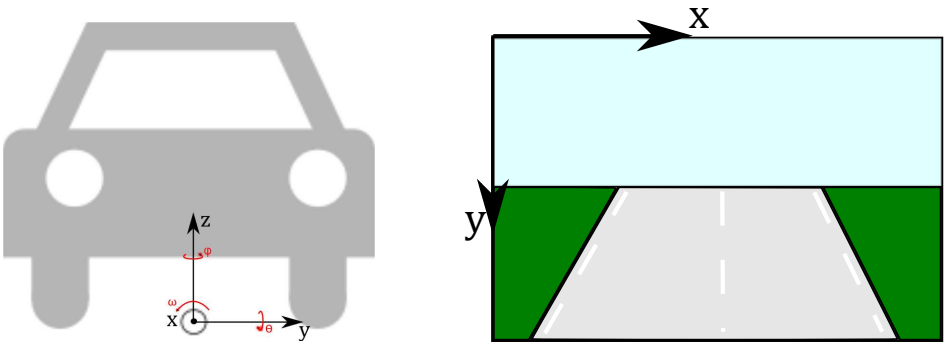

Figure 2.2: *To the left: World coordinate system with the angles roll (ω), pitch (θ) and yaw (ϕ). To the right: Coordinate system of the image.*

The world coordinate system is illustrated in Figure 2.2 and it is located on the road surface just beneath the camera, with the x axes pointing in the direction of travel. The image coordinate system has its origin in the top left corner of the image with x and y axes pointing to the right and downwards, respectively. The image coordinate system is also illustrated in Figure 2.2.

2.2 Camera Model and Camera Rotation

For a pinhole camera model, any point in the world coordinate system can be mapped to the image coordinate system using

$$s\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = A\left[R_e|t\right]\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, \tag{2.1}$$

where R_e is the extrinsic rotation matrix, t the translation matrix and s is a scale factor. This is described by Hartley and Zisserman [2004]. A is the intrinsic camera matrix, which maps the camera coordinate system to the image coordinate system. c_x and c_y are the principal point, γ the skewing of the image and f_x and f_y being the focal length measured in width and height of the pixels.

$$A = \begin{pmatrix} c_x & -f_x & \gamma \\ c_y & 0 & -f_y \\ 1 & 0 & 0 \end{pmatrix} \tag{2.2}$$

The extrinsic camera parameters, $\left[R_e|t\right]$, which transforms the world coordinate system to the camera coordinate system, is defined as

$$\left[R_e|t\right] = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \tag{2.3}$$

From the extrinsic camera rotation, R_e, the angles roll (ω), pitch (θ) and yaw (ϕ) can be extracted, as described in Lesk [1986], as

$$\begin{cases} \omega = \text{arctan2}(r_{32}, r_{33}) \\ \theta = \text{arctan2}(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ \phi = \text{arctan2}(r_{21}, r_{11}) \end{cases} . \tag{2.4}$$

2.3 Canny Edge Detection

The Canny Edge Detection is a commonly used algorithm to detect edges in an image. First the input image is smoothed with a Gaussian kernel, $G(x, y)$. This can be seen in Equation (2.5) below, where $*$ denotes the 2D convolution between the image and the kernel.

$$I_s(x, y) = G(x, y) * I(x, y) \tag{2.5}$$

Then the gradient magnitude of the image, $M(x, y)$, and the angle image, $\alpha(x, y)$ are calculated as

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \qquad \text{with} \qquad g_x = \frac{\partial I_s}{\partial x}, \;\; g_y = \frac{\partial I_s}{\partial y} \tag{2.6}$$

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_x}{g_y}\right] \tag{2.7}$$

$M(x, y)$ typically contains ridges around the local maxima. Therefore, nonmaxima suppression is used on the gradient image to thin the ridges. The nonmaxima-suppressed image, g_N is then thresholded using one low threshold, T_L and one high threshold, T_H.

$$\begin{aligned} g_{NH}(x, y) &= g_N(x, y) \geq T_H \\ g_{NL}(x, y) &= g_N(x, y) \geq T_L \end{aligned} \tag{2.8}$$

After the double thresholding, connectivity analysis is performed to detect and link the edges found in the images. A more comprehensive description of the Canny edge detection and its steps can be found in Gonzalez and Woods [2008].

2.4 Hough Transform

The Hough transform, as described in Gonzalez and Woods [2008], is a technique which for example can be used to detect straight lines within an image. Using a parametric notation a line can be described as

$$x \cos\alpha + y \sin\alpha = \rho. \tag{2.9}$$

The parameter ρ is the normal distance from origo and α the normal angle of a straight line and they can be geometrically interpreted as seen in Figure 2.3. A point in the (x, y)-space is mapped to a sinusoid in the (ρ, α)-space, the Hough space. Since all points on a line are mapped to a set of sinusoids, a line can be interpreted as the intersection of the curves, i.e. a line is mapped to a point.

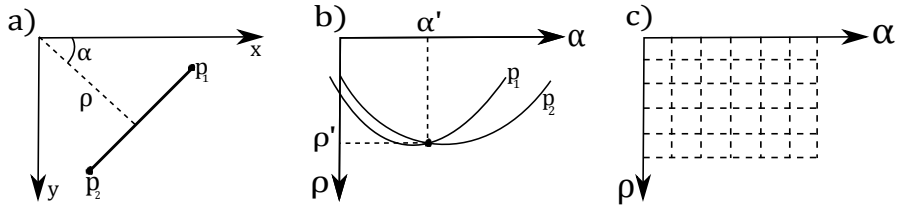

Figure 2.3: *a) Parameterized line in the (x, y)-space. b) Sinusoids of points p_1 and p_2 in the (ρ, α)-space; the intersection (ρ', α') corresponds to the line in a). c) Division of accumulator cells in the (ρ, α)-space*

When the transform is implemented the (ρ, α)-space is divided into finite intervals, so-called accumulator cells. Each (x_i, y_i) is transformed into a discretized (ρ, α)-curve and the accumulator cells which lie along this curve are incremented. The resulting peaks in the accumulator cells are the lines that are found in the image.

2.5 Optimal Global Thresholding Using Otsu's Method

Otsu's method is used in order to automatically threshold an image based on the shape of the image intensity histogram. It assumes that the image contains foreground and background pixels and aims to separate them, i.e. maximize the between-class variance. The method is described by Gonzalez and Woods [2008].

First the normalized histogram, with L number of components, of the image is calculated and denoted as p_i, $i = 0, 1, 2, ..., L-1$. The cumulative sums can be calculated as

$$P_1(k) = \sum_{i=0}^{k} p_i \tag{2.10}$$

The cumulative mean, $m(k)$, and the global mean, m_G, of the image intensity is also calculated

$$m(k) = \sum_{i=0}^{k} i p_i \tag{2.11}$$

$$m_G = \sum_{i=0}^{L-1} i p_i \tag{2.12}$$

The between class-variance is computed as

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]} \tag{2.13}$$

The threshold, k^*, is obtained as the value of k for which $\sigma_B^2(k)$ is maximum. Then the separability measure, η^* can be computed as

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2} \mid_{k=k^*} \tag{2.14}$$

3

Vertical Edge Method

This chapter describes the method that finds vertical lines and edges in an image and how they are used to estimate the roll angle. First the outline of the algorithm is presented, and then each step is more throughly explained. Figure 3.1 shows an example of an image with the detected vertical lines.

Figure 3.1: *Image with the detected edges.*

3.1 Overview

The method aims to calculate the roll angle by finding objects in the world that are vertical, such as house walls. These objects are used as a straight reference, i.e. the roll angle is calculated relative to objects in the world. The idea is also that detected lines that are not straight, from for example lamp posts, will cancel each other with time.

The algorithm is called for every frame the car is moving. This is so that edges found when the car is standing still do not affect the roll angle result more than any other edge.

In the first step of the vertical edge method the image is pre-processed and then the Canny edge detection, Section 3.3, is used to extract the edges in the image. Once these have been found the Hough transform, Section 3.4, is used to find the vertical straight lines from the edges. All lines are corrected against lens distortion and to get a more reliable result, all lines are weighted before the roll angle is calculated. The weighting of each line is based on its length as well as the number of lines that are parallel to the line.

In the vertical edge method the roll angle value is the angle of the detected lines. An overview of how the algorithm is constructed can be found in the flowchart below, Figure 3.2.

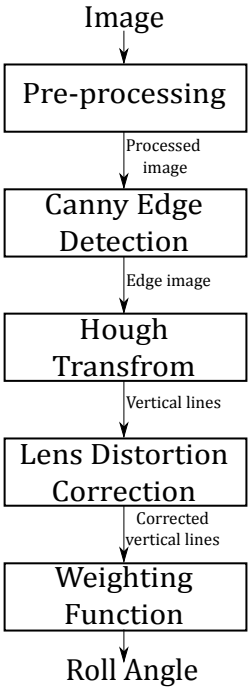

Figure 3.2: *Flowchart of the the method that uses vertical edges.*

As also will be stated later, in Section 5.2, by correcting the lens distortion before the Hough transform will probably increase the performance of the algorithm.

3.2 Pre-processing

The image is pre-processed in order to make the method more robust against illumination changes, i.e. make it invariant to scalings in the intensity. As the Canny edge detection contains many adjustable parameters which affect the result of the algorithm, the idea behind the pre-processing is also to obtain images that work with a fixed set of parameters.

The pre-processing is done by removing the mean intensity and divide with the standard deviation of the entire image

$$\hat{I}(\mathbf{x}) = \frac{I(\mathbf{x}) - \mu_I}{\sigma_I}. \tag{3.1}$$

μ_I and σ_I are the mean and the standard deviation of the image I, respectively. Figure 3.3, shows an image before and after the pre-processing step. A risk of dividing by the standard deviation is that the edges with a very low contrast, which actually should not be found in the image, are detected. But no case has been found when this has been a significant problem.

Figure 3.3: *The image before and after the pre-processing step.*

3.3 Canny Edge Detection

After the pre-processing all edges are detected in the image frame. This is done using the Canny edge detection, which is described in Section 2.3. The algorithm uses a Gaussian filter of size 13×13 with the standard deviation $\sigma = 3$. The thresholds are set to $T_H = 0.01$ and $T_L = 0.4 \cdot T_H$. In Figure 3.4 the edges detected by the Canny edge detection are shown.

In the vertical edge method Otsu's method was not used since it was not necessary in this method. This is because the vertical edge method is not sensitive to missed lines, unlike the license plate method.

Figure 3.4: *The edges detected in the image.*

3.4 Hough Transform

A Hough transform is applied to extract the lines that are straight from the edge image. All edges are transformed to the Hough space and the peaks are found using nonmaxima suppression. Since the vertical straight lines are wanted only $\alpha \in [-5°, 5°]$, is considered, with α defined as in Section 2.4. The resolution of the accumulator cells are set to 0.1°.

Some additional constraints are set on the lines that are detected. The lines should be at least 60 pixels long but a small gap in a line of 5 pixels or less is permitted. These parameters have been tuned after some empirical tests.

3.5 Lens Distortion Correction

Most camera lenses have some kind of lens distortion, which can distort the angles of the detected lines. However this distortion can be corrected in software. This is described by Hugemann [2010].

The distorted image points (x_d, y_d) can be transformed using the distortion center (x_c, y_c), the radial distortion coefficients, K_n, where n is the number of coefficients

used.

$$\begin{aligned} x_u &= (x_d - x_c)(1 + K_1 r^2 + K_2 r^4 + ...) \\ y_u &= (y_d - y_c)(1 + K_1 r^2 + K_2 r^4 + ...) \end{aligned} \tag{3.2}$$

$$\text{with} \qquad r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$$

The coefficients and the radial distortion center are received from Autoliv. All detected lines are transformed to correct for the distortion and new α values are calculated for all lines.

3.6 Weigthing and Roll Angle Calculation

An important step to gain robustness in the system is the weighting of all the lines that are found. Since the algorithm detects both houses, trees and lamp posts, many lines that are detected will result in pretty bad angles. Therefore the weighting may improve the results significantly.

The weighting in this algorithm aims to find geometrical patterns, such as many parallel lines, since this is something that often appears on houses. Also, lines that are long are highly weighted.

A weight measure is set for each line based on the two features described above. This is also described in Algorithm 1 below.

Algorithm 1 The weighting of the lines.

allRolls ← all calculated roll angle values
allLines ← all lines detected in the frame
num_{lines} ← number of lines detected
if $num_{lines} \geq 15$ **then**
  for $i = 1 \rightarrow num_{lines}$ **do**
    $line \leftarrow$ **allLines**(i)
    $w \leftarrow$ floor(length($line$)/60) ▷ Will always be 1 or higher
    $num_{parallel} \leftarrow$ number of lines that are parallel to $line$
    if $num_{parallel} \geq 2$ **then**
      $w = w \cdot num_{parallel}$
    end if
    for $j = 1 \rightarrow w$ **do**
      allRolls ← roll angle of $line$
    end for
  end for
end if

First the weight is set based on the length of the line and since all lines are at least 60 pixels long this weight measure will be set to one or higher, see Algorithm 1. If there are at least two other lines that are parallel to this line, the weight is

multiplied with the number of parallel lines. The final roll angle values are stored in a vector, **allRolls**, of dimension, $1 \times 100\,000$, which is used to calculate the mean and standard deviation values. This is done using the MATLAB functions mean() and std(). By using the last 100 000 roll angle samples only the calculated roll angle values from approximately the last half hour is used to estimate the roll angle.

If the number of lines detected are 15 or less the value of the roll angle is not updated. This is to make the system handle non-city environments as well, where most lines are bad. However it is almost only inside the cities, where there are houses, that the algorithm detects 15 lines or more.

4

License Plate Detection Method

This chapter describes the algorithm which uses other cars in front of the camera to estimate the roll angle. First, all assumptions that are made in this chapter are presented, together with an outline of the algorithm. Then the steps are described one by one. Figure 4.1 shows an example of a license plate that was detected by the algorithm.

Figure 4.1: *Car with the detected license plate.*

4.1 Overview

The idea behind the method is to use other cars in front of the camera as a reference in order to calculate the roll angle. The license plates on the cars were chosen as a reference since they are found on all cars and have roughly the same appearance, i.e. a rectangle containing some letters or numbers.

The algorithm runs when another car is within 5 m from the camera and the car is driving slowly. It is assumed that the two cars is on the same planar surface and that the car in front has a license plate that is clearly visible. The algorithm has data about the position of other cars in the image as its input. This data is received from Autoliv's other functions which, for example, tracks other cars.

First a rough estimate of the license plate's location is done and this smaller image of the license plate is then used in the edge detection step. A Canny edge detection is performed using automatic thresholds from Otsu's method. Using the Hough transform all horizontal and vertical lines can be found and these are used to find rectangular patterns with license plate dimensions. Once the plate has been detected the resulting value of the roll angle is calculated as the rotation of the plate relative the image axes.

In this method the lens distortion has a very small impact on the detected lines, since they are fairly close to the distortion center, and will therefore not be used.

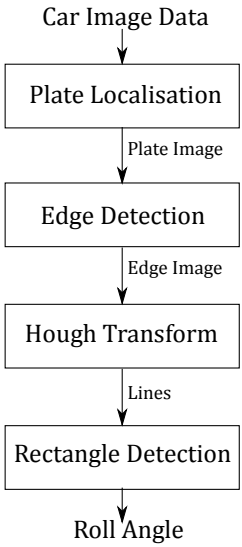

Figure 4.2: *Flowchart of the the method that detects and uses license plates on other cars to calculate the roll angle.*

4.2 License Plate Location

Once the image of the car has been received, a rough localisation of the plate's position is performed. This is done in order to narrow the area which is later used to find an automatic threshold, which will be described in Section 4.3. Lighting conditions, glitters on the car and the car headlights affect the edge detection. Since the algorithm only runs when another car is within 5 m from the camera, it is important to try to get good results when the opportunity arises. Narrowing the area of the license plate location increases the likelihood of success by the algorithm.

It is assumed that the plate is visible and that it contains letters and numbers. By applying a Sobel operator in the x-direction, as seen in (4.1), the derivatives in the horizontal direction can be found, i.e. the vertical lines on the car. The Sobel filter is described by Gonzalez and Woods [2008].

$$I_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I \tag{4.1}$$

The filter is applied on the part of the image as illustrated in Figure 4.3, since it is most likely to contain the plate and has small vertical variations except from the license plate. If the text on the plate is visible this area will have many small vertical changes in the horizontal direction, as can be seen in Figure 4.3.

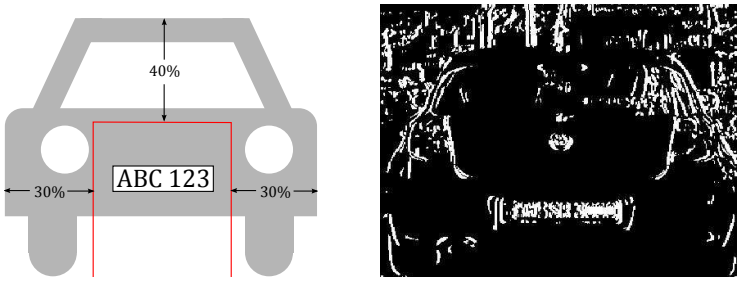

Figure 4.3: *To the left: A car with the area which is assumed to contain the license plate. To the right: The horizontal derivatives of a car as a binary image.*

The gradient image of the car is thresholded and the number of changes in the horizontal direction is counted for each row. A patch around the rows with most derivatives changes are extracted. The resulting image of the license plate is shown in Figure 4.4.

Figure 4.4: *The resulting detected license plate.*

4.3 Canny Edge Detection with Automatic Thresholding

In order to find the edges in the image the Canny edge detection, as described in Section 2.3, is applied on the image of the license plate. A Gaussian filter of size 5×5 is used and with $\sigma = 1$.

The thresholds, T_H and T_L, are chosen by first using Otsu's method to automatically find a threshold, T. This method is described in Section 2.5. The thresholds are chosen as $T_H = 0.4 \cdot T$ and $T_L = 0.1 \cdot T_H$. These value have empirically been determined by tuning of the parameters.

Figure 4.5: *Edge image of the detected license plate.*

4.4 Rectangle Detection Based on Hough Transform

A Hough transform, as described in Section 2.4, is applied on the edge image containing the license plate. Since angles as big as ±5° very rarely appears, a smaller interval of ±3° is chosen in this method, which makes the system faster. Therefore, both lines in the horizontal direction, $\alpha_1 \in [87°, 93°]$, and vertical direction, $\alpha_2 \in [-3°, 3°]$, are extracted from the peaks in the Hough space.

When the lines in the image have been computed these are used to find rectangular patterns. Some basic assumptions are made of the lines in a rectangle as well as in this case, that the detected rectangle has the dimensions of a license plate.

- There are two pairs of lines that are parallel and approximately has the same length.
- There are two such pairs that are perpendicular to each other and located close to each other.
- The dimension of a Swedish license plate is about 480×110 mm. The ratio between the detected pairs of lines should be approximately the same as the ratio of a license plate.

$$0.2 \leq \frac{\text{length of the vertical lines}}{\text{length of the horizontal lines}} \leq 0.25 \tag{4.2}$$

The rectangle detection is performed according to Jung and Schramm [2004]. The peaks in the Hough space are scanned and paired together if they satisfy the conditions that they are close to parallel and that the peaks are about the same

height according to

$$\begin{aligned} &\left|\alpha_i - \alpha_j\right| < T_\alpha \\ &\left|C(\rho_i, \alpha_i) - C(\rho_j, \alpha_j)\right| < T_C \frac{C(\rho_i,\alpha_i)+C(\rho_j,\alpha_j)}{2} \end{aligned} \quad . \tag{4.3}$$

α and ρ are the normal angle and the normal distance of a line, respectively. $C(\rho_i, \alpha_i)$ are the edge points in Hough space and the local maxima of $C(\rho_i, \alpha_i)$ are the peaks in the Hough space. Besides the constraints in (4.3) the system checks the lines' locations. The vertical lines should be located approximately at same place in the image y-direction and that the horizontal lines should be located approximately at the same place in the x-direction.

The angular threshold, T_α, is the permitted angular difference between the two lines and is set to 0.5°. T_C is a threshold that controls the lengths of the lines and is set to 0.3. Once the peaks have been paired together, each pair of peaks form an extended peak $P_k = (\pm\xi_k, \beta_k)$, where

$$\xi_k = \tfrac{1}{2}\left|\rho_i - \rho_j\right| \quad \text{and} \quad \beta_k = \tfrac{1}{2}(\alpha_i + \alpha_j) \tag{4.4}$$

The extended peaks are compared to find pairs of parallel lines that are orthogonal, i.e. satisfying the condition

$$\Delta\beta = \left|\left|\beta_k - \beta_l\right| - 90°\right| < T_\beta. \tag{4.5}$$

T_β is an angular threshold which is set to 0.5°. The algorithm also checks that the pair of parallel lines are close to each other, which means that the endpoints of the vertical lines should be close the end points of the horizontal lines. These pair of peaks, that satisfies these constraints, form a rectangle.

Lastly the the condition of the dimensions of the rectangle, (4.2), is checked. In the case that more than one rectangle is found, the one with the minimum $\Delta\beta$ is chosen by the algorithm.

The roll angle is then calculated as the rotation of the license plate in the image. This roll angle value is stored in a vector, similarly to Section 4.4, and but with the dimensions $1 \times 1\,000$. This vector is then used to calculate the mean and standard deviation value over multiple frames.

5
Experimental Results

This chapter contains all results from the methods that were implemented in this thesis. All datasets are collected from different cameras mounted on different cars. Since the angles are small and the offline calibration is rarely performed, ground truth is not available for all datasets. Therefore, an extensive comparison between the methods and their ability to cope in different driving environments, is important in order to evaluate the performance of the system. Since both methods work well in a city environment, all tests have been performed with some city scenery.

Three data collections have been chosen and the results of the methods on the different datasets are presented one by one.

5.1 Comparison

Both methods have all been tested on each dataset, however not on the exactly same frames. But since the angles are changing very slowly the roll angle is very likely to be the same for both methods when tested on the same dataset.

The license plate method is only called while the car is driving slowly and is less than 5 m behind another car. Therefore, the datasets that are used to evaluate this method are cut out sequences from the current dataset where these constraints are fulfilled.

For the vertical edge method the roll angle sample values are not displayed in the plots. Since they are very numerous and scattered they will cover the whole figure, which makes the mean and standard deviation values disappear.

5.1.1 Dataset 1

The first dataset is from a data collection that was made especially for this master's thesis. The camera had been calibrated before the data collection started and the offline calibrated roll angle was −0.8°.

The dataset starts in an environment outside a city, with few houses and buildings. Then the cars enters a city where it drives around for most of the set. The dataset concludes with the car leaving the city to return to the where it started. See Figure B.1 and Figure B.2 in Appendix B, as examples of the scenery.

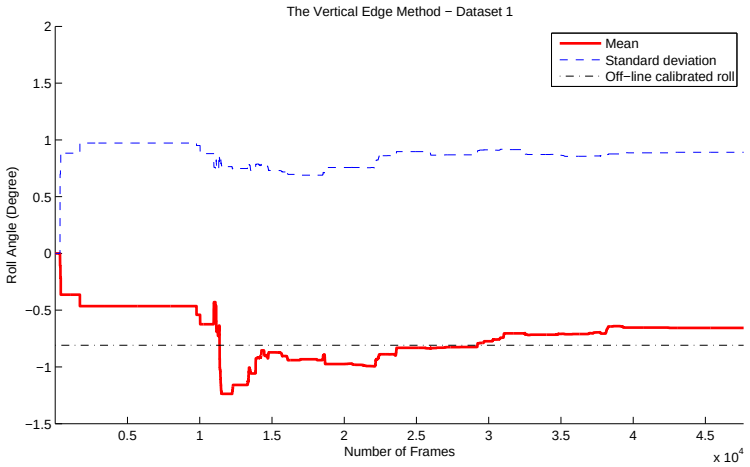

Figure 5.1: *The roll angle result using the vertical edge method.*

The vertical edge method, as seen in Figure 5.1, shows promising result. The method converges quite quickly to the ground truth value as soon as the car enters the city. The system also handles the scenery with few houses, in the beginning and end, pretty good. The final roll angle ends up at approximately −0.7°. However, the vertical edge method has a large standard deviation.

In Figure 5.2 the results using the license plate method are shown. On this dataset the calculated roll angles are quite varied and it is difficult to say what the final angle is since the roll angle samples vary greatly on different cars. The result from the set ends up close to the ground truth value. However, this is because of one car that, located approximately in the center of the set, which affects the final result extremely much. The final roll angle value is around −0.45° and the standard deviation is approximately 1.

On this dataset the method using vertical edges is superior to the license plate method, particularly when driving in the city. Its mean roll angle is closest to the ground truth value and stays there for most of the set.

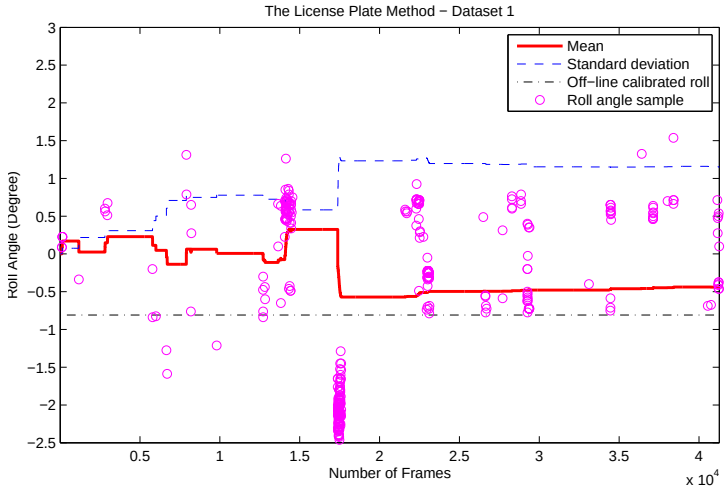

Figure 5.2: *The roll angle result using the license plate method.*

5.1.2 Dataset 2

The second dataset, that was used to evaluate the performance of the system, starts in the center of a city, drives around there for a while and then leaves for a more rural scenery. See Figure B.3 and Figure B.4 in Appendix B, as examples of the scenery.

There is no ground truth available for this dataset, therefore only a comparison between the methods can be used in order to evaluate the performance of the system.

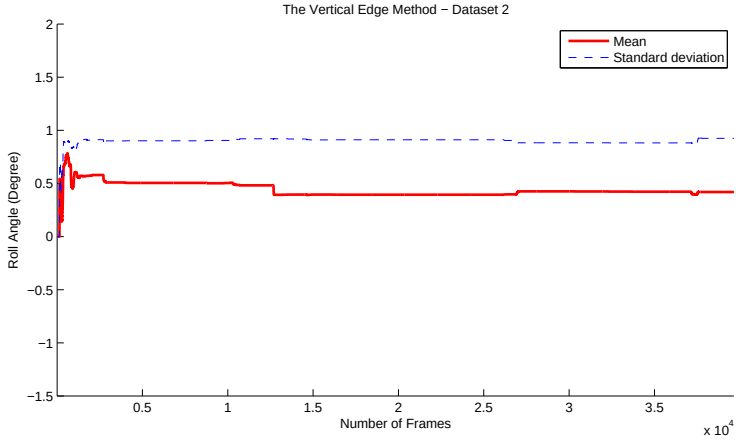

Figure 5.3: *The roll angle result using the vertical edge method.*

The results from the vertical line method are shown in Figure 5.3. Since the dataset starts by driving in a city with many houses and other edges the system shows a fast convergence. Then the roll angle value stays at that value throughout the dataset and ends up around 0.5°. On this dataset the vertical edge method handles the non-city scenery in the end very well.

On this dataset the calculated roll angle values from license plates are much more close to each other than on the first dataset, as seen in Figure 5.4. The method is converging towards the same value as the vertical edge method, around 0.5°.

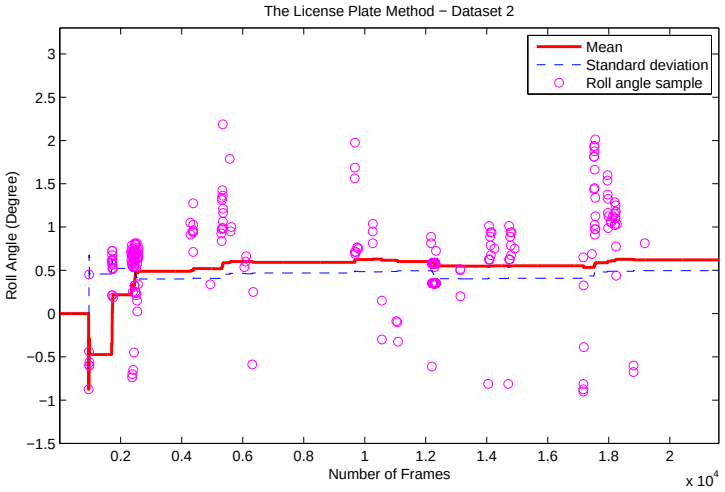

Figure 5.4: *The roll angle result using the license plate method.*

The vertical edge method and the license plate method are converging towards the same value. The mean of both methods are pretty stable around this value and converges quite quickly. This might indicate that the correct roll angle value may be close to 0.5°. The license plate method has a much lower standard deviation, however it should be noted that the vertical edge method uses much more samples in order to calculate the roll angle value.

On this dataset the license plate method shows better results since it has a much lower standard deviation, close to zero, unlike the vertical edge method.

5.1.3 Dataset 3

The third dataset is also a dataset without any ground truth values. Almost the entire set consists of a car driving on major city roads, which means that there are fewer houses located right next to the side of the road. See Figure B.5 and Figure B.6 in Appendix B.

The vertical edge method results in a final roll angle of approximately −0.1°. This is shown in Figure 5.5. This dataset does not result in the same rapid convergence

as for the second set. The vertical edge methods works a lot better in inner city environments, since the roll angle is then updated very regularly.

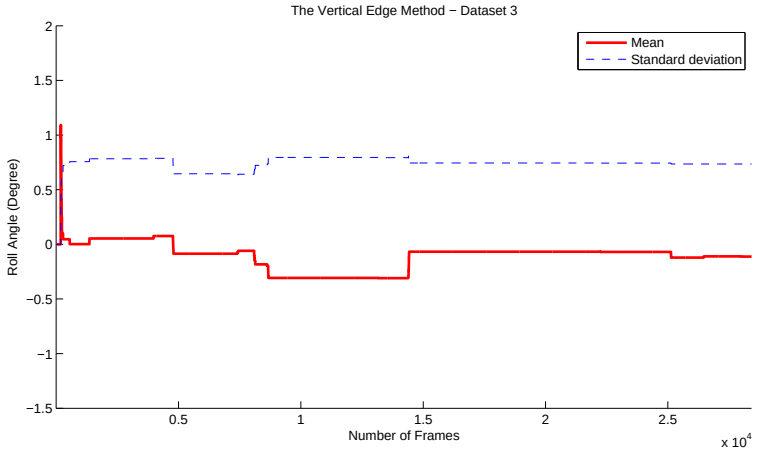

Figure 5.5: *The roll angle result using the vertical edge method.*

The final roll angle value using the license plate method, as shown in Figure 5.6, is around 0.25°. The results seems pretty good, however the resulting mean roll angle value changes its value quite a lot halfway through the set.

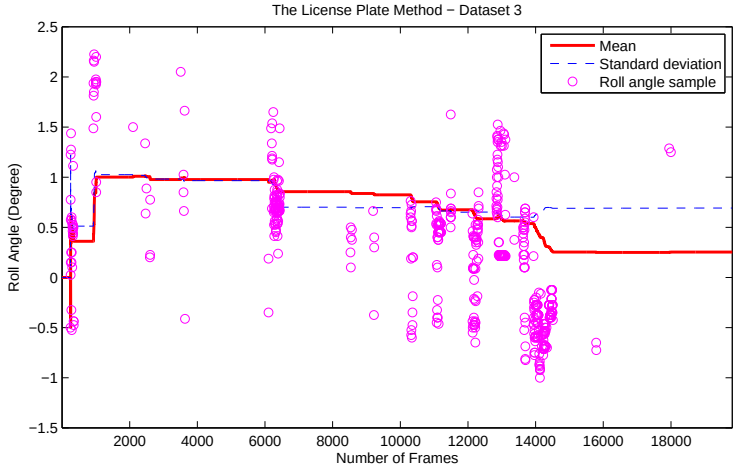

Figure 5.6: *The roll angle result using the license plate method.*

It is hard in this case to draw any conclusions about the true roll angle, since no method stands out as more superior and confident than the others.

On this dataset it is quite hard to determine which method that shows the best performance. Both the vertical edge method and the license plate method have approximately the same standard deviation and their mean value are quite close to each other.

5.2 Vertical Edge Method Evaluation

The method using vertical edges turned out to be the method that showed the best results in the performed tests. For the first dataset it ends up very close to the ground truth value. The standard deviation is very large for all datasets, but the number of samples that is used in order to calculate the roll angle is much larger than for the license plate method. The roll angle is also updated much more regularly.

If the car is driving in a city environment the roll angle value is updated regularly and in each frame many lines are detected. By using the constraint that at least 15 lines have to be detected in order to update the roll angle value, the vertical edge method can to some extent also cope with the case when the car is driving in a non-city environment, where mostly finds lines that are bad. Since it then rarely detects 15 vertical lines, this means that the method will not update the angle result.

The standard deviation of the result is large since the algorithm always will detect angles that are bad. But the weighting of the detected lines can in most cases weight the correct angles very well which cancels out these incorrect edges. Therefore, the impact of bad vertical edges and missed detections often disappears in the big amount of calculated values.

The algorithm is fast and it could probably be improved if the lens distortion correction is made on the entire image in the beginning of the algorithm. The algorithm could also be improved by making the weight function more continuous so that the weight measures do not need to be integers.

5.3 License Plate Method Evaluation

The license plate method gives varying result. The roll angle values are calculated in batches occasionally, since the algorithm runs rarely. This makes the impact of incorrect values very large. Also, each successful frame only provides one value, unlike the vertical edge method which provides at least 15 different values each for successful frame. In order to get good results from the algorithm many license plates need to be detected on many different cars. The datasets have too few cars for the roll angle value to be completely reliable. However, assuming that the car drives often in cities and sometimes during rush-hours it may be a good alternative since it is very fast and may give pretty good results after a couple of days of driving. In order for the method to work, probably a couple of hundred cars.

The system sometimes fails to detect license plates. Sometimes the localization step fails to extract the area of the plate. It can occur on dirty cars where the dirt forms vertical stripes, if the text on the plate is not visible, or if there are some other text on the car. The rectangle detection can also fail, for example if there is a small contrast difference between the plate and the car or if one of the sides of the rectangle is not visible for the system.

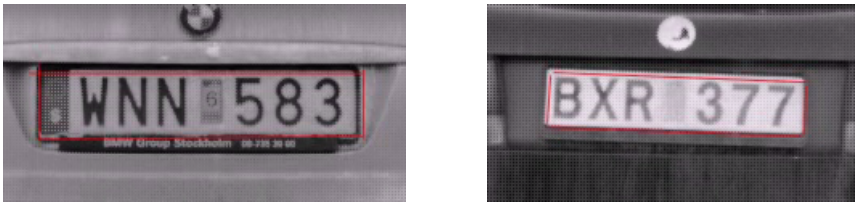

Figure 5.7: *Two plates that were successfully found by the system. The right plate is attached on the car with an inclination.*

In Figure 5.7 are two license plates that were successfully detected. The left plate is an example of an ideal plate that was easy to detect and that is mounted straight on the car. Unlike the right one that, just as many plates are, has a small inclination.

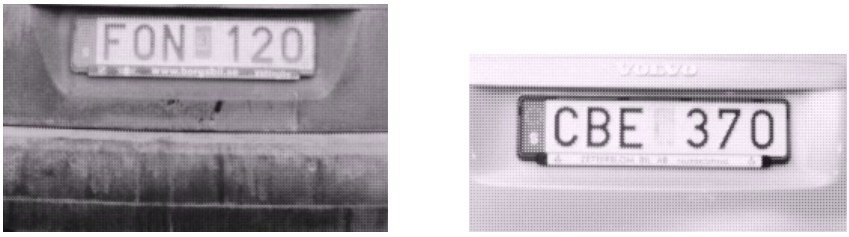

Figure 5.8: *Two license plates that were not found by the system.*

In Figure 5.8 are two examples of license plates that were not found by the system. The left one was not detected since the localization algorithm detected the dirt stripes instead of the actual plate. The license plate to the right was not detected since the lowermost side of the plate was not detected by the system.

6

Concluding Remarks

This final chapter includes the conclusions that were made during this master's thesis project. All results are discussed and possible future work are presented.

6.1 Conclusions

Two methods to dynamically calibrate the roll angle between the vehicle and the camera have been implemented and evaluated. One finds vertical edges in the camera's surroundings and one uses license plates on other cars in front of the camera.

The roll angles that shall be detected are very small and the precision wanted is very high, in the order of $0.1°$. It is hard to know what to consider as a straight reference in the world. The cars, both the one with the camera and others in front of the camera, have a small arbitrary inclination towards the road surface depending on the number of passengers and how the cars are loaded. All roads also have a small inclination to enable the water drainage. Furthermore, it is not certain that all houses are built entirely straight. Thus, even a camera that is completely straight on the car has an arbitrary rotation against most things that are used as a straight reference in this thesis. However, since everything is a little skewed the rotations will hopefully cancel out each other and the results may still be quite good after some time.

Since there is not much ground truth available it is hard to evaluate the performance of the system on all datasets. It has been a big difficulty throughout the project not knowing if any result is correct or not. Especially since the precision wanted is very high.

The vertical edge method proved to be the best one, since it gets a lot of value updates and it is quite easy and intuitive to weight them. A disadvantage with this method is that there are many external rotations that affects the result, such as the inclination of the road and the detected objects.

The license plate method showed varying results since it is updated rarely. However, it may be a good idea to use other cars if the method is improved or changed. If the system uses an object on the same inclined surface, and assuming that most cars are not heavily loaded, both the car with the camera and the one in front of the camera should probably lean approximately the same amount. This results in roll angle values that probably will be quite close to the actual angle.

6.2 Future Work

There are several things that can be done in future development of the online roll angle calibration system. To gain a more smooth value of the estimated roll angle some more extensive filtering could be applied on the output data. For example, a Kalman filter, could probably be good for noisy roll angle values and increase the performance of both methods.

The vertical edge method could be improved by moving the lens distortion correction step before applying the Hough transform. However, the system will probably be slower. Since the system only works in in a city environment it could perhaps be combined with a method that only works outside cities. For example a system that detects the horizon in the image frames.

The license plate method sometimes fails to detect the plates and could probably be improved to make it more robust, for example by tuning of all the parameters used in the method. If the car stands still for a long time, behind a license plate that is skewed or if any of the cars are inclined, it will have a very large impact on the final result. That is why it might be a good idea to calculate the mean roll angle for each license plate, instead of each frame it was detected, and then use that value to update the final roll angle result. This will yield a system with less sample updates but on the other hand a system that makes all license plates equally important.

Another idea is to remake the license plate method to detect horizontal lines on cars instead. For example the ceiling, the rear window and the bumper of the car generates lines that are horizontal in the image. Trying to detect these might be easier then detecting license plates and may make the system more robust.

A

Camera Extrinsic Error

In Appendix A.1 some calculations of the impact that can be caused by the camera extrinsic rotation error are presented. Appendix A.2 describes the theoretical limits of a camera extrinsic calibration.

The appendix is inspired by Tie-Qi Chen, who did some research of the impact of the roll angle for Autoliv Electronics America.

A.1 Impact of Camera Extrinsic Error on Target Position Estimation

A target on the ground is detected at (x_i, y_i) in the image coordinate system. What is wanted is the error in the estimated position of the target in the world coordinate system that is caused by the camera rotation.

Any point in the world coordinate system can be transformed in to the camera coordinate system as Equation (A.1). With $R_{3\times3}$ being the extrinsic camera rotation and $T_{3\times1} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ the camera translation.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{A.1}$$

The camera coordinate system can then be mapped to the image coordinate sys-

tem using Equation (A.2).

$$x_c \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} c_x & f_x & 0 & 0 \\ c_y & 0 & f_y & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{A.2}$$

By combinding Equation (A.1) and (A.2) above an expression of the mapping of any point in the world coordinate system to the image coordinate system can be found, as seen in Equation (A.3).

$$\begin{aligned} x_c \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} c_x & f_x & 0 & 0 \\ c_y & 0 & f_y & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_0 \\ 1 \end{bmatrix} = \\ \begin{bmatrix} c_x & f_x & 0 \\ c_y & 0 & f_y \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13}z_0 + t_x \\ r_{21} & r_{22} & r_{23}z_0 + t_y \\ r_{31} & r_{32} & r_{33}z_0 + t_z \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} \end{aligned} \tag{A.3}$$

The origin of the world coordinate system is located on the ground beneath the camera which means that, $z_0 = t_x = t_y = 0$. Equation (A.3) can be rewritten as the mapping from a point in the image coordinate system to the world coordinate system as Equation (A.4).

$$\begin{cases} x_w(\omega, \theta, \phi) = \frac{t_z \cdot (r_{22} - x_s r_{12})}{(r_{21} - x_s r_{11}) \cdot (r_{32} - y_s r_{12}) - (r_{22} - x_s r_{12}) \cdot (r_{31} - y_s r_{11})} \\ y_w(\omega, \theta, \phi) = \frac{-t_z \cdot (r_{21} - x_s r_{11})}{(r_{21} - x_s r_{11}) \cdot (r_{32} - y_s r_{12}) - (r_{22} - x_s r_{12}) \cdot (r_{31} - y_s r_{11})} \end{cases} \tag{A.4}$$

where

$$\begin{cases} x_s = \frac{y_c}{x_c} = \frac{x_i - c_x}{f_x} \\ y_s = \frac{z_c}{x_c} = \frac{y_i - c_y}{f_y} \end{cases} \tag{A.5}$$

For a camera with a FOV less than $90\,^\circ$, $| x_s | < 1$ and $| y_s | < 1$ always holds. Since the angles ω, θ and ϕ all are small, Equation (A.4) can be simplified as Equation (A.6).

$$\begin{cases} x_w(\omega, \theta, \phi) \approx t_z \cdot \frac{1 - x_s \phi}{y_s + x_s \omega - \theta} \\ y_w(\omega, \theta, \phi) \approx t_z \cdot \frac{x_s + \phi}{y_s + x_s \omega - \theta} \end{cases} \tag{A.6}$$

For all points on the ground, i.e. all the points below the horizon line in the image, $y_s + x_s \omega - \phi > 0$ holds. o_x is the x-component in the camera optical center. The Jacobian matrix of Equation (A.6) becomes as seen in Equation (A.7).

$$\mathbf{J}_w = \begin{bmatrix} \frac{\partial x_w}{\partial \omega} & \frac{\partial x_w}{\partial \theta} & \frac{\partial x_w}{\partial \phi} \\ \frac{\partial y_w}{\partial \omega} & \frac{\partial y_w}{\partial \theta} & \frac{\partial y_w}{\partial \phi} \end{bmatrix} \approx \frac{t_z}{y_s^2} \cdot \begin{bmatrix} -x_s & 1 & -x_s y_s \\ -x_s^2 & x_s & y_s \end{bmatrix} \approx \frac{1}{t_z} \cdot \begin{bmatrix} -x_w y_w & x_w^2 & -t_z y_w \\ -y_w^2 & x_w y_w & t_z x_w \end{bmatrix} \tag{A.7}$$

Assume a camera that has 40 ° of FOV and the height $t_z = 1.47$ m above the ground. The errors in the target postion estimation caused by 1 ° of camera rotational error can be found in Table A.1 and Table A.2.

Camera	**Target at center of FOV**		**Target at edge of FOV**	
Rotation	Longitudinal Error	Lateral Error	Longitudinal Error	Lateral Error
Roll	0 m	0 m	0.86 m	0.63 m
Pitch	1.19 m	0 m	1.19 m	0.86 m
Yaw	0 m	0.17 m	0.13 m	0.17 m

Table A.1: *Errors of target position estimation caused by* 1 ° *of camera rotation error. The target have the longitudinal distance of 10 m.*

Camera	**Target at center of FOV**		**Target at edge of FOV**	
Rotation	Longitudinal Error	Lateral Error	Longitudinal Error	Lateral Error
Roll	0 m	0 m	7.78 m	5.66 m
Pitch	10.69 m	0 m	10.69 m	7.78 m
Yaw	0 m	0.52 m	0.38 m	0.52 m

Table A.2: *Errors of target position estimation caused by* 1 ° *of camera rotation error. The target have the longitudinal distance of 30 m.*

Camera Rotation	**Ego Lane**	**Neighboring Lane**
Roll	0.08 m	0.41 m
Pitch	0.26 m	0.53 m
Yaw	0.21 m	0.21 m

Table A.3: *Errors in lane width estimation caused by* 1° *of camera rotation error. Both lanes are assumed to have US standard lane width of 3.7 m and the shortest visible longintudinal distance is 6 m.*

As seen in the tables above a small camera rotation can cause large errors in the position estimation of the target, especially on off-center targets. Therefore, dynamic camera calibration is very important.

A.2 Theoretical Limits of Vision-Based Camera Extrinsic Calibration

Consider the same system as in Appendix A.1. From Equation (A.4)

$$\begin{cases} x_s = \frac{r_{21}+r_{22}tan(\theta_w)}{r_{11}+r_{12}tan(\theta_w)} \\ y_s = \frac{t_z/x_w+r_{31}+r_{32}tan(\theta_w)}{r_{11}+r_{12}tan(\theta_w)} \end{cases} \tag{A.8}$$

Where $\alpha_w = arctan\left(\frac{y_w}{x_w}\right)$ is azimuth angle of the target. All extrinsic angles, ω, θ and ϕ are small which leads to that Equation (A.8) can be simplified.

$$\begin{cases} x_s \approx tan(\alpha_w) - \frac{\phi}{[cos(\alpha_w)]^2} \\ y_s \approx \frac{t_z}{x_w} - \omega \cdot tan(\alpha_w) + \theta - \phi \cdot \frac{t_z}{x_w} \cdot tan(\alpha_w) \end{cases} \tag{A.9}$$

The Jacobian matrix of Equation (A.9) can now be calculated.

$$J_s = \begin{bmatrix} \frac{\partial x_s}{\partial \omega} & \frac{\partial x_s}{\partial \theta} & \frac{\partial x_s}{\partial \phi} \\ \frac{\partial y_s}{\partial \omega} & \frac{\partial y_s}{\partial \theta} & \frac{\partial y_s}{\partial \phi} \end{bmatrix} \approx \begin{bmatrix} 0 & 0 & -\frac{1}{[cos(\alpha_w)]^2} \\ -tan(\alpha_w) & 1 & -\frac{t_z}{x_w} \cdot tan(\alpha_w) \end{bmatrix} \tag{A.10}$$

The camera roll must cause the image features to move at least 0.5 pixels vertically in order to to estimate the roll angle. This gives Equation (A.11).

$$\Delta\omega \approx \frac{1}{|tan(\alpha_w)|} \cdot \Delta y_s = \frac{1}{f_y \cdot |tan(\alpha_w)|} \cdot \Delta y_i \geq \frac{0.5}{f_y \cdot |tan(\alpha_w)|} \tag{A.11}$$

A camera with 40 ° of FOV at approximately $t_z = 1.47$ m and with $f_x \approx f_y \approx 1000$ pixels is assumed. The nearest point on the ground that the camera can see has at the longintudinal distance 6 m. The car is driving in the center of a lane with a width of 3.7 m and the largest azimuth angle of the lane markers is about 17°.

Using the assumptions above, in theory, the minimum detectable camera roll angle is 0.094°.

B

Datasets

This Appendix contains images of all datasets that have been used in order to evaluate the performance of the system. They illustrate and describes the environment that is referred to in Chapter 5.

B.1 Dataset 1

Figure B.1: *The dataset starts and concludes with non-city driving.*

Figure B.2: *The middle and most of the set consists of a city environment, like in this frame.*

B.2 Dataset 2

Figure B.3: *The set starts in the middle of a city, with a lot of buildings, and then starts to drive outside the city.*

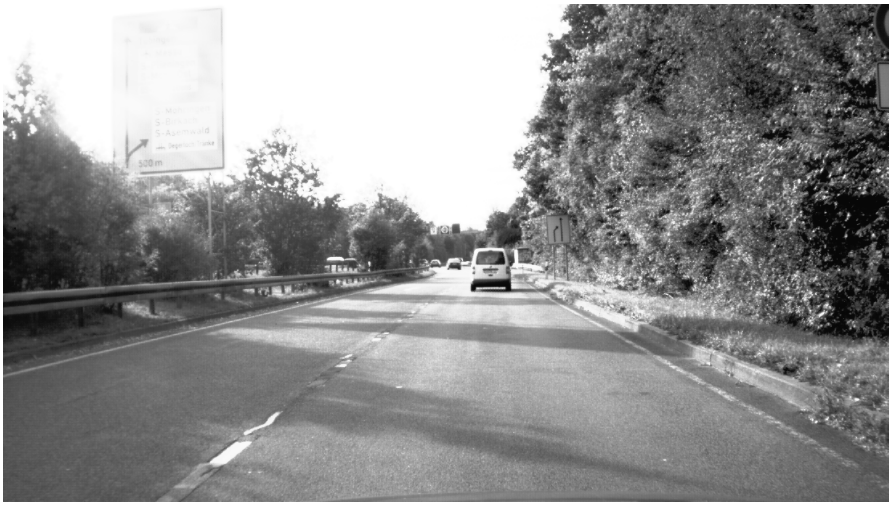

Figure B.4: *After a while it ends up on a large rural road.*

B.3 Dataset 3

Figure B.5: *The set consists of driving in a city. However unlike the other sets the car mostly drives on major city roads with fewer buildings located right next to the side of the road.*

Figure B.6: *This is the scenery for almost the entire dataset.*

Bibliography

Á Catalá-Prat, J Rataj, and R Reulke. Self-calibration system for the orientation of a vehicle camera. *ISPRS Image Engineering and Vision Metrology*, 36(5): 68–73, 2006. Cited on page 3.

Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson Education, third edition, 2008. Cited on pages 7, 8, and 19.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. Cited on page 6.

Wolfgang Hugemann. Correcting lens distortion in digital photographs. *EVU*, 2010. Cited on page 14.

Claudio Rosito Jung and Rodrigo Schramm. Rectangle detection based on a windowed hough transform. In *Proceedins of the XVII Brasilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'04)*, pages 113–120, 2004. Cited on page 20.

Arthur M. Lesk. On the calculation of euler angles from a rotation matrix. *International Journal of Mathematical Education in Science and Technology*, 17(3): 335–337, 1986. Cited on page 7.

Fengjun Lv, Tao Zhao, and Ram Nevatia. Self-calibration of a camera from a video of a walking human. *Proc. of International Conference on Pattern Recognition*, 2002. Cited on page 3.