

ADAPTIVE STRUCTURED AND UNSTRUCTURED ROAD
DETECTION USING LIDAR AND DOME-CAMERA

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

BY

SİNAN ÖZGÜN DEMİR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

SEPTEMBER 2017

Approval of the thesis:

**ADAPTIVE STRUCTURED AND UNSTRUCTURED ROAD
DETECTION USING LIDAR AND DOME-CAMERA**

submitted by **SİNAN ÖZGÜN DEMİR** in partial fulfillment of the requirements for
the degree of **Master of Science in Mechanical Engineering Department, Middle
East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Mehmet Ali Sahir Arıkan
Head of Department, **Mechanical Engineering** _____

Assoc. Prof. Dr. Erhan İlhan Konukseven
Supervisor, **Mechanical Eng. Dept., METU** _____

Assist. Prof. Dr. Ahmet Buğra Koku
Co-Supervisor, **Mechanical Eng. Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Yiğit Yazıcıoğlu
Mechanical Eng. Dept., METU _____

Assoc. Prof. Dr. Erhan İlhan Konukseven
Mechanical Eng. Dept., METU _____

Assist. Prof. Dr. Ahmet Buğra Koku
Mechanical Eng. Dept., METU _____

Assist. Prof. Dr. Ali Emre Turgut
Mechanical Eng. Dept., METU _____

Assist. Prof. Dr. Bülent İrfanoğlu
Mechatronics Eng. Dept., Atılım University _____

Date: 06.09.2017

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, Last name : Sinan Özgün Demir

Signature :

ABSTRACT

ADAPTIVE STRUCTURED AND UNSTRUCTURED ROAD DETECTION USING LIDAR AND DOME-CAMERA

Demir, Sinan Özgün

M.Sc., Department of Mechanical Engineering

Supervisor: Assoc. Prof. Dr. Erhan İlhan Konukseven

Co-Supervisor: Assist. Prof. Dr. Ahmet Buğra Koku

September 2017, 83 pages

Robotic ground vehicles are widely used in different road conditions to perform various tasks under semi- or fully-autonomous operation. To accomplish the given tasks, the vehicle should detect road regions accurately. Also, for a successful operation, the mobile robot requires a quick adaptation for changing road conditions. The objective of this study was developing an adaptive road detection algorithm for a semi-autonomous mobile platform (GOAT). For that purpose three different methods were developed for a robust classification of road regions ahead of the vehicle in both constructed and unconstructed environments. In the first method, LIDAR sensor was used to detect road regions by utilizing the data with adaptive parameter sets, which were estimated by utilizing discriminative learning approach. The experiments in structured environment showed that accuracy (ACC) of the output increased, while the false positive rate (FPR) decreased compared to the constant parameter

approach. However, for the tests conducted in unstructured environment desired results were not obtained. Therefore, the second road detection algorithm based on visual and range measurement data needed to be developed. By this algorithm, approximately 50% decrease in the FPR values for both structured and unstructured road conditions was observed by filtering the segmented point cloud based on the hue color channel values. In the third road detection method, an online supervised learning algorithm was developed, which used the outputs of the second road detection algorithm to create and/or update visual road models. In the conducted experiments, it was shown that road regions and general road boundary behaviors can be detected both in front and back directions of the vehicle independent from the road shape.

Keywords: Adaptive road detection, discriminative learning, online supervised learning, semi-autonomous mobile platform, structured and unstructured roads

ÖZ

LIDAR VE KÜRESEL KAMERA KULLANILARAK ADAPTİF YAPILI VE YAPISIZ YOL BULMA

Demir, Sinan Özgün

Yüksek Lisans, Makina Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Erhan İlhan Konukseven

Ortak Tez Yöneticisi: Yar. Doç. Dr. Ahmet Buğra Koku

Eylül 2017, 83 sayfa

Robotik kara araçları bir çok farklı amaç doğrultusunda, değişen yol şartlarında yarı ve tam otonom olarak sıklıkla kullanılmaktadırlar. Verilen görevleri başarıyla gerçekleştirmek için aracın yol bölgelerini yüksek doğruluk oranıyla bulması gerekmektedir. Ayrıca mobil robot değişen yol şartlarına hızlıca adapte olabilmelidir. Bu çalışmanın amacı, yarı-otonom mobil araç için değişen ortam koşullarına uyarlanabilen bir yol bulma algoritmasının geliştirilmesidir. Bu amaçla mobil aracın yapıları ve yapısız çevre şartlarında yol bölgelerini sürerlik içerisinde bulmasını hedefleyen üç farklı yöntem geliştirilmiştir. İlk yöntemde yol bölgeleri LIDAR sensöründen elde edilen verilerin ayırt edici öğrenme tekniğiyle adaptif olarak işlenmesiyle elde edilmiştir. Yapılı ortamlardaki testlerin sonuçlarına göre, algoritma parametrelerinin sabit tutulduğu yaklaşıma kıyasla, önerilen yöntemin sonuçlardaki doğruluk oranını arttırırken, yanlış pozitiflik oranında düşüş sağladığı

gözlemlenmiştir. Fakat yapısız ortamlarda yapılan testlerde beklenen sonuçlar elde edilememiştir. Bu sebeple, ortam görsellerini ve mesafe ölçüm verilerini kullanan ikinci bir yol bulma algoritması geliştirilmiştir. Geliştirilen bu ikinci algorithmada, seçilen nokta bulutunun renk tonuna göre filtrelenmesiyle yanlış pozitiflik oranında yaklaşık yarı yarıya bir düşüş gözlemlenmiştir. Önerilen üçüncü yöntemde ise, ortamdaki yol çevrimiçi denetimli öğrenmeye dayalı bir algoritma kullanılarak bulunmuştur. Bu yaklaşımda, ikinci yol bulma yönteminin çıktıları görsel yol modellerinin yaratılmasında ve/veya güncellenmesinde kullanılmıştır. Yapılan testler sonucunda, önerilen metot ile aracın ön ve arkasında yer alan yolun ve yol sınırlarının genel davranışlarının, yolun şeklinden bağımsız olarak bulunabildiği gösterilmiştir.

Anahtar Sözcükler: Adaptif yol bulma, ayırt ederek öğrenme, çevrimiçi denetimli öğrenme, yarı-otonom mobil araç, yapılı ve yapısız yollar

To My Father

ACKNOWLEDGMENTS

I would first like to express my sincere appreciation and gratitude to my supervisors Assoc. Prof. Dr. E. İlhan Konukseven and Asst. Prof. Dr. A. Buğra Koku for their continuous support, criticism and invaluable guidance throughout my thesis study.

For their comments and criticism, I would also like to thank to the examining committee members; Assoc. Prof. Dr. Yiğit Yazıcıoğlu, Asst. Prof. Dr. Ali Emre Turgut, and Asst. Prof. Dr. Bülent İrfanoğlu.

I am gratefully indebted to my colleagues and friends, Tayfun Efe Ertop, Onurcan Kaya, Musab Çağrı Uğurlu, Sedat Pala, Cenk Çetin and Gizem Şencan for their supports and collaborations throughout my thesis study. I appreciate the members of İmge Harita for sharing their technical knowledge. Moreover, I would also like to thank my fellow riders and sailors for their companion and our refreshing trips.

I would like to extend my heartfelt thankfulness and best wishes to Saadet Fatma Baltacı for her support, continuous effort to motivate me, sharing all my happiness and grief since we have met, and most importantly for her love and warm friendship.

I would like to state my grateful appreciation to my parents Hülya – Sadık Demir, and my brother Ulaş Eren Demir for their endless love and support. They have always been encouraging, patient and helpful to me throughout my years of study and in my life. Words cannot express my feelings and gratitude to you and your love.

Lastly, I would like to thank the Scientific and Technological Research Council of Turkey (TUBITAK) for their financial support with grant code 111M580.

TABLE OF CONTENTS

ABSTRACT.....	v
ÖZ	vii
DEDICATION	ix
ACKNOWLEDGMENTS.....	x
TABLE OF CONTENTS.....	xi
LIST OF TABLES	xiv
LIST OF FIGURES.....	xvi
LIST OF ABBREVIATIONS	xx
CHAPTERS	
1. INTRODUCTION	1
1.1. Mobile Robotics	1
1.2. Semi-Autonomous Operation.....	2
1.3. Road Detection Methods.....	3
1.4. Scope of the Thesis	7
1.5. Outline of the Thesis	9
2. MATERIAL AND METHOD	11

2.1. Mobile Robotic Platform Overview	11
2.2. Localization	15
2.2.1. Odometry Based Motion Model	15
2.2.1.1. Mathematical Model for Linear Motion	16
2.2.1.2. Mathematical Model for Curvilinear Motion	17
2.2.1.3. Covariance Matrix Determination	19
2.3. Registration of LIDAR Points to Images Captured By Dome-camera	21
2.4. Conversion of Rectified Pixel Locations to Raw Pixel Locations	23
2.5. Road Detection	27
2.5.1. Adaptive Road Detection Using 2D LIDAR Sensor	27
2.5.1.1. Breakpoint Detection	28
2.5.1.2. Line Segment Extraction	30
2.5.1.3. Line Segment Selection	32
2.5.1.4. Adaptive Parameter Selection	32
2.5.2. Adaptive Road Detection Using 2D LIDAR Sensor and Visual Data	34
2.5.2.1. Preprocessing and Color Space Conversion Of The Captured Image	34
2.5.2.2. Filtration Of The Selected Line Segment Based On Hue Color Channel Value	35

2.6. Adaptive Road Detection Using Online Supervised Learning Approach	37
2.6.1. Generation Of Training Data	37
2.6.2. Updating Previously Learned Road Models.....	38
2.6.3. Road Regions Detection	39
2.7. Experiments.....	39
3. RESULTS AND DISCUSSION	45
3.1. Localization.....	45
3.2. Conversion of Pixel Locations Between Rectified and Raw Images...	49
3.3. Road Detection.....	54
3.3.1. Adaptive Road Detection Using 2D LIDAR Sensor	54
3.3.2. Adaptive Road Detection Using 2D LIDAR Sensor And Visual Data.....	60
3.3.3. Adaptive Road Detection Using Online Supervised Learning Approach.....	65
4. CONCLUSION & FUTURE WORK.....	71
REFERENCES.....	75
APPENDICES	
A. ROS NODE USED FOR THE DEFINITION OF SENSORS' POSITIONS AND ATTACHED COORDINATE FRAMES	81

LIST OF TABLES

TABLES

Table 2-1 Standard deviation and variance values for MinIMU-9 in [rad]	20
Table 2-2 Standard deviation and variance of motion model in [mm]	20
Table 2-3 Pseudo code of breakpoint detection algorithm.....	29
Table 2-4 Pseudo code of line segment extraction algorithm	31
Table 3-1 Parameter sets used for the experiments conducted in SR environment ...	56
Table 3-2 ACC and FPR values for three different parameter sets computed on two different sections of SR	57
Table 3-3 Parameter sets used for the experiments conducted in UR environment ..	58
Table 3-4 ACC and FPR values for three different parameter sets computed for unstructured road.....	59
Table 3-5 Mean process times for one cycle of road detection algorithm (for $N \approx 7500$ LIDAR measurement sets)	60
Table 3-6 ACC and FPR values calculated for the proposed adaptive road detection algorithms on SR Section 2	61
Table 3-7 ACC and FPR values calculated for the proposed adaptive road detection algorithms on UR Section 1	62
Table 3-8 ACC and FPR values calculated for the proposed adaptive road detection algorithm throughout the entire structured environment.....	64

Table 3-9 ACC and FPR values calculated for the proposed adaptive road detection algorithm throughout the entire unstructured environment 64

Table 3-10 Parameter set used for the experiments in SR and UR environments 65

LIST OF FIGURES

FIGURES

Figure 1-1 Automated material transporter [2]	2
Figure 1-2 a) Tesla's autopilot system [3], b) Ford's self-parking system [4]	2
Figure 1-3 Actual road boundaries and statistical road model with road-edge confidence intervals [8].....	4
Figure 1-4 Steps of road detection algorithm proposed in [18], a) raw image of the environment, b) training image for the long range road detection algorithm, where green pixels shows the close range road area found by using surface roughness, c) output of the long range road detection algorithm, where green pixels shows the road region.....	6
Figure 2-1 Picture of UGV platform, GOAT	11
Figure 2-2 Schematic view of GOAT	12
Figure 2-3 Schematic view of GOAT with coordinate frames attached to the sensors, where x-, y- and z-axes shown in red, blue, and green, respectively	13
Figure 2-4 Example LIDAR readings taken by the sensors mounted to the front surface, red points measured by SICK and blue points measured by Hokuyo	14
Figure 2-5 Successive positions of the mobile robot while rotating on a plane - 1 ...	17
Figure 2-6 Successive positions of the mobile robot while rotating on a plane - 2 ...	18
Figure 2-7 Schematic view of LIDAR point conversion steps	21

Figure 2-8 Visual representation of bilinear interpolation.....	23
Figure 2-9 Captured image by the camera, a) raw image, b) rectified image.....	24
Figure 2-10 Camera images used in Camera Calibrator App of Matlab.....	25
Figure 2-11 Sample LIDAR measurements for structured environment (left) and unstructured environment (right)	28
Figure 2-12 Schematic view of breakpoint detection and D_{\max} determination.....	29
Figure 2-13 IEPF step of line segment extraction.....	31
Figure 2-14 Terrain labeling for discriminative learning step, LIDAR points labeled as road region are shown in blue.....	33
Figure 2-15 Raw images captured by the camera (a,c) and preprocessed images (b,d)	35
Figure 2-16 Hue channel values of the points in line segment before and after contrast adjustment.....	36
Figure 2-17 Hue channel values of the points forming the line segment and the average hue values of each point group after increasing contrast.....	37
Figure 2-18 Test path driven by the vehicle, red path shows the structured road and blue path shows the unstructured road	40
Figure 2-19 Sample images for structured road (a) and unstructured road (b).....	40
Figure 2-20 Raw image of checkerboard captured by the Ladybug	42
Figure 2-21 Effect of the radial distortion to the image [42]	43
Figure 2-22 Sample ground truth data generation by labeling road region manually	44
Figure 3-1 Travelled path by the mobile platform to test localization methods in a) structured environment and b) unstructured environment.	46

Figure 3-2 A sample section view of the computed path.....	47
Figure 3-3 Close-view of the estimated path. Data points generated by using GPS data are shown as circle, and data points generated by using IMU and OBMM data are shown as cross.	48
Figure 3-4 Traveled route by GOAT computed by EKF localization algorithm.....	48
Figure 3-5 Undistorted image by using the rectifying function of Ladybug API.....	50
Figure 3-6 Undistorted image obtained by using equations (2.28), (2.29) and (2.30) with the parameters found by Matlab Camera Calibrator App	51
Figure 3-7 Undistorted image obtained by using equations (2.28), (2.29) and (2.30) including the tangential distortion parameters found by Matlab Camera Calibrator App	52
Figure 3-8 Undistorted image obtained by using equations (2.28), (2.29) and (2.30) with the parameters estimated by PSO.....	53
Figure 3-9 Measured points on SR by LIDAR shown on the ground truth image a) raw point cloud, b) point cloud after breakpoint detection and line segment extraction by utilizing Fully-adapted PS for SR, c) point cloud forming the line segments labeled as road are by utilizing Fully-adapted PS for SR	55
Figure 3-10 Image of SR Section 2, a) raw image of the scene, b) LIDAR points labeled as road where TP and FP results shown in cyan and red, respectively	57
Figure 3-11 Points labeled as line segment shown on the ground truth image of UR section, a) Predefined PS, b) Fully-adapted PS for SR, and c) Fully-adapted PS for UR	59
Figure 3-12 Measured points on SR Section 2, a) image of the scene, b) raw point cloud on image, c) point cloud labeled as road region after running road detection algorithm based on LIDAR and visual data, where TP and FP results shown in cyan and red, respectively.....	62

Figure 3-13 Measured points on UR Section 1, a) raw point cloud on image, b) point cloud labeled as road region after running road detection algorithm based on LIDAR only, c) point cloud labeled as road region after running road detection algorithm based on LIDAR and visual data, where TP and FP results shown in cyan and red, respectively 63

Figure 3-14 Raw images of the scene on the top row, and detected road regions on the bottom row for SR Section 3. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras. 66

Figure 3-15 Raw images of the scene on the top row, and detected road regions on the bottom row for SR Section 2. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras. 66

Figure 3-16 Raw images of the scene on the top row, and detected road regions on the bottom row for SR Section 4. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras. 67

Figure 3-17 Raw images of the scene on the top row, and detected road regions on the bottom row for SR Section 5. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras. 68

Figure 3-18 Raw images of the scene on the top row, and detected road regions on the bottom row for UR Section 2. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras. 69

LIST OF ABBREVIATIONS

ACC	Accuracy
API	Application Programming Interface
ATV	All-Terrain Vehicle
BP	Breakpoint
EKF	Extended Kalman Filter
FP	False Positive
FPR	False Positive Rate
FN	False Negative
GPS	Global Positioning System
HSV	Hue Saturation Value Color Space
IEPF	Iterative End Point Fit Algorithm
IMU	Inertial Measurement Unit
IRC	Instantaneous Rotation Center
LIDAR	Light Detection and Ranging
METU	Middle East Technical University
OBMM	Odometry Based Motion Model
PS	Parameter Set
PSO	Particle Swarm Optimization
RGB	Red Green Blue Color Space
ROS	Robotic Operation System
SR	Structured Road
TN	True Negative
TP	True Positive
TPR	True Positive Rate
UGV	Unmanned Ground Vehicle
UR	Unstructured Road

CHAPTER 1

INTRODUCTION

In this section of the thesis, firstly mobile robotic systems and semi-autonomous operation are explained. As next, road detection algorithms in the literature are covered. Lastly, scope of the thesis and the outline of this report are given.

1.1. Mobile Robotics

Robotic technology has been an important research topic since 1960s. In the beginning, main usage area of the robots was limited to the industrial applications, to replace manpower in hazardous operations in production, such as welding and handling hot parts coming from die casting [1]. The common point in these early applications of the robotic systems was that their working environments were structured, which could be mathematically modeled, possible events could be foreseen, and at worst the environment could be adapted according to the working conditions and requirements of the robots. Improvements in the areas of image processing, pattern recognition, machine learning, artificial intelligence and control theory made it possible to adapt robotic systems themselves to the dynamic environment. As a result of this, usage area of the robotic technology has expanded to the mobile systems. Similar to robotic arm manipulators, mobile robots are commonly used in repetitive works, such as, transportation and delivery of materials in structured environments as shown in Figure 1-1, to minimize the human effort and prevent the possible failures caused by human operators. In addition to their industrial usage, mobile robotic systems are also utilized in accomplishing given

tasks or collecting information about the environments, such as search and rescue operations, surveying in hazardous regions and patrolling. Moreover, mobile robotic technology has recently started being used in daily life for human comfort. Self-parking and self-driving vehicles are some of the examples for their application areas.



Figure 1-1 Automated material transporter [2]

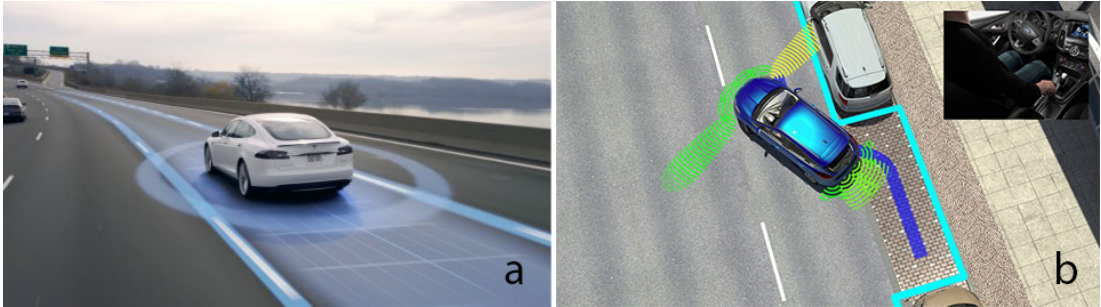


Figure 1-2 a) Tesla's autopilot system [3], b) Ford's self-parking system [4]

1.2. Semi-Autonomous Operation

Mobile robotic systems may be divided into three subgroups based on their control strategies. These are fully autonomous, semi-autonomous and manual systems. Fully autonomous systems are expected to operate in their working environments without any need for human intervention. In manual systems, on the other hand, all the tasks are done by human operation only, including moving the robot in environment. Different than fully autonomous and manual systems, semi-autonomous applications

aim accomplishing given tasks by a collaborative work done by the robot and the human operator, [5]. In these systems, mobile platform may be controlled either by the robot itself or the user based on the definition of given tasks. The decision on the task sharing can be done manually by the user or autonomously by the robot.

1.3. Road Detection Methods

Extraction of traversable areas and road characteristics is extremely vital for robust performance of fully autonomous and semi-autonomous UGVs in an unknown environment. Therefore, different approaches for the road detection and following are proposed in literature.

In [6], proposed solutions in the literature are grouped under model based, feature based and machine learning based methods. In model based methods, road boundaries are tried to be expressed with a mathematical equation. Although simple models may be robust and computationally cheap, their working environments are very limited and they may not provide accurate results. Even though complex models may be applicable for a wider range of road types, they can be more easily affected by the disturbances. Also, their computational load increases as the complexity of the model increases [7]. In [8], Aufrère et al. proposed a model based algorithm to detect and follow the road boundaries in the image. In this method, captured image was divided into several strips and a statistical model was formed to define the edge coordinates and the corresponding confidence levels on each of these strips, which can be seen in Figure 1-3. Based on the computed edge coordinates in continuously captured images, road model was updated recursively and the confidence level increased. Thus the statistical model converges to the actual lane edges.

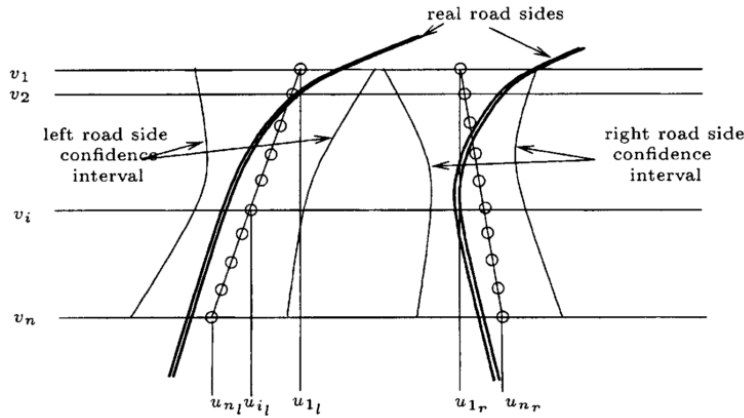


Figure 1-3 Actual road boundaries and statistical road model with road-edge confidence intervals [8]

Feature based methods, on the other hand, are insensitive to the road shapes and requires less a priori knowledge about the environment. Though, the success rate of these type of approaches strongly depends on choosing a proper feature descriptor [6]. Based on the utilized feature type, methods presented in the literature are divided into three main groups, which are methods based on color features, geometrical features and hybrid features obtained by integration of visual and geometrical features. In order to obtain actionable road feature information, various sensors and sensor fusion strategies are developed for both structured and unstructured roads. Some of the commonly used sensors for this purpose are cameras, radars and LIDARs.

Cameras are among the most preferred sensors in robotic applications due to their low cost, low power consumption and high information content. Therefore, a wide variety of solutions are available in the literature. In [9], a road detection algorithm using images captured by the monocular camera was presented to model the road boundaries. The developed algorithm was based on the assumption that the road boundaries were two parallel lines, which can be modeled by second-degree polynomials. The road area was labeled using gray levels and texture information obtained by applying Robert operator to the captured images, which was also utilized to update the parameters of the road model. In [10], on the other hand, the road area

was assumed to have a trapezoidal shape, and the rectangular area in front of the vehicle was always defined as road. Based on the average HSV color channel values obtained from this rectangular area, captured images are filtered to extract road area. In their studies [11] and [12], Rasmussen *et al.* applied Gabor filters to obtain texture features from the images. The extracted data was used to determine the position of vanishing point. Based on the found vanishing points in successive images, road boundaries were defined. In another study, Rasmussen *et al.* [13] used the extracted features obtained by color camera and LIDAR sensor to train a two-layer neural network, which was later used as road classifier. Utilized features were color, texture and geometrical features. Color features consisted of joint histograms of image patches, whereas the texture features were obtained by applying Gabor filters to the same image patches. Variance in the height was selected as geometrical feature due to its robustness to noise. Alternative to previous approaches, in [14], a near-to-far self-supervised traversable road and obstacle detection algorithm using stereovision data for natural terrains was proposed. Because the reliability in stereovision depth measurements are limited up to around 12 meters, generated 3D point clouds were used to detect ground, obstacle and footline classes close to the vehicle. Then these features were used to train the online classifier, which was responsible for long-range road detection based on visual data from images [15], [16] and [17]. A similar approach was also utilized in [18], which could successfully estimate long range road region for changing lighting conditions after training the road detection algorithm using the close range stereovision data. An example result of this method is shown in Figure 1-4 together with the raw input image and created training image. Some of the other successful examples for visual data based road detection solutions are RALPH [19], GOLD [20], autonomous driving system of Universität der Bundeswehr München [21], VIRTUOUS [22], and the study by Jansen *et al.* [23]. In [24], obtained successful results in the literature for road detection algorithms using visual data are linked to their high information content and absence of sweep time while capturing data. However, it is also stated that the performance of a camera based system is strongly dependent on lighting condition of the environment and may

decrease drastically in case of shadow or direct light coming to the camera, as well as high noise in the acquired images due to poor lighting conditions.



Figure 1-4 Steps of road detection algorithm proposed in [18], a) raw image of the environment, b) training image for the long range road detection algorithm, where green pixels shows the close range road area found by using surface roughness, c) output of the long range road detection algorithm, where green pixels shows the road region

Alternative to camera based systems, radars can capture a high-quality picture of the environment even under extraordinary weather conditions, such as snow and rain, with a range up to 200m [25]. However, their high-cost, slow scanning speed, possibility of interference with other closely positioned radars and dependence of their performance on the material and geometry of the driving environment makes them less favorable for autonomous UGV applications [26], [27].

Although the working environments for LIDAR sensors are limited compared to radar sensors, as stated in [28], continuous increase in resolution, ability to measure geometric properties of the environment directly, and their feasibility make LIDARs a well suited option for detection of road characteristics. In [29], Morales *et al.* used LIDAR measurements to detect traversable areas, which were assumed to be always flat and bounded by trees, grass and bushes. Due to this assumption, the working area of the algorithm was limited to a specific structured environment only. Similarly, [30], [31] and [32] used LIDAR measurements for road boundary detection purposes limited to structured environments. Alternatively, Siegemund *et al.* used curb model to detect and follow the road boundaries in [33] and updated the model parameters

based on the data obtained by classifying 3D point cloud using conditional random field. In [34], Stückler *et al.* presented a localization algorithm using LIDAR data for a known environment with accurate road map. In this method, LIDAR points were first converted to the height images and then by applying oriented edge filters to these images, position of the curbs was calculated. As next, position of the vehicle was estimated by matching the calculated curb positions with the actual values obtained from the road network model. In [35], on the other hand, position of the roadside curbs was calculated based on a probability threshold mechanism. Extracted results were used for decision-making by the autonomous mobile robot to navigate in urban areas with unreliable GPS measurements. Similarly, Jian *et al.* proposed a curb detection algorithm in [36] using raw 3D point cloud generated by a velodyne LIDAR sensor to detect road area. In this approach, straight and curved road curbs were found by utilizing an adaptive threshold, which made using the algorithm for changing road conditions possible. Although the system was proved to be robust for the environments with continuous curb structure, decrease in the success rate was observed in case of discontinuous environments. In [37] and [38], change in the distribution of collected LIDAR points were used to detect surface changes and road was detected based on the surface roughness and size of the point cloud segments. This algorithm was proved to be applicable to both structured and unstructured environments. In [39], LIDAR and monocular camera data were fused for long distance adaptive road detection. In this proposed algorithm laser readings were utilized to find drivable areas and obstacles in close range to the vehicle and to train the developed self-supervised long-range road detection algorithm as described in [40].

1.4. Scope of the Thesis

Primary objective of this study is developing an adaptive road detection algorithm using a single 2D LIDAR sensor and a dome-camera. The proposed algorithms are expected to operate robustly both in structured and unstructured environments assuming that the road boundaries are covered either by curb, berm or vegetation.

For road detection, three different approaches were proposed in this thesis. The first method detects road area based on only geometrical features generated by 2D LIDAR data and consists of three main steps, namely breakpoint detection to differentiate changes in surfaces, line segment extraction to label possible road regions, and line segment selection to decide on drivable area. Alternative to similar approaches in the literature, an adaptive approach is utilized to estimate algorithm parameters by using discriminative learning approach. This allows the robotic system respond to changes in road properties easily. The second method utilizes both 2D LIDAR data and the images captured by the front facing lens of the dome-camera. Following the line segment selection step, LIDAR points forming the line segment are registered to the captured and preprocessed image. Then, corresponding pixel's HSV color space values are read and integrated to the point cloud data. As next, selected line segment is filtered based on discontinuity in the hue color channel value and the area corresponding to the road is selected. The third and last road detection method is based on online supervised learning approach, which uses the outputs of the secondly proposed road detection algorithm as training data in order to generate visual road models. As next, previously learned and/or null road models are updated using these training data. Lastly, road regions in the front and rear images captured by the dome-camera are found by using learned and updated visual road models.

In the secondary and last road detection algorithms, geometrical features are fused with the color features by registering the range measurement points to the preprocessed images captured by the camera. Moreover, evaluation of the obtained results from road detection algorithms are made by comparing the corresponding pixel positions of the points, which are classified as road region, with the manually labeled road images. Therefore a simultaneous localization algorithm is designed to estimate the relative position of the vehicle, when LIDAR measurements are taken, with respect to its location, when the images are captured by the camera. Developed localization algorithm employs GPS, IMU and OBMM readings and these sensor data are fused by using EKF algorithm.

1.5. Outline of the Thesis

The rest of the thesis is constructed as follow.

In Chapter 2, following the explanation of hardware and software structure of the mobile robotic system used in this study, proposed localization algorithm is explained in detail. Then, registration procedure of LIDAR points to the images captured by the camera and conversion of rectified pixel locations to raw pixel locations are presented. Afterwards, theories and working steps of the proposed road detection algorithms are described. Lastly, the procedures of the conducted experiments are explained.

In Chapter 3, visual and numerical results of the experiments are presented at first. As next, performances of the designed localization, point registration and road detection algorithms are evaluated by comparing their results with the methods presented in the literature.

In Chapter 4, a conclusion on this study is given and suggestions for the future work are presented.

CHAPTER 2

MATERIAL AND METHOD

In this chapter, first hardware and software structure of the mobile robotic system used in this study is described. As next, registration process of LIDAR points to the captured images and conversion of pixel positions from undistorted image to the raw image are explained. Afterwards, proposed localization and adaptive road detection algorithms are presented in detail. Lastly, by covering the experiments conducted to evaluate the performance of the proposed algorithms, this section is concluded.

2.1. Mobile Robotic Platform Overview

The UGV used in this study is a renovated electrical ATV for semi-autonomous usage in the scope of 111M580 numbered TÜBİTAK project. This mobile platform was named as ‘GOAT’, and its picture is presented in Figure 2-1.



Figure 2-1 Picture of UGV platform, GOAT

GOAT had a distributed control architecture consisting of one main computer and two auxiliary onboard computers. The main computer had an Intel i7 type CPU with 3.2 GHz processing speed and 8 GB ram. It was responsible for running programs with high computational loads, such as, localization, road detection, image processing and machine learning. On the other hand, the auxiliary computers with 1.0 GHz CPU and 512 MB ram were responsible for low level processing, such as, sensor measurements and interfacing with the actuators attached to GOAT. Communication between these separate computers was established by utilizing ROS environment. Moreover, all the developed algorithms were running under ROS.

The mobile robotic platform was capable of performing autonomous tasks on its own as well as manual operation through long-range user intervention. This allowed a hybrid task control scheme to be realized in which the user might choose the amount of autonomy in the operation. In other words, during operation, user can step in at any point and take the control of the robotic platform partially or fully. The opposite case was also valid, in which the robot may stop working autonomously and wait for manual operation if the uncertainty about the current state of the system prevents completing the given tasks.

The GOAT had three separate LIDAR sensors (one URG-04LX Hokuyo, JAPAN and two LMS291-S05 SICK, GERMANY), a dome-camera (Ladybug-5 PointGrey, CANADA), two IMUs (one IMU-440 Xsens, USA and one MinIMU-9 v3 Pololu, USA), a GPS module and six encoders (four of them on the rear wheels and two of them on the front wheels) for localization and mapping purposes. Positions of all sensors are schematically shown on the mobile robotic platform in Figure 2-2.

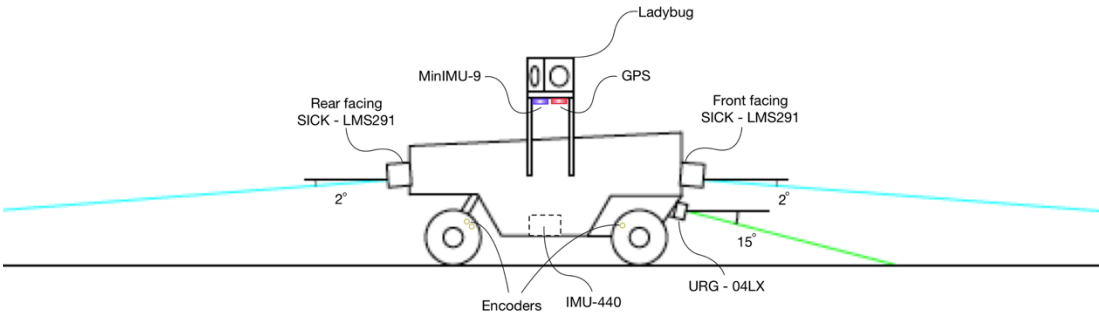


Figure 2-2 Schematic view of GOAT

All three LIDAR sensors were placed on the robotic platform making a specific angle with the horizontal plane as shown in Figure 2-2. Two of LIDARs (LMS291-S05 SICK, GERMANY) on the front and back surfaces of the robotic platform were utilized for long-range road boundary detection. These sensors were positioned at a height of 559.72 mm from the road surface and tilted by 2° around the z-axis with respect to the sensor's coordinate frame shown in Figure 2-3. Thus, the distance from LIDARs to the measurement line was set to be 16 m while the mobile platform moves on a planar road surface. On the other hand, the LIDAR, which was mounted on the front surface with 15° pitch angle relative to its coordinate frame shown in Figure 2-3, was used for short-range road detection as well as obstacle, bump and hole avoidance. While determining the pitch angle of this sensor, first turning radius of the GOAT was found by testing the vehicle in both structured and unstructured roads. Although the results were different for turning left and turning right, the highest turning radius value was found as 1 m. Considering the turning radius and the width of the GOAT, the distance from sensor to incident line of laser points for close range measurements were decided to be 1.5 meters in order to allow the vehicle escape from any kind of obstacles on its route. As a result, the height of the third LIDAR was determined to be 417.33 mm from the road surface with a pitch angle of 15° . Example LIDAR readings taken by the sensors mounted to the front surface is shown in Figure 2-4 as integrated to the camera image.

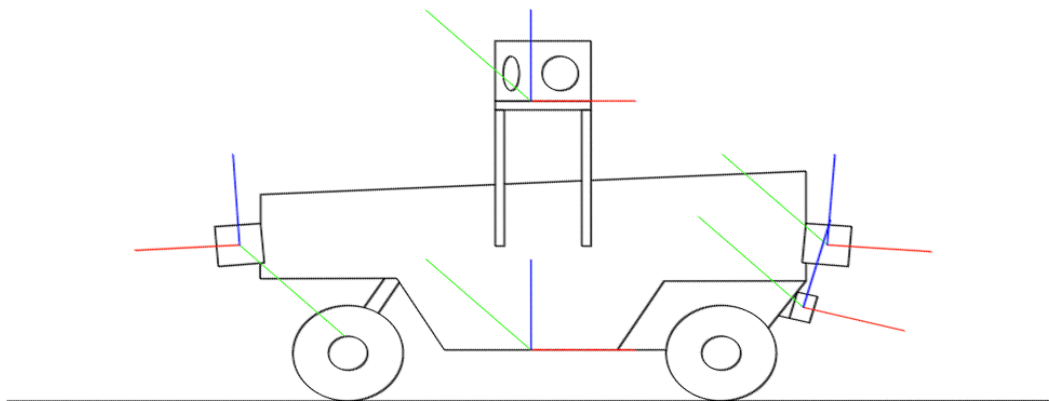


Figure 2-3 Schematic view of GOAT with coordinate frames attached to the sensors, where x-, y- and z-axes shown in red, blue, and green, respectively



Figure 2-4 Example LIDAR readings taken by the sensors mounted to the front surface, red points measured by SICK and blue points measured by Hokuyo

The dome-camera was centered on top of the robotic platform in order to maximize its field of view and cover the measurement range of LIDAR sensors.

The first IMU sensor, IMU-440, was mounted at the center of GOAT's bottom surface to minimize the magnification effects of base excitations. The other IMU sensor was planned to be a secondary measurement unit, and it might be used in the case of a disturbance in IMU-440 readings, which could be caused by noise in the system generated by the batteries and/or actuators. Therefore, the secondary IMU is placed on one side of the dome-camera at the top.

The GPS module was mounted on the opposite side of the secondary IMU at the top to have a clear and unobstructed view of sky.

Two encoders were mounted on each rear wheel to measure angular speed and direction of rotation. Additional encoders were placed on the non-actuated front

wheels to measure angular speeds, in order to minimize inaccuracy in the measurements due to the slippage in the rear wheels.

ROS node used for the definition of sensor positions and attached coordinate frames relative to the mobile robotic platform is given in APPENDIX A.

2.2. Localization

In order to localize the vehicle precisely, outputs of GPS, IMU and odometry based motion model were combined and used together. The first localization method was a GPS sensor for absolute positioning, which was operated at 1 Hz. The second method was based on double integration of linear acceleration and single integration of angular speed values read by IMU-440 at 10 Hz for incremental localization. The last positioning method was odometry based motion model explained in Section 2.2.1. In this model, linear and angular distances travelled by the vehicle were calculated at 10 Hz by using encoder and digital compass data without any integration. To combine the outputs of these three methods, Extended Kalman Filter algorithm was used, which was provided in the ROS environment.

2.2.1. Odometry Based Motion Model

The aim of the odometry based motion model is predicting the linear and angular distances traveled by the vehicle based on encoder and digital compass readings. This model was developed assuming that GOAT has an ideal Ackerman steering system, and it consists of two sub-models, which are linear and curvilinear motions in 3D space. The equations were derived with respect to the world fixed coordinate frame, whose x-axis points magnetic east, y-axis points magnetic north and z-axis points upwards. At the end, the outputs of the algorithm were transformed to the mobile robot fixed coordinate frame, whose x-axis points forward, y-axis points left and z-axis points upwards taking the GOAT's center as origin.

Before calculating the traveled distance by the vehicle, it is necessary to decide on whether the vehicle moves linearly or not. For that purpose, a simple threshold mechanism was employed in the proposed algorithm. In this approach, if the change in the heading angle between two successive readings was smaller than the user defined threshold value, the vehicle was assumed to make a linear motion. This threshold value was determined based on the distribution of collected compass readings when GOAT was stationary.

2.2.1.1. Mathematical Model for Linear Motion

In this part, equations for linear motion on a plane were derived in world fixed coordinate frame at first and then pitch angle reading was integrated to the equations to model the linear motion in 3D space.

Linear distances travelled by a ground vehicle on x and y-axes are given in (2.1) and (2.2) in terms of distance travelled by the rear axle's center, Δ_{cr} , and heading angle of the vehicle at previous time instance, $(\psi_c)_{t-1}$. Due to linear motion, the variable Δ_{cr} is equal to the distance travelled by the front axle's center, Δ_{cf} , and it depends on Δ_{fl} and Δ_{fr} , which are read from the front wheel encoders by one of the auxiliary computers. The same computer is also responsible for reading heading angle $(\psi_c)_t$ from secondary IMU sensor MinIMU-9.

$$\Delta x_p = \Delta_{cr} \sin(\psi_c)_{t-1} \quad (2.1)$$

$$\Delta y_p = \Delta_{cr} \cos(\psi_c)_{t-1} \quad (2.2)$$

$$\Delta_{cr} = \Delta_{cf} = \frac{\Delta_{fl} + \Delta_{fr}}{2} \quad (2.3)$$

Linear displacement in 3D space is defined in (2.4), (2.5) and (2.6) by adding the pitch angle $(\alpha_c)_{t-1}$ reading into (2.1) and (2.2).

$$\Delta x = \Delta x_p \cos(\alpha_c)_{t-1} \quad (2.4)$$

$$\Delta y = \Delta y_p \cos(\alpha_c)_{t-1} \quad (2.5)$$

$$\Delta z = \sqrt{(\Delta x)^2 + (\Delta y)^2} \sin(\alpha_c)_{t-1} \quad (2.6)$$

2.2.1.2. Mathematical Model for Curvilinear Motion

In this part, derivation of governing equations for curvilinear motion with respect to the world fixed coordinate frame are presented. Figure 2-5 and Figure 2-6 visualizes all the variables used in the equations schematically for two successive positions of the vehicle while making planar curvilinear motion. These two positions are also referred as initial and final positions of the vehicle's motion step throughout the derivation stages.

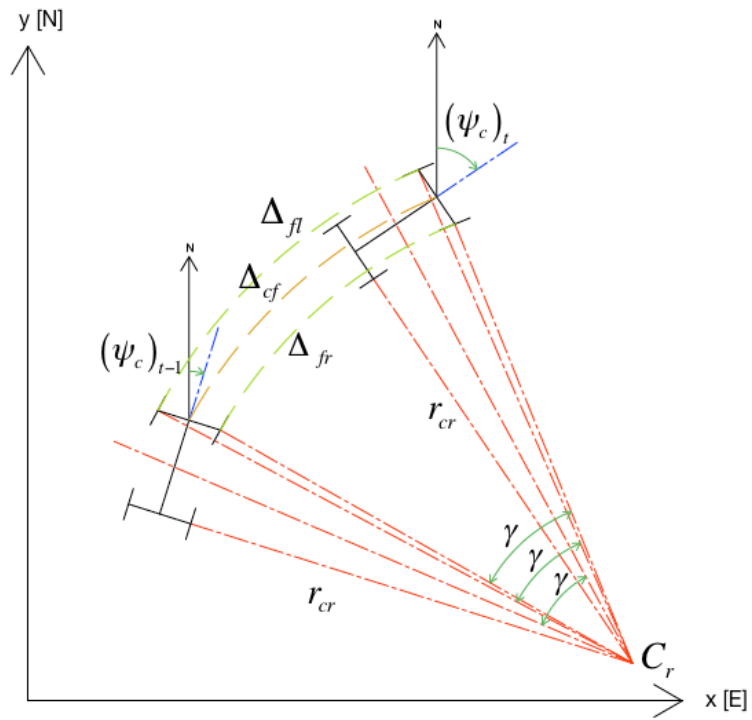


Figure 2-5 Successive positions of the mobile robot while rotating on a plane - 1

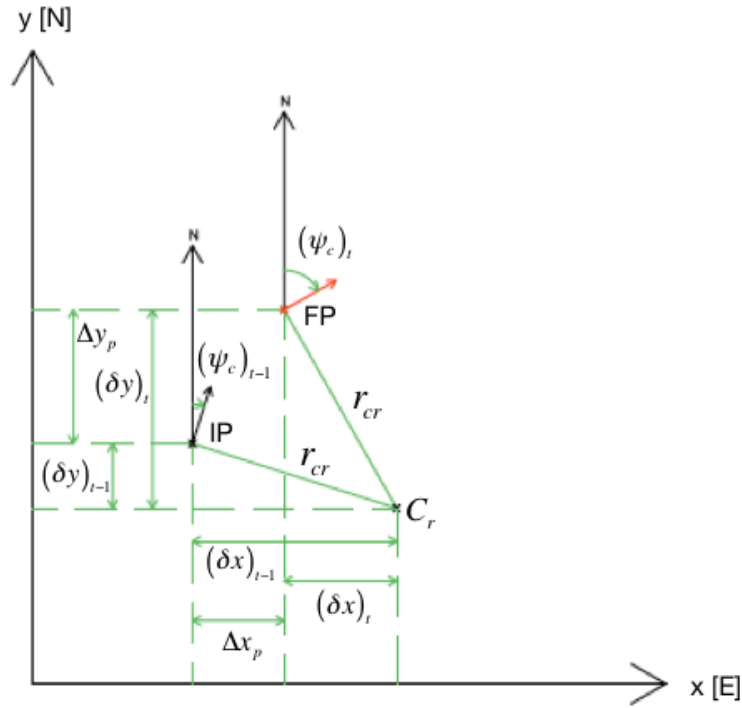


Figure 2-6 Successive positions of the mobile robot while rotating on a plane - 2

$$\Delta x_p = (\delta x)_t - (\delta x)_{t-1} \quad (2.7)$$

$$\Delta y_p = (\delta y)_t - (\delta y)_{t-1} \quad (2.8)$$

$$\Delta \psi_c = (\psi_c)_t - (\psi_c)_{t-1} \quad (2.9)$$

Variables δx and δy in (2.7) and (2.8) represents the horizontal and vertical distances between GOAT and IRC. These variables are calculated using (2.10) and (2.11).

$$(\delta x)_t = r_{cr} \cos(\beta_c)_t \quad (2.10)$$

$$(\delta y)_t = r_{cr} \sin(\beta_c)_t \quad (2.11)$$

$$(\beta_c)_t = \pi - (\psi_c)_t \quad (2.12)$$

Same as the linear motion case, the heading angle $(\psi_c)_t$ is read from the secondary IMU. The turning radius of rear axle's center, r_{cr} , utilized in (2.10) and (2.11) is calculated by (2.13).

$$r_{cr} = \sqrt{r_{cf}^2 - l^2} \quad (2.13)$$

$$r_{cf} = \frac{\Delta_{cf}}{\gamma} \quad (2.14)$$

$$\Delta_{cf} = \frac{\Delta_{fl} + \Delta_{fr}}{2} \quad (2.15)$$

$$\gamma = |(\psi_c)_t - (\psi_c)_{t-1}| \quad (2.16)$$

3D curvilinear motion is obtained by integration of pitch angle (α_c) into (2.10) and (2.11).

$$\Delta x = \Delta x_p \cos(\alpha_c)_t \quad (2.17)$$

$$\Delta y = \Delta y_p \cos(\alpha_c)_t \quad (2.18)$$

$$\Delta z = \sqrt{(\Delta x)^2 + (\Delta y)^2} \sin(\alpha_c)_t \quad (2.19)$$

2.2.1.3. Covariance Matrix Determination

Following the derivation of mathematical models for linear and curvilinear motions in 3D space, covariance matrices were determined. For this purpose, uncertainties and reading errors in the encoders and digital compass sensors were taken into consideration.

At first, the encoders attached on the front wheels were examined. Although it was necessary to include both lateral and longitudinal slippages for more accurate model, they were assumed to be negligible considering that the vehicle was moving with a low speed. To detect whether there was a measurement error or noise in the encoders, front wheels were rotated 50 full cycles with varying speeds. After testing each wheel for 15 times, total number of impulses read from the encoders was compared with the calculated ones, and it was seen that both of the encoders work properly and does not miss any pulses.

As next, the uncertainty of MinIMU-9 was evaluated by positioning GOAT in the environment with changing poses and recording sensor readings for 1 minute at each of these poses. Normal distribution was used to model the sensor noise. Standard deviation and variance values for roll, pitch and yaw angles were calculated in MS Excel and the result are presented in Table 2-1.

Table 2-1 Standard deviation and variance values for MinIMU-9 in [rad]

	Roll	Pitch	Yaw
Number of Collected Data	3226	3226	3226
Standard Deviation of Data	0.008074	0.010924	0.012700
Variance of Data	0.000065	0.000119	0.000161

After modeling the uncertainties of the sensors utilized in motion model, the uncertainty of the motion model itself was found by using sampling method. In this method, the vehicle’s displacement was simulated 1.000.000 time by using randomly generated sensor data, whose mean values were taken as the real sensor readings and standard deviation values were equalized to the previously determined values. At last, a Gaussian distribution was fitted to the generated data and the following results in Table 2-2 were obtained for linear and curvilinear motions.

Table 2-2 Standard deviation and variance of motion model in [mm]

	Linear Motion			Curvilinear Motion		
	X	Y	Z	X	Y	Z
Standard Deviation	12.963	10.382	14.237	24.706	32.325	11.835
Variance	168.032	107.790	202.684	610.391	1044.890	140.060

2.3. Registration of LIDAR Points to Images Captured By Dome-camera

As mentioned in the former chapters, LIDAR measurements were registered to the image captured by the camera to be used in road detection algorithm and the evaluation of its outputs. The procedure of converting LIDAR points measured in polar coordinates to the raw pixel locations on the i^{th} camera is schematically shown in Figure 2-7.

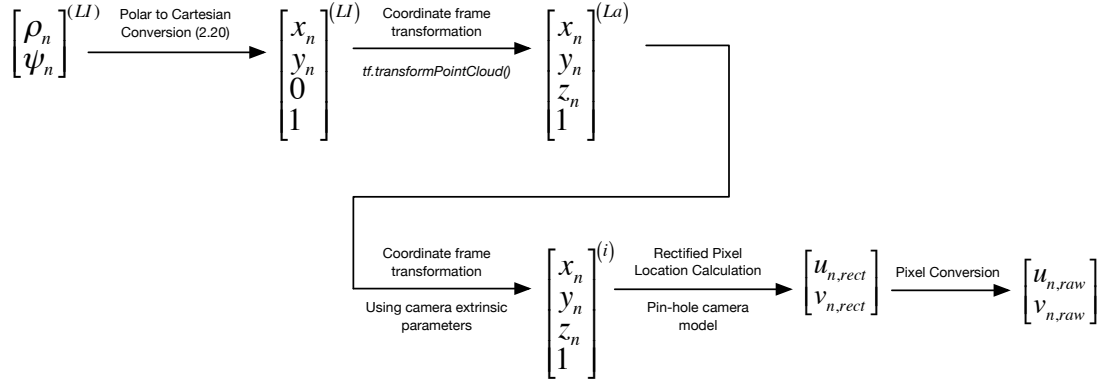


Figure 2-7 Schematic view of LIDAR point conversion steps

To start with, the measured range points were converted from polar coordinates to Cartesian coordinates using (2.20), where 'LI' stands for the coordinate frame attached to the LIDAR, and ρ_n and ψ_n are the range and the corresponding yaw angle values for the n^{th} LIDAR measurement. Since the utilized LIDAR was taking only planar measurements, position of the points in z-axis with respect to the LIDAR attached coordinate frame was always 0.

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix}^{(LI)} = \rho_n \begin{bmatrix} \cos(\psi_n) \\ \sin(\psi_n) \\ 0 \end{bmatrix} \quad (2.20)$$

As next, obtained LIDAR points were transformed from LIDAR attached coordinate frame to dome-camera attached coordinate frame using built-in function of ROS, $tf.TransformListener().transformPointCloud()$. Since the working frequencies and the time of measurements were different for camera and LIDAR, before transforming the

point cloud, 3D position of the LIDAR attached coordinate frame relative to the most recent position of the camera was calculated by using the built-in function *tf.TransformListener().waitForTransform()*. Afterwards, the point cloud was transformed one more time from dome-camera coordinate system to the i^{th} lens 3D coordinate system using the transformation matrix defined by the camera specific extrinsic parameters, which were retrievable for each camera attached to the Ladybug from the Ladybug API. Following the transformation of all LIDAR points to the i^{th} lens 3D coordinate system, direction vectors between local origin and scan points, which were redefined in the form of (2.21), were calculated using (2.22). Using the calculated direction vector, corresponding pixel locations on the rectified image, (c_n, r_n) , were found by (2.23) and (2.24), which were obtained by pinhole camera model as explained in [41].

$$P_n^{(i)} = [x_n, y_n, z_n, 1]^{(i)T} \quad (2.21)$$

$$\vec{v}_n^{(i)} = \left[\frac{x_n}{z_n}, \frac{y_n}{z_n}, 1 \right]^{(i)} \quad (2.22)$$

$$u_n = f_i \frac{x_n^{(i)}}{z_n^{(i)}} = c_n - c_0 \rightarrow c_n = f_i \frac{x_n^{(i)}}{z_n^{(i)}} + c_0 \quad (2.23)$$

$$v_n = f_i \frac{y_n^{(i)}}{z_n^{(i)}} = r_n - r_0 \rightarrow r_n = f_i \frac{y_n^{(i)}}{z_n^{(i)}} + r_0 \quad (2.24)$$

Lastly, pixel positions of the LIDAR points on the raw image were found by converting the obtained rectified pixel locations. Utilized algorithm for this conversion is explained in Section 2.4.

If the calculated pixel positions were non-integer values, then the color values for these pixels were estimated by using bilinear interpolation, which is represented in Figure 2-8 visually. In this interpolation method, four pixel positions with integer values, which covered the pixel in interest and shown as red in Figure 2-8, were found first. After reading the pixel values, Q_{11} , Q_{12} , Q_{21} and Q_{22} respectively, values R_{1x} and R_{2x} were estimated by doing a linear interpolation in x-axis using

(2.25) and (2.26). As next, by doing a secondary linear interpolation in y-axis using (2.27), the value of the pixel in interest was estimated as P_{yx} .

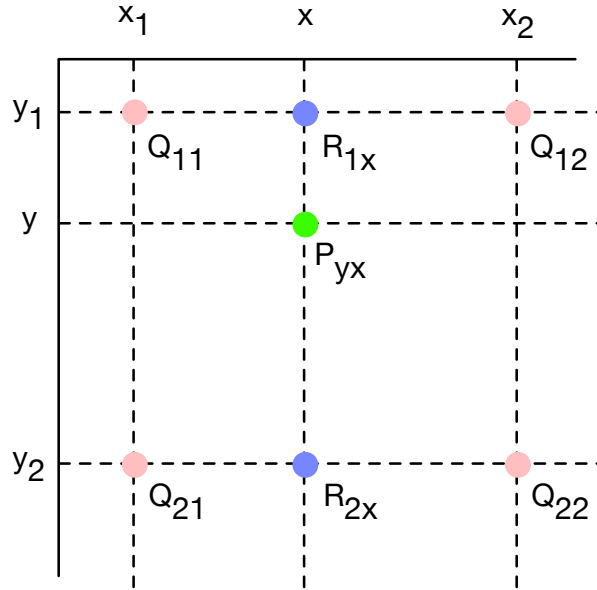


Figure 2-8 Visual representation of bilinear interpolation

$$R_{1x} \approx \frac{x_2 - x}{x_2 - x_1} Q_{11} + \frac{x - x_1}{x_2 - x_1} Q_{12} \quad (2.25)$$

$$R_{2x} \approx \frac{x_2 - x}{x_2 - x_1} Q_{21} + \frac{x - x_1}{x_2 - x_1} Q_{22} \quad (2.26)$$

$$P_{yx} \approx \frac{y_2 - y}{y_2 - y_1} R_{1x} + \frac{y - y_1}{y_2 - y_1} R_{2x} \quad (2.27)$$

2.4. Conversion of Rectified Pixel Locations to Raw Pixel Locations

In Section 2.3, it was explained in detail, how to calculate the measured LIDAR points' position on the rectified image. However, raw captured image by the camera is a distorted version of the actual scene as it is shown in Figure 2-9. In the figure, the image on the left is the raw image captured by the camera and is a barrel type distorted visual representation of the real scene. Hence, in order to read the color values for the LIDAR data, it was necessary to rectify the raw image or to transform

the calculated pixel locations from the undistorted image to the raw image. Since total number of pixels in the raw image was around 9800 times higher than the number of generated LIDAR points per cycle, and only a small portion of the whole image was used in the algorithms, it was decided to convert calculated rectified pixel locations to the raw pixel locations to minimize the computational load. Although a function to convert pixel values between raw and rectified images was available in Ladybug API, it was functional only on Windows OS, whereas all the algorithms and programs used in this study were running on UBUNTU OS. Therefore, five different methods for pixel conversion were developed and tested.



Figure 2-9 Captured image by the camera, a) raw image, b) rectified image

The first method was expressing the distortion in the raw image mathematically. As explained in [42], distorted images containing radial and tangential lens distortions can be modeled using (2.28), (2.29) and (2.30), where k_1 , k_2 and k_3 are radial distortion parameters, p_1 and p_2 are tangential distortion parameters, and u_{rect} , v_{rect} , u_{dist} and v_{dist} are the column and row coordinates in rectified and distorted images. To determine these parameters, Matlab Camera Calibrator App was used, which

estimated the parameters based on set of images captured by the camera. In all of these images, a distinct checkerboard with known square size was placed at different positions with changing rotation on the camera's field of view. The set of images used for the calibration process is given in Figure 2-10.

$$u_{dist} = u_{rect} \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \right) + 2p_1 u_{rect} v_{rect} + p_2 \left(r^2 + 2u_{rect}^2 \right) \quad (2.28)$$

$$v_{dist} = v_{rect} \left(1 + k_4 r^2 + k_5 r^4 + k_6 r^6 \right) + 2p_2 u_{rect} v_{rect} + p_1 \left(r^2 + 2v_{rect}^2 \right) \quad (2.29)$$

$$r^2 = u_{rect}^2 + v_{rect}^2 \quad (2.30)$$

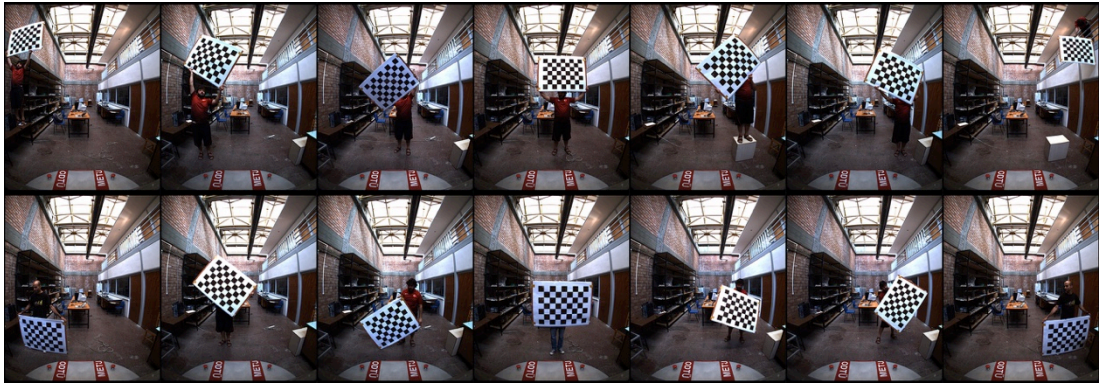


Figure 2-10 Camera images used in Camera Calibrator App of Matlab

The remaining methods utilized the pixel conversion table, which formed of both rectified-to-raw and raw-to-rectified conversions for each pixel in the image. This table was created by using the available function in Ladybug API, which can be used only on Windows OS.

In the second method, similar to the first one, the distortion in the image was expressed mathematically using (2.28) and (2.29), whose parameters were estimated based on the obtained pixel conversion table by using particle swarm optimization (PSO) technique. As explained in [43], PSO is an evolutionary optimization method, which uses particles and their motion to find their optimal positions and the corresponding optimal solution for the given problem. At the beginning of the PSO algorithm, a finite number of particles representing the parameter sets are generated with random positions and velocities. As next, fitness values of each of these

particles are calculated according to the defined cost function. Following the calculation of fitness values at the end of each iteration, best fitness value, p_{best} , for each particle is selected, and then the global best, g_{best} , is selected among them. Based on g_{best} , particles are updated and a new iteration starts till the maximum iteration number or the convergence is achieved. At the end, the parameter set corresponding to g_{best} is selected as the solution of the given problem. The success of this approach strongly depends on a well-defined cost function, which was taken in this study as the mean of the Euclidean distances between calculated and actual raw pixel positions for each pixel in the rectified image. The mathematical expression of the cost function is given in (2.31).

$$c = \frac{1}{N} \sum_{i=1}^N \sqrt{(x_{i,raw,calc} - x_{i,raw,act})^2 + (y_{i,raw,calc} - y_{i,raw,act})^2} \quad (2.31)$$

The third approach was reading the conversion file, in which the pixel conversion table was stored, and assigning this data to a variable at the beginning of the program. Then, required pixel conversions would be read from this variable during operation. However, this approach could never be tested, since reading the whole conversion table took more than a half hour and the ram usage reached to very high levels so that it slowed down other running processes.

The next method was writing the whole pixel conversion table to a file and reading the corresponding lines of it to convert the pixel locations at each LIDAR measurements' registration step.

The last method was storing the conversion table into a gray scale image instead of a text based file, where the rectified image pixels were used as pixel location in the generated gray scale image and the color value of the image was set to be equal to the output of pixel conversion, i.e. raw pixel position.

In the third and fourth pixel conversion methods, non-integer rectified pixel positions were estimated by using bilinear interpolation, which was explained at the end of Section 2.3.

2.5. Road Detection

In this study, three different adaptive road detection algorithms are proposed for a robust classification of road regions ahead of the vehicle in both constructed and unconstructed environments. The first proposed road detection algorithm utilized only one 2D LIDAR sensor attached to the front surface of the vehicle. The second approach, on the other hand, was based on using hybrid features, which consists of range measurements taken by the 2D LIDAR and the visual data coming from the camera. The third and the last algorithm used online supervised machine learning approach for road detection purpose. In the following sections, at first the working principle of single 2D LIDAR based adaptive road detection algorithm is explained. As next the second adaptive road detection algorithm using sensor fusion is covered in detail. At last, working mechanism of the machine learning based road detection algorithm is described.

2.5.1. Adaptive Road Detection Using 2D LIDAR Sensor

Proposed adaptive road detection algorithm consists of three main steps, namely breakpoint detection to differentiate changes in surfaces, line segment extraction to label possible road regions, and line segment selection to decide on drivable areas. Alternative to the similar studies in literature, to make the robotic system dynamically respond to the alterations in the road environment, an adaptive approach was utilized to estimate algorithm parameters by using discriminative learning approach. Moreover, all the calculations in the implemented road detection method were done using polar coordinates, which is the convention used by LIDAR sensors. Thus, the high computational cost of the coordinate conversion operation was avoided.

2.5.1.1. Breakpoint Detection

The aim of the breakpoint detection is determining the change in the road surface by looking any discontinuity in a range measurement. Furthermore, it prevents the line extractor from connecting two adjacent, linearly distributed point clouds in case of a presence of large discontinuity. A sample view of LIDAR points' distribution for structured and unstructured roads are given in Figure 2-11.

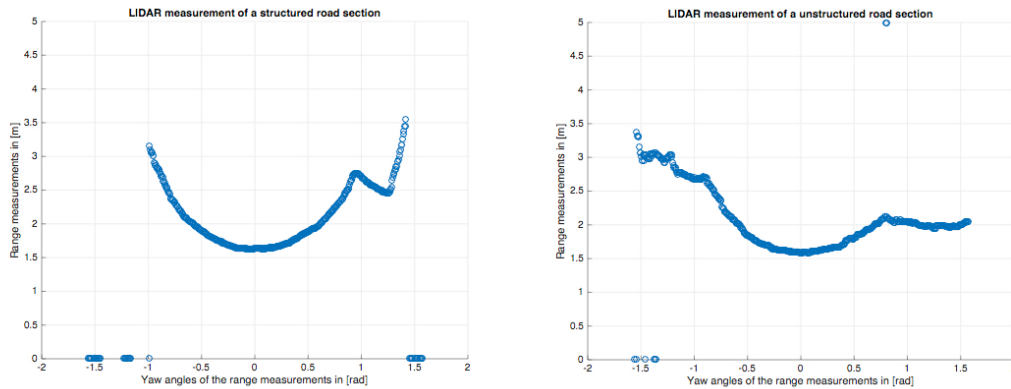


Figure 2-11 Sample LIDAR measurements for structured environment (left) and unstructured environment (right)

The method utilized for breakpoint detection in this study was based on the algorithm proposed by [38]. In this algorithm, at first the longest possible distance between current scan point p_n and previous scan point p_{n-1} was estimated. To calculate this distance, a virtual line passing through the scan point p_{n-1} making an angle λ with the scanning direction ψ_{n-1} was defined and a hypothetical scan point p_n^h was assumed to be on the intersection of this virtual line with the scanning direction ψ_n . The distance between p_{n-1} and p_n^h was taken as the threshold distance D_{\max} , whose mathematical expression was given in (2.32). Scan points p_{n-1} and p_n were labeled as breakpoints, if p_n was outside of the threshold circle, which was centered at p_{n-1} with a diameter of D_{\max} . Figure 2-12 visualizes the procedure followed for

breakpoint detection schematically, and the working steps are given as a pseudo code in Table 2-3. In Table 2-3, parameter N_{lp} stands for the total number of points scanned by the LIDAR in each cycle and BP is the cluster of detected breakpoints.

$$D_{\max} = r_{n-1} \frac{\sin(\Delta\psi)}{\sin(\lambda - \Delta\psi)} \quad (2.32)$$

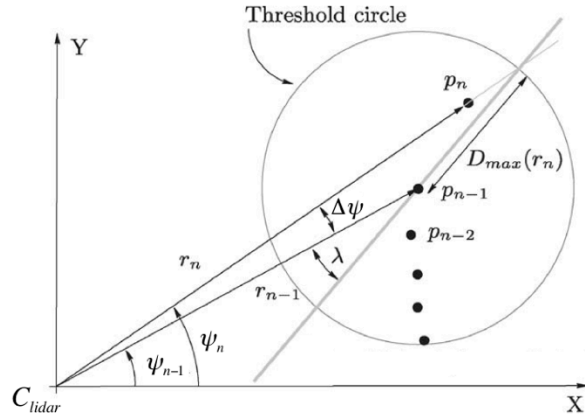


Figure 2-12 Schematic view of breakpoint detection and D_{\max} determination

Table 2-3 Pseudo code of breakpoint detection algorithm

01	$BP = \{ \}$	// Cluster of detected breakpoints
02	$p_0 \in BP$	
03	for n=1 to $(N_{lp} - 1)$	do
04	calculate D_{\max}	by (2.32)
05	if $d(p_n, p_{n-1}) > D_{\max}$	then
06	$p_n \in BP$	
07	$p_{n-1} \in BP$	
08	else	
09	$p_n \notin BP$	

Considering (2.32), calculation of D_{\max} depends on the range measurement r_{n-1} of p_{n-1} , angular increment between two consecutive LIDAR readings $\Delta\psi$ and λ . Only

adjustable variable in this equation is λ and the rest depends on the sensor specifications and environment. In this study, selection of λ was automated, whereas it was determined based on user experience in [25] and [38]. The method used for automated selection of λ is explained in detail at Section 2.5.1.4.

2.5.1.2. Line Segment Extraction

The next step of adaptive road detection algorithm is extraction of line segments, in other words possible road regions. In order to extract line segments, point distribution along a candidate point set was compared with the ideal range image of a flat ground using a similar algorithm proposed in [24] and the iterative end point fit algorithm (IEPF) presented in [25] and [38].

In the utilized algorithm, at first a hypothetical line representing the ideal range image of a flat surface between two consecutive breakpoints is defined by using (2.33) and (2.34). Next, distance of each measured point located between these breakpoints to the defined hypothetical line was calculated by (2.35) and compared with the threshold distance value d_{th} . If one or more of the points was found to be located in a farther position than d_{th} , the outermost scan point was labeled as a new breakpoint and the point set was split into two separate line segment candidates. This iteration was repeated until the distance criteria was satisfied for each point in the set and the set was selected to be a line segment, or the number of points in the set became less than the required minimum number of points, N_{min} , to represent a line segment and all the points were discarded at the end. The pseudo code of the algorithm is given in, and the IEPF step is visually presented in Figure 2-13.

$$\rho_{flat}(\psi_n) = \frac{z_{lidar}}{\cos(\psi_n)\alpha - \sin(\psi_n)\beta} \quad (2.33)$$

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 1 & -\tan(\psi_{n_s}) \\ 1 & -\tan(\psi_{n_e}) \end{bmatrix}^{-1} \begin{bmatrix} z_{lidar} / \cos(\psi_{n_s}) \rho(\psi_{n_s}) \\ z_{lidar} / \cos(\psi_{n_e}) \rho(\psi_{n_e}) \end{bmatrix} \quad (2.34)$$

$$d_n = |\rho(\psi_n) - \rho_{flat}(\psi_n)| \quad (2.35)$$

Table 2-4 Pseudo code of line segment extraction algorithm

```

01   $n_e = 0$ 
02   $LS = \{ \}$  // Cluster of extracted line segments
03  while  $n_e \leq N_{lp}$  do
04       $n_s = n_e$ 
05       $n_e = n_s + 1$ 
06      while  $n_e \notin BP$  and  $n_e < N_s$  do
07           $n_e = n_e + 1$ 
08      while  $n_e - n_s \geq N_{min}$ 
09          for  $n = n_s$  to  $n_e$  do
10              calculate  $d(p_n, p_{n,flat})$ 
11              if  $\max(d_n) > d_{th}$ 
12                   $n_e = index(\max(d_n))$ 
13              else
14                   $l_{n_s, n_e} \in LS$ 

```

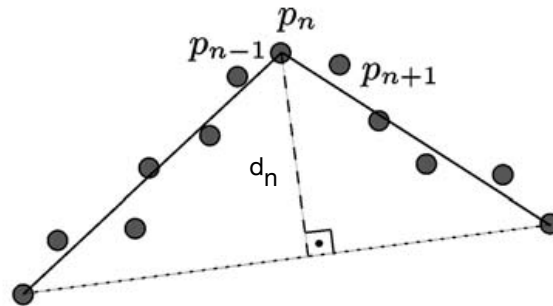


Figure 2-13 IEPF step of line segment extraction

Considering line segment extraction algorithm, determining whether a set of point was a line segment depends on the parameters N_{min} , d_{th} and the height of the LIDAR sensor z_{lidar} . Parameter z_{lidar} is a physical design parameter and cannot be changed during the operation. However, parameters N_{min} and d_{th} can be used to

dynamically control the output of the line segment extraction algorithm. In [25], these variables were set to be constant and same for different road types ($N_{\min} = 24$ and $d_{th} = 60mm$). Alternative to [25], determination of N_{\min} and d_{th} was automated in this study, which is explained in detail at Section 2.5.1.4.

2.5.1.3. Line Segment Selection

The last step of the proposed adaptive road detection algorithm is selection of line segment corresponding to the road area. For that purpose, at first central position of each line segment was calculated with respect to the GOAT. As next, the nearest line segment was selected and labeled as road among others based on the horizontal distance, in order to minimize the number of maneuvers.

2.5.1.4. Adaptive Parameter Selection

As mentioned in the previous parts of this chapter, deciding on the values of parameters used in road detection algorithm was automated. These parameters are λ , d_{th} and N_{\min} . This made the proposed road detection algorithm adaptable for changing road conditions.

Parameters λ and d_{th} were determined using a discriminative learning approach based on a labeled training data set. By LIDAR collected data is labeled as road regions using the path followed by mobile platform under manual operation, similar to [19] and [44]. In Figure 2-14, LIDAR points corresponding to the area passed over by GOAT were labeled as road and shown in blue, whereas the remaining points are colored with respect to their z values.

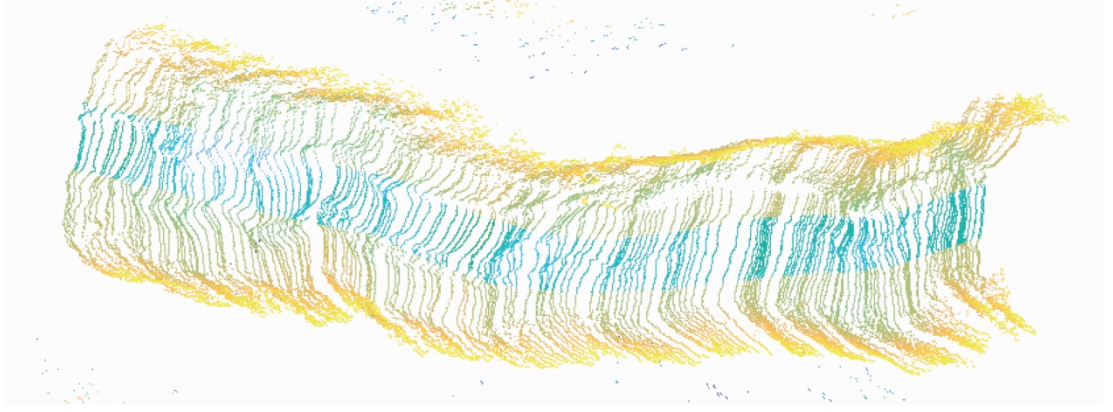


Figure 2-14 Terrain labeling for discriminative learning step, LIDAR points labeled as road region are shown in blue

To tune the parameter λ , Euclidean distance between each adjacent point p_{n-1} and p_n in the training data set was calculated at first and followed by determining the corresponding λ_n using (2.36). As next, optimum λ value is selected as the lowest peak value of the obtained distribution of λ_n .

$$\lambda_n = \sin^{-1} \left(r_{n-1} \frac{\sin(\Delta\psi)}{d(p_n, p_{n-1})} \right) + \Delta\psi \quad (2.36)$$

To tune the parameter d_{th} , distance d_n of each point labeled as road in the training data set to the virtual line representing the ideal range image of a flat surface was calculated using (2.35). Then the highest peak value in the obtained distance distribution was chosen to be d_{th} .

Different than the parameters λ and d_{th} , N_{min} was dynamically adapted throughout the motion of mobile platform, since the distance between the platform and the incident line may change, as the mobile platform passes on a bump and/or a hole. For that purpose, N_{min} was calculated depending on the width of the vehicle, w_{GOAT} , and the range measurement for $\psi = 0$, r_0 . The mathematical expression used to calculate N_{min} is given in (2.37).

$$N_{\min} = 2 \frac{\tan^{-1}\left(\frac{w_{GOAT}}{2r_0}\right)}{\Delta\psi} \quad (2.37)$$

2.5.2. Adaptive Road Detection Using 2D LIDAR Sensor and Visual Data

Proposed adaptive road detection algorithm, which uses hybrid features, consists of five main steps. First three stages of this algorithm, namely breakpoint detection, line segment extraction and line segment selection are same as in the previous method. Following the selection of line segment, remaining point cloud is registered to the preprocessed image and HSV color space values of the pixels matching with the range measurements are fused. At last, points, which form the selected line segment, are filtered based on discontinuity in their hue color channel values, to differentiate changes in surfaces with similar surface roughness.

One of the differences of this algorithm from the first approach is utilizing the visual data coming from the dome-camera to filter the point cloud. The other dissimilarity is conversion of the range measurement points from polar coordinate system to Cartesian coordinate system, in order to register them on the captured image.

2.5.2.1. Preprocessing and Color Space Conversion Of The Captured Image

In parallel to the road detection steps using LIDAR data only, captured image by the camera was first preprocessed in order to decrease the effect of noise and sharpen the color information. Afterwards its color space was converted from RGB to HSV to make the visual data less susceptible to the changing lighting conditions.

Application of the noise filter to the captured image was the first step of the preprocessing. In this step, a 3x3-averaging filter given in (2.38) was convolved with the raw image in order to remove grain noises due to poor lighting condition and high ISO values.

$$f = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.38)$$

Following the noise filtering, histogram equalization was applied on each color channels of the color image, in order to reduce the adverse effects of the unbalanced illumination and improve the contrast of the image. Although histogram equalization causes deformation in the color content, it was shown in [18] that utilization of histogram equalization improved the performance of color based road detection algorithms by decreasing the FP rate. Outputs of the preprocessing steps are presented together with the raw images in Figure 2-15.



Figure 2-15 Raw images captured by the camera (a,c) and preprocessed images (b,d)

After the preprocessing steps, image's color space was converted from RGB to HSV.

2.5.2.2. Filtration Of The Selected Line Segment Based On Hue Color Channel Value

After registering the selected line segment to the preprocessed image by following the methodology explained in Section 2.3, points were filtered based on the discontinuity in their hue color channel values to differentiate changes in surfaces with similar surface roughness.

The first step of the filtration was increasing the contrast in hue color channel by mapping color values to the interval of 0-255. Hue channel values of the points before and after contrast adjustment are plotted in Figure 2-16

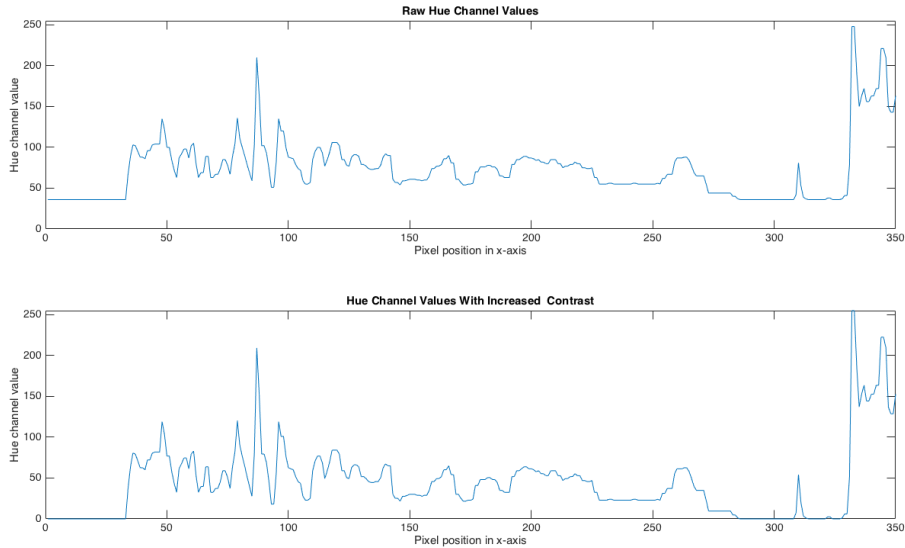


Figure 2-16 Hue channel values of the points in line segment before and after contrast adjustment

As next, mean and standard deviation values, μ_H and σ_H , in hue channel were calculated for all of the points in the line segment and upper and lower threshold values were defined as, $\mu_H + \sigma_H$ and $\mu_H - \sigma_H$. Following that, local average hue values for each 10 points in the line segment were calculated separately and compared with the threshold values. If average hue value of any point groups was found to be higher or lower than the threshold values, such as points between pixel positions 320-350 in Figure 2-17, those points were labeled as non-road region, assuming that the points corresponding to the actual road area has similar color distribution in hue channel and their ratio to the points on non-road area is higher. At last, if the line segment was divided into two or more parts, then central position of each line segment was calculated with respect to the GOAT. Next, the nearest line segment was selected and labeled as road among others based on the horizontal distance, in order to minimize the number of maneuvers.

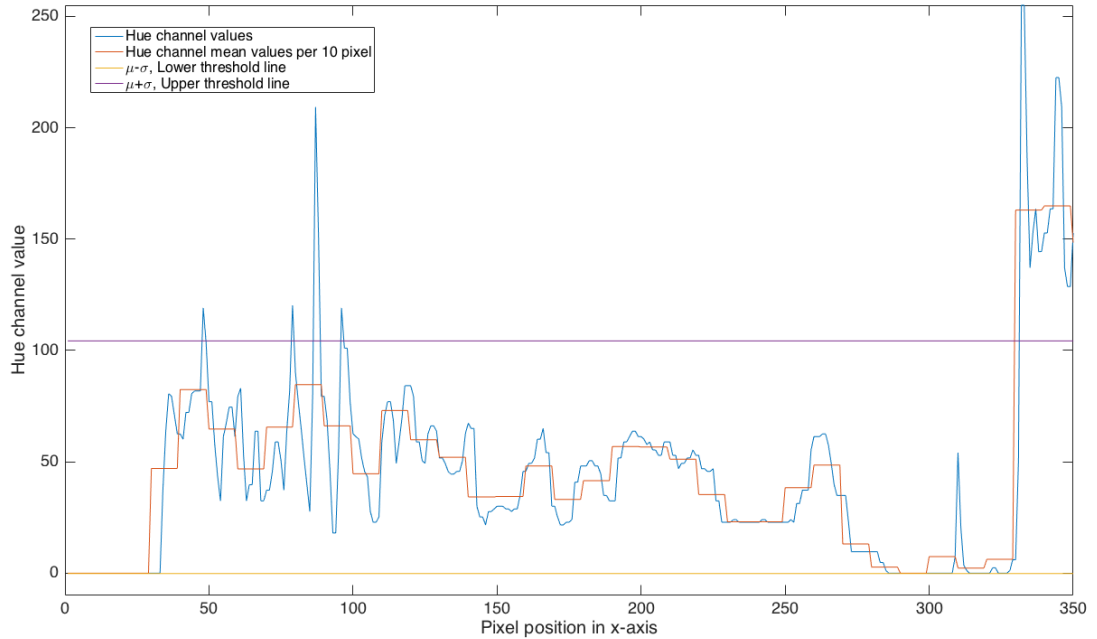


Figure 2-17 Hue channel values of the points forming the line segment and the average hue values of each point group after increasing contrast

2.6. Adaptive Road Detection Using Online Supervised Learning Approach

Proposed adaptive road detection algorithm consists of three main stages, which are generating training data, updating previously learned road models based on color properties and detecting road regions by using these road models and evaluating the images captured by the dome-camera.

2.6.1. Generation Of Training Data

For training data generation, road regions in front of the vehicle are detected by using adaptive road detection algorithm discussed in Section 2.5.2. As next, by following the procedure defined in Section 2.3 as road labeled LIDAR points are registered to the image, which is captured by the front facing lens of the dome-camera. By using the corresponding pixels' color values in HSV color space, visual model of the road region is generated as a mixture of Gaussians model. Similar to [18] and [39], these

Gaussian models are defined by their mean values, μ , covariance matrices, Σ and mass values, m , which are calculated by using k-means clustering. The mass value is equal to the number pixels forming the trained road model. While generating these models, different than [39], hue and saturation color channel values are used instead of RGB color space values, in order to make the models less susceptible to the changing light conditions.

2.6.2. Updating Previously Learned Road Models

After generating the k-many visual road models using training data, they are used either to update or to replace previously learned road models or to replace the null models. In order to decide on, whether these new models will be used for update or replacement, condition (2.39), which was proposed in [39], is used to compare new models with each of previous models.

$$1 \geq (\mu_L - \mu_T)^T (\Sigma_L + \Sigma_T)^{-1} (\mu_L - \mu_T) \quad (2.39)$$

If one or more of the newly generated model(s) fulfill the condition given in (2.39), then it/they are used to update the matching previously learned model(s) using (2.40), (2.41) and (2.42).

$$\mu_L = \frac{m_L \mu_L + m_T \mu_T}{m_L + m_T} \quad (2.40)$$

$$\Sigma_L = \frac{m_L \Sigma_L + m_T \Sigma_T}{m_L + m_T} \quad (2.41)$$

$$m_L = m_L + m_T \quad (2.42)$$

In case new models match none of the previously learned models, then they are used to replace null models, if there is any, or previously learned ones. If the number of null models is less than the number of new ones or equal to zero, then previously learned model with the lowest mass value is replaced.

2.6.3. Road Regions Detection

Following the update step, visual road models are used for road detection purpose. In order to do that, firstly pixels on the image captured by one of the lenses on the dome-camera are scored by the road models. For the calculation of these scores, Mahalanobis distance from pixel to the center of visual road models is used, which was shown to be most efficient distance measure compared to Manhattan, Euclidian, Chebyshev and Hellinger distances, [18]. The equation of Mahalanobis distance is given in (2.43).

$$d_M(p) = \sqrt{(p - \mu_L)^T \Sigma_L^{-1} (p - \mu_L)} \quad (2.43)$$

After calculating the scores for each pixel in the visual field, image is binarized by applying a threshold value equal to 3σ and drivability of each pixel is obtained. Before deciding on the road region, dilation and erosion filters with a kernel given in (2.44) are applied to the binary image to eliminate noises and impurities. At last, connected components in the binary image are found and the largest one is chosen as the road region.

$$f = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.44)$$

2.7. Experiments

Proposed localization and road detection algorithms were tested on the path shown in Figure 2-18, which consisted of structured and unstructured road regions, labeled as red and blue, respectively. The performance of the proposed localization method was observed throughout the whole route. On the other hand, two test areas for structured road (a road section bounded by sidewalks and another road section bounded by plants) and one test area for unstructured road conditions were selected for the evaluation of adaptive road detection algorithm. Figure 2-19 shows photos of the experiment sites for structured and unstructured road cases.

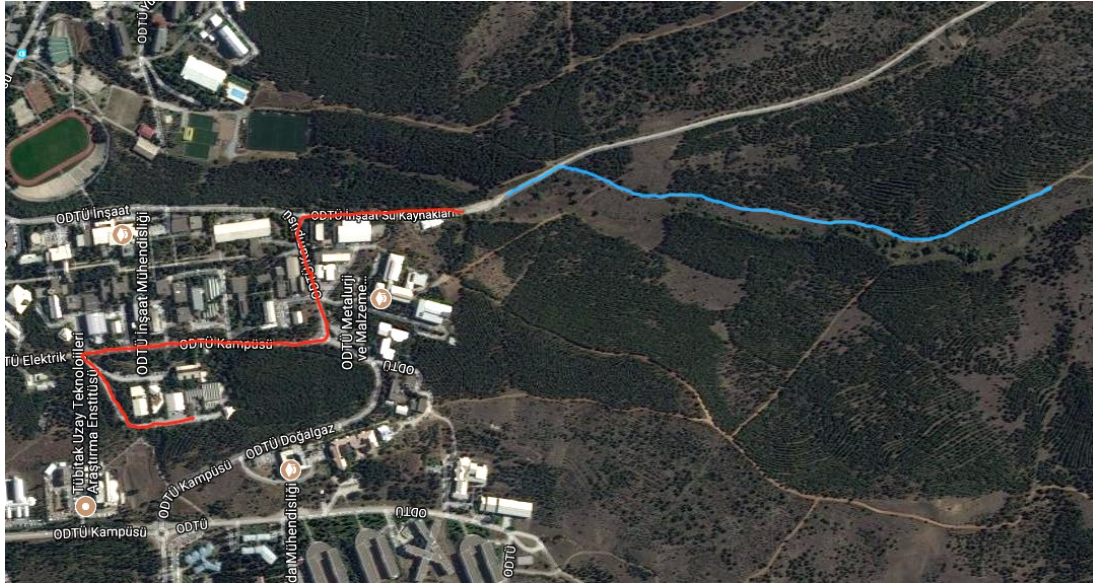


Figure 2-18 Test path driven by the vehicle, red path shows the structured road and blue path shows the unstructured road



Figure 2-19 Sample images for structured road (a) and unstructured road (b)

In order to compare the results of the developed algorithms objectively and to reduce the number of field tests, for each of which at least one hour needs to be spent excluding the data post-processing, the mobile platform was manually driven through

the path shown in Figure 2-18 with an average speed of 3 km/h, and whole sensor readings were recorded using the ROS environment. Later, the developed algorithms were run offline in the laboratory. By using the feature of playing recorded data in ROS, sensor readings were fed to the written programs like in real time, and made it possible to run and test all of the developed algorithms with the same inputs. Moreover, it made possible comparing the performances of each algorithm objectively.

The performance of the proposed localization algorithm was evaluated based on three different parameters. The first evaluation is done based on the coarse behavior of the mobile platform, where the calculated paths of the vehicle in structured and unstructured environments are visually compared with the aerial images published by ArcGIS. So that, it is checked whether the computed motion of the vehicle shows similar characteristics to the actual road. As next, the computed positions are evaluated in detail by considering the distances between two consecutive positions. The last evaluation criterion is the distance between estimated and actual final positions of the vehicle. To compare the final positions, the robotic platform was driven through a path, at the end of which the displacement of the vehicle was zero. So the distance between starting and ending positions gives the deviation between estimated and actual final positions of the vehicle.

Pixel conversion methods were evaluated both visually and quantitatively using the image presented in Figure 2-20.



Figure 2-20 Raw image of checkerboard captured by the Ladybug

Due to the distortion in the images, geometric features with linear shapes, such as straight edges of a cube, are bended. This effect is visualized using a mesh of squares in Figure 2-21 for negative and positive radial distortion cases together with the undistorted image. As the object gets closer to the boundaries, the effect of this phenomenon increases. The adverse effect of distortion caused by the camera's lens can be reversed by rectification, which was used to evaluate the visual performance of the proposed pixel conversion algorithms based on the straightness of the linear shaped geometrical features in the corrected images. In order to see the effect of rectification clearly, a checkerboard was chosen as the object in the image. Although the proposed method for visual evaluation is the inverse of the converting rectified pixels to raw pixels, rectification by using the same parameters with the distortion process gives the same input output pixel couples with the reverse order. Therefore it can be used for visual inspection.

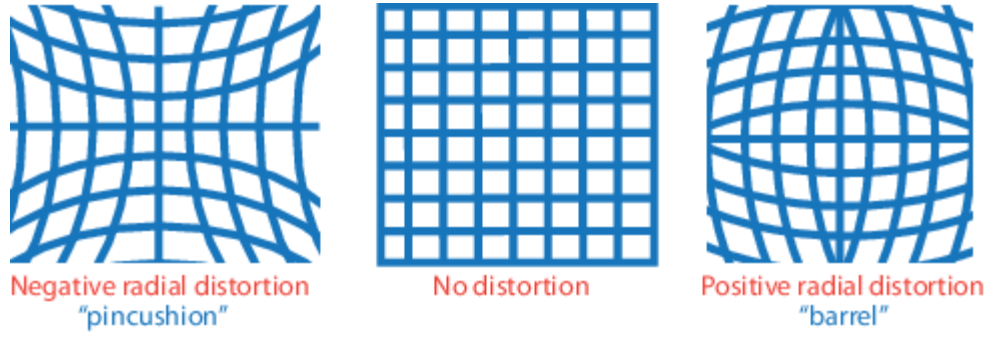


Figure 2-21 Effect of the radial distortion to the image [42]

The quantitative analysis was made based on the mean value of the Euclidean distances between the raw pixel positions obtained from Ladybug API and the calculated raw pixel positions by the proposed methods for each pixel location in the rectified image. The equation used for quantitative evaluation is given in (2.45).

$$\mu = \frac{1}{N} \sum_{i=1}^N \sqrt{(x_{i,raw,calc} - x_{i,raw,act})^2 + (y_{i,raw,calc} - y_{i,raw,act})^2} \quad (2.45)$$

Performance of the proposed road detection algorithms was evaluated based on ACC and FPR values of the outputs. To determine the ACC and FPR, line segments collected during the experiments and labeled as road area were compared with the ground truth data using (2.46) and (2.47). The ground truth data was generated by manual labeling of the road edges on the images, which were captured by the dome-camera. A sample view from the manual labeling process is given in Figure 2-22 for structured environment. FPR was used for comparison, however true negative and false negative results were not considered for the evaluation of algorithm's performance. This is due to that a false negative result can only make the robot deciding to halt and wait for manual operation, at worst. False positive output, on the other hand, may cause the vehicle under autonomous operation decide on going out of the road, and can lead to damaging the UGV platform. In addition to ACC and FPR, computational speeds of the proposed adaptive parameter approaches were compared against each other using the mean computational time required by them.

$$ACC = \frac{TP + TN}{\Sigma p} \quad (2.46)$$

$$FPR = \frac{FP}{\sum p_{negative}} \quad (2.47)$$



Figure 2-22 Sample ground truth data generation by labeling road region manually

CHAPTER 3

RESULTS AND DISCUSSION

In this chapter of the thesis, results of the conducted experiments are presented to evaluate the performance of the developed algorithms for localization, pixel conversion and road detection. Then the performance of the overall system is evaluated in detail based on these results.

3.1. Localization

In this section, first, the driven path for the evaluation of localization algorithm is virtually presented attached to the aerial images published by ArcGIS. This path was estimated by using the outputs of three different approaches, namely, GPS only, OBMM only and Extended Kalman Filter that combines GPS, OBMM and IMU sensor readings. Following the visual result, performance of the algorithm is evaluated numerically with respect to the criteria discussed in Section 2.7.

Visual results of the travelled path computed based on GPS only, OBMM only and EKF are shown in Figure 3-1 for structured and unstructured road environments. As it can be seen in Figure 3-1-a and -b, output of the localization method based on only GPS data visually agrees with the actual road area shown on ArcGIS World Aerial Images. Results of the OBMM, on the other hand, deviates from the actual path as expected, because small errors in the measurements cause cumulative increase in the final positioning error due to relative localization approach. However, considering

the general behavior of the computed path, it is seen that obtained results by OBMM based localization method is similar to GPS based method. Visual results of the estimated path by EKF algorithm also agree with the road area shown in ArcGIS World Aerial Images, and for the unstructured road case shown in Figure 3-1-b it coincides with the path computed by GPS data.

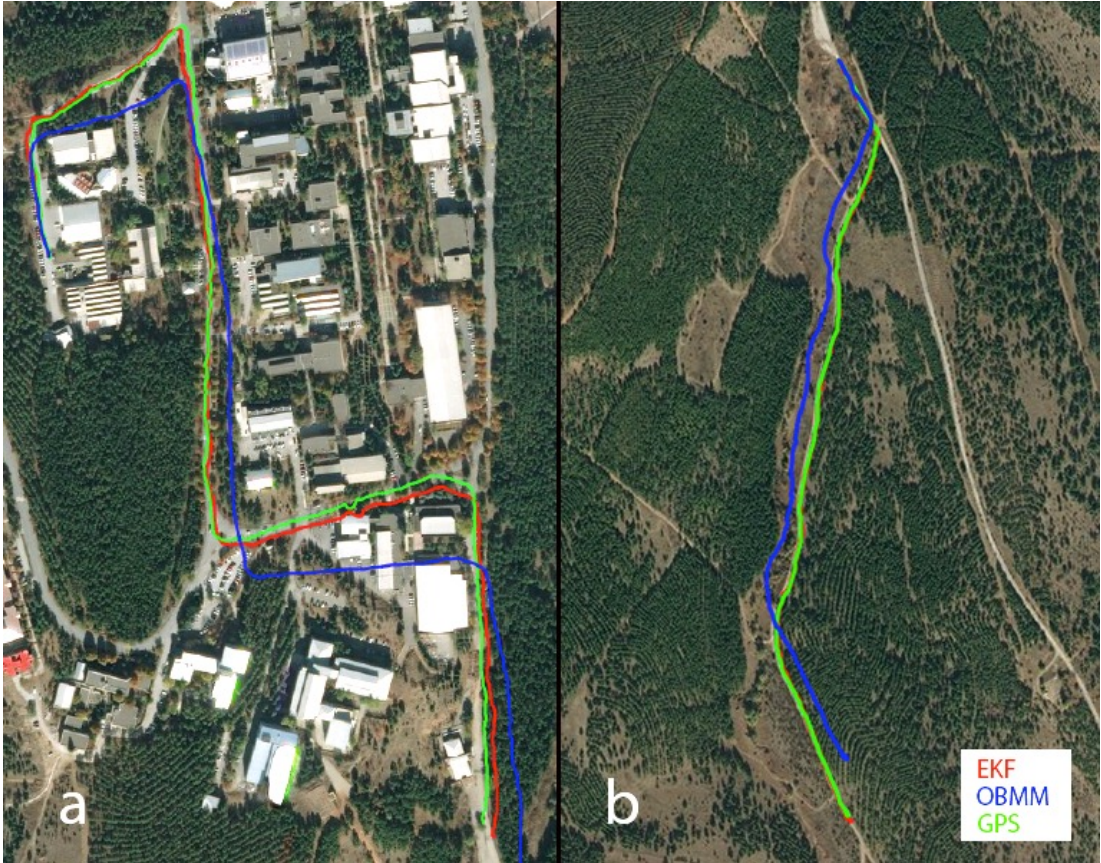


Figure 3-1 Travelled path by the mobile platform to test localization methods in a) structured environment and b) unstructured environment.

As next, the motion of the vehicle is inspected more closely. In Figure 3-2, a small portion of the path computed by EKF localization algorithm is shown. As it can be seen in the right most visual, generated path has a shape of sawtooth. Although it is rarely observed throughout the path, it occurred because of the difference in the working frequencies and the uncertainty values of the localization methods combined

by EKF algorithm. A sample close-view of the computed path is shown in Figure 3-3. As it can be seen in the figure, number of positions calculated by GPS data is less than the number of positions calculated by other two methods due to low working frequency. Between two successive GPS readings, localization is done based on IMU and OBMM data. Since these two methods are incremental localization methods, the error in the positioning increases cumulatively, whereas GPS is an absolute positioning sensor and its error in positioning only depends on the number of connected satellites, but not former measurements. Therefore, the relative localization error between two positions estimated by an incremental localization method, p_{t-1} and p_{t-2} , is smaller than the relative localization error between positions p_t and p_{t-1} , where p_t is estimated based on GPS data. As a result of this, some parts of the path have the sawtooth shape. While going through the sawtooth shaped sections of the path, it was seen that this fact only occurred when the vehicle made sharp and sudden turns. In other words, the reason for why the sawtooth shape is observed is that in these parts of the path the neglected lateral and longitudinal slippage values affect the motion more drastically, and cause increasing the error in pose estimation of OBMM.

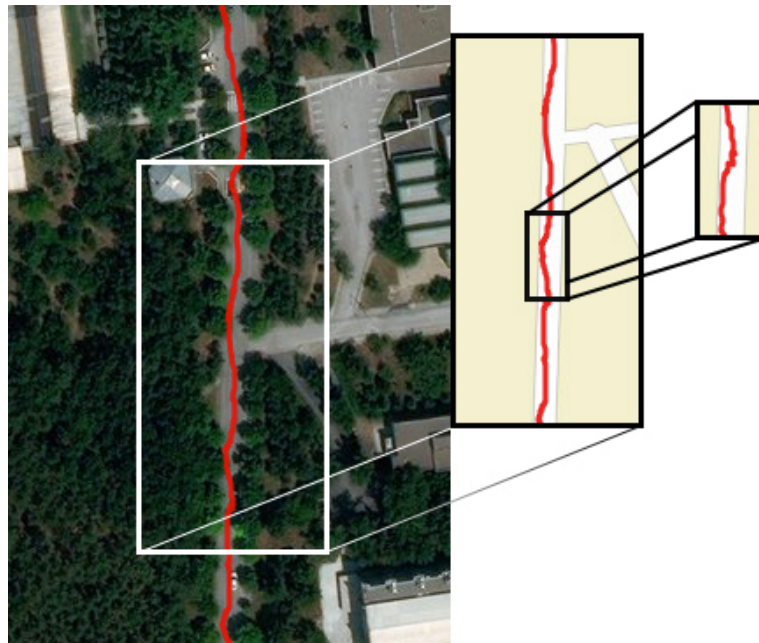


Figure 3-2 A sample section view of the computed path

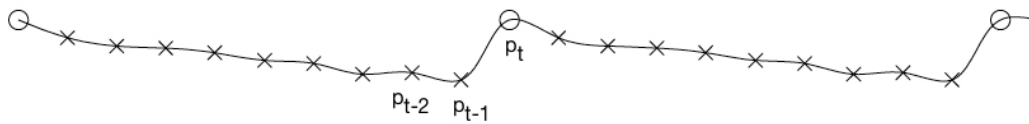


Figure 3-3 Close-view of the estimated path. Data points generated by using GPS data are shown as circle, and data points generated by using IMU and OBMM data are shown as cross.

The third and last evaluation criterion for the localization method is the distance between estimated and actual final positions of the mobile platform. To calculate this distance, the mobile platform was driven through the path shown in Figure 3-4, where the starting and ending positions of the path are on the left center of the image. Although the final position of the GOAT was actually the same with the starting position, computed values by the EKF localization algorithm differs. The distance between these two positions is calculated as 1.014 m after completing the 1.98km length path. In other words, the error in the final position relative to the total traveled distance is found to be 0.051%.



Figure 3-4 Traveled route by GOAT computed by EKF localization algorithm

Considering both the visual and numerical results, it was seen that the proposed localization algorithm, which fuses GPS, OBMM and IMU data, could be used both in structured and unstructured environments. However, since an increase in the positioning error was observed when the vehicle made sharp and sudden turns, speed of the vehicle should be controlled carefully to minimize the lateral and longitudinal slippages.

3.2. Conversion of Pixel Locations Between Rectified and Raw Images

In this section, pixel conversion methods explained in Section 2.4 are evaluated both visually and quantitatively.

To start with, the output image of the rectification function in the Ladybug API was visually interpreted to check the accuracy of the calibration file supplied by the PointGrey. As explained in the Section 2.7, visual evaluation of the algorithms was made based on the straightness of the lines in the rectified image, which is given in Figure 3-5. As it can be seen in Figure 3-5-a, correction of the curved lines in the raw image were successfully undistorted, even though the inspected area was considerably affected from the lens distortion due to its close position to the image's corner. In the second sub-image shown in Figure 3-5-b, on the other hand, a slight deviation was observed between one of the edges of the corrected squares and the ideal straight line. However, this deviation only occurred on one of the edges and this edge was close to the lower part of the checkerboard, which was deformed due to the weight of the board itself. So, it was concluded that this difference was due to the physical deformation in the board instead of any inaccuracy in the rectification algorithm. Since the result shown in Figure 3-5, which was obtained by using the rectification function of the Ladybug API, were visually accurate and successful, it was used to evaluate the performance of the proposed algorithms' outputs.

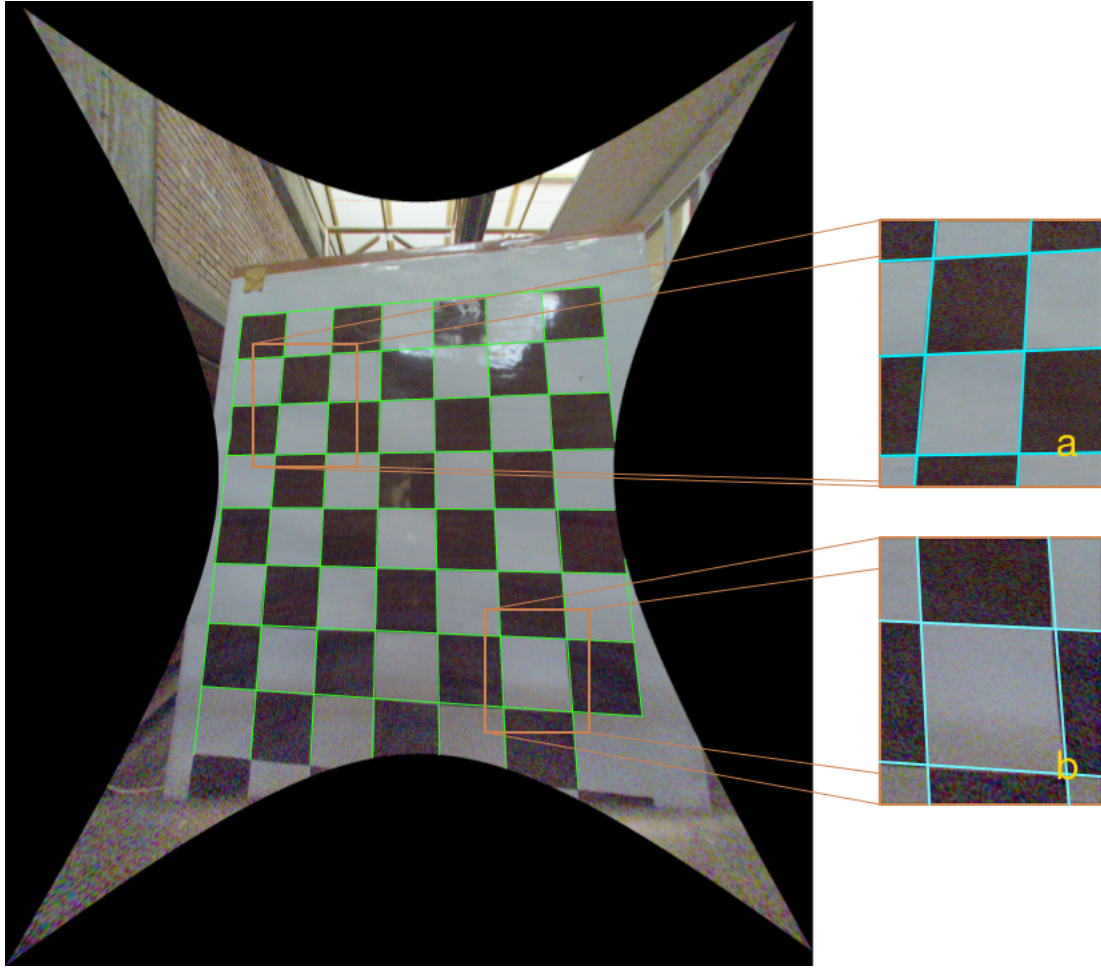


Figure 3-5 Undistorted image by using the rectifying function of Ladybug API

The first method for pixel conversion was expressing the distortion in the image mathematically using equations (2.28), (2.29) and (2.30), whose parameters were estimated using Matlab Camera Calibrator App. Conversion equations with the estimated parameter values are given in (3.48), (3.49) and (3.50). As it can be seen in Figure 3-6, mathematical model with Matlab estimated parameters could restore the distortion in the middle area of the image. However, performance of the algorithm decreased in the regions close to the image boundaries, which can be seen through Figure 3-6-a and Figure 3-6-b. In the quantitative analysis, the mean value of the Euclidian distances between actual and calculated raw pixel positions was found as 5.06 pixels.

$$u_{raw} = u_{rect} (1 - 0.2315r^2 + 0.0393r^4) \quad (3.48)$$

$$v_{raw} = v_{rect} (1 - 0.2315r^2 + 0.0393r^4) \quad (3.49)$$

$$r^2 = u_{rect}^2 + v_{rect}^2 \quad (3.50)$$

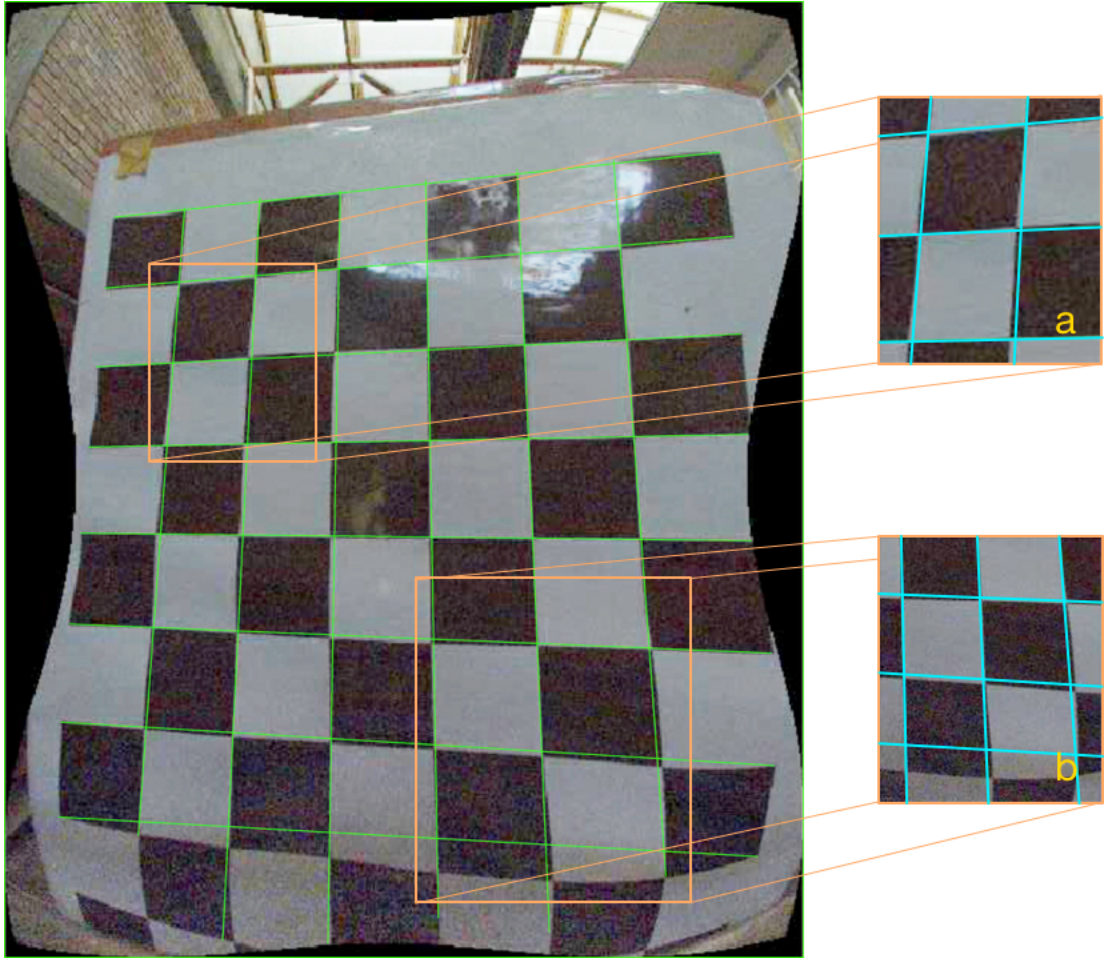


Figure 3-6 Undistorted image obtained by using equations (2.28), (2.29) and (2.30) with the parameters found by Matlab Camera Calibrator App

Although the proposed model initially included the tangential distortion, it was omitted from the final version of the equations, since the calibration toolbox could not estimate them correctly. After trying estimation of the parameters with different image sets for several times, the best visual result obtained with tangential distortion added model is given in Figure 3-7. One of the reasons for the unsuccessful rectification result was the small number of calibration images close to the edges.

Another reason was small coverage area of the checkerboard in the calibration images, which was suggested to be around 20% of the whole image. However, due to camera's large field of view and the necessity of capturing images at a distance close to the working distance, which is from the camera to the object of interest during the real usage of the setup, it was not possible to add extra calibration images, where the checkerboard was positioned close to the top edge of the image while covering at least 20% of the camera's field of view.

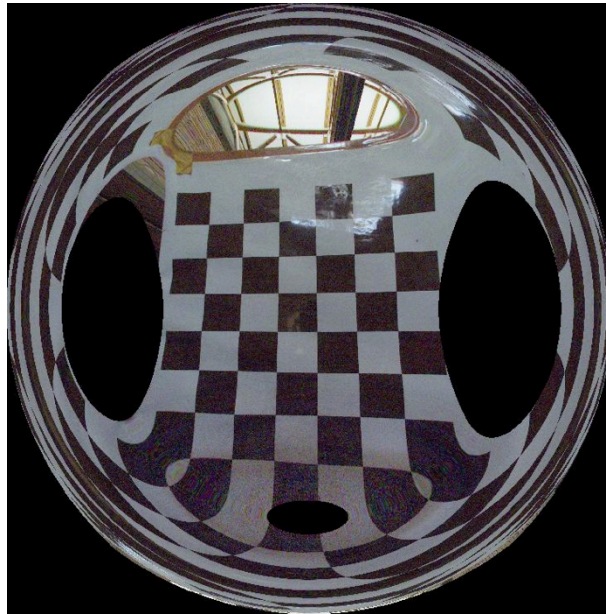


Figure 3-7 Undistorted image obtained by using equations (2.28), (2.29) and (2.30) including the tangential distortion parameters found by Matlab Camera Calibrator App

In the second proposed method, similar to the first one, the pixel conversion was made by using the mathematical expressions given in equations (2.28), (2.29) and (2.30). Parameters of these equations were estimated by using PSO with 50 particles after running 50 iterations and are given in the conversion equations (3.51), (3.52) and (3.53). As it can be seen in Figure 3-8, estimated parameters could only restore top right corner of the image locally with a poor performance. As expected, mean value for the distance between actual and calculated raw pixel positions was found higher than the first method, which was 23.99 pixels for the whole image.

$$u_{raw} = u_{rect} \left(1 - 0.3567r^2 + 0.1348r^4 - 1.0150 \times 10^{-14} r^6 \right) \quad (3.51)$$

$$v_{raw} = v_{rect} \left(1 - 0.3567r^2 + 0.1348r^4 - 1.0150 \times 10^{-14} r^6 \right) \quad (3.52)$$

$$r^2 = u_{rect}^2 + v_{rect}^2 \quad (3.53)$$



Figure 3-8 Undistorted image obtained by using equations (2.28), (2.29) and (2.30) with the parameters estimated by PSO

The next method was writing the whole pixel conversion table to a file and reading the corresponding lines of it to convert the pixel locations at each LIDAR measurements' registration step. Visual result of this algorithm was same as the Figure 3-5. Average Euclidian distance was calculated as 0.19 pixels. Although the outputs of the algorithm agree with the actual data both visually and quantitatively,

the conversions lasted around 2 seconds per cycle. Since the algorithm needed to operate at 10 Hz at least, it could not be utilized in this study.

The last method was saving the pixel conversion table as a gray scale image, where the color value of the pixels were set to the output of pixel conversion, in other words to the raw pixel locations. Similar to the previous method, the visual result of this algorithm was same as the Figure 3-5. Also, the average Euclidian distance was calculated as 0.19 pixels, which is equal to the previous one. The most important advantage of this algorithm was decreasing total time of pixel conversion and color reading steps to 0.0171 seconds while keeping accuracy of the results constant. Moreover, file size for storing the conversion data was reduced from 300 MB to 685 kB. As a result of these, it was decided to utilize this method for the conversion of rectified pixel locations to raw pixel positions.

3.3. Road Detection

In this section of the thesis, sample outputs from different steps of the adaptive road detection algorithms are presented at first. Secondly, performance of the proposed road detection algorithms are evaluated based on ACC and FPR values as discussed in Section 2.7.

3.3.1. Adaptive Road Detection Using 2D LIDAR Sensor

Point cloud captured by the LIDAR is shown at different stages of the road detection algorithm in Figure 3-9. In Figure 3-9-a, raw measured point cloud is attached on the ground truth image captured by the dome-camera after manual labeling. The cyan points correspond to the road area, whereas the magenta points correspond to the non-road area. Figure 3-9-b visualizes the resultant point cloud after the breakpoint detection and line segment extraction steps of the road detection algorithm. Breakpoints are given as red and the rest of the points forming the line segments are colored as cyan. In Figure 3-9-c, only as road labeled points are shown on the ground truth image, in which the blue color stands for TP and red color stands for FP results.

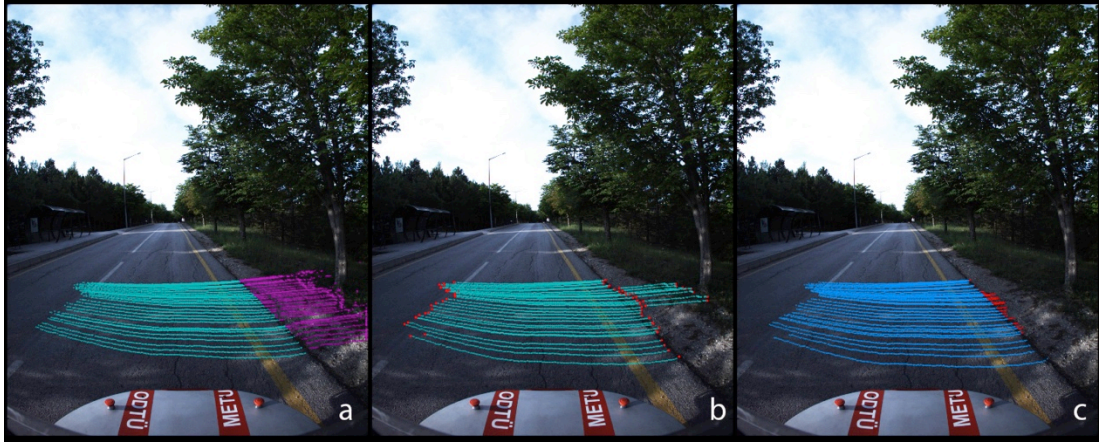


Figure 3-9 Measured points on SR by LIDAR shown on the ground truth image a) raw point cloud, b) point cloud after breakpoint detection and line segment extraction by utilizing Fully-adapted PS for SR, c) point cloud forming the line segments labeled as road are by utilizing Fully-adapted PS for SR

For the evaluation of adaptive road detection algorithm using 2D LIDAR sensor only, different parameter sets (PS) with changing levels of adaptation were tested for the same data sets obtained from SR and UR experiments. For constructed road case, three different parameter sets shown in Table 3-1 were used. Predefined PS consisted of predetermined parameter values, as the name suggests. Parameter values of Predefined PS were taken the same with the values specified in previous studies [38] and [25]. It was implemented as a baseline to compare adapted parameter sets. In Semi-adapted PS, parameter adaptation was performed for only λ and d_{th} during the learning period. N_{min} was left out from the adaptation process on purpose for this PS to single out the effects of surface based parameters. For Fully-adapted PS, adaptation process was extended to cover all three parameters as explained in Section 2.5.1.4.

Table 3-1 Parameter sets used for the experiments conducted in SR environment

	$\lambda [^\circ]$	$d_{th} [mm]$	N_{min}
Predefined PS	10.00	60.00	24.00
Semi-adapted PS for SR	11.77	35.00	24.00
Fully-adapted PS for SR	11.77	35.00	66.00

Results of the experiments for SR Sections 1 and 2 are given in Table 3-2. The results of Predefined and Semi-adapted parameter sets indicated that using an adaptive approach for the estimation of the surface dependent parameters, λ and d_{th} , can reduce the FPR up to 6.60%. Moreover, comparing the results for Semi-adapted and Fully-adapted parameter sets for SR showed that adaptive estimation of N_{min} could enhance the decrease in FPR up to 6.16% for one case and 14.91% for the other case. Although 1.33% relative drop for SR Section 2 was observed in ACC, when Fully-adapted PS was utilized, decrease in the FPR would improve the performance of the overall system by decreasing the chance of directing vehicle out of the road as explained in Section 2.7. In Figure 3-10-a, the image of the SR Section 2 is given. As it can be seen in the figure, surface elevations and roughness properties of the road and the start of the stairs next to the sidewalk were similar. Considering the visual result of the adaptive road detection algorithm shown in Figure 3-10-b, change in the surface type for this region cannot be detected by the algorithm using LIDAR data only, which caused a higher FPR value for SR Section 2.

Table 3-2 ACC and FPR values for three different parameter sets computed on two different sections of SR

	SR Section 1		SR Section 2	
	FPR	ACC	FPR	ACC
Predefined PS	6.95%	89.65%	26.74%	96.13%
Semi-adapted PS for SR	6.60%	89.87%	19.11%	94.35%
Fully-adapted PS for SR	6.16%	95.37%	14.91%	94.80%

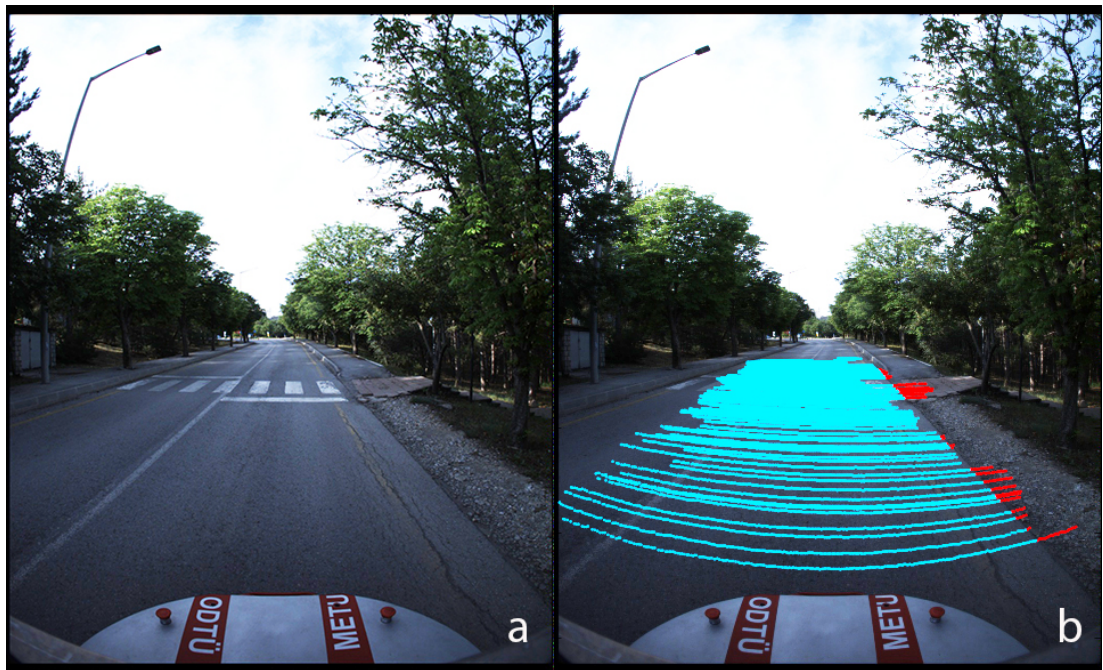


Figure 3-10 Image of SR Section 2, a) raw image of the scene, b) LIDAR points labeled as road where TP and FP results shown in cyan and red, respectively

In unstructured road experiment, again three different parameter sets were utilized which are given in Table 3-3. The same Predefined PS used in the SR case was also used as a baseline for UR experiments. Secondly, Fully-adapted PS for the constructed road case was selected to observe the effect of change in the environment

to the performance of an estimated parameter set. At last Fully-adapted PS for UR was determined based on adaptation process covering all the three parameters and tested.

Table 3-3 Parameter sets used for the experiments conducted in UR environment

	$\lambda [^\circ]$	$d_{th} [mm]$	N_{min}
Predefined PS	10.00	60.00	24.00
Fully-adapted PS for SR	11.77	35.00	66.00
Fully-adapted PS for UR	2.00	110.00	66.00

Results of tests made in UR environment Section 1 are shown in Table 3-4. According to these results, unlike in the constructed road case, using Fully-adapted PS for SR could only decrease the FPR by 0.33%. Moreover, ACC of the output decreased by 0.56% and as it can be seen in Figure 3-11, the algorithm could not detect left half of the road area in the environment. On the other hand, Fully-adapted PS for UR caused an increase in the FPR by 1.72%, while improving ACC from 66.36% to 86.04%. The reason of this increase in the FPR is due to that the algorithm could only detect one half of the road region when Fully-adapted PS for SR was utilized whereas it could find the whole road area when Fully-adapted PS for UR was used. The improvement in the detection rate is also led to an increase in the FPR by adding false positive results on the left hand side of the road. Moreover, the visual results pictured in Figure 3-11 support these numerical data. Comparing Figure 3-11-b and c shows that determining the parameter set specific to road condition improves the performance by increasing the number of truly labeled LIDAR points corresponding to road area, which agrees with the change in ACC presented in Table 3-4.

Table 3-4 ACC and FPR values for three different parameter sets computed for unstructured road

	UR Section 1	
	FPR	ACC
Predefined PS	17.67%	66.36%
Fully-adapted PS for SR	17.34%	64.92%
Fully-adapted PS for UR	19.39%	86.04%

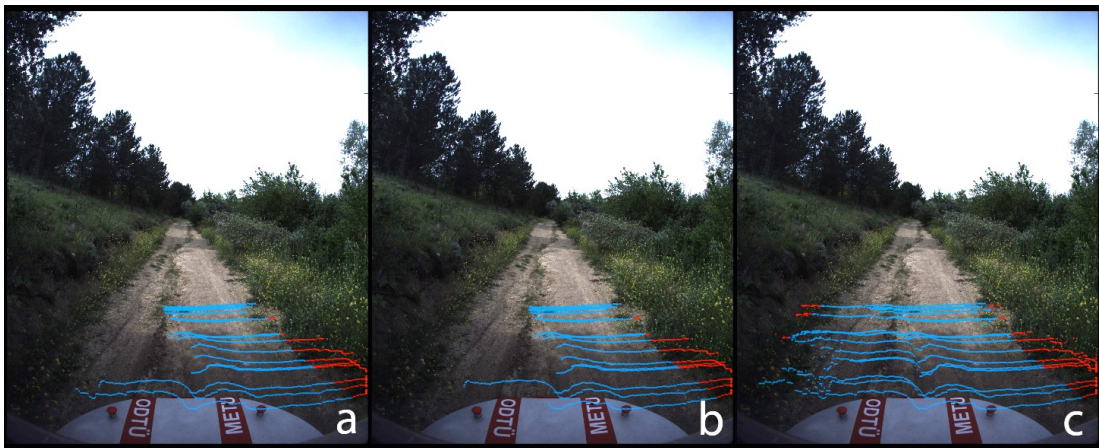
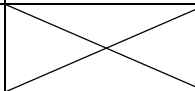
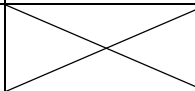


Figure 3-11 Points labeled as line segment shown on the ground truth image of UR section, a) Predefined PS, b) Fully-adapted PS for SR, and c) Fully-adapted PS for UR

In Table 3-5, mean process times for one cycle of road detection algorithm with indicated parameter sets are shown for both structured and unstructured roads. In addition to the improvements made in the algorithm's outputs in terms of ACC and FPR, the result for the SR case presented in Table 3-5 shows that using an adapted parameter set can enhance the performance of the overall road detection algorithm by decreasing the average process time by 50.93% relatively. This decrease in the process time is due to the less number of iterations made in the line segment

extraction step, which is because of more accurate extraction of breakpoints at the first step of the algorithm. On the other hand, the mean process time for the UR increased by 63.29% relatively, when the Fully-adapted PS for UR was utilized. The main reason for the increase in the process time is because of the increased number of breakpoints and iterations made in line segment extraction step due to higher surface roughness. However, the increase in the process time of the algorithm can be neglected considering that the successful detection of road regions is more important for navigation of the mobile platform in an unknown environment, and Fully-adapted PS for UR could detect road area with a 86.04% accuracy, which is 19.68% higher than the Predefined PS.

Table 3-5 Mean process times for one cycle of road detection algorithm (for $N \approx 7500$ LIDAR measurement sets)

Road Type	Parameter Set	Mean process time per scan [sec]	Decrease in process time
Constructed Road	Predefined PS	0.0088204	
	Fully-adapted PS for SR	0.0043281	50.93%
Unstructured Road	Predefined PS	0.0059070	
	Fully-adapted PS for UR	0.0096454	-63.29%

3.3.2. Adaptive Road Detection Using 2D LIDAR Sensor And Visual Data

In order to evaluate the performance of adaptive road detection algorithm utilizing geometrical and visual data, it was tested using the Fully-adapted PS in the structured and unstructured environments. Then, its results were compared with the previous findings.

The first road detection algorithm using LIDAR data only could not detect the change in the surfaces with similar geometrical properties, such as in the case of SR Section 2, where the algorithm could not separate the road area from the start of the stairs next to the sidewalk shown in Figure 3-10-b. Therefore, in order to observe the improvements in the algorithm's performance more clearly, proposed approach based on geometrical and visual data was firstly tested in SR Section 2. Result of the test together with the previous finding is given in Table 3-6. Considering these results, it is shown that using visual data together with geometrical data improved the performance of the algorithm by decreasing the FPR from 14.91% to 6.24%. Although 7.56% relative drop was observed in ACC, decrease in the FPR would improve the performance of the overall system by reducing the chance of directing vehicle out of the road as explained in Section 2.7. However, by inspecting the image of the road in Figure 3-12-a and the visual result in Figure 3-12-c, it can be seen that some of the line segments' end points correspond to the continuous road boundary line on the right hand side. Since this line defines the formal boundary of the road, the decrease in ACC can be neglected.

Table 3-6 ACC and FPR values calculated for the proposed adaptive road detection algorithms on SR Section 2

	SR Section 2	
	FPR	ACC
Based on LIDAR data only with Fully-adapted PS for SR	14.91%	94.80%
Based on LIDAR and visual data with Fully-adapted PS for SR	6.24%	87.24%

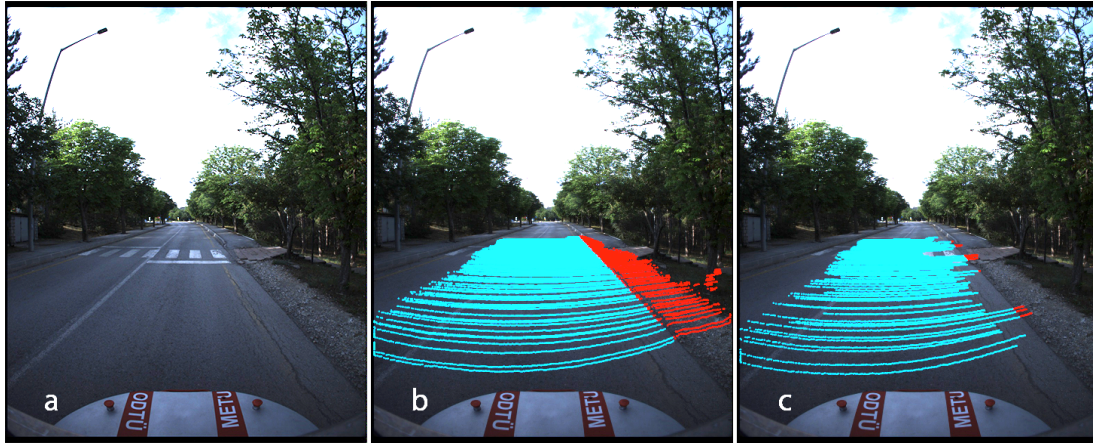


Figure 3-12 Measured points on SR Section 2, a) image of the scene, b) raw point cloud on image, c) point cloud labeled as road region after running road detection algorithm based on LIDAR and visual data, where TP and FP results shown in cyan and red, respectively

The second test for the proposed algorithm was made in unstructured road environment. As shown in Figure 3-13, the algorithm could separate the road region from the surrounding plants more successfully compared to the first road detection algorithm. Considering the numerical results given in Table 3-7, the integration of visual data caused a 10.76% drop in the FPR values relative to the first proposed approach. Moreover, ACC increased from 86.04% to 89.23%.

Table 3-7 ACC and FPR values calculated for the proposed adaptive road detection algorithms on UR Section 1

	UR Section 1	
	FPR	ACC
Based on LIDAR data only with Fully-adapted PS for UR	19.39%	86.04%
Based on LIDAR and visual data with Fully-adapted PS for UR	8.63%	89.23%

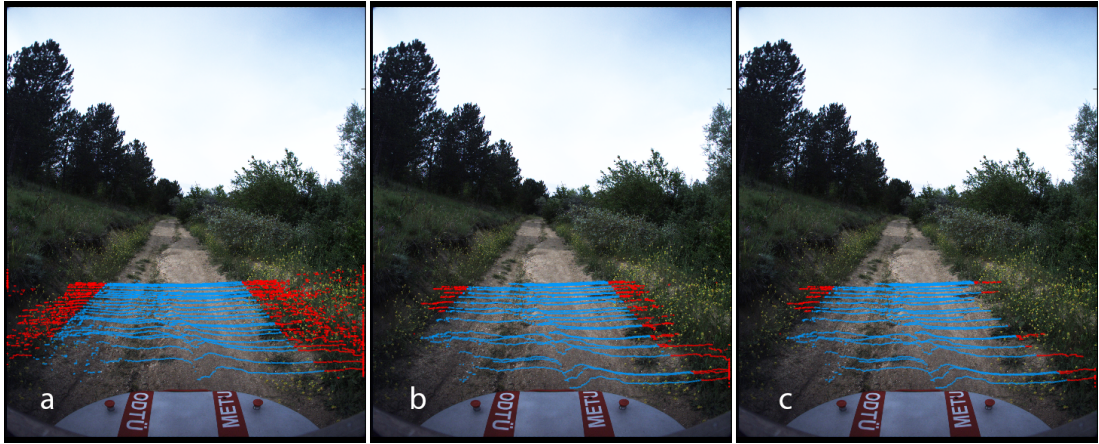


Figure 3-13 Measured points on UR Section 1, a) raw point cloud on image, b) point cloud labeled as road region after running road detection algorithm based on LIDAR only, c) point cloud labeled as road region after running road detection algorithm based on LIDAR and visual data, where TP and FP results shown in cyan and red, respectively

Lastly, proposed road detection algorithm was tested throughout the entire path both in structured and unstructured environments, each of which were around 1 km long. Obtained test results with different parameter sets are given in Table 3-8 and Table 3-9. After comparing these results with the previously presented ones, it can be seen that they agree with each other. According to these findings, increase in the TPR shows that using adaptive approach for determination of the road detection algorithm parameters improves the road detection rate in unstructured environment. Moreover, integrating visual data to the geometrical features enhances the performance of the road detection algorithm by decreasing the FPR value.

Table 3-8 ACC and FPR values calculated for the proposed adaptive road detection algorithm throughout the entire structured environment

	FPR	ACC
Based on LIDAR data only with Predefined PS	47.58%	88.83%
Based on LIDAR data only with Semi-adapted PS for SR	38.11%	93.05%
Based on LIDAR data only with Fully-adapted PS for SR	32.95%	91.86%
Based on LIDAR and visual data with Fully-adapted PS for SR	17.69%	88.23%

Table 3-9 ACC and FPR values calculated for the proposed adaptive road detection algorithm throughout the entire unstructured environment

	FPR	TPR	ACC
Based on LIDAR data only with Predefined PS	29.78%	82.55%	78.29%
Based on LIDAR data only with Fully-adapted PS for SR	23.22%	82.25%	80.36%
Based on LIDAR data only with Fully-adapted PS for UR	58.66%	96.67%	77.56%
Based on LIDAR and visual data with Fully-adapted PS for UR	31.40%	87.15%	80.73%

3.3.3. Adaptive Road Detection Using Online Supervised Learning Approach

Proposed road detection algorithm using online supervised learning approach was tested both in structured and unstructured environments, and its performance was evaluated based on visual results. In the experiments, algorithm parameters, namely number of models learned, n_l , and maximum number of learned models, n_{ml} , were chosen according to the previous studies [18] and [39]. In these studies, parameter values presented in Table 3-10 were shown to be effective.

Table 3-10 Parameter set used for the experiments in SR and UR environments

Parameter	# of Models Learned (n_l)	Max. # of Models Learned (n_{ml})
Value	3	10

Proposed adaptive road detection algorithm using online supervised learning approach was firstly tested in SR environment and visual outputs are given for different road conditions in Figure 3-14, Figure 3-15, Figure 3-16 and Figure 3-17. In Figure 3-14 and Figure 3-15, visual results obtained by the proposed algorithm for winding and straight road sections are shown. According to these results, it can be concluded that the proposed road detection algorithm can successfully find road regions and general road boundary behavior independent from the road shape. However, as it can be seen in Figure 3-14-d, road detection rate decreases under non-uniform lighting condition. Moreover, due to the similarity between color distributions of the road area and the camera-carrying frame, frame-structure was also classified as road region in all of the tests. Although the proposed algorithm can successfully separate the start of the stairs next to the sidewalk from the road area in Figure 3-15-c, the algorithm cannot distinguish road region from sidewalk in Figure 3-15-d and -f.

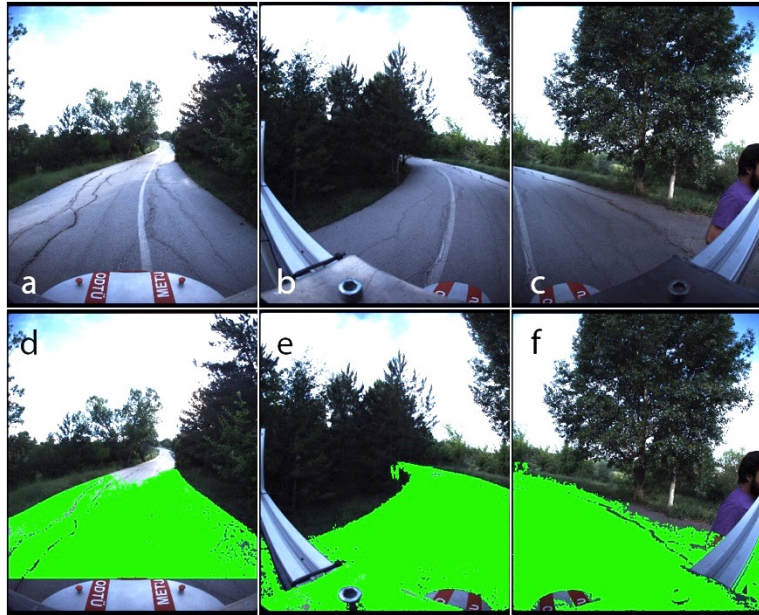


Figure 3-14 Raw images of the scene on the top row, and detected road regions on the bottom row for SR Section 3. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras.

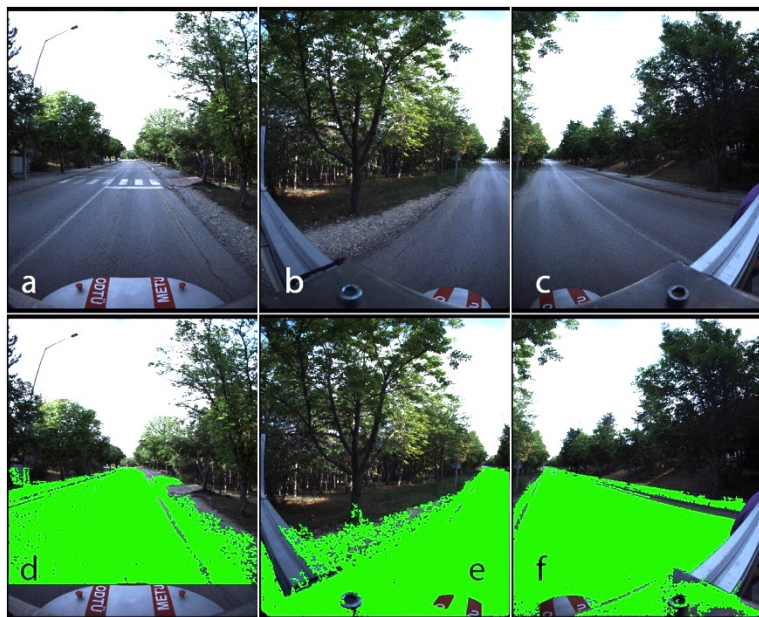


Figure 3-15 Raw images of the scene on the top row, and detected road regions on the bottom row for SR Section 2. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras.

In Figure 3-16 and Figure 3-17, two successive visual results obtained by the proposed road detection algorithm are presented. As it can be seen in Figure 3-16, the algorithm could only detect a small portion of the actual road area, whereas it could successfully classify the entire road region in the following time instance, as shown in Figure 3-17. The reason for low road detection rate in the first frame was due to the quick change in the lighting condition and lack of knowledge about the new visual properties of the road area. However, by collecting new training data from this area, visual road models were updated in the next frame, and road detection performance was improved in all directions, which can be seen in Figure 3-17.

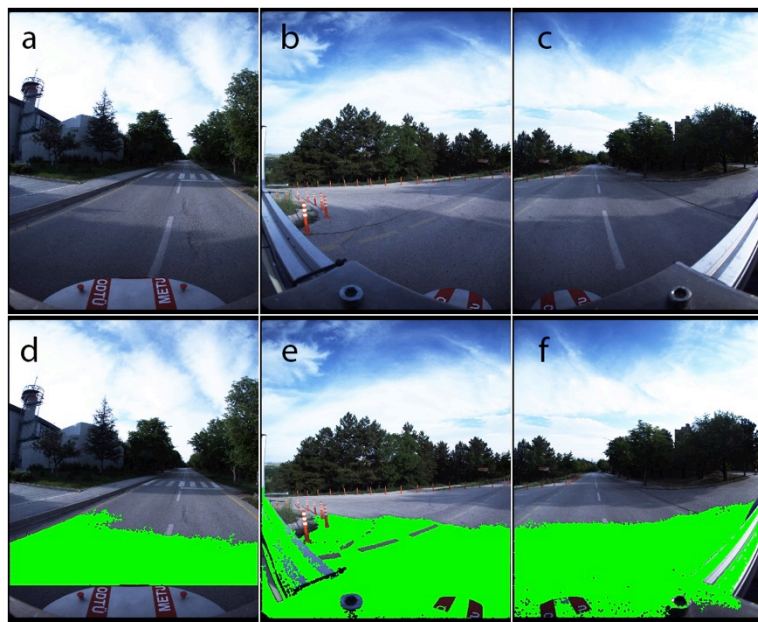


Figure 3-16 Raw images of the scene on the top row, and detected road regions on the bottom row for SR Section 4. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras.

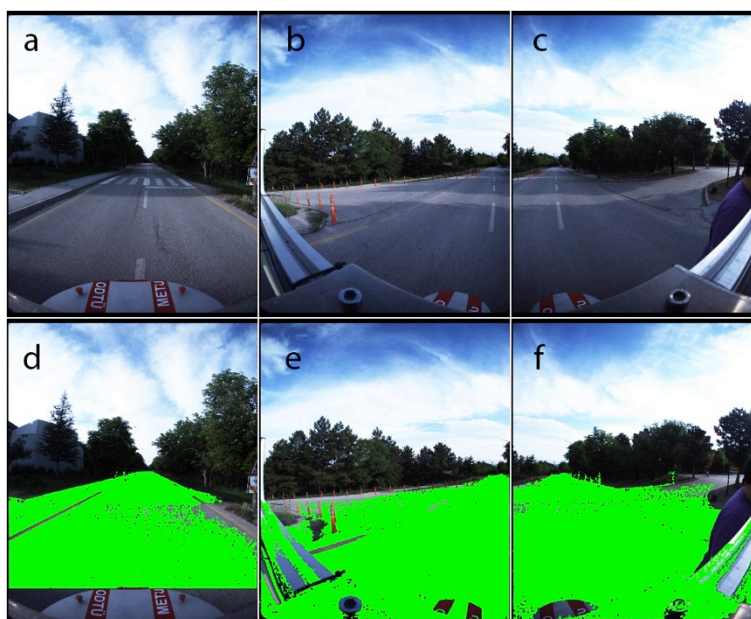


Figure 3-17 Raw images of the scene on the top row, and detected road regions on the bottom row for SR Section 5. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras.

After testing the proposed road detection algorithm in structured environment, its performance was examined one more time in unstructured environment. A sample output of the algorithm is given in Figure 3-18. As it can be seen in the figure, proposed algorithm can successfully find road regions and general road boundary behavior in unstructured environment. Unlike in the structured road case, algorithm can successfully distinguish the camera-carrying frame from the road regions in unstructured environment, due to different visual properties. However, the algorithm could not detect the road boundaries as accurate as in the structured road tests, because of the irregular and porous structure of the plants at the sides of the road.

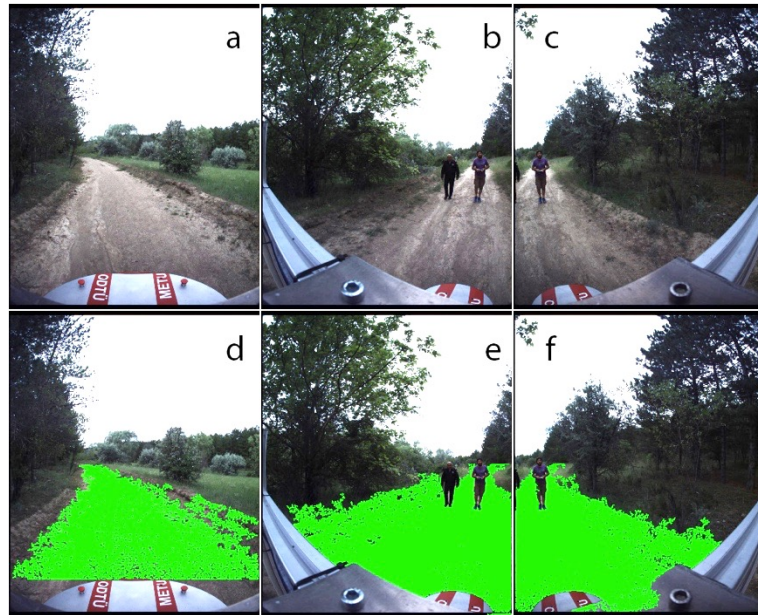


Figure 3-18 Raw images of the scene on the top row, and detected road regions on the bottom row for UR Section 2. Images on the left-most column were taken by the front facing camera, whereas other images on the remaining columns were taken by rear facing cameras.

Unlike the previous study presented in [18], proposed method in this thesis can operate in real-time without down-sampling the captured images, decreasing the resolution of color space values or using parallel processing. In the conducted experiments, average time required to complete each iteration, which includes generating training data, updating visual road models and detecting road regions, was found as 0.57 seconds. Although this result satisfies the working requirements, it can be further decreased by optimizing the program.

CHAPTER 4

CONCLUSION & FUTURE WORK

In this study, three different adaptive road detection algorithms were developed for a semi-autonomous mobile platform in order to be used in both structured and unstructured environments. The first algorithm detected road region by using the range measurements, which was collected by the LIDAR sensor mounted to the front surface of the vehicle. In this algorithm, discriminative learning approach was utilized to estimate the algorithm parameters, unlike the previous studies, which used constant parameter sets tuned according to the working environment, specifically. The second algorithm was based on LIDAR measurements and the visual data acquired from the camera. As a beginning, segmented point cloud was registered to the captured and preprocessed image. Then, the point cloud was filtered based on discontinuity in the hue color channel values in order to detect road regions. The third and last road detection algorithm used online supervised learning approach. In this method, firstly road regions were detected by using the secondly proposed road detection algorithm. As next, these range measurements were registered to the preprocessed image of the scene. Corresponding pixels' color values were used as training data to update visual road models by k-means clustering. Following that, road regions were found by using these visual road models based on Mahalanobis distance between pixel color values and the mean color values of the learned models.

In parallel to the implementation of the proposed road detection algorithms, the mobile robotic platform was renovated, which was firstly designed and constructed

in the scope of 111M580 numbered TÜBİTAK project. During the renovation stage, steering and breaking systems were repaired, electrical system was renewed, sensor positions were updated according to the need and missing libraries of the utilized sensors were written. After that, implemented road detection algorithms' individual performances were quantified according to ACC and FPR values for structured and unstructured environments.

Based on the results of the current study, the following conclusions can be drawn.

- Estimation of the parameter sets using discriminative learning approach simplified the tuning process and improved the performances of the algorithms in terms of false positive rate, accuracy and process time.
- Utilization of an adaptive approach for parameter estimation enabled application of the developed road detection algorithms to the changing environments easily.
- Integration of visual data to the geometrical features improved the performance of the road detection algorithm by increasing the detection rate of surface changes with similar geometrical properties.
- Using a dome-camera allowed detection of road regions both in front and back directions.

The present study can be further improved in the following ways:

- Revision of the erosion and dilation steps applied at the last stage of the thirdly proposed road detection algorithm may improve the road boundary detection accuracy.
- Revision of threshold value determination step in binarization stage of the thirdly proposed road detection algorithm may improve the road boundary detection accuracy.

- Integration of remaining LIDAR sensors mounted to the front and rear surfaces of the vehicle may decrease response time and improve the performance of proposed road detection algorithm using online supervised learning approach by increasing the number of training data.
- Integration of remaining LIDAR sensors mounted to the front and rear surfaces of the vehicle into the road detection algorithm for long-range road detection purpose may improve the road following and motion control algorithms' performance.
- Including the lateral and longitudinal slippages into the OBMM may increase the accuracy of localization algorithm and minimize the amount of saw-tooth shaped path sections.
- Development of an optic flow based localization technique, which uses images captured by the dome-camera, may increase the accuracy of vehicle's position.
- Replacement of the LIDAR sensors and integration of laser intensity values to the road detection algorithm may improve the road detection performance by increasing the surface distinguishing rate.

REFERENCES

- [1] P. Mickel, “1961: A peep into the automated future.” [Online]. Available: <http://www.capitalcentury.com/1961.html>. [Accessed: 06-Aug-2017].
- [2] MILVUS, “SEIT Automated Material Transport.” [Online]. Available: <http://www.milvusrobotics.com/products/seit#seit100>. [Accessed: 06-Aug-2017].
- [3] D. Matthew, “Tesla Autopilot wasn’t created so cars could drive themselves,” 2016. [Online]. Available: <http://www.businessinsider.com/tesla-autopilot-wasnt-created-so-cars-could-drive-themselves-2016-7>. [Accessed: 06-Aug-2017].
- [4] A. Turpen, “How self-parking car technology works: the first step to autonomous vehicles,” 2016. [Online]. Available: <http://newatlas.com/how-self-parking-works/46684/>. [Accessed: 06-Aug-2017].
- [5] A. B. Koku, E. İ. Konukseven, A. Hacinecipoglu, H. Ölmez, Y. M. Nazarabad, M. Ghaziani, M. K. Gönüllü, M. Lahroodi, and K. B. Özütemiz, “Uzaktan Kullanıcı Etkileşimli Yarı - Otonom İnsansız Kara Aracı Platformu,” Ankara, 2014.
- [6] E. Shang, X. An, L. Ye, M. Shi, H. Xue, and H. He, “Unstructured road detection based on hybrid features,” no. Iccia, pp. 926–929, 2012.
- [7] Qiang Chen and Hong Wang, “A Real-time Lane Detection Algorithm Based on a Hyperbola-Pair Model,” *2006 IEEE Intell. Veh. Symp.*, pp. 510–515, 2006.

- [8] R. Aufrere, R. Chapuis, and F. Chausse, “A model-driven approach for real-time road recognition,” *Mach. Vis. Appl.*, vol. 13, no. 2, pp. 95–107, 2001.
- [9] M. Ekinici, F. W. J. Gibbs, and B. T. Thomas, “Knowledge-based navigation for autonomous road vehicles,” *Turkish J. Electr. Eng. Comput. Sci.*, vol. 8, no. 1, pp. 1–29, 2000.
- [10] Z. Li, B. Dai, and H. He, “A Novel Fast Segmentation Method of Unstructured Roads,” *2006 IEEE International Conference on Vehicular Electronics and Safety*. pp. 53–56, 2006.
- [11] C. Rasmussen, “Texture-Based Vanishing Point Voting for Road Shape Estimation,” *Proceedings Br. Mach. Vis. Conf. 2004*, vol. 1, p. 7.1-7.10, 2004.
- [12] C. Rasmussen, “Grouping dominant orientations for ill-structured road following,” *Proc. 2004 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, 2004. CVPR 2004.*, vol. 1, p. I-470, 2004.
- [13] C. Rasmussen, “Combining laser range, color, and texture cues for autonomous road following,” *Proc. 2002 IEEE Int. Conf. Robot. Autom. (Cat. No.02CH37292)*, vol. 4, no. May, pp. 1–6, 2002.
- [14] R. Hadsell, P. Sermanet, J. Ben, A. N. Erkan, M. Scoffier, and K. Kavukcuoglu, “Learning Long-Range Vision for Autonomous Off-Road Driving,” *J. F. Robot.*, pp. 120–144, 2009.
- [15] R. Hadsell, P. Sermanet, A. N. Erkan, J. Ben, J. Han, B. Flepp, U. Muller, and Y. LeCun, “Online Learning for Offroad Robots: Using Spatial Label Propagation to Learn Long-range Traversability,” *Robot. Sci. Syst.*, vol. 11, p. 32, 2007.
- [16] R. Hadsell, A. N. Erkan, P. Sermanet, J. Ben, K. Kavukcuoglu, U. Muller, N. Technologies, and Y. Lecun, “a Multi-Range Vision Strategy for Autonomous Offroad,” *Robotics*, 2007.

- [17] A. N. Erkan, E. Raia, H. Pierre, J. Ben, U. Muller, and Y. Lecun, "Adaptive Long Range Vision in Unstructured Terrain," vol. 1, no. 1, pp. 2421–2426, 2007.
- [18] K. B. Özütemiz, "GPU-Accelerated Adaptive Unstructured Road Detection Using Close Range Stereo Vision," Middle East Technical University, 2013.
- [19] D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," *IEEE Expert*, vol. 11, no. 2. pp. 19–27, 1996.
- [20] M. Bertozzi and A. Broggi, "GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection," *IEEE Transactions on Image Processing*, vol. 7, no. 1. pp. 62–81, 1998.
- [21] R. Gregor, M. Lützel, M. Pellkofer, K.-H. Siedersberger, and E. D. Dickmanns, "A Vision System for Autonomous Ground Vehicles with a Wide Range of Maneuvering Capabilities," in *Computer Vision Systems: Second International Workshop, ICVS 2001 Vancouver, Canada, July 7--8, 2001 Proceedings*, B. Schiele and G. Sagerer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 1–20.
- [22] M. A. Sotelo, F. J. Rodriguez, and L. Magdalena, "VIRTUOUS: vision-based road transportation for unmanned operation on urban-like scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 2. pp. 69–83, 2004.
- [23] P. Jansen, W. Van Der Mark, J. C. Van Den Heuvel, and F. C. A. Groen, "Colour based off-road environment and terrain type classification," *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.* pp. 216–221, 2005.
- [24] L. B. Cremean and R. M. Murray, "Model-based estimation of off-highway road Geometry using single-axis LADAR and inertial sensing," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2006, pp. 1661–1666, 2006.

- [25] J. Han, D. Kim, M. Lee, and M. Sunwoo, "Road Boundary Detection and Tracking For Structured And Unstructured Roads Using a 2D Lidar Sensor," *Int. J. ...*, vol. 13, no. 2, pp. 293–300, 2012.
- [26] W. S. Wijesoma, K. R. S. Kodagoda, and A. P. Balasuriya, "Road-Boundary Detection and Tracking Using Ladar Sensing," *IEEE Trans. Robot. Autom.*, vol. 20, no. 3, pp. 456–464, 2004.
- [27] T. Kim and B. Song, "Detection and Tracking of Road Barrier Based on Radar and Vision Sensor Fusion," *J. Sensors*, vol. 2016, no. Imm, pp. 1–9, 2016.
- [28] K. Peterson, J. Ziglar, and P. E. Rybski, "Fast feature detection and stochastic parameter estimation of road shape using multiple LIDAR," *2008 IEEE/RSJ Int. Conf. Intell. Robot. Syst. IROS*, pp. 612–619, 2008.
- [29] Y. Morales, E. Takeuchi, A. Carballo, W. Tokunaga, H. Kuniyoshi, A. Aburadani, A. Hirose, Y. Nagasaka, Y. Suzuki, and T. Tsubouchi, "1Km autonomous robot navigation on outdoor pedestrian paths 'Running the Tsukuba Challenge 2007,'" *2008 IEEE/RSJ Int. Conf. Intell. Robot. Syst. IROS*, pp. 219–225, 2008.
- [30] Y. Shin, D. Kim, H. Lee, J. Park, and W. Chung, "Autonomous navigation of a surveillance robot in harsh outdoor road environments," *Adv. Mech. Eng.*, vol. 2013, 2013.
- [31] X. Yuan, C. X. Zhao, Y. F. Cai, H. Zhang, and D. B. Chen, "Road-surface abstraction using ladar sensing," *2008 10th Int. Conf. Control. Autom. Robot. Vision, ICARCV 2008*, no. December, pp. 1097–1102, 2008.
- [32] A. Hervieu and B. Soheilian, "Road side detection and reconstruction using LIDAR sensor," *2013 IEEE Intell. Veh. Symp.*, no. Iv, pp. 1247–1252, 2013.
- [33] J. Siegemund, U. Franke, and F. Wolfgang, "A Temporal Filter Approach for Detection and Reconstruction of Curbs and Road Surfaces based on Conditional Random Fields."

- [34] J. Stückler, H. Schulz, and S. Behnke, “In-lane localization in road networks using curbs detected in omnidirectional height images,” *VDI Berichte*, no. 2012, pp. 151–154, 2008.
- [35] Y. Kang, C. Roh, S.-B. Suh, and B. Song, “A Lidar-Based Decision-Making Method for Road Boundary Detection Using Multiple Kalman Filters,” *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4360–4368, 2012.
- [36] J. Liu, H. Liang, and Z. Wang, “A framework for detecting road curb on-line under various road conditions,” *2014 IEEE Int. Conf. Robot. Biomimetics, IEEE ROBIO 2014*, pp. 297–302, 2014.
- [37] J. Han, D. Kim, M. Lee, and M. Sunwoo, “Road Boundary Detection and Tracking For Structured and Unstructured Roads Using a 2D LIDAR Sensor,” *Int. J. Autom. Technol.*, vol. 15, no. 4, pp. 611–623, 2014.
- [38] G. A. Borges and M. J. Aldon, “Line extraction in 2D range images for mobile robotics,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 40, no. 3, pp. 267–297, 2004.
- [39] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski, “Self-supervised Monocular Road Detection in Desert Terrain,” *Proc Robot. Sci. Syst. RSS*, 2006.
- [40] S. Thrun, M. Montemerlo, and A. Aron, “Probabilistic Terrain Analysis For High-Speed Desert Driving.,” *Proc. Robot. Sci. Syst. Conf.*, pp. 16–19, 2006.
- [41] Point Grey Coop., “Geometric Vision Using Ladybug Cameras, Technical Application Not TAN20122009,” 2016.
- [42] Mathworks, “What is camera calibration?,” 2017. [Online]. Available: <https://www.mathworks.com/help/vision/ug/camera-calibration.html>. [Accessed: 13-Aug-2017].

- [43] Z. Bingül and O. Karahan, “A Fuzzy Logic Controller tuned with PSO for 2 DOF robot trajectory control,” *Expert Syst. Appl.*, vol. 38, no. 1, pp. 1017–1031, 2011.
- [44] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, “Stanley: The robot that won the DARPA Grand Challenge,” *Springer Tracts Adv. Robot.*, vol. 36, pp. 1–43, 2007.

APPENDIX A

ROS NODE USED FOR THE DEFINITION OF SENSORS' POSITIONS AND ATTACHED COORDINATE FRAMES

```
#include <ros/ros.h>
#include <tf/transform_broadcaster.h>
#include <math.h>

#define PI (3.141592653589793)

int main(int argc, char** argv){
    ros::init(argc, argv, "robot_tf_publisher");
    ros::NodeHandle n;

    ros::Rate r(100);

    tf::TransformBroadcaster broadcaster;
    tf::StampedTransform tf_base_to_center_;
    tf::StampedTransform tf_center_to_enc_;
    tf::StampedTransform tf_center_to_gps_;
    tf::StampedTransform tf_center_to_imu_;
    tf::StampedTransform tf_center_to_front_;
    tf::StampedTransform tf_front_to_laserfront_;
    tf::StampedTransform tf_front_to_laserurg_;
    tf::StampedTransform tf_front_to_laserback_;

    // lower center of the vehicle is 31.5 cm higher than ground
    // projection of vehicle's center on the ground is defined as base_link
    // front lidar is 50 cm higher than ground

    // set up parent and child frames
    // set up base_link center relation
```

```

tf_base_to_center_.frame_id_ = std::string("base_link");
tf_base_to_center_.child_frame_id_ = std::string("center");
tf_base_to_center_.setOrigin(tf::Vector3(0.0, 0.0, 0.315));
tf_base_to_center_.setRotation(tf::Quaternion(0.0, 0.0, 0.0));

// set up center encoder&compass relation
tf_center_to_enc_.frame_id_ = std::string("center");
tf_center_to_enc_.child_frame_id_ = std::string("odom_enc");
tf_center_to_enc_.setOrigin(tf::Vector3(-0.60800, 0.0, 0.0));
tf_center_to_enc_.setRotation(tf::Quaternion(0.0, 0.0, 0.0));

// set up map odom relation
tf_center_to_gps_.frame_id_ = std::string("center");
tf_center_to_gps_.child_frame_id_ = std::string("odom_gps");
tf_center_to_gps_.setOrigin(tf::Vector3(0.0, 0.0, 1.0));
tf_center_to_gps_.setRotation(tf::Quaternion(0.0, 0.0, 0.0));

// set up center imu relation
tf_center_to_imu_.frame_id_ = std::string("center");
tf_center_to_imu_.child_frame_id_ = std::string("odom_imu");
tf_center_to_imu_.setOrigin(tf::Vector3(0.0, 0.0, 0.0));
tf_center_to_imu_.setRotation(tf::Quaternion(0.0, 0.0, 0.0));

// set up center front_head relation
tf_center_to_front_.frame_id_ = std::string("center");
tf_center_to_front_.child_frame_id_ = std::string("front_head");
tf_center_to_front_.setOrigin(tf::Vector3(0.7851076, 0.0, 0.2907035));
tf_center_to_front_.setRotation(tf::Quaternion(0, 0, 0));

// set up front_head laser_front relation
tf_front_to_laserfront_.frame_id_ = std::string("front_head");
tf_front_to_laserfront_.child_frame_id_ = std::string("laser_front");
tf_front_to_laserfront_.setOrigin(tf::Vector3(0.1031436, 0.0, 0.0319805));
tf_front_to_laserfront_.setRotation(tf::Quaternion(0.034907, 0, 0));

// set up front_head laser_urg relation
tf_front_to_laserurg_.frame_id_ = std::string("front_head");
tf_front_to_laserurg_.child_frame_id_ = std::string("laser_URG");
tf_front_to_laserurg_.setOrigin(tf::Vector3(-0.0024205, 0, -0.1743725));
tf_front_to_laserurg_.setRotation(tf::Quaternion(0.261799, 0, 0));

// set up front_head laser_back relation
tf_front_to_laserback_.frame_id_ = std::string("front_head");
tf_front_to_laserback_.child_frame_id_ = std::string("laser_back");

```

```

tf_front_to_laserback_.setOrigin(tf::Vector3(-1.736, 0.0, 0.005));
tf_front_to_laserback_.setRotation(tf::Quaternion(0, 0, PI));

    while(n.ok()){

        tf_base_to_center_.stamp_ = ros::Time::now();
        broadcaster.sendTransform(tf_base_to_center_);

        tf_center_to_enc_.stamp_ = ros::Time::now();
        broadcaster.sendTransform(tf_center_to_enc_);

        tf_center_to_gps_.stamp_ = ros::Time::now();
        broadcaster.sendTransform(tf_center_to_gps_);

        tf_center_to_imu_.stamp_ = ros::Time::now();
        broadcaster.sendTransform(tf_center_to_imu_);

        tf_center_to_front_.stamp_ = ros::Time::now();
        broadcaster.sendTransform(tf_center_to_front_);

        tf_front_to_laserfront_.stamp_ = ros::Time::now();
        broadcaster.sendTransform(tf_front_to_laserfront_);

        tf_front_to_laserung_.stamp_ = ros::Time::now();
        broadcaster.sendTransform(tf_front_to_laserung_);

        tf_front_to_laserback_.stamp_ = ros::Time::now();
        broadcaster.sendTransform(tf_front_to_laserback_);

        r.sleep();
    }
}

```