FAST FEATURE EXTRACTION FROM 3D POINT CLOUD

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

BY

SERKAN TARÇIN

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

FEBRUARY 2013

Approval of the thesis:

FAST FEATURE EXTRACTION FROM 3D POINT CLOUD

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

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Suha Oral
Head of Department, **Mechanical Engineering** _____

Asst. Prof. Dr. A. Buğra Koku
Supervisor, **Mechanical Engineering** _____

Assoc. Prof. Dr. E. İlhan Konukseven
Co-supervisor, **Mechanical Engineering** _____

Examining Committee Members:

Prof. Dr. Tuna Balkan
Mechanical Engineering Dept., METU _____

Asst. Prof. Dr. A. Buğra Koku
Mechanical Engineering Dept., METU _____

Assoc. Prof. Dr. E. İlhan Konukseven
Mechanical Engineering Dept., METU _____

Asst. Prof. Dr. Yiğit Yazıcıoğlu
Mechanical Engineering Dept., METU _____

Dr. Refik Toksöz
Industrial Design Dept., METU _____

Date: _____

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

Name, Last Name: SERKAN TARÇIN

Signature :

ABSTRACT

FAST FEATURE EXTRACTION FROM 3D POINT CLOUD

Tarçın, Serkan

M.S., Department of Mechanical Engineering

Supervisor : Asst. Prof. Dr. A. Buğra Koku

Co-Supervisor : Assoc. Prof. Dr. E. İlhan Konukseven

February 2013, 54 pages

To teleoperate an unmanned vehicle a rich set of information should be gathered from surroundings. These systems use sensors which sends high amounts of data and processing the data in CPUs can be time consuming. Similarly, the algorithms that use the data may work slow because of the amount of the data. The solution is, preprocessing the data taken from the sensors on the vehicle and transmitting only the necessary parts or the results of the preprocessing. In this thesis a 180 degree laser scanner at the front end of an unmanned ground vehicle (UGV) tilted up and down on a horizontal axis and point clouds constructed from the surroundings. Instead of transmitting this data directly to the path planning or obstacle avoidance algorithms, a preprocessing stage has been run. In this preprocess first, the points belonging to the ground plane have been detected and a simplified version of ground has been constructed then the obstacles have been detected. At last, a simplified ground plane as ground and simple primitive geometric shapes as obstacles have been sent to the path planning algorithms instead of sending the whole point cloud.

Keywords: Unmanned Ground Vehicle, Point Cloud, Obstacle Avoidance, Bounding Box, Ground Plane

ÖZ

3 BOYUTLU NOKTA BULUTUNDAN HIZLI NİTELİK ÇIKARIMI

Tarçın, Serkan

Yüksek Lisans, Makina Mühendisliği Bölümü

Tez Yöneticisi : Yrd. Doç. Dr. A. Buğra Koku

Ortak Tez Yöneticisi : Doç. Dr. E. İlhan Konukseven

Eylül 2011, 54 sayfa

İnsansız kara araçlarının uzaktan kontrolü veya otonom hareket etmesi için çevreden çok çeşitli bilgi alınması gerekmektedir. Yüksek miktarda veri gönderen pek çok sensör kullanılan bu sistemlerde, bu verilerin bir bilgisayar işlemcisi tarafından kullanılması zor olabilmektedir. Benzer şekilde, bu veriyi kullanan algoritmalar için de, verinin işlenmesi çok zaman almakta ve algoritmalar yavaş çalışabilmektedir. Bunun çözümü, araç üzerindeki sensörlerden alınan bilginin, bir ön işlemden geçerek sadece gerekli kısmının iletilmesi veya bu ön işlem sonucunda elde edilen sonuçların iletilmesidir. Bu tez kapsamında bir insansız kara aracının (İKA) önünde bulunan 180 derecelik lazer mesafe sensörü, yatay eksende yukarı aşağı döndürülerek ortamdan nokta bulutu elde edilmiştir. Bu bilgi, cihazın gideceği yolu tasarlamasını sağlayan yol planlama veya engelden kaçma algoritmalarına direk gönderilmek yerine, bir ön işleme tabi tutulmuştur. Bu ön işlemde öncelikle nokta bulutunda zemine ait olan noktalar tespit edilmiş ve aracın ilerleyebileceği yüzey ortaya çıkarılmış, bunun ardından da ortamdaki engeller tespit edilmiştir. Bahsedilen yol planlama programlarına nokta bulutu değil, zemini temsil eden bir düzlem ve engelleri temsil eden basit geometrik şekiller gönderilmiştir.

Anahtar Kelimeler: İnsansız Kara Aracı, Nokta Bulutu, Engelden Kaçınma, Sınırlayan Kutu, Zemin Düzlemi

to my family,

ACKNOWLEDGMENTS

First of all, I would like to express my sincere appreciation to my supervisors Assoc. Prof. Dr. E. İlhan Konukseven and Asst. Prof. Dr. A. Buğra Koku for their invaluable guidance, advice, and criticism and especially their extreme support that made this study possible.

I would like to express my thanks to my dear friends and colleagues Buğra Özütemiz, Ozan Erol and Kasım Gönüllü for their friendship and technical support throughout the thesis period. I would also like to thank to my research group colleagues Akif Hacınecipoğlu and Matin Ghaziani. I also want to thank Ayşegül Çilli for her understanding and love which made life easier for me. And especially during the last weeks of this thesis Emre Akgül helped me a lot and I should also thank him for his invaluable support.

Also, I would like to thank my lovely family for their understanding and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AABB	Axis Aligned Bounding Box
AABR	Axis Aligned Bounding Rectangle
LIDAR	Light Detection And Ranging
OBB	Oriented Bounding Box
OBR	Oriented Bounding Rectangle
UGV	Unmanned Ground Vehicle

CHAPTER 1

INTRODUCTION

In this chapter, the motivation of this work and the general concept of visual systems on UGVs are given. Also, the necessity of simplifying the data on mobile robots is mentioned.

1.1 Motivation

Robotics has been growing in many fields with many applications such as medical, rescue, assistance, cleaning, production and security. As a result of this growth, unmanned ground vehicles (UGV) became an interesting research area. The main motivation behind those studies is to create vehicles which do not require any human help to function. They must understand their surroundings very well by using their sensors. As [29] mentions, the vehicle must estimate its states and finds its path in order to achieve its goal. Highly advanced techniques as, terrain labeling, road estimation, boundary estimations, path planning and real time control must be inserted into those machines.

Figure 1.1: Camera stabilization platform mounted on unmanned air vehicle.

Since the UGV research as highly future potential, our research topic as a group is focused in this manner. As a start an unmanned battery powered vehicle seen in Fig. 1.1 is designed and converted from a regular ATV. Moreover it is clear that one of the main functions of a vehicle in this size and power is its ability to avoid obstacles. A failure in obstacle avoidance algorithm may have serious consequences. So the speed and accuracy of those algorithms is crucial for an UGV. Advanced sensors are used for those algorithms for environmental data

and those sensors send large amount of data which is difficult to process. More importantly those processing takes great amount of time. Since a very important aspect of an algorithm is its processing time, the need of decreasing the amount of time which is used by obstacle avoidance algorithms rises.

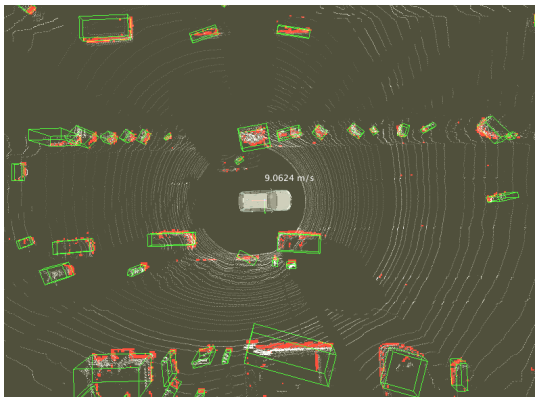

Figure 1.2: Obstacles in a 3D point cloud are represented with 3D bounding boxes. [16]

To decrease the computing time, downsizing the data is one of the many methods. For a mobile robot, unless it is trying to detect certain types of objects in the environment, it is enough for it to know the position of the object which is an obstacle for its path planning or mapping purposes. So, in this thesis, an approach has been developed to downsize the data without losing any critical information for localizing, mapping, path planning or similar algorithms of mobile robots.

1.2 Visual Systems on UGVs

It is clear that visual perception must be advanced in a capable UGV since visual navigation is crucial. Advanced visual navigation, enables the UGV to have a high level of autonomy and capabilities. Obstacle detection, path planning, navigation, state estimation, object recognition and many different algorithms and behaviors could be developed in this manner. Moreover UGV can interact with its immediate surrounding more efficiently. Sensors such as Velodyne, Sick, Hokuyo lidars, Kinect, Stereo Vision Cameras (BumbleBee) are used to obtain a succesfull visual data. Detailed explanation about these visual sensing systems are given in chapter 2. However, this study focus on a bounding box algorithm to help the navigation process, by using a SICK LIDAR which is rotated to collect a cloud of points.

Moreover one must note that SICK lidar is a 2D sensor, so only X and Y axis information could be obtained by using a simple setup which is not enough for our purpose. So as explained in chapter 4, a basic setup is manufactured for SICK sensor, which combines the 2D lidar information with the angle of the reference point of the sensor beam. As a result, a 3D point cloud is obtained.

1.3 Simplifying Data

Using sophisticated sensors is necessary to be able to move a vehicle in a terrain without a human operator but the a downside of these sophisticated sensors is the high amount of data which needs to be processed. For an UGV, processing speed is important since it is the way that it sees the world and navigates through it. If the processing of the data is slow, then the vehicle must wait for the end of the process before making a decision. Vehicle may even encounter fatal accidents because of the long waiting time for processing of the data.

1.4 Scope of the Thesis

In this thesis, point clouds taken from a rotating laser scanner have been processed to have simplified data. For this purpose, a mechanism has been made to rotate the sensor and some routines have been written in C# to gather and process the information. To be able to test the algorithms on some unstructured terrains, a cart have been constructed and a power supply, a computer and a laser scanner mounted on this cart. After improving the algorithms, the system will be deployed to an UGV in the future.

The thesis is organized as follows. A literature survey is presented about unmanned ground vehicles, sensing hardware on these UGVs and using bounding box approach in chapter 2. The theoretical approach and methods that are used are given in chapter 3. In chapter 4, the experimental setup is been mentioned which enabled the data gathering and tests and the results of these tests are presented in chapter 5.

CHAPTER 2

LITERATURE SURVEY

The main scope of this thesis is to reduce the amount of data to be processed which is taken from the environmental sensors. Reducing the data increases the efficiency of computation on the UGV which is designed in the control lab in Mechanical Engineering Department in METU. Moreover, the data is reduces without losing any important information for localization, obstacle detection and the similar algorithms so these algorithms can still operate effectively. As a survey, initially UGVs manufactured by different researchers are mentioned in detail. Then sensing tools for visual systems are mentioned for the purposed algorithms. Finally, the investigation results for obtaining an effective bounding box from the sensor readings are mentioned.

2.1 Unmanned Ground Vehicles

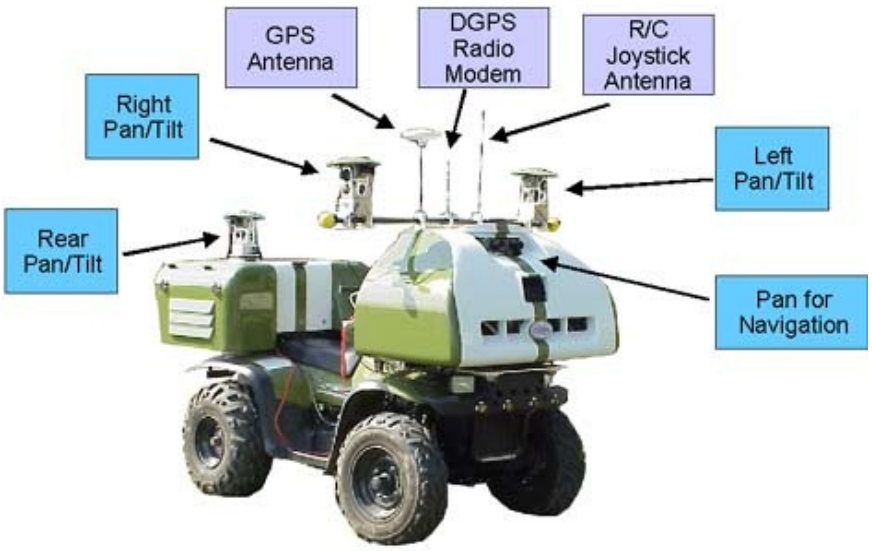

Figure 2.1: Sensors of CyberATV

Figure 2.2: CyberATV, modified vehicle (left) and original vehicle (right).

Researchers all around the world focus on the unmanned ground vehicle design and implementation in the last couple of decades. Both hardware and software studies are carried on. Since those vehicles must be designed for outdoor conditions and even for off-road challenges, they maintain a high level of mechanical complexity. In addition to those challenges, operating a vehicle fully autonomously under those chaotic environmental challenges require efficient and sophisticated algorithms. Also the implemented algorithms must be aided with qualified environmental feedbacks from the sensors. Advanced inertial measurement units [20], cameras, laser range sensor are commonly used on those kind of machines. And a capable processor which could be considered as a state of the art machine must be used for advanced algorithms.

Figure 2.3: ENSCO, original vehicle (left) and modified vehicle (right)

Since designing of an off-road vehicle is beyond the scope of the studies that are mentioned, many researchers use ATVs as based platforms and then convert that machine into unmanned ground vehicles with different capabilities by putting many different kind of sensors on it. A

very good example is the CyberScout Project [31] which is constructed in Carnegie Mellon University in United States. Main motivation behind that study was to develop a robotic vehicle for tactical distributed surveillance. So they developed a robotic ATV as seen in Fig. 2.1. The research topics of that specialized group was developing algorithms for multiagent collaboration, efficient perception which is similar to the scope of the present thesis, sensor fusion, distributed command and control.

CyberATV is built on a Polaris Sportsman 500 ATV. Which could be seen in Fig. 2.2. Initially throttle, steering and breaking functions are changed by using high powered R/C servomotors. And by using basic sensors like potentiometer, they implement a closed control steering control. A basic tachometer is used at the gearbox to take the localization data of the vehicle. Moreover a PC104 computer is used to actuate the system and process the visual data obtained from the stereo and mono vision systems for perception and localization. In order to support similar studies United Stated Army funded DARPA Grand Challenge [19] which is a competition where autonomous unmanned ground vehicles are all alone in the completion of their tasks.

Figure 2.4: ENSCO barebone

Although the challenge hosted many different kind of converted UGVs like a Volkswagen, this thesis focused on UGVs converted from ATV, like team ENSCO in Fig. 2.3. Team ENSCO was sponsored by a private engineering company ENSCO for DARPA Grand Challenge 2004 [9]. Team modified a Honda Rincon ATV by using servo motors for actuating and sensors for environmental data. GPS, DGPS, stereo cameras, magnetic compass and LIDAR systems have been used for perception of the environment. As a cost of using that much amount of sensor the computational need increases, so ENSCO has three computers on the vehicle: One for sensor measurements, one for actuation and one for the map processing and developing the path logic.

Also many teams participated in DARPA by modifying ATVs into UGVs like the team Spirit of Las Vegas[21] in 2003 in Fig. 2.5. The team also uses GPS, DGPS, gyroscope and camera systems for their purpose.

Figure 2.5: The Spirit of Las Vegas

Figure 2.6: Redcar Scout

In addition to studies conducted for DARPA, many other groups like United States Air Force are conducting studies. Protection Battlelab focused one of their studies on an ATV based UGV. As a difference from the other studies, Redcar Scout [6] is equipped with a thermal imager, image intensifier and low-light CCD cameras which can be seen in Fig. 2.6.

In addition to the studies around the globe, ASELSAN in Turkey also focused a division in the field of UGVs. As a result İZCİ [3] is created from modification of Polaris Sportsman-500 ATV. İZCİ could be used manually and autonomously by choice. Linear actuators and RC servos are used for actuation and basically SICK LIDAR systems, IMUs and cameras are used for perception.

Also our research group manufactured an ATV based UGV and various type of sensors are used for perception of the environment.

Figure 2.7: Aselsan's IZCI

2.2 Sensing Hardware on UGVs

For autonomous vehicles it is crucial to sense the environment correctly. A failure may cause great damage to vehicle itself and to its surrounding. There are many different kinds of sensors that could be integrated into such kind of academic platform. Sensors which are needed for motion planning, obstacle detection and mapping are mentioned in this section since the focus of this thesis is process this data and simplify it into bounding boxes.

Figure 2.8: Stereo Camera Bumblebe

Most common sensors used on UGVs are cameras and laser range scanners. Cameras are divided into two: mono and stereo cameras. Mono cameras are similar to the webcams and they are basically used for surveillance and driving the vehicle remotely. On some studies, they are also used for road extraction by image processing. On the other hand, stereo cameras seen in Fig. 2.8 are capable of providing depth information in addition to the 2D image data. So, for algorithms like obstacle detection, mapping the environment and depth enhanced vision operation, stereo cameras are used.

In addition to the camera data, getting the distances to the environmental obstacles is mostly helpful for algorithms. In range finding sensors, light detection ranging (LIDAR) sensors are

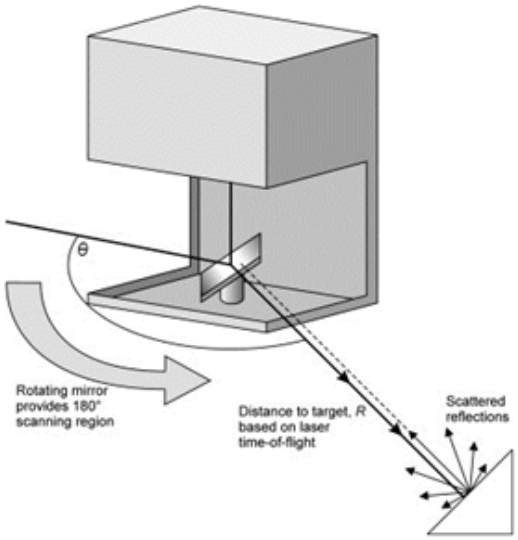

Figure 2.9: Working principle of 2-D LIDAR systems

most common. LIDAR is an optical measurement technology that can measure distance and the direction of an object by emitting a light beam from a laser source and measuring the time elapsed from the received laser beam. The working principle of the laser sensors used in UGVs are basically shown in Fig. 2.9. Most of these systems could obtain 2D planar measurements. Also more advanced models could create 3D point clouds of its environment.

In some systems, cameras and LIDAR systems are used in conjunction to obtain more reliable data. By combining these data, behaviors of the vehicles can be improved significantly. Also in the case of a sensor failure, vehicle could maintain operation. Most of the UGV studies use this method to improve performance and avoid collisions. For instance Alice UGV in Fig. 2.10 uses many different kinds of sensors. Alice uses two planar laser scanners on front of it with different pitch angles. One sensor is parallel to the ground to obtain positions of the obstacles from environment and other is faced to the ground with a certain angle to analyze the road structure from 3 meters earlier. In addition, three other LIDAR sensors are mounted at the top of the vehicle which has the capability to measure 20, 35, 50 meters ahead of the vehicle. And with a successful sensor fusion of cameras and LIDARS, effective 3D information could be obtained.

Moreover, a very advanced sensor could be used for 3D laser measurement itself called Velodyne seen in Fig. 2.12. However the price of that sensor makes it very difficult for researchers to use it. So, basically 2 or 3 SICK sensors are used for environmental data. In addition to using multiple 2D laser sensors, rotating a 2D sensor with a mechanism [1] as shown in Fig. 2.13 can also be used to get a 3D measurement. TEAM ENSCO, Jackbot, Overbot, İZCİ and many other vehicles use SICK Fig. 2.11 as the LIDAR, which is explained in the following sections. Table 2.1 shows the comparative numbers of sensors used in different kind of unmanned ground vehicles.

Figure 2.10: Alice project and sensor locations

Figure 2.11: SICK LMS 221 LIDAR.

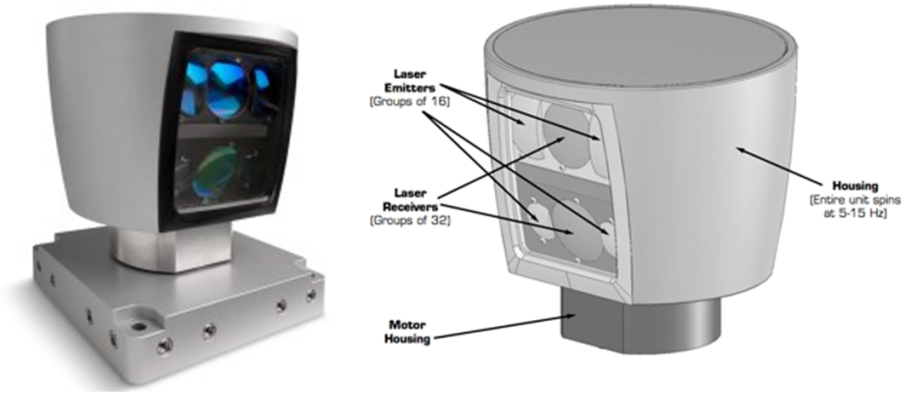

Figure 2.12: Velodyne HDL-64E 3-D LIDAR.

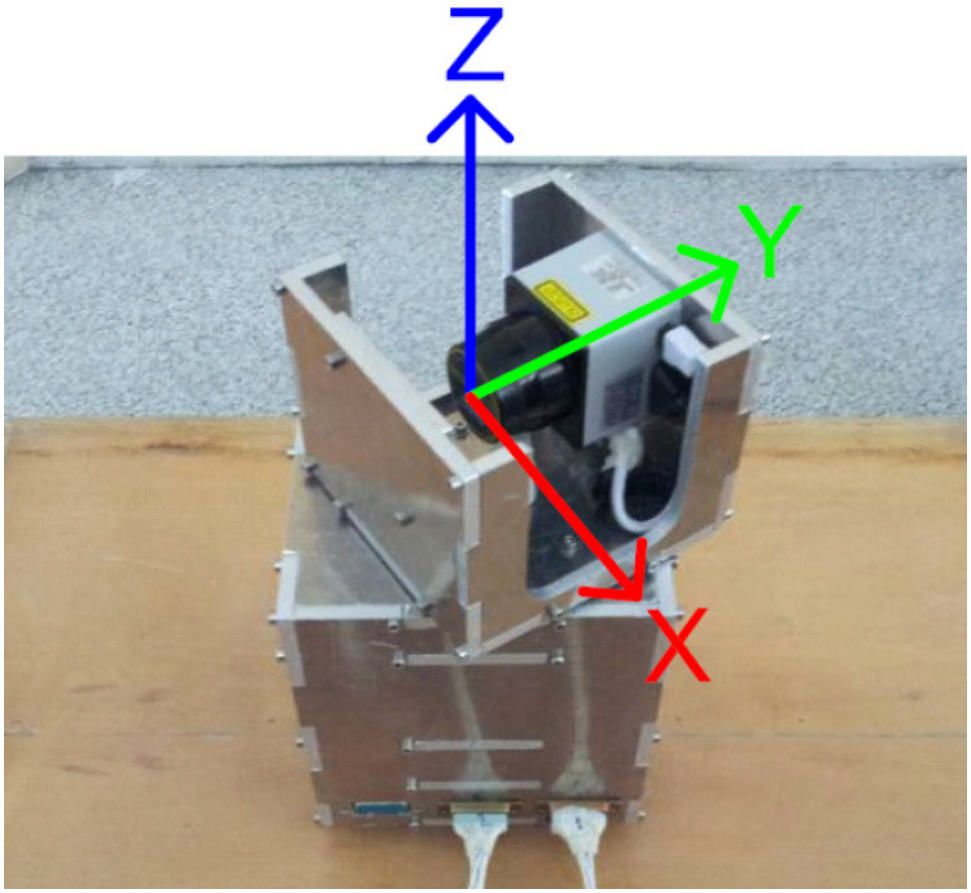

Figure 2.13: A Hokuyo LIDAR attached to a rotating mechanism. [1]

Table 2.1: A summary of numbers of sensors used for environmental sensing.

Vehicle	Number of LIDARs	Number of Cameras
Alice	5	3
Ensco	1	1
JackBot	4	None
Izci	2	2
CyberATV	None	5

2.3 Bounding Box Approach

There are a couple of methods proposed in order to compute a bounding box enclosing a given point cloud. The simplest of them is the computation of the axis-aligned bounding box for a point cloud. The 2D variants of these methods are given in [23] using packed R-tree, [26] using R+-tree, [4] using R*-tree; and others [15] use a minimum-volume AABB trimmed in a fixed number of directions to accelerate obstacle detection. The authors of [14] implement the OBB-tree (a tree of arbitrarily-oriented bounding boxes). In this implementation each box is aligned with the principle components of the distribution of a set of polygon vertices. A variant of this approach is proposed by [8] which aligns the boxes with only one principal component of the point distribution (corresponding to the smallest and the largest eigenvalue of the coordinate covariance of the points), the other directions are determined via the projection of orthogonal of the principal axis.

The author of [22] presents one of the best algorithms for the computation of the exact arbitrarily-oriented minimum-volume bounding box of a 3D point cloud. In this study the implementation is done in two steps. First, the convex polyhedron of the points is found then the minimal box is found by using each face of the polyhedron. It is obvious that if one face does not touch to the box, that face can be moved to inwards to reduce the volume.

In [5], the authors proposed a solution to min-max two-box problem. Given a set S of n points in d-dimensional space, they found two axis-parallel boxes b_1 and b_2 which together cover the set S and minimize the maximum of some measurements $\mu(b_1)$ and $\mu(b_2)$ (for example the perimeter of the box). The measurement to be selected is a monotonic function of the box, that is, $b_1 \subseteq b_2$ implies $\mu(b_1) \leq \mu(b_2)$.

In another study done by Barequet et al. [2], two nearly linear time complex efficient algorithms to calculate an approximate minimum-volume bounding box of a 3D point cloud set are proposed. In the first algorithm proposed, the authors first approximated the diameter of the point set. Then an approximated minimum bounding box can be computed by using the convex hull and the direction of the approximated diameter around the dataset as the direction of the box. Since the implementation of this algorithm is complicated and creates computational complexity, a more practical alternative of this algorithm is also proposed which is less efficient but easier to implement. In this second alternative, the first step is the computation of convex hull of the dataset and then making an exhaustive grid search to find the approximated bounding boxes.

The authors of [16] perform segmentation of objects from LIDAR data on an occupancy grid yielding connected components of grid cells not belonging to the ground surface. In their implementation, each cell contains a single value expressing the probability of occupancy of that cell by an obstacle. They also added that since there are enough data coming for each sampling, they create a new grid for each time new data obtained. That is, they do not accumulate data for a longer time. Next they apply connected components algorithm and segments the objects. For each segmented object, the coordinate covariance matrix and the eigenvectors of this matrix are calculated and by using these, a rectangular prism that encloses all the points for that object is formed. This type of object detection does not involve any assumptions about the object's shape but the area of enclosing boxes is not minimal.

Two different bounding box representations for laser range finder data is proposed by [18]. Their representation (Oriented Bounding Box (OBB)) is developed for the representation of

dynamic objects. It is reported that the proposed representation is suitable only for convex objects. However, concave objects can also be represented by multiple OBBs. The bounding box representation is calculated in two main steps. First a convex contour of the object is found and then a method called Rotating Calipers [30] is used to construct the OBB which is best aligned with the contour. The convex contour calculation is done in two different ways: by using Convex-hull or by using line segment extraction. The computation of the minimal-volume OBB is solved by formulating the problem as an unconstrained optimization problem on rotation and solved using combining genetic and Nelder-Mead algorithms in [7]. In [12], the authors propose a method of representation of the dynamic obstacles as a box in order to realize an obstacle map. Instead of boxes, studies done for other generic shapes, such as sphere [17], cone [24] or prism [11], [8].

Stuerzlinger proposed a method for the calculation of tight bounding volumes for primitives under arbitrary transformations and hierarchical scene descriptions. In his work [28], an optimized form of storing a convex hull is presented. This storing scheme allows efficient calculation of the bounding volume type used for later processing. Results of the study are shown for the ray tracing of natural phenomena.

In their work Rivera et al. [10] formulated a method to represent, in hierarchical structure, oriented boxes involving segments of object contours defined by closed cubical B-splines curves. Each of these oriented boxes is computed by fitting to these segments. While fitting, a multi-resolution fitting approach is used.

In a recent research done by Schnabel et al. [25] an automatic algorithm for the detection of basic shapes in unorganized point clouds is proposed. The algorithm first decomposes the point cloud data into a concise and hybrid structure of inherent shapes and a set of remaining points. Each detected shape is used as a proxy for the set of points corresponding to that shape. The basic shapes found in this algorithm are planes, spheres, cylinders, cones and tori, and these shapes are found by applying RANSAC on the point cloud.

By looking to the techniques implemented in the literature, it is easy to see that these techniques include computationally expensive techniques such as singular value decomposition and matrix inversion. These complex implementations are not very efficient in terms of computation time and they are not suitale for parallel computing hence, they do not have the benefit of parallel computing. It is acceptable if the applications do not have real-time concerns but for a realtime application, a faster method is necessary.

CHAPTER 3

THEORETICAL APPROACH

Simplifying the data and retrieving features in an efficient way requires many steps and in this chapter, these steps are mentioned in detail. After giving some information about point clouds, processing methods are given.

3.1 Point Cloud

One of the most widely used sensor kind on UGVs is laser scanners. These sensors have some different types according to their capability. Some can gather measurements on 180 degrees on a plane, some 240. Some laser sensors can measure 360 degrees on a plane and some sensors can gather measurements on multiple planes like Velodyne.

In this thesis, as the laser sensor SICK LMS291 has been used. Unlike Velodyne, this sensor can scan only 180 degrees on one plane. So all the data gathered from this sensor is placed on a plane and this gives a 2D point cloud. Fig. 3.1 shows a simple measurement taken from SICK LMS291.

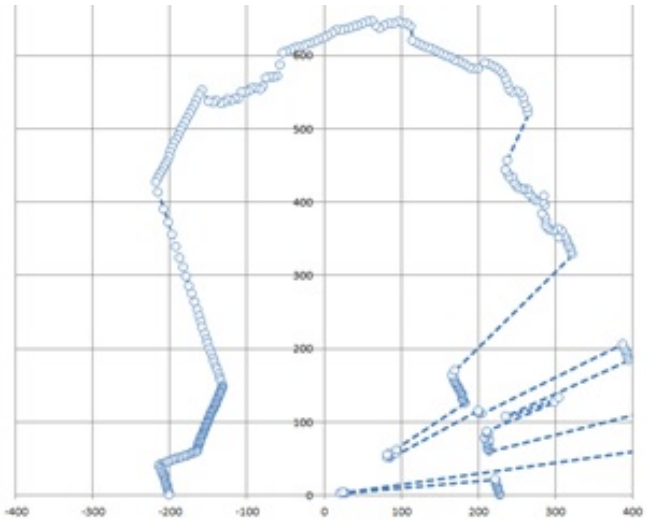

Figure 3.1: A Scan of a room taken with a SICK LMS291.

Even though most of the mapping algorithms use 2D laser scans, for obstacle detection this data is usually not enough because a 2D scanner cannot scan various height but it can scan only a plane with a predetermined height, so the scanner may miss some obstacles which has a height lower than the scan plane. One of the main reasons of those mapping algorithms to use a 2D data instead of 3D is its simplicity. Handling a 2D point cloud is easier than handling a 3D one and needs less computing time but in the case of obstacle detection, 2D data may become insufficient.

There are many methods and devices to gather 3D point clouds. One of the methods is using a 3D laser scanner like Velodyne which was mentioned before. Another method is using a depth camera or a stereo camera. These cameras provide images like a normal camera but each pixel represents a depth measurement.

Stereo cameras are very widely used in UGV researches. These sensors measure the distance by comparing two images taken from two side-by-side cameras like our eyes. This method works on almost every surface but for some conditions, they cannot give a measurement. For example textured surfaces, too bright or one colored surfaces or insufficient light conditions may cause holes in the 3D point cloud.

A different depth camera type is the kind that uses structured light. On these devices, beside an infrared camera, there is an infrared projector which projects a predefined map of points. Then the camera gets this image from a different point of view and a processor computes the distance according to the deformation on the projection. This method is appropriate for especially indoor applications because the light condition or textured surfaces does not affect the measurement but sunlight outshines the infrared projections so the measurements cannot be taken on the outside at daytime. An image captured with a Microsoft Kinect can be seen in Fig. 3.2.

Figure 3.2: Capture of a person by Microsoft Kinect sensor. Bright areas are closer to the sensor than dark areas.

Data of depth camera sensors is structured as 2D matrices. The indices of these matrices are X and Y coordinates of the captured image and the value of a cell of the matrix is the depth (mostly in millimeters or centimeters). The data gathered from the laser sensors are mostly in polar coordinates. For 2D laser sensors, the data is in the structure of a vector. The indices of the elements represent the angular order of the laser beam with respect to a coordinate frame which has the origin on the source of the laser beam. The value of a cell is the depth like the data of depth cameras.

In this thesis a 2D laser sensor, SICK LMS291, has been used as a 3D laser sensor with some mechanical adaptation. The laser sensor has been mounted on a mechanical apparatus which enables the sensor to rotate on an axis which is parallel to the ground plane and front plane. This mechanical apparatus, which can be seen in Fig. 3.3, has been made by a former lab member Emre Sezginalp [27]. By mounting laser sensor on this, 3D measurement can be taken with a 2D laser sensor. To be able to construct a 3D point cloud with the data taken from the 2D sensor, the pitch angle of the sensor must be known at all times. To get the pitch angle, first a low cost IMU has been tested. The selected IMU is CHR-UM6 Orientation Sensor which can be seen in Fig. 3.4. It has a 3 dimensional gyroscope, a 3 dimensional accelerometers and magnetic sensors to measure orientation at 500 Hz. After some tests, it appeared that the repeatability of the IMU was not good enough to construct a 3D point cloud because the minor errors on the pitch angle were causing some major positioning errors for long distance measurements. In place of the IMU, a potentiometer was integrated to the system as it is mentioned in chapter 4. Since potentiometers can give absolute angle measurements, repeatability is not a problem provided that the mounting has no backlash problems.

Figure 3.3: A simple apparatus to rotate Sick LMS291 around its X axis.

Figure 3.4: CHR-UM6 Orientation Sensor.

3.2 Ground Plane Detection

Obstacles are some objects or physical characteristics of the terrain which impedes the mobility of the UGV. Obstacles may be in different shapes and sizes. There may be man-made structures, walls, posts, tree trunks, bushes, rocks or even some big lumps on the terrain. To detect these obstacles there are many methods and some of these were mentioned in the literature survey. Since the focus of this thesis is to decrease the computing time, a computationally simple method has been used. To extract the obstacles, the ground is removed from the point cloud. There are several methods in literature to find the ground plane. Some of them are suitable for planar surfaces, some of them can detect curvy terrains. The UGV in our lab is planned to move on unstructured roads, so the terrain does not supposed to be planar but using an algorithm which can find curvy ground surfaces is costly in computational time. Since the aim is to find the obstacles and not the ground plane, a very simple algorithm can be used as a start then some post-processing can be done.

RANSAC (RANdom SAmple Consensus) has been used to find the ground and detect the obstacles. RANSAC is an iterative method to find a specific mathematical model in a data set. Outliers in the dataset don't affect the model. Even though the terrain is not planar, RANSAC has been set to find the largest plane in the 3D point cloud. In RANSAC algorithm that is used, first 3 randomly selected sample points are used to form a plane and the distance from every point to this plane is found. If a point's distance to plane is below a threshold, it is labeled as " inlier " . If the distance is above the threshold, it is labeled as " outlier". Then the loop starts again by selecting another 3 random points. Loop ends after some predefined number of repeats and the plane which has the maximum number of inliers is selected as the ground plane representation. The inliers of this plane are defined as the points belong to the ground. The pseudo code can be seen in Fig. 3.5.

A modification is made to the 3D point cloud before the RANSAC process starts which should be mentioned. The RANSAC algorithm counts the inlier points which stand close to the plane and this method works fine on the point clouds which have approximately same density across the cloud. But in this thesis, the point cloud is structured on a spherical coordinate system

```
void Ransac(Point[] pointSet, int threshold, int maxSampling, out int[]
inliers, out Plane ransacPlane)
{
maxInliers=0;
for (i = 1; i <= maxSampling; i++){
      int random1 = RandomNumGenerator(0, pointSet.Length);
      int random2 = RandomNumGenerator(0, pointSet.Length);
      int random3 = RandomNumGenerator(0, pointSet.Length);
      new Point[] ransacPoints={pointSet[random1], pointSet[random2],
pointSet[random3]};
      new Plane hypothesisPlane = CalculatePlane(ransacPoints);
      int numOfInliers=0;
      for (j = 0; j < pointSet.Length; j++){
            float distance = CalculateDistance(pointSet[j],
hypothesisPlane);
            if (distance <= threshold)
                  numOfInliers++;
      }
      if (numOfInliers > maxInliers){
            maxInliers = numOfInliers;
            ransacPlane = hypothesisPlane;
      }
}
```

Figure 3.5: Code of the RANSAC algorithm used in this thesis.

and the density is much higher near the rotation axis of the sensor than the other parts of the point cloud. This problem can be seen in Fig. 3.6. The points in the red circles are the inliers of the plane which is selected as the ground plane by the RANSAC algorithm.

Around the axis, the point density is so high that a RANSAC plane which crosses that area can have high amount of inliers and this may misdirect the algorithm. To avoid having false ground planes because of this situation, those points are removed from the set which goes through the RANSAC algorithm. The selection parameters of these problematic points can be seen in Fig. 3.7. Since the laser sensor has a height from the ground because of the mounting position on the vehicle and it has a limited rotation towards the ground, the nearest point which can belong to the ground has a calculable distance from the vehicle. This means, any point which is nearer to the vehicle than this distance has a very low probability to belong to the ground. Therefore these points don't need to be included to the RANSAC process. This simple filtering has improved the results of the RANSAC algorithm and most of the points which belong to the ground can be detected.

After the filtering has been done, the algorithm is no more misdirected by the high density areas and as it can be seen in Fig. 3.6, the ground plane can be found even there are big gaps on the ground surface in the 3D point cloud.

The points which are found by the RANSAC algorithm mostly belong to the ground just in front of the vehicle and the further ground points may be missed because of the curvy characteristics of the terrain. Filtering of these unfound ground points is explained in Chapter 3.5 in detail. At this point, those points don't affect the algorithm.

After finding the ground points in front of the vehicle, these points simply removed from the

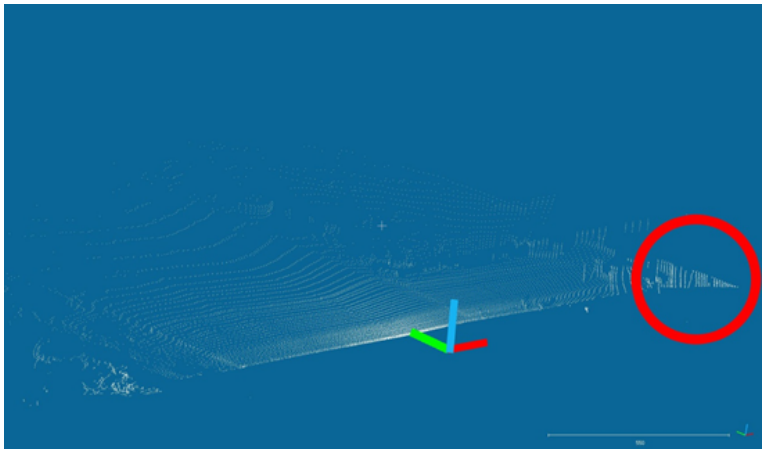

Figure 3.6: The area with high density is marked with red circle. These points cause RANSAC to work problematically.

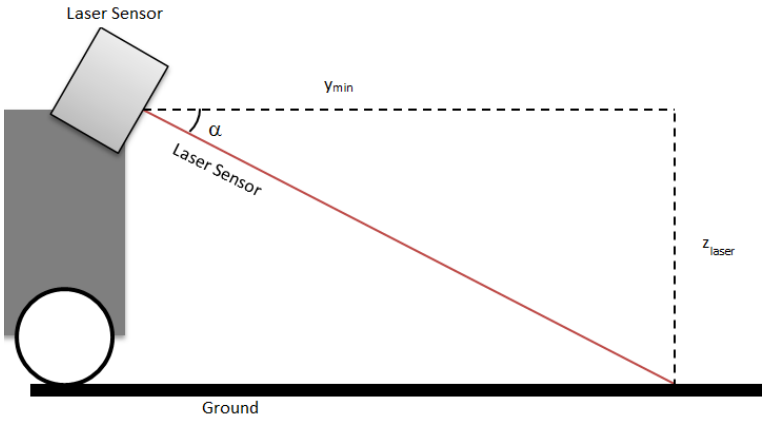

Figure 3.7: Position of the closest point to the vehicle which belongs to the ground relative to the vehicle.

3D point cloud. All left is the points higher (or lower) than the ground, noise and the ground points mentioned before which are missed by the RANSAC algorithm. These missed points can also be taken as noise then all left is obstacles and noise. In Fig. 3.8, a 3D point cloud before and after the ground removing process can be seen.

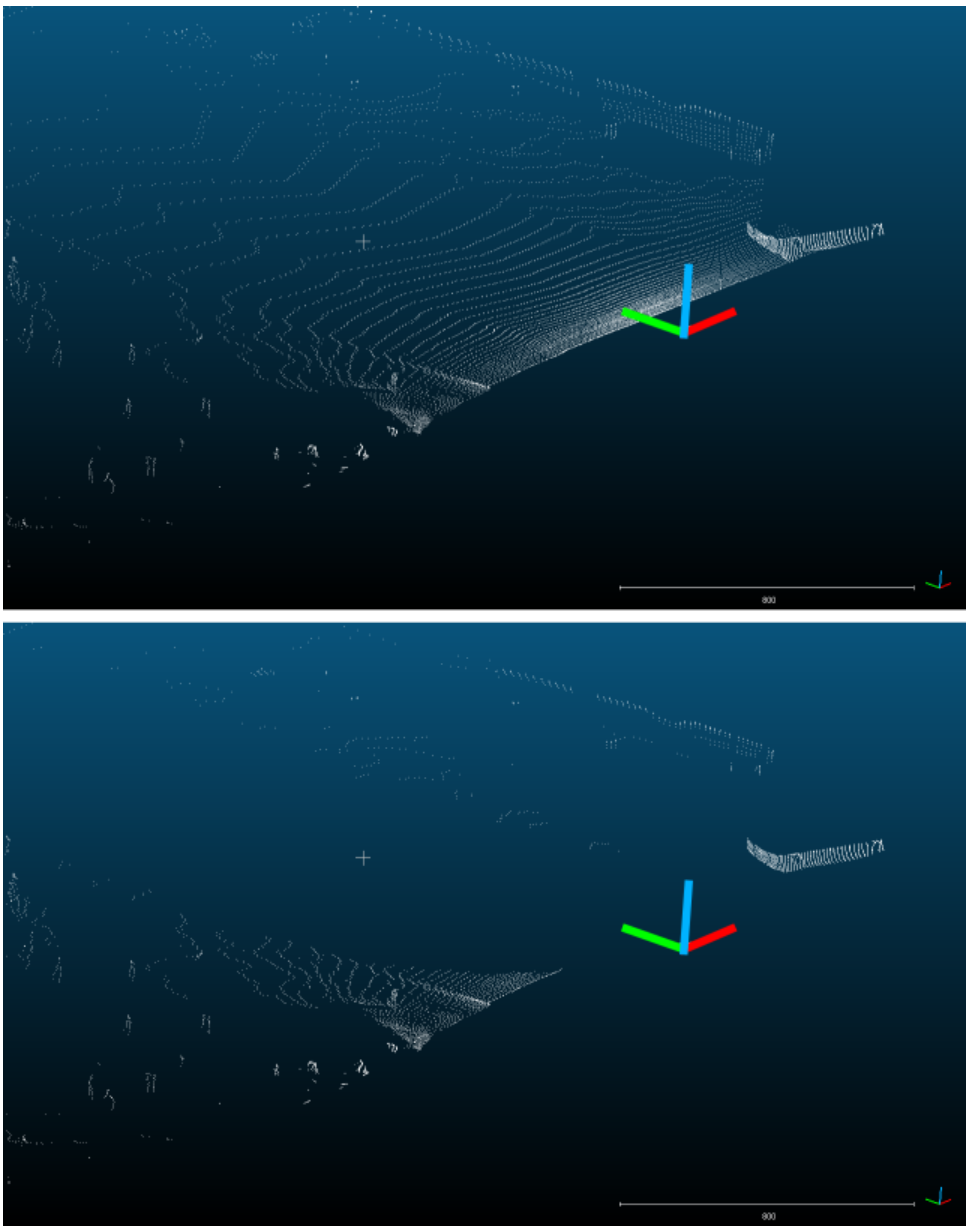

Figure 3.8: A 3D point cloud (top) and the same point cloud without the inlier points found by RANSAC algorithm (bottom).

3.3 3D to 2D Projection

Some unmanned vehicle projects requires a 3D map. For example for a flying vehicle like a quadrocopter, a 3D map of the surroundings can be very useful because it can move in 3 directions. For ground vehicles it can be handled differently. Even though the UGV travels on a 3D terrain with hills and slopes, it doesn't mean that it needs a 3D map. A UGV never travel upwards or downwards.

Since the aim of this work is simplifying the sensor data with as little computational time as possible and a UGV can be assumed that it travels in 2D, the dimension vertical to the ground can be omitted. Actually this process doesn't take time because all the algorithm needs to do is just not to consider the Z values of the points in the 3D point cloud. Then the 3D point cloud is flattened on to the X-Y plane and it can be considered as a planar 2D point cloud. In Fig. 3.9 a 3D point cloud and the same cloud which is flattened on to X-Y plane can be seen.

This process doesn't decreases the number of points but since the number of dimensions reduced to 2 from 3, the data to be processed is reduced to 2 by 3 because now the following algorithms will use only X and Y values instead of X, Y and Z. And this and the following processes are applied only the points which are labeled as "outliers" which means the points that are not belong to the ground.

3.4 Creating Density Map

Using a 2D point cloud has some other advantages. One of these advantages is being able to use image process algorithms. There are lots of resources in the literature about image processing algorithms and many libraries have been optimized through many years for different platforms. Being able to use these optimized libraries would be a big advantage since the aim is being fast. To be able to use the image processing algorithms there needs to be some modifications. First of all, the point cloud needs to be divided into grids. The grid size depends of the application and the size of the vehicle. It should support enough resolution to provide a map which can be used in path planning and SLAM algorithms but if the resolution is too high, image processing algorithms may need too much time. In this thesis, dividing the 2D point cloud into grids has been done by simple integer dividing. First, a grid size must be determined according to the application and the scene. Then a 2D matrix is created. The size of this matrix is the maximum range to be mapped over grid size in one dimension and two times of this quantity in the other dimension. The larger dimension is the X axis since the laser that is used in this thesis can give measurements from 0 to 180 degrees and the line passes through 0 and 180 degrees is assumed to be X axis. The 2D matrix is initialized with zeros in all of its cells. After the initialization of the 2D matrix, the X and Y values of all the points in the 2D point cloud (which doesn't have the points that belong to the ground anymore) are divided by the grid size and converted into integers. The result of the solution gives the cell indices of that point in the newly constructed 2D matrix. Every point in the 2D point cloud increases the value of its own cell in the 2D matrix by one. After the process has completed, the cells have the number of points in their area. The result is the density map of the 2D point ground which is divided into grids. This 2D matrix can be interpreted as a grayscale image. In Fig. 3.10, the point cloud and the density map of a scene can be seen.

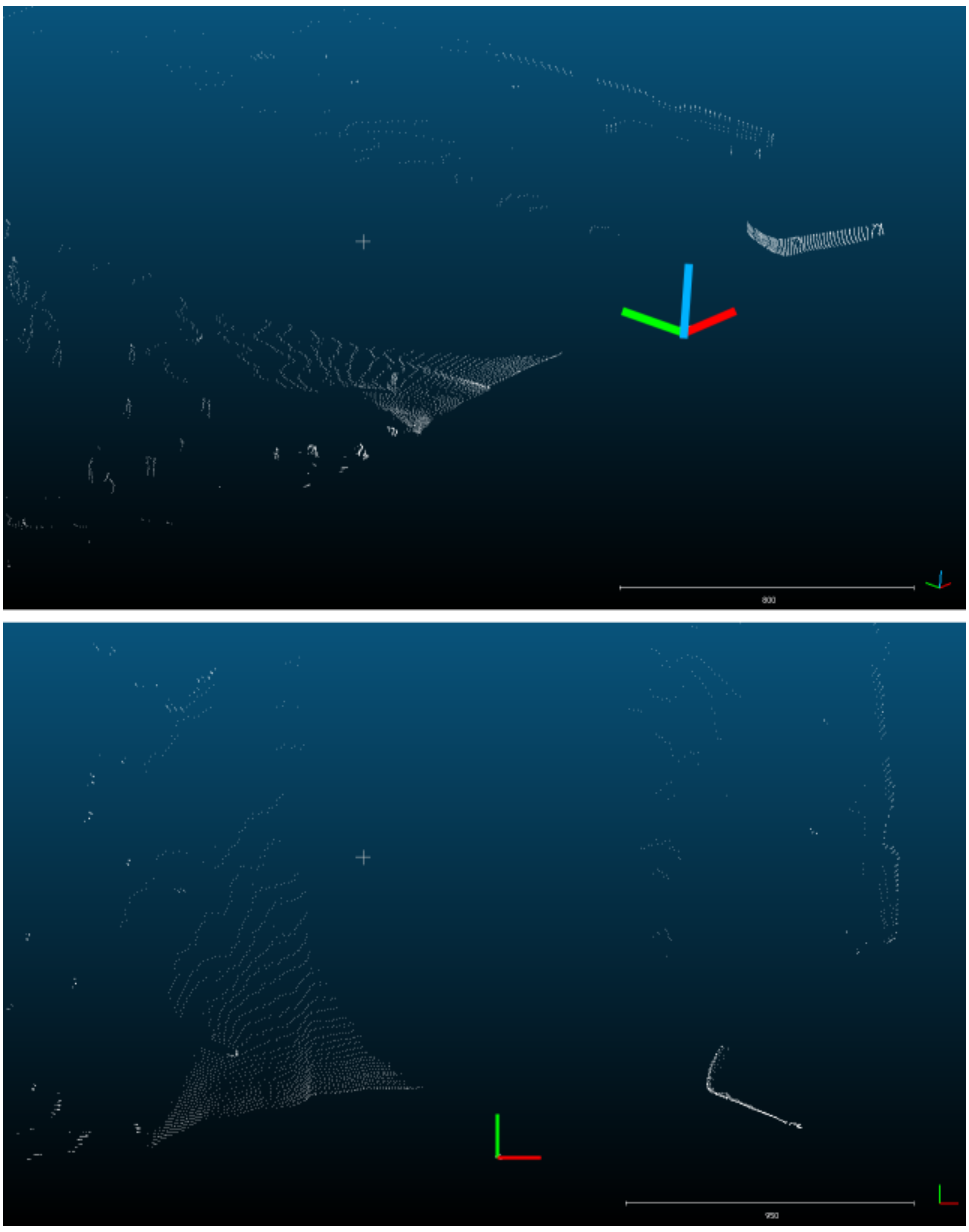

Figure 3.9: A 3D point cloud (top) and the same point cloud projected on the X-Y plane (bottom).

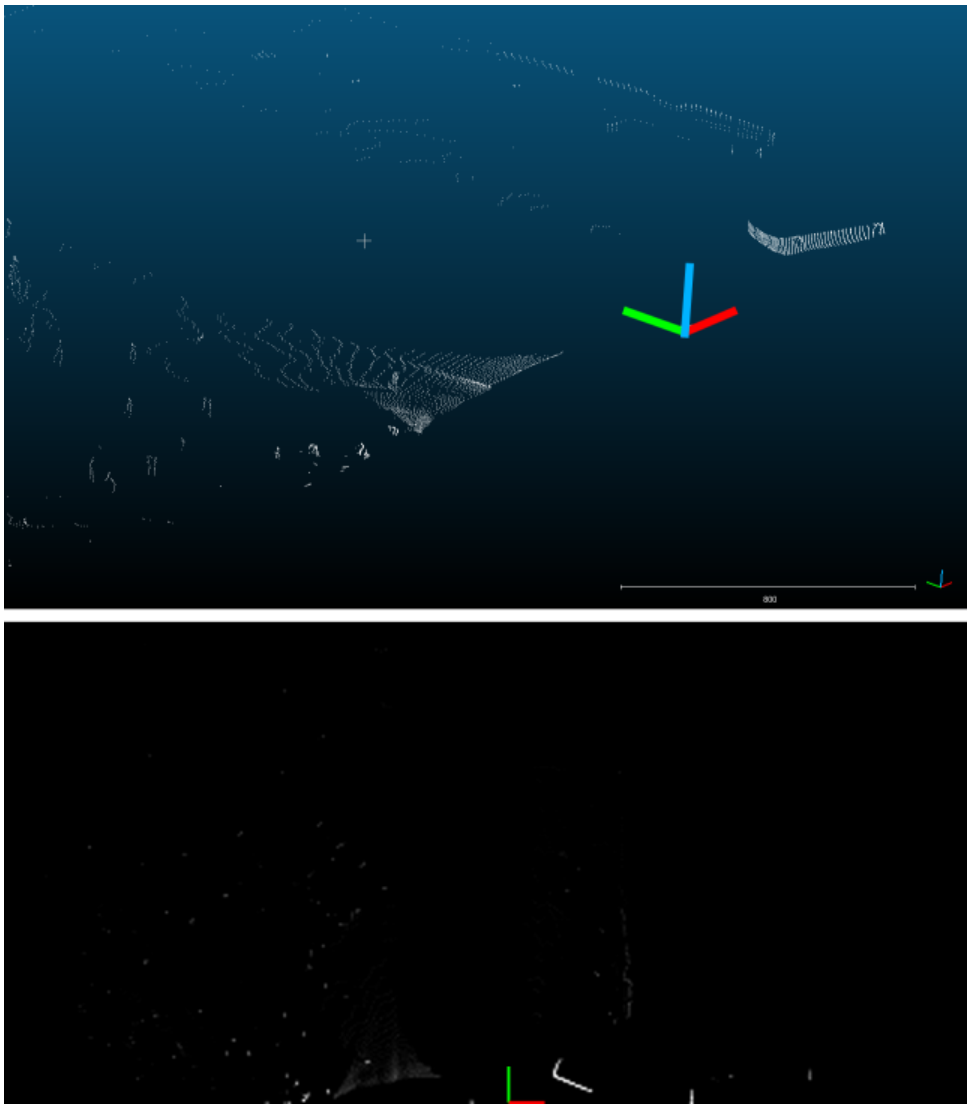

Figure 3.10: A 3D point cloud (top) and the density map of the same point cloud as a grayscale image (bottom).

3.5 Connected Component Labeling

Obstacles have some height which means from top view, their edges have higher density than horizontal surfaces like ground. The density map gives the number of points in the 3D space divided in vertical square prism volumes. This means, obstacles have higher numbers in these prisms so they have higher numbers in the density map than the noisy areas or some patches of undetected ground. Since the density map is a 2D matrix with integers as cell values, it can be interpreted as a grayscale image as mentioned in Chapter 3.4. This means the pixels of the image are the cells of the matrix and the white level of the pixel is the value of the cell. More points in a cell means brighter the pixel at that coordinate. This means vertically sprayed points mark their area in the image brighter than horizontally sprayed points like the ones belong to the ground. With this characteristic, to find the obstacles in the point cloud, simple image threshold algorithms can be used. A simple threshold can convert the grayscale image to a monochrome black and white image (without the shades of gray) which keeps only the obstacles and noisy or horizontal areas are removed. The threshold value should be set according to the structure of the point cloud and grid size.

The threshold algorithm gives a new image which shows only the vertically crowded areas. In the Fig. 3.11 a density map and the same image after threshold algorithm can be seen. Even it looks like there are many separate white islands of pixels on a black background, for the computer it is just a big one piece image. Those islands must be labeled and separated from the image. To achieve this, a connected component labeling algorithm can be used. Connected component labeling algorithm is an image processing algorithm which is used to detect connected regions on black and white (without gray) images. There are some different methods for connecting component labeling and image process libraries use the most efficient one for their structure. This is also a big advantage for the aim of this thesis because these libraries are being tested and optimized in many years.

Using a connected component labeling algorithm on the image, every separate region is labeled differently. A color coded image can be seen in Fig. 3.12. After this process, the image doesn't need to be considered one image but it can be considered as many little images and every image contains only one connected component in it. The obstacles are available as separate binary images.

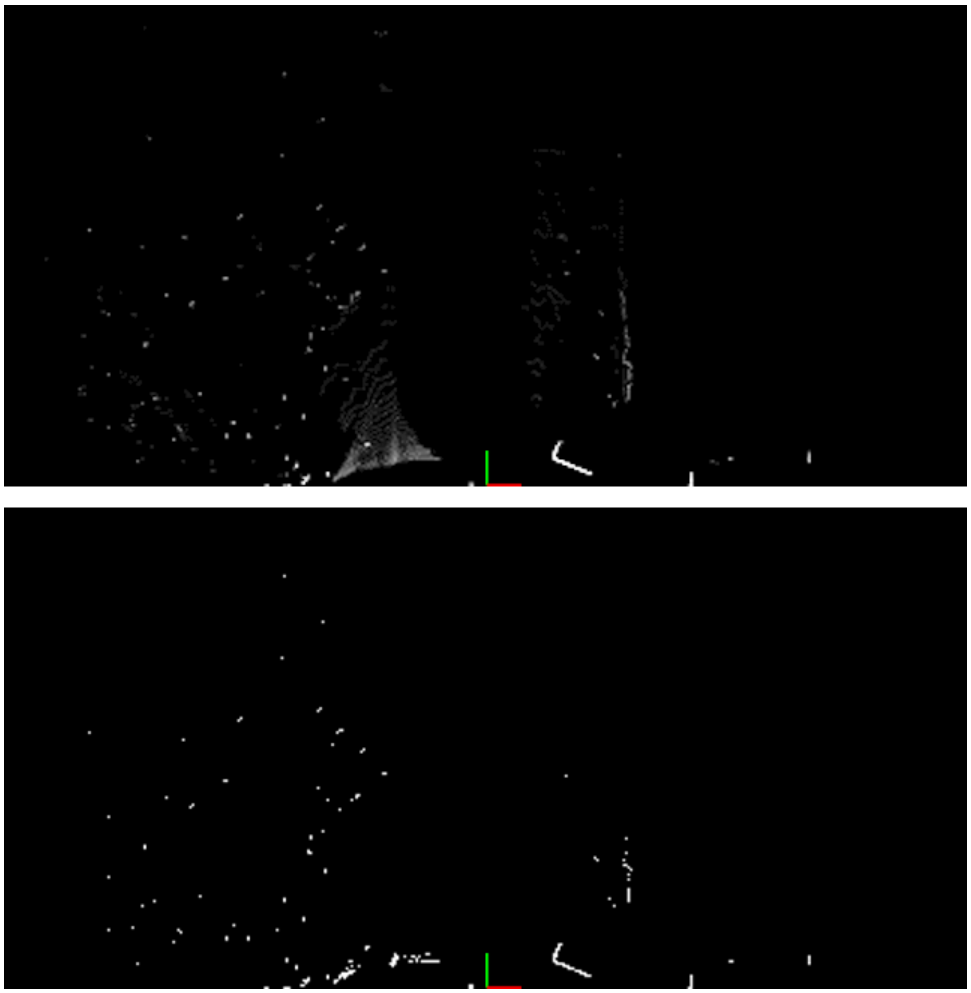

Figure 3.11: A density map (top) and the same map after the threshold process (bottom).

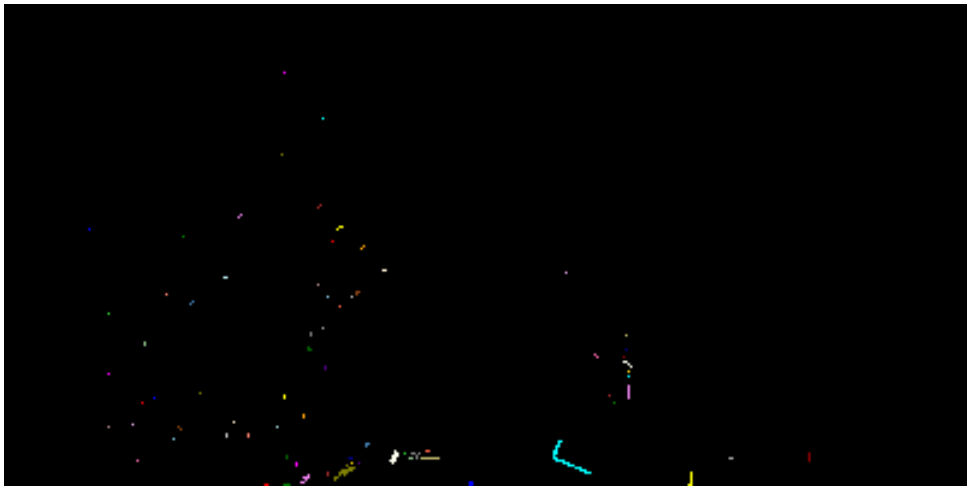

Figure 3.12: Color coded connected components of the threshold map.

3.6 Edge Detection

Bounding box of a point set is the same as the bounding box of its outer edge points which means the bounding boxes are relevant with only the outer pixels of the regions they surround which can be considered as the edge pixels of the region. This fact can be used to speed up computation of bounding boxes. This means that, finding the edges of the regions must be done before finding the bounding boxes.

A simple edge detection algorithm can achieve this very efficiently. Edge detection is a very fundamental tool in image processing. It can find the pixels where the image brightness or color changes sharply. It is a very widely used algorithm because contour of a shape gives basic yet for most applications enough information for image processing. Finding edges in an image is not a trivial problem because there may not be sharp color changes in the image because of light conditions or similar problems, so giving a threshold value is not a solution. There are many solutions given in the literature but still there may be missing edge segments after an edge detection algorithm applied on a non-trivial image.

In this thesis, the image which an edge detection algorithm is applied on is a binary image. So the edge detection algorithms work flawlessly.

A list of edges is constructed and for every edge a new list is constructed which keep the pixels of that edge. Every edge is related with its connected component. With this structure, there is a list of connected components, in this list there is an edge for every component. And these edges, which are also lists, keep the indices of the pixels they own. This structure creates a simple stage to calculate and construct the bounding boxes.

3.7 Bounding Box Representation Estimation

Bounding box term is commonly used for "Bounding Volume" or "Bounding Shape". A bounding volume is the smallest possible 3D geometry or 2D convex shape, depending on the application, and completely contains one object or a set of objects in it. Bounding volumes are mostly used to improve performance or efficiency of geometrical applications, because by using them, the algorithms can use simpler geometries instead of the original object. Bounding volumes do not necessarily be boxes. Bounding Sphere, Ellipsoid, Cylinder, Capsule, Polytope and Convex Hull can be given as a few examples. The first four geometrical objects are self-explanatory; they are the smallest possible object which contains whole of the object. A polytope is a polygon in 2D, a polyhedron in 3D and the number of sides should be determined according to the application. A convex hull is the smallest convex volume that surrounds the object. For example in 2D, convex hull of a convex shape is itself but convex hull of a five point star is a pentagon.

The bounding volumes have another subtitle depending on their orientation. They can be oriented or axis aligned. These two options mostly used in bounding boxes or rectangles. For an axis aligned bounding box, the edges of the box must be parallel with X, Y and Z axes no matter what the orientation of the object which is bounded. For an oriented bounding box, the orientation of the box should be determined by the algorithm to find the smallest possible volume.

In this thesis, the aim is converting the 3D point cloud data to a simpler structure for UGVs.

Using bounding shapes decreases data size but it may also create a misconfigured map if the bounding shape type is not selected correctly. To find the best bounding shape for this purpose, some types tested with the data. Since the aim is using a 2D map and reducing the number of data points, only 2D bounding shapes which have the least number of parameters have been taken into account: Axis Aligned Bounding Box, Circle and Oriented Bounding Box.

The fastest approach would be using "Axis Aligned Bounding Rectangle" as the bounding shape. To find the edges of the axis aligned bounding rectangle for a connected component, finding the leftmost, rightmost, topmost and bottommost pixels of that component. These pixels state the left, right, top and bottom edges of the rectangle respectively. The X value of the leftmost pixel can be named as "Xmin" and for the rightmost pixel, "Xmax". The Y value of the topmost pixel can be named as "Ymin" (since the Y value increases downward in images) and for the bottommost pixel, "Ymax". Then the corners of the bounding rectangle would be: $\{(Xmin, Ymin), (Xmax, Ymin), (Xmax, Ymax), (Xmin, Ymax)\}$

Finding axis aligned bounding rectangle is a very fast process but it works for only for the obstacles which lie parallel to X or Y axes correctly. For an obstacle which lies diagonally, the resulting bounding rectangle has a big portion of empty area. In Fig. 3.13, a shape which lies diagonally can be seen in white and the axis aligned bounding rectangle of it can be seen in red. Note that the empty area is multiple times larger than the shape itself. If the environment which the scan is gathered from is like a corridor, axis aligned bounding rectangles of the two walls can block the whole corridor. This state can be seen in Fig. 3.14. In the real application, the UGV can move near walls or through corridors and it may be in any orientation relative to the global coordinate system. This make axis aligned bounding rectangle unusable for the application.

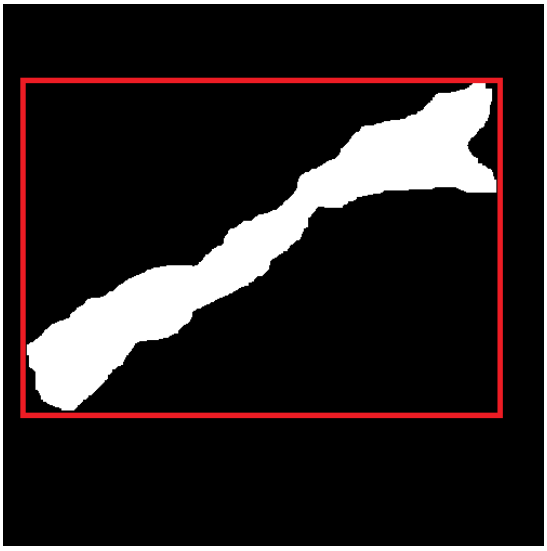

Figure 3.13: A shape which lies diagonally in white and the axis aligned bounding rectangle of it in red.

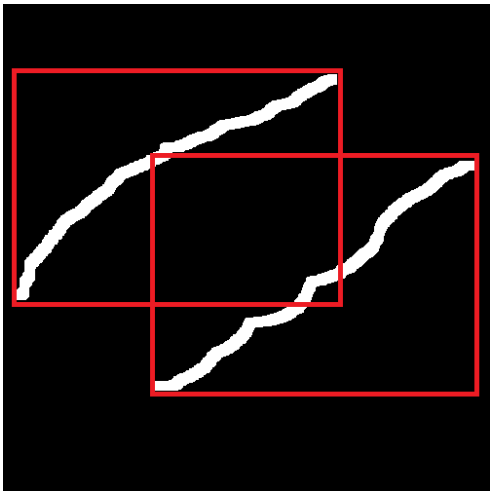

Figure 3.14: Two walls of a corridor in white and the axis aligned bounding rectangles of the walls in red.

Another bounding shape which can be computed in a trivial way is a circle. The "bounding circle" or "smallest possible circle" problem is shown to be solved in linear time for a set of points in [32]. The algorithm proposed simply tries to include all points to a circle in random order by growing it. In this thesis, there is a big advantage: The algorithm doesn't need to test all of the points because since the points are in Cartesian coordinate system, all needed is to look for the points with the minimum X, maximum X, minimum Y and maximum Y values. Then the smallest possible circle algorithm can be run only on these four points.

This algorithm is very similar to the axis aligned bounding rectangle algorithm. It takes a little more time than axis aligned bounding rectangle algorithm but since a part of the algorithm is random, the additional time may vary.

Even though finding smallest possible circle is a fast algorithm, it has the same problem with the axis aligned bounding rectangle algorithm. Axis aligned bounding rectangle algorithm works fine on objects which lie on X or Y axes but creates large empty areas in the bounding rectangle for the objects which lie diagonally. Smallest possible circle algorithm has this problem for the objects which have high height to width proportion. This problem can be seen in Fig. 3.15. The empty area is larger than the object itself and for environments like corridors, it will cause the same problems as the axis aligned bounding rectangle.

Another commonly used bounding shape type is oriented bounding rectangle. Oriented bounding rectangle is an ordinary rectangle but unlike axis aligned bounding box, it doesn't need to be aligned to the main axes of the coordinate system. Its orientation can be set by the algorithm to find the smallest possible rectangle which surrounds the dataset. Because of this, it is also called minimum bounding box or rectangle. In Fig. 3.16 axis aligned bounding rectangle and oriented bounding rectangle of a shape can be seen.

It is known that finding oriented bounding rectangle can be done in linear time. In [13], it is proposed that an edge of a minimum area bounding rectangle must be collinear with an edge of the convex hull of the shape to be enclosed. A method named rotating calipers is proposed in [30] to find minimum area bounding rectangle in linear time.

Figure 3.15: A shape in white and the bounding circle of it in red.

Rotating calipers method tries to find all the antipodal pairs on a convex polygon by putting a blade of an imaginary caliper on every edge of the convex polygon until the caliper reaches a 180 degrees rotation and taking measurements. By this way it can find the orientation of the minimum width of the convex polygon then create an oriented bounding rectangle.

In this thesis, it is not necessary to find the perfect orientation of the oriented bounding rectangle. The aim is to simplify the point cloud data by putting simple geometrical shapes in places of obstacles. With this aim in mind, the rotating calipers method can be simplified. The simplification can be done by testing the rotating calipers on a number of predetermined orientations instead of the edges of the convex polygon. As an example, the blades of the caliper can be rotated with 5 degree intervals and check for the measurement by noting the angle with it. Rotating the caliper only 90 degrees would be sufficient since the angles above 90 degrees would be redundant. In these measurements, the one with the minimum value can be assumed as the width and the angle of the caliper for that measurement can be said to be the angle of the minimum bounding rectangle. Since the angle is known at this point, the oriented bounding rectangle can be calculated as an axis aligned bounding rectangle but the reference axis should be taken as the orientation given by the rotating calipers method.

All the three of these types of bounding shapes can be computed easily and they have at most four parameters for adding themselves to the final map (like the corners of an oriented bounding rectangle). The problem for the first two is although the calculation of these types are very simple, the problems they create for the cases like moving near a wall or through corridors are unacceptable. Because of this, the third type, oriented bounding rectangle is selected as the bounding shape to represent the obstacles in the final, simplified map.

3.8 Dilation Process

To increase the usability of this work in the mobile robotics field, another process is added to it. In the mobile robotics field, it is very common to use a traversability map which is

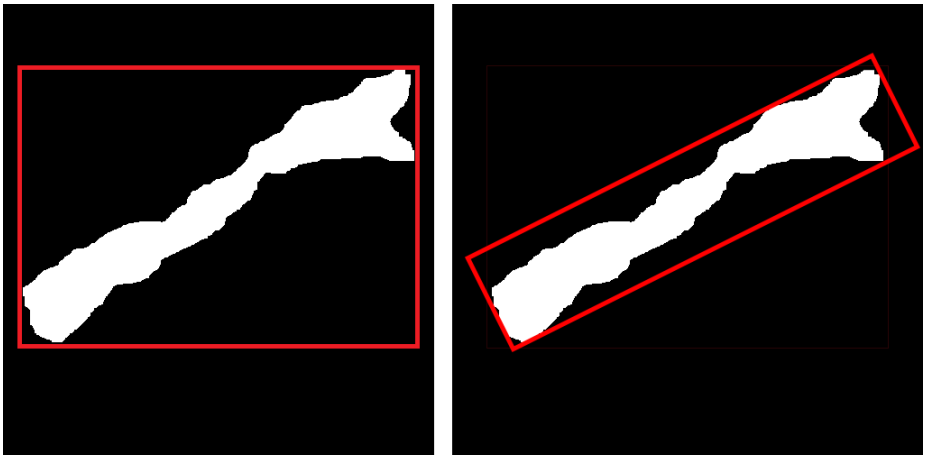

Figure 3.16: A shape in white and AABR (left) and OBR (right) of it in red.

mostly a binary image where the white pixels indicate the traversable areas on the map. This map is constructed from the obstacles. But using a map where the obstacles are marked directly is not a solution because there may be some obstacles which are close to each other and even it looks like that there is a gap between them, the mobile robot cannot fit in there. To cancel out those gaps on the map, dilation filter is used. Dilation filter assigns a value to the surrounding pixels of the white pixels on a binary image. This assignment is done by a structuring element which is a matrix with odd width and height. It mainly used on binary images to grow separate objects or join close objects.

The mobile robot which the lab team is working on is about 1.5 meters in length and about 80 cm in width. Even though the robot can't move sideways, to be on the safe side it is represented by a structuring element of a circle with 1.4 meters in diameter. As it is mentioned before, the pixels on all of the images through the processes in this thesis represent an area of 20cm by 20cm. So, the structuring element can be constructed as a 7 by 7 square matrix which can be seen Fig. 3.17.

0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0

Figure 3.17: Structuring element of the dilation process.

Using this structuring element on the threshold images which are mentioned in Chapter 3.5, a dilated image can be obtained. On this image the robot is represented with one pixel and all the black pixels are tagged as traversable areas. A threshold image and the dilated version of it can be seen in Fig. 3.18.

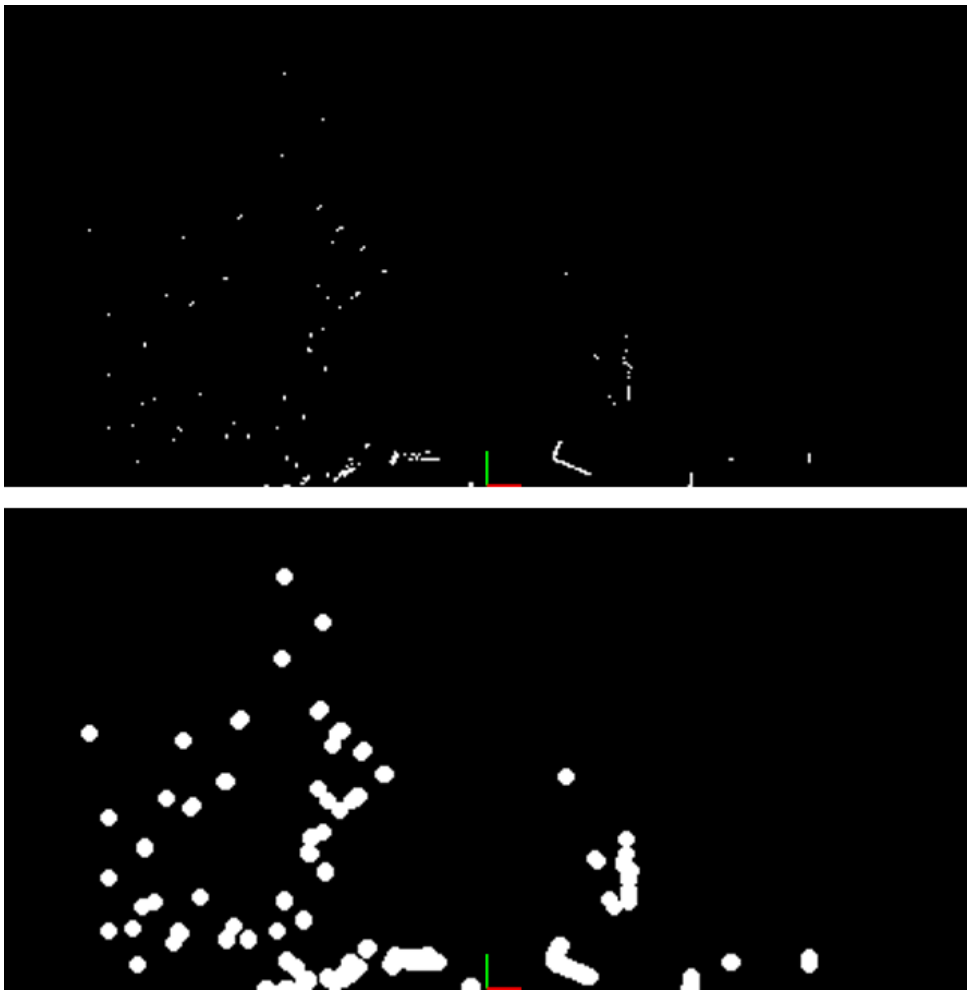

Figure 3.18: Threshold map (top) of a scene and the dilated version of it (bottom).

The number of connected components in the dilated versions of threshold images is less than the threshold images themselves. This is used as an advantage in the aim of downsizing the amount of data in the sensor scans. Besides, the walls in the environment can't be captured as a whole with a laser scanner because of shadowing problem. If there is a tree or a pole in front of the wall, it blocks the laser beam and casts a shadow on the wall. This causes the wall to be captured with gaps on it. Dilation process closes these gaps and gives better results. In parallel to bounding rectangles, dilated images are also tested for some algorithms in Chapter 5.

3.9 Application with C#

The methods given through Chapter 3 have been applied using C#. The interface of the test application can be seen in Fig. 3.19. Before the procedure starts, the application connects to the laser sensor to get the range data and a microcontroller which is used as an interface to get the orientation data from a potentiometer.

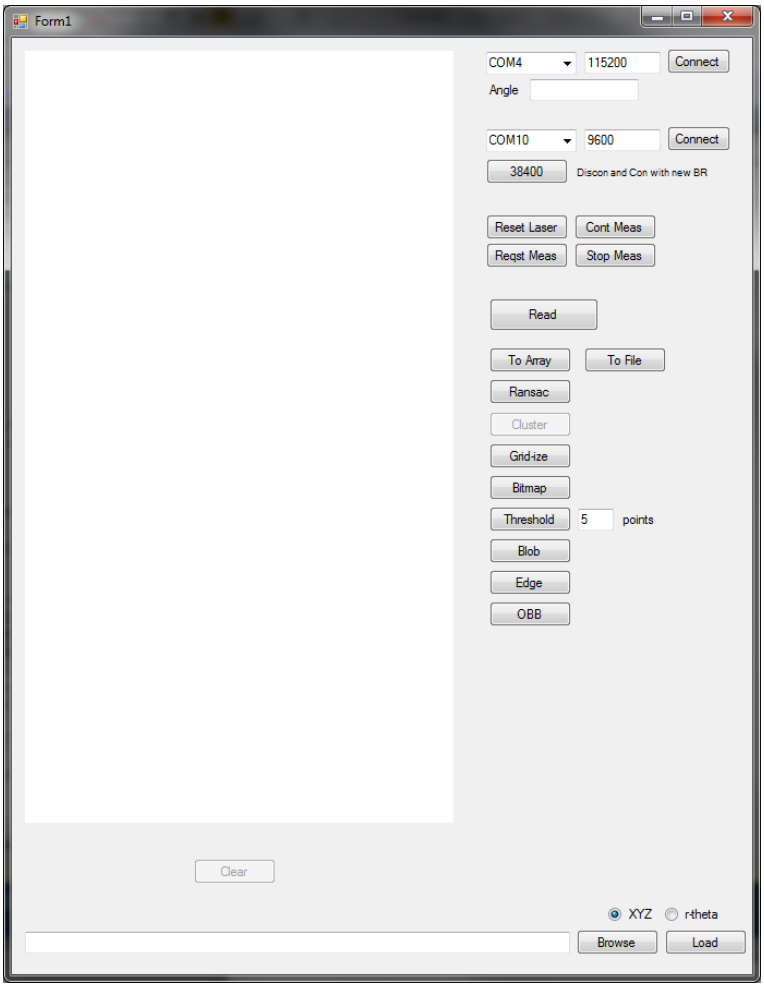

Figure 3.19: Interface of the application.

The Sick LMS291 sensor is a sophisticated sensor and to use it efficiently, a simple library has been written in C#. This library automates various standard routines like connecting the device, creating start bytes in commands, calculating checksums, preparing the structures of commands or throwing an event when data comes to the serial port of the computer. When a full reading is ready, the data is filled to an array with an additional data which is the orientation data coming from the potentiometer. The Sick LMS291 gets data from 0 to 180

degrees with 0.5 degrees intervals. This means every reading has 361 range values and these range values have an angle difference of 0.5 degrees. With this information and the orientation information coming from the potentiometer, coordinates of the points that are measured can be calculated easily. The calculations can be seen in Equations. In these equations, phi is the angle of a laser beam measured from the X axis (or the first laser beam) of the laser sensor and theta is the pitch angle of the sensor.

After registering the measurement to a Cartesian coordinate system, the first step is applying RANSAC algorithm to find a rough estimate of the ground plane. As mentioned in Chapter 3.2, some points which have the Y value lower than the first sight of ground from the point of view of laser sensor are removed from the list temporarily before the RANSAC algorithm is run. With some random trials, the RANSAC algorithm finds a representation of ground plane and outputs a list of points. This list includes the points which have the distance to the plane below the predetermined threshold. Since some points were removed before the RANSAC process, their distances to the plane are measured in a separate loop and the inliers are added to the list created by RANSAC. The rest of the points are considered as outliers and the rest of the algorithm works on this point cloud from which the ground is removed. A scan can be seen in Fig. 3.20 with the plane which is found by the RANSAC process.

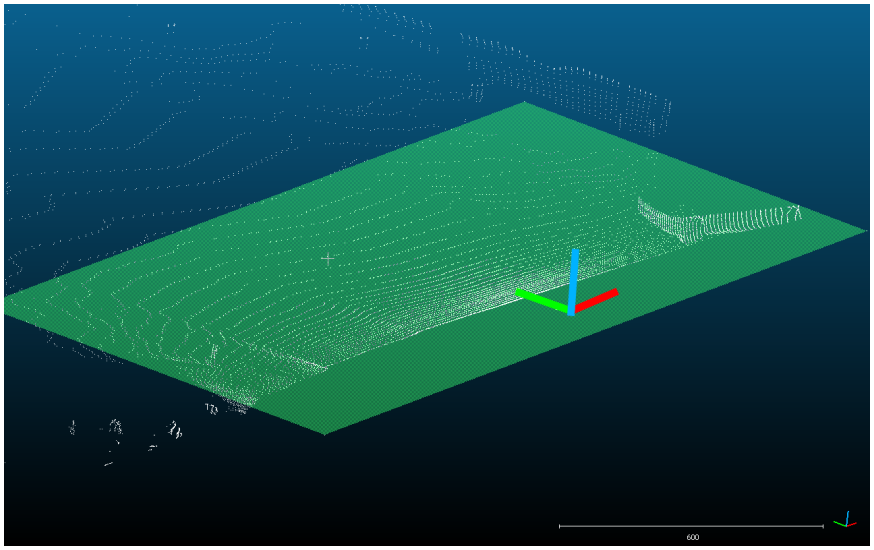

Figure 3.20: 3D point cloud of a scene and the ground plane (green) which is found by the RANSAC algorithm.

Since most of the ground is removed from the point cloud, all left are obstacles, noise and some points which belong to the ground but not on the same plane with the one that found by RANSAC. After this point, image process algorithms will be used to proceed in the most efficient way. To use image processing algorithms, first the point cloud must be discretized. For discretization, a grid size must be determined according to the application and the sensor that is being used. In this thesis, the size of a grid is selected as 20cm. Even though the maximum range of Sick LMS291 is 80 meters, maximum range for this application is selected

as 40 meters since the point density becomes too low for discretization for long ranges. Then a 2D matrix is created with the sizes of two times maximum range over grid size in X axis and maximum range over grid size for Y axis. The 3D point cloud is converted to a 2D point cloud just by removing (or ignoring) the Z data of the points. This gives the projection of the points to the XY plane.

After the 2D matrix is created, the number of points which lies in the area of a grid is noted as the value of that grid by looping through the points. To evaluate this information with image process algorithms, this 2D matrix is converted to a grayscale image where every cell is treated as a pixel and the value of the cell is the brightness level of that pixel. Since this image actually gives the density of the point cloud, it can be named as density map.

The density map is constructed from the points which are labeled as outliers by the RANSAC algorithm. The outlier points have noise and some ground points. An efficient filtering would be emptying the grids which have a density lower than a predetermined value. This value should be determined according to the type of application, sensor type and grid size. Since the density information is held as a grayscale image, the filtering can be performed by a simple threshold function. In this thesis AForge library is used for the image process algorithms. AForge is a widely used library which has many methods for computer vision, image process, machine learning and some other fields. It is very user friendly and written for C# which makes it a suitable selection for this work. The density map undergoes threshold method and the pixels which have a density lower than 5 points per grid are cleared to black and the others are registered as white on a new image.

The map consists of several white islands on a black background after the threshold step. The first thing to do at this point is to identify the separate islands. The best way to do this is using a connected component method. AForge has a filter named Connected Component Labeling for this purpose. Using this filter creates separate smaller images from the original image which contain only one connected component in them.

As mentioned in Chapter 3.6, to find the bounding boxes, the algorithm doesn't need to look at every pixel in a connected component; it just needs the edge pixels. To get the lists of edge pixels of every connected component, an edge detection algorithm is applied. In AForge library, a method named "GetBlobsEdgePoints" gives a list of edge pixels for given connected component. By using this method, lists of edge pixels of every connected component are created.

Creating oriented bounding boxes is not a trivial job. As it is mentioned in Chapter 3.7, several trials are done for different rectangles with different orientations and the one with the minimum area is selected. In application, rotating a rectangle before determining its size is not a simple job but rotating the edge points then creating a rectangle around them is much simpler. Because of that, the edge points of every component are rotated by 5 degrees from 0 to 85 degrees and the minimum and maximum X and Y values are noted in a separate list for every orientation of every component. Then a simple method calculates rectangle areas for these X and Y values and outputs the best orientation by finding the rectangle with the minimum area. When the orientation for the minimum area rectangle is found, the negative of the orientation value is applied to the four points of the rectangle as rotation resulting the minimum oriented bounding rectangle of the connected component. At the end of this procedure, the result is only four points for each of the bounding rectangles per connected component in the scan of the environment.

CHAPTER 4

EXPERIMENTAL SETUP

In order to prove the concepts that are been discussed, detailed experimantal tests are conducted. Main idea of this test setup is to collect the laser scanning data and combine them with the angular data to obtain a 3D scanning result.

This chapter includes the details of the proposed hardware design. Firstly sensor background is mentioned which is used during the thesis. Then hardware setups are mentioned in detail.

4.1 Design of the Rotating Mechanism of Laser Sensor

The laser sensor that is used in this work is Sick LMS291. This sensor can only get measurements on a plane. To get a volumetric scan, the sensor can be rotated around its X axis. Sick LMS291 weights approximately 4.5 kg and its weight must be supported by the rotating mechanism. A structure made by a former lab member Emre Sezginalp is used as a base for this construction. This structure can be seen in Figure 3.5. This mechanism has four mounting points to mount the sensor on it and it lets the sensor to rotate around its X axis. Even though it lets the sensor to rotate and supports its weight, it doesn't have a sensor to measure the orientation and it doesn't have an actuation member. It is basically just a swing.

To register the measurements taken by the laser sensor correctly to the global coordinate frame, the orientation of the laser sensor must be known. A basic method to measure the orientation of the sensor is mounting a potentiometer on its rotating axis. A 270 degree potentiometer is mounted on the rotating axis of the system as seen in Figure. This potentiometer is connected to a microcontroller board which only reads the potentiometer value and sends the readings via serial port to the computer. On the computer side, the algorithm takes the resistance value of the potentiometer and converts it to angle. Then the necessary geometrical calculations are made to register the measurements to the global coordinate frame.

To actuate the mechanism, several concepts are evaluated like manual, using a crank-rocker mechanism with a DC motor and using a servo motor rotating back and forth. Since there is a potentiometer connected to the system as a tilt sensor, the actuation mechanism doesn't affect the readings. Actuating the mechanism manually can be useful during the tests of the mechanism and the potentiometer. However, during the actual usage of the system, the mechanism must be actuated automatically. A simple actuation would be connecting a servo motor directly to the sensor by a simple four-bar mechanism as seen in the Fig. 4.1. Since the structure of the system supports the weight of the laser sensor and the center of gravity of the

sensor is almost on the rotating axis, the servo can easily rotate the sensor back and forth.

Figure 4.1: The mechanism which can rotate the sensor by an R/C servo and gets the information by a potentiometer.

4.2 Design of the Data Acqusition Setup

In the development process of the algorithms, gathering data from various environments is important to improve the robustness of the algorithms. The algorithm must be tested on these data and must be improved to eliminate any possible errors. In this project, all the algorithms are developed to be used on a UGV and ideally the data should be gathered by using the UGV but it is not simple because of some reasons. One of the most important reasons is, the other members of the laboratory group are working on the UGV and taking the vehicle out of the lab disrupts their work. In addition to this, some of the work being done on the vehicle may involve cabling removing some elements from it. At these times, the vehicle is unavailable and to gather data with it, one needs to wait it to be available. Another reason that using the UGV is not the preferred choice is that using the vehicle needs some

computerized control. One cannot drive the device by simply pushing it; it must be controlled via a computer. Since the vehicle is still under development process, another lab member may be working on the software which may cause problems driving the device at that moment. To overcome this problem, a manual controlled cart has been constructed. The cart is designed to have two wheels on sides and an aluminum chassis structure. In Fig. 4.2, design of the chassis can be seen. As wheels, 26" bicycle wheels were used. This cart is designed as a development platform so its structure is subject to change during the development process. Several different sensors may be mounted on or position of some components may be changed. To have a modular structure, as structural element, 30x30mm Sigma profiles were used. Sigma profiles have canals on them as can be seen in Fig. 4.3. These canals let mounting without drilling the material with the help of special nuts which can be seen in Fig. 4.4.

Figure 4.2: The preliminary design of the cart.

To have a multipurpose mobile platform to gather data, encoders are integrated to the wheels. Two encoder wheels with 128 teeth were produced with plasma cut and mounted on wheel rims with double sided tape and metal epoxy. To get data from the encoders, Omron E2A-S08KN04 proximity sensors were used which can be seen in Fig. 4.5. Omron E2A-S08KN04 is an inductive type proximity sensor. It can sense ferrous metals from 4mm and its output type is NPN open collector output. The response frequency of this sensor is 1kHz (average) as stated in its datasheet. It means that 1000 edges can be detected in 1 second. The encoder which is used on the cart has 128 teeth which is equal to 256 edges. Eqn. 4.1 shows that the

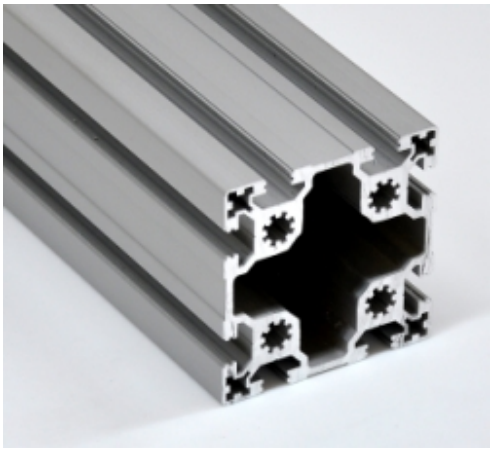

Figure 4.3: A sigma profile.

Figure 4.4: The special nuts which are used on sigma profiles.

maximum revolution that can be detected is 3.9 revolutions per second. The wheel diameter is about 65cm and Eqn. 4.2 gives the perimeter as 204.2cm. Eqn. 4.3 show that proximity sensors that are used lets the cart to be driven at about 8m per second which is equal to 28.8km/h. This speed is far more than the planned speed since the cart is designed to be manually driven.

$$\frac{1000edge/sec}{256edge} = 3.9rev/sec \tag{4.1}$$

$$\pi * D = \pi * 65cm = 204.2cm \tag{4.2}$$

$$204.2cm * 3.9rev/sec = 796.4cm/sec \tag{4.3}$$

Two sensors per wheel are used as quadrature encoders to have the direction information. The positioning of the sensors and the encoder can be seen in Fig. 4.6. A computer is installed to gather data from the sensors that planned to install on the cart. Since this cart is planned to be a manual copy of the actual vehicle, these sensors include stereo cameras, laser range finders or similar high level sensors. So the computer is selected to be a computationally strong one instead of a simple laptop. To transfer the data from the proximity sensors to the computer, an Arduino Uno board is used. Arduino Uno is a cheap microcontroller board based on the ATmega328 processor. It gets the encoder data from the proximity sensors and transmits it to the computer via USB port. In the future, some other low level sensors can be added to this setup like ultrosonic or infrared range sensors. Since the Arduino Uno has many input/output and processing capabilities, in addition to sending the encoder data, it can be used to control these sensors without the need of an extra processing unit.

Figure 4.5: Omron E2A-S08KN04 proximity sensor.

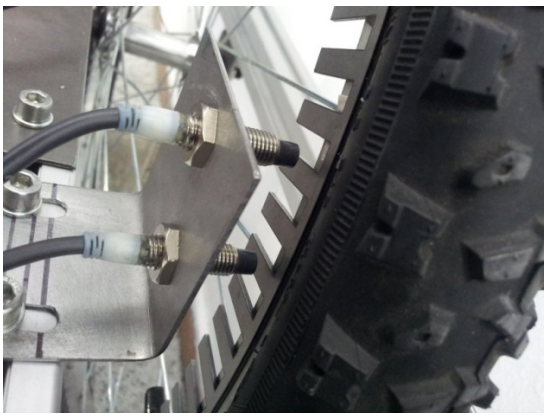

Figure 4.6: Proximity sensors and the encoder on the wheel.

The power of the computer and the sensors are given by a 2000W Honda generator which is also installed on the cart. Since generators are not reliable devices as uninterruptible power sources and a power outage even for a short period of time may cause loss of data, in addition to the generator, a 1KVA on-line uninterruptible power supply is also installed on the device. The setup can be seen in Fig. 4.7 with a Sick LMS291, a Bumblebee2, a Hokuyo URG-04LX and the power sources.

Figure 4.7: Data acqusition setup and the generator.

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Downsizing Map Data

Creating maps based on point clouds is mostly a memory requiring job. Using a 3D point cloud directly requires at least 3 values, which are X, Y and Z values, for every point in the cloud to be stored in the memory. The amount of bytes to be reserved in the memory to store the map depends on the dimensions and number of points of the point cloud. In addition to this, the algorithms used in localization, mapping, path finding or obstacle avoidance usually use the scan data. The amount of time taken by the algorithm is mostly related to the amount of the data. In this thesis, a preprocessing method is presented to downsize the amount of data and still leaving it as useful as the original scanned data. Using bounding rectangles instead of obstacles in the environment means using only 4 corner points for each obstacle instead of using all the data points of the obstacle. In Fig. 5.1, the point cloud of a laser scan and the bounding rectangle versions of the same scene is shown. This map gives the position and approximate area of the obstacles just with 4 points for each obstacle. To test the algorithms, 3 environments have been selected and several consequent scans have been taken in each of these environments to form 3 datasets. One of these datasets has been taken in an almost empty parking lot with a building on one side and woods on the other side. The second dataset has been taken in another almost empty parking lot but some boxes were in the middle of the parking lot to form some artificial features. For the third dataset, the setup has been moved between trees with a building on one side. A shot for each of the environments are given in Fig. 5.2.

5.2 Obstacle Avoidance and Path Planning

Bounding rectangles give the approximate position and size information of the obstacles in the environment. With this information, a mobile robot can navigate between these obstacles without trying to compute its algorithms for every single data point but just for four data points per obstacle. For example, if the robot is trying to avoid a collision with a tree trunk, it doesn't have to analyze the sensor measurements of the tree trunk but it just need to avoid with a collision with a simple rectangle which is in the place of the tree trunk on the map. In this representation, the rectangles represent the obstacles and let an object avoidance or a path planning algorithm to have a very simple map instead of a point cloud or some kind of occupancy map.

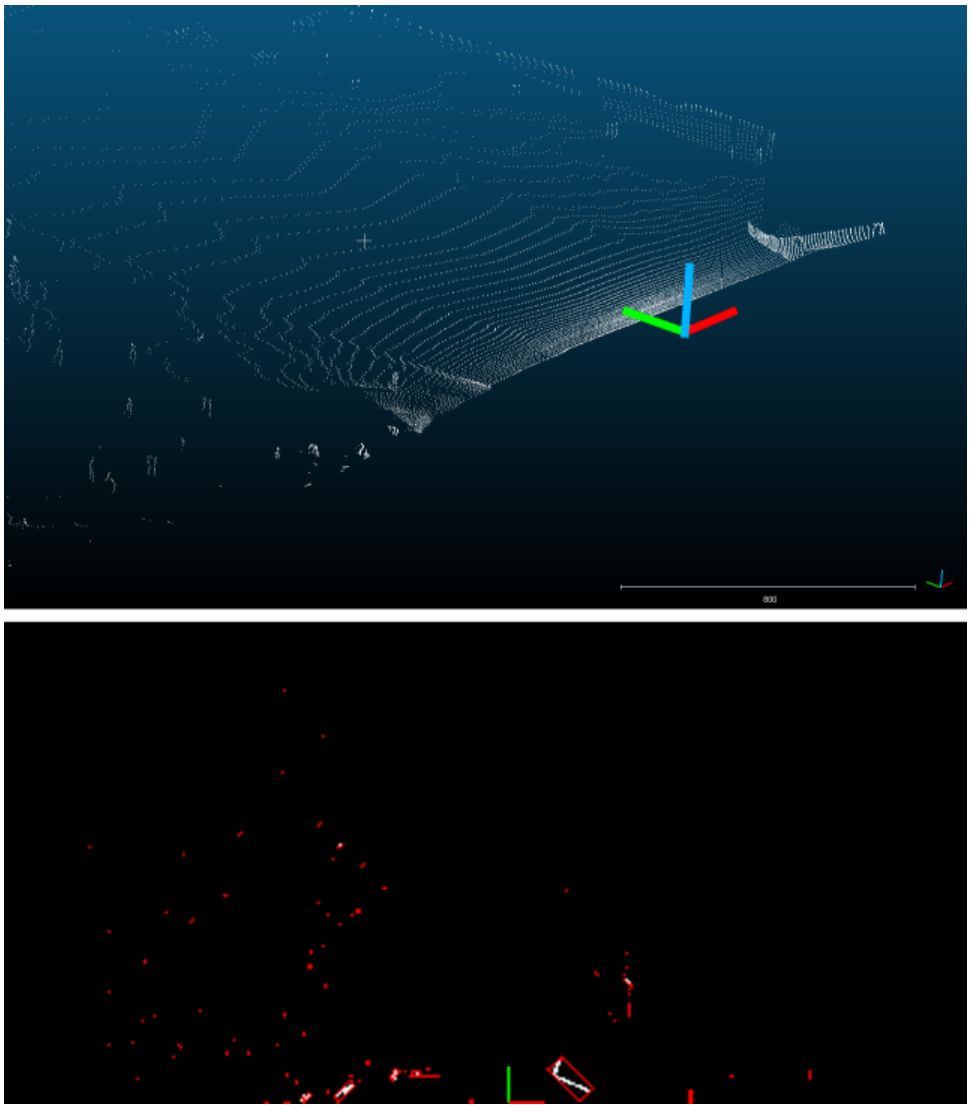

Figure 5.1: 3D point cloud of a scene (top) and bounding rectangle representation of it (bottom).

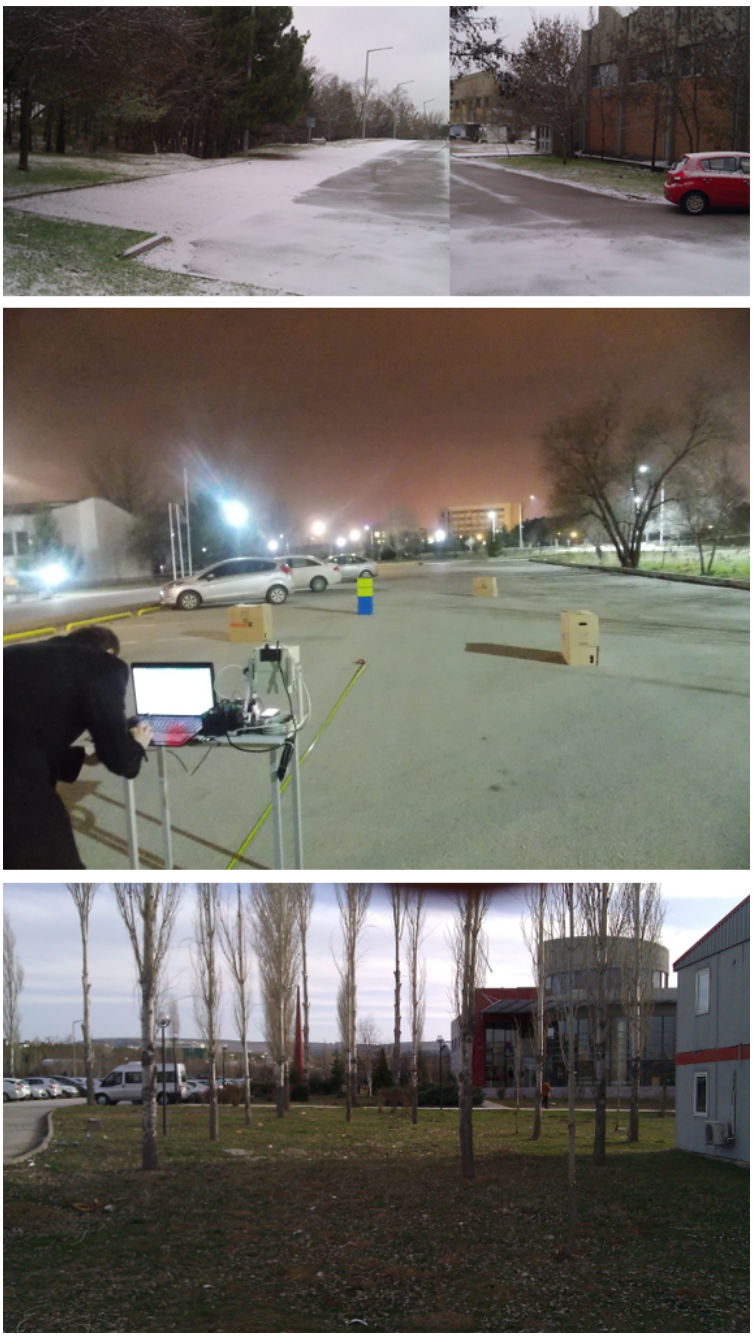

Figure 5.2: The environment of first dataset (top), second dataset (middle), and third dataset (bottom).

5.3 Dead Reckoning and Localization

The simplifying method given in this thesis uses the obstacles in the environment. Most of the localization algorithms use obstacles as features to calculate the location of the mobile robot. In this thesis, the simplifying methods give a simpler option to localization algorithms. As it is mentioned in Chapter 3.8, dilated version of a threshold map gives a new map with less connected components plus it closes small gaps on the map. The connected components on this image can be used directly as features for the dead reckoning or localization algorithms. In Fig. 5.3, a threshold map and dilated version of it are shown.

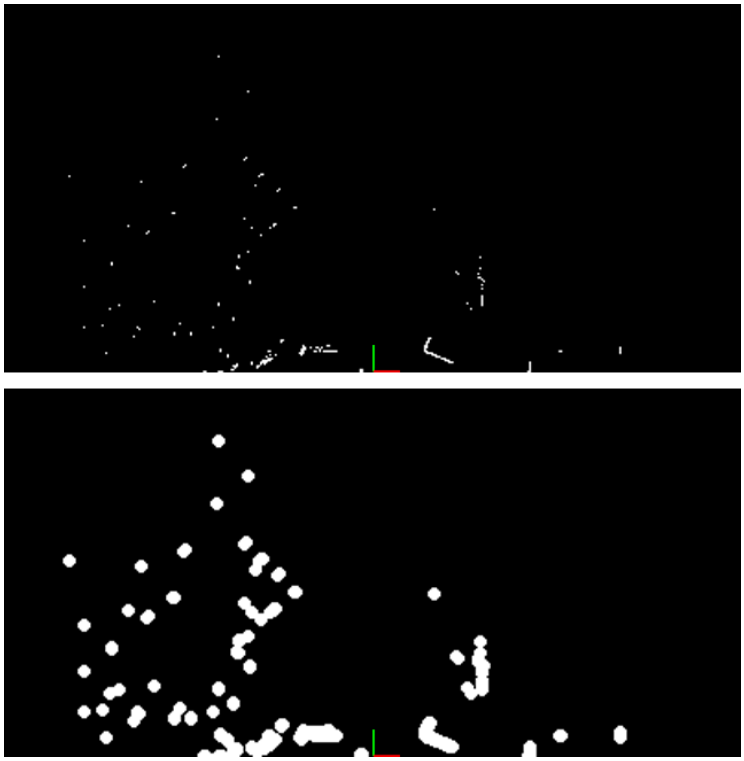

Figure 5.3: Threshold map of a scene (top) and dilated version of it (bottom).

In the same environment, some consecutive data has been captured. In Fig. 5.4, threshold maps of 9 consecutive measurements can be seen. To get these images, the sensor is moved 50cm between each measurement in the +Y direction. As it can be seen, to find some traceable features is not a simple job.

Dilating all these images gives the images in Fig. 5.5. In these dilated images, the gaps are closed and there are less connected components to track. And some of the points around the middle of the images are traceable through the image set. But the features which are near the side edges of the image are not that reliable. They sometimes vanish and sometimes comeback or vanishes completely. The reason of this phenomenon is shadowing.

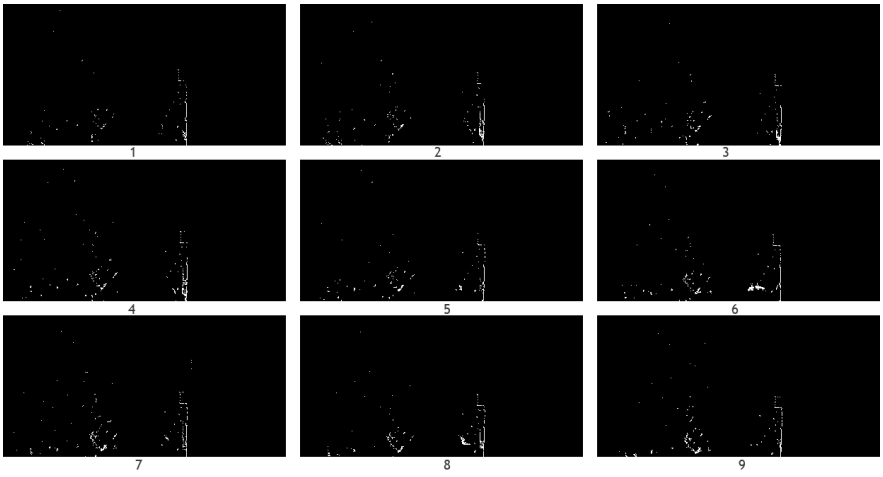

Figure 5.4: Threshold maps of 9 consecutive measurements. The distance between images is 50cm in Y direction.

The sensor that is used in this work is a laser sensor and the angle between its beams is 0.5 degrees. The edges of these images are at around 40 meters from the sensor. A simple calculation gives the distance between the beams at 40 meters as 35 cm. This means, if some obstacle near the sensor blocks even 1 laser beam during the scan, it can hide obstacles with a width of 35cm at 40 meters form the sensor. This causes some of the obstacles to stay in the shadow of other obstacles time to time during the movement. But this problem rises for the objects far from the sensor. The images in Fig. 5.6 are the same images from the Fig. 5.5 but they are all cropped up to 25 meters. As it can be seen, the shadowing problem is not dominant at this range and almost all of the features are traceable. A simple localization algorithm can use this data to find the displacement between two measurements.

5.4 Occupancy Grid Map

Occupancy grid maps are mostly used in probabilistic robotics to generate maps from noisy and uncertain sensor data. Occupancy grid map is basically a grid which represents the map by having probability values of the presence of an obstacle at that location in its cells. It is mostly represented as a grayscale image with black pixels as obstacles and white pixels as traversable areas. The gray value of a pixel represents the occupancy probability of that area. For example a 50% gray pixel means that nothing is known about that area it can be occupied or not by a 50% probability. But a dark gray pixel means that there is most likely an obstacle at that position. These maps are created by adding sensor measurements to each other according to the displacement between the measurements. The occupancy maps created with the consecutive measurements of three datasets are given in Fig. 5.7. This figure has been created with dilated and inversed versions of threshold maps. These measurements have been gathered by moving the sensor 50cm between each measurement along 7 meters. The opacity values of the images have been lowered separately and they have been added with 50cm offset. As it can be seen, the obstacles can be easily located through consecutive measurements even they disappear in some images because of shadowing.

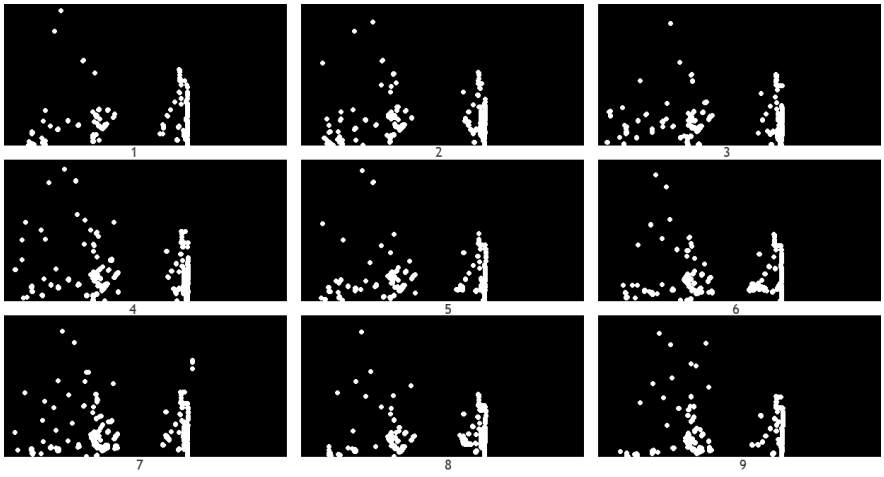

Figure 5.5: Dilated versions of the images in Fig. 5.4.

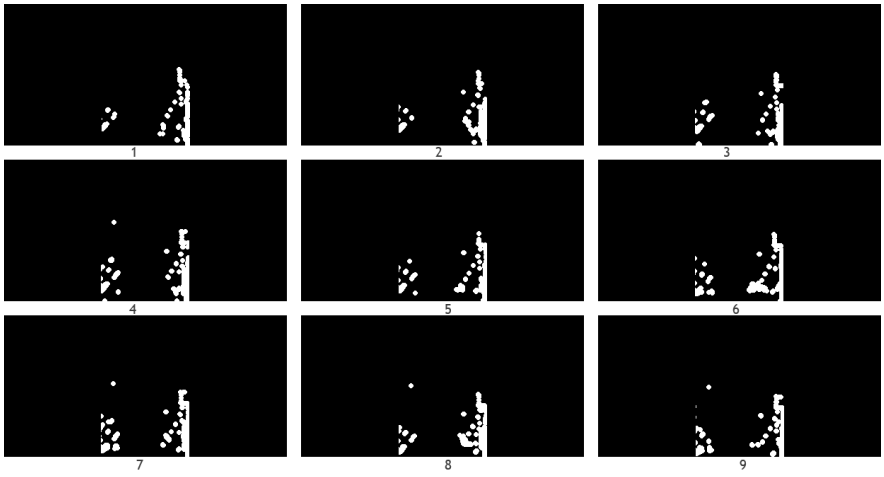

Figure 5.6: Cropped versions of the images in Fig. 5.5 up to 25 meters.

In addition to standard procedure of creating occupancy grid map, an improvement can be made. Normally, sensors gives noisy data and some probabilistic methods like creating occupancy grid maps reduces the probability of taking a noisy measurement as an obstacle. But throughout the algorithms given in this thesis, in the threshold part which is given in Chapter 3.5 a simple filtering for noise is done. Since the data is already filtered, the dilated maps or threshold maps which are the base for the occupancy grid map, the probability algorithm of this map can be altered. Since the data is relatively reliable with respect to the original sensor measurements, a threshold can be applied to the probability values. For example, if a feature is present in 60 images out of 100, its probability can be taken as 100% instead of 60%. But if a feature is present on only 20 images out of 100, the probability is left untouched.

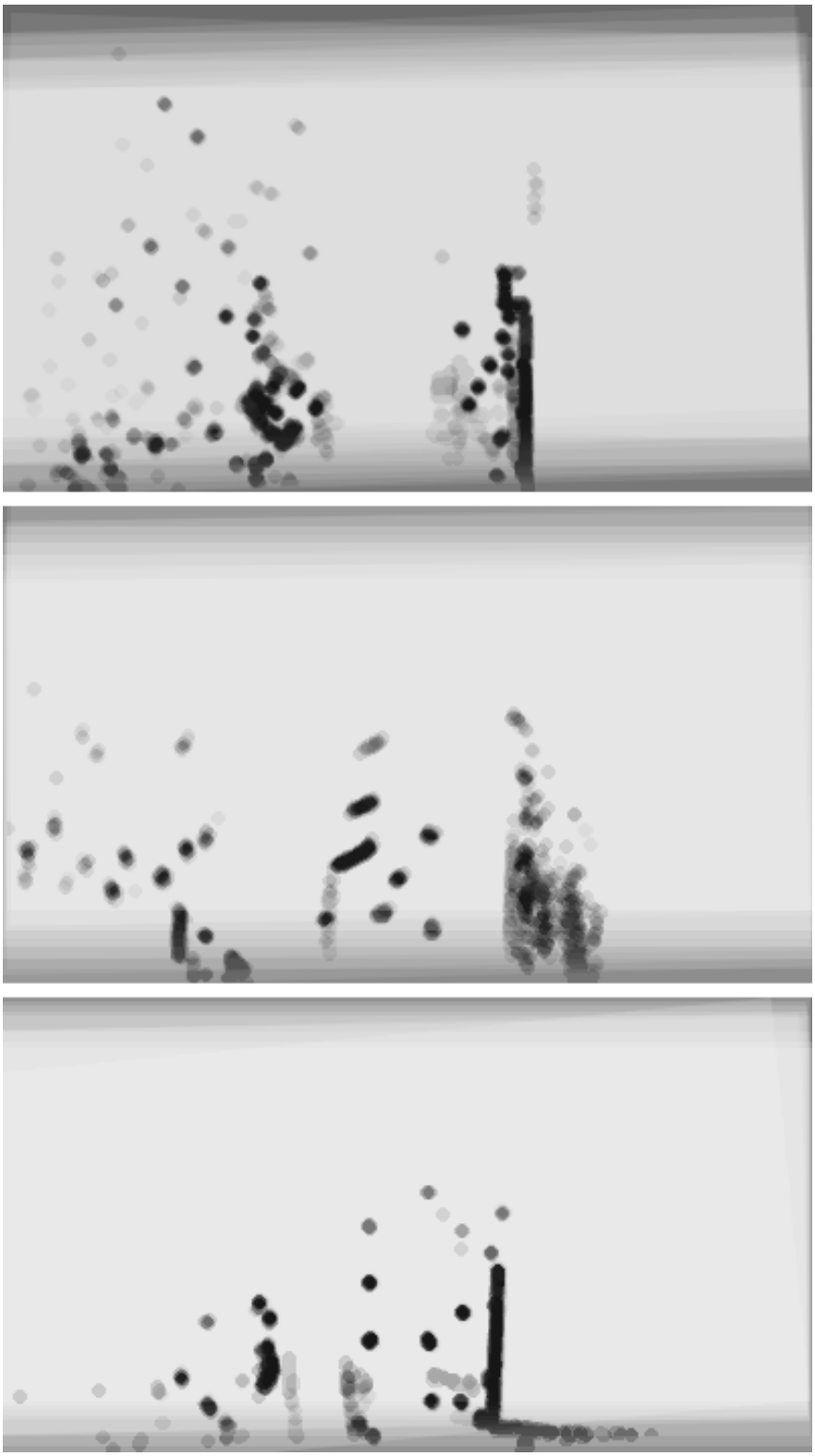

Figure 5.7: Occupancy map of 15 consecutive measurements of first dataset (top), second dataset (middle) and third dataset (bottom).

CHAPTER 6

CONCLUSION AND FUTURE WORK

Main motivation behind this study is to decrease the amount of data that will be proceseed by the main processor. By this procedure, downsizing map data, obstacle avoidance, path planning, dead reckonning, localization and mapping using occupancy grid have been evaluated effectively. As expained during thesis, those methods are used and real experimental results are given which are conducted by using a test system.

In addition to the advantage of downsizing the processed data, by applying threshold to the density map, accuracy of the resultant map increased. In other words, the probability of the resultant maps being accurate are increased by using less amount of data.

As a future work our purpose is to increase the effectiveness of the algorithms that are proposed. Note that using SICK and the presented test setup one can obtain 3-D point cloud of the environment. And as a initial data processing step, ground plane must be detected and subtracted from the rest of the image. And failure during this process, results unexpected obstacle presence.

Also as a next experimantal proceure, the proposed algorithms will be tested on our UGV which is desinged in our laboratory. So the processor in this machine will be operate more effectively since the amount of data that must be processed is decreased significantly.

REFERENCES

[1] G. Aslan. Design and implementation of a scanning platform for mobile robotics. Master's thesis, METU, 2012.

[2] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38:91–109, 2001.

[3] G. Bayar, İlhan Konukseven, B. Koku, T. Balkan, and A. Erdener. Atv tabanlı İnsansız kara aracı geliştirilmesi. *Makina Tasarım ve İmalat Dergisi*, 8:54–66, 2007.

[4] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for ppoints and rectangles. *ACM SIGMOD Int. Conf. on Manag. of Data*, 1990.

[5] S. Bespamyatnikh and M. Segal. Covering a set of points by two axis-parallel boxes. *Proc. 9th Canad. Conf. Comput. Geom.*, 1997.

[6] D. M. Carroll, K. Mikell, and T. Denewiler. Unmanned ground vehicles for integrated force protection. *Unmanned Ground Vehicle Technology VI*, 1:367–377, 2004.

[7] C.-T. Chang, B. Gorissen, and S. Melchior. Fast oriented bounding box optimization on the rotation group so(3,r),. *ACM Transactions on Graphics*, 30:122, 2011.

[8] G. B. B. Chazelle, L. J. Guibas, J. S. B. Mitchell, and A. Tal. Boxtree: A hierarchical representation for surfaces in 3d. *Proc. Eurographics'96*, 1996.

[9] Ensco. Team ensco darpa grand challenge technical paper. *Technical Paper*, 1:1, 2004.

[10] L. A. R. Escriba, L. A. Rivera, V. V. Estrela, L. Velho, V. V. Estrela, P. C. Carvalho, and V. V. Estrela. Oriented bounding boxes based on multi-resolution contours. *JOURNAL OF WSCG 2004*, 2004.

[11] O. D. Faugeras and J. Ponce. An object-centered hierarchical representation for 3-d objects: The prism-tree. *Computer Vision, Graphics, and Image Processing*, 38:1–28, 1987.

[12] D. Ferguson, M. Darms, C. Urmson, and S. Kolski. Detection, prediction, and avoidance of dynamic obstacles in urban environments. *IEEE Intelligent Vehicles Symposium*, 2008.

[13] H. Freeman and R. Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Comm. ACM*, 1:409–413, 1975.

[14] S. Gottschalk, M. C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. *Proc. SIGGRAPH*, 1996.

[15] M. Held, J. T. Klosowski, and J. S. B. Mitchell. Evaluation of collision detection methods for virtual reality fly-throughs. *Proc. 7th Canad. Conf. Comput. Geom.*, 1995.

[16] M. Himmelsbach, A. M. T. Lüttel, and H. J. Wünsche. Lidar-based 3d object perception. *Proceedings of 1st International Workshop on Cognition for Technical Systems*, 2008.

[17] P. M. Hubbard. Collision detection for interactive graphics applications. *Visualization and Computer Graphics*, 1995.

[18] P. Kmiotek and Y. Ruichek. Object's oriented bounding box based representation using laser range finder sensory data. *IEEE International Conference on Vehicular Electronics and Safety*, 2008.

[19] U. S. Military. Darpa Grand Challenge. `http://http://archive.darpa.mil/grandchallenge05/`, 2005. Online; accessed 26.02.2013.

[20] S. Murphy, Karl N.and Legowik. Gps-aided retrotraverse for unmanned ground vehicles. *Navigation and Control Technologies for Unmanned Systems*, 2738:133–142, 1996.

[21] T. S. of Las Vegas. Team ensco darpa grand challenge technical paper. *Technical Paper*, 1:1, 2004.

[22] J. O'Rourke. Finding minimal enclosing boxes. *Internat. J. Comput. Inform. Sci.*, 14:183–199, 1985.

[23] N. Roussopoulos and D. Leifker. Direct spatial search on pictorial databases. *Proc. ACM SIGACT-SIGMOD Conf. Principles Database Systems*, 1985.

[24] H. Samet. *Spatial Data Structures: Quadtrees, Octrees, and Other Hierarchical Methods.* Addison-Wesley, 1989.

[25] R. Schnabel, R. Wahl, and R. Klein. Shape detection in point clouds. *The Eurograpics Association and Blackwell Publishing*, 2006.

[26] T. Sellis, N. Roussopoulos, and C. Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. *Proc. 13th VLDB Conference*, 1997.

[27] E. Sezginalp. Simultaneous localization and mapping for a mobile robot operating in outdoor environments. Master's thesis, METU, 2007.

[28] W. Stuerzlinger. Bounding volume construction using point clouds. *Spring Conference on Computer Graphics*, 1996.

[29] S. Thrun. Winning the darpa grand challenge. *Journal of Field Robotics*, 1:22, 2006.

[30] G. T. Toussaint. Solving geometric problems with the rotating calipers. *Proc. MELECON '83*, 1983.

[31] A. Trebi-Ollennu and J. Dolan. An autonomous ground vehicle for distributed surveillance: cyberscout. *Robotics Institute*, 1:213, 1999.

[32] E. Welzl. *Smallest enclosing disks (balls and ellipsoids.* New Results and New Trends in Computer Science, 1991.