Kamerabasierte Fahrbahnerkennung zur automatisierten Fahrbahnführung eines Modellauto

Bahri Enis Demirtel

Master Thesis – 02. November 2017

Betreuer: Dr. -Ing. Eric Lenz

TECHNISCHE UNIVERSITÄT DARMSTADT

REGELUNGSTECHNIK UND MECHATRONIK rtm

Aufgabenstellung

Für schriftliche Arbeiten (Pro-/Projektseminar, Studien-, Bachelor-, Master-, Diplomarbeiten, etc.) soll Studierenden ein LaTeX-Dokument zur Verfügung gestellt werden, das die Vorgaben aus den *Richtlinien zur Anfertigung von Studien- und Diplomarbeiten* [**?**] umsetzt. Die Dokumentation soll die Funktionen des Dokumentes beschreiben und Hinweise zu ihrer Anwendung geben.

Grundlage ist die `tudreport`-Klasse. Die damit erstellten Arbeiten müssen sowohl zum Ausdrucken geeignet sein als auch für die Bildschirmdarstellung und die elektronische Archivierung als PDF-Datei.

Beginn:	02. May 2017
Ende:	02. November 2017
Seminar:	15. November 2017

Prof. Dr.-Ing. Ulrich Konigorski

Dr. -Ing. Eric Lenz

Technische Universität Darmstadt
Institut für Automatisierungstechnik und Mechatronik
Fachgebiet Regelungstechnik und Mechatronik
Prof. Dr.-Ing. Ulrich Konigorski

Landgraf-Georg-Straße 4
64283 Darmstadt
Telefon 06151/16-25200
www.rtm.tu-darmstadt.de

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 02. November 2017

Bahri Enis Demirtel

Kurzfassung

Das LaTeX-Dokument `sada_tudreport` ist eine Vorlage für schriftliche Arbeiten (Proseminar-, Projektseminar-, Studien-, Bachelor-, Master- und Diplomarbeiten, etc.) am Institut für Automatisierungstechnik der TU Darmstadt. Das Layout ist an die *Richtlinien zur Anfertigung von Studien- und Diplomarbeiten* [**?**] angepasst und durch Modifikation der Klasse `tudreport` realisiert, so dass in der Arbeit die gewohnten LaTeX-Befehle benutzt werden können. Die vorliegende Anleitung beschreibt die Klasse und gibt grundlegende Hinweise zum Verfassen wissenschaftlicher Arbeiten. Sie ist außerdem ein Beispiel für den Aufbau einer Studien-, Bachelor-, Master- bzw. Diplomarbeit.

Schlüsselwörter: Studienarbeit, Bachelorarbeit, Masterarbeit, Diplomarbeit, Vorlage, LaTeX-Klasse

Abstract

The LaTeX document `sada_tudreport` provides a template for student's research reports and diploma theses (" Proseminar-, Projektseminar-, Studien-, Bachelor-, Master- und Diplomarbeiten") at the Institute of Automatic Control, Technische Universität Darmstadt. The layout is adapted to the *"Richtlinien zur Anfertigung von Studien- und Diplomarbeiten"* [**?**] and is implemented by modification of the standard `tudreport` class, so that common LaTeX commands can be used in the text. This manual describes the class and dwells on general considerations on how to write scientific reports. Additionally, it is an example for the structure of a thesis.

Keywords: Research reports, diploma theses, template, LaTeX class

Inhaltsverzeichnis

Symbole und Abkürzungen vii

1. Introduction 1

1.1. Carolo-Cup . . . 2

1.2. Problem Statement and Objective Target . . . 2

1.3. Structure of Thesis . . . 2

2. Fundamentals 5

2.1. Properties of Truck at Carola-Cup . . . 5

2.2. Inverse Perspective Mapping . . . 6

2.3. Edge Detection . . . 9

2.3.1. Sobel Operator . . . 10

2.3.2. Canny Edge Detector . . . 13

2.4. Hough-Transformation . . . 15

2.4.1. Standard Hough Line Transformation . . . 15

2.4.2. Probabilistic Hough-Transformation . . . 17

2.5. K-Nearest Neighbors Algorithm . . . 18

2.6. Curve Fitting . . . 19

3. Implementation 23

3.1. Test Track . . . 23

3.2. Hardware . . . 24

3.2.1. Model Auto . . . 24

3.2.2. Microcontroller and Main Board . . . 25

3.2.3. Camera . . . 25

3.3. Response Time of the System . . . 27

3.4. Software . . . 27

3.4.1. Development Environment and Related Softwares . . . 27

3.4.2. Preprocessing . . . 28

3.4.3. Method 1 : Hough Transformation + Rectangle Method + Curve Fitting + IPM (v0.9.2) . . . 30

3.4.4. Method 2 : IPM + Hough Transformation + KNN + Curve Fitting(v0.4.2) 36

3.4.5. Method 3 : IPM + Hough Transformation + Rectangle Method + Curve Fitting(v0.5.1) . . . 39

3.4.6. Method 4 : Resize + IPM + Hough Transformation + KNN + Curve Fitting(v0.7.1) 40

4. Evaulation and Discussion **41**

5. Related Works **43**

6. Conclusion **45**

Literaturverzeichnis **47**

A. Appendix **49**

A.1. Default Values of Parameters 49

Symbole und Abkürzungen

Lateinische Symbole und Formelzeichen

Symbol	Beschreibung	Einheit
I	Strom	A
R	Widerstand	Ω
U	Spannung	V

Griechische Symbole und Formelzeichen

Symbol	Beschreibung	Einheit
$\boldsymbol{\Psi}$	Datenmatrix	
σ	Standardabweichung	
ω	Kreisfrequenz	s^{-1}

Abkürzungen

Kürzel	vollständige Bezeichnung
Dgl.	Differentialgleichung
LS	Kleinste Quadrate (*Least Squares*)
PRBS	Pseudo-Rausch-Binär-Signal (*Pseudo Random Binary Signal*)
ZVF	Zustandsvariablenfilter

1 Introduction

Automobiles have been an essential part of modern life for the better part of a century and have, for just as long, been a source of injury and death due to accidents. According to data from Federal Statistical Office in Wiesbaden, in 2016, roughly 2.6 million road traffic accidents occured in Germany and because of these accidents, 3,206 people died. When compared with 2015, although the number of deaths due to traffic accidents decreased by 7.3%, the number of road traffic accidents increased by 2.7% in 2016.[1]

The number one cause of traffic accidents is human error. Like all industries, the automotive industry continues to change and develop rapidly. In a couple of years, there will be more autonomous cars on the road and these cars will eventually replace human drivers. Because of this reason, a large portion of road traffic accidents will be eliminated. This is not the only advantage of autonomous cars. Thanks to autonomous cars, people in traffic will experience less stress, and will also have more time for other things. While driving, the people will be able to work, eat, read and even sleep. But it is also not so easy to build such reliable cars. Because of this reason, nowadays, one of the biggest research areas in the automotive industry is autonomous cars. This research area includes many different fields. Some of these fields are: Car-2-Car/Car-2-X communication, lane detection, sign recognition, object detection, path planning, and so on. In this master's thesis, some lane detection methods, their implementations, advantages, and disadvantages will be discussed.

As in all industries, technology in the automotive industry is continuing to develop day by day. For example, the number of sensors, and their corresponding features, is increasing exponentially. One such sensor is the color camera. To begin with, in the automotive industry, cameras were used only to assist drivers in parking and reversing.

Nowadays, however, one of the main functions of color cameras is lane detection, in both autonomous cars and in cars equipped with a lane departure warning system. In this master's thesis, the lanes will be detected and then formulated mathematically.

The results of this master's thesis will be utilized and expanded upon by the students who will participate in the Echtzeitsysteme Projektseminar at the Technical University of Darmstadt. One of the aims of this seminar is to attend the Carolo-Cup organized annually by the Technical University of Braunschweig. Because of that, the width, the curvature, and the changes of the curvature of the track used in this master's thesis are the same as those belonging to the track used in the Carolo-Cup. In a real-life situation, there are of course oftentimes more factors that can hinder lane detection, including shadows cast by trees, buildings, and other structures; sunlight directly entering the lens of the camera and similarly less-than-ideal lighting conditions; dirt and debris on the road surface; and so on.

Therefore, the lanes of the track must be detected in a sufficiently short amount of time and there should be no dead time between lane detection and mathematical formulation. Lane detection must also be sufficiently robust, so that it should not be disrupted by less-than-ideal lighting conditions.

1.1 Carolo-Cup

The Carolo-Cup is a student competition, providing student teams with a platform for the design and implementation of automated RC cars. The main challenge is to implement cutting-edge algorithmic solutions for vehicle control and environment perception, based on a realistic application scenario.

In the annual competition, the students will present their solutions to a jury from academia and industry, while competing with other international teams from different universities.

Each student team is put in charge of developing, producing and demonstrating a cost- and energy-efficient 1:10 concept of an automated vehicle by a fictional OEM.

During the competition several driving tasks have to be executed as fast and precisely as possible.

In addition, the developed concept must be presented and explained.

In 2017 additional challenges have been introduced: The teams must not only stop at intersections and evade obstacles on the road, but also recognize and adhere to traffic signs. This enables more complex situations at intersections and shall provide an even more realistic urban setting.

1.2 Problem Statement and Objective Target

Autonomous driving is a topic currently being actively researched. Research on autonomous driving can be conducted in two fundamental areas: lane detection and lane guidance. With regard to lane detection, there are different scientific techniques that can be utilized, according to the literature, all with their own advantages and disadvantages under different conditions. For example, some techniques are suitable for straight lines, but not for curves. Others are suitable for curves as well but do not function well under certain light conditions. Others still are quite robust and suitable for curves, yet are computationally intensive (resulting in a video feed with significant gaps).

In this master's thesis, my aim is to research and implement the most appropriate and effective method for use in the Carola-Cup.

1.3 Structure of Thesis

In Chapter 2, the fundamentals of lane detection are explained. All methods utilized in this thesis, along with their respective justifications, are also explained in this chapter. Some methods are also compared with regard to their advantages and disadvantages.

In Chapter 3, the steps of implementation are explained. The components can be divided broadly into the properties of the track, the hardware of the model car, and the software libraries and programs to be utilized. In the software section, all cases will be explained in detail.In this chapter, the program flow will also be explained in detail.

In Chapter 4, the results of the methods utilized will be compared. The computing time of all phases in this thesis will be presented and discussed. Also, all parameters utilized and their effects on this thesis will be also presented and discussed. In this chapter, the researcher will attempt to find an answer to the question, 'How can computing time be reduced?'.

In Chapter 5, the state of the art will be discussed. The other possible solutions for lane detection will also be explored here and their advantages and disadvantages will be compared.

In Chapter 6, all results of this master thesis will be presented and possible improvements and/or enhancements will be discussed.

2 Fundamentals

In this chapter, the fundamentals of lane detection will be explained. All methods, which are used in this thesis, will be theoretically focused and the use of their functions in software will be also explained. Also, some advantages and disadvantages of methods will be discussed.

2.1 Properties of Truck at Carola-Cup

The Carolo-Cup is an annual competition at the Technical University of Braunschweig which are attended by students. Every year the truck and some properties of the competition are changing. For example, in the competitions until 2017 there was no traffic sign, but starting in 2017 there are also some traffic signs, speed limit zones, blocked areas and crosswalks for pedestrian. Because of this, in the competitions until 2017, there was only one way to understand who had the right of way. If there is a stop line on the road in front of an intersection, it means the car has to wait until the intersection is free. In the competitions starting from 2017, the intersections are in different parts: They are 'Intersections with stop lines', 'Intersections with give-way lines', 'Intersections with priority to right', 'Enforced crossing direction - give-way condition', 'Enforced crossing direction - right of way condition'. Except 'Intersections with priority to right', they all have traffic signs stating who has priority. If there is a no traffic sign, it means the right side always has priority.[2]

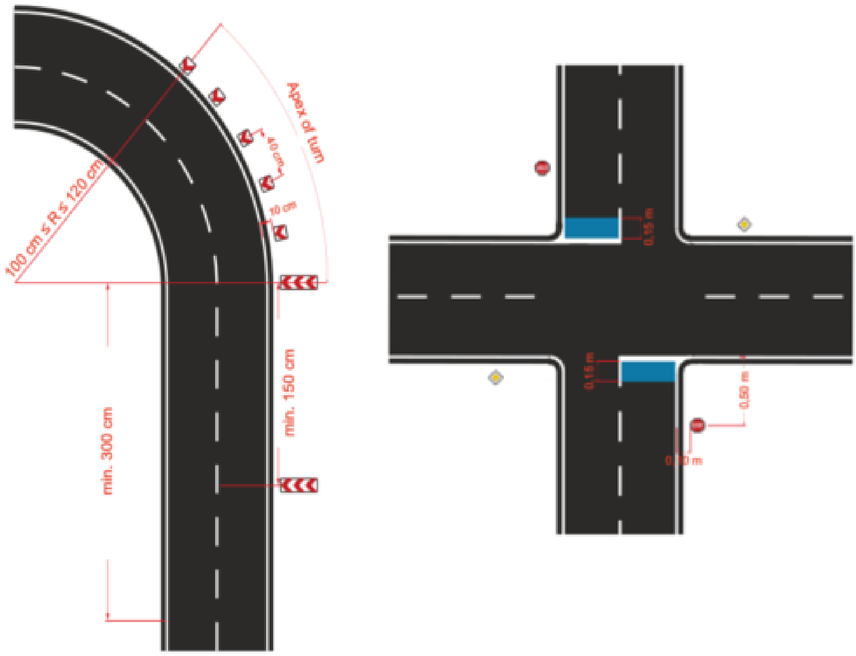


Abbildung 2.1.: Left: Markings for sharp turns at Carolo-Cup. Right: Intersections with stop lines at Carolo-Cup[2]

2.2 Inverse Perspective Mapping

Inverse Perspective Mapping(IPM) is an algorithm which is able to obtain accurate bird's-eye view images from the sequential of forward looking cameras. With the IPM algorithm, each image pixel is remapped, and a new array of pixels is created where the lines in perspective are transformed into straight lines and objects are distorted. IPM is one of the most used methods in lane detection. In lane detection, IPM ensures that the lanes are shown vertical and parallel to each other. On the other hand, because of the re-mapping of pixels, IPM is a computationally expensive method. Because of this reason, in some cases in this master's thesis, rather than remapping all pixels of the images, only the pixels relevant to the lane and accordingly, the fitted curve, were remapped. Thanks to this, in some cases, a lot of computing time was saved.

In order to use the IPM method, the intrinsic and extrinsic parameters of camera are necessary to process images for coordinate transformation and calibration.

- **Intrinsic Parameters :** Intrinsic parameters are camera-specific. It includes information of the focal length (f_x, f_y) and optical centers (c_x, c_y). It is also called a camera matrix. Although the intrinsic parameters are camera-specific, once the camera is calibrated, the modified intrinsic parameters can be stored for future purposes. It is expressed as a 3x3 matrix:

$$\text{camera matrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

In order to find the parameters of the camera matrix, the camera must be calibrated. In order to do this, there is a node in **R**obotic **O**peratic **S**ystem(ROS) which is programmed in the Python programming language. In the tutorial for camera calibration**??** an 8x6 checkerboard with 108mm squares is used. The following command must be used for the calibrating the camera.

rosrun camera_calibration cameracalibrator.py –size 8x6 –square 0.108 image:=/usb_cam/image_raw camera:=/usb_cam

As seen in the above command, the number and the size of the checkerboard's squares must be written in the command and then the camera starts automatically after the command is run. This opens up the calibration window which then highlights the checkerboard. The checkerboard must be moved around in front of the camera. During this process, the camera takes some measurements from the checkerboard. When enough data is collected, the *CALIBRATE* button is highlighted. After this button is clicked, the camera matrix is shown and simultaneously saved as the instrinsic parameters in the camera.

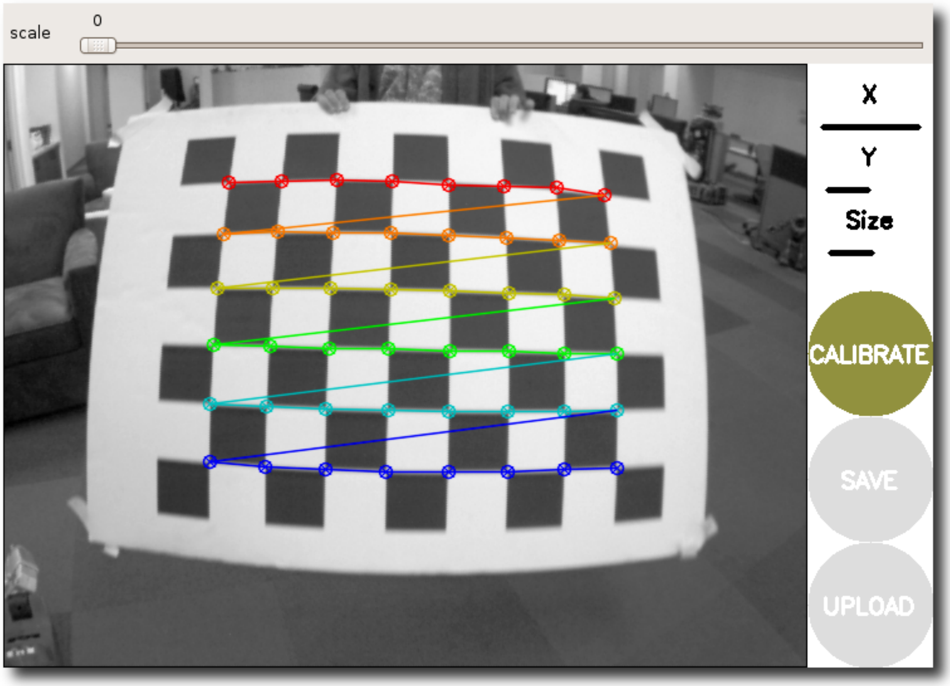


Abbildung 2.2.: Calibration of Camera??

For this master's thesis, the camera was calibrated and the parameters of the camera matrix can be found in the Table **??**

- **Extrinsic Parameters :** Extrinsic parameters are dependent on the camera position. The parameters are H and θ. H is the distance between the camera and ground. θ is the camera tilt angle. These values must be measured because they have to be used for Inverse Perspective Mapping. The extrinsic parameters of the camera can be found in the Table **??**.

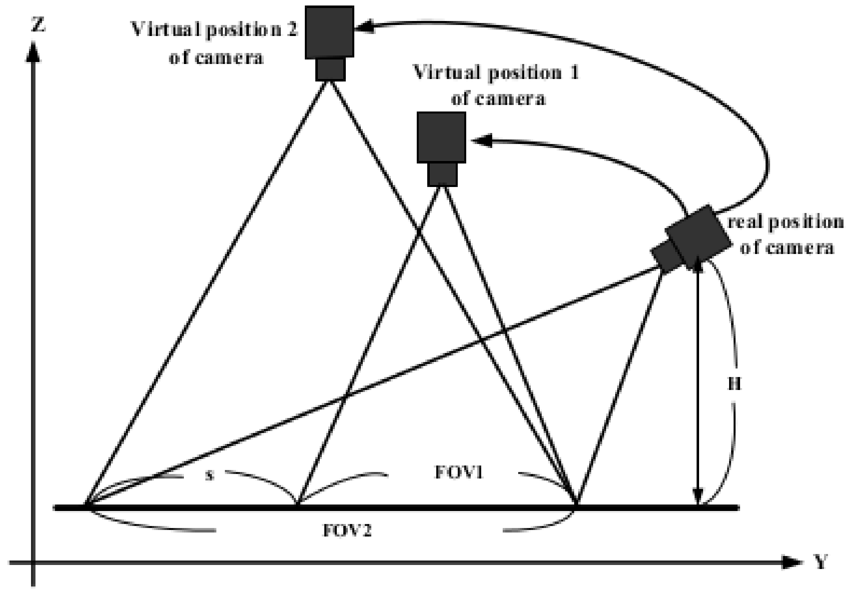


Abbildung 2.3.: Related Positions of the Camera [3]

As seen at Figure **??**, the camera on the car has field of view 2(FOV2) at the real position of the camera but in this case, the view is not a bird's-eye view, so if the same FOV is to be observed from a bird's-eye view, IPM will virtually change the position to Virtual Position 2 of the camera. In this case, although the camera is at its real position, it will appear as though it is at Virtual Position 2. For that, the image coordinates must also be changed. Below, the steps of IPM calculations from the paper of [3] will be detailed.

In the formula, the original image coordinates will be defined as (x,y), the destination image coordinates will be defined as (x^*,y^*), the distance between the ground and the camera will be defined as H, the focal length of camera will be defined as f, and the tilt angle of camera will be defined as θ.

$$x^* = H\frac{xsin\theta + fcos\theta}{-ycos\theta + fsin\theta} \; ; \; y^* = H\frac{ysin\theta + fcos\theta}{-ycos\theta + fsin\theta}$$

In this equation, the transformed component values of x^* and y^* may be less than or equal to zero. Because of this reason, a constant d is defined as $\left|H(sin\theta + cos\theta)/(fsin\theta - cos\theta)\right| + 1$. This means that the coordinate point in the original source image has been mapped into the point of the destination image coordinate system. Below there is the proposed equation :

$$x^* = H\frac{x sin\theta + f cos\theta}{-y cos\theta + f sin\theta} + \text{d}\ ,\ y^* = H\frac{y sin\theta + f cos\theta}{-y cos\theta + f sin\theta} + \text{d}\ ,\ \text{where d} = \left|\frac{H(sin\theta + f cos\theta)}{f sin\theta - cos\theta}\right| + 1$$

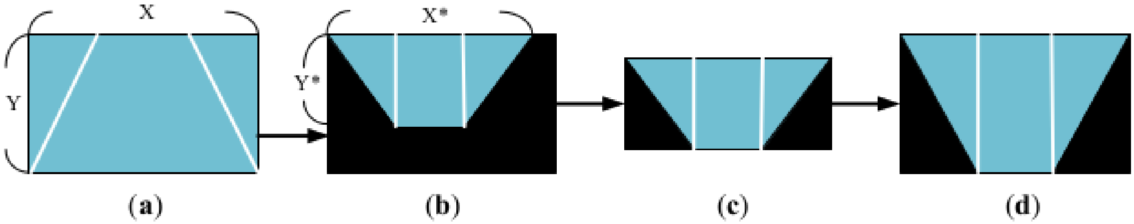


Abbildung 2.4.: Procedures of IPM

2.3 Edge Detection

Edge detectors are essential parts of most computer vision systems. Edge detectors dramatically decrease the amount of data to be processed and extract the useful parts of images. They work by detecting discontinuities in brightness. In this project, the edge detector was used in order to detect the lanes and to exclude unnecessary information from images. There are different methods for edge detection, but they can be grouped into two categories. They are :

- **Gradient method :** This method searches for the maximum and minimum in the first derivative of the image and with that, the edges can be found. For this method, the first order derivative filter must be used. For example : Sobel-Operator.
- **Laplacian method :** This method searches for the zero crossing in the second derivative of the image and with that, the edges can be found. For this method, the second order derivative filter must be used. For example : Laplacian Filter.

According to [4],here are three steps of the edge detection algorithm. They are :

- **Filtering :** For edge detection, it is required to use a suitable smoothing filter. The filters sharpen the edges and ignore the unnecessary information. It is often utilized to improve the functioning of an edge detector against noise. The more filtering is applied, however, the greater the loss of edge strength.
- **Enhancement :** To be able to better detect edges, changes in the intensity in the area surrounding a point must be determined. Pixels in which a significant change in intensity occurs are emphasized by enhancement, which is usually applied by calculating the gradient magnitude.
- **Detection :** Though many points in an image have a nonzero value for the gradient, not all of these points are actually edges. Because only points with strong edge content are desired, a method must be applied to determine which points are actual edge points. Thresholding is often utilized to do so.

Well known smoothing filters are :

- Sobel-Operator
- Canny Edge Detector
- Laplacian-Filter
- Prewitt-Operator

In this master's thesis, the Sobel Operator was utilized, so it will be described in more detail.

2.3.1 Sobel Operator

The Sobel Operator, sometimes called the Sobel filter is one of the most used edge detectors in image processing and computer vision. The Sobel Operator uses vertical and horizontal masks. These masks used are odd-numbered square matrices and they are generally 3x3 matrices. Approximations of the derivatives for the horizontal changes and for the vertical changes are calculated by the operator by using two 3x3 kernels and convolving them with the original image. If A is defined as the source, if G_x is an image which contains the horizontal derivative approximations at each point, and if G_y is an imagine which contains the vertical derivative approximations at each point, then the calculations are:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ +1 & -2 & -1 \end{bmatrix} * A$$

where * here denotes the 2-dimensional signal processing convolution operation.

Since the Sobel kernels can be decomposed as the products of an averaging and a differentiation kernel, they compute the gradient with smoothing. For example, G_x can be written as

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

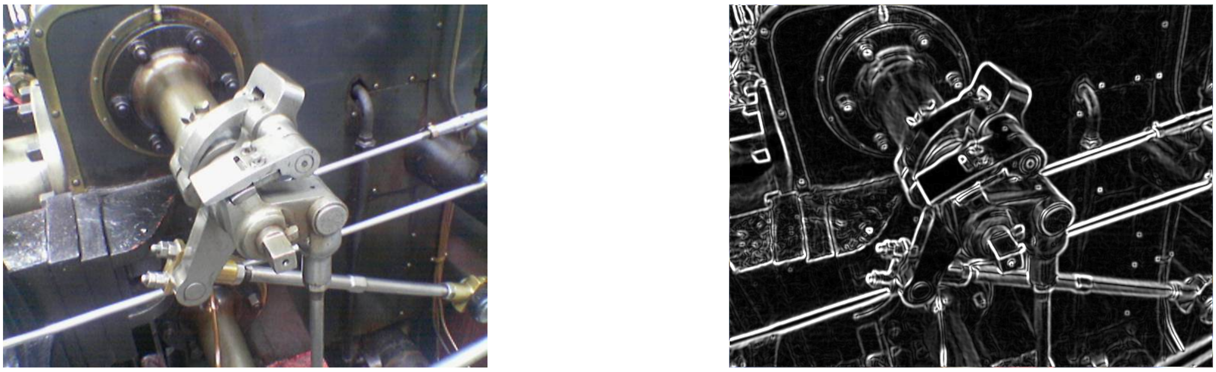
The x-coordinate is defined here as increasing in the 'right'-direction, and the y-coordinate is defined as increasing in the 'down'-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathrm{G} = \sqrt{G_x^2 + G_y^2}$$

Using this information, we can also calculate the gradient's direction:

$$\theta = atan(\frac{G_y}{G_x})$$

where, for example, θ is 0 for a vertical edge which is lighter on the right side.[5]



(a) Original Image **(b)** Sobel Operator applied Image

Abbildung 2.5.: Sobel Operator[5]

In this master's thesis, the Sobel Operator function in OpenCV was used. Before the Sobel operator was used, in order to reduce the noise, the 'GaussianBlur' function was used. As can perhaps be inferred from the name, the 'GaussianBlur' function blurs the image using a Gaussian filter. In order to apply the 'GaussianBlur' function, the following command must be run.

```
void GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT )
```

The parameters of the function will be described in detail.[6]

- **src :** Input image, which can have any number of channels but the depth of which must be one of the following: CV_8U, CV_16U, CV_16S, CV_32F and CV_64F.
- **dst :** Output image, which must be the same size and type as the input image.
- **ksize :** Gaussian kernel size. This size shows the width and height of the Gaussian kernel. These sizes must not be the same value and the values must be odd and positive.
- **sigmaX :** Gaussian kernel standard deviation in the X direction.
- **sigmaY :** Gaussian kernel standard deviation in the Y direction.
- **borderType :** Pixel extrapolation method.

After the Gaussian Filter is applied, the picture must be converted from color to grayscale. For that, there is a small function in OpenCV. With the following command, a color picture can be converted easily to grayscale.

```
void cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0 )
```

The parameters of the function will be described in detail.

- **src :** Input image
- **dst :** Output image which is the same size and depth with input image.
- **code :** Color space conversion code(here used CV_BGR2GRAY).
- **dstCn :** Number of channels in the destination image.

Before the Sobel operator is used, the Gaussian Filter must be applied and the image must be converted to gray scale. After this, the image is ready for Sobel operator to be applied. In order to calculate the 'derivatives' in the x and y directions, the following command must be run twice because gradient X and gradient Y must be calculated separately.

void Sobel(InputArray src, OutputArray dst, int ddepth, int dx, int dy, int ksize=3, double scale=1, double delta=0, int borderType=BORDER_DEFAULT)

The parameters of the function will be described in detail.

- **src :** Input image.
- **dst :** Output image which is in the same size and depth with input image.
- **ddepth :** The depth of the output image.
- **xorder :** Order of the derivative x.
- **yorder :** Order of the derivative y.
- **ksize :** Size of the extended Sobel kernel; it must be 1, 3, 5, or 7.
- **scale,delta and borderType :** Optional values. In this project, the default values were used.

The last step of the application of the Sobel operator is approximating the gradient by adding both directional gradients. In the previous step, the gradients of the x and y coordinates were calculated separately. With following command, the weighted sum of these two gradients must be calculated.[7]

void addWeighted(InputArray src1, double alpha, InputArray src2, double beta, double gamma, OutputArray dst, int dtype=-1)

The parameters of the function will be described in detail.

- **src1 :** First input array.
- **alpha :** Weight of the first array elements.
- **src2 :** Second input array. This array must have the same size and channel number of the first input array.
- **beta :** Weight of the second array elements.

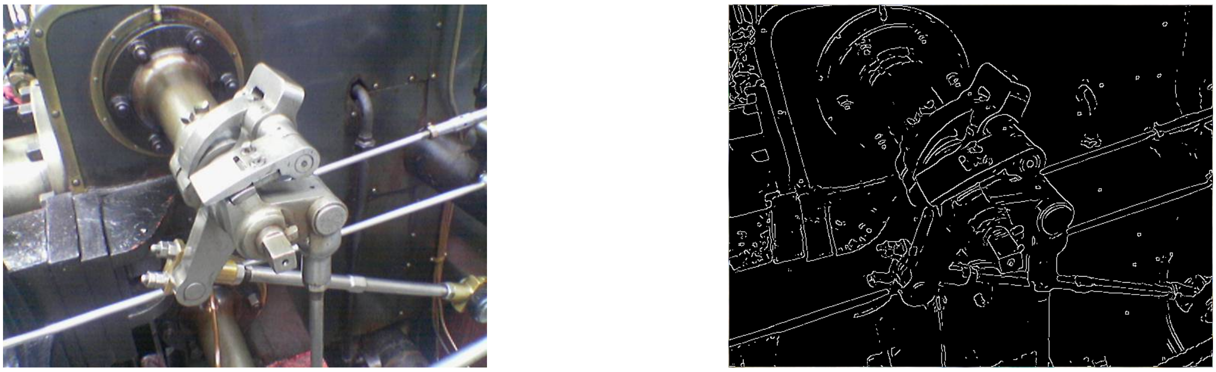
- **dst :** Output array, which has the same size and channel number of the the input arrays.
- **gamma :** Scalar added to each sum.
- **dtype :** Optional depth of the output array.

2.3.2 Canny Edge Detector

Canny edge detector was developed in 1986 and called with the name of its developer John F. Canny. Canny edge detector is also so popular edge detector like Sobel operator. Canny edge detector is a multi-stage algorithm and it can be analyzed in 5 different stages.[8]

1. **Noise Reduction :** To get so stabile lane detection results, we have to reduce/remove all noise from frames. Lane detection without filtering out the noise can cause false detection. Gaussian filter is used for removing noise in the frames. Gaussian filter blurs images and removes detail and noise. The size of Gaussian filter kernel must be (2k+1)x(2k+1). It is important to choose the size of Gaussien filter because if the size of kernel is larger, detector's sensitivity to noise is lower but on the other hand, with the increase in size of the Gaussian filter kernel, the localization error in the edge detection will also increase slightly. [9]
2. **Finding Intensity Gradient of the Image :** Essentially, the Canny algorithm locates edges in image where the grayscale intensity changes most starkly. In order to find these areas, the gradients of the image must be determined. In order to determine the gradients at each pixel in the smoothed image, the Sobel operator is applied. The Sobel operator has already been thoroughly discussed in section 2.3.1.
3. **Non-maximum Suppression :** Non-maximum suppression is an edge thinning technique which is used as an intermediate step in many computer vision algorithms. The image is scanned along the image gradient direction, and pixels that are not part of the local maxima are set to zero. This way, all image information that is not part of the local maxima is effectively suppressed.
4. **Double Thresholding :** The edge pixels remaining after applying non-maximum suppression provide a more accurate depiction of real edges in an image. Despite this, there are still some remaining edge pixels resulting from noise and color variation. Therefore, it is necessary to filter out edge pixels with a weak gradient value while preserving edge pixels with a high gradient value. In order to do this, high and low threshold values must be selected. Edge pixels are marked as strong edge pixels when gradient values are higher than the high threshold value. They are marked as weak edge pixels when gradient values are lower than the high threshold value and higher than the low threshold value. They are suppressed when their values are lower than the low threshold value. The two threshold values are determined empirically and are dependent on the content of a given image.

5. **Hysteresis Thresholding :** Hysteresis Thresholding is the last part of Canny Edge Detector. Until this step, strong edge pixels are extracted from the true edges but there are also some weak edge pixels, some of them are extracted from true edges and some of them are extracted from some noise. So the weak edge pixels which are extracted from true edge, should be strong edge pixels and the weak edge pixels which are extracted from noises must be removed. If there is a weak edge pixel, 8 neighbour pixels of that weak edge pixel is checked and if at least, one pixel of these neighbour pixels is a strong edge pixel then, this weak edge pixels stay as edges in the end picture.



(a) Original Image **(b)** Canny Edge Detector applied Image

Abbildung 2.6.: Canny Edge Detector[9]

The Canny Edge Detector is also a function in OpenCV. As is the case with the Sobel Operator, before the Canny Edge Detector is used, a filter should be applied (for example, the Gaussian Filter) and the image must be converted to gray scale. With the following command, the Canny Edge Detector can be run in OpenCV.[10]

```
void Canny(InputArray image, OutputArray edges, double threshold1, double threshold2, int apertureSize=3, bool L2gradient=false)
```

The parameters of the function will be described in detail.[10]

- **image :** Input image, which has to have an 8-bit single channel.
- **edges :** Output image, which is the same size and type.
- **threshold1 :** First threshold of the hysteresis procedure.
- **threshold2 :** Second threshold of the hysteresis procedure.
- **apertureSize :** Size of the extended Sobel kernel.
- **L2gradient :** A flag for image gradient magnitude.

2.4 Hough-Transformation

In order to isolate features of a particular shape in an image, a method called the Hough Transformation can be utilized. The Hough Transformation which is universally used today was further developed by Richard O. Duda and Peter E. Hart in 1972, although a more rudimentary version had been patented by Paul Hough in 1962.[11] In computer vision, it is often necessary to detect simple edges like straight lines, curves, and ellipses. As a result, this technique is used often in computer vision and image processing. The Hough Transformation is used mostly after an edge detection algorithm. There are actually variations of the Hough Transformation. In this master's thesis, two variations of the Hough Transformation were used. They are:

- **Standard Hough Line Transformation :** This type of Hough Transformation is used mostly for detecting straight lines. How the Standard Hough Transformation works will be explained in detail and as a result, why it is more suitable for detecting straight lines will become more clear.
- **Probabilistic Hough Transformation :** This type of Hough Transformation is suitable for both straight lines and curves, so in this master's thesis, the Probabilistic Hough Transformation was used for detecting lanes. How the Probabilistic Hough Transformation works will also be explained.

2.4.1 Standard Hough Line Transformation

The Standard Hough Line Transformation, which is also called the Linear Hough Transformation, is used mostly for detecting straight lines. There are many different formulas to represent a line segment analytically. A convenient formula for representing a line in an Image(Cartesian) space is :

$$\rho = xcos\theta + ysin\theta$$

In this equation, ρ is the length of a normal from the origin to the line and θ is the orientation of r with respect to the x-axis.

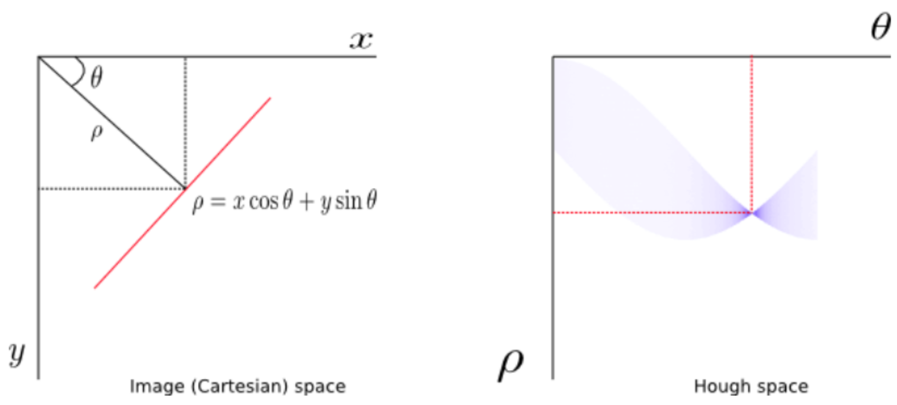


Abbildung 2.7.: Standard Hough Line Transformation[12]

In the left side of Figure **??**, a line is shown in an Image (Cartesian) space and on the right side of same Figure, the same line is shown in its transformed state in the Hough space. A point in the Image (Cartesian) space is transformed into a sinusoidal curve in the Hough space. A line can be defined as a set of points, so a line is transformed into a set of sinusoids intersecting at a point in the Hough space. In this case, detecting points in the Hough space also means detecting the lines in the Cartesian space. After detecting points in the Hough Space, the points must be inversely transformed in order to get the corresponding lines in the Cartesian space.

Although Hough transform (HT) has been widely used in curve detection, it has two major drawbacks: First, for each nonzero pixel in the image, the parameters for the existing curve and redundant ones are both accumulated during the voting procedure. Second, the accumulator array (or Hough space) is predefined in a heuristic way. The more accuracy needed, the higher parameter resolution should be defined. These two needs usually result in a large storage requirement and low speed for real applications. Therefore, RHT was brought up to tackle this problem.

In this master's thesis, in order to detect the Hough lines, the function of *cv2.HoughLines()* in OpenCV was used. The function is:

$$HoughLines(dst, lines, \rho, \theta, threshold, srn, stn)$$

The parameters of the function will be described in detail.[13]

- **dst :** Output of the edge detector. It should be a grayscale image (although in fact it is a binary one).
- **lines :** A vector that will store the parameters (ρ, θ) of the detected lines. At the end of the Standard Hough Line Transformation, only the ρ and θ values will be returned, so only straight lines can be drawn in OpenCV.
- ρ **:** The resolution of the parameter ρ in pixels. In this master's thesis, **1 pixel** was used.
- θ **:** The resolution of the parameter θ in radians. In this master's thesis, **180 degree**(CV_PI) was used.

- **threshold :** The minimum number of intersections to 'detect' a line.
- **srn and stn :** Both of these parameters are for the multi-scale Hough transformation. srn is a divisor for the distance resolution ρ and stn is a divisor for the distance resolution θ. Default values of both parameters are zero.

2.4.2 Probabilistic Hough-Transformation

Probabilistic Hough Transformation is also called Randomized Hough Transformation, is a variant of the Standard Hough Transformation. Hough Transformation is computationally expensive. Probabilistic Hough Transformation is an optimization of Standard Hough Transformation which does not analyze all points in the image. Instead, it looks at only a random subset of points sufficient for line detection. *In order to apply Probabilistic Hough Transformation, the threshold must be decreased. Probabilistic Hough Transformation is commonly used to detect curves (straight line, circle, elipse,etc.)*

The basic idea of Hough transform (HT) is to implement a voting procedure for all potential curves in the image, and at the termination of the algorithm, curves that do exist in the image will have relatively high voting scores. Randomized Hough transform (RHT) is different from HT in that it tries to avoid conducting the computationally expensive voting process for every nonzero pixel in the image by taking advantage of the geometric properties of analytical curves, and thus improve the time efficiency and reduce the storage requirement of the original algorithm.

$$HoughLinesP(dst, lines, \rho, \theta, threshold, minLinLength, maxLineGap)$$

The parameters of the function will be described in detail.**??**

- **dst :** Output of the edge detector. It should be a grayscale image (although in fact it is a binary one).
- **lines :** A vector that will store the parameters $(x_{start}, y_{start}, x_{end}, y_{end})$ of the detected lines.

  At the end of the Probabilistic Hough Transformation, the end points of detected lines will be returned. With the Probabilistic Hough Transformation in OpenCV, straight lines can be detected. In addition, the small straight lines in other shapes can be combined, and with this technique, all shapes can be detected.

  In this master's thesis, all lanes were detected with Probabilistic Hough Transformation. However, instead of Hough Lines, the points which were returned as the result of Probabilistic Hough Transformation were used.
- ρ **:** The resolution of the parameter ρ in pixels. In this master's thesis, **1 pixel** was used.
- θ **:** The resolution of the parameter θ in radians. In this master's thesis, **1 degree**(CV_PI/180) was used.

- **threshold :** The minimum number of intersections to 'detect' a line.
- **minLinLength :** The minimum number of points that can form a line. Lines with less than this number of points are disregarded.
- **maxLineGap :** The maximum gap between two points to be considered in the same line.

2.5 K-Nearest Neighbors Algorithm

K-Nearest Neighbor(KNN) is an non-parametric lazy learning algorithm. The non-parametric technique means that it doesn't make any assumptions on the underlaying data distribution. In the definition of KNN, the term 'lazy learning algorithm' is used. It means it doesn't use the data training points to do any generalization. In other words, there is no explicit training phase or it is very minimal. It also means that the training phase is pretty fast. Most of the lazy algorithms - especially KNN - make decisions based on the entire training data set. On the other hand, KNN is one of the top 10 data mining algorithms[14].

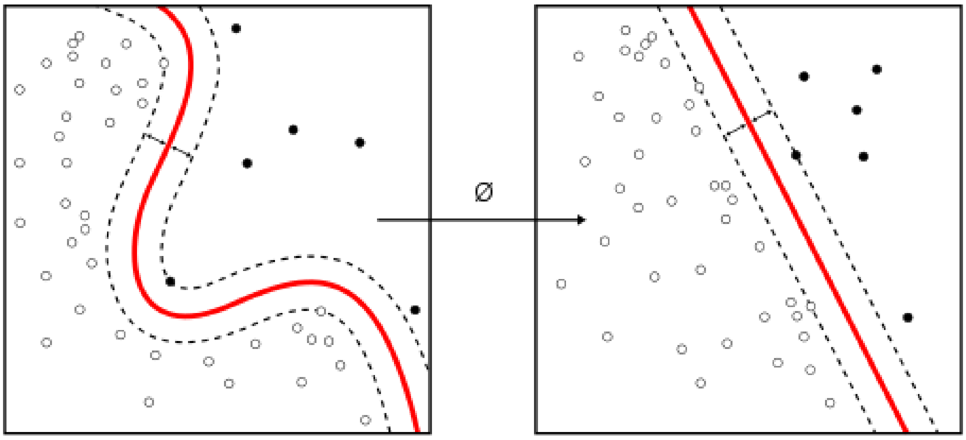


Abbildung 2.8.: K-Nearest-Neighbors Algorithm[15]

By using the training dataset directly, KNN makes predictions. By searching the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances, predictions are made for a new instance (x). For regression this might be the mean output variable, in classification this might be the mode (or most common) class value. Distance measure methods are used for the determining which of the K instances in the training dataset are the nearest to a new input. There are so many different distance measure methods. Some of them are :

- **Euclidean Distance :** It is one of the most popular methods for distance measure. In this method, the square root of the sum of the squared differences between a new point and an existing point is calculated.

$$\textbf{Euclidean} \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

- **Manhattan Distance :** It is called also City Block Distance. It calculates the distance between real vectors using the sum of their absolute difference.

$$\textbf{Manhattan}\ \sum_{i=1}^{k}\left|x_i - y_i\right|$$

- **Minkowski Distance :** The Minkowski distance is a metric in a normed vector space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance.

$$\textbf{Minkowski}\ (\sum_{i=1}^{k}(\left|x_i - y_i\right|)^q)^{1/q}$$

There are also some other distance measure methods like Tanimoto, Jaccard, Mahalanobis, cosine distance and so on. It must be decided which distance measure method should be used according the properties of the dataset. For example, if the input variables are similar in type (e.g. all measured widths and heights), the Euclidean distance measure method is good. However, if the input variables are not similar in type (such as age, gender, height, etc.), the Manhattan distance measure method is better than the Euclidean measure distance method.

The K-Nearest Neighbors Algorithm has advantages and disadvantages. According to [16],the main advantages of KNN are simplicity, effectiveness, intuitiveness and competitive classification performance in many domains. On the other hand, KNN can have poor run-time performance when the training set is large. It is very sensitive to irrelevant or redundant features because all features contribute to the similarity and thus to the classification. The computation cost is also quite high because we need to compute distance of each query instance to all training samples.

K-Nearest Neighbors Algorithm is also a function in OpenCV. The following command shows the usage of the KNN function in OpenCV.

```
void flann::Index_<T>::knnSearch(const vector<T>& query, vector<int>& indices,
        vector<float>& dists, int knn, const SearchParams& params)
```

The parameters of the function will be described in detail.[17]

- **query :** The query point.
- **indices :** The indices of the KNN are found in this vector.
- **dists :** The distances of the KNN are found in this vector.
- **knn :** Number of nearest neighbors to search for.
- **params :** There are some different optional parameters, which can be used in this function.

2.6 Curve Fitting

Curve fitting is used to find the 'best fit' line or curve for a series of data points. Curve fitting produces mostly mathematical equations that can be used to find points anywhere along the curve.[18] They are several different types of curve fitting. Some of them are: linear, exponential, polynomial, exponential, power, logarithmic, etc. In this master thesis, polynomial curve fitting was used. Polynomial curve fitting differs from order of the polynomial. Polynomial curve fittings are called different names depending on their orders. First order polynomial curve fittings are called linear regression, second order as quadratic regression, and third order as cubic regression.

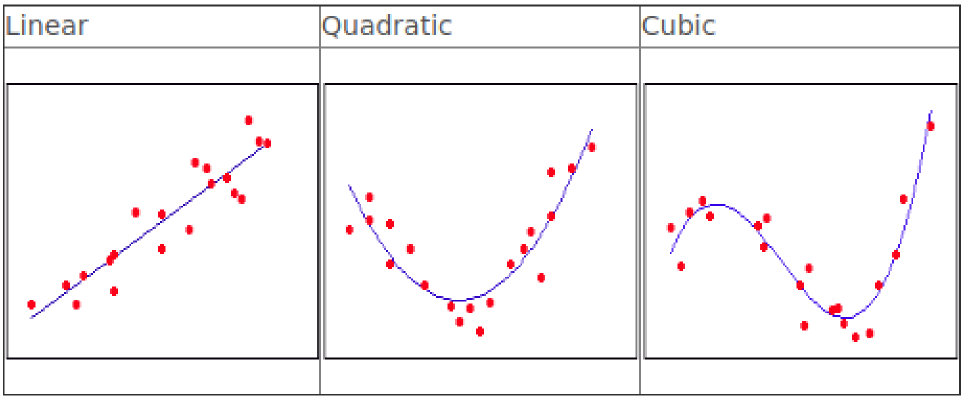


Abbildung 2.9.: Types of Polynomial Curve Fitting[19]

In this master's thesis, curve fitting is used for getting the best mathematical descriptions of lanes. Curve fitting uses the Hough points which appear on the lanes as input, and give a mathematical equation as output. First order polynomial curve fitting is more suitable for defining straight lines and second order polynomial curve fitting is more suitable for defining curves. There are both straight and curved lanes, so first order polynomial curve fitting is not sufficient for our project. Accordingly, in this master's thesis, second order polynimal curve fitting is used.

There are so many different methods for curve fitting. One of the most famous methods is the least squares method, which is also the method utilized in this master's thesis. Next, how a polynomial curve fitting is generated using the least squares method will be described.

A data set can be mostly expressed the relationship between variable with an equation which is mostly representated with a k^{th} order polynomial. The general description of k^{th} is :

$$y = a_k x^k + ... + a_1 x + a_0 + \epsilon$$

The general polynomial regression model with the error ϵ typically provides an estimate rather than an implicit value of the dataset for any given value of x. A data set which has N data points can be described with the maximum order of the polynomial which is k = N - 1, but the lowest possible order of the polynomial is generally chosen.

The aim of the least squares' method is to minimize the variance between dataset values and estimated values from the polynomial equation.

In order to determine the coefficients of the polynomial regression model $(a_k, a_{k-1}, ..., a_1)$ must be solved the following linear equations.

$$\begin{bmatrix} N & \sum_{i=1}^{N} x_i & \dots & \sum_{i=1}^{N} x_i^k \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 & \dots & \sum_{i=1}^{N} x_i^{k+1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N} x_i^k & \sum_{i=1}^{N} x_i^{k+1} & \dots & \sum_{i=1}^{N} x_i^{2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} x_i y_i \\ \vdots \\ \sum_{i=1}^{N} x_i^k y_i \end{bmatrix}$$

The equations in the stardard form of Ma = b can be solved with many different methods. In this master's thesis, *Cramer's Rule* was used to solve for the polynomial coefficients of curve fitting. With the help of the determinants of the square matrix, M can be solved for in any linear system of equations in order to find the coefficients. Each of the coefficients a_k may be determined using the following equation :

$$a_k = \frac{det(M_i)}{det(M)}$$

In order to find the coefficients of polynomial curve fitting, we have to use the equation shown above, so the determinate of M_i must be divided by the determinate of M. Above, the equation Ma = b was shown, so the determinate of the M matrix can be calculated. However, for the determinate of M_i matrix, the M matrix must be modified. To find the M_i matrix, the i^{th} column must be replaced with the column vector b, which was used in the equation Ma = b. For example, if the M_0 matrix is to be determined, the M matrix must be modified like at the following[20] :

$$M = \begin{bmatrix} N & \sum_{i=1}^{N} x_i & \dots & \sum_{i=1}^{N} x_i^k \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 & \dots & \sum_{i=1}^{N} x_i^{k+1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N} x_i^k & \sum_{i=1}^{N} x_i^{k+1} & \dots & \sum_{i=1}^{N} x_i^{2k} \end{bmatrix} \quad M_0 = \begin{bmatrix} \sum_{i=1}^{N} y_i & \sum_{i=1}^{N} x_i & \dots & \sum_{i=1}^{N} x_i^k \\ \sum_{i=1}^{N} x_i y_i & \sum_{i=1}^{N} x_i^2 & \dots & \sum_{i=1}^{N} x_i^{k+1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N} x_i^k y_i & \sum_{i=1}^{N} x_i^{k+1} & \dots & \sum_{i=1}^{N} x_i^{2k} \end{bmatrix}$$

In this master's thesis, the 2^{nd} order polynomial equation is used, so only the values of $a_0, a_1 and a_2$ must be calculated. To understand curve fitting better, we are going to develop the 2^{nd} order polynomial curve fit for the given dataset.

x	-3	-2	-1	-0.2	1	3
y	0.9	0.8	0.4	0.2	0.1	0

$$M = \begin{bmatrix} 6 & -2.2 & 24.04 \\ -2.2 & 24.04 & -8.008 \\ 24.04 & -8.008 & 180.0016 \end{bmatrix} \quad M_0 = \begin{bmatrix} 2.4 & -2.2 & 24.04 \\ -4.64 & 24.04 & -8.008 \\ 11.808 & -8.008 & 180.0016 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} 6 & 2.4 & 24.04 \\ -2.2 & -4.64 & -8.008 \\ 24.04 & 11.808 & 180.0016 \end{bmatrix} \quad M_2 = \begin{bmatrix} 6 & -2.2 & 2.4 \\ -2.2 & 24.04 & -4.64 \\ 24.04 & -8.008 & 11.808 \end{bmatrix}$$

$$a_0 = \frac{det(M_0)}{det(M)} \Longrightarrow a_0 = \frac{2671.1962}{11661.2736} = 0.2291$$

$$a_1 = \frac{det(M_1)}{det(M)} \Longrightarrow a_1 = \frac{-1898.4602}{11661.2736} = -0.1628$$

$$a_2 = \frac{det(M_2)}{det(M)} \Longrightarrow a_2 = \frac{323.7632}{11661.2736} = 0.0278$$

In this case, the fitted curve function is :

$$y = 0.0278x^2 - 0.1628x + 0.2291$$

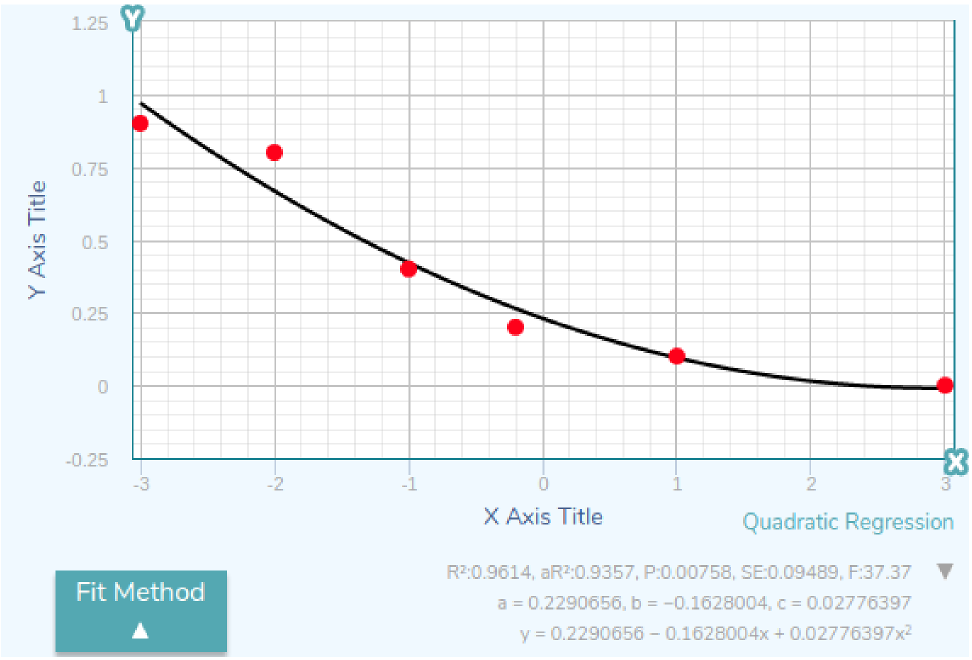


Abbildung 2.10.: Curve Fitting[21]

3 Implementation

In this chapter, the implementation part of this master's thesis will be described. The test track, which was implemented before by another student; the hardware part of the model automobile, which was used in this master's thesis; and finally, the software part which was programmed for detecting the lanes will all be explained in detail.

In the Software section in 3.4, all methods programmed and utilized during this master's thesis will be descriebed. In order to better demonstrate the process, pictures of each step, as well as block diagrams of each method, will be shown.

3.1 Test Track

As previously mentioned, the medium-term goal of this master thesis is attending the Carola-Cup at Braunschweig University, so the test truck was prepared according to the Carola-Cup criteria by Nicolas Acero Sepulveda, who also did his bachelor's thesis with this model automobile. For this test truck, two black PVC floor carpets were used and on these floor carpets, the lanes of the track were made by using white electrical tape. The straight part of the track was made on one of these PVC floor carpets and the curved part of track was made on the second PVC floor carpet. The straight part of the track is approximately 2 meters long and the curve radius of the curved part of test track is approximately 1 meter. This curve is the tightest curve at Carola-Cup, so with this, the test track can be tested in the worst case scenario. In the Carola-Cup competition, the track is much larger; however, for the purposes of this master thesis, a larger test track in not needed. *In Figure 3.1, the test track used in the testing of this master's thesis is shown.*

Abbildung 3.1.: Test Track

3.2 Hardware

3.2.1 Model Auto

During the course of this master's thesis, a model automobile was being used which was prepared for the Projectseminar Echtzeitsysteme at Technical University of Darmstadt. The chassis, steering mechanism, power train, and engine control were derived from the model-building of a Japanese company, Tamiya. The maximum velocity of the model automobile is approximately 1 m/s and the minimum steering radius is around 90 cm.

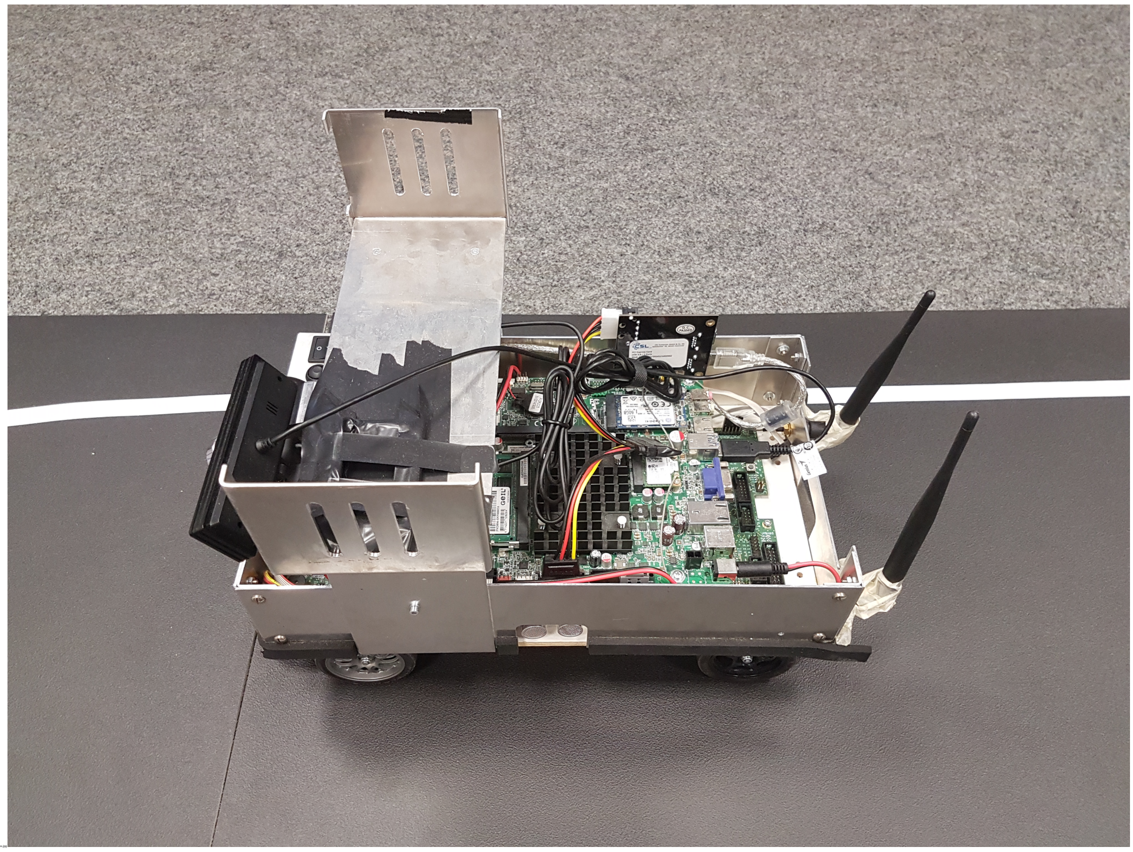


Abbildung 3.2.: Model Auto

3.2.2 Microcontroller and Main Board

In this model automobile, there is a microcontroller and a main board. The microcontroller is used for controlling steering and receiving the measurements from ultrasonic sensors and hall effect sensors. The 16-bit microcontroller is from MB96300 series from Fujitsu company.

The main board on the model car is from PD10BI-MT ThinMini-ITX series from MiTAC company. This main board communicates with the microcontroller over via UART interface through USB connection. On this main board, there is an Intel Quadcore-Processor and an Intel HD Graphics card. Furthermore, there is an 8 GB DDR3-1600 RAM and 1Gbit/s Ethernet, VGA, HDMI, USB 2.0/3.0, SATA ports and an Intel Dual Band Wireless AC 7260 Network adapter, which is connected to two external WLAN antennas. A 60 GB Kingston SSD-Harddisk is connected over an integrated PCI-Express Port. A 3200 mAh Li-Fe battery is used as a power supply.

3.2.3 Camera

The camera is one of the main components of lane detection and accordingly, autonomous driving. For this thesis, I had to research the most suitable camera because all cameras have different properties.

At the beginning of the Projectseminar Echtzeitsysteme, the Logitech C270 HD Webcam was being used. The resolution of the camera is 1280x960 pixels and the Frame per Second (FPS) value is 30 Hertz (Hz) at a 640x480 pixel resolution. The field of View (FOV) is just 60 degrees. The problem with this camera is that if there is a curve, the camera cannot see all of the lanes, and thus is not very suitable for lane detection. When I started my master's thesis, there was a Kinect v2 camera on the model car. The Kinect v2 camera was developed by Microsoft and released in 2013. This camera has a depth sensor with a resolution of 512x424 pixels and its FOV is 70x60 degrees. The FPS value is 30 Hz at a 512x424 pixel resolution. This camera also has a color camera with resolution of 1920x1080 pixels and a FOV of 84.1x53.8 degrees. The FPS value is 30 Hz at a 1920x1080 pixel resolution. This camera had two main disadvantages for this master's thesis. The first disadvantage is the FOV value of camera. This value is better than the value of Logitech C270 camera but it is still not enough for curve lane detection. The second main disadvantage is the location of the color camera. The color camera of this camera is not in the middle of camera, but rather, on the right. This is a disadvantage for us because when there are curves going left as opposed to right, the camera is unable to see the left and even perhaps the middle lane of the truck. Thus, this is problematic for lane detection.

Due to these reasons, I had to choose a camera which has a sufficiently high FOV value. After doing research, I decided that the Genius Widecam F100 camera is the best choice for this master's thesis because this camera has a FOV value of 120 degrees and it can also be used with the Linux Operating System. The resolution of this camera is 1920x1080 pixels and the FOV value is 120 degrees. The FPS is 30 Hz at a 1920x1080 pixel resolution. With this camera, it is possible to detect most if not all lanes, including when there are curves.

Abbildung 3.3.: Genius 120-degree Ultra Wide Angle Full HD Conference Webcam(WideCam F100)

3.3 Response Time of the System

Before lane detection project has started, an experiment was done with the system. Aim of this experiment is to measure the response time of the all system. For measuring it, camera has to detect something and then the software must set an I/O pin on the main board. In other words, if the camera detects something, how long does it take to get a response from the main board.

For this experiment, a camera, a power supply, A LED and an oscilloscope are needed. A circuit with a power supply and a LED was designed. The LED was placed in a small box with a camera, and a probe of an oscilloscope was connected to the LED and the other probe of it was connected to an I/O Pin on the mainboard which can be setted by software.

Before the results were measured,

and the frequency of the power supply was setted to 3,28 Hz. square wave

3.4 Software

In this chapter, the software algorithms defined in this master's thesis will be focused on. With the aid of program flow charts and explanations of all their steps, the algorithms will themselves be better explained. In order to find the best solution, five different source code versions(?variants/?methods) were generated. For all these source codes, the computing times were calculated and compared in terms of which solution can detect the lanes better. In the following pages, there are detailed explanations of the versions utilized (?variants/?methods). The development environment and the software utilized in this master's thesis will be also described.

3.4.1 Development Environment and Related Softwares

As also mentioned at subsection 3.2.2, in this project the previously introduced main board was utilized. One of the compact and fast versions of the Linux 16.04 operating system, *Lubuntu* was installed in this main board.

The version *Kinetic* of ROS was used for implementation of this master's thesis. ROS is the abbreviation of **R**obotic **O**perating **S**ystem, which is a robotics middleware (i.e. collection of software frameworks for robot software development). On the ROS wiki page[22], ROS is defined as an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

For using prepaid image processing functions, an open-source computer vision and machine learning software library called OpenCV was used. According to the OpenCV website[**?**], there are more than 2500 optimized algorithms in the OpenCV library and OpenCV has a user community of more than 47 thousand people.

ROS can be programmed with Python, C++ or Lisp programming languages and OpenCV can be programmed with Python or C++ programming languages. In this master's thesis, C++ was used.

3.4.2 Preprocessing

There are many different possibilities for lane detection algorithms. Of course, each has its own advantages and disadvantages. In this master's thesis, some methods were defined, and in this chapter, these methods will be explained in detail.

In all of these methods, some processes are common, and this is called the preprocessing phase. At the beginning, the frames are obtained from the camera via ROS-Topic. ROS uses different image formats than OpenCV, which uses the image format Matrix(Mat). The frame obtained via ROS-Topic must be converted from the ROS image data type to a Mat object. In order to convert the frame, a ROS-Package *cvbridge*[23] was used. cvbridge converts the ROS image format to a Mat Object, which is the OpenCV Image Format.

Mat is a class which has two parts. The parts are a matrix header and a pointer to the matrix containing the pixel values. The matrix header contains information like the size of matrix, storing method, etc. It always has a fixed size but the size of matrix is variable from image to image.

There are so many methods which can store the pixel values to the Mat object. In this case, the color space and the data type utilized can be chosen. For gray images, it is easy to choose the color space because there are just two colors : black and white. By changing the density of colors(black and white), it is possible to create many shades of gray. There are more methods for color images. Color images generally have three or four channels. These three channels are used for RGB color values. The RGB colors are based on the colors red, green, and blue, which can all be detected by the human eye. For the transparency of a color, a fourth channel called alpha(A) can be used.

There are also another color formats, which have some advantages[24].

- The RGB format is quite similar to the human eye, but the OpenCV display system uses the BGR format, which uses another row of colors.
- The HSV and the HLS formats are more natural ways to display colors. They decompose colors into their hue, saturation, and value/luminance components. Another advantage of the HSV and the HLS formats is that they are less sensitive to the light conditions of the input image.
- In JPEG image formats the YCrCb format is used.

- if the distance of a given color to another color is to be measured, CIE L*a*b* format is more suitable than others.

After the frame from Camera via ROS-Topic was received, the color frame had to be converted to a grayscale frame. For detection lanes, Hough Transformation is used and for Hough Transformation, the grayscale format of the input image is needed. Converting a frame from BGR format to grayscale format has some advantages, the main advantage being the processing time. Normally color frame matrix content has three or four channels, but grayscale frame matrix content has just one channel, so grayscale frame matrix size is much smaller compared to color frame matrix size. Because of this, the image processing time is much more less for the grayscale format compared to BGR format. In order to convert the BGR formatted frame to a grayscale formatted frame, the *cvtColor* function from OpenCV is used.

For stable lane detection, the light conditions must be considered. Because of this, after converting the frame with BGR format to grayscale format, the lightest and darkest pixels have to be searched for. *For the finding the lightest and the darkest pixels, the following function in OpenCV was used. This function will be explained in detail.* [7] (S.

)

void minMaxLoc(InputArray src, double* minVal, double* maxVal=0, Point* minLoc=0, Point* maxLoc=0, InputArray mask=noArray())

- **src :** Input single channel array.
- **minVal :** Pointer to the returned minimum value.
- **maxVal :** Pointer to the returned maximum value.
- **minLoc :** Pointer to the returned minimum location.
- **maxLoc :** Pointer to the returned maximum location.
- **mask :** Optional mask used to select a sub-array.

After finding the lightest and darkest pixels, it is possible to estimate the lighting conditions. These values are then used in the next step. After this processing, a filter is applied to all frames, transforming images into binary images by transforming each pixel according to whether it is inside or outside a specified range. The user chooses a threshold value to process. If a pixel is greater than this value, it is assigned an 'inside' value; otherwise, it is assigned an 'outside' value. Depending on the lightest and darkest pixel values, the threshold value changes. Through this dynamic parameter, the lanes are more able to be more clearly detected and noise can be cancelled more successfully. *For this thresholding operation, the following function in OpenCV was used. This function will be explained in detail.* [**?**] (S.

)

double threshold(InputArray src, OutputArray dst, double thresh, double maxval, int type)

- **src :** Input single channel array, which is 8-bit or 32-bit floating point.
- **dst:** Output array which has the same size and type as src.
- **thresh :** Threshold value.
- **maxval :** Maximum value to use with the THRESH_BINARY and THRESH_BINARY_INV thresholding types.
- **type :** Thresholding type, which can be THRESH_BINARY, THRESH_BINARY_INV, THRESH_TRUNC, THRESH_TOZERO and THRESH_TOZERO_INV.

After the threshold filter, an edge detection filter must be applied. In this master's thesis, the Sobel-Operator is used. This is explained in Chapter 2 in detail.

This preprocessing part is common for all cases but after this process, the cases diverge.

3.4.3 Method 1 : Hough Transformation + Rectangle Method + Curve Fitting + IPM (v0.9.2)

The algorithms and their orders which was used in this method, will be explained step by step. At Figure 3.4, the block diagram of Method 1 is shown.

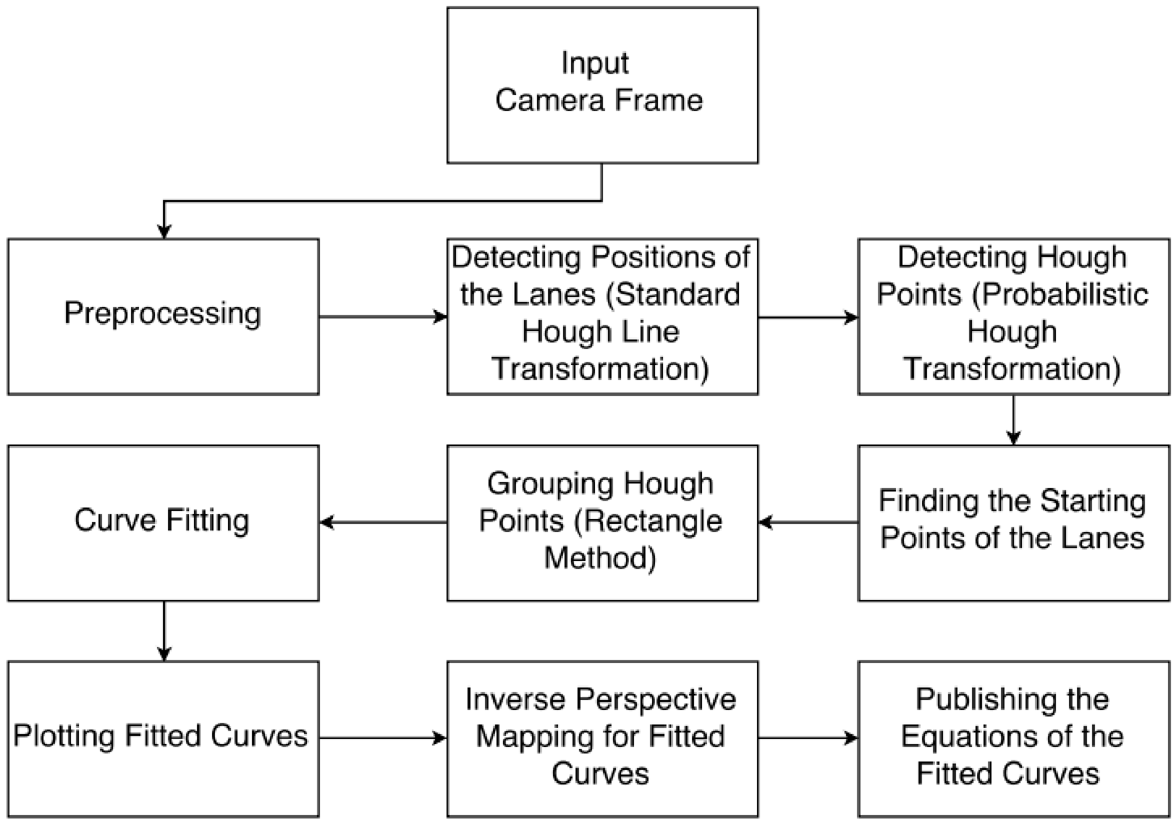


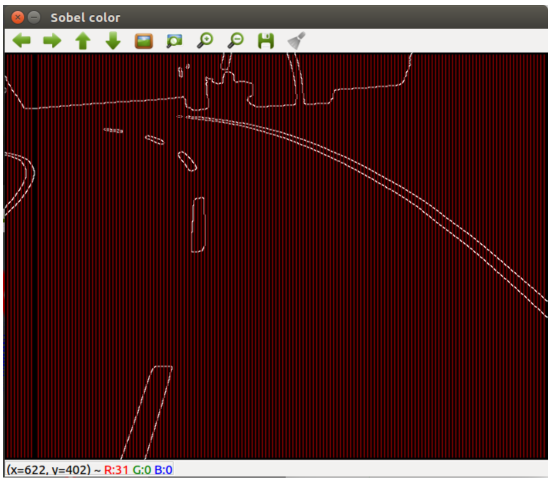
Abbildung 3.4.: Block Diagram of Case 1

Step 1 : As with all methods, the first step is preprocessing part. When the preprocessing part is over, the lanes must be detected.

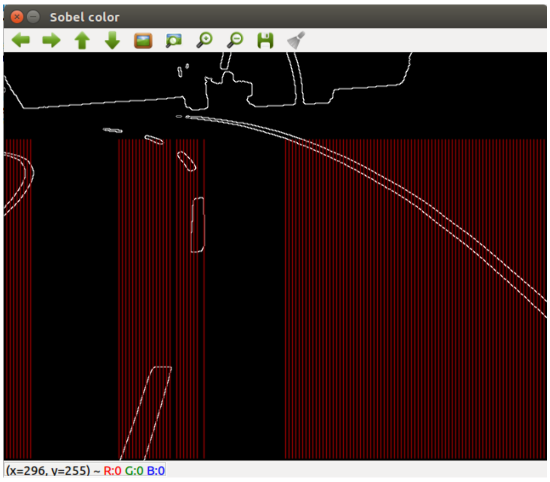
Step 2 : Secondly, the positions of the lanes should be found. In order to find the positions of the lanes, the Standard Hough Transformation is used. But the Standard Hough Transformation should not utilize the entire frame; rather, the frame is cropped. There is an advantage to cropping the frame, which will be explained in detail.

The frames obtained from the camera are at a resolution of 640x480 pixels, which is the default value. The resolution of the camera can be increased by adjusting its settings, but it is not possible to decrease it this way. In order to decrease the resolution of the camera, there is another technique that can be used. This technique is used with another method, and will be explained there. With this method, the minimum resolution the camera allows is used. There are some reasons for this, the most important being that if the resolution is increased, there are also more pixels, and thus the computing time increases as well. High computing time is of course undesirable.

The original height of the frame was 640 pixels, but as previously mentioned, the frame was cropped. As seen in Figure 3.5a, when there is a curve, the camera also detects points that do not belong to the track. When the frame is not cropped, the detection of irrelevant points would result in the undesired production of red Standard Hough Transformation lines. As a result, first 100 pixels of the frame height were removed and the last 540 pixels were used. As seen at Figure 3.5b, the red lines are shown only in the last 540 pixels of frame height and do not cover the irrelevant parts of the frame. If there is a lane, the red lines are very close each other, but if there is no lane, there is a distance of at least 50 pixels between the red lines. In Figure 3.5b, there are three different groups of red lines. In each group, there is a lane. Thanks to the Standard Hough Line Transformation, we know which lanes are between which pixel columns.



(a) Hough Transformation without Cropping Image

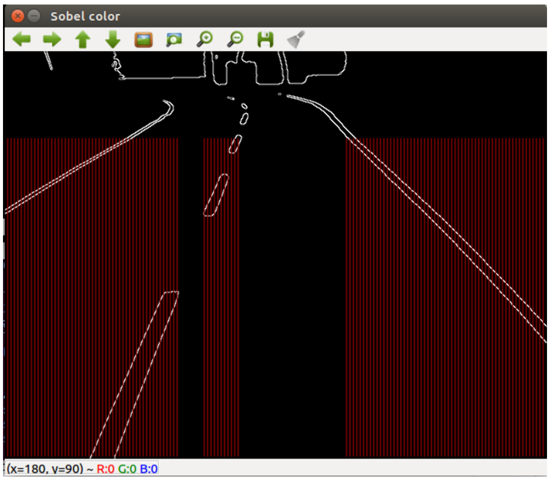


(b) Hough Transformation with Cropping Image at Curve Lane

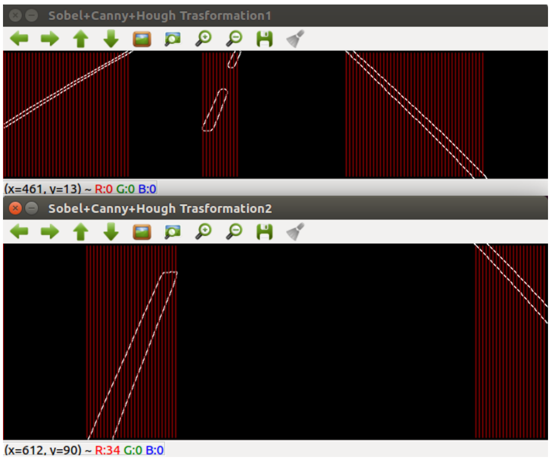
Abbildung 3.5.: Detecting Lane Positions

But here there is also another problem. If a curve is to be detected, everything in this step works as it should (see Figure 3.5b), but if a straight lane is to be detected, a problem occurs. As seen in Figure 3.6a, the beginning of the middle lane can be grouped with the left lane. In other words, there are again three different groups for three different lanes but each group of red lines fails to separate the lanes clearly from each other. As a result, the correct starting points of the middle and left lanes are not able to be found. In order for the lanes to be detected clearly, the frame must be divided into two horizontal rows.

As previously mentioned, 100 pixels were already cropped from the top of the image in order to detect the positions of the lanes clearly and now, the rest of the frame must be divided into two rows. Based on the experimental results, the top row is 150 pixels high and the bottom row is 230 pixels high. As a result, the lanes can be grouped into two small frames and in each small frame, the starting points of lanes must be found. As seen in Figure 3.6b, when the frame is divided into two pieces, the lanes can be grouped separately.



(a) Hough Transformation with Cropping Image at Straight Lane in 1 piece



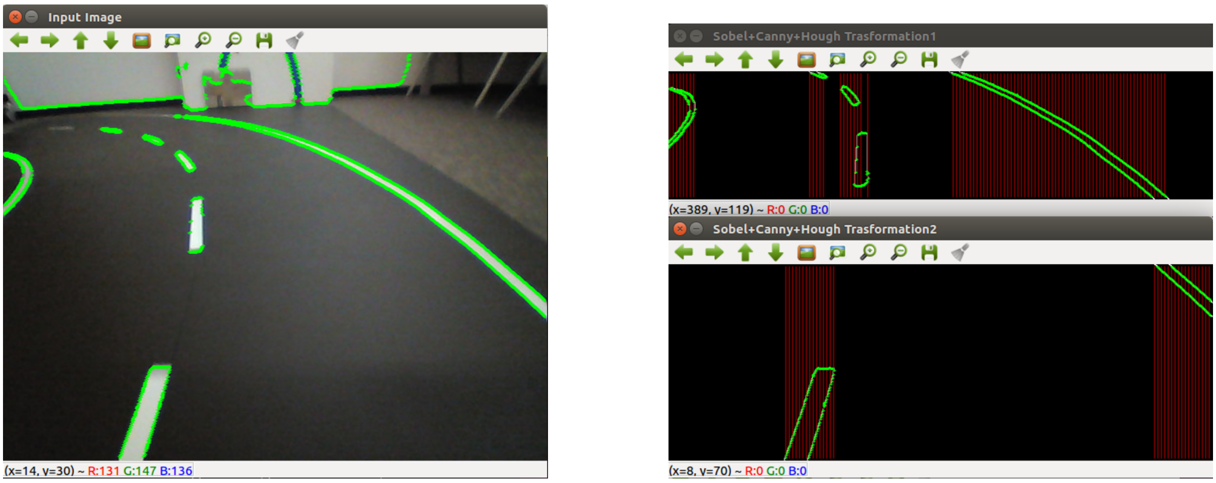
(b) Hough Transformation with Cropping Image at Straight Lane in 2 pieces

Abbildung 3.6.: Standard Hough Transformation in Straight Lanes

Step 3 : After the lanes are grouped separately in two different small frames, the heights of which are 230 and 150 pixels, the points (pixels) on the lanes must be found in two small frames. For that, we have to use Probabilistic Hough Transformation. The Probabilistic Hough Transformation is a bit different than Standard Hough Line Transformation because Standard Hough Transformation is more suitable for straight lanes. However, if there is a curve, the Standard Hough Transformation is not able to detect lanes very well. As previously mentioned, a line can be represented as $y = mx+c$ or in parametric form, as $\rho = x\cos\theta + y\sin\theta$ where ρ is the perpendicular distance from origin to the line, and θ is the angle formed by this perpendicular line and horizontal axis measured in counter-clockwise. However, is different in the case of the Probabilistic Hough Transformation. A line is represented by two or more points. If the

Probabilistic Hough Transformation finds at least two points from the same line, it represents the two end points of these lines.

In this master's thesis, the Probabilistic Hough lines are not drawn because just the points (pixels) which are detected in the lanes with the Probabilistic Hough Transformation are needed.



(a) Original Image with Probabilistic Hough Transformation

(b) A Image with Sobel Operator and Probabilistic Hough Transformation

Abbildung 3.7.: Probabilistic Hough Transformation Points

In Figure 3.7a and Figure 3.7b, the Probabilistic Hough Transformation Points (green pixels) are shown. The Probabilistic Hough Transformation detected a large number of pixels. By changing the parameters of the Probabilistic Hough Transformation found in OpenCV, fewer Hough Points on the lanes are able to be detected, but detecting too few Hough points can also cause some problems for detecting the lanes. On the other hand, detecting a large number of Hough Points results in more computing time, which is undesirable. So in this case, the parameters of Probabilistic Hough Transformation fuction in OpenCV must be optimized. Thanks to optimization, the best solution (less computing time and good lane detection) is found.

Step 4 : It is now known in which pixel columns the lanes lie (Step 2), and the points (pixels) on the lanes are also shown thanks to the Probabilistic Hough Transformation (Step 3), and after these two steps, the starting points (pixels) of the lanes can be found. In order to do that, the Hough Points for each group of Hough lines that were found in Step 2 are compared with each other, and then the last pixels in the columns should be found. If the camera can see three lanes, then the different starting points for the three different lanes must be found. This means that this process must be done for all lanes which can be seen in the frame.

Step 5 : The next step in this method is getting the Hough Points which are relevant to the lanes. Now, for each lane, the Hough Points must be also grouped. For getting the Hough Points which are relevant to the lanes, what I term the 'rectangle' method is to be used. In the rectangle method, a rectangle is drawn for each lane at the starting point found in Step 4, and the coordinates of all of the Hough Points in that rectangle are saved in a vector. The highest

Hough Point in the rectangle must then be found. From that point, another rectangle must be drawn, but the sizes of the rectangles are becoming progressively smaller, because the objects seem smaller the further away they are from the camera. The sizes of the rectangles are also similar but there is an exception with regard to the middle lane. The middle lane has dashed lines so the rectangles in the middle lane must be bigger than in the left and right lanes. As seen in Figure **??** (rectangles shown in blue), with this method, the noise and the Hough Points which are not relevant to the lanes are removed.

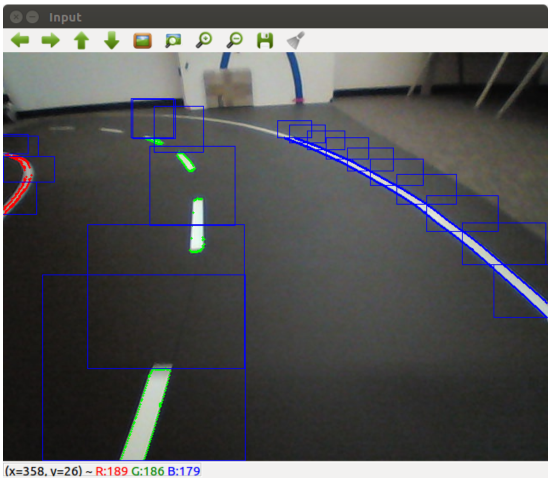


Abbildung 3.8.: Rectangle Method

Step 6 : The next step of this method is curve fitting. Curve Fitting is an algorithm which gives a mathematical description and this mathematical description provides the best fit for a series of data points. But in this master's thesis, the curve fitting to be performed is modified. In this curve fitting, the coordinates of Hough Points are used but the x and y coordinates of these Hough Points are swapped, because the y-axes of these coordinates have more range than the x-axes. As a result, this swap raises the stability of the curve fitting.

After using the Hough Points as input, three different mathematical equations are produced. One of these equations is for the left lane, another is for the middle lane, and the last equation is for the right lane. In order to produce these equations, naturally, only the relevant Hough points are used. For example, for left lane curve equation, only the Hough Points from the left lane were used.

Step 7 : The next step of this method is plotting the fitted curves produced by the curve fitting function. We start from first pixel and continue to the 480th pixel vertically and the output values of the equation for each pixel in the column are found. For each lane, the fitted curves are plotted in different color. In Figure **??**, the fitted curve can be seen.

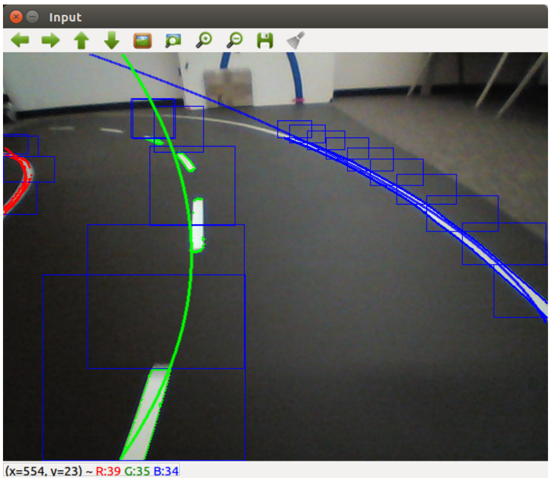


Abbildung 3.9.: Fitted Curve

Step 8 : Until this step, the fitted curves were plotted from the camera view, but at the end of the project, the fitted curves of the lanes need to be able to be seen from a bird's-eye view. As a result, the perspective of lanes must be changed from the camera side to the top of the track side. This step can be termed 'Inverse Perspective Mapping(IPM)'. For IPM, a function from OpenCV, which is called 'findHomography', was used. Thanks this function, all fitted curve points can be converted to the perspective of the top of the track side. It was also possible to convert all pixels from the camera perspective to the top of the track perspective, but in that case, it would have been necessary to convert 640x480 pixels, which means 307200 pixels in total. In this case, it is only necessary to convert 480 pixels for each lane, and so 1440 pixels in total. As a result, this method is more efficient than converting all pixels.

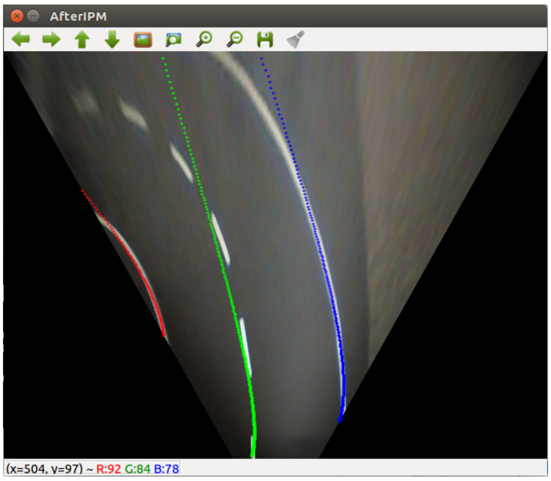


Abbildung 3.10.: Curve Fitting

Step 9 : The last step of this method is publishing the coefficients of the equations of the fitted curves from the bird's-eye view. In order to activate the automobile for autonomous driving, only the equations of the fitted curves are needed. For this step, a ROS Topic must be created. The frequency of the rostopics can be adjusted.

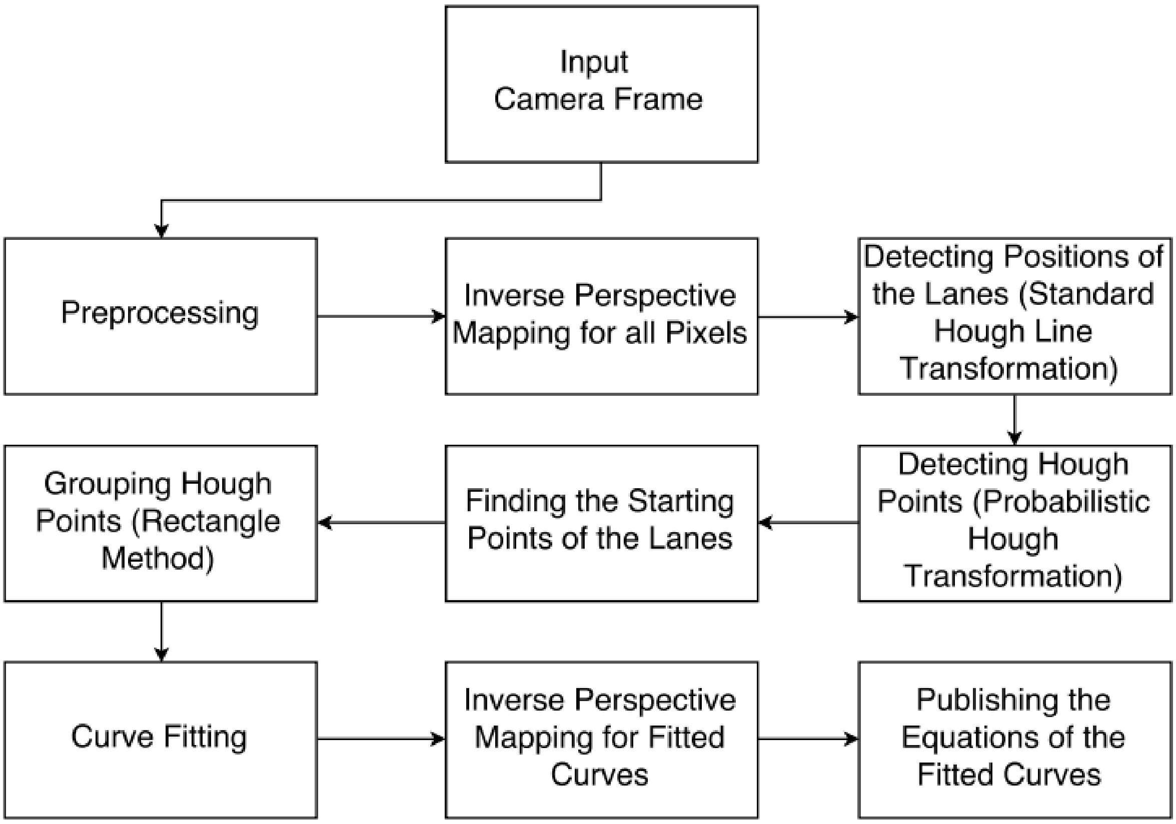
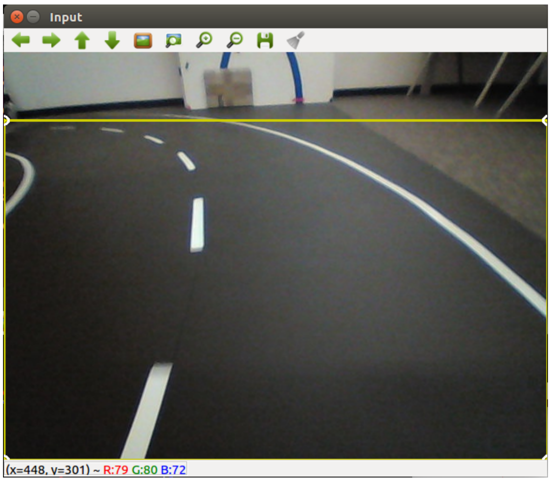


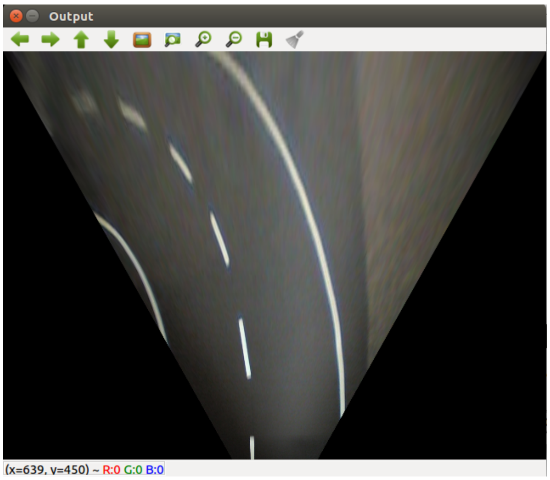
Abbildung 3.11.: Block Diagram of Case 2

Step 1 : As previously mentioned, the preprocessing part is common for all methods. So in this method, the preprocessing part was also the first step.

Step 2 : In this method, the frames taken from the camera, are converted directly from the camera perspective to the bird's-eye view perspective. In Section 3.4.3 (Method 1), only the curve fitting pixels were converted from the camera perspective to the top of the truck perspective. The time needed to convert 640x480 pixels (307200 pixels in total) is a bit more when compared to Method 1. The original frame, which can be seen in Figure 3.12a, is converted to the frame which can be seen in Figure 3.12b by OpenCV 'findHomography' function.



(a) Original Image



(b) A Image with Inverse Perspective Mapping

Abbildung 3.12.: Inverse Perspective Mapping

Step 3 : The next step of this method is finding the vertical pixels which the lanes lie between. In Method 1(Section 3.4.3, the first 100 pixels from the top of the frame were cropped and then rest of the frame was divided into two pieces. This was necessary because otherwise, the straight lanes would have caused a problem in the grouping of the lanes. In this method, it is not necessary to crop the frame. In Method 1, except for the first 100 pixels of the frame, it was necessary to apply the Standard Hough Line Transformation for 380x640 pixels. In this method, it sufficient to apply the Standard Hough Line Transformation to only the last 200 pixels of the frame, because these 200 pixels are able to cover all lanes in the frame. In Figure 3.13a, all lanes can be distinguished thanks to the Hough lines.

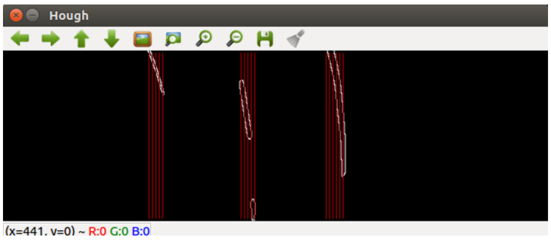


Abbildung 3.13.: Standard Hough Line Transformation after IPM Method

Step 4 : After finding which pixel columns the lanes lie between in Step 3, the Probabilistic Hough Transformation was used. Thanks to Probabilistic Hough Transformation, the Hough Points appear on the lanes. As mentioned in Method 1 in Section 3.4.3, there are some parameters in Probabilistic Hough Transformations, so the number of Hough Points can be descreased or increased. Decreasing the number of Hough Points also decreases the computing time but decreasing the number of Hough Points by too much can decrease the accuracy of lane detection. Therefore, the parameters must be set in an optimal way.

Step 5 : We have already found out which pixel columns the lanes lie between. The Hough Points must now be grouped according to lanes. If the camera can see all three lanes, then there must also be three different groups of Hough Points. However, if there is a curve, the camera can see just the right and middle lane, so the Hough Points must be grouped into two groups in this case. For each lane, the starting points of the lanes must be found. For that, all Hough Points in a group must be compared with each other. The Hough Points which are the closest to the bottom of the frame are the starting points.

Step 6 : The next step of this method is the k-nearest neighbors algorithm (KNN). KNN is a learning algorithm. Here KNN is used instead of the rectangle method used in Method 1. The KNN algorithm divides the frames into some equal-sized pieces. The nearest neighbor points are found in each piece. This process begins from the starting points of the lanes which were found in Step 5. The KNN functions are used from OpenCV. There are some parameters in these functions. For example, we can set the number of neighbor points to be found. The parameters must be set optimally, meaning the project must work fast and efficiently.

Step 7 : After finding the points from KNN in Step 6, which are to be used as the input values of the curve fitting function, then the function of the curve fitting is applied. The advantage of the KNN algorithm is that it does not return so many Hough Points, which are the input values of the curve fitting, when compared to rectangle method. So here, the curve fitting function produces the mathematical equations of the lane much faster when compared to Method 1.

Step 8 : After the equations of the fitted curves for each of the lanes are calculated, these equations have to be plotted like in Method 1. Here, the x and y coordinates of the Hough Points were also swapped in order to get more stable curves. For each of the vertical pixels, the results of all of the equations were plotted in the frame.

Step 9 : The last step of this method is also to publish the coefficients of the equations of the fitted curves of each of the lanes which can be seen by the camera.

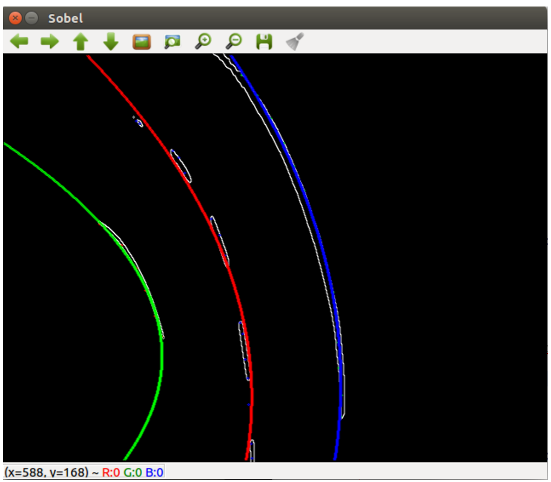


Abbildung 3.14.: Output Image

3.4.5 Method 3 : IPM + Hough Transformation + Rectangle Method + Curve Fitting(v0.5.1)

This method is a composite method of the Method 1 in Section 3.4.3 and Method 2 in Section 3.4.4. As in Method 2, at the beginning of this method, the preprocessing part is applied, and then Inverse Perspective Mapping for all pixels is implemented. After that, for 200x640 pixels, the Standard Hough Transformation is used and then for all pixels Probabilistic Hough Transformation is implemented. Until this point, the implementation is completely identical to Method 2. In Method 2, however, at the point in the process where the k-nearest neighbours (KNN) method is used, the rectangle method used in Method 1 in Section 3.4.3 is applied. There is, however, a difference between the rectangle method utilized here and the rectangle method from Method 1. In Method 1, the size of rectangles decreased progressively. However, in this method, the size of the rectangles is always the same because the Inverse Perspective Method(IPM) is used at the beginning. As a result, the lanes are shown from the top of track. The size of the rectangles in the middle lane is bigger than the size of the rectangles in the left and right lanes, however. Because of the dashed lines, it was necessary to use larger rectangles in the middle lane.

After the rectangle method is used, the curve fitting method is used. As with the other methods, the x and y coordinates of Probabilistic Hough Points are swapped after the rectangle method is applied. Thanks to this swap, the curve fitting is more accurate. After producing three different equations for three different lanes, the output values for all pixel columns are calculated and plotted in the frame. In Figure 3.15a, the fitted curves are shown.

The last step of this method is as in the others: to publish the equations of the fitted curves produced from the lanes. These fitted curves are published with Ros Topics.

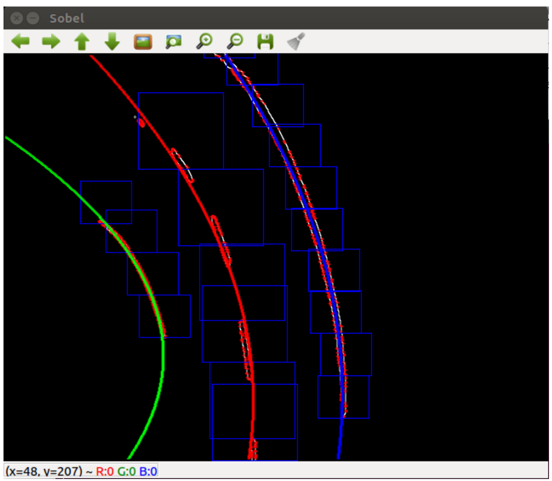


Abbildung 3.15.: Output Image

3.4.6 Method 4 : Resize + IPM + Hough Transformation + KNN + Curve Fitting(v0.7.1)

This method is very similar to Method 2. There is just one difference between Method 2 in 3.4.4 and this method. The only difference is resizing the frame at the beginning. The frame comes from the camera with a 640x480 pixel resolution, but this frame is resized from a 640x480 pixel resolution to a 320x240 pixel resolution. In order to do this, a 'resize' function from OpenCV is used because the smallest resolution able to be obtained from the camera is 640x480 pixels. Due to the resizing, it was sometimes necessary to use different parameters than in Method 2 in order to detect lanes optimally (efficiently and accurately). Results of this method are shown in Figure 3.16a.

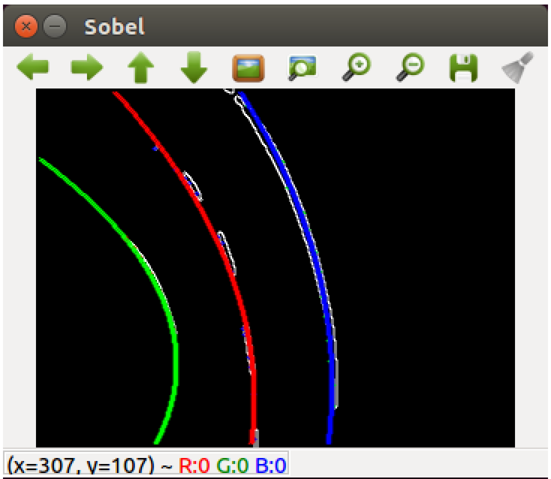


Abbildung 3.16.: Output Image(320x240 pixels resolution)

4 Evaulation and Discussion

In this chapter, the algorithms for each case will be evaluated, which are used in this master thesis. Also the parameters in the algorithms will be changed and their effects to the computing time and to the reliability of lane detection will be observed. The cases will be also compared with each other and trid to find the most efficient and reliable case. One of the part of this thesis, the average computation times for each algorithms and for the total lane detection process for each cases will be also measured. End of this chapter, the problems which can occur, during the lane detection will be defined.

5 Related Works

6 Conclusion

Literaturverzeichnis

[1] Kraski, Gerhard: *More traffic accidents but fewer fatalities than ever before in 2016*, Juli 2017.

[2] Braunschweig, Technische Universität: *Carolo-Cup Regulations 2017*, October 2016.

[3] Lin, Chien-Chuan und Ming-Shi Wang: *A Vision Based Top-View Transformation Model for a Vehicle Parking Assistant*. Sensors, Department of Engineering Science, National Cheng Kung University Taiwan, No.1, University Road,Tainan City 701, Taiwan, 2012.

[4] Ramesh Jain, Rangachar Kasturi, Brian G. Schunck: *Machine Vision*. McGraw-Hill, 1995.

[5] Wikipedia: *Sobel operator*, October 2017.

[6] Team, OpenCV Dev: *Image Filtering*, October 2017.

[7] Team, OpenCV Dev: *Operations on Arrays*, October 2017.

[8] Technology Delhi, Indian Institute of: *Canny Edge Detector*, March 2009.

[9] Wikipedia: *Canny Edge Detector*, May 2017.

[10] Team, OpenCV Dev: *Canny Edge Detector*, October 2017.

[11] Duda, Richard O. und Peter E. Hart: *Use of the Hough Transformation To Detect Lines and Curves in Pictures*. Stanford Research Institute, Menlo Park, California, 15, January 1972.

[12] Sharma, Nabin: *Linear Hough Transform Using Python*, December 2012.

[13] Team, Opencv Dev: *Hough Line Transform*, October 2017.

[14] Xindong Wu, Vipin Kumar, J. Ross Quinlan Joydeep Ghosh Qiang Yang Hiroshi Motoda Geoffrey J. McLachlan Angus Ng Bing Liu Philip S. Yu Zhi-Hua Zhou Michael Steinbach David J. Hand Dan Steinberg: *Top 10 algorithms in data mining*. Springer-Verlag London Limited 2007, 2007.

[15] Wikipedia: *k-nearest neighbors algorithm*, August 2017.

[16] Sadegh Bafandeh Imandoust, Mohammad Bolandraftar: *Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background*. S B Imandoust et al. Int. Journal of Engineering Research and Applications, 3:605–610, October 2013.

[17] Team, OpenCV Dev: *Fast Approximate Nearest Neighbor Search*, October 2017.

[18] *The KaleidaGraph Guide to Curve Fitting*.

[19] Frost, Jim: *Curve Fitting with Linear and Nonlinear Regression*, August 2013.

[20] *FITTING OF A POLYNOMIAL USING LEAST SQUARES METHOD*.

[21] MyCurveFit: *MyCurveFit*, 2017.

[22] Dirk, Thomas: *What is ROS*, May 2014.

[23] Wiki, ROS: *cv bridge ROS Wiki*, December 2013.

[24] team opencv dev: *Mat The Basic Image Container*, October 2017.

A Appendix

A.1 Default Values of Parameters

Parameter	Identification	Standard Value
Y^*	Height of IPM picture	-2
X^*	Width of IPM picture	0.8
f_x	x-value of local length	318.503
f_y	y-value of local length	318.266
c_x	x-value of optical center	320.129
c_y	y-value of optical center	208.651
h	0.9	0.8
α	0.9	0.8
β	0.9	0.8