# SENSOR FUSION OF A CAMERA AND 2D LIDAR FOR LANE DETECTION AND TRACKING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YASİN YENİAYDIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

AUGUST 2019

Approval of the thesis:

**SENSOR FUSION OF A CAMERA AND 2D LIDAR FOR LANE DETECTION AND TRACKING**

submitted by **YASİN YENİAYDIN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Klaus Verner Schmidt
Supervisor, **Electrical and Electronics Engineering, METU** _____

**Examining Committee Members:**

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering, METU _____

Prof. Dr. Klaus Verner Schmidt
Electrical and Electronics Engineering, METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _____

Prof. Dr. Umut Orguner
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Halit Ergezer
Mechatronics Engineering, Çankaya University _____

Date:

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Surname: Yasin Yeniaydın

Signature :

# ABSTRACT

## SENSOR FUSION OF A CAMERA AND 2D LIDAR FOR LANE DETECTION AND TRACKING

Yeniaydın, Yasin
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Klaus Verner Schmidt

August 2019, 114 pages

This thesis proposes a novel lane detection and tracking algorithm based on sensor fusion of a camera and 2D LIDAR. The proposed method is based on the top down view of a grayscale image, whose lane pixels are enhanced by the convolution with a 1D top-hat kernel. The convolved image is horizontally divided into a predetermined number of regions and the histogram of each region is computed. Next, the highest valued local maxima in a predefined ratio in the histogram plots are determined as candidate lane pixels. In addition, we segment 2D LIDAR data to detect objects on the road and map them to the top down view to determine object pixels. Pixels occluded by the detected objects are then turned into background pixels to obtain a modified top down view. Next, the Hough Transform is applied to the modified top down view to detect lines. These detected lines are merged based on their slopes and the interception points between the lines and bottom and top border of the image frame. After the merging process, the best lane pair is selected based on length, slope and interception points of the lines. Lastly, lane detection is carried out on the selected pair using a second-order polynomial with similar curvatures for the left and

right lane markings. The polynomial coefficients are determined via the least squares method and tracked by a Kalman Filter. In addition, the thesis provides methods for the reference trajectory generation, the computation of the lateral error and heading error of a vehicle for lane keeping. Computational and experimental evaluations show that the proposed method significantly increases the lane detection accuracy.

Keywords: Sensor Fusion, Lane Detection, Lane Tracking, Camera, 2d Lidar

# ÖZ

## ŞERİT TESPİTİ VE TAKİBİ İÇİN KAMERA VE 2D LIDAR SENSÖR FÜZYONU

Yeniaydın, Yasin
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Klaus Verner Schmidt

Ağustos 2019 , 114 sayfa

Bu tez çalışması kamera ve 2D LIDAR sensör füzyonuna dayanan yeni bir şerit tespit ve takip algoritması önermektedir. Önerilen yöntem kuş bakışı görünümünde, şerit pikselleri 1D kafa şapkası çekirdeği ile evriştirilerek yükseltilmiş gri seviye imgelere dayanmaktadır. Evriştirilen imge yatay olarak önceden belirlenmiş sayıda bölgelere ayrılır ve her bir bölgenin çubuk grafiği hesaplanır. Sonra, daha önce tanımlanmış belirli orandaki en yüksek değerlikli yerel maksimumlar, aday şerit pikselleri olarak belirlenir. Daha sonra, 2D LIDAR verileri nesneleri tespit etmek için bölütlenir ve kuş bakışı görünüme nesne piksellerini belirlemek için haritalanır. Tespit edilen nesneler tarafından işgal edilen pikseller, geliştirilmiş kuş bakışı görünümü elde etmek için arka plan piksellerine çevrilir. Bir sonraki adımda, çizgileri tespit etmek için geliştirilmiş kuş bakışı görünümde Hough dönüşümü uygulanır. Tespit edilen çizgilerin eğim ve çizgiler ile imgelerin alt ve üst sınırlarının kesişim noktaları kullanılarak, tespit edilen çizgiler birleştirilir. Çizgileri birleştirme işleminden sonra, çizgilerin uzunluk, eğim ve kesişim noktalarına bağlı olarak en iyi şerit çifti seçilir. Son olarak, seçilen şerit çifti üzerinde sol ve sağ şerit için benzer eğrilikli ikinci dereceden polinom kulla-

nılarak şerit tespiti gerçekleştirilir. Bu polinom katsayıları en küçük kareler yöntemi ile hesaplanır ve Kalman süzgeci ile takip edilir. Ek olarak, bir aracı şeritte tutmak için referans yörüngesinin oluşturulması ve yanal sapma hatası ile yönelme hatasının hesaplanması verilmektedir. Hesaplamalı ve deneysel değerlendirmeler, önerilen yöntemin şerit tespit doğruluğunu yüksek oranda artırdığını göstermektedir.

Anahtar Kelimeler: Sensör Füzyonu, Şerit Tespiti, Şerit Takibi, Kamera, 2d Lidar

*To my family*

## ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my supervisor Prof. Dr. Klaus Werner SCHMIDT for his guidance, understanding, patience and valuable comments. Throughout the thesis, it was a great honor to work with him for me.

I am very grateful to the Electrical and Electronics Engineering Department for the autonomous vehicle kit and TÜBİTAK-SAGE for the facilities provided for me throughout the thesis.

I would like to extend my appreciation to Berkay BAYKARA and Volkan SÜEL for their technical support and encouragement.

Lastly, I would like to send my endless thanks to my family, Ramazan YENİAYDIN, Hatice YENİAYDIN and Çağrı YENİAYDIN.

This work was supported by the Middle East Technical University Research Project GAP-301-2018-2740.

## TABLE OF CONTENTS

## LIST OF TABLES

TABLES

## LIST OF FIGURES

FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| 1-BT | 1-Bit Transform |
| 1D | 1 Dimensional |
| 2D | 2 Dimensional |
| LCF | 2D LIDAR Coordinate Frame |
| 3D | 3 Dimensional |
| ABD | Adaptive Breakpoint Detection |
| BEV | Bird's Eye View |
| CCF | Camera Coordinate Frame |
| CNN | Convolutional Neural Network |
| DBSCAN | Density-Based Spatial Clustering of Applications with Noise |
| dvRBNN | Distance Varying Radially Bounded Nearest Neighbor |
| EDLines | Edge Drawing Lines |
| FBD | Fixed Breakpoint Detection |
| HT | Hough Transform |
| ICF | Image Coordinate Frame |
| IPM | Inverse Perspective Mapping |
| KF | Kalman Filter |
| $k$NN | $k$ Nearest Neighbor |
| LDW | Lane Departure Warning |
| LIDAR | Light Detection And Ranging |
| LKAS | Lane Keeping Assist System |
| LoG | Laplacian of Gaussian |
| LSD | Line Segment Detector |
| LSM | Least Squares Method |

| | |
|---|---|
| NN | Nearest Neighbor |
| NAND | Neighborhood AND |
| RBNN | Radially Bounded Nearest Neighbor |
| RANSAC | Random Sample Consensus |
| ROI | Region of Intereset |
| ROS | Robotic Operating System |
| SVD | Singular Value Decomposition |
| SLT | Symmetrical Local Thresholding |
| VCF | Vehicle Coordinate Frame |

# CHAPTER 1

# INTRODUCTION

Vehicle crashes are mostly caused by faults of drivers [1, 2] due to drowsiness, fatigue, lack of attention, etc. To increase car safety independent from drivers, advanced driver-assistance systems (ADAS) which are electronic systems are designed and it is proven that safety attributes of the ADAS decrease the vehicle crashes by reducing the driver's faults [3]. Lane Departure Warning System (LDWS) and Lane Keeping Assist System (LKAS) are two examples of ADAS, which have a high potential to decrease the number of vehicle crashes [4, 5]. LDWS is a warning system informing drivers through vision, acoustic alarm, vibration of seat, etc. if the vehicle is on the way of crossing its current lanes without intention of the driver. LKAS is a little bit more advanced system which warns and supports the driver by keeping the vehicle in the center of the lane through controlling the lateral movement of the vehicle. The LDWS and LKAS mostly rely on two phases, lane detection in the first phase and computation of output signals from the lane detection phase for warning drivers (LDWS) or controlling vehicles (LKAS) in the second phase.

In the first phase, the detection of the lanes is mostly performed by utilizing vision-based techniques [5] - [6] using cameras mounted at the top of windscreen in the rear-view mirror unit. The detection of the lanes is basically performed in four stages. In the first stage, Region of Interest (ROI) which contains road area is determined to decrease computational complexity and discard non-relevant area which can be evaluated as noise. In the second stage, edge and brightness level of lane features are extracted mostly to determine candidate lane pixels through edge filters which enhance edges of the lanes or segmentation of the pixels based on the brightness level in the ROI through predetermined fixed threshold or dynamically computed adaptive threshold values. In the third stage, noise removal is realized based on geometric

attributes of the lanes such as slope of the lanes, lane marking width, lane width, parallelism of the lanes, etc. To compute the slope of the lanes, Hough Transform (HT) which detects lines is generally performed on the candidate lane pixels and the slopes of the lines are directly computed. The lane marking width is generally detected by means of blob filters which share features of the lane markings such as brightness level, width, length, etc. The candidate lane pixels which do not share the features of blob filter are removed. To make use of parallelism of the lanes and compute the lane width, Bird's Eye View (BEV) which is top-down view of the ROI in perspective image is obtained and HT is performed on the candidate lane pixels on the BEV image to detect the lines. After detecting the lines, interception points of the detected lines and top and bottom border of the BEV image frame are employed for determination of the parallelism and computation of the lane width between the detected lines. In this way, the candidate lane pixels forming current lanes based on the geometric features are specified as lane pixels. In the last stage, lanes are generally detected by estimating coefficients of lane models such as polynomials, splines, etc. on the lane pixels and the coefficients are tracked through Kalman or Particle Filter to take advantage of temporal relationship of captured images from the camera. As for expectations from the lane detection phase, these are being robustness against noise, shadows, being adaptable to different lighting conditions, ignoring objects in the view, having low computational complexity, etc.

In the second phase, reference trajectory generation and computation of lateral deviation from centerline of the detected lanes and heading error which is an angle between desired path and vehicle's heading direction are performed by employing the detected lanes in the first phase for controlling lateral movement of the vehicle or warning the driver.

The recent literature mostly offers diversified lane detection methods for LDWS and LKAS without proposing solutions for the second phase as follows.

## 1.1 Literature Research

[7] firstly determine a static ROI. In the ROI, feature extraction is performed by using simple filter. The filter is built based on the thickness of the lanes and the thickness are determined offline throughout the lane in the perspective view. Next, HT is applied for lane detection by considering the slopes of the lanes. Lastly, Kalman Filter (KF) is utilized for tracking the parameters of HT.

The study in [8] firstly detects the road to determine the ROI by the help of a stereo camera. After that, adaptive threshold is applied to extract the features. For that operation, different window size is set according to the distance from camera location. Furthermore, an edge filter emphasizing the vertical and diagonal edges is carried out on the feature extracted image. Finally, dynamic programming is performed to detect the optimal lane path.

The work [9] uses stereo camera, hence 3D information of the field of view and the RGB data are obtained. The authors firstly determine a seed region in front of the vehicle in the assumption that the seed region is a part of the road area. Then, they compute the normal vector of the seed region. Afterwards, the road is segmented by comparing the normal vector of the seed region with the remaining region in the view. Hereby, the ROI is specified. Later, a modified local adaptive thresholding using the mean value and the standard deviations of the gray value of pixels in certain sized-windows is applied for feature extraction. After that, orientation and the number of pixels constraints are applied to the candidate lane pixels for eliminating vehicles, obstacles, etc. Thereafter, initial position of the left and right lanes intersecting the horizontal line at the bottom of the image are determined. Lastly, lanes are recognized via least square method in the region determined by the initial positions and the orientation of the lanes.

[10] basically consists of two parts: line and curve fitting in the near and the far fields. Firstly, static ROI is determined for the near field and lines are extracted via Line Segment Detector (LSD) algorithm in the perspective view. Then, the lines are transformed to the BEV. Thereafter, DBSCAN algorithm is used to cluster line segments. After the clustering, line segments in the same cluster are fused by taking into account the length and geometric characteristics of the lanes. Hence, lines in the near field are obtained. Next, edges are extracted via Laplacian of Gaussian (LoG) filter

on the far field of BEV to realize curve fitting. Finally, weighted hyperbolic model including the parameters of the lines is fit to the lane pixels to complete lane detection.

In [11], the authors firstly transforms perspective image with a predefined static ROI to the BEV image. After that, temporal blurring is carried out to improve the quality of the worn-out lanes and first order Gaussian kernel is performed vertically to smooth the image, sequentially. Afterwards, second order Gaussian kernel is convolved horizontally to enhance lane features. Later, the road component are separated into four states: non-lane, clear lane, obstacle and unclear lane by using the pixel values of the filtered image. To specify the states, a window smaller than the dashed type lane region is exploited through the image. After specifying the states, an adaptive threshold value is applied to each window sized region in the image to extract the candidate lane pixels. Thereafter, angles of the candidate lane pixels are computed via Steerable filters. Next, feedback Random Sample Consensus (RANSAC) which uses the angles from the previous step and the curvature of the lanes from the previous frame as feedback is performed to detect the lanes. Lastly, KF is utilized to track the lane model parameters.

The authors in [12] firstly determine the dynamic ROI based on vanishing points and the accumulated statistical data of the location of the lanes. After that, line segments are extracted via EDLines using the grayscale image and the YUV color model. Next, noise removal is performed through slope filters parallel to the left and right lanes. Afterwards, line segments having approximate slopes in close region are merged and then the merged line segments are tested with the low-high-low intensity characteristics of the lanes to detect them. Lastly, KF is applied for tracking the slopes and the intersection point of the lanes at the bottom border of the image.

[13] simply consists of 3 steps. First, inverse perspective mapping (IPM) based on vanishing point is performed. Second, lane extraction via complement LoG and binary blob filtering which reduces noise by using the properties of the lanes such as the shape, orientation, etc. Third, HT and RANSAC algorithm are applied for the lane detection.

[14] makes use of dynamically changing ROI, Sobel operators for edge detection and HT for lane detection. It defines the ROI by scanning each row of the image for sudden change between the mean values of the rows since the sky and the road part of

the image have different brightness values.

In [15], feature extraction is performed via Step-Raw filter on grayscale image. Then, lanes are detected by HT and then corrected by KF on BEV. Lastly, lane is modeled as spline through Particle Filter around the detected lanes.

In [16], static ROI is determined to discard irrelevant part and to reduce computational complexity for lane detection, then feature extraction are accomplished by a 1-bit transform and a Sobel operator, consecutively. Lane markings are detected based on HT and lastly, temporal relationship for lane parameters are built by KF.

[17, 18] detect lanes to analyze whether or not aggressive driving is performed. To do that, [17, 18] first applies temporal filtering to grayscale image. After that, road pixels are determined through mean and variance of low gradient pixels on the temporally filtered grayscale image. Next, binary image is acquired on the non-road pixels by utilizing an adaptive threshold to determine candidate lane pixels. Then, binary BEV image and histogram plot of the binary BEV image are acquired, sequentially. After that, lanes are detected through HT by utilizing the candidate lane pixels around peaks of the histogram plot. In addition to that, ego-lane are determined by the peaks closest to camera center. Lastly, lanes are tracked by KF.

In [19], Symmetrical Local Thresholding is utilized for feature extraction and it processes independently left and right hand side of each row. Then, it applies one dimensional connected component to obtain connected features and to remove the salt and pepper type noise. After that, further noise is ejected on the feature map based on the idea in which the difference of the slope of left and right lanes do not be more than a few degrees. Lastly, HT is performed to detect the lanes.

In [20], Gaussian kernel smoothes the BEV in the vertical direction firstly. After that, the lanes are made more certain via Second Order Gaussian kernel in the horizontal direction of the BEV. Then, the pixel values in each row is sorted from lowest to the highest and it keeps brightest certain number pixels for each row. Next, the resultant image is binarized. Lastly, lanes are robustly modeled as lines by HT and then as Bezier splines via RANSAC algorithm around the detected lines, consecutively.

In [21], firstly ROI with fixed ratio rectangle is determined. Then, it converts the RGB image to grayscale image by using minimum over R and G channels of the image and afterwards Median Local Threshold is applied on the grayscale image. Next, morphological dilation operation is performed in the vertical direction to connect the lane

pixel candidates. After that, the graycale image is convolved with a Gaussian kernel in horizontal direction to find the peaks in each row and the lane pixel candidates which are not around the peaks are evaluated as noise and discarded. Lastly, HT is applied to model the lanes as lines.

[22] firstly apply an edge filter to detect vertical edges. Then, perspective images are converted to BEV. Next, candidate lanes are detected via HT. Lastly, candidate lanes which are parallel and having a predetermined lane width on BEV are detected as actual lane markings.

[23] makes use of the parallelism of lane markings and develops a parallel snake lane approach which forces being parallel on left and right lane models on BEV.

[24] firstly determines a static ROI with fixed ratio rectangle to decrease the computational complexity and to discard the non-lane features which can be evaluated as lane features out of the ROI. Then, binary image is obtained by employing Sobel operator to grayscale image. After that, lanes are detected through HT. Lastly, position and orientation of the road boundaries are tracked by KF.

[25] firstly obtains grayscale image and dynamically changing ROI is determined based on the prediction of the road model. Then, each row is analyzed independently to determine if the pixels in that row are foreground pixels or not. For a pixel point, a threshold value is computed by using statistical data and mean intensity values within the surrounding region of the pixel. After computing threshold value for each pixel in the ROI, the image is binarized. Next, foreground pixels are examined by taking morphological properties and decided if they are lane markings or not. Lastly, positions of the Bots Dot type lane are tracked by KF.

[26] firstly extracts lane features by means of LSD. Then, direction priority search is applied for eliminating non-lane features. Direction priority search simply connects line segments having similar slopes in close region. Line segments having small number of pixels are eliminated after direction priority search. Next, ROI is determined through vanishing point estimated by the intersection of remaining line segments. Afterwards, line segments out of the ROI is filtered out, as well. Lastly, single lane pair is detected by a score function choosing the line segments intersecting vanishing point closely in the ROI.

[27] firstly determines a static ROI. Then, feature extraction is carried out via thresholding. The threshold value is computed by the help of Otsu's method. Furthermore,

noise is removed by taking the geometric characteristics of the lane segment features in detection window such as number of pixels, edge number, etc. Next, center points of each lane features in detection window are computed. Afterwards, linear type lane fitting is realized by RANSAC algorithm. Finally, lane tracking is achieved around the detected lane region if the lanes are detected in the previous frame.

[6] is a lane feature extraction algorithm and firstly performs camera calibration in the pre-processing step. After that, images are converted from RGB to HSI color space. Lastly, binary image is obtained via thresholding H, S and I channels of the HSI color space by taking into account white and yellow lane markings.

In addition to machine vision-based techiques for lane detection and tracking given above [7]-[6], the literature offers deep learning-based approaches [28, 29, 30, 31] for that problem. [28] firstly transforms the image to the BEV. Then, yaw angle of the vehicle with respect to the lanes is estimated to correct the lanes in linear part. Afterwards, BEV image is convolved with Gaussian kernels in horizontal and vertical direction to enhance lane pixels. Furthermore, they train a Convolutional Neural Network (CNN) classifier to discard non-lane region. The training data is constructed automatically by using the peak points of the local waveforms of the subparts of the vertically divided BEV image and the geometric properties of the lanes. Lastly, the outputs of the CNN classifier are the absence or the presence of the lane markings in each classification blocks which has a predefined rectangular size.

[29] is based on CNN consists of two branches. In the first branch, lane features are extracted, thus binary image is obtained via trained network. In the second branch, lane pixels are clustered based on each lane. In this way, each lane pixel has a lane ID representing its lane. Lastly, IPM is carried out to fit a polynomial to each lane on BEV via Homography matrix having 6 degrees of freedom. The Homography matrix is computed by the network referred as H-Net to resolve the issues of road plane changes.

[30] proposes an approach estimating lane positions based on CNN network referred to as DeepLanes employing the images from down-facing cameras on the left and right sides of the vehicle instead of front-facing cameras. The network is trained using manually-labeled and artificially generated images using real road images with no lane markings as background. Lastly, the network outputs row index of the lane marking which is closest to the vehicle on the left and right side.

[31] presents a unified multi-task CNN-based network which accomplishes road and lane marking detection. To maintain annotated lanes along the pooling and convolutional layers, grid-based annotation is acquired. In this way, lane annotations become wider. Furthermore, the network is trained to predict the vanishing point in order to make the kernels gain global status of the lanes in the images.

## 1.2 Shortcomings of the Existing Methods

In most of the machine vision-based methods as given in [7]-[6], each pixel is evaluated for candidate lane pixel extraction. However this approach leads to extract noisy pixels due to shadows, cracks on the roads, etc. and high computational complexity since all the pixels are utilized in the lane detection. For the lane detection methods, HT and RANSAC are mostly performed on the candidate lane pixels as they are robust to noise and allow gaps between the pixels. However, they are both computationally complex. Moreover, HT is generally utilized for line detection and this results in coarse lane detection in the case of curved roads. When it comes to lane tracking, KF and Particle Filter are mostly used. Whereas KF is less computationally complex, Particle Filter is more robust to noise.

Deep learning-based methods as given in [28]-[31] require copious number of labeled images covering almost all conditions such as different weather conditions, days and nights, simple and complex shaped roads, objects in the view, etc. In addition to that, determining an appropriate network type and structure is very time-consuming since they are generally determined in an ad hoc way. Besides, it is really hard to specify the weaknesses and strengths of the trained network by analyzing the outputs of the hidden layers.

In addition to the problems stated above, vision-based lane detection methods using one camera have limitations in severe conditions [5]. For example, it is highly probable to get noisy feature extracted frames when there exist vehicles in the camera view, impairing the lane detection accuracy. To overcome such limitations, there is little work on the fusion of different sensor modalities. [8] and [9] utilize stereo camera for distance information to extract road region. After that, they detect lanes on

the extracted road. However, using stereo camera for distance measurement can be misleading in outdoor environments due to varying illuminations on the left and right cameras of the stereo camera. [32] is based on the fusion of LIDARs and cameras since they reliably have complementary perception capabilities. [32] first extracts the drivable region via segmentation of LIDAR points and then detects lanes on the extracted region using cameras. However, the algorithm requires eight 2D LIDARs and seven cameras.

## 1.3 Scope of the Thesis

The thesis proposes a novel lane detection and tracking algorithm based on sensor fusion between a camera and 2D LIDAR. Moreover, it proposes methods for computation of output signals for LDWS and LKAS and addresses the stated problems of the lane detection.

The proposed lane detection and tracking algorithm based on sensor fusion of a camera and 2D LIDAR addresses the stated machine vision feature extraction and lane detection shortcomings covering the objects in the camera view. Hereby, machine vision-based approach is selected due to the explained shortcomings of the deep learning-based approaches in previous Section 1.2 for such a vital application related to human life. Additionally, the thesis proposes methods for the reference trajectory generation and computation of the heading and the lateral errors for further use by lateral controllers of autonomous vehicles. Throughout the thesis, all proposed solutions are tested experimentally and quantitatively. For the proposed lane detection and tracking algorithm, quantitative and illustrative results are presented on the CalTech database [20] and synthetic data generated from this database. In addition to that, the proposed method is evaluated on the autonomous vehicle kit [33] which has camera, 2D LIDAR and a main board for developing algorithms for ADAS. To evaluate the method with the autonomous vehicle kit, firstly straight lanes are manually constructed. Then, the method is implemented and illustrated experimental results are obtained. Computational and experimental evaluations show that the proposed lane detection and tracking method significantly increases the lane detection accuracy. For the proposed sensor fusion method between camera and 2D LIDAR, quantitative

and illustrated evaluation results on synthetic data generated from [20] are obtained. To evaluate the method with the autonomous vehicle kit, objects are placed on the straight lanes in the camera view. Then, the method is implemented and illustrated experimental results are obtained. Computational and experimental evaluations show that the proposed sensor fusion method significantly increases the lane detection accuracy. Additionally, the thesis proposes solutions for estimating camera height from road plane and tilt angle of the camera with respect to the road plane. Furthermore, formulation of projection of 2D LIDAR points to the road plane is derived. Illustrated results of the projection of 2D LIDAR points to the road plane prove the correctness of the estimation of the camera height, tilt angle and the formulation for the projection of the 2D LIDAR points to the road plane. Moreover, mostly employed 2D LIDAR data segmentation methods for object detection are compared in terms of accuracy and computational complexity on synthetic 2D LIDAR data generated via a novel algorithm given in the scope of the thesis.
The contributions of the thesis are summarized as follows:

- A novel lane detection and tracking algorithm
- A novel sensor fusion method using one camera and 2D LIDAR for lane detection and tracking in case of occluded lanes by objects
- Projection of 2D LIDAR points to road plane for mapping the LIDAR points in road level to BEV image
- Estimation of height and tilt angle of a camera with respect to road plane for projection of 2D LIDAR points to the road plane
- Comparison of 2D LIDAR data segmentation methods for object detection in terms of accuracy and computational complexity
- A novel synthetic 2D LIDAR data generation algorithm
- Computations of reference trajectory, heading error and lateral error for control applications
- Introducing arc-splines for lane modeling

The remainder of the thesis is organized as follows. Chapter 2 gives necessary background for vision-based, LIDAR-based methods and the auxiliary techniques for them in a detailed way. In Chapter 3, we explain and evaluate the proposed lane detection and tracking algorithm by addressing the stated shortcomings. In Chapter 4, we fuse 2D LIDAR data to the proposed lane detection and tracking algorithm to address the stated shortcoming of the objects in the camera view. Besides, the fusion of a camera and 2D LIDAR for lane detection and tracking is evaluated experimentally and quantitatively. Chapter 5 generates reference trajectory and computes heading and lateral errors as the outputs for further use by longitudinal and lateral controllers of autonomous vehicles. In Chapter 6, the conclusions of the thesis are described. Lastly, published and accepted papers throughout the thesis are summarized in Appendix A.

# CHAPTER 2

# BACKGROUND

In this Chapter, we give the necessary background for vision-based and LIDAR-based methods that are used and developed in the scope of this thesis. Section 2.1 gives general background information and Section 2.2 provides information for vision-based techniques. Segmentation methods for 2D LIDAR data are described in Section 2.3.

## 2.1 General Background

In this Section, the auxiliary techniques for vision-based and LIDAR-based methods are presented in a detailed way.

### 2.1.1 Least Squares Method

Least Squares Method (LSM) is a mathematical tool for fitting the best curve to a set of given samples by minimizing the least squares error between the curve and the samples [34].

Consider a set of $N$ measurement pairs $\{(i_1, j_1), (i_2, j_2), \ldots, (i_N, j_N)\}$ and a second order polynomial curve model as in equation 2.1. Hereby, it is assumed that the measurement pairs and the unknown coefficients of the second order polynomial constitute an overdetermined system, that is, $N > 3$.

$$ai^2 + bi + c = j \tag{2.1}$$

Then, the LSM formulation determines the coefficients $a, b, c$ as

$$x_{\text{LSM}} = (A_{\text{LSM}}^{\text{T}} A_{\text{LSM}})^{-1} A_{\text{LSM}}^{\text{T}} y_{\text{LSM}} \tag{2.2}$$

Hereby, the matrices and vectors

$$A_{\text{LSM}} = \begin{bmatrix} i_1^2 & i_1 & 1 \\ i_2^2 & i_2 & 1 \\ \vdots & \vdots & \vdots \\ i_N^2 & i_N & 1 \end{bmatrix}, x_{\text{LSM}} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, y_{\text{LSM}} = \begin{bmatrix} j_1 \\ j_2 \\ \vdots \\ j_N \end{bmatrix} \tag{2.3}$$

are used and the the squared error vector

$$||A_{\text{LSM}} x_{\text{LSM}} - y_{\text{LSM}}||_2^2 = (A_{\text{LSM}} x_{\text{LSM}} - y_{\text{LSM}})^{\text{T}} (A_{\text{LSM}} x_{\text{LSM}} - y_{\text{LSM}}) \tag{2.4}$$

is minimized, where $A_{\text{LSM}}$, $x_{\text{LSM}}$ and $y_{\text{LSM}}$ are called as data matrix, model parameter vector and model measurement vector, respectively in the scope of the thesis.

### 2.1.2 Euclidean Distance

The Euclidean distance $(d_{\text{E}})$ is utilized as a tool for measuring the distance between two points, $q$ and $p$ as follows:

$$d_{\text{E}}(q, p) = ||q - p||_2 \tag{2.5}$$

### 2.1.3 Rotation Matrices

Rotation matrices, denoted as $R$, are examples of orthogonal matrices [35, 36, 37]. They are utilized for a rotation of a positional vector $v$ by a given angle $\theta$ in a fixed coordinate frame or for a rotation between two coordinate frames (to represent a vector in another coordinate frame) in the scope of the thesis.

Formally, the rotation of a positional vector $v$ by the angle $\theta$ with the resulting vector $v'$ is computed as

$$v' = R(\theta)\, v. \tag{2.6}$$

Hereby, $R(\theta)$ is the rotation matrix by the angle $\theta$.

In addition to the rotation of the positional vectors in fixed coordinate frames, positional vectors in a coordinate frame can be represented in the rotated version of this coordinate frame through a rotation matrix. With the convention in the thesis, the

rotation matrix, $R_{\mathrm{B}}^{\mathrm{A}}$, represents vectors in the coordinate frame B according to the coordinate frame A. The mathematical relation for this purpose is:

$$v_{\mathrm{A}} = R_{\mathrm{B}}^{\mathrm{A}}\, v_{\mathrm{B}}. \tag{2.7}$$

Hereby, $v_{\mathrm{A}}$ is the representation of $v_{\mathrm{B}}$ in the coordinate frame B according to the coordinate frame A.

Using the convention that clockwise rotation represents a positive angle in a right-handed coordinate frame, the rotation matrices in $\mathbb{R}^3$ are

$$\begin{aligned} R_{\mathrm{x}}(\theta) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \\ R_{\mathrm{y}}(\theta) &= \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \\ R_{\mathrm{z}}(\theta) &= \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{2.8}$$

where $R_{\mathrm{x}}(\theta)$, $R_{\mathrm{y}}(\theta)$ and $R_{\mathrm{z}}(\theta)$ are the rotation matrices by the angle $\theta$ around the x-axis, y-axis and z-axis of the right-handed coordinate frame, respectively.

### 2.1.4 Translation Vectors

Translation vectors are the vectors between the origins of two coordinate frames. With the convention in the thesis, the translation vector, $t_{\mathrm{B}}^{\mathrm{A}}$, represents the position of the origin of the coordinate frame B in the coordinate frame A.
An illustration of the vector is given in Figure 2.1.

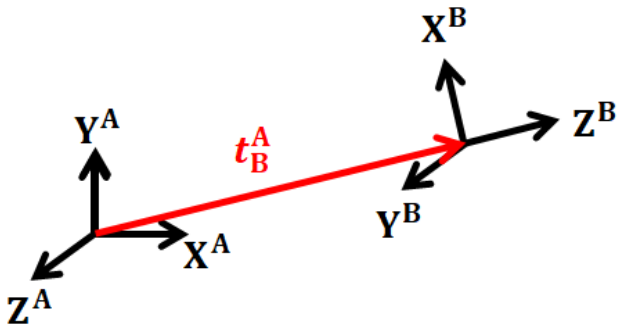



Figure 2.1: The translation vector $t_{\mathrm{B}}^{\mathrm{A}}$

### 2.1.5 Transformation Matrices

Transformation matrices are utilized to represent a vector in different coordinate frames. With the convention in the thesis, $T_{\mathrm{B}}^{\mathrm{A}}$, represents vectors in the coordinate frame B according to the coordinate frame A. The mathematical relation for this purpose is

$$\begin{bmatrix} v_{\mathrm{A}} \\ 1 \end{bmatrix} = T_{\mathrm{B}}^{\mathrm{A}} \begin{bmatrix} v_{\mathrm{B}} \\ 1 \end{bmatrix} = \begin{bmatrix} R_{\mathrm{B}}^{\mathrm{A}} v_{\mathrm{B}} + t_{\mathrm{B}}^{\mathrm{A}} \\ 1 \end{bmatrix}. \tag{2.9}$$

Hereby, $v_{\mathrm{A}}$ is the representation of $v_{\mathrm{B}}$ in the coordinate frame B according to the coordinate frame A.

**Remark 1** *A transformation matrix consists of a rotation matrix and a translation vector. For the vectors in* $\mathbf{R}^3$, $T_B^A$ *can be written as:*

$$\begin{bmatrix} R_B^A & t_B^A \\ 000 & 1 \end{bmatrix} \tag{2.10}$$

### 2.1.6 Convex Sets

A set of points, $P = \{p_1, p_2, \ldots, p_n\} \in \mathbf{R}^N$ is convex if the following holds that [38]:

$$\text{for all} \lambda_i > 0 \text{ such that } \sum_{i=1}^{n}(\lambda_i) = 1 : p = \sum_{i=1}^{n}(\lambda_i p_i) \in P. \tag{2.11}$$

**Remark 2** *For a geometrical interpretation, a set of points is said to be convex if all the line segments formed by any two points in the set are also in the set.*

An illustration of a convex and non-convex sets are given in Figure 2.2.

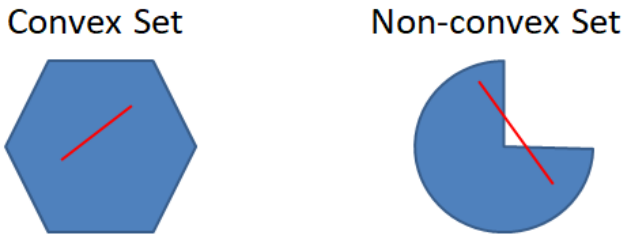



Figure 2.2: An illustration of a convex and non-convex sets

As can be seen in Figure 2.2, the convex set includes all the line segments formed by the points in the set. However, the non-convex set does not.

## 2.2 Vision-based Background

In this Section, the necessary background for image processing and machine vision principles are described in a detailed way.

### 2.2.1 Image Formats

In the scope of the thesis, RGB, grayscale, convolved and binary images are utilized to perform lane detection and lane tracking.

We write $M$, $N$ for the sets of rows and columns of an image respectively. Then, we consider RGB, grayscale, convolved and binary images as the maps $I_{\text{RGB}} : M \times N \times \{\text{R}, \text{G}, \text{B}\} \to [0, 1]$, $I_{\text{G}} : M \times N \to [0, 1]$, $I_{\text{C}} : M \times N \to (-\infty, \infty)$ and $I_{\text{B}} : M \times N \to \{0, 1\}$, respectively. $I_{\text{RGB}}(i, j, k)$, $I_{\text{G}}(i, j)$, $I_{\text{C}}(i, j)$ and $I_{\text{B}}(i, j)$ denote the pixel value at row $i$ and column $j$ for color $k$ of the RGB image, for the grayscale image, for the convolved image and for the binary image, respectively.
From this point on, $I(i, j)$ represents the pixel value at $i^{\text{th}}$ row and $j^{\text{th}}$ column of the images in general if the type of the image is clear from the context.

### 2.2.2 1D and 2D Image Convolutions

Image convolution is an operation symbolized by '$*$' sign for modifying the certain features of the input image. To modify the certain features, kernels ($\kappa$) are utilized with the dimensions $1 \times k_\kappa$ in 1D (one row and $k_\kappa$ number of columns) and $k_\kappa \times k_\kappa$ in 2D ($k_\kappa$ number of rows and columns), where $k_\kappa$ is an odd number for symmetry.

The convolutions for the $i^{\text{th}}$ row and $j^{\text{th}}$ column of an image are computed through the 1D and 2D kernels in equation 2.12 and 2.13, respectively in the scope of the thesis. Hereby, $\lfloor \bullet \rfloor$ denotes the floor operation that converts real numbers to the greatest integer less than or equal to the real numbers.

$$I_{\text{C}}(i,j) = I(i,j) * \kappa = \sum_{n=-\lfloor k_\kappa/2 \rfloor}^{\lfloor k_\kappa/2 \rfloor} I(i, j+n)\kappa(1, n + \lfloor k_\kappa/2 \rfloor + 1) \tag{2.12}$$

$$\begin{aligned} I_{\text{C}}(i,j) = I(i,j) * \kappa = \sum_{m=-\lfloor k_\kappa/2 \rfloor}^{\lfloor k_\kappa/2 \rfloor} \sum_{n=-\lfloor k_\kappa/2 \rfloor}^{\lfloor k_\kappa/2 \rfloor} I(i+m, j+n) \\ \kappa(m + \lfloor k_\kappa/2 \rfloor + 1, n + \lfloor k_\kappa/2 \rfloor + 1) \end{aligned} \tag{2.13}$$

Some of the well-known kernels are given in 2.14-2.19. Unweighted $3 \times 3$ smoothing kernel:

$$1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.14}$$

Weighted $3 \times 3$ smoothing kernel with Gaussian:

$$1/8 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.15}$$

$3 \times 3$ sharpening kernel:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{2.16}$$

$3 \times 3$ horizontal Sobel kernel for horizontal edge detection:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & -2 & 1 \end{bmatrix} \tag{2.17}$$

$3 \times 3$ vertical Sobel kernel for vertical edge detection:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.18}$$

$1 \times 9$ top-hat kernel for lane marking extraction [39]:

$$\begin{bmatrix} -1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 \end{bmatrix} \tag{2.19}$$

### 2.2.3 RGB to Grayscale Image Conversion

An RGB image is converted to a grayscale image as follows [40]:

$$I_{\mathrm{G}}(i,j) = 0.299\, I_{\mathrm{RGB}}\,(i,j,\mathrm{R}) + 0.587\, I_{\mathrm{RGB}}(i,j,\mathrm{G}) + 0.114 I_{\mathrm{RGB}}(i,j,\mathrm{B}) \tag{2.20}$$

### 2.2.4 Global Thresholding

A binary image is obtained from a grayscale image $I_{\mathrm{G}}$ or a convolved image $I_{\mathrm{C}}$ through global thresholding in the scope of the thesis as follows:

$$I_{\mathrm{B}}(i,j) = \begin{cases} 1 & \text{if } I_{\mathrm{G}}(i,j), I_{\mathrm{C}}(i,j) \geq th_{\mathrm{f}} \\ 0 & \text{if } I_{\mathrm{G}}(i,j), I_{\mathrm{C}}(i,j) < th_{\mathrm{f}} \end{cases} \tag{2.21}$$

Here, $th_{\mathrm{f}}$ represents a fixed threshold value.

Note that candidate lane features are represented as 'ones' in the binary images throughout the thesis.

### 2.2.5 Histogram

Histogram ($hist$) is a plot which is the summation of the pixel values in each column of an image and computed as follows:

$$hist(j) = \sum_{i=1}^{N_{\text{row}}} I(i,j) \tag{2.22}$$

where $N_{\text{row}}$ represents the number of rows in the images.

### 2.2.6 Inverse Perspective Mapping

Inverse Perspective Mapping (IPM) transforms perspective images to bird's eye view (BEV) images via a homography matrix ($H$) which has eight degrees of freedom as follows [41]:

$$\alpha \begin{bmatrix} u^{\text{h}} \\ v^{\text{h}} \\ 1 \end{bmatrix} = H \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{2.23}$$

where $\alpha$ is a scalar, $H$ is the homography matrix, $u$, $u^{\text{h}}$ and $v$, $v^{\text{h}}$ are the column and row correspondences in the perspective and BEV images, respectively.

To estimate the homography matrices, four or more pixel position correspondences are needed. In a lane detection application, the pixel position correspondences are chosen such that lanes in the BEV image become almost parallel. To do that, four lane pixel positions (e.g. two pixel positions from the left, two pixel positions from the right) in the perspective image and four pixel position correspondences in parallel for BEV image are specified.

Let $(u_1, v_1), (u_2, v_2), (u_3, v_3), (u_4, v_4)$ and $(u_1^{\text{h}}, v_1^{\text{h}}), (u_2^{\text{h}}, v_2^{\text{h}}), (u_3^{\text{h}}, v_3^{\text{h}}), (u_4^{\text{h}}, v_4^{\text{h}})$ become the correspondences, then the parameters of the homography matrix, $[h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^{\text{T}}$ as a vector, are estimated by finding the input vector to the following matrix, which generates the smallest output, via Singular Value Decompo-

sition (SVD):

$$\begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1u_1^{\mathrm{h}} & -v_1u_1^{\mathrm{h}} & -u_1^{\mathrm{h}} \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1v_1^{\mathrm{h}} & -v_1v_1^{\mathrm{h}} & -v_1^{\mathrm{h}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_4 & v_4 & 1 & 0 & 0 & 0 & -u_4u_4^{\mathrm{h}} & -v_4u_4^{\mathrm{h}} & -u_4^{\mathrm{h}} \\ 0 & 0 & 0 & u_4 & v_4 & 1 & -u_4v_4^{\mathrm{h}} & -v_4v_4^{\mathrm{h}} & -v_4^{\mathrm{h}} \end{bmatrix} \tag{2.24}$$

For better estimation, the selected pixel positions in the perspective and BEV images are normalized as follows [42]:

- The pixel positions are translated such that the centroid of the pixels moves to the origin.

- The pixel positions are scaled such that average distance from the origin becomes $\sqrt{2}$.

### 2.2.7 Hough Transform

Hough Transform (HT) is used for line detection in the scope of the thesis since it is robust against noise and allows gaps between the pixels forming the line [43].

A line segment can be described in the following form according to the image coordinate frame (ICF) given in Figure 2.3:

$$u\cos\left(\theta_{\text{hough}}\right) + v\sin\left(\theta_{\text{hough}}\right) = r_{\text{hough}} \tag{2.25}$$

where $\theta_{\text{hough}}$ is the angle between the normal line from the origin to this line segment and the u-axis as in Figure 2.3 and $r_{\text{hough}}$ is the length of this normal line. In the figure, the u-axis and v-axis of the ICF represent the columns and the rows of the images, respectively.

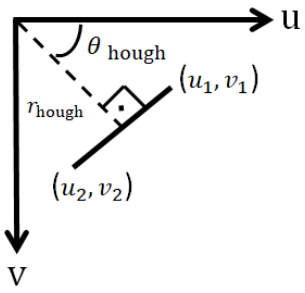



Figure 2.3: The line segment form performed in HT

For all the points of a line in the form of equation 2.25, $\theta_{\text{hough}}$ and $r_{\text{hough}}$ are constant.

HT first obtains the Hough parameter space $(r_{\text{hough}} - \theta_{\text{hough}})$ for each pixel point by different values of $\theta_{\text{hough}}$ with certain resolution in $[0, \pi)$. Next, the computed $r_{\text{hough}}$ values are quantized based on the predetermined quantization parameter of HT. After this procedure, certain number of common $r_{\text{hough}} - \theta_{\text{hough}}$ pairs greater than a threshold, $th_{\text{hough}}$, constitute a line by the corresponding pixel points.

As a result, the end point coordinates, $(u_1, v_1)$ and $(u_2, v_2)$ as illustrated in Figure 2.3, of each detected line via HT are utilized in the scope of the thesis.

### 2.2.8 Feature Extraction Methods

We have applied the following feature extraction methods for comparison with the proposed feature extraction method given in Section 3.2.2: Neighborhood AND operator (NAND) [44, 45], 1-Bit Transform (1-BT) [16], Symmetrical Local Thresholding (SLT) [19], Canny edge detector [46].

#### 2.2.8.1 Neighborhood AND Operator

The NAND operator first convolves a grayscale image $I_{\text{G}}$ via a horizontal and vertical Sobel kernel (see (2.17) and (2.18)) to get two convolved images $I_{\text{C,u}}$ and $I_{\text{C,v}}$, respectively. Then, the gradient magnitude image, $I_{\text{C,gm}}$, is computed as follows:

$$I_{\text{C,gm}} = \sqrt{I_{\text{C,u}}^2 + I_{\text{C,v}}^2} \tag{2.26}$$

Next, two binary images are acquired, $I_{\text{B1}}$ and $I_{\text{B2}}$, applying global thresholding given in equation 2.21 to the grayscale image and to the gradient magnitude image, respectively.

The binary images are then merged via the neighborhood AND operator [45] with the proximity parameter $k_{\text{NAND}}$.

$$I_{\text{B}}(i,j) = \begin{cases} 0 & \text{if } \sum_{m=i-k_{\text{NAND}}}^{i+k_{\text{NAND}}} I_{\text{B1}}(m,j) \cdot \sum_{m=i-k_{\text{NAND}}}^{i+k_{\text{NAND}}} I_{\text{B2}}(m,j) \\ 1 & \text{otherwise} \end{cases} \tag{2.27}$$

Equation 2.27 determines $I_{\text{B}}(i,j) = 0$ if all neighboring pixels with distance $k_{\text{NAND}}$ in column $j$ are zero for $I_{\text{B1}}$ and/or $I_{\text{B2}}$. Otherwise, $I_{\text{B}}(i,j) = 1$.

#### 2.2.8.2 1-Bit Transform

1-BT firstly obtains $I_{\text{C}}$ by convolving the $I_{\text{G}}$ with the $17 \times 17$ kernel $(\kappa_{\text{1-BT}})$ defined as:

$$\kappa_{\text{1-BT}}(i,j) = \begin{cases} 1/25 & \leftarrow \quad i,j \quad \in [0,4,8,12,16] \\ 0 & \leftarrow \quad i,j \quad \notin [0,4,8,12,16] \end{cases} \tag{2.28}$$

Hereby, $I_{\text{B}}$ is obtained using the fixed threshold value $th_{\text{f}}$:

$$I_{\text{B}}(i,j) = \begin{cases} 1 & \leftarrow \quad I_{\text{G}}(i,j) \geq I_{\text{C}}(i,j) + th_{\text{f}} \\ 0 & \leftarrow \quad I_{\text{G}}(i,j) < I_{\text{C}}(i,j) + th_{\text{f}} \end{cases} \tag{2.29}$$

#### 2.2.8.3 Symmetrical Local Thresholding

SLT processes each row of a grayscale image $I_{\text{G}}$ independently and computes the mean intensity values in a predefined range on the left ($I_{\text{G,l}}$) and right ($I_{\text{G,r}}$) hand side of each pixel. Then, $I_{\text{B}}$ is computed using the fixed threshold value $th_{\text{f}}$:

$$I_{\text{B}}(i,j) = \begin{cases} 1 & \leftarrow \text{ if } I_{\text{G}}(i,j) \geq I_{\text{G,l}}(i,j) + th_{\text{f}} \text{ and } I_{\text{G}}(i,j) \geq I_{\text{G,r}}(i,j) + th_{\text{f}} \\ 0 & \leftarrow \text{ otherwise} \end{cases} \tag{2.30}$$

#### 2.2.8.4 Canny Edge Detection

A Canny edge detector firstly smoothes the grayscale image $I_{\text{G}}$ via the discretized version [47] of continuous Gaussian filter given as follows:

$$\kappa_{\text{Gauss}}(u, v) = \frac{1}{2\pi\sigma_{\text{Gauss}}^2} \exp\left(-\frac{u^2 + v^2}{2\sigma_{\text{Gauss}}^2}\right) \tag{2.31}$$

where $\sigma_{\text{Gauss}}$ is the standard deviation of the continuous Gaussian filter. Then, $I_{\text{G}}$ is convolved via horizontal and vertical Sobel kernel to get $I_{\text{C,u}}$ and $I_{\text{C,v}}$. After that, the gradient magnitude image as in equation 2.26 and gradient direction image, $I_{\text{C,gd}}$, as in equation 2.32 are computed.

$$I_{\text{C,gd}} = \tan^{-1}\left(\frac{I_{\text{C,v}}}{I_{\text{C,u}}}\right) \tag{2.32}$$

After that, a non-maximum suppression is performed by checking for each pixel if it is a local maximum in its neighborhood in the direction of the gradient.
Lastly, the resultant image is thresholded with two thresholds ($T_1$ and $T_2$, where $T_1 < T_2$) to get two binary images ($I_{\text{T1}}$ and $I_{\text{T2}}$). The foreground pixels in $I_{\text{T2}}$ are considered as certain edges and if the foreground pixels in $I_{\text{T1}}$ has a connection with the edges in $I_{\text{T2}}$, they are also considered as edges. Thus, a binary image representing edges is formed.

### 2.2.9 Lane Models

#### 2.2.9.1 Second-Order Polynomials

Polynomials are one possible lane model [5]. In the scope of the thesis, we use the second-order polynomial lane model with the same $a$ coefficient for the left and right lanes:

$$\begin{aligned} a\, i_{\text{l}}^2 + b_{\text{l}}\, i_{\text{l}} + c_{\text{l}} &= j_{\text{l}} \\ a\, i_{\text{r}}^2 + b_{\text{r}}\, i_{\text{r}} + c_{\text{r}} &= j_{\text{r}} \end{aligned} \tag{2.33}$$

Here, $i_{\text{l}}$, $i_{\text{r}}$ and $j_{\text{l}}$, $j_{\text{r}}$ represent the row and column indexes of the left and right lane pixels on the BEV image, respectively. Hence, $(a, b_{\text{l}}, c_{\text{l}})$ and $(a, b_{\text{r}}, c_{\text{r}})$ are the left and right lane model coefficients.
The advantage of such lane model is the need for only five parameters in total for both

lanes, the direct use of the pixel coordinates and the parallelism constraint with the help of the same $a$ coefficient in a relaxed way since the $a$ coefficient is dominant for curvature of lanes as can be seen in equation 5.1.

#### 2.2.9.2 Clothoids and Arc-splines

Clothoids [48, 49, 50] are spiral curves, whose curvature $k_{\text{arc}}$ changes linearly with their arc-length $s_{\text{arc}}$:

$$k_{\text{arc}}(s_{\text{arc}}) = \frac{(k_{\text{arc,f}} - k_{\text{arc,i}})s_{\text{arc}}}{S_{\text{arc}}} \tag{2.34}$$

Hereby, $k_{\text{arc,i}}$ and $k_{\text{arc,f}}$ denote the initial and final curvature, respectively and $S_{\text{arc}}$ is the total arc-length of the curve. Two additional parameters ($P_{\text{arc,s}}, \psi_{\text{arc,s}}$) are needed to define a clothoid curve, where $P_{\text{arc,s}}$ and $\psi_{\text{arc,s}}$ are the starting point and initial tangent angle, respectively. Since clothoids allow for a smooth change of curvature, they are frequently used in road construction and are hence suitable for representing lanes. Nevertheless, a main disadvantage of clothoids is that they do not have an analytical representation, which makes computations with clothoids difficult [50, 51, 52].
As a remedy, it is possible to tightly approximate clothoids by arc-splines [51, 52]. An arc-spline consists of concatenated arc segments, whose radius linearly increases/decreases along the arc length $s_{\text{arc}}$. In addition to the clothoid parameters $P_{\text{arc,s}}$, $\psi_{\text{arc,s}}$, $k_{\text{arc,i}}$, $k_{\text{arc,f}}$, $S_{\text{arc}}$, an arc-spline is defined by the integer $n$, whereby $n+1$ is the number of arc segments. Specifically, the consecutive arcs of an arc-spline are represented as follows:

- Curvature of arc $j : (j = 0, \ldots, n) : k_{\text{arc},j} = k_{\text{arc},i} + j\frac{k_{\text{arc,f}} - k_{\text{arc,i}}}{n}$
- Length of arc $0$ and $n : S_{\text{arc},0} = S_{\text{arc},n} = \frac{S_{\text{arc}}}{2n}$
- Length of arc $j : (j = 1, \ldots, n-1) S_{\text{arc},j} = \frac{S_{arc}}{n}$
- Overall change in tangent angle : $\Delta\theta_{\text{arc}} = \frac{k_{\text{arc,f}} + k_{\text{arc,i}}}{2} S_{\text{arc}}$

The comparison of an arc-spline and the corresponding clothoid curve with $k_{\text{arc,i}} = 0, k_{\text{arc,f}} = 0.01, S_{\text{arc}} = 100$ is shown in Figure 2.4 for $n = 2$ (left) and $n = 5$ (right).

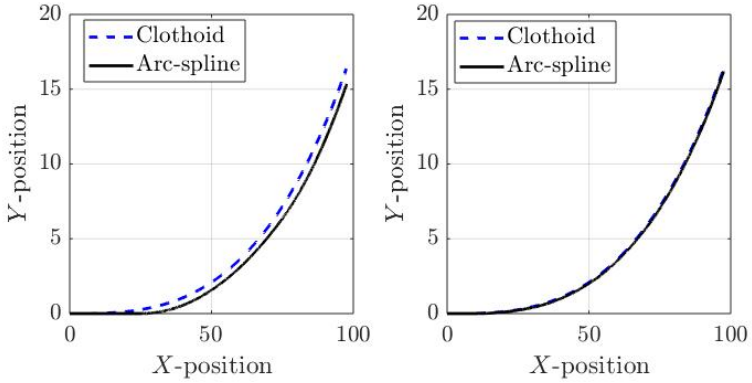



Figure 2.4: Clothoid and Arc-spline comparison

Arc-splines are utilized in Section 5.1 for reference trajectory generation in this thesis. There are two main advantages of using arc-splines as a reference trajectory. First, the arc segments can be represented analytically. Second, the offset curve of an arc-spline is again an arc-spline such that parallel lane markings can be easily represented by arc-splines with an offset in the arc radius. Moreover, it is the case that the approximation accuracy increases quadratically with the number $n$ of arc segments [51].

### 2.2.10 Kalman Filter

To estimate the lane model parameters, we perform Kalman Filter (KF). KF is an optimal estimator which minimizes the estimation error and operates on the linear stochastic systems in discrete state space model:

$$\begin{aligned} x_{k+1} &= A_k x_k + G_k w_k \\ y_k &= C_k x_k + F_k v_k \end{aligned} \tag{2.35}$$

A linear time invariant state space model is utilized in this thesis, $k$ subscripts are removed from $A_k, G_k, C_k, F_k$ as:

$$\begin{aligned} x_{k+1} &= A x_k + G w_k \\ y_k &= C x_k + F v_k \end{aligned} \tag{2.36}$$

The description of the system is given in Table 2.1. where $n_x$ and $n_y$ are the dimensions of the state and measurement vector defined in Table 2.1. The formulation of the

Table 2.1: Description of the system

| Variable | Description | Dimension |
|---|---|---|
| $x$ | State Vector | $n_x \times 1$ |
| $y$ | Measurement Vector | $n_y \times 1$ |
| $w$ | Process Noise Vector | $n_x \times 1$ |
| $f$ | Measurement Noise Vector | $n_y \times 1$ |
| $A$ | State Matrix | $n_x \times n_x$ |
| $G$ | Process Noise Matrix | $n_x \times n_w$ |
| $C$ | Output Matrix | $n_y \times n_x$ |
| $F$ | Measurement Noise Matrix | $n_y \times n_v$ |

KF is valid under the case that the process noise and measurement noise are Gaussian and all the samples of the noise are independent from each other.

KF basically consists of prediction and filtering steps. At the prediction step, KF predicts the next state based on the measurements up to current one. The formulation for the prediction step:

$$\begin{aligned} x_{k+1|k} &= Ax_{k|k} \\ \Sigma_{k+1|k} &= A\Sigma_{k|k}A^{\mathrm{T}} + GQG^{\mathrm{T}} \end{aligned} \tag{2.37}$$

At the filtering step, KF corrects the current step by the help of incoming measurement. The formulation for the filtering step:

$$\begin{aligned} x_{k+1|k+1} &= x_{k+1|k} + \Sigma_{k+1|k}C^{\mathrm{T}}(C\Sigma_{k|k}C^{\mathrm{T}} + FLF^{\mathrm{T}})^{-1}(y_{k+1} - Cx_{k+1|k}) \\ \Sigma_{k+1|k+1} &= \Sigma_{k+1|k} - \Sigma_{k+1|k}C^{\mathrm{T}}(C\Sigma_{k|k}C^{\mathrm{T}} + FLF^{\mathrm{T}})^{-1}C\Sigma_{k+1|k} \end{aligned} \tag{2.38}$$

KF formulations are as above with the initial step estimation:

$$\begin{aligned} x_{0|0} &= \Sigma_0 C^{\mathrm{T}}(C\Sigma_0 C^{\mathrm{T}} + FLF^{\mathrm{T}})^{-1}y_0 \\ \Sigma_{0|0} &= \Sigma_0 - \Sigma_0 C^{\mathrm{T}}(C\Sigma_0 C^{\mathrm{T}} + FLF^{\mathrm{T}})^{-1}C\Sigma_0 \end{aligned} \tag{2.39}$$

Here, $x_{k+1|k}$ represents the expectation value of $x$ at $(k+1)^{\text{th}}$ step given the observed measurements up to $k^{\text{th}}$ step and $\Sigma_{k+1|k}$ represents the covariance matrix of $x$ at the $(k+1)^{\text{th}}$ step given the observed measurements up to the $k^{\text{th}}$ step, $Q$ represents the covariance matrix of the process noise and $L$ represents the covariance matrix of the measurement noise.

In the case of using second-order polynomials with the same $a$ coefficients for lane modeling as given in Section 2.2.9.1, the state vector and state matrix are:

$$x = \begin{bmatrix} a \\ b_{\text{l}} \\ c_{\text{l}} \\ b_{\text{r}} \\ c_{\text{r}} \\ \Delta a \\ \Delta b_{\text{l}} \\ \Delta c_{\text{l}} \\ \Delta b_{\text{r}} \\ \Delta c_{\text{r}} \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.40}$$

where $\Delta a$, $\Delta b_{\text{l}}$, $\Delta c_{\text{l}}$, $\Delta b_{\text{r}}$, $\Delta c_{\text{r}}$ are estimated by taking difference of $a$, $b_{\text{l}}$, $c_{\text{l}}$, $b_{\text{r}}$, $c_{\text{r}}$ between the current and previous steps. Output matrix $C$ is defined as an identity matrix with appropriate dimension. The process and measurement noise are assumed to be Gaussian and independent from each other for all samples. Thus, the covariance matrices of process and measurement noise are diagonal and constant.

### 2.2.11 Camera Calibration

For further use in the scope of the thesis, the camera calibration process via a planar chekerboard as a calibration target is followed to estimate the intrinsic camera matrix and transformation matrix from the chekerboard coordinate frame (CHCF) to the camera coordinate frame (CCF). Whereas the intrinsic camera matrix ($K$) consists of focal length, principal point and skew coefficient parameters of the camera, the transformation matrix consists of rotation matrix ($R^{\text{c}}_{\text{ch}}$) and translation vector ($t^{\text{c}}_{\text{ch}}$) of the CHCF according to the CCF. When using the pinhole camera model [53], the pixel position of a point in the CCF is determined through the intrinsic camera matrix as follows:

$$\alpha \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K P^{\text{c}} \tag{2.41}$$

where $P^{c}$ is a point in the CCF, $\alpha$ is a scaling parameter, $u$ and $v$ are the column and row indexes of the point $P^{c}$ in the perspective image. The rotation matrix, $R^{c}_{ch}$, and translation vector, $t^{c}_{ch}$, show the relative orientation and position between CHCF and CCF. A point in CHCF is represented in CCF through the following relation:

$$P^{c} = R^{c}_{ch} P^{ch} + t^{c}_{ch} \tag{2.42}$$

where $P^{ch}$ is the point in CHCF. For illustration purposes, a schematic of the CCF and CHCF in a pose is given in Figure 2.5, where $[X^{c}, Y^{c}, Z^{c}]^{T}$ and $[X^{ch}, Y^{ch}, Z^{ch}]^{T}$ represent the coordinates of points according to the CCF and CHCF, respectively.

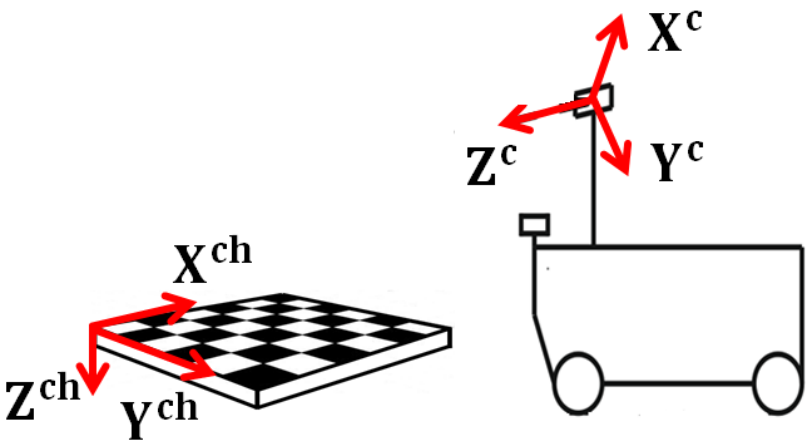



Figure 2.5: The schematic of the CCF and CHCF in a pose

Note that 3$^{rd}$ column of $R^{c}_{ch}$ is the normal vector of the chekerboard according to the CCF and $t^{c}_{ch}$ is the point in CCF showing the origin of CHCF.

To follow the camera calibration process, a planar chekerboard is utilized as a calibration target and should completely be captured from the camera in different poses. To have a unique solution for the intrinsic camera matrix and transformation matrix, we need at least three different poses in general [53]. Examples of the captured chekerboard images are given in Figure 2.6.

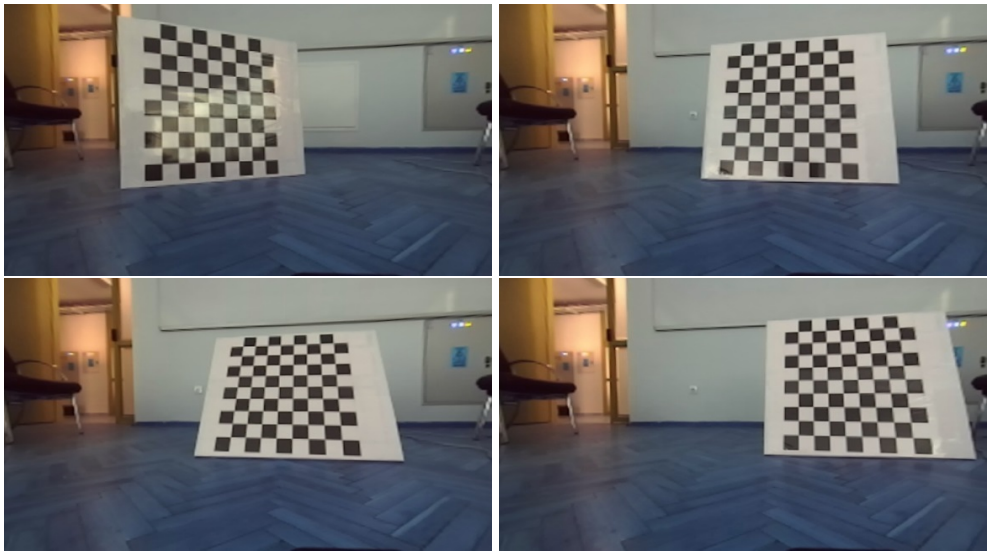


Figure 2.6: Captured checkerboard images for camera calibration process

In the camera calibration process, the Computer Vision System Toolbox of Matlab [53, 54, 55, 56] is utilized after capturing the checkerboard in different poses. One of the corner detection results, all estimated checkerboard poses and the mean reprojection errors per image of the calibration procedure are given in Figure 2.7, Figure 2.8 and Figure 2.9, respectively. For better illustration of the detected and reprojected corners, the image in Figure 2.7 is cropped.

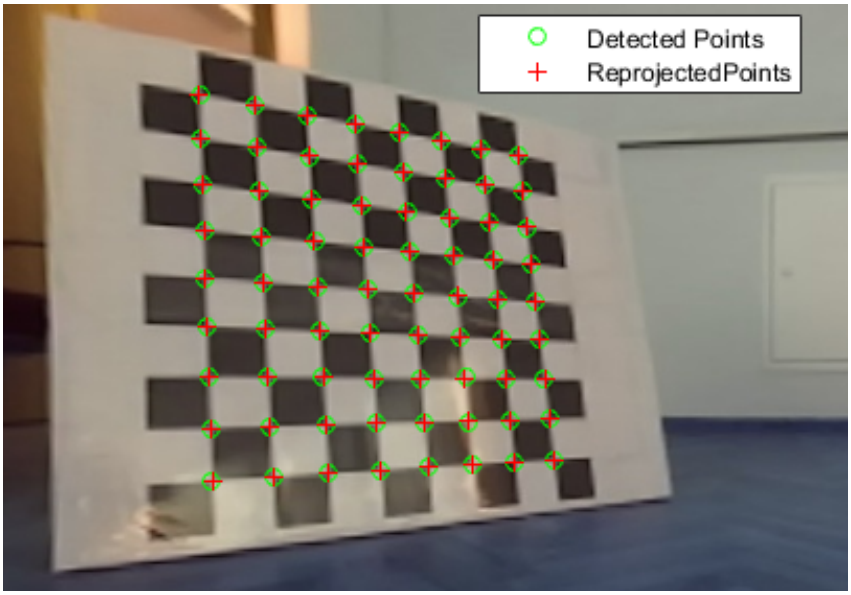



Figure 2.7: Corner detection result for camera calibration procedure

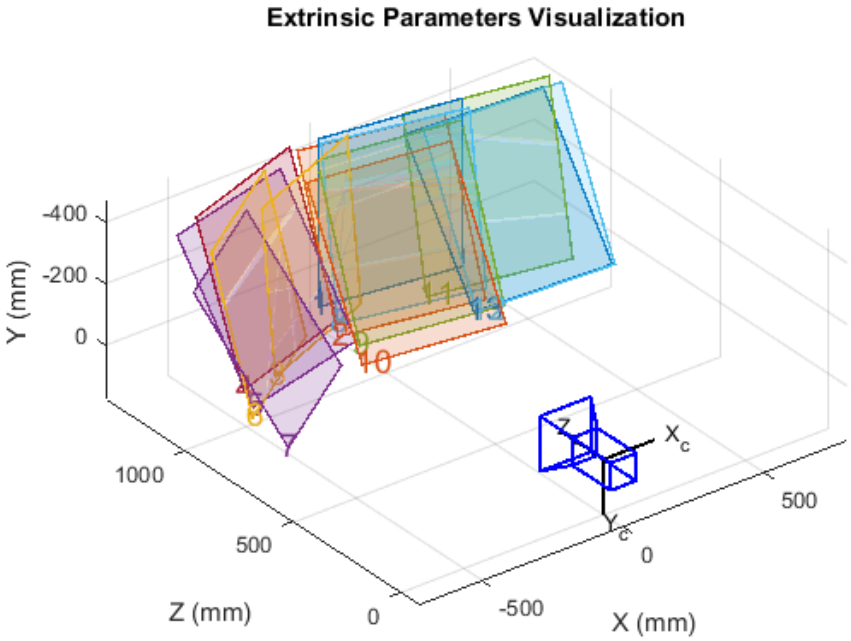



Figure 2.8: Poses of the chekerboard with respect to the camera

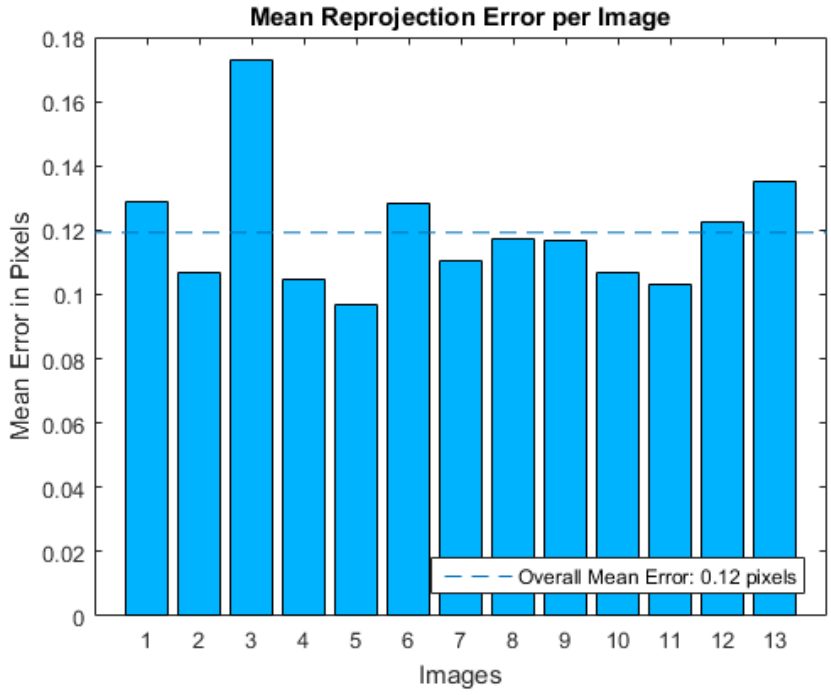



Figure 2.9: Mean reprojection error per image

## 2.3 LIDAR-based Background

This Section gives a detailed explanation of the necessary background for 2D LIDAR and its segmentation methods.

### 2.3.1 Definitions

A 2D LIDAR outputs the position of scanned points of the objects in its range in the polar coordinate frame as illustrated in Figure 2.10. Here, the 2D LIDAR is represented by the circle and the scanned object is represented by a rectangle, $\rho_i^{\mathrm{l}}$ is the distance value and $\theta_i^{\mathrm{l}}$ is the angle value of the $i^{\text{th}}$ LIDAR point $p_i^{\mathrm{l}}$. That is, $p_i^{\mathrm{l}}$ is defined as follows:

$$p_i^{\mathrm{l}} = \left\{\rho_i^{\mathrm{l}}, \theta_i^{\mathrm{l}}\right\} \quad \text{for} \quad i = 1, \ldots, N_{\mathrm{l}} \tag{2.43}$$

where $N_{\mathrm{l}}$ is the number of LIDAR points. $N_{\mathrm{l}}$ is determined by the angle resolution $(\Delta\theta^{\mathrm{l}})$ of the 2D LIDAR. Using $N_{\mathrm{l}} = 360°/(\Delta\theta^{\mathrm{l}})$, the LIDAR angle set $\theta^{\mathrm{l}}$ in degrees is given as

$$\theta^{\mathrm{l}} = \left\{0°, \Delta\theta^{\mathrm{l}}, 2\Delta\theta^{\mathrm{l}}, \ldots, 360° - \Delta\theta^{\mathrm{l}}\right\} \tag{2.44}$$

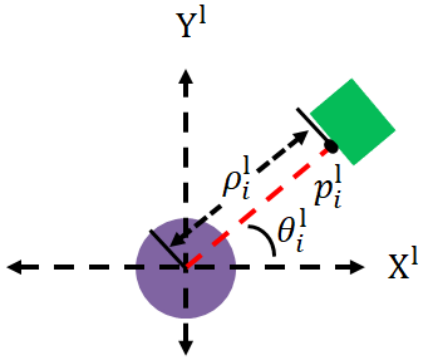



Figure 2.10: 2D LIDAR output format

Each LIDAR point $p_i^{\mathrm{l}}$ can as well be represented in LIDAR coordinate frame (LCF)

given in 2.10 as follows, where $\mathrm{X^l}$ and $\mathrm{Y^l}$ represent the x-axis and y-axis of LCF:

$$\begin{aligned} x_i^{\mathrm{l}} &= \rho_i^{\mathrm{l}} \cos\theta_i^{\mathrm{l}} \quad \text{for} \quad i = 1, \ldots, N_{\mathrm{l}} \\ y_i^{\mathrm{l}} &= \rho_i^{\mathrm{l}} \sin\theta_i^{\mathrm{l}} \quad \text{for} \quad i = 1, \ldots, N_{\mathrm{l}} \\ z_i^{\mathrm{l}} &= 0 \quad \text{for} \quad i = 1, \ldots, N_{\mathrm{l}} \\ p_i^{\mathrm{l}} &= [x_i^{\mathrm{l}}, y_i^{\mathrm{l}}, z_i^{\mathrm{l}}]^{\mathrm{T}} \quad \text{for} \quad i = 1, \ldots, N_{\mathrm{l}} \end{aligned} \tag{2.45}$$

Without loss of generality, $z_i^{\mathrm{l}} = 0$ in the scope of the thesis.

### 2.3.2 2D LIDAR Segmentation Methods

2D LIDARs are utilized for object detection [57, 58] in the scope of this thesis. When detecting objects, segmentation of the 2D LIDAR data needs to be performed in order to group LIDAR points and associate them to the objects in the LIDAR range. The methods, fixed breakpoint detection (FBD) [59, 60], adaptive breakpoint detection (ABD) [61], nearest neighbor (NN) [62], $k$ nearest neighbor ($k$NN) [62], radially bounded nearest neighbor (RBNN) [62] and distance varying radially bounded nearest neighbor (dvRBNN) [63], are highly used for 2D LIDAR data segmentation in the literature. The necessary background for the methods is given from Section 2.3.2.1 to Section 2.3.2.3

#### 2.3.2.1 FBD and ABD

FBD and ABD identify abrupt changes in the sequence of LIDAR data through fixed ($th_{\mathrm{f}}^{\mathrm{l}}$) and adaptive ($th_{\mathrm{a}}^{\mathrm{l}}$) thresholds. Whereas the fixed threshold in FBD has to be selected manually, the $i^{\text{th}}$ adaptive threshold $th_{\mathrm{a},i}^{\mathrm{l}}$ in ABD is computed for each LIDAR point except for the last one as:

$$th_{\mathrm{a},i}^{\mathrm{l}} = \rho_i^{\mathrm{l}} \frac{\sin(\Delta\theta^{\mathrm{l}})}{\sin(\lambda - \Delta\theta^{\mathrm{l}})} + 3\sigma_{\mathrm{r}} \tag{2.46}$$

where $\sigma_{\mathrm{r}}$ represents the distance resolution of the LIDAR and $\lambda$ is the worst case incidence angle. As the incidence angle decreases, the corresponding threshold value increases. Then, segmentation with FBD and ABD is defined as follows:

$$p_{i+1}^{\mathrm{l}} \in \begin{cases} \Xi_h^{\mathrm{l}} & \text{if} \quad d_{\mathrm{E}}(p_i^{\mathrm{l}}, p_{i+1}^{\mathrm{l}}) < th_{\mathrm{f}}^{\mathrm{l}}(\text{FBD}), th_{\mathrm{a},i}^{\mathrm{l}}(\text{ABD}) \\ \Xi_{h+1}^{\mathrm{l}} & \text{otherwise} \end{cases} \tag{2.47}$$

where $\Xi_h^{\mathrm{l}}$ represents the $h^{\text{th}}$ set of segmented data.

#### 2.3.2.2 NN and kNN

NN, $k$NN apply graph-based segmentation. A graph is a pair of sets $(n^{\mathrm{l}}, E)$, where $n^{\mathrm{l}}$ is the set of nodes and $E$ is the set of edges connecting the nodes. Here, nodes represent LIDAR points and an edge $(e)$ connnecting the $i^{\text{th}}$ and $j^{\text{th}}$ node is defined as $e = \left\{n_i^{\mathrm{l}}, n_j^{\mathrm{l}}, d_{\mathrm{E}}(p_i^{\mathrm{l}}, p_j^{\mathrm{l}})\right\}$.
NN and $k$NN connect every node to the nearest neighbor and $k_{\mathrm{NN}}$ number of nearest neighbors, respectively. The set of edges of NN ($E_{\mathrm{NN}}$) and $k$NN ($E_{k\mathrm{NN}}$) is defined in this thesis as:

$$\begin{aligned} E_{\mathrm{NN}} &= \left\{\left\{n_i^{\mathrm{l}}, n_j^{\mathrm{l}}, d_{\mathrm{E}}(p_i^{\mathrm{l}}, p_j^{\mathrm{l}})\right\} \mid \; n_j^{\mathrm{l}} \in \mathrm{NN}(n_i^{\mathrm{l}})\right\}, \quad \forall n_i^{\mathrm{l}} \in n^{\mathrm{l}} \\ E_{k\mathrm{NN}} &= \left\{\left\{n_i^{\mathrm{l}}, n_j^{\mathrm{l}}, d_{\mathrm{E}}(p_i^{\mathrm{l}}, p_j^{\mathrm{l}})\right\} \mid \; n_j^{\mathrm{l}} \in k\mathrm{NN}(n_i^{\mathrm{l}})\right\}, \quad \forall n_i^{\mathrm{l}} \in n^{\mathrm{l}} \end{aligned} \tag{2.48}$$

where, $\mathrm{NN}(n_i^{\mathrm{l}})$ and $k\mathrm{NN}(n_i^{\mathrm{l}})$ represent the nearest and $k_{\mathrm{NN}}$ number of nearest neighbor(s) of $n_i^{\mathrm{l}}$ using the Euclidean distance given in equation 2.5. Applying equation 2.48, all nodes, which are connected to one another via a chain of edges, belong to the same segmented group.

#### 2.3.2.3 RBNN and dvRBNN

RBNN and dvRBNN connect every node to the neighbors which are inside a fixed radius, ($th_{\mathrm{f}}^{\mathrm{l}}$), and distance varying radius, ($th_{\mathrm{a}}^{\mathrm{l}}$), respectively. While $th_{\mathrm{f}}^{\mathrm{l}}$ in RBNN has to be chosen manually, the $i^{\text{th}}$ distance varying radius, $th_{\mathrm{a},i}^{\mathrm{l}}$, in dvRBNN is computed for each LIDAR point as:

$$th_{\mathrm{a},i}^{\mathrm{l}} = th_{\mathrm{scale}} \rho_i^{\mathrm{l}} \tan\left(\Delta\theta^{\mathrm{l}}\right) + 2\sigma_{\mathrm{r}} \tag{2.49}$$

with the scaling parameter $th_{\mathrm{scale}}$ of the distance value of the 2D LIDAR. The set of edges of RBNN ($E_{\mathrm{RBNN}}$) and dvRBNN ($E_{\mathrm{dvRBNN}}$) are defined as:

$$\begin{aligned} E_{\mathrm{RBNN}} &= \left\{\left\{n_i^{\mathrm{l}}, n_j^{\mathrm{l}}, d_{\mathrm{E}}(p_i^{\mathrm{l}}, p_j^{\mathrm{l}})\right\} \mid \; d_{\mathrm{E}}(p_i^{\mathrm{l}}, p_j^{\mathrm{l}}) < th_{\mathrm{f}}^{\mathrm{l}}\right\}, \quad \forall n_i^{\mathrm{l}} \in n^{\mathrm{l}} \\ E_{\mathrm{dvRBNN}} &= \left\{\left\{n_i^{\mathrm{l}}, n_j^{\mathrm{l}}, d_{\mathrm{E}}(p_i^{\mathrm{l}}, p_j^{\mathrm{l}})\right\} \mid \; d_{\mathrm{E}}(p_i^{\mathrm{l}}, p_j^{\mathrm{l}}) < th_{\mathrm{a},i}^{\mathrm{l}}\right\}, \quad \forall n_i^{\mathrm{l}} \in n^{\mathrm{l}} \end{aligned} \tag{2.50}$$

Applying equation 2.50, all nodes, which are connected to one another via a chain of edges, belong to the same segmented group.

# CHAPTER 3

# LANE DETECTION AND TRACKING

In this Chapter, we first formalize and explain the proposed lane detection and tracking method step by step. After that, the proposed method is evaluated computationally and experimentally on the real and generated synthetic images. Besides, illustrative lane detection and tracking results are given. Hereby, we assume that input images are given in the RGB format for the proposed method.

## 3.1 Overview of the Proposed Method

The proposed method includes five main steps as given in Figure 3.1.

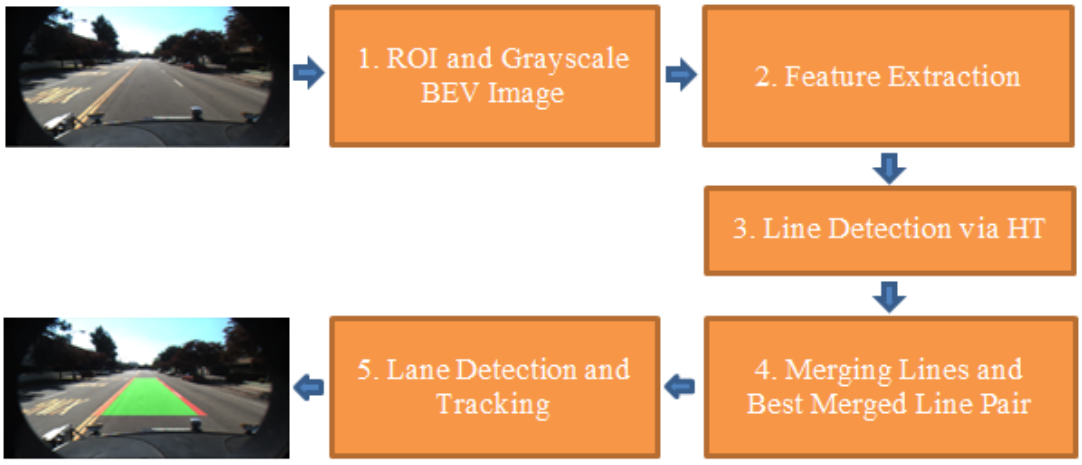



Figure 3.1: Overview of the proposed lane detection and tracking method

First, RGB image is obtained and converted to grayscale image. Then, a fixed region of interest (ROI) is determined on the grayscale image and the ROI is transformed to

grayscale BEV image via inverse perspective mapping (IPM). Second, candidate lane pixels are extracted on the grayscale BEV image through the proposed feature extraction method. Third, the Hough transform (HT) is applied to the candidate lane pixels to detect lines. Fourth, the lines are merged based on the following features: slope and distance of the interception points of the detected lines at the top and bottom border of the image frame. Furthermore, the best lane pair representative is determined over all the lines through the proposed cost function. Fifth, lane model parameters are computed by utilizing the lines of the best pair through the least squares method (LSM) and tracked via Kalman Filter (KF).

## 3.2 Detailed Steps of the Proposed Method

### 3.2.1 Step-1

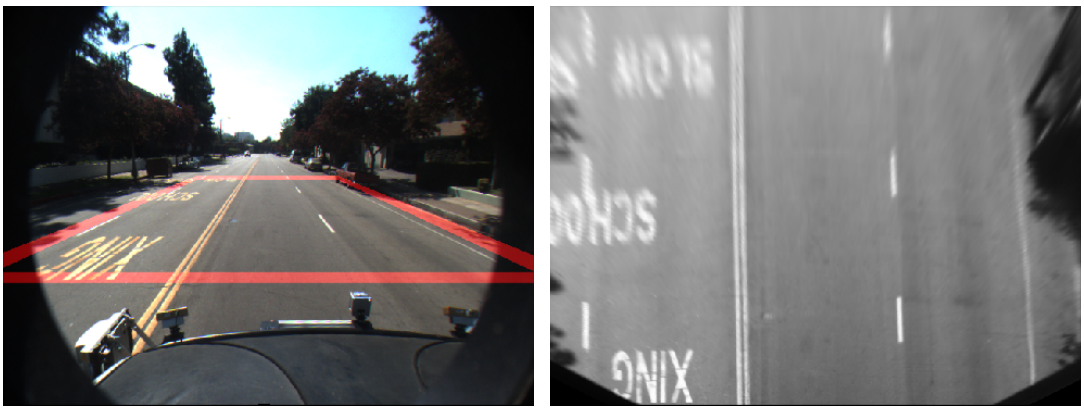
In the first step, the obtained RGB image ($I_{\mathrm{RGB}}$) [20] given in Figure 3.2 (a) is converted to grayscale image ($I_{\mathrm{G}}$) given in Figure 3.2 (b) by using the equation 2.20.

(a) (b)

Figure 3.2: RGB image (a) and grayscale image (b)

Furthermore, a fixed ROI on the $I_{\mathrm{G}}$ is determined to focus on the lanes and reduce the image processing time. In Figure 3.3 (a), the red trapezoid area given on the $I_{\mathrm{RGB}}$ for illustration purposes represents the fixed ROI in the scope of thesis. Next, we transform the specified ROI on the $I_{\mathrm{G}}$ to grayscale BEV image ($I_{\mathrm{G,BEV}}$) via IPM

described in Section 2.2.6. In Figure 3.3 (b), the $I_{\text{G,BEV}}$ obtained from the $I_{\text{G}}$ with the fixed ROI is shown.



(a) (b)

Figure 3.3: The fixed ROI (a) and grayscale BEV image (b)

### 3.2.2 Step-2

In this step, we extract candidate lane pixels on the $I_{\text{G,BEV}}$ by performing 1D top-hat kernels and histograms.

Here, we first convolve the $I_{\text{G,BEV}}$ as in equation 2.12 via 1D top-hat kernel to get convolved image ($I_{\text{C}}$), an example of a top-hat filter is given in Section 2.2.2. Note that the kernel should be tuned for lane marking width to enhance the lanes.

Next, the $I_{\text{C}}$ is horizontally divided into a predetermined number of region, $N_{\text{I}_\text{C}}$, and the histogram of each region is computed, $hist_i, i = 1, \cdots, N_{\text{I}_\text{C}}$ as explained in Section 2.2.5.

Next, local maxima which are greater than a predetermined threshold value $th_{\text{FE}}$ in each histogram plot are determined and then all the determined local maxima are sorted. Lastly, the highest $p_{\text{FE}}\%$ local maxima are specified as candidate lane pixels. That is, feature extraction is realized and binary image ($I_{\text{B}}$) is obtained. The $I_{\text{B}}$ obtained from the $I_{\text{G,BEV}}$ in Figure 3.3 (b) is given in Figure 3.4, in the figure white pixels represent the candidate lane pixels and the black pixels represent the background evaluated as the object pixels other than the lane pixels.

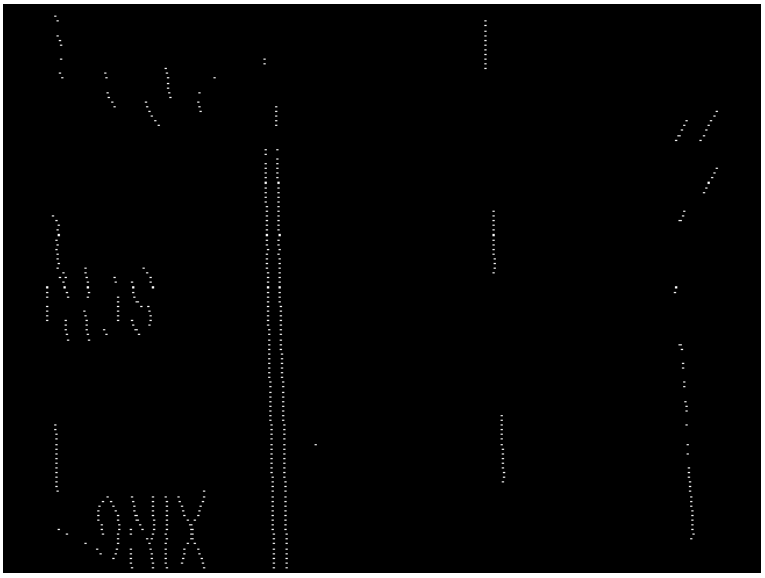


Figure 3.4: Obtained binary image, $I_{\mathrm{B}}$

With this feature extraction approach, 1D top-hat kernel enhances the lanes and the enhanced lane pixels supports each other for the local maxima in the close vicinity of the histograms of the horizontally divided convolved subimages. Since the local maxima constituted from a number of lane pixels are distinctive from the noisy local maxima which are generally constituted from shadows, cracks on the roads, etc. as they have unregular shapes, we can eliminate the noisy ones easily through low threshold values. Additionally, adaptivity to the different lighting conditions is provided since the local maxima which are greater than a low threshold value are sorted and the highest ones are counted as candidate lane pixels. Lastly, low number of candidate lane pixels are extracted as they are determined by the local maxima of the histograms.

### 3.2.3 Step-3

In this step, we perform HT as explained in Section 2.2.7 on the obtained $I_{\mathrm{B}}$ to detect the lines, $l_i, i = 1, \cdots, N_{\text{line}}$ as illustrated in Figure 3.5 since HT is robust to noise and allows gaps between pixels. Here, $l_i$ is the $i^{\text{th}}$ line and $N_{\text{line}}$ is the number of the lines. Thus, we obtain the end points of each line, $(u_{i1}, v_{i1})$ and $(u_{i2}, v_{i2}), i = 1, \cdots, N_{\text{line}}$ in the ICF defined in Section 2.2.7. Note that the extracted low number of candidate

lane pixels in Section 3.2.2 greatly reduces the computational complexity of the HT.

Figure 3.5: Line detection via HT on the $I_{\mathrm{B}}$

### 3.2.4 Step-4

After detecting the lines, the $i^{\text{th}}$ features (lengths, slopes and the interception points at the top and bottom border image) of the detected lines are computed as:

$$\zeta_i = \sqrt{(u_{i2} - u_{i1})^2 + (v_{i2} - v_{i1})^2} \tag{3.1}$$

$$s_i = \frac{u_{i2} - u_{i1}}{v_{i2} - v_{i1}} \tag{3.2}$$

$$\eta_i = s_i(N_{\text{row}} - v_{i2}) + u_{i2} \tag{3.3}$$

$$\mu_i = -s_i v_{i2} + u_{i2} \tag{3.4}$$

Here, $\zeta_i$, $s_i$, $N_{\text{row}}$, $\eta_i$ and $\mu_i$ represent the length and slope of the $i^{\text{th}}$ line, number of the rows in the image, interception points between the $i^{\text{th}}$ line and the bottom and top border of the image, respectively.
Furthermore, the lines $l_i$ and $l_j$, $i = 1$ and $j = 1, \cdots, N_{\text{line}}, i \neq j$ are merged if:

$$|s_i - s_j| < th_{\text{s}} \text{ and } |\eta_i - \eta_j| < th_{\text{intMerge}} \text{ and } |\mu_i - \mu_j| < th_{\text{intMerge}}$$

where $th_{\text{s}}$, $th_{\text{intMerge}}$ are the threshold values for allowable maximum distances between the slope and interception point features of each line, respectively. With this procedure, the merged lines using the detected lines in Figure 3.5 are illustrated in the circles in Figure 3.6.

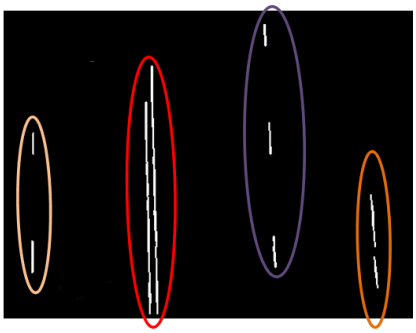


Figure 3.6: Merged lines using the detected lines in Figure 3.5

With the merging process, we obtain $M_{\mathrm{C}}$ number of line clusters ($C_i,\ \ i = 1, \cdots, M_{\mathrm{C}}$), where $M_{\mathrm{C}} \leq N_{\mathrm{line}}$ and each cluster contains the merged lines. Then, the features of each cluster are computed as follows:

$$\zeta_{\mathrm{c}i} = \sum_{k \in C_i} \zeta_k \qquad i = 1, \cdots, M_{\mathrm{C}} \tag{3.5}$$

$$s_{\mathrm{c}i} = \frac{\sum_{k \in C_i} s_k}{N_{\mathrm{C}i}} \qquad i = 1, \cdots, M_{\mathrm{C}} \tag{3.6}$$

$$\eta_{\mathrm{c}i} = \frac{\sum_{k \in C_i} \eta_k}{N_{\mathrm{C}i}} \qquad i = 1, \cdots, M_{\mathrm{C}} \tag{3.7}$$

$$\mu_{\mathrm{c}i} = \frac{\sum_{k \in C_i} \mu_k}{N_{\mathrm{C}i}} \qquad i = 1, \cdots, M_{\mathrm{C}} \tag{3.8}$$

Here, $\zeta_{\mathrm{c}i}$ and $s_{\mathrm{c}i}$ represent the summation of the lines and the mean of the slopes of the lines in the $i^{\mathrm{th}}$ cluster, respectively. The $\eta_{\mathrm{c}i}$ and $\mu_{\mathrm{c}i}$ are the mean of the interception points between the lines in the $i^{\mathrm{th}}$ cluster and the bottom and top border of the image, respectively. The $N_{\mathrm{C}i}$ represents the number of lines in the $i^{\mathrm{th}}$ cluster.

Thereafter, we compute the following cost function in equation 3.9 for each cluster pair, $C_i$ and $C_j$, $i = 1$ and $j = 1, \cdots, M_{\mathrm{C}}, i \neq j$, to determine the pair giving the maximum nonzero value.

$$\begin{aligned}\chi(C_i, C_j) = (\zeta_{\mathrm{c}i} + \zeta_{\mathrm{c}j} + w_{\mathrm{s}} \exp\left(-(s_{\mathrm{c}i} - s_{\mathrm{c}j})^2\right)) \cdot if(|\eta_{\mathrm{c}i} - \eta_{\mathrm{c}j}| > th_{\mathrm{intMin}}) \cdot \\ if(|\eta_{\mathrm{c}i} - \eta_{\mathrm{c}j}| < th_{\mathrm{intMax}}) \cdot if(|\mu_{\mathrm{c}i} - \mu_{\mathrm{c}j}| > th_{\mathrm{intMin}}) \cdot if(|\mu_{\mathrm{c}i} - \mu_{\mathrm{c}j}| < th_{\mathrm{intMax}}) \cdot \\ \exp\left(-(N_{\mathrm{midColumn}} - \frac{\eta_{\mathrm{c}i} + \eta_{\mathrm{c}j} + \mu_{\mathrm{c}i} + \mu_{\mathrm{c}j}}{4})^2 / 2(\frac{\sigma_{\mathrm{mid}}}{3})^2\right)\end{aligned} \tag{3.9}$$

where $w_{\text{s}}$ is the importance weighting between slope difference and the total length of the pairs, $N_{\text{midColumn}}$ is the middle column index in the u-axis of the ICF defined in Section 2.2.7, $\sigma_{\text{mid}}$ represents allowable deviation of the mean of the interception points from the $N_{\text{midColumn}}$, $th_{\text{intMin}}$ and $th_{\text{intMax}}$ are the allowable minimum and maximum threshold values for the interception point distances between the pairs, respectively. The $if(\text{condition})$ function outputs zero if the condition is false or one if the condition is true.

The cost function generates high values for the cluster pair whose mean of the interception points are close to the $N_{\text{midColumn}}$, which are long in total length and which have similar slopes if the interception points of the clusters are in allowable distance. The cluster pair giving the maximum nonzero value are called as the best pair from this point on and it should represent the current left (left cluster of the pair) and right (right cluster of the pair) lanes with the assumption that the CCF is placed above the center of the gravity of the vehicle as described in Section 5.2. As a result, we obtain the end points of the left $(u_{i1,\text{l}}, v_{i1,\text{l}})$, $(u_{i2,\text{l}}, v_{i2,\text{l}}), i = 1, \cdots, n_{\text{l}}$ and right $(u_{i1,\text{r}}, v_{i1,\text{r}})$, $(u_{i2,\text{r}}, v_{i2,\text{r}}), i = 1, \cdots, n_{\text{r}}$ lines in the clusters of the best pair, where $n_{\text{l}}$ and $n_r$ represent the number of lines in the left and right cluster of the best pair, respectively. In Figure 3.7, the left and right line clusters of the best pair chosen according to the cost function are shown in the red and purple circles, respectively.

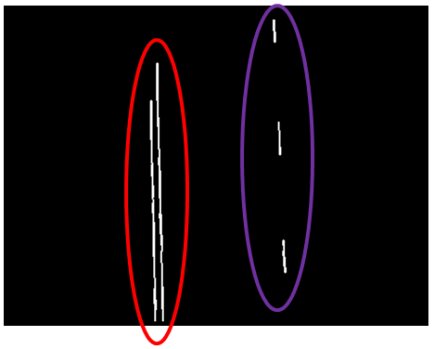
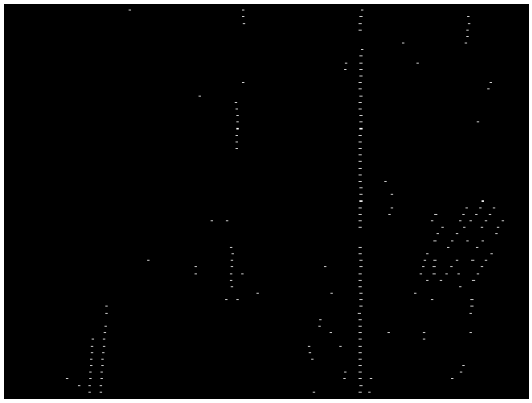


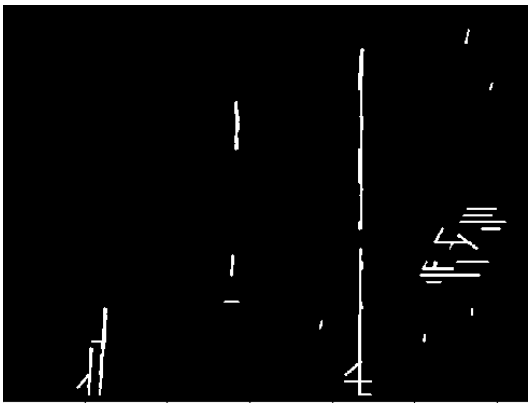
Figure 3.7: The best cluster pair representing the current left and right lanes

To show the effectiveness of the method from obtaining binary images to the best pair, Figure 3.8 - 3.13 (a), (b), (c), (d), (e) which show the $I_{\text{RGB}}$ in CalTech database [20], $I_{\text{B}}$, hough lines, some merged hough lines in the circles and the chosen best pair are given, respectively.

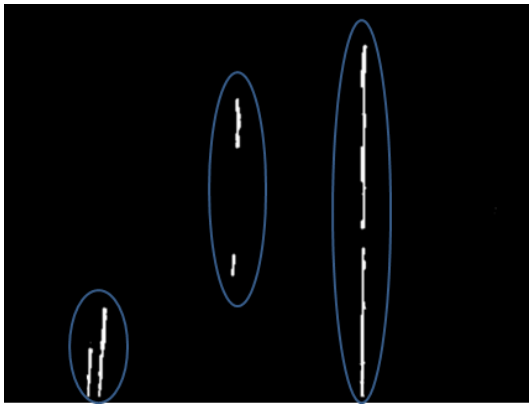
(a)



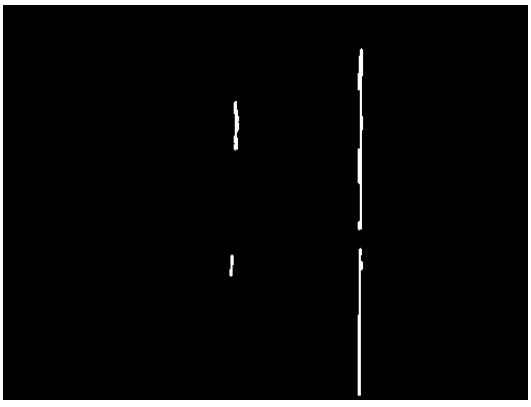
(b)



(c)



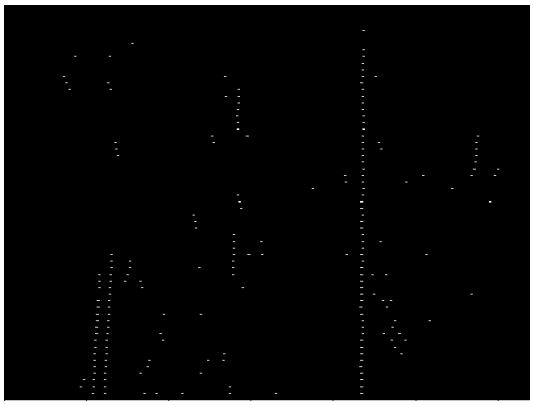
(d)



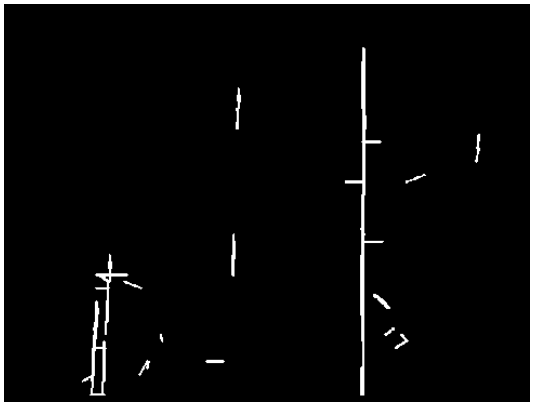
(e)

Figure 3.8: The illustrated process of obtaining the best pair

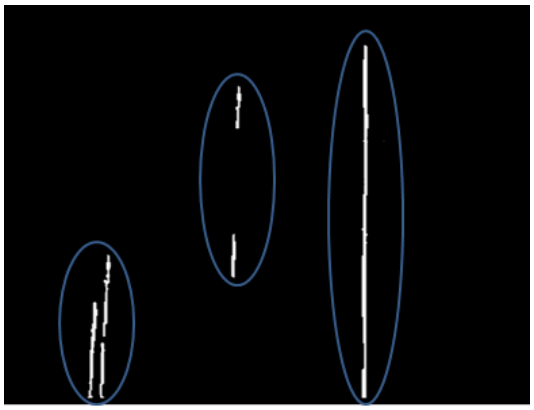
(a)



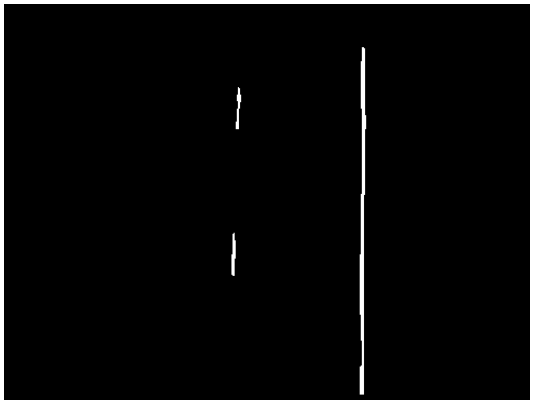
(b)



(c)



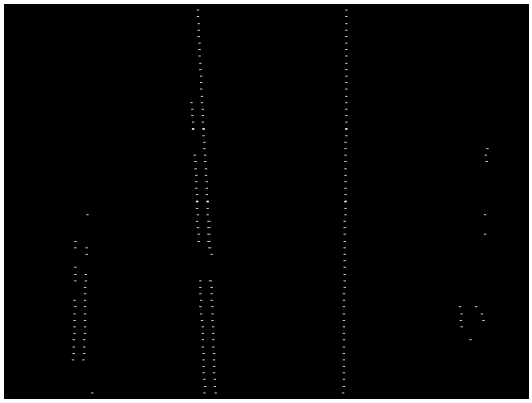
(d)



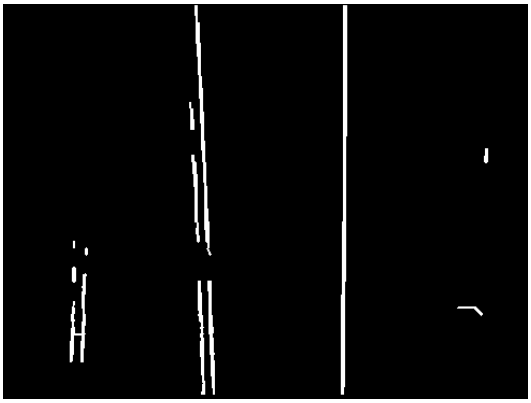
(e)

Figure 3.9: The illustrated process of obtaining the best pair

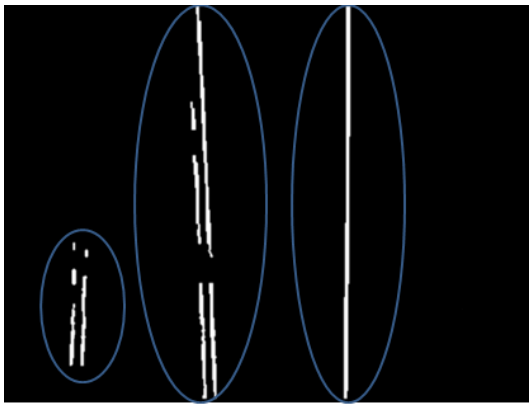
(a)



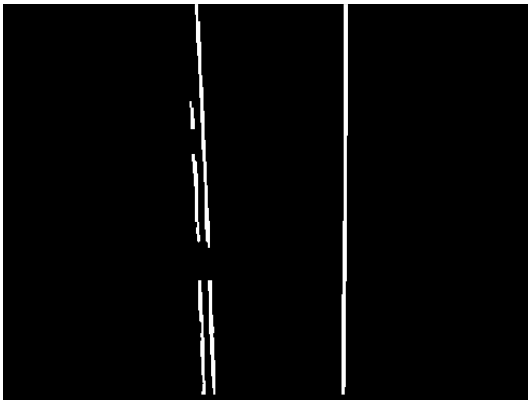
(b)



(c)



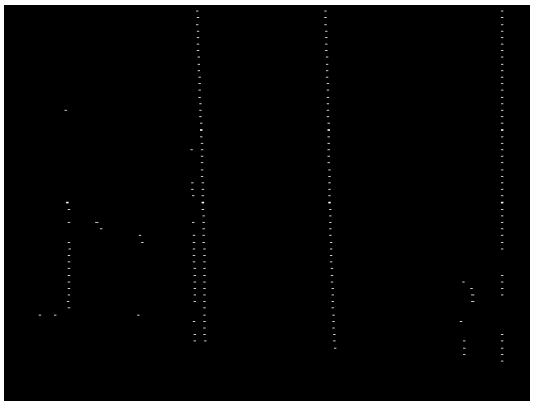
(d)



(e)

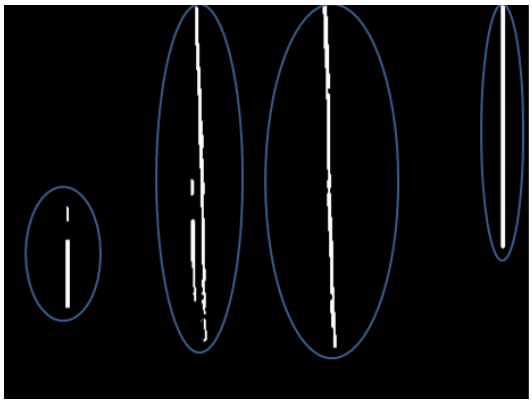
Figure 3.10: The illustrated process of obtaining the best pair

(a)



(b)

(c)



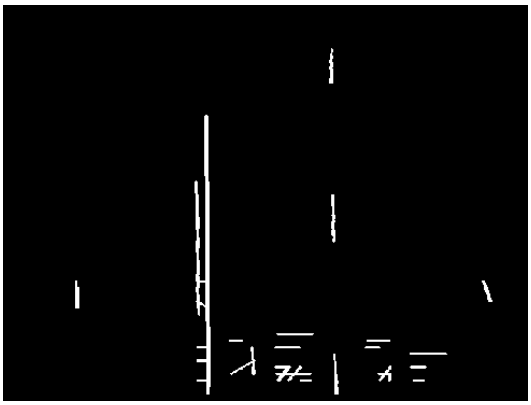
(d)

(e)

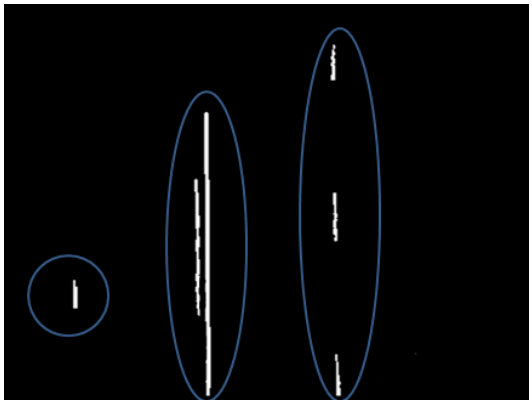
Figure 3.11: The illustrated process of obtaining the best pair


(a)

(b)



(c)



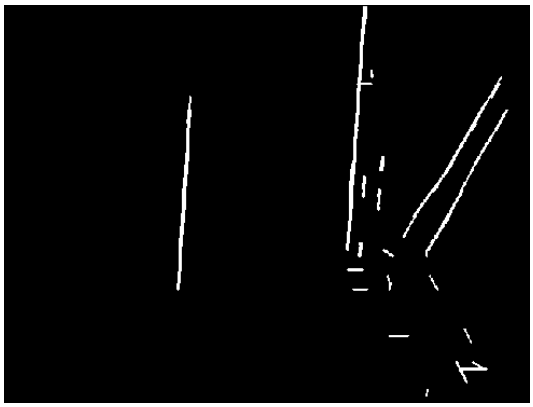
(d)

(e)

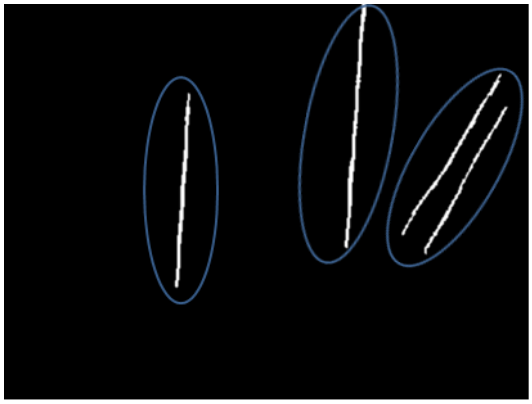
Figure 3.12: The illustrated process of obtaining the best pair

(a)

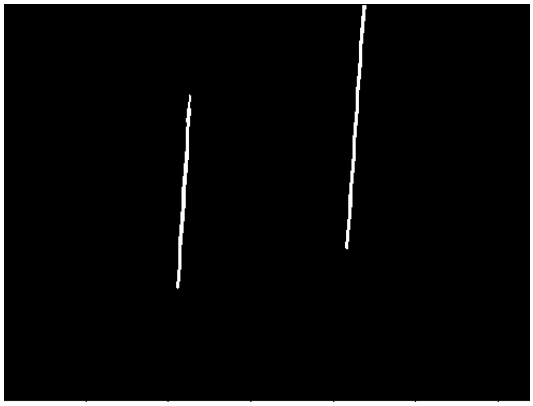
(b)



(c)



(d)



(e)

Figure 3.13: The illustrated process of obtaining the best pair

As can be seen from the Figures 3.8 and 3.9, correct line cluster pairs are extracted in the case where lanes are affected by extreme shadows and close to the other vehicles. Moreover, the algorithm correctly extracts line cluster pair in Figure 3.10 where there exist road signs close to the lanes. In addition to that, accurate line cluster pair are extracted in Figure 3.11, 3.12 and 3.13 where the texture of road changes, there exist letters on the road, curbs, traffic lights close to the lanes.

### 3.2.5 Step-5

In this step, we detect and track the lane model parameters. As a lane model, we utilize second-order polynomial with the same $a$ coefficient given in Section 2.2.9.1. The model parameters are computed through the LSM described in Section 2.1.1 by using the end points of the lines of the best cluster pair. In this case, the data matrix ($A_{\text{LSM}}$), model parameter vector ($x_{\text{LSM}}$) and model measurement vector ($y_{\text{LSM}}$) defined in Section 2.1.1 become:

$$A_{\text{LSM}} = \begin{bmatrix} u_{11,\text{l}}^2 & u_{11,\text{l}} & 1 & 0 & 0 \\ u_{12,\text{l}}^2 & u_{12,\text{l}} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{n_\text{l}1,\text{l}}^2 & u_{n_\text{l}1,\text{l}} & 1 & 0 & 0 \\ u_{n_\text{l}2,\text{l}}^2 & u_{n_\text{l}2,\text{l}} & 1 & 0 & 0 \\ u_{11,\text{r}}^2 & 0 & 0 & u_{11,\text{r}} & 1 \\ u_{12,\text{r}}^2 & 0 & 0 & u_{12,\text{r}} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{n_\text{r}1,\text{r}}^2 & 0 & 0 & u_{n_\text{r}1,\text{r}} & 1 \\ u_{n_\text{r}2,\text{r}}^2 & 0 & 0 & u_{n_\text{r}2,\text{r}} & 1 \end{bmatrix}, x_{\text{LSM}} = \begin{bmatrix} a \\ b_\text{l} \\ c_\text{l} \\ b_\text{r} \\ c_\text{r} \end{bmatrix}, y_{\text{LSM}} = \begin{bmatrix} v_{11,\text{l}} \\ v_{12,\text{l}} \\ \vdots \\ v_{n_\text{l}1,\text{l}} \\ v_{n_\text{l}2,\text{l}} \\ v_{11,\text{r}} \\ v_{12,\text{r}} \\ \vdots \\ v_{n_\text{r}1,\text{r}} \\ v_{n_\text{r}2,\text{r}} \end{bmatrix} \tag{3.10}$$

After computing the lane model parameter vector through LSM as described in Section 2.1.1 with the matrices and vectors given above, we track them via KF as in the given example in Section 2.2.10. The lane detection and tracking result through the proposed method on the $I_{\text{RGB}}$ in Figure 3.2 (a) is given Figure 3.14.

Figure 3.14: The lane detection and tracking result through the proposed method

## 3.3 Computational Evaluation

In this Section, quantitative and illustrative results of the proposed method are presented.

To evaluate the proposed method quantitatively, CalTech - Cordova1 database [20] and synthetic data generated from this database are utilized.

### 3.3.1 Synthetic Data Generation

Synthetic data are generated by manipulating the images of the Cordova $1$ dataset [20] to evaluate the proposed method quantitatively under the effect of noise and in different lighting conditions.

To generate synthetic images with noise, Gaussian noise images ($I_{\text{Gauss}}$s) with zero mean and $0.01$ variance are added to the images ($I_{\text{RGB}}$s) of the database. As a result, images with Gaussian noise which have a specific variance are generated without changing the brightness of the images for the evaluation of the algorithm under the effect of noise. The variance value of the Gaussian noise is determined experimentally such that the noise can represent the effects of rain, impairments of the image data, shadows, etc.

To generate synthetic images with different intensity values for simulating different lighting conditions, we add the same intensity values based on the uniform distribution between $[-0.3, 0.3]$ to the each pixel of the images ($I_{\text{RGB}}$s) of the database. That is, we obtain the images of the database with different brightness. The limit values

of the uniform distribution are determined experimentally such that the generated images can represent the lighting conditions from the morning until the night without lack of light and direct exposure to the light.
In total 750 synthetic images are generated from the images in CalTech - Cordova1 database. Three examples of the synthetic images generated from an original image are given in Figure 3.15.

(a)

(b)

(c)

(d)

Figure 3.15: Original and synthetic images

Figure 3.15 (a), (b), (c) and (d) show an original image in the database, the original image with noise, the original image with low intensity and high intensity, respectively.

### 3.3.2 Quantitative Comparison and Illustrative Examples

For the quantitative comparison, we firstly compare all combinations of the proposed and described (NAND, 1-BT, Canny, SLT) feature extraction methods by performing the proposed lane detection and tracking algorithm to evaluate the suitability of the proposed feature extraction method. Secondly, quantitative comparison between the proposed and three different state of the art algorithms [20, 64, 65] are made to evaluate the algorithm's effectiveness in the literature.

To evaluate the algorithms quantitatively, different norms for the accuracy metric given in equation 3.12 are performed based on the deviation of manually labeled ground truth lane data from the detected lane data: Mean $L_1$-norm (mean absolute difference), Mean $L_2$-norm (mean square error) and $L_\infty$-norm (maximum error) are given in 3.11 from top to bottom, respectively.

$$\begin{gathered} \frac{1}{N_{\text{lane}}} \sum_{i=1}^{N_{\text{lane}}} |d_i| \\ \frac{1}{N_{\text{lane}}} \sum_{i=1}^{N_{\text{lane}}} (d_i)^2 \\ \max(|d_i|), \;\; i = 1, \cdots, N_{\text{lane}} \end{gathered} \tag{3.11}$$

where $d_i$ and $N_{\text{lane}}$ represent the distance of the $i^{\text{th}}$ pixel position of the detected lane from the pixel position of the ground truth at the same row and the number of detected lane pixels in the perspective view, respectively.

$$\frac{\text{number of correctly detected objects}}{\text{number of all objects}} \tag{3.12}$$

Here, 'number of correctly detected lanes' is the number of lanes where the specified distance bound (Dist) is greater than the distance computed by the defined norms and 'number of all lanes' is the number of all ground truth lanes.

Table 3.1 - 3.3 show the average norm values (Norm) for the correctly detected lanes and the accuracy results (Acc) of the proposed lane detection and tracking algorithm with the proposed and described feature extraction methods and lane models evaluated on the original and synthetic images together based on the Mean $L_1$-norm, Mean $L_2$-norm and $L_\infty$-norm, where the respective norm value is below the indicated distance bound (Dist). The accuracy results are given in the form of the percentage.

Table 3.1: Accuracy results and average norm values based on Mean $L_1$-norm

| Feature Extraction Methods | Dist $< 5$ | | Dist $< 8$ | |
|---|---|---|---|---|
| | Same $a$ | Independent $a$ | Same $a$ | Independent $a$ |
| Proposed method (Acc) | 88.04 | 82.40 | 93.62 | 90.24 |
| NAND (Acc) | 77.04 | 59.60 | 83.96 | 67.86 |
| 1-BT (Acc) | 77.47 | 64.59 | 82.40 | 71.83 |
| SLT (Acc) | 78.27 | 72.91 | 82.73 | 77.25 |
| Canny (Acc) | 78.70 | 38.84 | 82.46 | 43.24 |
| Proposed method (Norm) | 1.87 | 1.88 | 2.13 | 2.25 |
| NAND (Norm) | 2.08 | 2.2 | 2.45 | 2.76 |
| 1-BT (Norm) | 2.04 | 2.02 | 2.3 | 2.46 |
| SLT (Norm) | 1.87 | 1.86 | 2.12 | 2.11 |
| Canny (Norm) | 1.91 | 2.23 | 2.10 | 2.65 |

Table 3.2: Accuracy results and norm values based on Mean $L_2$-norm

| Feature Extraction Methods | Dist $< 5$ | | Dist $< 8$ | |
|---|---|---|---|---|
| | Same $a$ | Independent $a$ | Same $a$ | Independent $a$ |
| Proposed method (Acc) | 85.25 | 79.94 | 92.27 | 88.63 |
| NAND (Acc) | 73.87 | 57.24 | 82.30 | 65.88 |
| 1-BT (Acc) | 75.64 | 62.29 | 81.49 | 70.55 |
| SLT (Acc) | 76.34 | 71.83 | 81.76 | 76.45 |
| Canny (Acc) | 77.09 | 36.91 | 81.33 | 41.79 |
| Proposed method (Norm) | 2.10 | 2.10 | 2.42 | 2.51 |
| NAND (Norm) | 2.27 | 2.40 | 2.72 | 2.97 |
| 1-BT (Norm) | 2.26 | 2.21 | 2.56 | 2.73 |
| SLT (Norm) | 2.11 | 2.11 | 2.40 | 2.37 |
| Canny (Norm) | 2.15 | 2.39 | 2.36 | 2.83 |

Table 3.3: Accuracy results and norm values based on $L_\infty$-norm

| Feature Extraction Methods | Dist $< 5$ | | Dist $< 8$ | |
|---|---|---|---|---|
| | Same $a$ | Independent $a$ | Same $a$ | Independent $a$ |
| Proposed method (Acc) | 62.92 | 59.31 | 80.35 | 76.86 |
| NAND (Acc) | 52.82 | 39.40 | 71.54 | 56.08 |
| 1-BT (Acc) | 53.94 | 42.64 | 77.51 | 62.32 |
| SLT (Acc) | 58.47 | 54.74 | 75.24 | 72.17 |
| Canny (Acc) | 59.84 | 25.96 | 79.68 | 38.91 |
| Proposed method (Norm) | 3.22 | 3.13 | 3.90 | 3.87 |
| NAND (Norm) | 3.31 | 3.26 | 4.17 | 4.32 |
| 1-BT (Norm) | 3.26 | 3.08 | 4.18 | 4.10 |
| SLT (Norm) | 3.29 | 3.28 | 3.97 | 4.00 |
| Canny (Norm) | 3.34 | 3.41 | 4.07 | 4.35 |

According to Table 3.1-3.3, the second-order polynomial with common $a$ coefficient is superior to the second-order polynomial with independent $a$ coefficient for all combination of the different feature extraction methods and metrics. This superiority of the proposed lane model comes from its suitability for modeling parallel lane markings in a relaxed way. Specifically, this property of this lane model is highly important for cases where lane markings in one region (left or right) are worn-out such that most of lane marking pixels could not be extracted. In addition to that, the proposed feature extraction method gives the best results for all defined metrics. Lastly, smaller average norm values for the correctly detected lanes are obtained via proposed lane model in most of the cases. This indicates that smaller deviations from the ground truth are achieved for the correctly detected lanes through the proposed lane model.

For the quantitative comparison between the proposed and the state of the art algorithms [20, 64, 65], the metrics [10] given in equation 3.13 are utilized on the original images of the CalTech - Cordova1 database.

$$\text{AR} = \frac{N_{\text{t}}}{N_{\text{gt}}},\ \text{FP} = \frac{N_{\text{f}}}{N_{\text{all}}},\ \text{FN} = \frac{N_{\text{m}}}{N_{\text{gt}}} \tag{3.13}$$

where AR, FP, FN are accuracy rate, false positive rate, false negative rate, respectively and $N_{\text{t}}$, $N_{\text{f}}$, $N_{\text{m}}$, $N_{\text{gt}}$, $N_{\text{all}}$ represent number of correct lane detections, false lane detections, missed lanes, ground truth lanes, all detected lanes, respectively.

To evaluate a detected lane as a correctly detected lane, criteria in [20] are utilized and Table 3.4 shows the results of the proposed and the state of the art algorithms based on the given metrics. The results of the state of the art algorithms are obtained from [10].

Table 3.4: Performance results of the proposed and the state of the art algorithms

| Algorithms | AR | FP | FN |
|---|---|---|---|
| Proposed algorithm | 0.987 | 0.013 | 0 |
| Aly [20] | 0.972 | 0.3 | - |
| Niu et al. [64] | 0.927 | - | 0.054 |
| Ruyi et al. [65] | 0.974 | - | 0.13 |

According to the results, the proposed algorithm has more number of correct lane detections and fewer number of false lane detections than the state of the art algorithms. Moreover, the proposed algorithm does not miss any ground truth lanes in the given database.

Additionally, the results for the Precision (Pre), Recall (Rec) and F1-measure (F1) metrics given in equation 3.14 [66] are computed as $0.987$, $1$ and $0.993$, respectively.

$$\text{Pre} = \frac{N_{\text{t}}}{N_{\text{t}} + N_{\text{f}}},\ \text{Rec} = \frac{N_{\text{t}}}{N_{\text{t}} + N_{\text{m}}},\ \text{F1} = 2 \cdot \frac{\text{Pre} \cdot \text{Rec}}{\text{Pre} + \text{Rec}} \tag{3.14}$$

According to the results, the proposed algorithm detects lanes in high accuracy and does not miss any ground truth lanes in the given database.

A representative selection of the illustrated results of the proposed lane detection and tracking algorithm are given in Figure 3.16. In the figure, detected lanes and the area between the lanes are shown by red and green colors, respectively. This selection is made from the original and synthetic images from CalTech database in [20]. As can

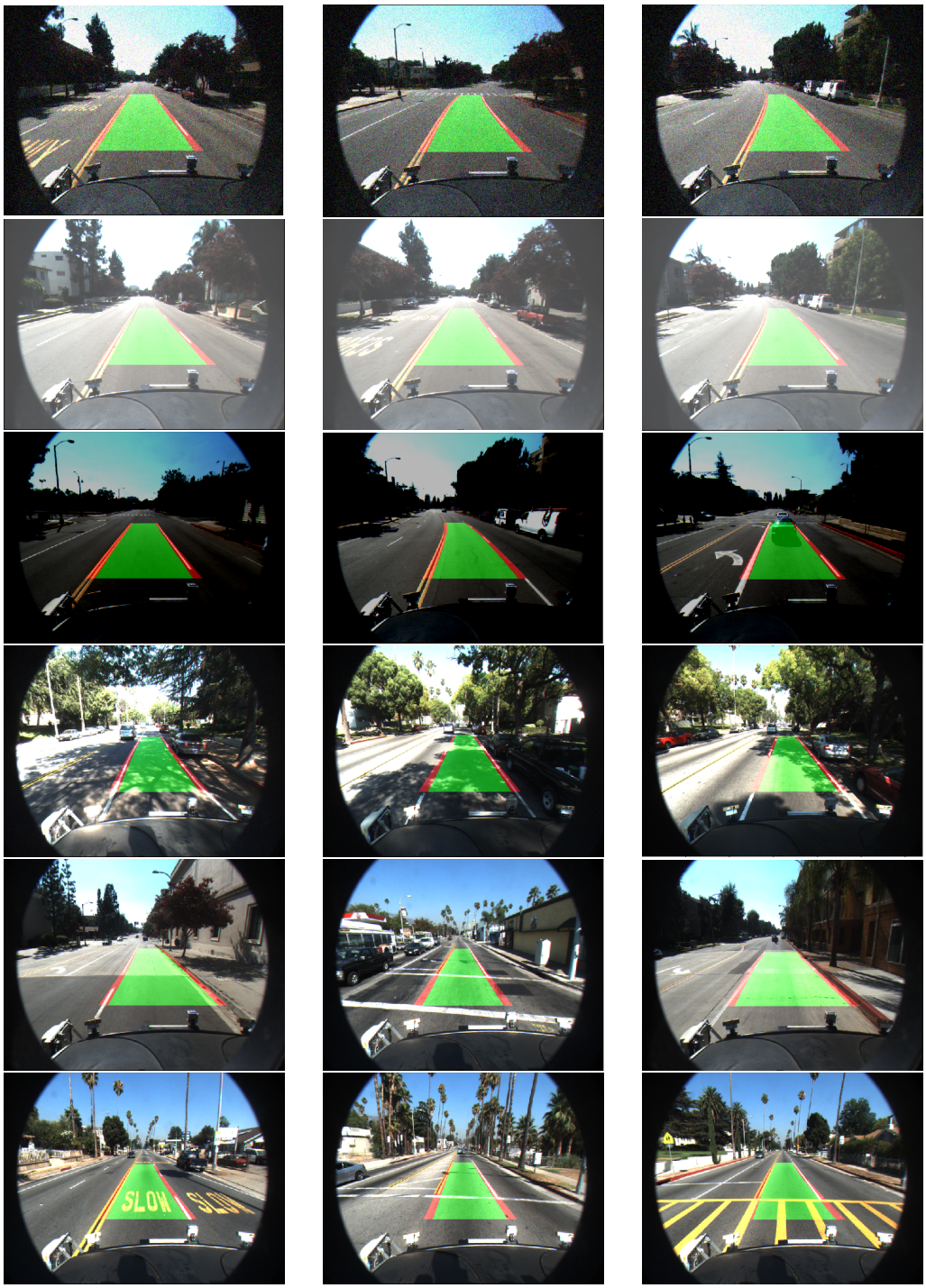



Figure 3.16: A representative selection of lane detection and tracking results

be seen from Figure 3.16 - (first, second and third rows), the proposed method can accurately detect the straight and curved lanes in the generated synthetic images with noise, high intensity and low intensity (see Section 3.3.1 for the generation of the synthetic images), respectively. Also, a correct lane detection where a vehicle close to the camera view is realized as in the third row. Besides, the proposed algorithm performs well for the complex examples of original images in CalTech database [20] where lanes are affected by extreme shadows and close to the other vehicles as in the fourth row of the figure. Moreover, accurate lane detection results can be seen in the fifth row of the original images in the database where the texture of road changes. In addition to that, the algorithm correctly recognize lanes in the original images of the database where there exist road signs close to the lanes as in the last row.

## 3.4 Experimental Tests

In this Section, the proposed lane detection and tracking algorithm is evaluated on the autonomous vehicle kit [33]. For the utilized hardware (Jetson TX2 developer kit and ZED stereo camera) and software, see Section 4.7.1.
To evaluate the proposed algorithm, firstly straight lanes are manually constructed on the ground. Then, the algorithm is implemented and illustrated experimental results are obtained.
A representative selection of lane detection and tracking results from the vehicle are shown in Figure 3.17.

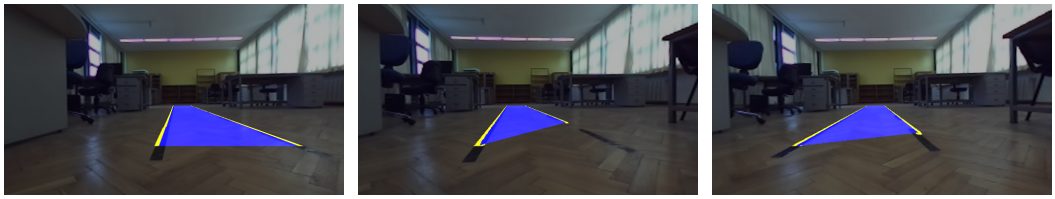


Figure 3.17: Experimental lane detection and tracking results

As can be seen in Figure 3.17, the proposed algorithm detects and tracks the lane markings in cases where the test vehicle is located in different positions and at differ-

ent orientations with respect to the lanes.

# CHAPTER 4

## SENSOR FUSION OF A CAMERA AND 2D LIDAR FOR LANE DETECTION AND TRACKING

Vision-LIDAR systems have complementary perception capabilities and they are frequently used in various autonomous robotic applications such as obstacle avoidance, simultaneous localization and mapping, etc. Since the attributes of the LIDARs and vision systems complement each other, it is plausible to use them together to recognize the lanes for autonomous vehicles. To do that, we firstly detect objects via 2D LIDAR. Then, LIDAR points of the detected objects in the LCF are transformed to the CCF. After that, ground level LIDAR points of the detected objects in the CCF are computed. Furthermore, the ground level LIDAR points of the objects in the CCF are mapped to the BEV image and the pixels occupied by the detected objects from the ground level are evaluated as background on the BEV image. Thus, we obtain a BEV image where the detected object pixels are cleared, modified BEV image. Lastly, the lanes are detected and tracked on the modified BEV image.

The overview of the proposed sensor fusion algorithm is given in Algorithm1.

Algorithm1: Sensor fusion of a camera and 2D LIDAR for lane detection and tracking

1. Get the image frame, extract the candidate lane features and obtain binary BEV image
2. Segment 2D LIDAR data to identify groups/objects
3. Map identified groups/objects to the BEV
4. Turn pixels of groups/objects into background on BEV
5. Perform lane detection and tracking on the modified BEV image

## 4.1 Algorithm1-Step1

In the first step, we obtain binary BEV images as described in Section 3.2.1 and 3.2.2 in a detailed way.

## 4.2 Algorithm1-Step2

In this step, we segment 2D LIDAR data to identify groups/objects. To evaluate the 2D LIDAR data segmentation methods, we generate synthetic 2D LIDAR data and make a quantitative comparison between the methods described in Section 2.3.2 in terms of accuracy and computational complexity by using the generated data.

### 4.2.1 Literature Research

The literature offers several 2D LIDAR data segmentation methods in different applications [57, 58, 32, 67].
[68] segments available gaps using FBD for path planning of mobile robots. In [61], two 2D LIDARs are exploited for pothole detection. After filtering out the high frequency component of the laser scans via a median filter, ABD is applied for abrupt changes to LIDAR points to recognize the potholes. [69] first applies FBD for segmenting LIDAR data and then imposes the minimum number of LIDAR points criterion to the segmented LIDAR data to detect humans from the knee-level for the applications of mobile-robots. In [59, 60], obstacle detection is performed based on 2D LIDAR. To do that, FBD is utilized for segmentation of the data after applying a median filter. In [62], RBNN algorithm is developed for 3D LIDAR data, but can as well be applied to 2D LIDAR data. The advantages of RBNN against the NN and $k$NN algorithm are discussed. Moreover, synthetic data for 3D LIDAR are used without providing a data generation algorithm. [63] segments data from two 2D LIDARs using the dvRBNN for outdoor applications and discusses the advantages over RBNN.

### 4.2.2 Synthetic 2D LIDAR Data Generation Algorithm

Although all segmentation methods come with an individual evaluation, there is no quantitative comparison. In particular, to the best of our knowledge, the existing literature does not offer a common labeled 2D LIDAR database. To make a quantitative comparison between the segmentation methods described in Section 2.3.2, this thesis also proposes a novel synthetic 2D LIDAR data generation algorithm, Algorithm2, and performs a comparative study of the methods based on the generated synthetic data in terms of accuracy and computational complexity. The algorithm first places objects, triangles and parallelograms, with different shapes, orientations and sizes in the range of the 2D LIDAR based on a predefined minimum distance between the objects. After that, we generate the LIDAR points hitting the each edge of the object. This procedure is applied until reaching a predetermined number of objects. As the last step of the algorithm, we eliminate LIDAR points which are not scanned by the 2D LIDAR because of the occlusions from other objects. In order to parametrize the generated LIDAR data, we introduce $\zeta_{\text{ave}}$ as the average edge length of objects, $d_{\text{MD}}$ as the minimum distance between objects and $N_{\text{O}}$ as the desired number of objects. The Algorithm2 is presented in a detailed way in this Section.

Algorithm2: Synthetic 2D LIDAR data generation algorithm

1. Determine the shape and size of the next object using $\zeta_{\text{ave}}$
2. Locate and orient the object
3. Determine the minimum distance to other objects
4. If the minimum distance is larger than $d_{\text{MD}}$ and line equation of each edge of the object is definite
   Accept the object and generate LIDAR points hitting the object
5. If the desired number of objects $N_{\text{O}}$ has not been generated yet
   Goto 1
6. Postprocessing to remove occluded LIDAR points

We next give a detailed explanation of Algorithm2 step by step. Note that the uniform distribution is used for all random assignments.

#### 4.2.2.1 Algorithm2-Step1

First, we select the object type. In this work, we use triangles and parallelograms. Then interior angles ($\psi^{\text{l}}$s) and lengths of the edges ($\zeta^{\text{l}}$s) are determined randomly within predefined intervals $[\psi^{\text{l}}_{\text{min}},\ \psi^{\text{l}}_{\text{max}}]$ and $[\zeta^{\text{l}}_{\text{min}},\ \zeta^{\text{l}}_{\text{max}}]$.

#### 4.2.2.2 Algorithm2-Step2

Second, we rotate the object from its center point by an angle $\phi^{\text{l}}$, which is randomly determined in a predefined interval $[\phi^{\text{l}}_{\text{min}},\ \phi^{\text{l}}_{\text{max}}]$. Then, we randomly shift the center point of the object by $s^{\text{l}}_{\text{x}}$ ($\text{X}^{\text{l}}$-axis) and $s^{\text{l}}_{\text{y}}$ ($\text{Y}^{\text{l}}$-axis) in a predefined interval $[s^{\text{l}}_{\text{x,min}},\ s^{\text{l}}_{\text{x,max}}]$ for $s^{\text{l}}_{\text{x}}$ and $[s^{\text{l}}_{\text{y,min}},\ s^{\text{l}}_{\text{y,max}}]$ for $s^{\text{l}}_{\text{y}}$.

#### 4.2.2.3 Algorithm2-Step3

In this step, the minimum distances from the object to the other objects are computed [70].

#### 4.2.2.4 Algorithm2-Step4

If the minimum distance is smaller than the predefined minimum distance $d_{\text{MD}}$, we discard the object and directly go to Step-1. After that, we consider each edge of the object, $e^{\text{l}}$, defined by the line equation:

$$y^{\text{l}} = a_{\text{e}} x^{\text{l}} + b_{\text{e}} \tag{4.1}$$

between the corner points $c^{\text{l}}_w = (x^{\text{l}}_w, y^{\text{l}}_w)$ and $c^{\text{l}}_t = (x^{\text{l}}_t, y^{\text{l}}_t)$ as in Figure 4.1 (a) without considering occlusions. That is, all other edges (shown under the cross sign) are not considered in the view of the 2D LIDAR. For the indefinite cases of the edge, $a_{\text{e}} = \pm\infty$, we discard the object and directly go to Step-1. Otherwise, we accept the object and generate the LIDAR points hitting the object. To this end, the angles $\gamma^{\text{l}}_w$ and $\gamma^{\text{l}}_t$ as given in Figure 4.1 (b) from the positive $\text{X}^{\text{l}}$-axis to the vectors formed by the origin of LCF to the corner points of the corresponding edge are computed for

$k \in \{w, t\}$ and $|c_k^{\mathrm{l}}| = \sqrt{(x_k^{\mathrm{l}})^2 + (y_k^{\mathrm{l}})^2}$ as

$$\gamma_k^{\mathrm{l}} = \begin{cases} \arccos\left(x_k^{\mathrm{l}}/|c_k^{\mathrm{l}}|\right) & \text{if} \quad y_k^{\mathrm{l}} \geq 0 \\ 180^\circ + \arccos\left(-x_k^{\mathrm{l}}/|c_k^{\mathrm{l}}|\right) & \text{otherwise} \end{cases} \tag{4.2}$$

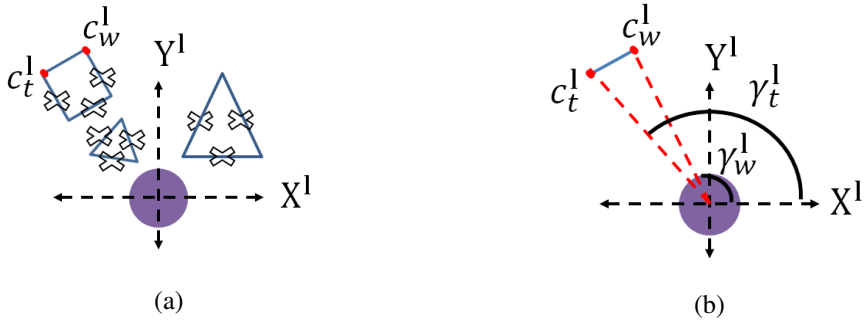



Figure 4.1: The edges and their corresponding angles

Thereafter, the set $\Gamma^{\mathrm{l}}$ of LIDAR angles hitting the edge $e^{\mathrm{l}}$ in multiples of the angle resolution $\Delta\theta^{\mathrm{l}}$ with the assumption that $\gamma_t^{\mathrm{l}}$ is greater than $\gamma_w^{\mathrm{l}}$ is determined as:

$$\Gamma^{\mathrm{l}} \subset \begin{cases} [0, \ \gamma_w^{\mathrm{l}}], \ [\gamma_t^{\mathrm{l}}, \ 360 - \Delta\theta^{\mathrm{l}}] & \text{if} \ -\dfrac{b_{\mathrm{e}}}{a_{\mathrm{e}}} > 0 \ \text{ and } \ y_w^{\mathrm{l}} \cdot y_t^{\mathrm{l}} \leq 0 \\ [\gamma_w^{\mathrm{l}}, \ \gamma_t^{\mathrm{l}}] & \text{otherwise} \end{cases} \tag{4.3}$$

Using the line equation of the edge in 4.1, the x-coordinate and y-coordinate of the LIDAR point in the LCF hitting the edge for each LIDAR angle $\gamma^{\mathrm{l}} \in \Gamma^{\mathrm{l}}$ is computed as follows:

$$x^{\mathrm{l}} = \frac{b_{\mathrm{e}}}{\tan(\gamma^{\mathrm{l}}) - a_{\mathrm{e}}} \quad y^{\mathrm{l}} = \tan(\gamma^{\mathrm{l}}) \frac{b_{\mathrm{e}}}{\tan(\gamma^{\mathrm{l}}) - a_{\mathrm{e}}} \tag{4.4}$$

The procedure explained above is applied for all edges to specify all the LIDAR points hitting the corresponding object.

An example of generating LIDAR points hitting 30 symbolized objects is given in Figure 4.3 (a).

#### 4.2.2.5 Algorithm2-Step5

In this step, we check if the desired number $N_{\mathrm{O}}$ of objects is reached. If not, we go to Step 1 to generate new objects.

#### 4.2.2.6 Algorithm2-Step6

When reaching this step, a sufficient number of $N_O$ objects has been generated. Then, we eliminate the LIDAR points which are not scanned by the 2D LIDAR due to occlusions from other edges of objects. To do that, the distance values of LIDAR points for each angle in the set $\Gamma^l$ of all edges are determined and only the one with the minimum distance to the origin (LIDAR position), is kept. The procedure for the $k^{th}$ angle $\gamma_k^l$ in the set $\Gamma^l$ of all edges is given in Figure 4.2. The point corresponding to the distance $\rho_{ik}^l$ in the red circle is kept as it is closest to the origin of LCF.

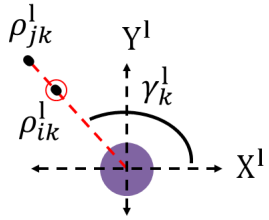
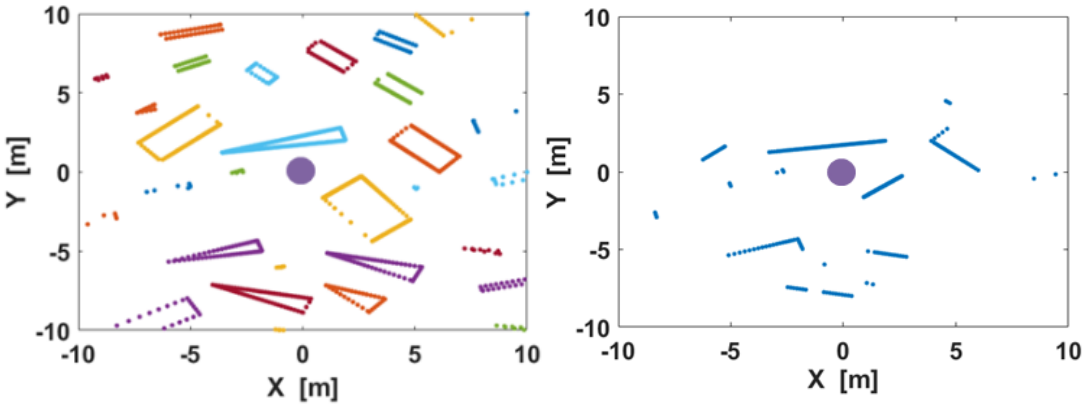



Figure 4.2: Eliminating LIDAR points not in the view of 2D LIDAR

Finally, we add uniform noise to the LIDAR points in the interval specified by the distance accuracy of the 2D LIDAR.
An example of LIDAR points in Figure 4.3 (a) after the postprocessing explained here is given in Figure 4.3 (b).




(a) (b)

Figure 4.3: LIDAR point before (a) and after (b) postprocessing step

### 4.2.3 Computational Evaluation

This Section performs a quantitative comparison of the methods explained in Section 2.3.2 based on different test cases of generated synthetic data. The parameters of the segmentation methods are $th_{\text{f}}^{\text{l}} = 0.65$, $\lambda = 18$, $\sigma_{\text{r}} = 0.13$, $k_{\text{NN}} = 3$, and $th_{\text{scale}} = 3.4$. In total, $9000$ different test cases with varying values of $d_{\text{MD}}$, $\zeta_{\text{ave}}$ and $N_{\text{O}}$ are generated. In the synthetic data generation phase, the LIDAR specific parameters are defined as $\Delta\theta^{\text{l}} = 1°$, $[\zeta^{l}_{\text{min}},\ \zeta^{l}_{\text{max}}] = [0.1m,\ 4m]$, $[s^{\text{l}}_{\text{x,min}},\ s^{\text{l}}_{\text{x,max}}] = [s^{\text{l}}_{\text{y,min}},\ s^{\text{l}}_{\text{y,max}}] = [-10m,\ 10m]$, $[\psi^{\text{l}}_{\text{min}},\ \psi^{\text{l}}_{\text{max}}] = [\phi^{\text{l}}_{\text{min}},\ \phi^{\text{l}}_{\text{max}}] = [-45°,\ 45°]$, and the accuracy of the 2D LIDAR is $0.1m$. For comparison, we use the accuracy metric in equation 4.5.

$$\frac{\text{number of correctly detected objects}}{\text{number of all objects}} \tag{4.5}$$

Here, 'number of correctly detected objects' is the number of objects whose scanned LIDAR points are all detected correctly. 'number of all objects' is the number of objects scanned by the 2D LIDAR.
The quantitative results depending on $d_{\text{MD}}$, $\zeta_{\text{ave}}$ and $N_{\text{O}}$ are given in Table 4.1, Table 4.2 and Table 4.3, respectively.

Table 4.1: Accuracy results according to $d_{\text{MD}}$ ($N_{\text{O}} = 20$ and $\zeta_{\text{ave}} = 0.8m$)

| Methods | $d_{\text{MD}} = 0.3m$ | $d_{\text{MD}} = 1m$ | $d_{\text{MD}} = 2m$ |
|---|---|---|---|
| FBD | 0.7848 | 0.8762 | 0.8562 |
| ABD | 0.7924 | 0.8915 | 0.9019 |
| RBNN | 0.7801 | 0.8992 | 0.8589 |
| dvRBNN | 0.8102 | 0.9234 | 0.9207 |
| $k$NN | 0.5798 | 0.5819 | 0.4848 |
| NN | 0.6029 | 0.6176 | 0.6112 |

Table 4.2: Accuracy results according to $\zeta_{\text{ave}}$ ($d_{\text{MD}} = 1m$ and $N_{\text{O}} = 20$)

| Methods | $\zeta_{\text{ave}} = 0.4m$ | $\zeta_{\text{ave}} = 1.1m$ | $\zeta_{\text{ave}} = 2.4m$ |
|---|---|---|---|
| FBD | 0.9722 | 0.8719 | 0.7391 |
| ABD | 0.9753 | 0.9152 | 0.8154 |
| RBNN | 0.9924 | 0.8848 | 0.7415 |
| dvRBNN | 0.9982 | 0.9432 | 0.8442 |
| $k$NN | 0.5354 | 0.7396 | 0.7507 |
| NN | 0.7321 | 0.6604 | 0.5764 |

Table 4.3: Accuracy results according to $N_{\text{O}}$ ($d_{\text{MD}} = 1m$ and $\zeta_{\text{ave}} = 0.95m$)

| Methods | $N_{\text{O}} = 10$ | $N_{\text{O}} = 20$ | $N_{\text{O}} = 30$ |
|---|---|---|---|
| FBD | 0.8451 | 0.8528 | 0.8580 |
| ABD | 0.8968 | 0.8962 | 0.8950 |
| RBNN | 0.8491 | 0.8608 | 0.8685 |
| dvRBNN | 0.9147 | 0.9194 | 0.9179 |
| $k$NN | 0.5964 | 0.5155 | 0.4728 |
| NN | 0.5888 | 0.6129 | 0.5988 |

For all test cases, the same tunable parameters are used for each method and the parameters giving the possibly best results in all cases are determined experimentally. In the test cases, the number of the occlusions increases with a decreasing $d_{\text{MD}}$ and increasing $\zeta_{\text{ave}}$, $N_{\text{O}}$. Besides, the variance of all LIDAR points hitting the objects increases with $d_{\text{MD}}$, $\zeta_{\text{ave}}$ and $N_{\text{O}}$.

In Table 4.1, the accuracies of the methods generally decrease with decreasing $d_{\text{MD}}$ as the objects become closer to each other. In Table 4.2, the accuracies of the methods generally increase with decreasing $\zeta_{\text{ave}}$ as the objects become more distant due to the uniform distribution of $s_{\text{x}}^{\text{l}}$ and $s_{\text{y}}^{\text{l}}$ in the range of the 2D LIDAR. In Table 4.3, the accuracy results of each method for the specified $N_{\text{O}}$s are close to each other as $d_{\text{MD}}$ is the same for all the cases and the number of occlusions as in Figure 4.4 (b) is small. In particular, the accuracy results of FBD and RBNN are close to each other for the

chosen values of $d_{\text{MD}}$ and $\zeta_{\text{ave}}$.

According to the results, dvRBNN is superior to the other methods. This superiority arises from its adaptive threshold based on the distance value of 2D LIDAR and the robustness against the occlusions of small parts of the objects by other objects. Figure 4.4 shows a dvRBNN segmentation result and each color represents a different LIDAR group.

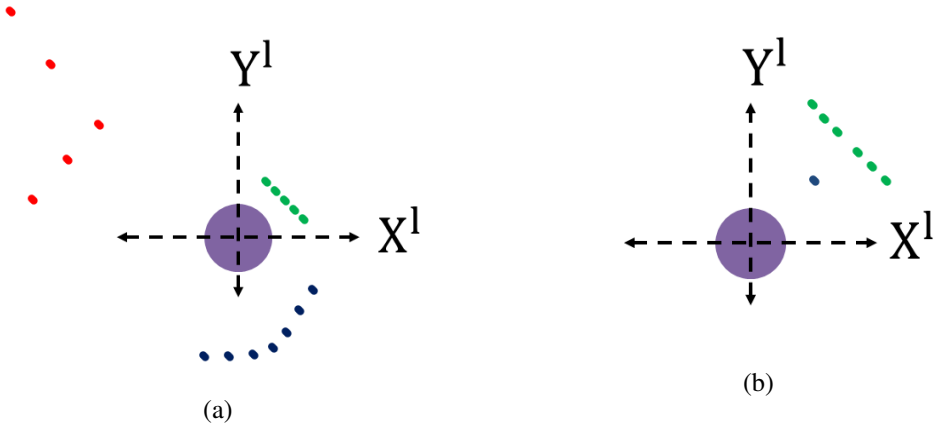



Figure 4.4: Segmentation results of dvRBNN

Unlike FBD, RBNN, NN, $k$NN, the adaptive threshold of dvRBNN compensates for the increasing distances between the LIDAR points as in Figure 4.4 (a) since the distance varying radius increases with the distance values of LIDAR points. Besides, unlike the methods FBD and ABD using the distance between consecutive points, the occluded objects as in Figure 4.4 (b) are detected accurately since dvRBNN connects every LIDAR point to its neighbors which are inside the computed radius irrespective of the sequence of LIDAR points.

Regarding the computational complexity, FBD and ABD are the most simple methods as there is only one query per LIDAR point. Considering the number of LIDAR points $N_{\text{l}}$ as the input, FBD and ABD are performed in $O(N_{\text{l}})$. As for NN, $k$NN, RBNN, dvRBNN, the algorithm consists of constructing and searching a graph, which can be done in $O(N_{\text{l}}^2)$ in the worst case. That is, dvRBNN comes with a higher complexity compared to FBD and ABD but with a considerable performance improvement. Furthermore, dvRBNN shows the best performance among the algorithms whose complexity is in the order of $O(N_{\text{l}}^2)$.

## 4.3 Algorithm1-Step3

Here, we first transform the LIDAR points of the detected objects in the LCF to the CCF. To do that we first do extrinsic calibration between the 2D LIDAR and camera such that we could estimate the transformation between the sensors. After that, ground level LIDAR points of the detected objects in the CCF are computed. Furthermore, the ground level LIDAR points of the objects in the CCF are projected to the BEV image.

### 4.3.1 Extrinsic Calibration between a Camera and 2D LIDAR

To match the corresponding LIDAR and camera data, the relative coordinate systems of the camera and 2D LIDAR should be estimated via extrinsic calibration.

#### 4.3.1.1 Literature Research

Several methods have been performed for the extrinsic calibration in the literature. The methods can be seperated into two groups; explicit and implicit methods.
In the explicit methods [71, 72, 73], the laser beams are visible thanks to infra-red cameras, etc. in the images. In this category, the corresponding observed LIDAR and vision features such as points, lines, etc. are known and the correspondences are employed to compute the relative position and orientation between these sensors.
In the implicit methods, the laser scans are invisible in the images as in this thesis. In this category, the transformation between the sensors are determined through features such as lines, edges, corners, intersection points, etc. in each sensor modality. [74, 75, 76, 77, 78] utilize a planar checkerboard as a calibration target. The whole calibration target should be scanned by the LIDAR and captured by the camera simultaneously in different poses. The methods firstly estimate the origin and normal direction of the calibration target according to the camera coordinate frame by the help of camera calibration technique. [74] is the state of art in this category and it takes advantage of the constraint which enforces the equality of the distances between the any point on the chekerboard in the direction of the estimated normal.

They estimate rotation and translation between 2D LIDAR and a camera coordinate frames by minimizing the distance between the origin of chekerboard and the laser scans on the chekerboard according to the camera coordinate frame in the direction of the estimated normal by least square square error method, first. Then, the estimated transformation matrix is refined by nonlinear optimization by the same constraint. The method estimates the rotation matrix and the translation vector in coupled way. The works in [75, 76] are analogous to [74] with the difference that a 3D LIDAR and a camera are extrinsically calibrated, instead of 2D LIDAR. [77] takes advantage of the constraint which enforces the perpendicularity of the estimated normal and any vector on the calibration target. They firstly estimate the rotation matrix by the perpendicularity of the estimated normal and the vector which is composed of the laser scans on the calibration target via SVD. After that, rotation matrix is refined by nonlinear optimization. Next, translation vector is estimated via nonlinear optimization techniques followed by least square error by the fact that the estimated normal is perpendicular to the vector which is between the origin of the chekerboard and the centeroid of the laser scans on the chekerboard. Furthermore, RANSAC algorithm is used to eliminate effect of the outliers from the rotation and translation. The proposed method firstly estimates rotation matrix and then translation vector between 2D LIDAR and camera coordinate frames in a decoupled way. [78] uses the fact that estimated normal and any vector on the calibration target are perpendicular to each other. They firstly estimate the rotation matrix by the perpendicularity of the estimated normal and the vector which is composed of the laser scans on the calibration target via SVD. In addition to that, they imposes the orthogonality properties of the rotation matrix. After that, rotation matrix is refined by nonlinear optimization. Next, translation vector is estimated via nonlinear optimization techniques followed by least square error by the fact that the estimated normal is perpendicular to the vector which is between the origin of the chekerboard and the centeroid of the laser scans on the chekerboard. The proposed method firstly estimates rotation matrix and then translation vector between 2D LIDAR and camera coordinate frames in a decoupled way. [79] uses a black right-angled triangular object as a calibration target. The presented work firstly detects depth edges and the perpendicular edges as features to constitute the constraints by using a 2D LIDAR and camera, respectively. Then, they calculate the extrinsics by minimizing the total squared distance between corresponding LIDAR and camera

edges for each pose. Besides, different types of features such as points, lines, etc. are discussed to associate the measurements. [80, 81] uses a v-shaped calibration target with distinct black stripes on the edges and the centerline. They associate the left-right edges and centerline features in the image to the left-right depth and center point features in LIDAR measurements, respectively. To compute the extrinsic parameters, range based total weighted-squared distance between the correspondences are minimized via nonlinear optimization for each pose.

#### 4.3.1.2 The Implemented Method for the Extrinsic Calibration between a Camera and 2D LIDAR

To obtain rotation matrix and translation vector between the camera and 2D LIDAR, the implicit method mostly based on [77] is implemented in the scope of the thesis.
The method utilizes a planar checkerboard as a calibration target. The whole calibration target (plane) must be scanned by the 2D LIDAR and captured by the camera simultaneously in different poses. Each different pose of the calibration target puts constraints on the extrinsic parameters which are the rotation matrix and translation vector between the camera and 2D LIDAR. In the method, the rotation matrix is first estimated and then the translation vector is determined.
The proposed method aims to solve the transformation from the LCF to the CCF. The transformation can be defined as:

$$P^{\mathrm{c}} = R_{\mathrm{l}}^{\mathrm{c}} P^{\mathrm{l}} + t_{\mathrm{l}}^{\mathrm{c}} \tag{4.6}$$

where $P^{\mathrm{c}}$ and $P^{\mathrm{l}}$ represent the same location in the CCF and LCF, respectively. $R_{\mathrm{l}}^{\mathrm{c}}$ is a rotation matrix describing the rotation from LCF to the CCF and $t_{\mathrm{l}}^{\mathrm{c}}$ is a translation vector representing origin of the LCF in the CCF.
For illustration purposes, a schematic of the CCF and LCF are given in Figure 4.5, where $[X^{\mathrm{l}}, Y^{\mathrm{l}}, Z^{\mathrm{l}}]^{\mathrm{T}}$ represents the coordinates of laser scans in the LCF and without loss of generality, the laser scans are assumed to be on the plane $Z^{\mathrm{l}} = 0$.

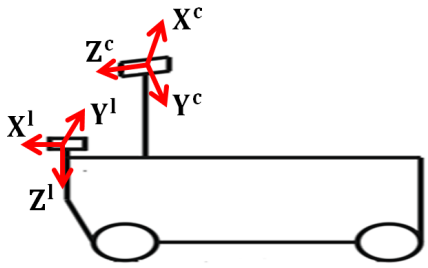



Figure 4.5: The schematic of the CCF and LCF on a vehicle

Let $n_i^{\mathrm{c}}$ be the normal vector of the calibration target in the $i^{\mathrm{th}}$ pose according to the camera coordinate frame, which is the third column vector of $R_{\mathrm{ch}}^{\mathrm{c}}$ which can be estimated as in Section 2.2.11. Let $\bar{p}_{ij}^{\mathrm{l}} = p_{ij}^{\mathrm{l}} - \bar{p}_i^{\mathrm{l}}$ be the difference vector between $p_{ij}^{\mathrm{l}}$ and $\bar{p}_i^{\mathrm{l}}$ points in the LCF, where $p_{ij}^{\mathrm{l}}$ is the $j^{\mathrm{th}}$ laser scans hitting the chekerboard and $\bar{p}_i^{\mathrm{l}}$ is the center point of the laser scans hitting the checkerboard in the $i^{\mathrm{th}}$ pose.
Since $\bar{p}_{ij}^{\mathrm{l}} = [\bar{x}_{ij}^{\mathrm{l}}, \bar{y}_{ij}^{\mathrm{l}}, \bar{z}_{ij}^{\mathrm{l}}]^{\mathrm{T}}$ vector is on the calibration target in the LCF and $n_i^{\mathrm{c}}$ is perpendicular to the calibration plane in the CCF, we have the following relation for each laser scan hitting the calibration target in each pose:

$$(n_i^{\mathrm{c}})^T (R_{\mathrm{l}}^{\mathrm{c}} \bar{p}_{ij}^{\mathrm{l}}) = 0 \tag{4.7}$$

Equation 4.7 can be rewritten as follows due to the fact that the laser scans lie in the plane $Z_{\mathrm{l}}$=0.

$$[\bar{x}_{ij}^{\mathrm{l}} n_{i1}^{\mathrm{c}} \quad \bar{x}_{ij}^{\mathrm{l}} n_{i2}^{\mathrm{c}} \quad \bar{x}_{ij}^{\mathrm{l}} n_{i3}^{\mathrm{c}} \quad \bar{y}_{ij}^{\mathrm{l}} n_{i1}^{\mathrm{c}} \quad \bar{y}_{ij}^{\mathrm{l}} n_{i2}^{\mathrm{c}} \quad \bar{y}_{ij}^{\mathrm{l}} n_{i3}^{\mathrm{c}}] r = 0 \tag{4.8}$$

where $r = [r_1^{\mathrm{T}}, r_2^{\mathrm{T}}]^{\mathrm{T}}$ is a $6 \times 1$ vector, $r_1$ and $r_2$ represent the first and second column of the rotation matrix $R_{\mathrm{l}}^{\mathrm{c}}$, respectively. Furthermore, $n_{i1}^{\mathrm{c}}$, $n_{i2}^{\mathrm{c}}$ and $n_{i3}^{\mathrm{c}}$ represent the first, second and third component of the normal vector $n_i^{\mathrm{c}}$, respectively.
Several equations are in the form of 4.8 for each pose. So, all the equations have the form $Ar = 0$. The solution $r$ can be computed via SVD of $A = USV^{\mathrm{T}}$. In that case, $r$ becomes the last column of the $V$. Thus the rotation matrix $R_{\mathrm{l}}^{\mathrm{c}}$ can be constituted as:

$$[r_1, r_2, r_1 \times r_2] \tag{4.9}$$

Since the computed $R_{\mathrm{l}}^{\mathrm{c}}$ does probably not satisfy attributes of a rotation matrix, nearest rotation matrix ($\hat{R}_{\mathrm{l}}^{\mathrm{c}}$) to $R_{\mathrm{l}}^{\mathrm{c}}$ based on the Frobenius norm can be computed as fol-

lows [82]:

$$\hat{R}_{\mathrm{l}}^{\mathrm{c}} = U_{\mathrm{R}} V_{\mathrm{R}}^{\mathrm{T}} \tag{4.10}$$

where $U_{\mathrm{R}}$ and $V_{\mathrm{R}}$ are the unitary matrices and obtained by the SVD of the computed $R_{\mathrm{l}}^{\mathrm{c}} = U_{\mathrm{R}} S_{\mathrm{R}} V_{\mathrm{R}}^{\mathrm{T}}$ in equation 4.9.

After estimating $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ via the linear methods, we refine $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ by minimizing the cost function in equation 4.11 through a nonlinear optimization with the fact that the vectors ($n_i^{\mathrm{c}}$ and $\hat{R}_{\mathrm{l}}^{\mathrm{c}} \bar{p}_{ij}^{\mathrm{l}}$) should be orthogonal to each other in noise-free environment.

$$\sum_i \sum_j ((n_i^{\mathrm{c}})^{\mathrm{T}} (\hat{R}_{\mathrm{l}}^{\mathrm{c}} \bar{p}_{ij}^{\mathrm{l}}))^2 \tag{4.11}$$

$\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ can be parameterized by the Rodrigues formula not to violate the form of the rotation matrix. A rotation matrix is represented as a $3 \times 1$ vector by Rodrigues formula. The minimization of the cost function is realized using Levenberg-Marquardt algorithm [83].

After refining $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ via the nonlinear optimization, we can estimate the translation vector, $t_{\mathrm{l}}^{\mathrm{c}}$, via LSM described in Section 2.1.1. The translation vector satisfies the following equation:

$$(n_i^{\mathrm{c}})^{\mathrm{T}} t_{\mathrm{l}}^{\mathrm{c}} = -(n_i^{\mathrm{c}})^{\mathrm{T}} (\hat{R}_{\mathrm{l}}^{\mathrm{c}} \bar{p}_i^{\mathrm{l}} - p_i^{\mathrm{c}}) \tag{4.12}$$

where $p_i^{\mathrm{c}}$ is any point on the calibration plane in the CCF and $t_{\mathrm{ch}}^{\mathrm{c}}$ which can be estimated as in Section 2.2.11 can be used for $p_i^{\mathrm{c}}$.

After estimating rotation matrix $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ via nonlinear optimization and translation vector $t_{\mathrm{l}}^{\mathrm{c}}$ via LSM, $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ and $t_{\mathrm{l}}^{\mathrm{c}}$ are reestimated together in a coupled way. To do that, the cost function given in 4.13 is minimized using Levenberg-Marquardt algorithm with the fact that the vectors ($n_i^{\mathrm{c}}$ and $\hat{R}_{\mathrm{l}}^{\mathrm{c}} \bar{p}_i^{\mathrm{l}} + t_{\mathrm{l}}^{\mathrm{c}} - p_i^{\mathrm{c}}$) are orthogonal to each other in noise-free environment. The vector, $\hat{R}_{\mathrm{l}}^{\mathrm{c}} \bar{p}_i^{\mathrm{l}} + t_{\mathrm{l}}^{\mathrm{c}} - p_i^{\mathrm{c}}$, is a line segment from the point $p_i^{\mathrm{c}}$ to the center point of the laser scans hitting the checkerboard in the $i^{\mathrm{th}}$ pose according to the CCF and lies on the calibration plane.

$$\sum_i ((n_i^{\mathrm{c}})^{\mathrm{T}} (\hat{R}_{\mathrm{l}}^{\mathrm{c}} \bar{p}_i^{\mathrm{l}} + t_{\mathrm{l}}^{\mathrm{c}} - p_i^{\mathrm{c}}))^2 \tag{4.13}$$

To eliminate the effect of outliers from $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ and $t_{\mathrm{l}}^{\mathrm{c}}$, RANSAC algorithm can be used. Estimation of the $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ via RANSAC algorithm is as follows:

1. Select randomly a determined number of images and the corresponding laser scans hitting the chekerboard. Then, compute $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ using the linear techniques described in this Section.

2. Compute the distance ($d_{Ri}$) given as follows for each pose:

$$d_{Ri} = \sum_{j=1}^{m_i} |(n_i^{\mathrm{c}})^{\mathrm{T}}(\hat{R}_{\mathrm{l}}^{\mathrm{c}}\vec{p}_{ij}^{\,\mathrm{l}})| \tag{4.14}$$

   where $m_i$ is the number of laser scans hitting the chekerboard in $i^{\text{th}}$ pose.

3. Determine the number of poses which satisfies the condition-1 ($d_{Ri} < th_{\mathrm{R}}$) where $th_{\mathrm{R}}$ is the predetermined threshold value for a valid rotation matrix.

4. Repeat Step-1 to Step-3 for $n_{\mathrm{R}}$ times and choose $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ having the maximum number of poses that satisfies the condition-1.

5. Reestimate $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ by using the images and the corresponding laser scans in the poses which satisfies the condition-1 via linear techniques.

6. Refine the rotation $\hat{R}_{\mathrm{l}}^{\mathrm{c}}$ by minimizing the cost function given in equation 4.11 via Levenberg-Marquardt algorithm using the images and the corresponding laser scans in the poses which satisfies the condition-1 in the previous step.

7. Compute the distance, $d_{Ri}$, and determine the poses satisfying the condition-1.

8. Repeat Step-5 to Step-7 until the number of poses that satisfies the condition-1 converges.

Estimation of the $t_{\mathrm{l}}^{\mathrm{c}}$ via RANSAC algorithm is as follows:

1. Select randomly a determined number of images and the corresponding laser scans hitting the chekerboard. Then, compute $t_l^c$ using the LSM.

2. Compute the distance ($d_{ti}$) given as follows for each pose.

$$d_{ti} = |(n_i^{\mathrm{c}})^{\mathrm{T}}(\hat{R}_{\mathrm{l}}^{\mathrm{c}}\vec{p}_i^{\,\mathrm{l}}) + t_{\mathrm{l}}^{\mathrm{c}} - p_i^{\mathrm{c}}| \tag{4.15}$$

3. Determine the number of poses which satisfies the condition-2 ($d_{ti} < th_{\mathrm{t}}$) where $th_{\mathrm{t}}$ is the predetermined threshold value for a valid translation vector.

4. Repeat Step-1 to Step-3 for $n_t$ times and choose $t_l^c$ having the maximum number of poses that satisfies the condition-2.

5. Reestimate $t_l^c$ by using the images and the corresponding laser scans in the poses which satisfies the condition-2 via the LSM.

6. Refine the translation $t_l^c$ and $\hat{R}_l^c$ by minimizing the cost function given in 4.13 via Levenberg-Marquardt algorithm using the images and the corresponding laser scans in the poses which satisfies the condition-2 in the previous step.

7. Compute the distance, $d_{ti}$, and determine the poses satisfying the condition-2.

8. Repeat Step-5 to Step-7 until the number of poses that satisfies the condition-2 converges.

To estimate $\hat{R}_l^c$ and $t_l^c$ experimentally, the followed instructions [84] are:

- Place the chekerboard such that it is visible by both camera and 2D LIDAR.

- At least 20-30 laser scans should hit the chekerboard.

- The chekerboard corners should be distinctly detectable in the image.

- At least 15-20 valid pose set in different position and orientation of the chekerboard.

After estimating $\hat{R}_l^c$ and $t_l^c$, laser scans which hit the calibration target and in the image view are projected to the image as given in Figure 4.6 and Figure 4.7, respectively by the help of intrinsic camera parameters estimated in Section 2.2.11. As can be seen in Figure 4.6, the laser scans hit the calibration target also hit the calibration target in the image view.

Figure 4.6: Laser scans hitting the chekerboard

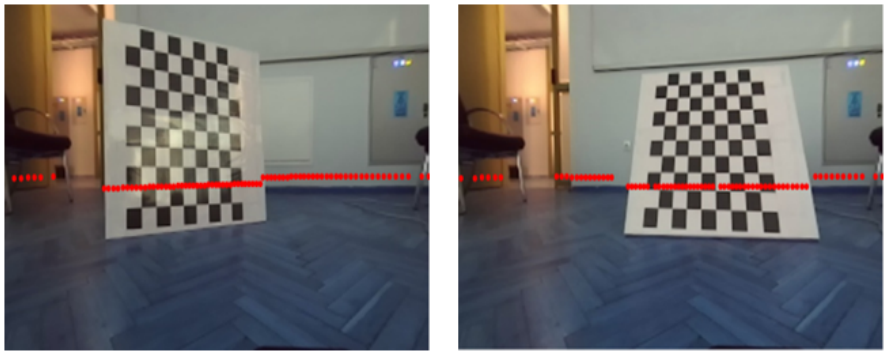


Figure 4.7: Laser scans in the image view

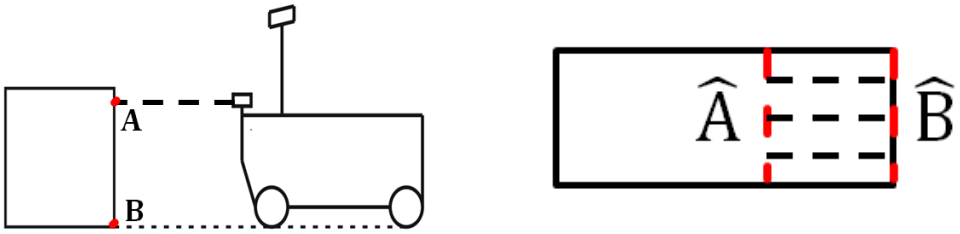
### 4.3.2 Projection of LIDAR Points to the Road Level

In practice, it has to be considered that the camera is not located at the road level and 2D LIDAR scans hit objects above the road level as shown in Figure 4.8 (a) (side view). As a result, the camera can see different points (such as point B at the road level and point A not at the road level in Figure 4.8 (a)) that are located at the same distance from the vehicle. When the object and the LIDAR points taken from this configuration are projected to the BEV in Figure 4.8 (b) (top view), the points A and B correspond to the lines $\hat{A}$ and $\hat{B}$. As the occluded area is bounded by the location of LIDAR points in the BEV, it is necessary to project the LIDAR points to the road level before sensor fusion. As a result, all object pixels are correctly evaluated as background as indicated in Figure 4.8 (b). The formulation of the projection of the LIDAR points to the road level with the assumption that the placement of the CCF is

as described in Section 5.2 and the road is flat is:

$$\alpha_i^{\mathrm{g}} \begin{bmatrix} u_i^{\mathrm{g}} \\ v_i^{\mathrm{g}} \\ 1 \end{bmatrix} = HK \left[ R_{\mathrm{x}}^{-1}(\phi) \left( R_{\mathrm{x}}(\phi) \begin{bmatrix} x_i^{\mathrm{c}} \\ y_i^{\mathrm{c}} \\ z_i^{\mathrm{c}} \end{bmatrix} + \begin{bmatrix} 0 \\ h_{\mathrm{c}} - [2^{\mathrm{nd}} \text{ row of } R_{\mathrm{x}}(\phi)][x_i^{\mathrm{c}} \;\; y_i^{\mathrm{c}} \;\; z_i^{\mathrm{c}}]^{\mathrm{T}} \\ 0 \end{bmatrix} \right) \right] \tag{4.16}$$

where $\phi$ is the tilt angle, $h_{\mathrm{c}}$ is the height, $\alpha_i^{\mathrm{g}}$ is a scaling parameter of the $i^{\mathrm{th}}$ projected LIDAR point, $u_i^{\mathrm{g}}$ and $v_i^{\mathrm{g}}$ are the column and row indexes of the $i^{\mathrm{th}}$ projected LIDAR points to the road level in the BEV image, $R_{\mathrm{x}}(\phi)$ is the rotation matrix performing rotation as $\phi$ in the x-axis of the CCF and $[x_i^{\mathrm{c}} \;\; y_i^{\mathrm{c}} \;\; z_i^{\mathrm{c}}]^{\mathrm{T}}$ is the coordinate of the $i^{\mathrm{th}}$ LIDAR point in the CCF.




(a) (b)

Figure 4.8: The representation of the LIDAR points on BEV image

With the experimental setup in Figure 4.9 (a), the height of the camera can be defined as the length of the projection of $t_{\mathrm{ch}}^{\mathrm{c}}$ vector in the direction of $\mathrm{Z}^{\mathrm{ch}}$-axis plus thickness of the chekerboard ($s_{\mathrm{ch}}$) with the assumption that the ground is flat and the calibration target is planar. The vector, $t_{\mathrm{ch}}^{\mathrm{c}}$, and the direction of $\mathrm{Z}^{\mathrm{ch}}$-axis (i.e. the normal vector of the chekerboard) can be estimated through camera calibration process described in Section 2.2.11. For the camera calibration process, the poses of camera calibration are captured in different orientations and positions on the ground as in Figure 4.9 (a). After estimating $R_{\mathrm{ch}}^{\mathrm{c}}$ and $t_{\mathrm{ch}}^{\mathrm{c}}$, the solution for the $h_{\mathrm{c}}$ is as follows:

$$h_{\mathrm{c}} = [3^{\mathrm{rd}} \text{ column of } R_{\mathrm{ch}}^{\mathrm{c}}]^{\mathrm{T}}[t_{\mathrm{ch}}^{\mathrm{c}}] + s_{\mathrm{ch}} \tag{4.17}$$

The tilt angle of the camera ($\phi$) can be computed as:

$$90 - \phi_{c,ch} \tag{4.18}$$

where the angle $\phi_{c,ch}$ can be defined as the angle between the $Z^{ch}$-axis and $Z^{c}$-axis as illustrated in Figure 4.9 (b) and the solution for the the $\phi_{c,ch}$ is as follows:

$$\arccos\left([0 \;\; 0 \;\; 1][3^{rd} \text{ column of } (R_{ch}^{c})^{-1}]\right) \tag{4.19}$$

where $[0 \;\; 0 \;\; 1]^{T}$ and $[3^{rd}$ column of $(R_{ch}^{c})^{-1}]^{T}$ are the unit vectors representing $Z^{ch}$-axis and $Z^{c}$-axis in the CHCS, respectively.

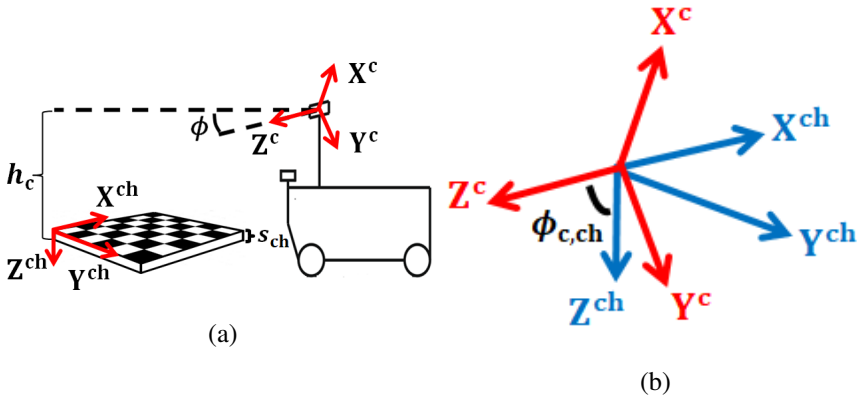



Figure 4.9: The schematic of the experimental setup for the $h_c$ and $\phi$

An example of the LIDAR points mapped to the perspective image and their corresponding road level LIDAR points in blue color are given in Figure 4.10 and 4.11, respectively.

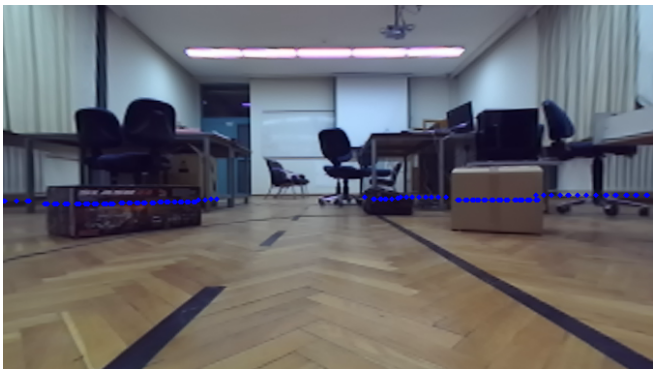


Figure 4.10: The LIDAR points mapped to the perspective image

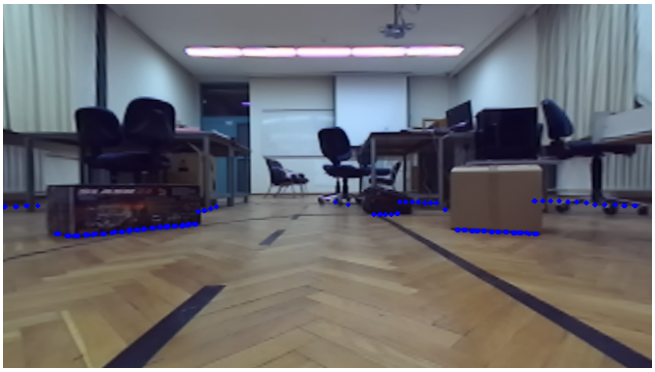


Figure 4.11: The corresponding LIDAR points of Figure 4.10 mapped to the road level perspective image

### 4.4 Algorithm1-Step4

In this step, pixels occluded by objects on binary BEV images become background, where the extracted noise due to objects in front view are removed, thus modified BEV images are acquired. To obtain the modified BEV images, firstly rectangle with the corners ($c_i^k,\ i = 1, \ldots, 4$), which are the minimum and maximum LIDAR points in u-axis and v-axis of the BEV image, is placed around the $k^{\text{th}}$ segmented group of LIDAR points on the BEV images. Then, straight lines are drawn from the camera location (bottom midpoint) through the consecutive corners ($c_i^k$, $c_{i+1}^k$) of the $k^{\text{th}}$ rectangle up to corresponding intersection points ($b_i^k$, $b_{i+1}^k$) of these lines and the image boundaries. Lastly, a convex polygon determined by the boundary lines with the end points $c_i^k$, $c_{i+1}^k$, $b_i^k$, $b_{i+1}^k$ including the top corners of the image frame which do not violate the convexity as described in Section 2.1.6 determines an area. Applying this for all consecutive corners determines the occluded areas where the pixels are evaluated as background. This procedure is applied for all segmented laser groups and it is illustrated for the $k^{\text{th}}$ segmented laser group in Figure 4.12 (a), (b), (c) and (d), sequentially. In the figure, the imaginary LIDAR points on the road level of the vehicle are the dots in red color, the minimum-sized rectangle is shown in blue color, the occluded area is shaded by yellow dashed lines, camera location is the dot in green color and the boundary points are the dots in white color.

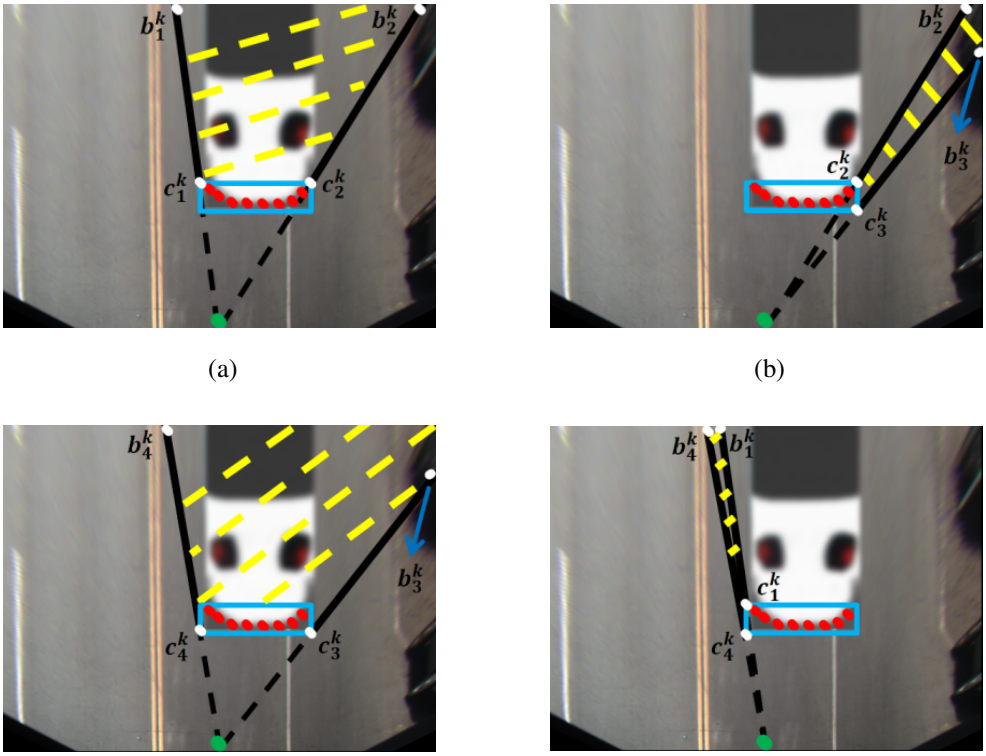



Figure 4.12: The visualization of obtaining the modified BEV image

Total occluded area is the combination of the all occluded areas and it is shaded by yellow dashed lines in Figure 4.13.

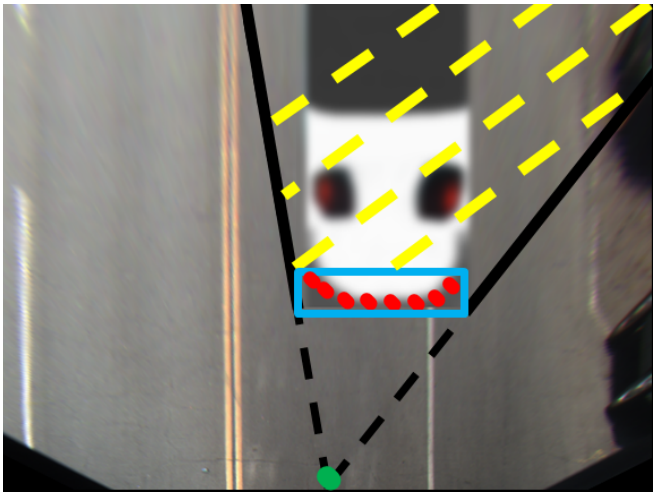
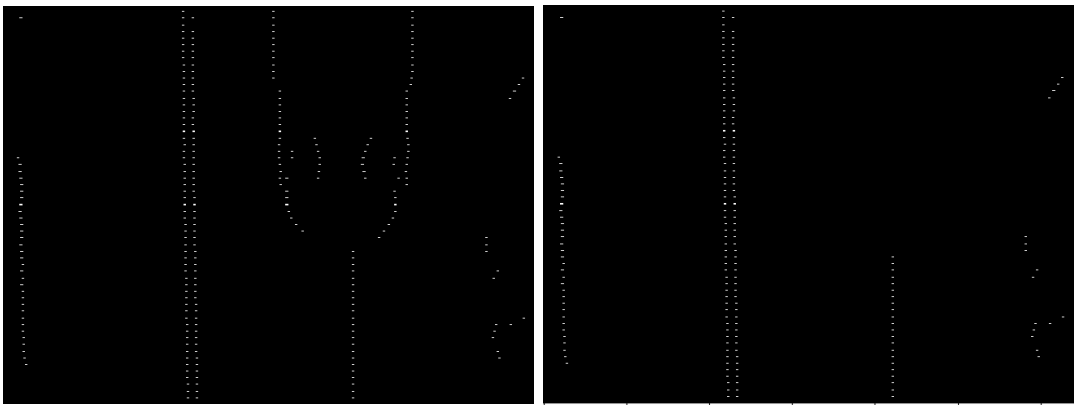


Figure 4.13: The total occluded area

For illustration purposes, BEV and modified BEV binary images obtained from the image in Figure 4.13 are given in Figure 4.14 (a) and (b), respectively.



(a) (b)

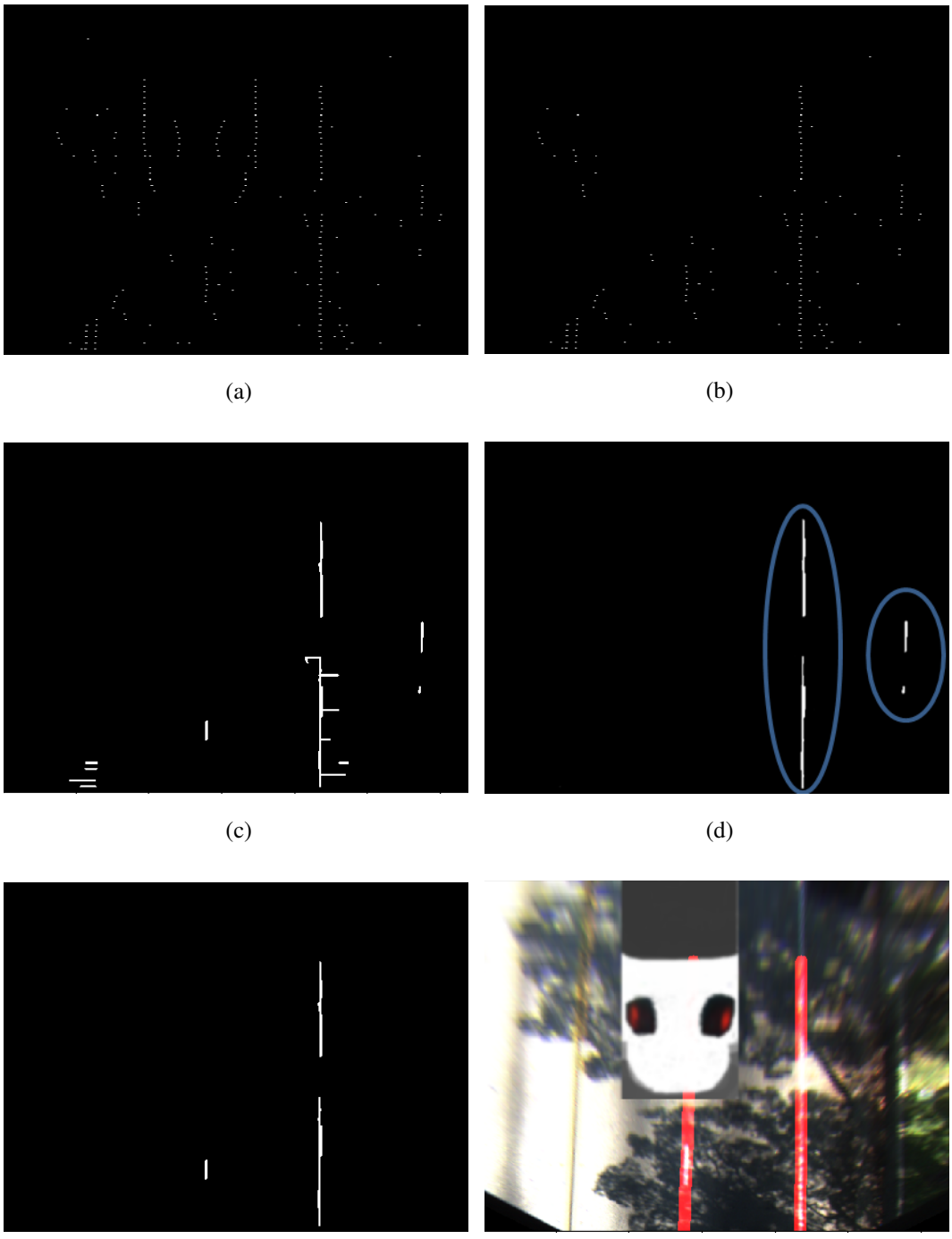
Figure 4.14: The BEV and modified BEV binary images

With the process explained above, the occluded area on the binary BEV image in Figure 4.14 (a) is removed to obtain the modified BEV image in Figure 4.14 (b). Thus, the noise extracted due to the vehicle does not exist on the modified BEV image.

## 4.5 Algorithm1-Step5

In the last step, line detection, specification of the best line cluster pair and the lane detection and tracking are performed as described in Section 3.2.3, 3.2.4 and 3.2.5 on the modified BEV image, respectively.

To show the effectiveness of the proposed sensor fusion algorithm from obtaining binary images to the lane detection and tracking result, Figure 4.15 - 4.19 (a), (b), (c), (d), (e), (f) which show binary BEV image, modified binary BEV image, hough lines, some merged hough lines in the circles, the best pair and the recognized lanes in the red color on the generated synthetic images with a vehicle based on CalTech database [20] are given, respectively. The generation of the synthetic images is described in Section 4.6.1 in a detailed way.

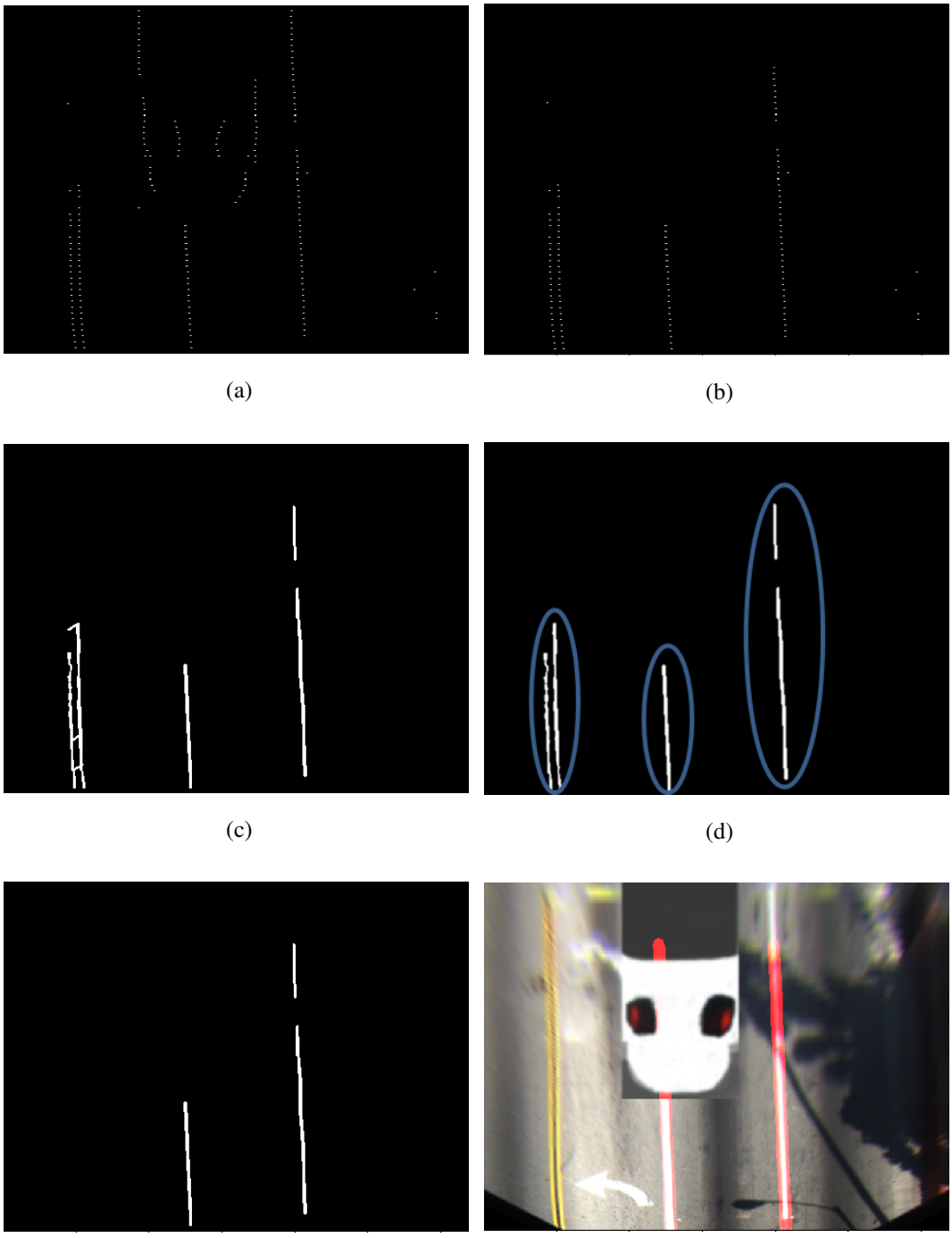


(a) (b)

(c) (d)

(e) (f)

Figure 4.15: The process of recognizing the lanes with the sensor fusion algorithm

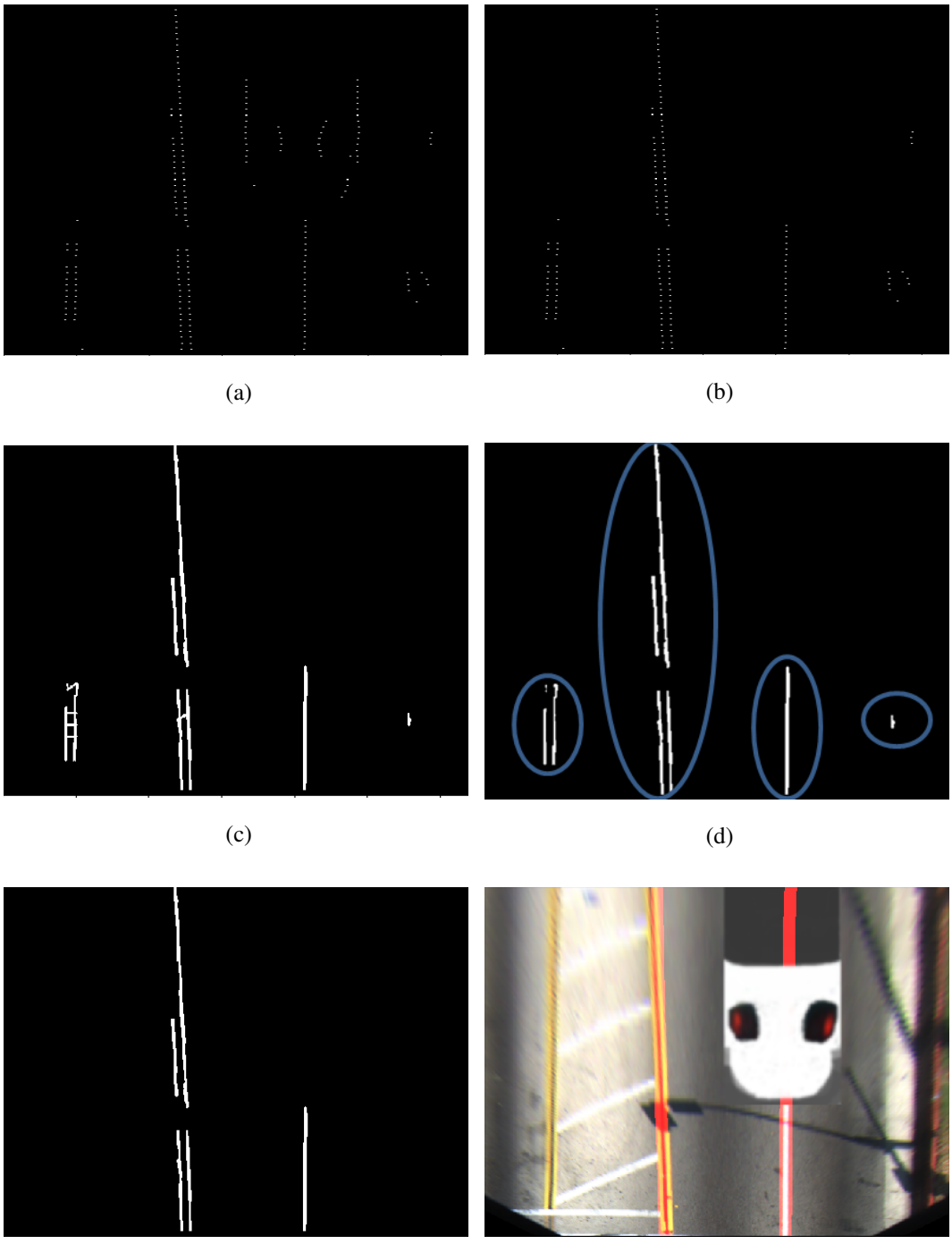


(a) (b)

(c) (d)

(e) (f)

Figure 4.16: The process of recognizing the lanes with the sensor fusion algorithm

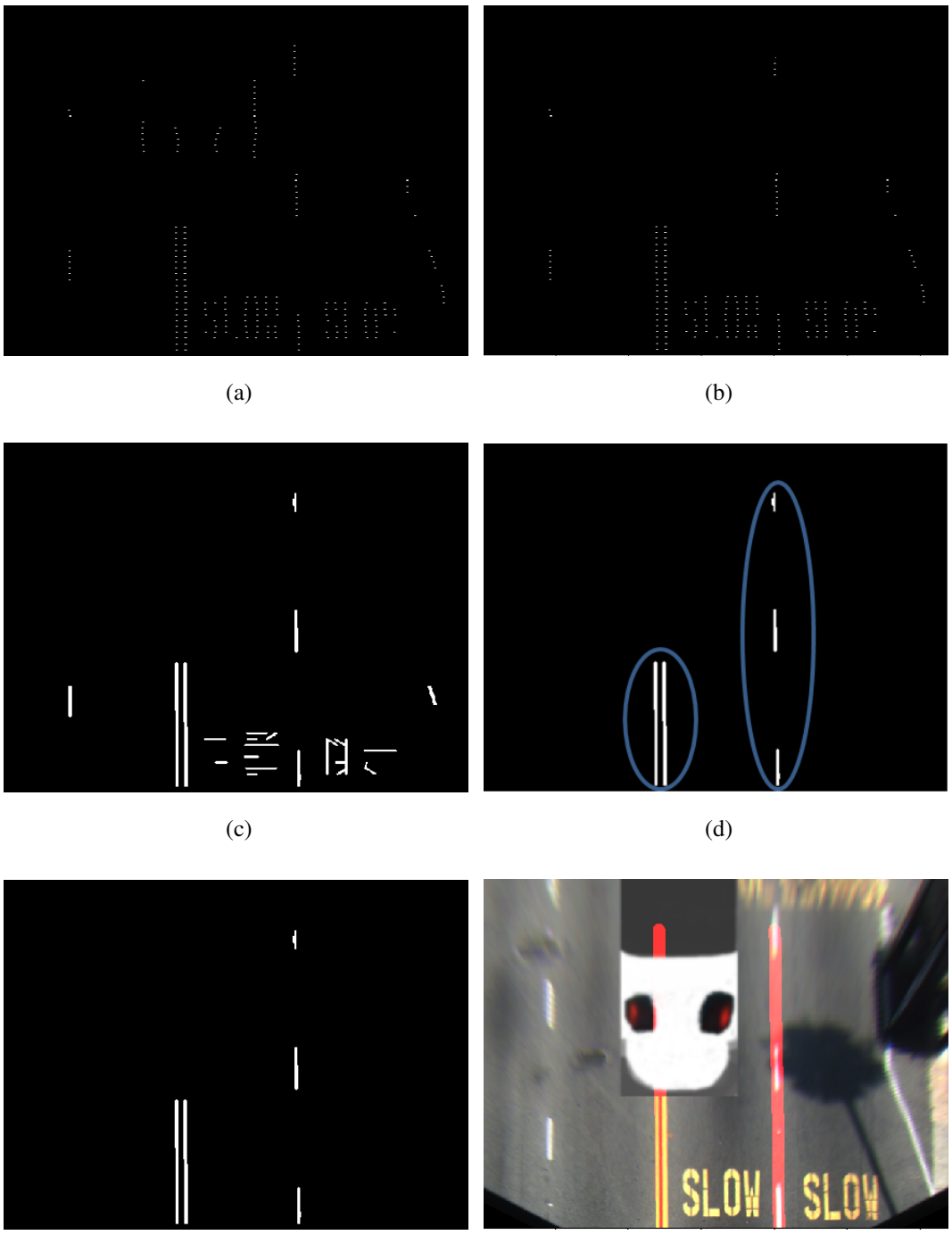


(a) (b)

(c) (d)

(e) (f)

Figure 4.17: The process of recognizing the lanes with the sensor fusion algorithm

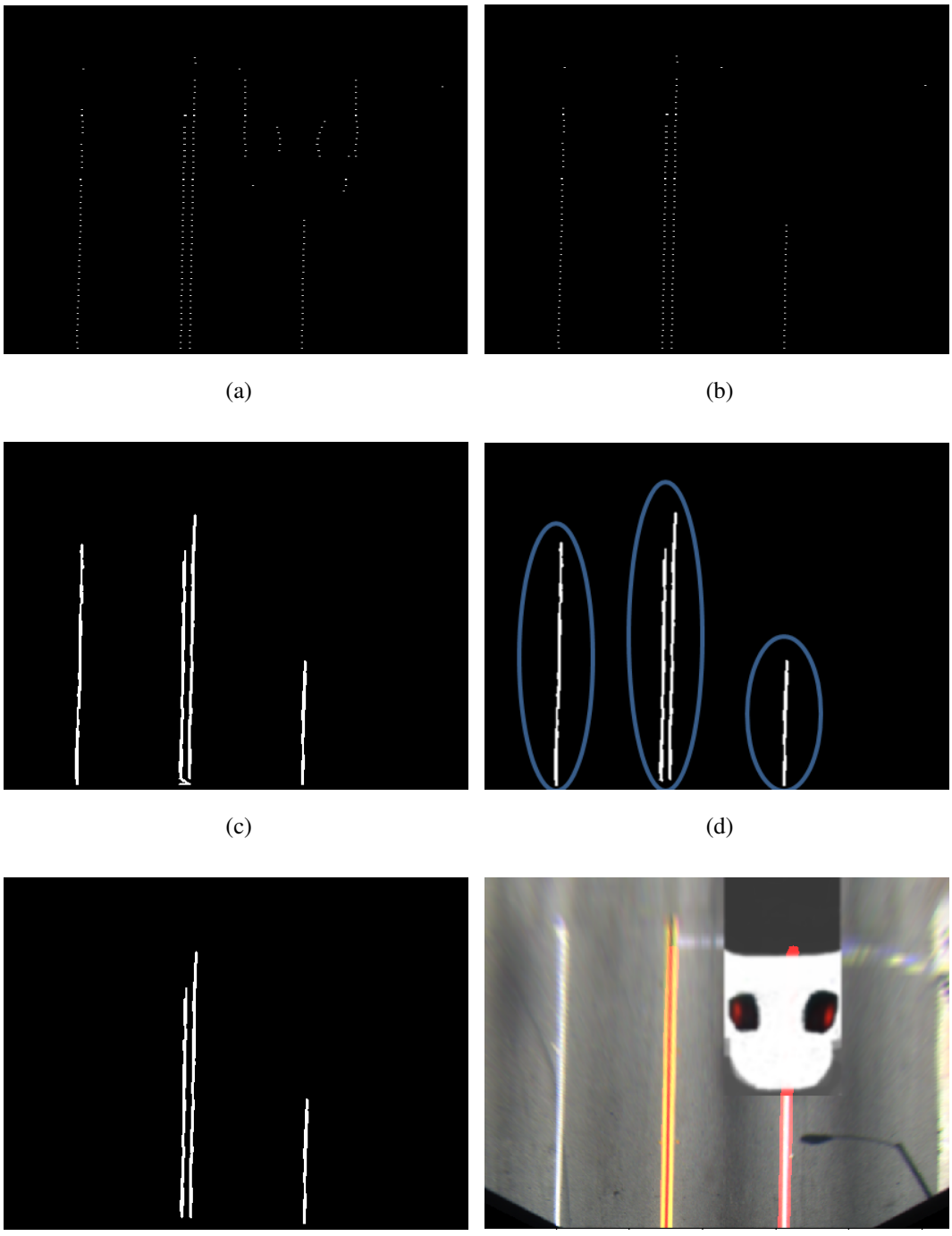



(a) (b)

(c) (d)

(e) (f)

Figure 4.18: The process of recognizing the lanes with the sensor fusion algorithm



(a) (b)

(c) (d)

(e) (f)

Figure 4.19: The process of recognizing the lanes with the sensor fusion algorithm

As can be seen from the Figure 4.15, correct lane recognition is performed in the case where lanes are affected by extreme shadows and there exists the manually extracted vehicle in the view. Moreover, the algorithm correctly recognize the lanes in Figure 4.16 - 4.19 where there exist the shadows, road signs, letters and the vehicles close to the lanes. Lastly, it is noted that the non-parallel lanes are correctly detected as can be seen in Figure 4.17.

## 4.6 Computational Evaluation

In this Chapter, quantitative and illustrated evaluation results of the proposed sensor fusion algorithm are given in a detailed way. To evaluate the proposed method quantitatively, we generate synthetic image data since there are no labeled data with compatible camera images and 2D LIDAR points in the existing literature.

### 4.6.1 Synthetic Data Generation

Synthetic data are generated by manipulating the real road images of the CalTech Cordova 1 dataset [20]. Firstly, perspective images of the dataset are transformed to the BEV images. After that, a real vehicle in one of the BEV images of the dataset is manually extracted. Then, the extracted vehicle is painted to white color. Furthermore, the white colored vehicle is located in different horizontal positions with respect to the lanes in the top part of the obtained BEV images. Figure 4.20 (a), (b) and (c) show three synthetic image examples with the vehicles in the left, center and right part of the image, respectively. In total, 750 synthetic images are generated for quantitative evaluation of the proposed sensor fusion method.

(a) (b) (c)

Figure 4.20: Generated synthetic images

### 4.6.2 Feature Extraction and Lane Detection

We next evaluate the improvement when utilizing the proposed method by comparing the lane detection and tracking results when using a BEV image and a modified BEV image. In order to provide a comprehensive comparison, we employ the proposed and described (NAND, 1-BT, Canny, SLT) feature extraction methods given in this thesis for obtaining binary BEV images. Then, modified binary BEV images are acquired as explained in Section 4.4 by considering that the LIDAR points hit the road level boundaries of the vehicles on the BEV images. Finally, lanes are detected and tracked as explained in Section 4.5 on the modified and binary BEV images.

### 4.6.3 Computational Results

The procedure explained in previous Section 4.6.2 is applied and the norms defined in equation 3.11 are utilized for the accuracy metric given in equation 3.12. Tables 4.4 - 4.6 show the average norm values (Norm) for the correctly detected lanes and the accuracy results (Acc) of the lane detection and tracking methods on the BEV and modified BEV images based on the Mean $L_1$-norm, Mean $L_2$-norm and $L_\infty$-norm. The accuracy results are given in the form of the percentage.

Table 4.4: Accuracy results and average norm values based on Mean $L_1$-norm

| Feature Extraction Methods | Dist $< 5$ | | Dist $< 8$ | |
|---|---|---|---|---|
| | BEV | Modified BEV | BEV | Modified BEV |
| Proposed method (Acc) | 60 | 81 | 66 | 89 |
| NAND (Acc) | 59 | 83 | 70 | 88 |
| 1-BT (Acc) | 67 | 81 | 77 | 86 |
| SLT (Acc) | 69 | 78 | 81 | 87 |
| Canny (Acc) | 57 | 70 | 65 | 76 |
| Proposed method (Norm) | 2.19 | 1.89 | 2.59 | 2.26 |
| NAND (Norm) | 2.57 | 1.99 | 3.14 | 2.26 |
| 1-BT (Norm) | 2.40 | 1.83 | 2.86 | 2.08 |
| SLT (Norm) | 2.44 | 2.08 | 2.97 | 2.50 |
| Canny (Norm) | 2.42 | 1.91 | 2.93 | 2.28 |

Table 4.5: Accuracy results and average norm values based on Mean $L_2$-norm

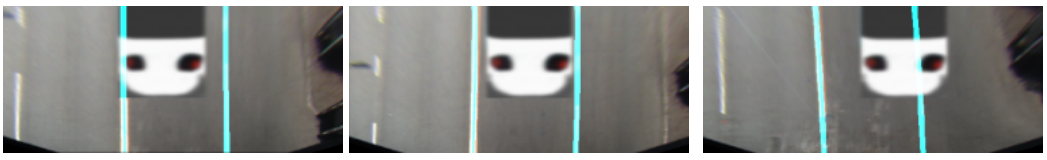
| Feature Extraction Methods | Dist $< 5$ | | Dist $< 8$ | |
|---|---|---|---|---|
| | BEV | Modified BEV | BEV | Modified BEV |
| Proposed method (Acc) | 57 | 79 | 65 | 88 |
| NAND (Acc) | 53 | 81 | 66 | 88 |
| 1-BT (Acc) | 63 | 80 | 75 | 86 |
| SLT (Acc) | 64 | 75 | 78 | 85 |
| Canny (Acc) | 52 | 68 | 63 | 76 |
| Proposed method (Norm) | 2.42 | 2.02 | 2.86 | 2.44 |
| NAND (Norm) | 2.83 | 2.19 | 3.49 | 2.51 |
| 1-BT (Norm) | 2.63 | 1.98 | 3.19 | 2.28 |
| SLT (Norm) | 2.67 | 2.28 | 3.31 | 2.73 |
| Canny (Norm) | 2.60 | 2.06 | 3.20 | 2.47 |

Table 4.6: Accuracy results and average norm values based on $L_\infty$-norm

| Feature Extraction Methods | Dist $< 5$ | | Dist $< 8$ | |
|---|---|---|---|---|
| | BEV | Modified BEV | BEV | Modified BEV |
| Proposed method (Acc) | 34 | 67 | 53 | 83 |
| NAND (Acc) | 20 | 61 | 41 | 80 |
| 1-BT (Acc) | 32 | 70 | 57 | 83 |
| SLT (Acc) | 34 | 52 | 58 | 73 |
| Canny (Acc) | 28 | 62 | 49 | 77 |
| Proposed method (Norm) | 3.33 | 2.93 | 4.37 | 3.59 |
| NAND (Norm) | 3.64 | 3.31 | 5.1 | 4.02 |
| 1-BT (Norm) | 3.51 | 2.95 | 4.78 | 3.46 |
| SLT (Norm) | 3.56 | 3.40 | 4.72 | 4.27 |
| Canny (Norm) | 3.47 | 2.95 | 4.62 | 3.65 |

According to the Tables 4.4-4.6, better accuracy results and smaller average norm values for the correctly detected lanes are achieved when using the modified BEV image for all combinations of the feature extraction methods, the distance bounds and the norms. This superiority arises from the modified BEV image, where pixels which can be considered as noise due to objects are cleared.

For illustration purposes, a distinctive selection of lane detection and tracking results on the modified BEV image are given in Figure 4.21.



(a) (b) (c)

Figure 4.21: Detected lanes for representative synthetic images

## 4.7 Experimental Evaluation

In this Section, the proposed sensor fusion algorithm between a camera and 2D LIDAR for lane recognition are evaluated on the autonomous vehicle kit [33] and the utilized hardware and software are described in a detailed way.

### 4.7.1 Hardware Setup & Software

In this Section, utilized hardware setup and software for the proposed sensor fusion algorithm are described in a detailed way.

#### 4.7.1.1 Hardware Setup

To implement the proposed algorithm experimentally, Jetson RACECAR [33] which mainly has NVIDIA Jetson TX2 as developer board, RPLIDAR A2M6 for 2D LIDAR data and ZED camera for RGB image data is utilized in the scope of the thesis.
An image of Jetson RACECAR with the main products are given in Figure 4.22.

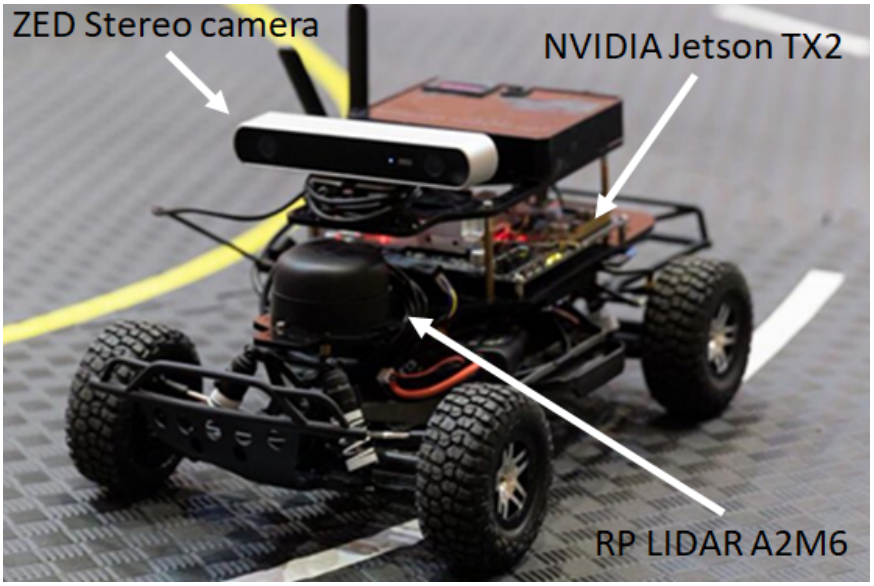



Figure 4.22: Jetson RACECAR with ZED Stereo camera, NVIDIA Jetson TX2 and RP LIDAR A2M6

Main specifications of the NVIDIA Jetson TX2 developer kit, RPLIDAR A2M6 and ZED camera are given in Table 4.7, 4.8 and 4.9, respectively.

Table 4.7: Main specifications of the NVIDIA Jetson TX2 developer kit

| | |
|---|---|
| GPU | NVIDIA Pascal, 256 CUDA Core |
| CPU | HMP Dual Denver 2/2 MB L2 +<br>Quad ARM A57/2 MB L2 |
| Video | 4K x 2K 60 Hz Encode (HEVC)<br>4K x 2K 60 Hz Decode (12-Bit Support) |
| Memory | 8 GB 128 bit LPDDR4, 58.3 GB/s |
| Display | 2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4 |
| Others | CAN, UART, SPI, I2C, I2S, GPIOs, USB 3.0 + USB 2.0 |

Table 4.8: Main specifications of the RPLIDAR A2M6 2D LIDAR

| Measurement | Unit | Minimum | Typical | Maximum |
|---|---|---|---|---|
| Distance Range | $m$ | 0.2 | - | 16 |
| Angular Range | Degree | - | 0 - 360 | - |
| Angular Resolution | Degree | 0.45 | 0.9 | 1.35 |
| Sample Frequency | Hz | 2000 | 4000 | 4100 |
| Scan Rate | Hz | 5 | 10 | 15 |

Distance range and angular resolution performances are based on white objects with 70 % reflectivity and 10 Hz scan rate, respectively.

Table 4.9: Main specifications of the ZED Camera

| | Video Mode | Frames per second | Output Resolution |
|---|---|---|---|
| Video | 2.2K | 15 | 4416x1242 |
| | 1080p | 30 | 3840x1080 |
| | 720p | 60 | 2560x720 |
| | WVGA | 100 | 1344x376 |

#### 4.7.1.2 Software

The proposed algorithms are developed in Python language on the autonomous vehicle kit [33] for the experimental tests through the features of Robotic Operating System (ROS).

ROS is actually not an operating system, instead it is a free and open-source framework consisting of set of libraries, tools, etc. to facilitate writing programs in supported languages such as C++, Python, etc. on robotic hardware [85, 86, 87]. Moreover, it also shares the features of an operating system such as hardware abstraction, device drivers, package management, etc.

The development of ROS began in the mid-2000s with the STAIR [86] project at Stanford University and the Personal Robots Program [87] at Willow Garage to satisfy some challenges such as sensor integrations, device drivers, communications, message scheduling etc. Now, ROS has been increasingly attracting attention in all over the world since it is applicable to the wide variety of robotic platforms and it handles the specified challenges which can be daunting at the beginning of robot software development.

ROS implementation is comprised of four primary notions: nodes, topics, messages, ROS master. Nodes are responsible for performing computational tasks and transfer data between each other based on publish/subscribe mechanism via topics. A node which sends/takes data publishes/subscribes to an appropriate topic for data transfer on the host operating systems. Messages are the related data and standard data types such as floating point, string, integer, etc. are supported. Lastly, ROS master informs nodes to communicate peer to peer when new data are available.

In the scope of the thesis, the schematic of the ROS network is given in Figure 4.23. In the figure, there exist three different nodes: 'camera', 'LIDAR' and 'sensorFusion'. The nodes, 'camera' and 'LIDAR', publishes the RGB camera data ('Camera/RGB') and 2D LIDAR data ('LIDAR/LaserScan') to the topics, respectively. The node, 'sensorFusion', subscribes to those topics for the camera and 2D LIDAR data (message). Next, the node generates the reference trajectory and computes the lateral and heading errors by processing the camera and 2D LIDAR data. Lastly, the 'sensorFusion' node publishes the generated arc-spline reference trajectory parameters, lateral error and heading error to the 'Lane/outputs' topic.

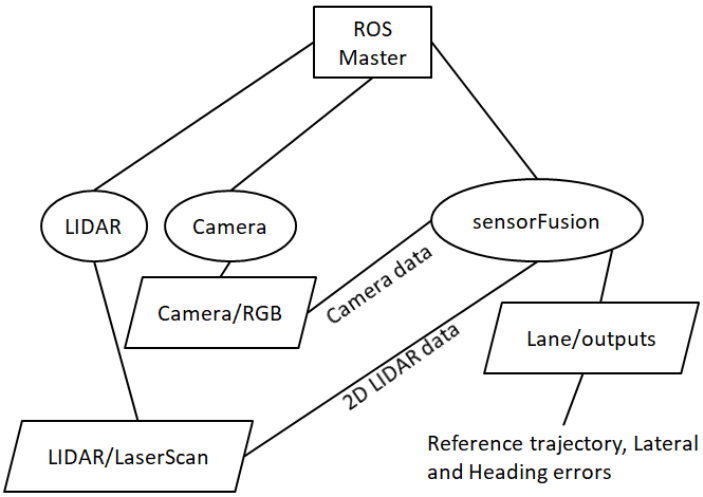
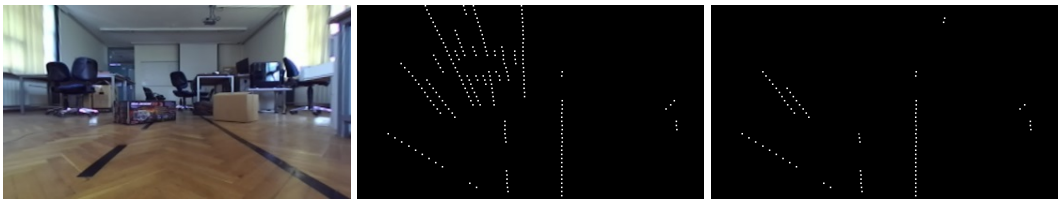



Figure 4.23: ROS network graph of the proposed sensor fusion method

### 4.7.2 Experimental Tests

To evaluate the proposed sensor fusion algorithm, firstly some boxes are put on/-between the manually constructed lanes on the ground. Then, the proposed sensor fusion algorithm is implemented and illustrative experimental results are obtained.
An example of experimentally obtained binary BEV image and modified binary BEV image from the perspective image with the obstacles in Figure 4.24 (a) are given in Figure 4.24 (b) and (c), respectively.



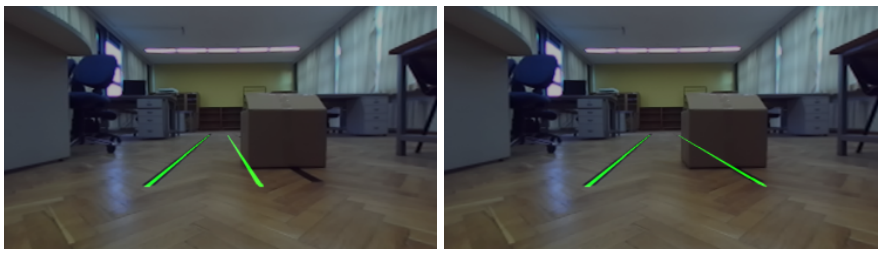
(a) (b) (c)

Figure 4.24: The BEV and modified BEV binary images

As can be seen from Figure 4.24, the noise arising from the obstacles is cleared on the modified binary BEV image.
Experimental lane detection and tracking results are given in Figure 4.25 - 4.27.

(a) (b)

Figure 4.25: Lane detection illustrations without and with 2D LIDAR, respectively



(a) (b)

Figure 4.26: Lane detection illustrations without and with 2D LIDAR, respectively

(a) (b)

Figure 4.27: Lane detection illustrations without and with 2D LIDAR, respectively

The lane detection and tracking results show that the benefit of using 2D LIDAR in the proposed algorithm. As can be seen in the figures, the proposed algorithm with sensor fusion correctly detects the lanes even when there exist objects near the lanes in the ROI.

# CHAPTER 5

# COMPUTATION OF OUTPUT SIGNALS FOR CONTROL APPLICATIONS

In this Chapter, the computations of reference trajectory generation, heading error and lateral error are described in a detailed way after recognizing the lanes.

## 5.1 Reference Trajectory Generation

In this chapter, reference trajectory based on arc-spline model is generated for lane keeping purposes. To specify the arc-spline parameters, obtained polynomial coefficients $(a, b_{\text{l}}, c_{\text{l}}, b_{\text{r}}, c_{\text{r}})$ in Section 4.5 are utilized. Remember that $(a, b_{\text{l}}, c_{\text{l}})$ and $(a, b_{\text{r}}, c_{\text{r}})$ are the left and right lane model parameters.

Let $a, b, c$ are the parameters of the $y = ax^2 + bx + c$ polynomial, then arc-spline model parameters can be computed as:

$$\begin{aligned} P_{\text{arc,s}} &= c \\ \psi_{\text{arc,s}} &= \arctan\left(\left.\frac{dy}{dx}\right|_{x\to 0^+}\right) \\ k_{\text{arc,i}} &= \frac{2a}{(1+b^2)^{3/2}} \\ k_{\text{arc,f}} &= \frac{2a}{(1+(2aN_{\text{row}}+b)^2)^{3/2}} \\ S_{\text{arc}} &\approx \sum_{i=0}^{(N_{\text{row}}/\Delta x)-1} \left[y(\Delta x\cdot(i+1)) - y(\Delta x\cdot i)\right]^2 \end{aligned} \tag{5.1}$$

where $\Delta x$ is the sampling intervals of $x$. For $S_{\text{arc}}$, we take samples of the polynomial at specified intervals $\Delta x$ and compute the Euclidean distance between the consecutive sampling points.

Hereby, we first compute the arc-spline parameters as in 5.1 from the recognized

left and right lanes, then take the average for reference trajectory. For illustration purposes, a generated reference trajectory from an image in CalTech database [20] is given in Figure 5.1 in green color.

Figure 5.1: Generated reference trajectory

## 5.2 Heading Error

The heading error $\psi$ is the angle between the desired path of the vehicle and the vehicle's heading on the origin of the CCF and it is represented as in Figure 5.2 (a). The heading error computation is based on the assumption that the road is flat and the lane is linear in the near field of the vehicle. The computation is realized according to the vehicle coordinate frame (VCF), which has zero tilt angle $\phi$ (i.e. whose y-coordinate is normal to the road plane), whose z-coordinate is in the direction of the vehicle's heading and where the origin is on the origin of the CCF as given in Figure 5.2 (b). Note that CCF is the tilted version of VCF as in the Figure 5.2 (b).

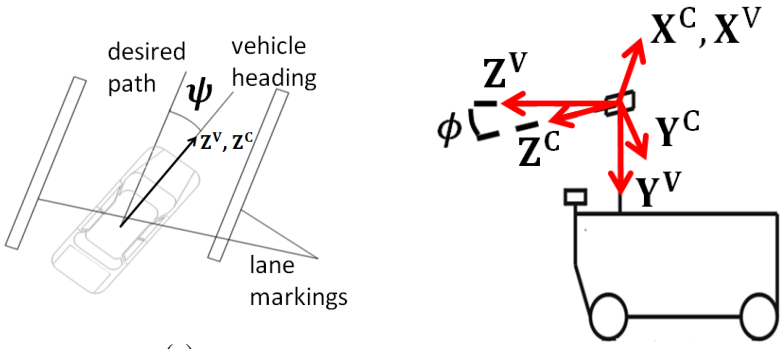



Figure 5.2: Heading error representation

The rotation of the CCF with respect to the VCF is represented by the matrix $R_{\mathrm{c}}^{\mathrm{v}}$ and it is the rotation matrix by the tilt angle $\phi$ estimated in equation 4.18 in the x-axis of a cartesian coordinate frame, $R_{\mathrm{x}}(\phi)$. To compute the heading error, we first choose two pixel positions $(u_1, v_1)$ and $(u_2, v_2)$ on the detected lane in the near field of the vehicle according to the ICF. The selected pixel positions on the left lane are shown in Figure 5.3.

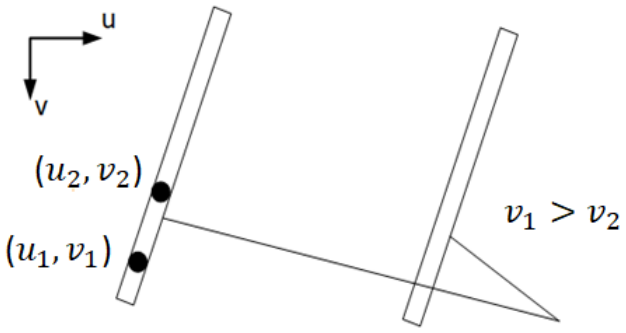



Figure 5.3: Selected pixel positions for heading error

Then, the heading error is evaluated as

$$\psi = \arctan \frac{x_2^{\mathrm{v}} - x_1^{\mathrm{v}}}{z_2^{\mathrm{v}} - z_1^{\mathrm{v}}} \tag{5.2}$$

where $x_1^{\mathrm{v}}$, $x_2^{\mathrm{v}}$ and $z_1^{\mathrm{v}}$, $z_2^{\mathrm{v}}$ are the x-coordinates and z-coordinates of the corresponding pixel positions according to the VCF. The coordinates are computed from the following equality with the aid of the height of the camera, $h_{\mathrm{c}}$, estimated in equation 4.17.

$$\begin{bmatrix} x^{\mathrm{v}}/z^{\mathrm{c}} \\ y^{\mathrm{v}}/z^{\mathrm{c}} \\ z^{\mathrm{v}}/z^{\mathrm{c}} \end{bmatrix} = R_{\mathrm{c}}^{\mathrm{v}} K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{5.3}$$

Since $y^{\mathrm{v}} = h_{\mathrm{c}}$ is the known distance of the camera from the road, we first compute $z^{\mathrm{c}}$, then $x^{\mathrm{v}}$ and $z^{\mathrm{v}}$. Using the resulting parameters from 5.3, the heading error follows from 5.2.

## 5.3 Lateral Error

The lateral error is the error between the desired path of the vehicle and the origin of the VCF in the direction of x-axis of the VCF with zero heading error. It is illustrated in Figure 5.4.

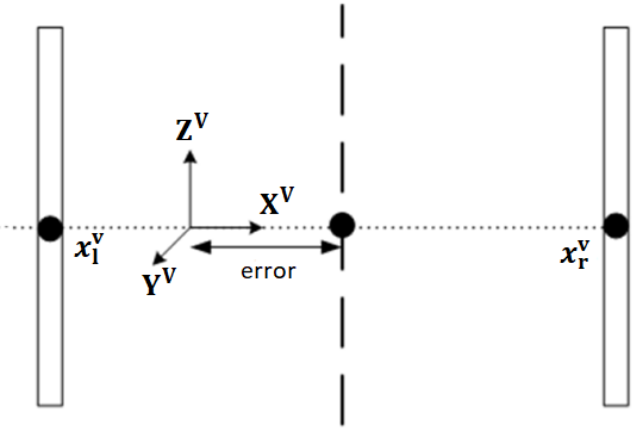



Figure 5.4: Lateral error illustration

After computing the heading error using equation 5.2, the lateral error can be computed in the VCF with zero heading error as:

$$\frac{x_{\mathrm{l}}^{\mathrm{v}} + x_{\mathrm{r}}^{\mathrm{v}}}{2} \tag{5.4}$$

where $x_{\mathrm{l}}^{\mathrm{v}}$ and $x_{\mathrm{r}}^{\mathrm{v}}$ are the left and right x-coordinates corresponding to the nearest detected left and right lane pixel positions, respectively according to the VCF with zero heading error. These coordinates are computed from the following relation with the aid of the height of the camera, $h_{\mathrm{c}}$.

$$\begin{bmatrix} x^{\mathrm{v}}/z^{\mathrm{c}} \\ y^{\mathrm{v}}/z^{\mathrm{c}} \\ z^{\mathrm{v}}/z^{\mathrm{c}} \end{bmatrix} = R_{\mathrm{y}}(\psi) R_{\mathrm{c}}^{\mathrm{v}} K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{5.5}$$

Since $y^{\mathrm{v}} = h_{\mathrm{c}}$ is the known distance of the camera from the road, we first compute $z^{\mathrm{c}}$, then $x^{\mathrm{v}}$.

# CHAPTER 6

# CONCLUSIONS

Lane Departure Warning System (LDWS) and Lane Keeping Assist System (LKAS) are two examples of advanced driver-assistance systems. LDWS is a warning system informing drivers through vision, acoustic alarm, vibration of seat, etc. if the vehicle is on the way of crossing its current lanes without intention of the driver. LKAS is a little bit more advanced system which warns and supports the driver by keeping the vehicle in the center of the lane through controlling the lateral movement of the vehicle. LDWS and LKAS designs mostly rely on lane detection in the first phase and computation of output signals for warning drivers (LDWS) or controlling vehicles (LKAS) in the second phase. To compute the output signals correctly, the lane detection must be performed correctly and expectations from the lane detection phase are being robustness against noise, shadows, being adaptable to different lighting conditions, ignoring objects in the view, etc. However, the recognition of the lanes in different conditions is a highly challenging problem.

This thesis contributes to the literature in several ways for the stated problems above and accordingly, the conclusions are as follows.

Firstly, it proposes a novel lane detection and tracking algorithm which is robust to noise due to shadows, cracks on the roads, etc. and which is adaptable to the different lighting conditions. The quantitative comparison shows the superiority of the proposed method over the representative existing methods in the cases where there exist different lighting conditions, noises, etc. In addition to that, the illustrated lane detection and tracking results show the efficacy of the proposed method in the cases where lanes are affected by extreme shadows, close to the other vehicles, road signs, curbs, etc.

Secondly, the thesis proposes a novel sensor fusion method based on a camera and 2D

LIDAR to address the problems of objects in camera view, impairing the lane detection accuracy. The main idea is to identify objects via 2D LIDAR and clear the object pixels, which can be considered as noise, on a BEV image. The results show that the sensor fusion method increases the lane detection and tracking accuracy and robustness against the objects in the view. Additionally, the thesis proposes solutions for projecting LIDAR points to road plane to evaluate all object pixels as background on the BEV image. To do that, the thesis also derives formulations for estimating camera height from the road plane and tilt angle of the camera with respect to the road plane. Illustrated results of the projection of 2D LIDAR points to the road plane prove the correctness of the estimation of the camera height, tilt angle and the formulation for the projection of the 2D LIDAR points to the road plane.

Thirdly, the computations of reference trajectory generation, heading error and lateral error are described for control applications in a detailed way after recognizing the lanes. Lastly, a novel algorithm for the generation of synthetic 2D LIDAR data is proposed and a comparative study of different 2D LIDAR data segmentation methods based on the generated synthetic data is presented. The obtained accuracy results show the superiority of distance varying radially bounded nearest neighbor (dvRBNN) over the other methods, which are fixed breakpoint detection (FBD), adaptive breakpoint detection (ABD), nearest neighbor (NN), $k$ nearest neighbor ($k$NN) and radially bounded nearest neighbor (RBNN), for all different test cases. This superiority arises from its adaptive threshold based on the distance value of 2D LIDAR and the robustness against the occlusions of small parts of the objects by other objects. Regarding the computational complexity, FBD and ABD prove to be the most computationally efficient methods since there is only one query per LIDAR point. As for NN, $k$NN, RBNN and dvRBNN, the algorithm consists of constructing and searching a graph which can be done by asking queries from a LIDAR point to all other LIDAR points in the worst case.

# REFERENCES


[1] K. A. Brookhuis, D. De Waard, and W. H. Janssen, "Behavioural impacts of advanced driver assistance systems–an overview," *European Journal of Transport and Infrastructure Research*, vol. 1, no. 3, 2019.

[2] I. Daza, L. Bergasa, S. Bronte, J. Yebes, J. Almazán, and R. Arroyo, "Fusion of optimized indicators from advanced driver assistance systems (adas) for driver drowsiness detection," *Sensors*, vol. 14, no. 1, pp. 1106–1131, 2014.

[3] U. Z. A. Hamid, F. R. A. Zakuan, K. A. Zulkepli, M. Z. Azmi, H. Zamzuri, M. A. A. Rahman, and M. A. Zakaria, "Autonomous emergency braking system with potential field risk assessment for frontal collision mitigation," in *2017 IEEE Conference on Systems, Process and Control (ICSPC)*, pp. 71–76, IEEE, 2017.

[4] I. J. Reagan, J. B. Cicchino, L. B. Kerfoot, and R. A. Weast, "Crash avoidance and driver assistance technologies–are they used?," *Transportation research part F: traffic psychology and behaviour*, vol. 52, pp. 176–190, 2018.

[5] S. P. Narote, P. N. Bhujbal, A. S. Narote, and D. M. Dhane, "A review of recent advances in lane detection and departure warning system," *Pattern Recognition*, vol. 73, pp. 216–234, 2018.

[6] T.-T. Tran, C.-S. Bae, Y.-N. Kim, H.-M. Cho, and S.-B. Cho, "An adaptive method for lane marking detection based on hsi color model," in *International Conference on Intelligent Computing*, pp. 304–311, Springer, 2010.

[7] D.-K. Lee, J.-S. Shin, J.-H. Jung, S.-J. Park, S.-J. Oh, and I.-S. Lee, "Real-time lane detection and tracking system using simple filter and kalman filter," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 275–277, IEEE, 2017.

[8] J.-G. Kim, J.-H. Yoo, and J.-C. Koo, "Road and lane detection using stereo camera," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 649–652, IEEE, 2018.

[9] C. Yuan, H. Chen, J. Liu, D. Zhu, and Y. Xu, "Robust lane detection for complicated road environment based on normal map," *IEEE Access*, vol. 6, pp. 49679–49689, 2018.

[10] W. Li, F. Qu, Y. Wang, L. Wang, and Y. Chen, "A robust lane detection method based on hyperbolic model," *Soft Computing*, pp. 1–14, 2018.

[11] Y. Son, E. S. Lee, and D. Kum, "Robust multi-lane detection and tracking using adaptive threshold and lane classification," *Machine Vision and Applications*, vol. 30, no. 1, pp. 111–124, 2019.

[12] C. Lee and J.-H. Moon, "Robust lane detection and tracking for real-time applications," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–6, 2018.

[13] J. Piao and H. Shin, "Robust hypothesis generation method using binary blob analysis for multi-lane detection," *IET Image Processing*, vol. 11, no. 12, pp. 1210–1218, 2017.

[14] K. Manoharan and P. Daniel, "Image processing-based framework for continuous lane recognition in mountainous roads for driver assistance system," *Journal of Electronic Imaging*, vol. 26, no. 6, p. 063011, 2017.

[15] R. F. Berriel, E. de Aguiar, A. F. De Souza, and T. Oliveira-Santos, "Ego-lane analysis system (elas): Dataset and algorithms," *Image and Vision Computing*, vol. 68, pp. 64–75, 2017.

[16] A. Küçükmanisa, R. Duvar, and O. Urhan, "Real-time lane marking detection using modified 1-bit transform based pre-processing," in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2017.

[17] O. Kumtepe, G. B. Akar, and E. Yuncu, "Driver aggressiveness detection via multisensory data fusion," *EURASIP Journal on Image and Video Processing*, vol. 2016, no. 1, p. 5, 2016.

[18] Ö. KUMTEPE, "Driver aggressiveness analysis using multisensory data fusion," Master's thesis, MIDDLE EAST TECHNICAL UNIVERSITY, 2016.

[19] U. Ozgunalp and N. Dahnoun, "Robust lane detection & tracking based on novel feature extraction and lane categorization," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8129–8133, IEEE, 2014.

[20] M. Aly, "Real time detection of lane markers in urban streets," in *2008 IEEE Intelligent Vehicles Symposium*, pp. 7–12, IEEE, 2008.

[21] G. Liu, S. Li, and W. Liu, "Lane detection algorithm based on local feature extraction," in *2013 Chinese Automation Congress*, pp. 59–64, IEEE, 2013.

[22] G. Küçükyildiz and H. Ocak, "Development and optimization of a dsp-based real-time lane detection algorithm on a mobile platform," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 22, no. 6, pp. 1484–1500, 2014.

[23] X. Li, X. Fang, C. Wang, and W. Zhang, "Lane detection and tracking using a parallel-snake approach," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 3-4, pp. 597–609, 2015.

[24] F. A. Siddiqui, S. Amir, M. Asif, and Z. A. Ali, "Lane tracking and autonomous cruise control for automatic highway system," in *2011 IEEE 19th Signal Processing and Communications Applications Conference (SIU)*, pp. 542–545, IEEE, 2011.

[25] S. K. Gehrig, A. Gern, S. Heinrich, and B. Woltermann, "Lane recognition on poorly structured roads-the bots dot problem in california," in *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, pp. 67–71, IEEE, 2002.

[26] T. Youjin, C. Wei, L. Xingguang, and C. Lei, "A robust lane detection method based on vanishing point estimation," *Procedia computer science*, vol. 131, pp. 354–360, 2018.

[27] S. Zhu, J. Wang, T. Yu, and J. Wang, "A method of lane detection and tracking for expressway based on ransac," in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, pp. 62–66, IEEE, 2017.

[28] Y. Y. Ye, X. L. Hao, and H. J. Chen, "Lane detection method based on lane structural analysis and cnns," *IET Intelligent Transport Systems*, vol. 12, no. 6, pp. 513–520, 2018.

[29] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 286–291, IEEE, 2018.

[30] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, and V. N. Murali, "Deeplanes: End-to-end lane position estimation using deep neural networksa," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 38–45, 2016.

[31] S. Lee, J. Kim, J. Shin Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. Seok Hong, S.-H. Han, and I. So Kweon, "Vpgnet: Vanishing point guided network for lane and road marking detection and recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1947–1955, 2017.

[32] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nüchter, "A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 2, pp. 540–555, 2013.

[33] "Openzeka vehicle configuration." `https://openzeka.com/marc/arac-konfigurasyonu/`. Accessed: 2018.

[34] S. J. Miller, "The method of least squares," *Mathematics Department Brown University*, vol. 114, 2006.

[35] E. W. Weisstein, "Rotation matrix," 2003.

[36] J. P. Fillmore, "A note on rotation matrices," *IEEE Computer Graphics and Applications*, vol. 4, no. 2, pp. 30–33, 1984.

[37] R. M. Murray, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[38] F. A. Valentine, "Convex sets," *New York*, vol. 1964, 1964.

[39] T. Veit, J.-P. Tarel, P. Nicolle, and P. Charbonnier, "Evaluation of road marking feature extraction," in *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pp. 174–181, IEEE, 2008.

[40] C. Saravanan, "Color image to grayscale image conversion," in *2010 Second International Conference on Computer Engineering and Applications*, vol. 2, pp. 196–199, IEEE, 2010.

[41] D. Zhang, B. Fang, W. Yang, X. Luo, and Y. Tang, "Robust inverse perspective mapping based on vanishing point," in *Proceedings 2014 IEEE International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pp. 458–463, IEEE, 2014.

[42] R. I. Hartley, "In defence of the 8-point algorithm," in *Proceedings of IEEE international conference on computer vision*, pp. 1064–1070, IEEE, 1995.

[43] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer vision, graphics, and image processing*, vol. 44, no. 1, pp. 87–116, 1988.

[44] Y. Yeniaydin and K. W. Schmidt, "A lane detection algorithm based on reliable lane markings," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2018.

[45] Y. Yeniaydin and K. W. Schmidt, "Robust lane recognition based on arc splines," in *International Conference and Exhibition on Digitial Transformation and Smart Systems*, pp. 1–4, 2018.

[46] A. Mammeri, G. Lu, and A. Boukerche, "Design of lane keeping assist system for autonomous vehicles," in *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, IEEE, 2015.

[47] T. Lindeberg, "Scale-space for discrete signals," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 3, pp. 234–254, 1990.

[48] K. Baass, "Use of clothoid templates in highway design," tech. rep., 1982.

[49] E. Jahnke, "Tables of functions with formulae and curves," *New York: Dover Publications,| c1945, 4th ed.*, 1945.

[50] P. J. Davis, W. Gautschi, and A. Iserles, *Spirals: from Theodorus to chaos*. AK Peters Ltd, 1993.

[51] D. Meek and D. Walton, "An arc spline approximation to a clothoid," *Journal of Computational and Applied Mathematics*, vol. 170, no. 1, pp. 59–77, 2004.

[52] D. Meek and D. Walton, "A note on finding clothoids," *Journal of Computational and Applied Mathematics*, vol. 170, no. 2, pp. 433–453, 2004.

[53] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, 2000.

[54] J. Heikkila, O. Silven, *et al.*, "A four-step camera calibration procedure with implicit image correction," in *cvpr*, vol. 97, p. 1106, 1997.

[55] J.-Y. Bouguet, "Camera calibration toolbox for matlab," *http://www. vision. caltech. edu/bouguetj/calib_doc/index. html*, 2004.

[56] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[57] D. Reiser, M. Vázquez-Arellano, D. S. Paraforos, M. Garrido-Izard, and H. W. Griepentrog, "Iterative individual plant clustering in maize with assembled 2d lidar data," *Computers in Industry*, vol. 99, pp. 42–52, 2018.

[58] Y.-T. Wang, C.-C. Peng, A. Ravankar, and A. Ravankar, "A single lidar-based feature fusion indoor localization algorithm," *Sensors*, vol. 18, no. 4, p. 1294, 2018.

[59] Y. Peng, D. Qu, Y. Zhong, S. Xie, J. Luo, and J. Gu, "The obstacle detection and obstacle avoidance algorithm based on 2-d lidar," in *2015 IEEE International Conference on Information and Automation*, pp. 1648–1653, IEEE, 2015.

[60] A. N. Catapang and M. Ramos, "Obstacle detection using a 2d lidar system for an autonomous vehicle," in *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, pp. 441–445, IEEE, 2016.

[61] B.-h. Kang and S.-i. Choi, "Pothole detection system using 2d lidar and camera," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 744–746, IEEE, 2017.

[62] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for efficient segmentation of 3d laser data," in *2008 IEEE International Conference on Robotics and Automation*, pp. 4043–4048, IEEE, 2008.

[63] Y. Choe, S. Ahn, and M. J. Chung, "Fast point cloud segmentation for an intelligent vehicle using sweeping 2d laser scanners," in *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 38–43, IEEE, 2012.

[64] J. Niu, J. Lu, M. Xu, P. Lv, and X. Zhao, "Robust lane detection using two-stage feature extraction with curve fitting," *Pattern Recognition*, vol. 59, pp. 225–233, 2016.

[65] J. Ruyi, K. Reinhard, V. Tobi, and W. Shigang, "Lane detection and tracking using a new lane model and distance transform," *Machine vision and applications*, vol. 22, no. 4, pp. 721–737, 2011.

[66] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[67] G. Csaba, L. Somlyai, and Z. Vámossy, "Mobil robot navigation using 2d lidar," in *2018 IEEE 16th world symposium on applied machine intelligence and informatics (SAMI)*, pp. 000143–000148, IEEE, 2018.

[68] S. Karakaya, G. Küçükyildiz, and O. Hasan, "Detection of obstacle-free gaps for mobile robot applications using 2-d lidar data," *Uluslararası Doğa ve Mühendislik Bilimleri Dergisi*, no. 3, pp. 23–27, 2016.

[69] T. Taipalus and J. Ahtiainen, "Human detection and tracking with knee-high mobile 2d lidar," in *2011 IEEE International Conference on Robotics and Biomimetics*, pp. 1672–1677, IEEE, 2011.

[70] H. Edelsbrunner, "Computing the extreme distances between two convex polygons," *Journal of Algorithms*, vol. 6, no. 2, pp. 213–224, 1985.

[71] H. Yang, X. Liu, and I. Patras, "A simple and effective extrinsic calibration method of a camera and a single line scanning lidar," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 1439–1442, IEEE, 2012.

[72] J. Li, X. He, and J. Li, "2d lidar and camera fusion in 3d modeling of indoor environment," in *2015 National Aerospace and Electronics Conference (NAECON)*, pp. 379–383, IEEE, 2015.

[73] J. Gräter, T. Strauss, and M. Lauer, "Photometric laser scanner to camera calibration for low resolution sensors," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1552–1557, IEEE, 2016.

[74] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2301–2306, IEEE, 2004.

[75] L. Huang and M. Barth, "A novel multi-planar lidar and computer vision calibration procedure using 2d patterns for automated navigation," in *2009 IEEE Intelligent Vehicles Symposium*, pp. 117–122, IEEE, 2009.

[76] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Extrinsic calibration of a 3d laser scanner and an omnidirectional camera," *IFAC Proceedings Volumes*, vol. 43, no. 16, pp. 336–341, 2010.

[77] L. Zhou and Z. Deng, "Extrinsic calibration of a camera and a lidar based on decoupling the rotation from the translation," in *2012 IEEE Intelligent Vehicles Symposium*, pp. 642–648, IEEE, 2012.

[78] L. Zhou and Z. Deng, "A new algorithm for the extrinsic calibration of a 2d lidar and a camera," *Measurement Science and Technology*, vol. 25, no. 6, p. 065107, 2014.

[79] G. Li, Y. Liu, L. Dong, X. Cai, and D. Zhou, "An algorithm for extrinsic parameters calibration of a camera and a laser range finder using line features," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3854–3859, IEEE, 2007.

[80] K. Kwak, D. F. Huber, H. Badino, and T. Kanade, "Extrinsic calibration of a single line scanning lidar and a camera," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3283–3289, IEEE, 2011.

[81] S. Sim, J. Sock, and K. Kwak, "Indirect correspondence-based robust extrinsic calibration of lidar and camera," *Sensors*, vol. 16, no. 6, p. 933, 2016.

[82] N. J. Higham, *Matrix nearness problems and applications*. Citeseer, 1988.

[83] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*, pp. 105–116, Springer, 1978.

[84] R. Unnikrishnan and M. Hebert, "Fast extrinsic calibration of a laser rangefinder to a camera," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-09*, 2005.

[85] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[86] M. Quigley, E. Berger, A. Y. Ng, *et al.*, "Stair: Hardware and software architecture," in *AAAI 2007 Robotics Workshop, Vancouver, BC*, pp. 31–37, 2007.

[87] K. A. Wyrobek, E. H. Berger, H. M. Van der Loos, and J. K. Salisbury, "Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot," in *2008 IEEE International Conference on Robotics and Automation*, pp. 2165–2170, IEEE, 2008.

# APPENDIX A

# PUBLISHED AND ACCEPTED PAPERS

## A.1 A Lane Detection Algorithm Based on Reliable Lane Markings

This published paper [44] develops a robust and effective vision-based lane detection approach. In the proposed method, gray-scale images are converted to two binary images from a fixed region of interest. These images are then merged using a novel neighborhood AND operator and then transformed to a bird's eye view via inverse perspective mapping. A histogram image is extracted from the BEV and two gaussian probability density functions are fit to its left and right regions to determine the variance of the left and right lane markings. Furthermore, a reliable region for the detection of the left or right lane markings is chosen based on the distribution of the candidate lane pixels using the histogram plot and the variances of the regions on the BEV. High number of candidate lane pixels and small standard deviation of the pixels along the columns imply a reliable region. After determining reliable region, second-order polynomial $y = ax^2 + bx + c$ is fit to the actual lane pixels on that region. Lastly, we shift the detected lane to the less reliable region by as many pixel as the lane width by changing the parameter $c$ of the polynomial.

The proposed method can detect the lanes in complex cases such as lanes that are affected by shadows, occupied by other vehicles and when there exit signs on the road since the lane detection is carried out on the reliable region and then shifted to the appropriate location around the peak of the histogram plot of the less reliable region. In addition to that, it can detect curved lanes thanks to the second-order polynomials. Moreover, since the feature extraction evaluates the both bright and edge pixels in horizontal and vertical vicinity as candidate lane pixels, low threshold values to obtain gradient and segmentation-based binary images can be predefined.

Unlike the stated advantages of the proposed method in the published work, it can fail when there exist heading error or when the curvature of the road is high such that one lane (left or right) crosses the other region (right or left) since the regions are divided from the middle column in the images. In addition to that, even if low threshold values can be predefined, they are still fixed and this situation can be problematic at nights since the brightness level is low at nights. Lastly, the shifting of the detected lane on the reliable region to the less reliable region by changing the $c$ coefficient as many pixels as the lane width can be erroneous in the case of very curved lanes since changing $c$ parameter does not analytically represent the offset lane of the detected lane.

### A.2 Robust Lane Recognition Based on Arc Splines

This published paper [45] proposes a lane detection algorithm with a particular feature extraction technique and an arc-spline lane model. The proposed algorithm first determines a static region of interest. Then, feature extraction is applied to establish candidate lane pixels in a binary image. This binary image is then transformed to a bird's eye view. In the BEV, a reliable region for lane markings is selected based on the number and distribution of foreground pixels in the left and right regions. After that, a lane model for the reliable region is computed using the least square error from the candidate lane pixels as in [44]. Afterwards, the computed model is shifted to the less realible region by the lane width to detect the other lane marking. As a specific feature, a new Neighborhood AND operator is introduced for feature extraction and arc-splines are used as a lane model for the first time in the literature. Two main advantages of applying arc-splines are that they can be represented analytically and the offset curve of an arc-spline is again an arc-spline such that parallel lane markings can be easily represented by arc-splines with an offset in the arc radius.
In addition to the stated advantages of the work [44] in Appendix A.1, the proposed method in the published work improves and reduces the computational complexity of the feature extraction method presented in [44] by taking the pixels of the gradient and segmentation-based binary images in the vertical vicinity only. More importantly, this work contributes arc-spline curve for lane modeling for the first time in the litera-

ture. Thanks to the arc-spline model, the detected lanes are shifted to the less reliable region without violating the parallelism constraint.

Images on the bird's eye view are obtained to make the lanes are parallel. This holds in most cases, however when the road is not flat or when the vehicles shake, the lanes may not be parallel on the bird's eye view. In those situations, applying strict parallelism constraint to the lanes through shifting the arc-spline model can lead to coarse lane detection results on the less reliable region. Additionally, even if the feature extraction method is improved, it can be problematic at nights since fixed threshold values are utilized. Additionally, the stated disadvantage of the work [44] described in Appendix A.1, when one lane crosses the other region, also holds in this published work.

To address the disadvantages of the published works [44, 45], we switch to the method for feature extraction and lane detection described in a detailed way in Chapter 3. For the feature extraction method, 1D top-hat kernel enhances the lanes and the enhanced lane pixels supports each other for the local maxima in the histograms of the horizontally divided convolved subimages. Since the local maxima constituted from a number of lane pixels are distinctive from the noisy local maxima, we can eliminate the noisy ones easily through low threshold values. Also, since the local maxima which are greater than a low threshold value are sorted and the highest ones are counted as candidate lane pixels, adaptivity to the different lighting conditions is provided. In addition to that, as the best pair for the left and right lanes are selected irrespective of the location of the detected lines, the problems stated in Appendix A.1 and arising from the heading error and very curved lanes are addressed in this thesis.

### A.3 Sensor Fusion of a Camera and 2D LIDAR for Lane Detection

This published paper presents a novel lane detection algorithm based on fusion of camera and 2D LIDAR data and the Chapter 4 is based on this published paper. On the one hand, objects are detected via 2D LIDAR. On the other hand, a binary BEV image is acquired through feature extraction methods. Then, the location of the detected objects is estimated on the BEV image after extrinsic calibration between the camera and 2D LIDAR. The main contribution of this paper is to obtain a modified BEV

image, where the pixels occluded by detected objects are converted to background pixels on the binary BEV image. Then, lane detection is performed on the modified BEV image. Computational and experimental results show that using the modified BEV image significantly increases the lane detection accuracy.

## A.4 Comparison of 2D LIDAR Data Segmentation Methods Based on Synthetic Data Generation

This accepted paper proposes a novel method for the generation of synthetic 2D LIDAR data and presents a comparison of existing 2D LIDAR data segmentation methods in the literature: fixed breakpoint detection, adaptive breakpoint detection, nearest neighbor, $k$ nearest neighbor, radially bounded nearest neighbor and distance varying radially bounded nearest neighbor. The Chapter 4.2 is based on this work. To generate the synthetic data, objects with different shapes, orientations and sizes are placed in the range of a 2D LIDAR, respecting a predefined minimum distance between objects. After placing a predefined number of objects, the algorithm determines the LIDAR points hitting the objects that are not occluded by other objects. A comparative study of accuracy results based on the synthetic data is presented and the computational complexity of the described segmentation methods is evaluated. To achieve a comprehensive comparison, the synthetic data generation is parametrized by the object size, number of objects and minimum distance between objects.