MAPPING AND OBSTACLE AVOIDANCE ALGORITHMS FOR QUADROTORS IN THE INDOOR ENVIRONMENTS

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

BY

ÖMER ORAL

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

NOVEMBER 2019

Approval of the thesis:

MAPPING AND OBSTACLE AVOIDANCE ALGORITHMS FOR QUADROTORS IN THE INDOOR ENVIRONMENTS

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

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

Prof. Dr. M. A. Sahir Arıkan
Head of Department, **Mechanical Engineering** _____

Assist. Prof. Dr. Ali Emre Turgut
Supervisor, **Mechanical Engineering Department, METU** _____

Assist. Prof. Dr. Kutluk Bilge Arıkan
Co-supervisor, **Mechanical Engineering Department, TEDU** _____

Examining Committee Members:

Assoc. Prof. Dr. Mehmet Bülent Özer
Mechanical Engineering Department, METU _____

Assist. Prof. Dr. Ali Emre Turgut
Mechanical Engineering Department, METU _____

Assist. Prof. Dr. Kutluk Bilge Arıkan
Mechanical Engineering Department, TEDU _____

Assoc. Prof. Dr. Ulaş Yaman
Mechanical Engineering Department, METU _____

Assist. Prof. Dr. Selçuk Himmetoğlu
Mechanical Engineering Department, Hacettepe University _____

Date: 28.11.2019

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

Name, Surname: Ömer Oral

Signature :

ABSTRACT

MAPPING AND OBSTACLE AVOIDANCE ALGORITHMS FOR QUADROTORS IN THE INDOOR ENVIRONMENTS

Oral, Ömer
M.S., Department of Mechanical Engineering
Supervisor: Assist. Prof. Dr. Ali Emre Turgut
Co-Supervisor: Assist. Prof. Dr. Kutluk Bilge Arıkan

November 2019, 82 pages

Recently, there has been increased interest for search and rescue missions with autonomous flying vehicles. However, as most of the designed techniques are suitable for outdoors, only a few techniques have been developed for indoors. SLAM (Simultaneous Localization and Mapping) is a method that allows autonomous robots to navigate in both indoor and outdoor environments. Localization part can be easily performed using a GPS(Global Positioning System) outdoors. On the contrary, GPS cannot be used indoors. In this study, the aim is to obtain 2D map of indoor environments without hitting any obstacles by using a quadrotor that is capable of autonomous navigation. Local positioning system is established with a UWB(Ultra Wide-Band) sensor and a LIDAR(Laser Imaging Detection and Ranging) is used to obtain the map of the unknown indoor environment. A novel algorithm, which maps indoor environments autonomously, is designed and presented. It is compared with two known navigation algorithms with the help of various metrics in order to measure the performance of the presented algorithm. One of the known algorithms directly fails on bigger maps with obstacles while the other one is overtaken by the presented

novel algorithm during comparisons although it successfully completes the mapping process. The algorithms have obtained similar results in some simulations on small map. However, the novel algorithm beats the opponents by completing the tasks with better scores regardless of the size of the indoor environments.

Keywords: LIDAR, SLAM, Mapping, UWB Technology, Trilateration Technique, Quadrotor, UAV, Indoor, Obstacle Avoidance

ÖZ

İHA İLE GPS KULLANMADAN VE ENGELLERE ÇARPMADAN KAPALI ALANLARIN HARİTASININ ÇIKARILMASI

Oral, Ömer
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Ali Emre Turgut
Ortak Tez Yöneticisi: Dr. Öğr. Üyesi. Kutluk Bilge Arıkan

Kasım 2019 , 82 sayfa

Son dönemlerde, otonom uçuş kabiliyetine sahip hava araçlarlarıyla yapılan arama kurtarma operasyonlarına ilgi artmıştır. Geliştirilen tekniklerin büyük kısmı dış mekanlarda kullanılmaya elverişli olduğundan iç mekanlar için geçerli az sayıda teknik bulunmaktadır. Eş zamanlı konum tespiti ve haritalama otonom robotların açık ve kapalı alanlarda gezinim yapmasına olanak veren bir yöntemdir. Açık alanlarda konum tespiti GPS kullanılarak kolaylıkla gerçekleştirilebilir. Buna karşın kapalı alanlarda GPS kullanılamamaktadır. Bu çalışmada, otonom gezinim yapabilen bir kuadrotor yardımıyla herhangi bir engele çarpmadan kapalı alanların iki boyutlu haritasının çıkartılması hedeflenmiştir. Ultra Geniş-Bant teknolojisi kullanılarak bir lokal konumlandırma sistemi oluşturulmuştur ve bir lazer mesafe ölçer yardımıyla bilinmeyen kapalı alanların haritası çıkartılmıştır. Kapalı alanlarda otonom bir şekilde dolaşırken haritalama yapabilecek yeni bir algoritma tasarlanmış ve sunulmuştur. Bilinen iki gezinim algoritması ile kıyaslamalar yapılarak sunulan algoritmanın performansı ölçülmeye çalışılmıştır. Bu kıyaslama esnasında çeşitli performans kriterlerinden faydalanılmıştır. Bilinen algoritmalardan biri içerisinde engeller barındıran büyük hari-

talarda direkt olarak başarısız olurken, diğer algoritma haritalama işlemini düzgün bir şekilde gerçekleştirmesine rağmen kıyaslamalar esnasında sunulan yeni algoritmaya yenik düşmüştür. Küçük çaplı haritalarda gerçekleştirilen bazı benzetimlerde algoritmalar benzer sonuçlar elde etmiştir fakat sunulan yeni algoritma harita büyüklüğünden bağımsız olarak görevini her durumda tamamlamış ve aldığı başarılı sonuçlarla rakiplerine üstünlük sağlamıştır.

Anahtar Kelimeler: Lazer Ölçüm Cihazı, Haritalama, Eş Zamanlı Konum Tespiti ve Haritalama, Ultra Geniş-Bant Teknolojisi, Trilaterasyon Tekniği, Kuadrotor, İnsansız Hava Aracı, Kapalı Alanlar, Engellerden Kaçınma Algoritması

To my family

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere appreciation and gratitude to my supervisor Asst. Prof. Dr. Ali Emre Turgut and co-supervisor Asst. Prof. Dr. Kutluk Bilge Arıkan for their continuous support, encouragement, criticism and invaluable guidance throughout my thesis study. Their support was not only limited to my academic studies, but they also always helped me whenever I had a problem in my personal life.

I would like to specially thank my colleague and friend M. Burak Macit for his immense support and precious knowledge that he shared. He is always available to discuss my academic questions regardless of time and condition. I also need to apologize for the hours he missed in his life while helping me.

I would like to state my grateful appreciation to my parents Meral and Akın Oral, and my sister Zeynep Oral for their endless love and support. They have always been encouraging, patient and helpful to me throughout my years of study and in my life. Their enlightened vision helped me to choose my goals and pursue them. This study would not be finished without their patience, support and encouragement.

I would like to thank Musab Çağrı Uğurlu for his friendship and technical support, and my friends Semih İnyurt, Kadir Akkuş, M. Burak Atak, Naci Koray Koç, Berk Alparslan, Mustafa Süyüm, Fatih Apaydın, Cesur Bahadır Çelik, Furkan Celen, Ege Kayalı, H. Alpay Küçüker, Derya Uğurlu and Aybeniz Akbaba for their invaluable friendship and encouragement.

Last, but not least, I express my deep-hearted thanks to Turkish Aerospace Industries Inc. for giving an opportunity to pursue academic studies.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

UAV	Unmanned Aerial Vehicle
SLAM	Simultaneous Localization and Mapping
UWB	Ultra Wide-Band
GPS	Global Positioning System
EKF	Extended Kalman Filter
MMW	Milimeter-Wave Radar
RFID	Radio Frequency Identification
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging or Laser Imaging Detection and Ranging
SONAR	Sound Navigation and Ranging
$R_x(\phi)$	Rotation Matrix About X-axis
$\hat{C}^{(a,b)}$	Transformation matrix from Body-a to Body-b
LTS	Long Term Support
ROS	Robot Operating System
Rviz	ROS Visualization Tool
OGRE	Object-Oriented Graphics Rendering Engine
IDE	Integrated Development Environment
URDF	Universal Robot Description Format
LQR	Linear-Quadratic Regulator
PID	Proportional Integral Derivative

CHAPTER 1

INTRODUCTION

Advances in air vehicles have been the key development of the century. Especially, aircrafts played a vital role during World War I and World War II. However, cost of the air vehicles and the difficulty and cost of pilot training led the aerospace industry to develop unmanned aerial vehicles. Not only civil industry but also military industry have been affected by these developments. Today, it is possible to encounter unmanned aerial vehicles that are easy to use, easy to manufacture and cost effective. UAV sector attracted the attention of researchers as a result of recent advances. At first, UAV's were controlled remotely in order perform dangerous tasks. Then, it was thought that the UAV's can move autonomously with the use of inertial sensors. Quadrotors gained popularity due to mostly their high maneuverability. Moreover, quadrotors do not need airfield for take-off and landing. Nevertheless, they have also disadvantages. Quadrotors have a limited flying time and small payload capabilities. In addition, a major part of the energy of the quadrotors is spent against gravity and providing stability. Still, they are preferred by researchers since making trade-offs is a mandatory part of engineering.

1.1 UAV Applications

UAV's can be used in a variety of tasks with the help of different sensors. Type of the UAV and the related sensors are altered depending on the operating area being indoors or outdoors. One of them is on the area of photography/cinematography. During celebrations, activities and events, quadrotors are used as an effective tool. Recently, it has been indispensable in cinema productions as well. Moreover, it is

commonly used in racing as a hobby. It can be used for entertainment purposes by children and adults as well as remote controlled vehicles in speed races. In addition, UAV's may perform pick up or drop tasks. They can be used to control remote areas. Similarly, they can transport the products from one point to another. For instance, Amazon Prime Air, which was introduced as future transport system for goods, aims to deliver products safely within 30 minutes by using drones. Finally, search, rescue and mapping are well-known usage areas of UAV's as they are objectives of this study. Drones might be used to inspect a dangerous area that is hard to explore by humans. In parallel, security forces may take advantage of UAV's in order to rescue the hostages in unknown environments.

1.2 What is Quadrotor?

Quadrotor is the member of multirotor aircraft family that involves the versions with 6, 8 and 10 rotors as well. Quadrotor has four independent rotors on the fixed base. It was tested first by scientist Etienne Oehmichen in 1920 [16]. It has vertical take-off and landing capability without the need of any airfield. In quadrotors, rotor 1 and 3 turn in clockwise direction, while rotor 2 and 4 turn in counter clockwise direction in order to eliminate torque effects as it is shown in Figure 1.1.

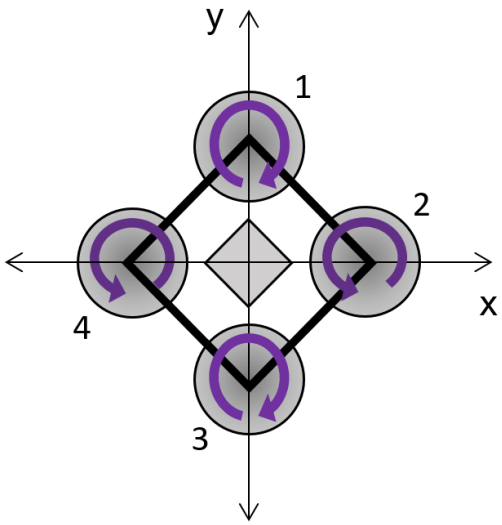


Figure 1.1: Direction of Rotation for Quadrotor Propellers

The reason of this kind contra-rotation is to eliminate yawing moment just like the task of the tail rotor in the helicopter. If they are turning in the same direction, the quadrotor would rotate where it was. Altering the speed of these independent rotors, it is possible to control the position and orientation of the robot. Increasing or decreasing the speed of four rotors simultaneously leads the quadrotor to move up and down in Figure 1.2 as follows,

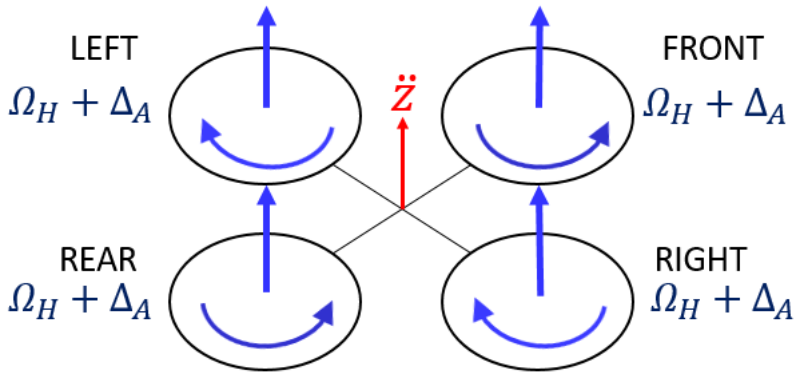


Figure 1.2: Thrust Movement

Roll, pitch and yaw movements are shown in Figures 1.3, 1.4, 1.5 as follows,

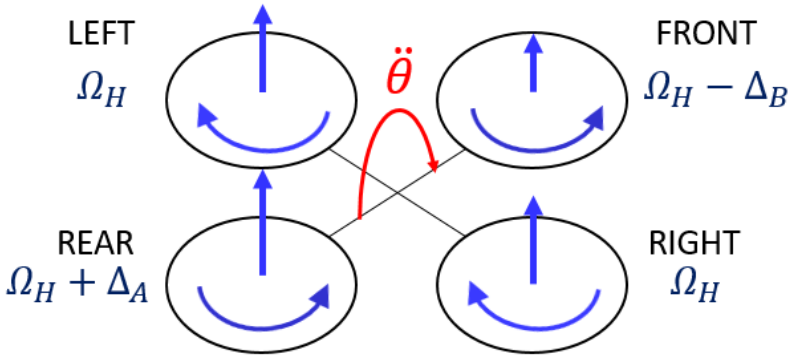


Figure 1.3: Pitch Movement

1.3 Aim of the Thesis

In literature, mapping process is performed in several ways. Camera and laser range finder are commonly used to map of an unknown environment. Many researchers have used different methods such as scan matching, SLAM, and computer vision.

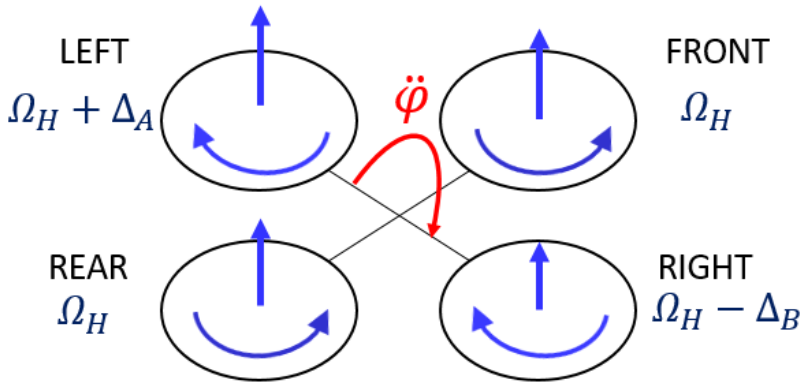


Figure 1.4: Roll Movement

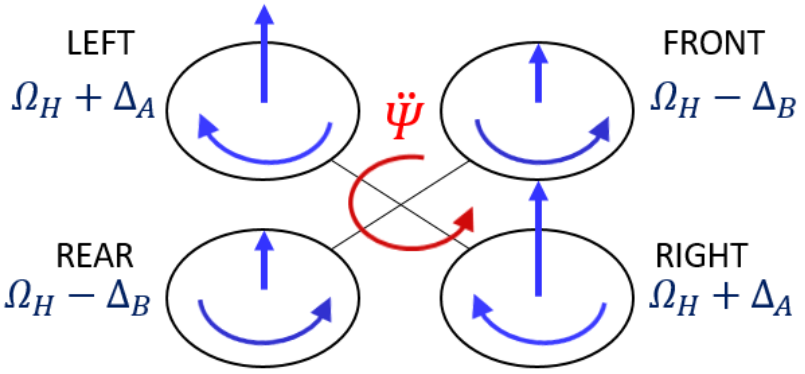


Figure 1.5: Yaw Movement

In this thesis, we aim to obtain a 2D map of an unknown indoor environment without hitting any obstacles by using a quadrotor that is capable of autonomous navigation.

1.4 Thesis Outline

Chapter 2 presents the literature survey of mapping applications with UAV's. Firstly, a short description is given about search and rescue operations that are performed by quadrotors equipped with various sensors. Secondly, the studies on SLAM problem are examined. Finally, indoor and outdoor SLAM applications carried out with different platforms and sensors in the literature are introduced.

Chapter 3 focuses on the the mathematical model and the exploration methods, which are used to explore indoor environments. At first, kinematics and dynamics of the

quadrotor are mentioned. Then, the navigation and mapping algorithms used in the simulation environment are expressed. Lastly, this section is completed with the explanation of control method.

Chapter 4 is devoted to experimental setup and experimental procedure. In the experimental setup part, each and every hardware and the software of the system are clarified. Then the procedure that is followed on the simulation environment is mentioned and the scenarios are explained. Finally, the performance metrics are introduced.

Chapter 5 includes the results of the performed simulations. Comparisons of different algorithms are interpreted.

Chapter 6 is the discussion section for the results presented in the previous chapter.

Chapter 7 summarizes the work done throughout the dissertation. It concludes the achievements of the thesis.

CHAPTER 2

LITERATURE SURVEY

In recent years, exploration, search and rescue operations in challenging and dangerous tasks started to be carried out by using unmanned and autonomous aerial vehicles. Performing these tasks indoors is much more complicated than outdoors since Global Positioning System (GPS) cannot be used indoors. Therefore, SLAM applications first started for outdoor environment due to ease of localization using GPS.

2.1 Simultaneous Localization and Mapping(SLAM)

SLAM is the process of simultaneous map extraction and robot positioning. The difficulty of this process can be easily understood from its definition. A map is required for correct positioning, while an accurate positioning is required for mapping. There are three main paradigms used for SLAM [36]. The first and the oldest one is Extended Kalman Filter (EKF) SLAM. It has lost its popularity due to computational complexity. Second one is the Graph-based SLAM. Since it can be successfully applied to non-linear optimization method, it has become the main paradigm for the solution of all SLAM problems. The last method, which is a non-parametric statistical filtering technique and a popular method for online SLAM, is the Particle Filter. It provides a new solution to SLAM's data association problem. Dissanayake et al. [14] introduce a solution for SLAM problem after clarifying the underlying reason of this problem. Also, they present an implementation of the SLAM algorithm by using milimeter-wave radar (MMW) to obtain map in an outdoor environment. Data assosciation and map management,which were mentioned theoretically, were verified with this implementation. Finally, the obtained results and actual positions of the map's

landmarks are checked. Nguyen et al. [28] demonstrate the experimentally validated Orthogonal SLAM (OrthoSLAM) algorithm. This algorithm is suitable for embedded robotic systems and capable of running real-time with low computational cost. They try to reduce complexity with an assumption of related environment. Considering the indoor environment, they manage to solve this issue by mapping just parallel and perpendicular lines of the primary structure. They obtain the map of their laboratory hallway and compare the test results with the measurements to check the accuracy of results. Dissanayake et al. [13] introduce a computationally efficient solution for SLAM process. They show that the efficiency of the SLAM process increases when the landmarks are selected properly.

2.1.1 Outdoor Environment

Arth et al. [7] come up with a new method for large-scale geo-localization and tracking of mobile devices in urban area by using video stream. They register a SLAM map by localizing the first frame according to 2.5D map that is transformed from the capture as shown in Figure 2.1. At this point, 2.5D map represents the area covered by the building in 2D and the height of the building is estimated approximately. During the process, the absolute camera orientation, which is estimated by using straight-line segments and the camera translation of the sides of the building in the image are matched. Reasonable pose is obtained to initiate SLAM at the end of this operation. Similarly, Cole et al. [11] use basic segmentation algorithm in order to separate the data stream into distinct point clouds that are referred to the position of the vehicle. A 3D scanning laser range finder is integrated to the vehicle and 3D SLAM is performed using improved 2D SLAM technology. Moreover, a new registration technique which is created by combining and optimizing the previous techniques is used to match frames.

Brenneke et al. [9] present a SLAM approach that is stand on leveled range scans. 2D SLAM is merged with 3D perception instead of using full 3D modelling in order to reduce the computational cost. This paper also describes the other steps of the process such as data acquisition and obstacle segmentation with showing the experimental results. Paz et al. [32] present a new SLAM technique that is valid for both large

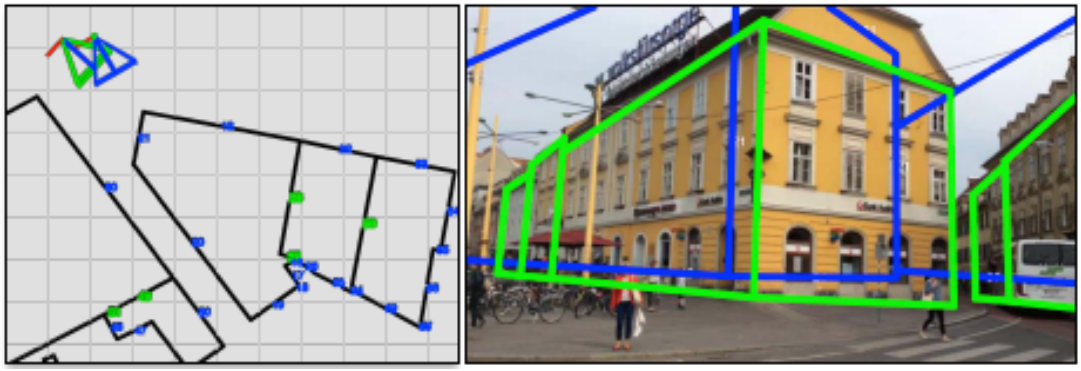


Figure 2.1: 2.5D Map and Straight-Line Segments [7]

indoor and outdoor environments. Different from existing visual SLAM systems that use both monocular information and 3D stereo information, their system cover both monocular and stereo. In order to get to map both near and far features, textured point features are taken out and stored as three-dimensional points. The SLAM algorithm creates the local maps first then the novel conditionally independent divide and conquer algorithm forms the full map. Experimental results in indoor and outdoor environment are shared to testify the robustness and the scalability of the system.

2.1.2 Indoor Environment

Indoor mapping process can be performed by using several methods. In order to discover indoor environments, ground based robots are commonly used due to ease of control [6], [10]. In studies with ground robots, while laser range finder and inertial measurement unit are generally used, Omara et al. [30] use Kinect instead of laser range finder since it is more economical. Different sensors are utilized when the unmanned aerial robots are used to obtain map of an indoor area. Johnson [21] ,in his thesis, manage the control of a quadrotor using image processing with camera. On the other hand, Ahrens et al. [5] manage to avoid collisions and obtain the map of indoor area. In addition to camera, Roberts et al. [34] prefer a quadrotor equipped with ultrasonic sensor and infrared sensor. By using these sensors, it is aimed to control the unmanned aerial vehicle without hitting any obstacle. The trials on test environment end up with reasonable results. Mohamed et al. [22] propose a new, cost-effective and simple indoor navigation system. In this system, three laser beams

integrated to the body of the UAV and they are oriented to the ground at a certain angle as shown in Figure 2.2. In order to determine the position and orientation of the UAV, several computer vision algorithms and a camera are used by tracking the laser marks on the ground.

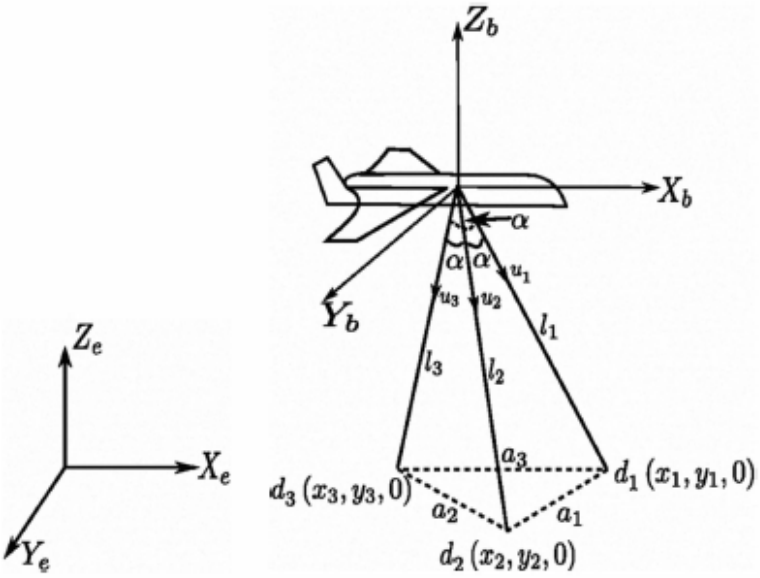


Figure 2.2: The Schematic Diagram of Navigation System Proposed by Mohamed et. al. [22]

Achtelika et al. [4] find a solution for navigation, exploration and object detection of quadrotors in unknown indoor environments. They define the design and operation of quadrotor by showing the architecture of the software and algorithms. The IMU is used as base structure. Additionally, visual or laser odometry algorithm is implemented in order to estimate the quadrotor's postion with respect to local environment. Extended Kalman Filter (EKF) is added to integrate odometry estimates and IMU data. Lastly, simple obstacle avoidance algorithm is applied with LQR controller to get more stability. Parallel to Achtelika's study, Grzonka et al. [18] perform mapping and exploration tasks using microcopter, inertial measurement unit and laser range finder. In contrast, a mirror is used to reflect laser beams to the ground and height of the air vehicle is determined with the help of laser marks. Wang et al. [37] present navigation and control for an UAV system that is operating in the indoor environment. The testbed quadrotor platform is equipped with an inertial measurement unit,

a camera which is looking downward, a barometer for height measurement, and a laser range finder to get planar map at the quadrotor's level. The robot can estimate its position and velocity, and fly in the room without colliding the obstacles. The velocity estimation by vision optical flow is shown in Figure 2.3. The system, which has algorithms running on board in real time, is verified in the test environment.

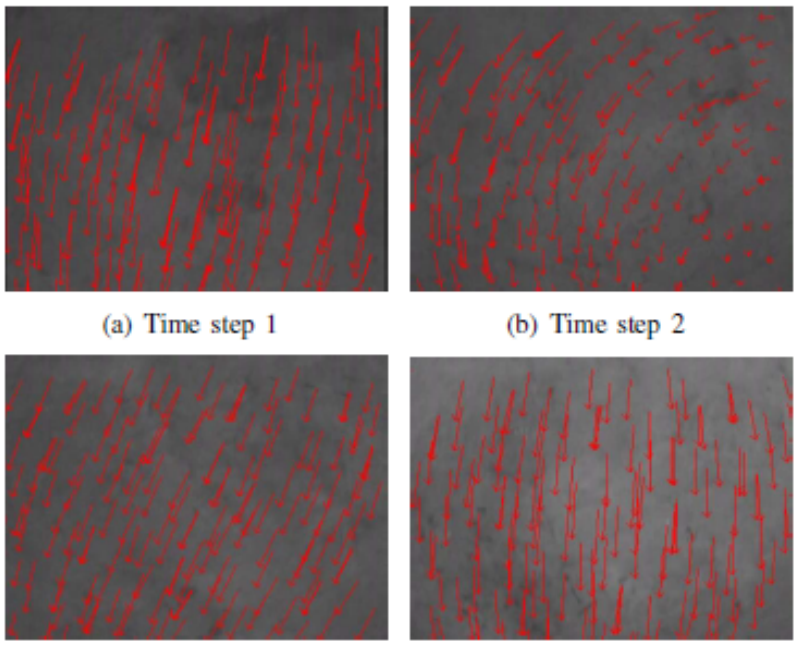


Figure 2.3: The Result of the 2D Optical Flow Implementation [37]

In many studies, inertial measurement unit and laser range finder are found enough for localization of the robot. However, wireless localization systems is also included in some studies. The system which is called RFID (Radio Frequency Identification) is divided into two types: active and passive. Basically, RFID systems are working using radio frequency signals to locate the position of the robot [26, 25]. It is possible to estimate the robot's position with another wireless localization system which is called UWB(Ultra-Wideband) technology. Although the communication with UWB technology is not preferred since it is a very costly system, Kempke et al. [23] manage to establish an adequetly and economic system. Barral et al. [8] develope a plugin including UWB communication for tracking forklifts in an indoor environment. They test various real scenarios on Gazebo simulator by defining different modes according to the UWB signal permeability of the obstacles in the environment.

In most of the studies, Gazebo, USARSim, V-REP and Webots are used as the physics based simulation platform[33, 29, 8]. Hector Slam, GMapping and Laser Scan Matcher are used as SLAM algorithm[15]. Generally, inertial measurement unit, laser range finder and camera are used to obtain map of an unknown areas.

2.1.3 Contribution of the Thesis

The focus of this study is to obtain 2D map of the unknown indoor environments without hitting any obstacles by using a quadrotor. Various sensors such as laser range finder, sonar and IMU are used to operate quadrotor autonomously. UWB sensor is used for localization and an UWB plugin is integrated to project to establish a realistic communication between quadrotor and UWB sensors. A novel navigation algorithm is presented and it is compared with two known algorithms. In order to measure performance, different maps are designed and various performance metrics are used.

CHAPTER 3

MATHEMATICAL MODEL AND METHODS

In this chapter, kinematics and dynamics of whole system are presented. Position and velocity analysis, equation of motions and the state space representations are mentioned in these subjects. Firstly, kinematics of the quadrotor is modelled. Secondly, dynamics of the quadrotor is derived by using the kinematic equations. Lastly, state space representation of the overall system and the control method are explained.

3.1 Kinematic Model

The basic rotation matrices, which are based on unit directions and rotation angles, are written both exponential and matrix forms in 3.1 as follows:

$$R_x(\phi) = e^{\tilde{u}_1\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{3.1}$$

$$R_y(\theta) = e^{\tilde{u}_2\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.2}$$

$$R_z(\psi) = e^{\tilde{u}_3\psi} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

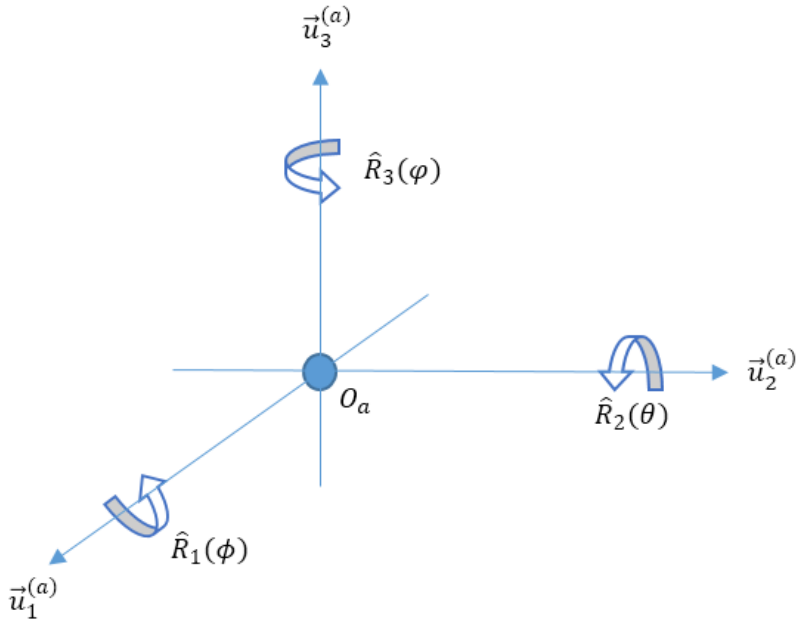


Figure 3.1: Basic Rotation Matrices of the Quadrotor

Transformation matrix is shown as follows:

$$R_{zyx}\left(\phi\theta\psi\right) = R_{zyx}\left(\psi\right) R_{zyx}\left(\theta\right) R_{zyx}\left(\phi\right) \tag{3.4}$$

or,

$$\hat{C}^{(i,b)} = e^{\tilde{u}_3\psi} e^{\tilde{u}_2\theta} e^{\tilde{u}_1\phi} \tag{3.5}$$

$$R_{zyx}\left(\phi,\theta,\psi\right) = \hat{C}^{(i,b)}$$
$$= \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{3.6}$$

Basic column matrices are required to carry out position and velocity analysis. Basic column matrices are expressed as follows:

$$\bar{u}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \bar{u}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \bar{u}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.7}$$

The position of the quadrotor can be defined in matrix form as follows:

$$\bar{p}_q^{(i)} = \begin{bmatrix} x & y & z \end{bmatrix}^T \tag{3.8}$$

Linear velocity matrix of the quadrotor can be defined by taking time derivative of the position matrix as,

$$\dot{\bar{p}}_q^{(i)} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{3.9}$$

Similarly, angular velocity of the quadrotor can be defined in the body-fixed reference frame as follows:

$$\bar{\omega}_q^{(b)} = \begin{bmatrix} p & q & r \end{bmatrix}^T \tag{3.10}$$

Angular velocity matrix can be written in another form as,

$$\bar{\omega}_q^{(b)} = \hat{L}\dot{\bar{\gamma}} \tag{3.11}$$

where $\hat{L}$ is the mapping matrix and $\dot{\bar{\gamma}}$ is the matrix which symbolises derivative of Euler angles as,

$$\hat{L} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{3.12}$$

The angular velocity in the inertial fixed reference frame can be obtained by using the angular velocity in the body-fixed reference frame as follows:

$$\bar{\omega}_q^{(i)} = \hat{C}^{(i,b)} \bar{\omega}_q^{(b)} \tag{3.13}$$

$$\bar{\omega}_q^{(i)} = \hat{C}^{(i,b)} \hat{L} \dot{\bar{\gamma}} = \hat{T} \dot{\bar{\gamma}} \tag{3.14}$$

In equation (3.14), $\hat{T}$ stands for the mapping matrix for the derivative of the Euler angles to the quadrotor's angular velocity in the inertial reference frame.

The generalized coordinates and velocities of the system can be defined using the obtained linear and angular velocities as,

$$\bar{q} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T \tag{3.15}$$

$$\dot{\bar{q}} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{3.16}$$

Then, linear and angular velocities are defined in terms of the generalized velocities as follow,

$$\bar{v} = \sum_{i=1}^{6} \bar{V}(i) \bar{q}(i) \tag{3.17}$$

$$\bar{\omega} = \sum_{i=1}^{6} \bar{\Omega}(i) \bar{q}(i) \tag{3.18}$$

Here, in equations (3.17) and (3.18), $\bar{V}$ represents linear velocity influence coefficient and $\bar{\Omega}$ sybolises angular velocity influence coefficient. Using these coefficients, the velocity of the quadrotor can be rewritten as,

$$\dot{\bar{p}}_q^{(i)} = \begin{bmatrix} \hat{I}_{(3x3)} & \hat{0}_{(3x3)} \end{bmatrix} \dot{\bar{q}} = \hat{V}_q \dot{\bar{q}} \tag{3.19}$$

$$\dot{\bar{\omega}}_q^{(i)} = \begin{bmatrix} \hat{0}_{(3x3)} & \hat{T} \end{bmatrix} \dot{\bar{q}} = \hat{\Omega}_q \dot{\bar{q}} \tag{3.20}$$

3.2 Dynamic Model

The Lagrange-d'Alembert formulation is used to obtain the equation of motion of the system. The Lagrange-d'Alembert formula is in the following form,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\bar{q}}}\right) - \frac{\partial L}{\partial \bar{q}} = \bar{u} + \bar{u}_{ext} \tag{3.21}$$

$$L = K - U \tag{3.22}$$

In order to get the equation of motion, kinetic and potential energies that is involved in equation (3.22) should be calculated as follow,

$$K = \frac{1}{2}\dot{\bar{p}}_q^{(i)T} m_b \dot{\bar{p}}_q^{(i)} + \frac{1}{2}\dot{\bar{\omega}}_q^{(i)T} \hat{C}^{(i,b)} \hat{I}_b \hat{C}^{(i,b)T} \dot{\bar{\omega}}_q^{(i)} \tag{3.23}$$

$$U = m_b g \bar{u}_3^t \bar{p}_q^{(i)} \tag{3.24}$$

Then, the equation of the motion of the system is obtained by plugging equation (3.23) and equation (3.24) into equation (3.22).

$$\hat{M}(\bar{q})\ddot{\bar{q}} + \hat{C}(\bar{q}, \dot{\bar{q}})\dot{\bar{q}} + \hat{G}(\bar{q}) = \bar{u} + \bar{u}_{ext} \tag{3.25}$$

$$K = \frac{1}{2}\dot{\bar{q}}^T \hat{M}(\bar{q})\dot{\bar{q}} \tag{3.26}$$

$$\hat{M}(\bar{q}) = \hat{V}_q^T m_b \hat{V}_q \hat{\Omega}_q^T \hat{C}^{(i,b)T} \hat{\Omega}_q \tag{3.27}$$

$$c_{a,b} = \sum_{j=1}^{6} \frac{1}{2} \left\{ \frac{\partial m_{a,b}}{\partial q_j} + \frac{\partial m_{a,j}}{\partial q_b} - \frac{\partial m_{j,b}}{\partial q_a} \right\} \tag{3.28}$$

$$\hat{G}(\bar{q}) = \frac{\partial U}{\partial \bar{q}} \tag{3.29}$$

The navigation algorithms that will be verified in simulation environment use "Hector Quadrotor" package and this package has its own dynamic equations. Flight dynamics, motor dynamics and thrust calculation are given in the following lines.

Meyer et al. [27] who prepared the Hector Quadrotor package used the following formulas for the calculations of flight dynamics,

$$\dot{p^n} = v^n \tag{3.30}$$

$$\dot{v^n} = m^{-1} C_n^b \mathbf{F} \tag{3.31}$$

$$\dot{w^b} = J^{-1} \mathbf{M} \tag{3.32}$$

In the equations, all the loads acting on the platform are indicated by **F** and all torques are indicated by **M**, while the relevant position and speed information calculated based on the world reference center is indicated by $\dot{p^n}$ and $\dot{v^n}$. C_n^b is the transformation matrix that transforms the robot reference system to the world reference system. $\dot{w^b}$ indicates the angular velocity in the robot reference system.

The force vector **F** includes the motor thrust $\mathbf{F}_M$, the drag forces $\mathbf{F}_d$ and the gravity vector $\mathbf{F}_g$. Similarly, the torque vector contains the propulsion torque $\mathbf{M}_M$ and drag moments $\mathbf{M}_d$. Moments and drag forces are obtained as,

$$\mathbf{F}_d = -\mathbf{C}_{d,F} \cdot \mathbf{C}_n^b \cdot |v^n - v_w^n| \cdot (v - v_w) \tag{3.33}$$

$$\mathbf{M}_d = \mathbf{C}_{d,M} \cdot \left|\omega^b\right| \cdot \omega^b \tag{3.34}$$

In equations 3.33 and 3.34 $\mathbf{C}_{d,F}$ and $\mathbf{C}_{d,M}$ represent the diagonal drag coefficient matrices while the wind vector is shown as v_w^n . Lastly, the gravity force $\mathbf{F}_g$ is given as,

$$\mathbf{F}_g = m \cdot \mathbf{C}_n^b \cdot \begin{bmatrix} 0 & 0 & g_e \end{bmatrix}^T \tag{3.35}$$

Motor dynamics behaviour that is based on four brushless DC motor is expressed with the following formulas,

$$U_A = R_A I_A + \psi \omega_{\mathbf{m}} \tag{3.36}$$

$$M_e = \psi I_A \tag{3.37}$$

$$\dot{\omega}_M = \frac{1}{J_M} \cdot (M_e - M_m) = \frac{1}{J_M} \left(\frac{\psi}{R_A} \cdot (U_A - \psi \omega_M) - M_m \right) \tag{3.38}$$

$$T = C_{(T,0)} \omega_M^2 + C_{(T,1)} v_1 \omega_M \pm C_{(T,2)} v_1^2 \tag{3.39}$$

$$C_T(J) = C_{(T,0)} + C_{(T,1)} J \pm C_{(T,2)} J^2 \tag{3.40}$$

3.3 State Space Representation

The vector of state can be congregated as,

$$X = \begin{bmatrix} \phi & \theta & \psi & x & y & z \end{bmatrix}^T \in \Re^6 \tag{3.41}$$

Then, the equations of the kinematics of the quadrotor can be written in state space representation form as follows,

$$\begin{cases} \dot{\phi} = p + r[c(\phi)t(\theta)] + q[s(\phi)t(\theta)] \\ \dot{\theta} = q[c(\phi)] - r[s(\phi)] \\ \dot{\psi} = r\frac{c(\phi)}{c(\theta)} + q\frac{s(\phi)}{c(\theta)} \\ \dot{x} = \omega[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] - v[c(\phi)s(\psi) - c(\psi)s(\phi)s(\theta)] + u[c(\psi)c(\theta)] \\ \dot{y} = v[c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)] - \omega[c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta) + u[c(\theta)s(\psi)]] \\ \dot{z} = \omega[c(\phi)c(\theta)] - u[s(\theta)] + v[c(\theta)s(\phi)] \end{cases} \tag{3.42}$$

where $s(\theta) = sin(\theta)$, $c(\theta) = cos(\theta)$, $s(\phi) = sin(\phi)$, $c(\phi) = cos(\phi)$, $s(\psi) = sin(\psi)$, $c(\psi) = cos(\psi)$, $t(\theta) = tan(\theta)$

3.4 Localization and Mapping

The most important and challenging part of this study is to obtain accurate location and map information. The robot processes the data that is received via inertial measurement device and sensors into a huge matrix. The matrix detects information about the location by assigning a value based on the density of the object. This process is repeated until the entire map is obtained and it is verified continuously.

3.4.1 Ultra Wide-Band Localization

Ultra wide-band (UWB) technology is an innovative wireless active marker sensor system that can be used for localization in the indoor environment. It makes localization so easy for the UAV to be able to determine its position, especially at the first moment when it begins the mission. Afterwards, the reliability of the map is increased by cross-checking with the data from other sensors. Compared to other wireless systems, UWB technology stands out because of its low energy consumption and the ability to work independently of the effects of surrounding objects. However, in order to use the UWB technology in positioning, at least three UWB transmitters must

be placed in the area before mapping starts. This is one of the most important factors that make it difficult to use in search and rescue operations. Yet, before the SLAM, the UAV can accurately place the transmitters on the area and record these locations for later use. In this study, it is assumed that the sensors are already placed in the field. The operating logic of the system is shown in Figure 3.3 and trilateration technique is used to calculate the position of robot according to the information gathered from UWB transmitters as shown in Figure 3.2. This technique localize the robot using circle equations 3.43.

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = {R_i}^2 \text{ , } i = 1, 2, 3 \tag{3.43}$$

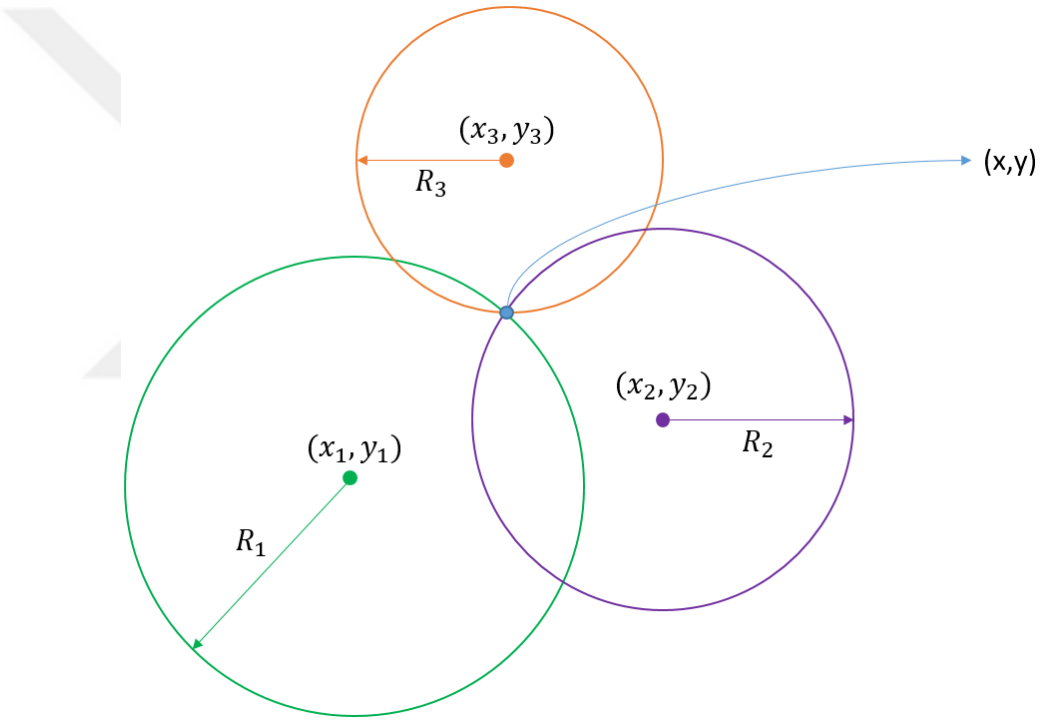


Figure 3.2: UWB Communication with Trilateration Method

3.4.2 SLAM

SLAM (Simultaneous Localization and Mapping) is abbreviation given to simultaneous map extraction and robot positioning. The difficulty of this process can be easily understood from the definition. A map is required for correct positioning, while an

accurate positioning is required for mapping. SLAM is one of the major problems that must be solved in fully autonomous robots. Autonomous robots are classified according to their operating areas like indoor robots, outdoor robots, underwater robots and aerial robots. Therefore, SLAM methods differ as in this classification. Many methods have been developed since the mid-80s. The SLAM process can be simply described as shown in Figure 3.3.

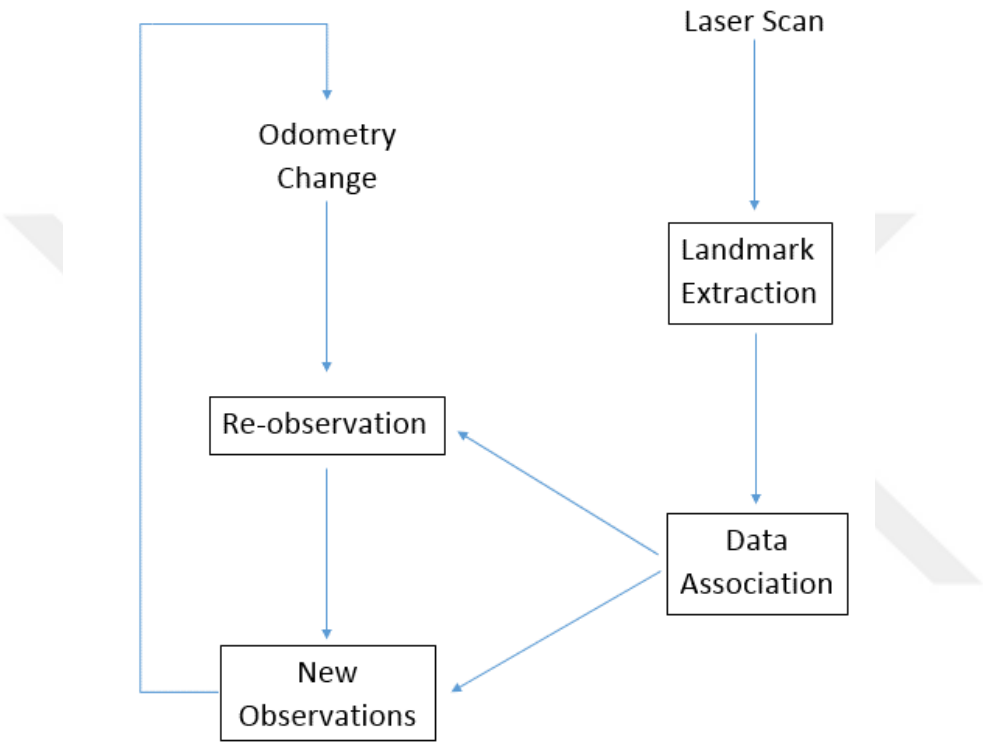


Figure 3.3: SLAM Algorithm Flow Chart [19]

3.4.3 Navigation Algorithms

In this section, navigation algorithms are introduced. Working principles and the logic behind the scene are explained. Although ROS allows coding in many programming languages, all algorithms are written in C++ to take an advantage of QT Creator opportunities.

3.4.3.1 Wall Following Algorithm

Wall following algorithm is a simple but highly successful method used for exploring and mapping of maze-like indoor environments[12]. An example of wall following algorithm is presented in related study [24]. There are two different versions where the robot moves to the right or left of the wall. In this study, the UAV covers the wall on the left side by controlling its orientation and speed. The decisions that the robot will make according to the situations it will meet are shown in Table 3.1 and pseudocode that belongs to wall follow algorithm is expressed in Algorithm 1,

Table 3.1: Wall Following Algorithm Decision Table[12]

Left	**Right**	**Front**	**Action**
0	X	X	Turn counter clockwise direction
1	0	X	Go straight
1	1	X	Turn clockwise direction

In table 3.1, "0" indicates that no wall is detected while "1" represents that wall is detected. "X" means that it is not considered according the information getting from other sides.

Algorithm 1: Pseudocode for Wall Following Algorithm

```
while Exploration is not finished do
    if Wall is not spotted ahead of the UAV then
        Adjust the turning velocity;
        Adjust the forward velocity;
        Adjust the altitude velocity;
    else
        Turn 90 degrees right
    end
end
```

Elements used in this algorithm is shown in Figure 3.4. Laser range finder gives information about ranges for various directions around the UAV, $Range_1$ and $Range_2$ denote the ranges at 10 ° to the left of the quadrotor and $Range_3$ denotes the range

in front of the quadrotor. If $Range_3$ is larger than a certain threshold, quadrotor continues to go forward with constant velocity, while yaw velocity is continuously adjusted as follows:

$$v_{yaw} = K_{yaw}(Range_2 - Range_1) \tag{3.44}$$

where K_{yaw} is the gain to determine turning velocity. Altitude velocity is adjusted so that quadrotor stays at a constant altitude throughout the task(1 meter in this case). When $Range_3$ is below a certain threshold, 1.5 m is used in simulations, i.e. a vertical wall is reached, UAV turns 90 ° and continues to follow the wall to the left of it as explained above. Translation velocity of the quadrotor is simply found by adding forward velocity and avoidance velocity.

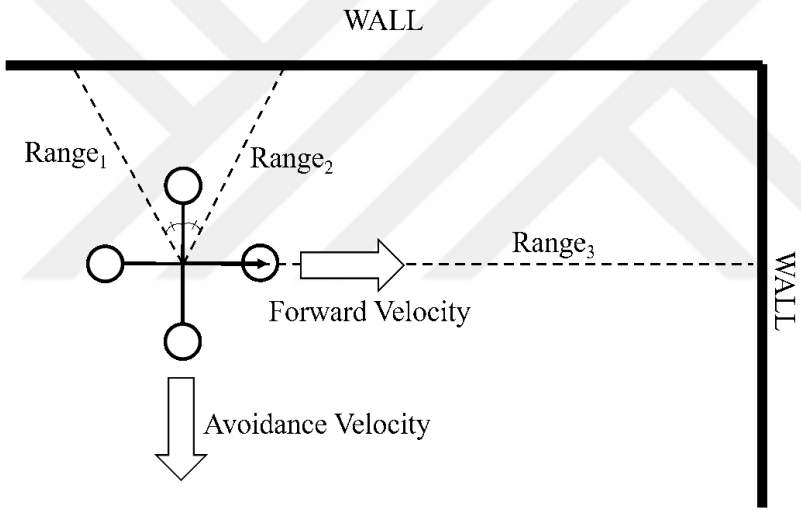


Figure 3.4: Operating Logic of Wall Following Algorithm

3.4.3.2 Exploration Algorithm

The purpose of this algorithm is to explore each part of a closed area by avoiding any collision and by going the least. The speed contribution required for each of two sub-tasks is calculated. The total velocity command for the UAV is obtained by the vectoral addition of velocity contributions from the opening detection and obstacle avoidance algorithms. This algorithm is proposed in related paper[24].

Opening Detection Algorithm It is important to identify the openings and select the destination accordingly to explore each part of a region. The opening points are obtained by comparing the proportions of distances obtained from two consecutive angles over the entire scanning area. The distance of the opening from the starting point is found in Equation 3.45,

$$r_{corner} = max \frac{r_i}{r_{i+1}}, i = 1, 2, 3... \tag{3.45}$$

Since the UAV's orientation, estimated position, UWB data, distance to the corner point and angle information are known, the position of the corner point can be obtained using Equation 3.46 and Equation 3.47. It refers to the angle of deviation in these equations and the angle between the UAV and the corner point in the deviation correction.

$$x_{corner} = x_{UAV} + r_{corner} \cos(\psi + \theta) \tag{3.46}$$

$$y_{corner} = y_{UAV} + r_{corner} \cos(\psi + \theta) \tag{3.47}$$

Once the corner point has been determined, a destination must be selected. The distance to the target is selected as the average of the distances between the left side and right side of the corner. The angle between the target and the UAV is selected as 10 ° away form the corner compared with the the angle between the corner point and the UAV. In most cases, a target point is assigned in the middle of the open area and it leads to extract more space to explore.

While the UAV's orientation is always towards the target, its forward speed is adjusted according to its distance from the target as stated in the equation 3.48.

$$v_{forward} = K_0 + K_p \sqrt[3]{l} \tag{3.48}$$

Obstacle Avoidance Algorithm The speed contribution required to cross obstacles safely and move around walls without impact is calculated using the information

obtained from the laser range finder at each angle. Using laser scans at each angle, velocity vectors are generated such that the magnitude is inversely proportional to and opposite to the distance as shown in Equation 3.49.

$$v_i = \frac{K}{r_i}\hat{r_i} \tag{3.49}$$

By summing obstacle avoidance velocity contributions from each laser scan and the forward velocity contribution to the target calculated by the opening detection algorithm, the UAV can move to the target smoothly without any collision.

3.4.3.3 Target-Based Navigation Algorithm

This algorithm is based on "Frontier Based Probabilistic Approach" principle [17]. This method aims to explore map of the environment by moving the border between discovered and undiscovered areas. During the process, the robot does not need to large-scale map for navigation. Therefore, it reduces the computational burden.

The algorithm is written in C++ programming language by following the open-closed and generic programming principles. Open-closed principle states that software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification. On the other hand, generic programming allows to write common functions or types to reduce duplication. It emphasizes the importance of the reusability criterion.

This algorithm includes three nodes that are named Mission Controller, Flight Controller and Navigation Controller. Features and structures of the nodes are expressed with pseudocodes below.

Mission Controller

Description: Mission Controller may hold different tasks although it is currently performing a single task that is called "Search and Rescue".

Flight Controller

Description: It is a node for managing the messages that are send to "cmd vel" topic. FlightController first enables the motors and ascends to the "OperatingAltitude". Then, it waits for destination commands from Navigation Controller. If a new destination is given, it corrects heading of the quadrotor first, and then flies to the destination. All of the velocity commands are controlled by a PID controller. When it reaches the destination, maintains the Quadrotor in hovering position until a new destination command is received from the Navigation Controller. The details of the PID is given in section 3.5.

Important Parameters:

Important parameters used in Flight Controller algorithm are listed as follows,

- OperatingAltitude = 0.5; (This is the operating altitude of the quadrotor in meters)
- PIDGain.Kp = 0.1; PID control, proportional component coefficient
- PIDGain.Ki = 0.0005; PID control, integral component coefficient
- PIDGain.Kd = 0.00005; PID control, derivative component coefficient
- ThresholdParameters: The threshold parameters are the tolerances that the quadrotor is accepted to be at the speficied position or angle

Main States:

Main states of the Flight Controller algorithm are listed as follows,

- GroundAtRest
- AscendingToFlightAltitude
- CorrectingHeading
- FlyingToDestination
- Hovering

Pseudocode:

The pseudocode of the "FlightController" node is shown in Algorithm 2 and Algorithm 3 ,

Algorithm 2: Pseudocode for Flight Controller Algorithm Main Function

```
FlightState = GroundAtRest; // At the start;
EnableMotors(); // Start the motors of the quadrotor;
while Quadrotor.Altitude < Operating Altitude do
    FlightState = AscendingToFlightAltitude;
    Ascend in positive z direction with PID control;
    if Quadrotor is within the threshold of the operating altitude then
        FlightState = Hovering;
    end
end
while FlightState == Hovering do
    CalculateZOutput(); // Maintain altitude by calculating velocity in z;
    PublishVelocity(); // Publish the command to cmd_vel topic;
    if A new destination is given then
        FlightState = CorrectingHeading;
        PublishDestinationNotReached();
    end
end
while FlightState == CorrectingHeading do
    CalculateZOutput(); // Maintain altitude with PID control;
    CalculateYawOutput(); // Calculates the yaw velocity with PID;
    PublishVelocity(); // Publish the calculated command to cmd_vel topic;
    if Quadrotor heading is correct (within threshold) then
        FlightState = FlyingToTheDestination;
    end
end
```

Algorithm 3: Pseudocode for Flight Controller Algorithm Main Function Cont'd

```
while FlightState == FlyingToDestination do
    CalculateZOutput(); // Maintain altitude;
    CalculateVelocityOutput(); // Calculates the speed command in x-axis;
    PublishVelocity(); // Publish the calculated command to cmd_vel topic;
    if Quadrotor is in the vicinity of the destination point then
        FlightState = Hovering; // Publish this to the Navigation Controller;
        PublishDestinationReached();
    end
end
```

Navigation Controller

Description: It decides on the next destination to fly looking at the "map" topic data published by hector mapping and using unique algorithm to find open spaces and add those points to the ToBeVisitedList. It uses A* path finding algorithm if it anticipates that next destination is not reachable by direct flight. It subscribes to FlightController and monitors if it reaches the destination, and gives new destination points until there are no points left to be visited.

Important Parameters:

Important parameters used in Navigation Controller algorithm are listed as follows,

- double mapResolution = 0.5; //Map resolution in meters. It must be same as in mapping-default-omer launch file
- double mapSize = 256; //Map grid count (i.e map is mapSize x mapSize). It must be same as in mapping-default-omer launch file.
- int clearanceForFlight = 2; //To be able to understand that a point is clear to be moved onto, system checks if the number of map grids around that point is

not occupied. //If the mapSize is larger and resolution is smaller, the clearances must be specified accordingly.(must be increased)

- int clearanceForOpenSpace = 2; //When a point is processed as a potential discovery point candidate, this specifies the number of map grids around that must not be occupied. // a=1 means 3x3 grid, a=2 means 4x4 grid and goes like this.
- double minDiscoverDistance = 4; //Min discovery distance in meters. The new discovery point must be far away from other discovery points for this much.

Main States:

There are two main states in the Navigation Controller :

- DestNotReached
- DestReached

Pseudocode:

The pseudocode of the main function is shown in Algorithm 4.

The pseudocode of the FindOpenSpaces function is shown in Algorithm 5.

The pseudocode of the SelectNextDestination function is defined in Algorithm 6.

The pseudocode of the SelectNextDestinationByAStar() function is defined in Algorithm 7.

The pseudocode of the isFlyable function is shown in Algorithm 8.

A* Search Algorithm

A* search algorithm calculates the cost of each adjacent node by using heuristic evaluation function as shown in equation 3.50. After all the calculations are done, it

Algorithm 4: Pseudocode for Navigation Controller Algorithm Main Function

```
prevNavStatus = DestNotReached;
NavStatus = DestNotReached;
while True do
    if NavStatus == DestReached && prevNavStatus == DestNotReached
     then
        Add current destination to AlreadyVisitedPoints;
        FindOpenSpaces(); //Reexamine the map;
        if PointsToBeVisitedList.empty() then
            Print a Message that all of the map is visited;
            return 0;
        else
            SelectNextDestination();
        end
    end
end
```

creates a suitable path by combining the nodes with minimum cost.

$$F(n) = G(n) + H(n) \tag{3.50}$$

In equation 3.50, n is the previous node of the path, G(n) is the cost of the path from the start node to adjacent node, and H(n) is heuristic that estimates the cost of the cheapest path from n to the target node. In our case, H(n) equals to air distance between corresponding node and target node.

As the map is divided into grids, the robot moves through those grids just like in chess game. A representative movement of the robot is shown in Figure 3.5.

Algorithm 5: Pseudocode for Navigation Controller Algorithm FindOpenSpaces Function

```
// currentMap is the latest map published by hector mapping;
For each Grid Point on currentMap;
CheckOccupancy(); // Occupancy Value must be 0 for the point;
CheckClearance(); // Occupancy Value must be 0 for the neighbour points
  specified by clearanceForOpenSpace parameter;
CheckAlreadyVisitiedList(); // Confirm that the point is far away from every
  point on the already visited points list by minDiscoverDistance parameter;
CheckToBeVisitiedList(); // Confirm that the point is far away from every
  point on the to be visited points list by minDiscoverDistance parameter;
if Point meets criteria listed above then
    Add the point to PointsToBeVisited list;
end
```

3.5 Control Method

3.5.1 PID Control

Hector Quadrotor cascaded PID controller scheme is shown in Figure 3.6. Controller of the Hector Quadrotor is implemented as a set of cascaded PID controllers. Controlling the vertical velocity, attitude and yaw rate occurs in the inner loop while controlling the horizontal velocity, altitude and heading runs in the outer loop. Each axis and the altitude can be controlled independently with this system. The output of the inner loop are managed vertical thrust and torques.

In this work, PID control is executed to determine desired yaw rate, horizontal and vertical velocities of the Quadrotor. These outputs are published to ROS so that the controller of Hector Quadrotor uses them as inputs.(see Figure 3.6).

$$v_{x,d} = K_P e_x + K_I \int e_x + K_D \frac{de_x}{dt} \tag{3.51}$$

Algorithm 6: Pseudocode for Navigation Controller Algorithm Select-NextDestination Function

```
NearestPoint = FindTheNearestPointToBeVisited(); // Gets the nearest point
  in the PointsToBeVisitedList;
if isFlyable(CurrentPoint,NearestPoint) then
    NextDestination = NearestPoint; // If the nearest point is directly flyable,
      then it is the next destnation;
    publish(NextDestination); // Publish this to the FlightController;
else
    SelectNextDestinationByAStar(); // If the nearest point is not directly
      flyable, let A* algorithm decide the next destination point;
end
```

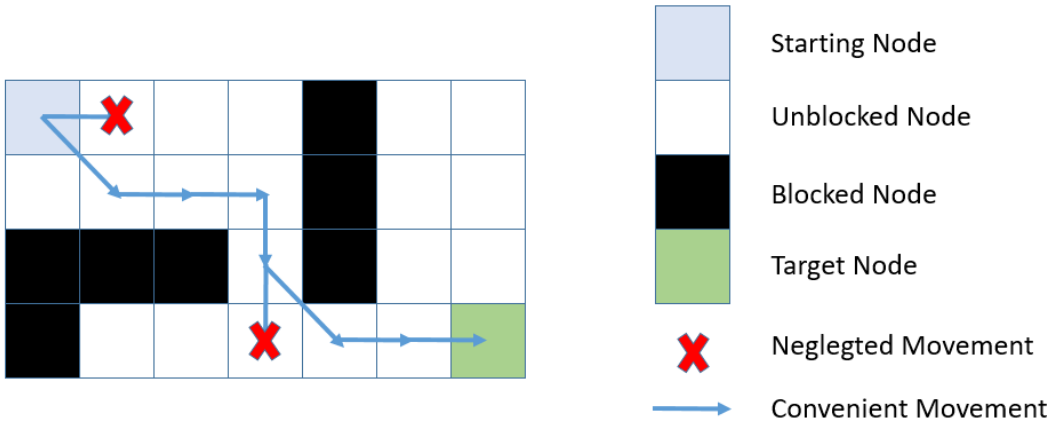


Figure 3.5: Sample Navigation Path with A* Search Algorithm

$$v_{z,d} = K_P e_z + K_I \int e_z + K_D \frac{de_z}{dt} \tag{3.52}$$

$$\omega_{z,d} = K_P e_\psi + K_I \int e_\psi + K_D \frac{de_\psi}{dt} \tag{3.53}$$

Where, $v_{x,d}$ is the desired velocity in the heading direction, $v_{z,d}$ is the desired velocity in the altitude direction, $\omega_{z,d}$ is the desired yaw velocity, e_x is the position error in the heading direction, e_z is the position error from the desired altitude in the altitude direction, e_ψ is the error of angular deviation from the desired attitude angle. Also,

Algorithm 7: Pseudocode for Navigation Controller Algorithm SelectNextDestinationByAStar Function

```
NearestPoint = FindTheNearestPointByAStar(); // Gets the least cost path
  by A* as the nearest point amongst the points listed in the
  PointsToBeVisited list;
NavPath = CreateAStarPath(); // Gives the next destination from a
  navigation path consisting of points found by the A* algorithm for the next
  couple of turns. // CreateAStarPath() draws all the paths and then decide
  the shortest one that includes minimum point;
while NavPath.isNotEmpty() do
    NavPath.erase(); // Erase already given point;
    NextDestination = NavPath.next // Give the next point from the
      navigation path constructed by A* as the new destination point;
    // Note : For simplicity some of the details of the A* path selection is
      not shown. For example, while navigating through the path created by
      A*, if nav controller detects that the final destination is flyable it
      overrides the rest of the path and flies directly;
end
```

K_P, K_I and K_D are the proportional, integral and derivative gains of the PID, respectively. Publishing these velocities with $v_{y,d} = 0$ into the Hector Quadrotor's cmd_vel topic, position control of the quadrotor will be managed.

Algorithm 8: Pseudocode for Navigation Controller Algorithm isFlyable Function

ConstructALine(); // Construct a line from the start point to the target point For Selected Points On The Line; // 1 meter intervals CheckOccupancy(); // Check that the grid location on the "map" corresponding to this point is not occupied CheckClearance(); // Check that the vicinity of the point specified by clearanceForFlight parameter that is also not occupied // Since the quadrotor is bigger than grid size, it needs to check the neighbour of the related grid.

if *Above Criteria Holds* **then**
    return true;
else
    return false;
end

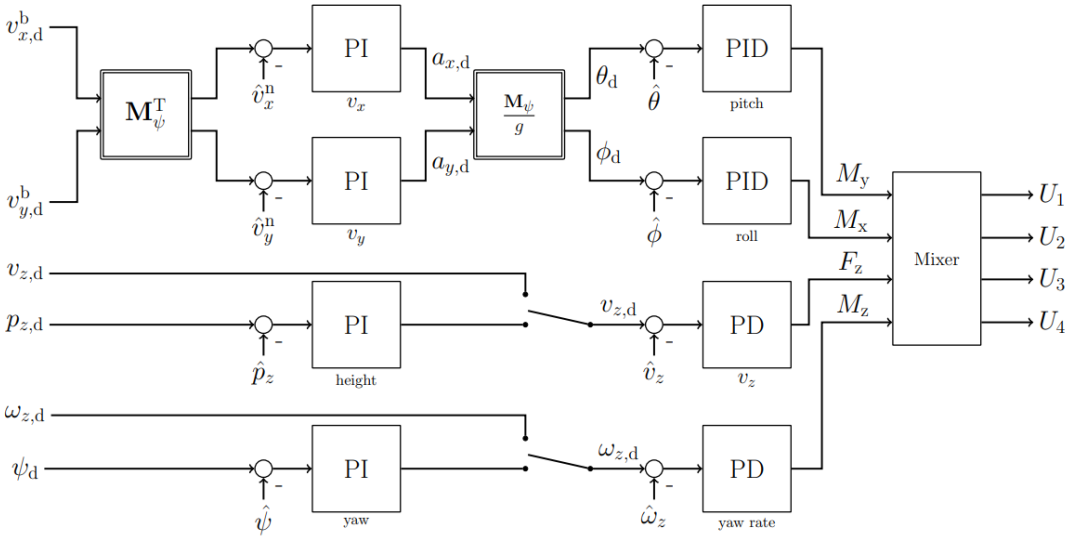


Figure 3.6: Controller of the Hector Quadrotor with Separate Cascaded PID Controllers[27]

CHAPTER 4

EXPERIMENTS

4.1 Experimental Setup

In this section, software and hardware parts of the overall system are described. Versions of software packages are shared as well to avoid compatibility issues.

4.1.1 Software Packages

In this section, the software used during the study will be mentioned. Ubuntu 16.04 LTS(Xenial Xerus), which is a simple, useful and one of open source Linux distributions, is used as operating system. The LTS (Long Term Support) version is released every two years and lasts five years. Moreover, it provides a version that errors in previous versions are corrected and the problems that may arise are solved quickly. Therefore, the LTS version is particularly selected to work smoothly. ROS (Robot Operating System) Kinetic Kame distribution is chosen as a framework because it is recommended with Ubuntu 16.04 LTS(Xenial Xerus). Similar to Linux distributions such as Ubuntu, Kubuntu, Lubuntu and so on, a ROS distribution is a group of ROS packages. The purpose of the ROS distributions is to allow developers to work with a relatively stable code base until the next release is available. Rviz (ROS Visualization Tool) comes with a suitable version according to the ROS distribution. Lastly, Gazebo 7.1 is used as a 3D simulation environment. In spite of the fact that it may seem trivial, time to time the usage different versions of the same software may cause compatibility problems in the world of open source.

4.1.1.1 Ubuntu 16.04-LTS (Xenial Xerus)

Ubuntu is a Linux-based open source software operating system. It has been the focus of interest of developers worldwide since it is simple, fast and secure.

4.1.1.2 Robot Operating System (ROS) - Kinetic Kame

ROS which stands for "Robot Operating System" is an open source software for controlling robots and robot components contrary to first connotation. It provides the features involved low-level device control, hardware abstraction, message-passing among processes, and management of packages. While it was only working on Linux-based operating systems, it has recently become a part of the MATLAB - Robotics System Toolbox and available on other platforms. The Kinetic Kame version of ROS is primarily published for 16.04 (Xenial Xerus) in Ubuntu. There are lots of ROS distribution even though ROS Kinetic Kame was used in this study. (see Figure 4.1)

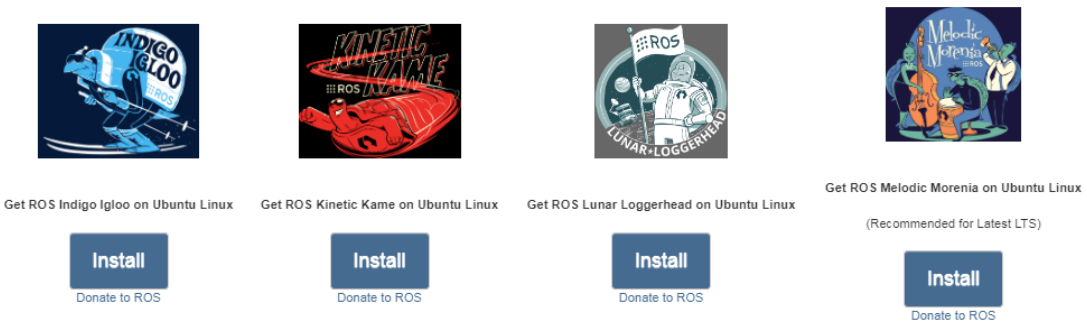


Figure 4.1: Ros Disributions [2]

ROS, that works with the logic of subscribing and publishing, is enabling communication among nodes thanks to rosmaster network based server. It is preferred by researchers as it contains many libraries and it has a structure that can be integrated with the most common programming languages. In addition, it allows developers to use the study of different researchers in their project. Furthermore, it is possible to integrate ROS with realtime code, though ROS is not a realtime framework.

4.1.1.3 ROS Visualization Tool (Rviz)

Rviz is a ROS graphical interface that allows developers to visualize the data gathered by using plugins for many available topics. A robot's map and the traces left behind it can be observed, as well as a variety of commands can also be given.

4.1.1.4 Gazebo 7.1

Gazebo, a three-dimensional simulation tool, is a software that can show the physical interaction between objects while simulating robots and objects. In this Linux-based system, the rendering task is undertaken by the open source graphics engine OGRE. Many libraries such as physics, rendering and sensors are located in Gazebo. These libraries are used by the server (gzserver) that generates the sensor data and runs the physical loops, and the client (gzclient), which enables visualization with user interaction. It is the choice of researchers because it provides convenience in robot design for real environments thanks to the sensors with noise added in it. Gazebo version 7.1 is used in this study. Gazebo installation, integration with ROS, various examples of ground and air robots are described in detail in the relevant document [35]. Gazebo software architecture, communications and connections realized in it are shared in Figure 4.2.

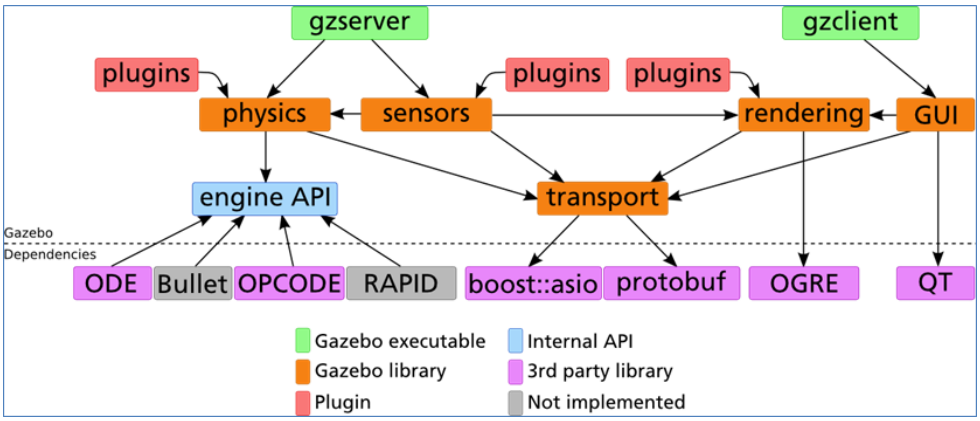


Figure 4.2: Gazebo Software Architecture [1]

4.1.1.5 QT Creator 4.8

QT is the cross-platform C++, Java script and QML integrated development environment(IDE). IDEs increase programmer efficiency by centralizing common activities of writing software into a single application: editing source code, building executables, and debugging. In this study, QT Creator is preferred due to its ability to create ROS workspace. It means that QT constitutes all required folders and files automatically to initiate programming in ROS. Moreover, it includes the ROS terminal that allows the programmers to make and build operations. QT-ROS plug-in should be installed into QT Creator base to benefit opportunities mentioned above. Graphically user interface of QT Creator is shown in Figure 4.3,

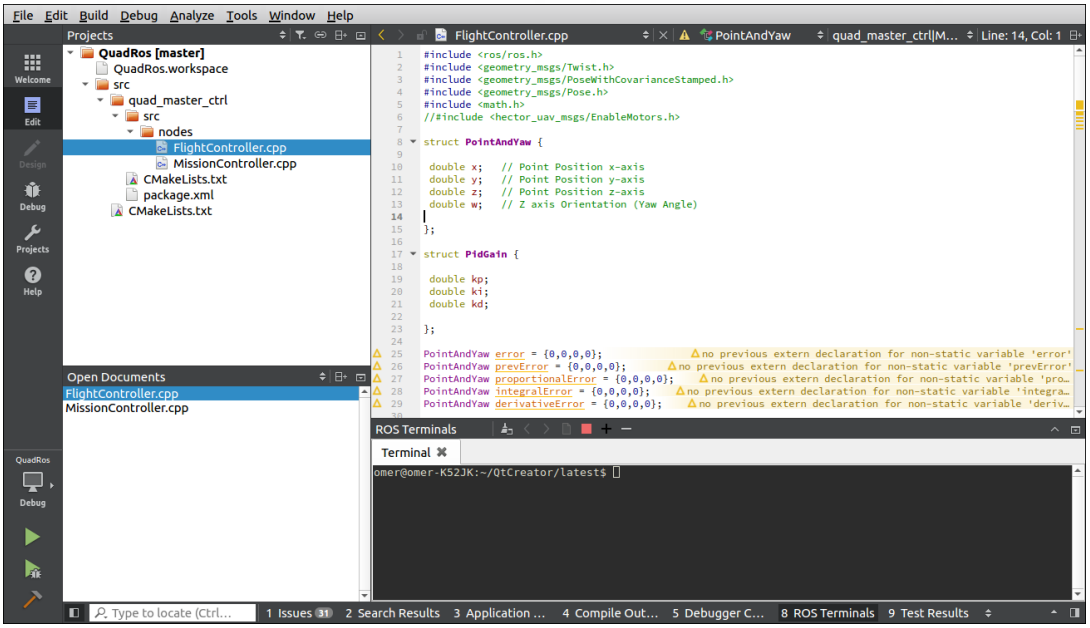


Figure 4.3: QT Creator User Interface

4.1.2 Quadrotor

The selected platform for using in the three-dimensional simulation program is defined as in the special format named URDF (Universal Robot Description Format) that exists in "hector-quadrotor-urdf" ROS package. This detailed model has the image information of the robot limbs and joints as well as their position and orientation. In addition, thanks to the collision zone definition of robot, various algorithms can be

tested on Gazebo. The Hector Quadrotor model, which is included in the package and shown with the integrated laser range finder in Figure 4.4, can get ready for operation by adding the necessary sensors.

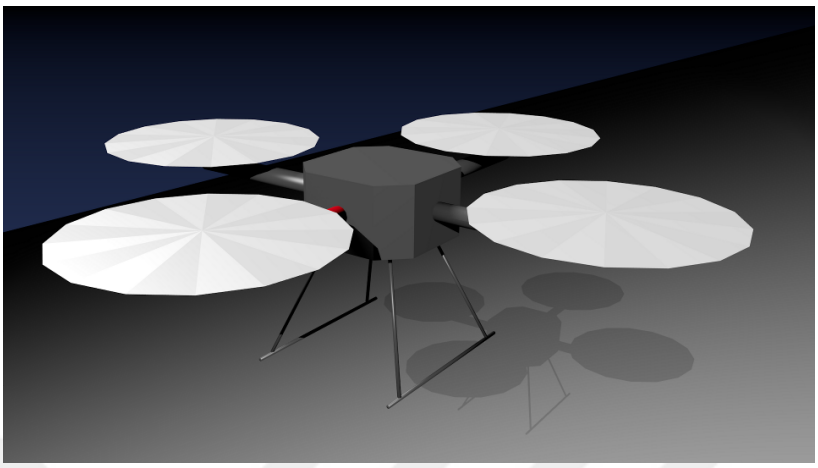


Figure 4.4: Hector Quadrotor [27]

"Hector SLAM" package, which is developed especially for indoor operations, is used in this study. There are many different packages available under this package such as "Hector Mapping, Hector IMU Tools and Hector Nav Msgs" for simultaneous localization and mapping.

4.1.2.1 Laser Imaging Detection and Ranging(LIDAR)

Laser range finder device "Hokuyo UTM-30LX LIDAR" is selected and mounted under the platform. The laser range finder has a scanning speed in the 40 Hz band and field of vision of 270. Detailed information is available at the relevant address [20].

4.1.2.2 Sound Navigation and Ranging(Sonar)

Sonic sensors emit high-frequency sound waves that people cannot hear and measure distance by calculating the time between these waves hitting and returning. It is suitable for measurements up to 30 meters. In this study, it is aimed to control the height of the vehicle from the ground by placing it under the aircraft.

4.1.2.3 Inertial Measurement Unit(IMU)

The inertial measurement device (IMU) is the most important sensor for stable control and movement of the aircraft. It is responsible for measuring the angular velocity and acceleration of the aircraft according to the world reference system. This electronic unit can measure the acceleration in 3 axes by means of the accelerometer in it and measure the rotational force in 3 axes thanks to the gyroscope. Since it is so difficult to calculate the acceleration of the platform according to the world reference system without knowing the position and direction of the air vehicle, it is an indispensable part of the unmanned aerial vehicles and many spacecrafts.

4.1.2.4 Ultra Wide-Band Sensor(UWB Sensor)

Ultra wide-band is a radio transmission system used for precise positioning, data acquisition and monitoring. It allows more data exchange over a period of time compared to traditional technologies. The UWB with low power spectral density can use other frequency spectra allocated for communications without interference. It is capable of detecting 10 cm accuracy for reconnaissance robots even at a very high speed of 20 km / h. Three transmitters, that is located in approximately the known positions, are available in the indoor environment as active markers, while one receiver is located on the UAV for position detection.

In this study, a new plugin named "Gazebosensorplugin" that is developed by Barral et al. [8] was used. This plugin contains many features that can be used in Gazebo Simulator. Even though it was written for Gazebo 9 version, all functions and definitions were converted to Gazebo 7 version in order to avoid compatibility issues. They implemented UWB localization technology to forklift trucks in their study. UWB system can be established by putting anchors on walls and mounting tag to vehicle. It is applied to hector quadrotor in a similar way. Calculations for communication between anchors and tag is provided by using trilateration technique. A node is written for this technique and it is aimed to localize robot's position using at least three anchor. In this node, position data is calculated by using circle equations and assigning a weight proportional to distance to related anchor in order to gather reliable

information.

4.2 Experimental Procedure

In 3D simulation environment, two maps are designed in different complexity. One of them is designed by considering the future studies that are performed in test environment. On these maps, various navigation and mapping algorithms are experienced. The map 1, that can be constructed in physical environment easily, is introduced in related paper [31] first. Also, the same map is equipped with obstacles to to test the behavior of algorithms against objects. In order to check the effect of external geometry on algorithms, different obstacles are chosen. Corresponding maps are presented in Figures 4.5 and 4.6.

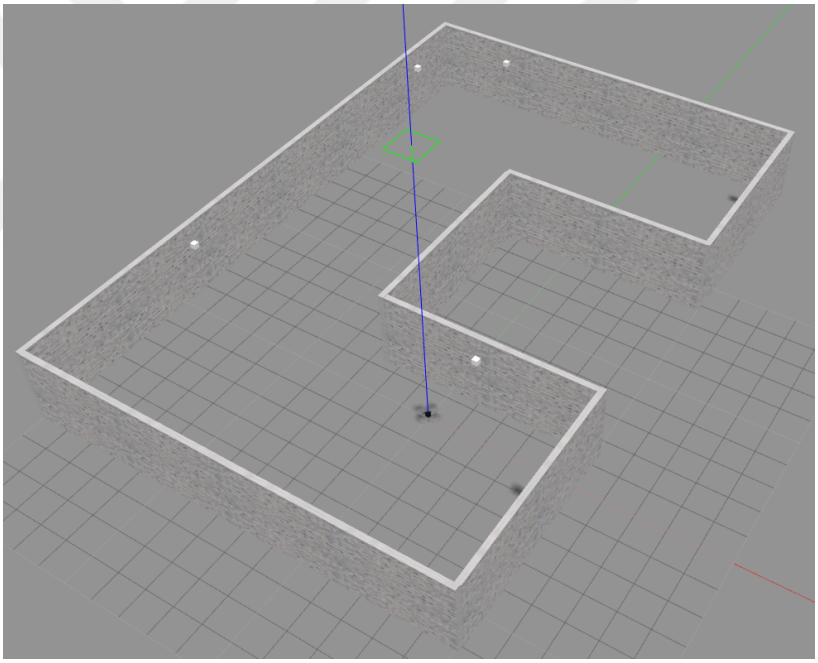


Figure 4.5: Map 1 without Obstacles [31]

Similarly, another challenging map is design to evaluate performance of the algorithms. As it is three times bigger than the other map, the course becomes difficult. Obstacles are added to the same map to increase the level of difficulty one more step. Standing man, cube, barrier are some of these obstacles. Corresponding maps are shown in Figures 4.7 and 4.8.

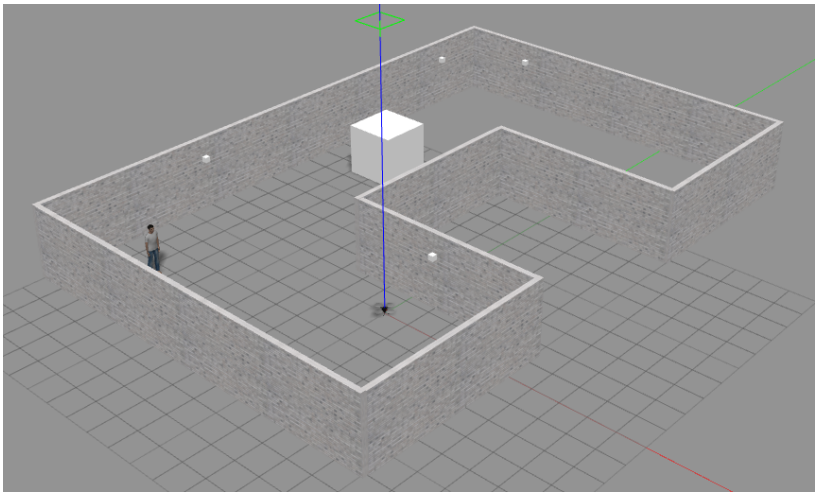


Figure 4.6: Map 1 with Obstacles [31]

Simulations are performed on two maps. All three algorithms are tested on map 1. However, exploration algorithm is failed on map 2 since it is not smart enough to carry out navigation and mapping on huge maps with obstacles. Therefore, only two algorithms can be tested on map 2. In addition, quadrotor is released on the map from different positions in order to check the repeatability performance of the algorithms. In map 1, three initial points are determined as shown in Figure 4.9. On the other hand, two starting points are assigned on map 2 as shown in Figure 4.10.

4.2.1 Simulations

4.2.2 Performance Metrics

All described algorithms perform the same task that is to obtain map of a unknown indoor environment without hitting any obstacle. However, it is required to determine some criteria in order to measure the effectiveness of algorithms in different scenarios. In this section, the performance criteria in which the algorithms are superior to each other are determined and expressed.

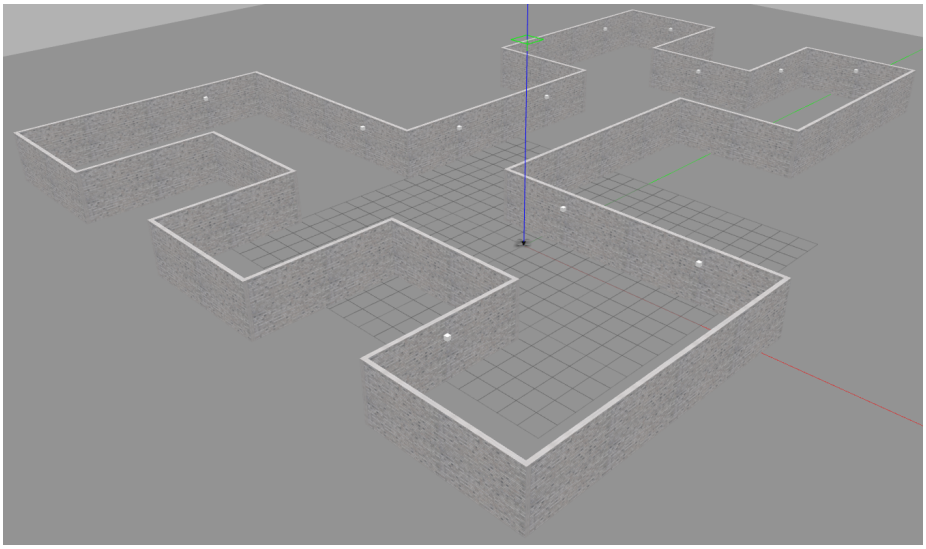


Figure 4.7: Map 2 without Obstacles

4.2.2.1 Distance

The distance travelled by the robot can be compared among algorithms and scenarios. The smaller the distance value, the more successful the algorithm.

d represents the distance of the robot in meters.

4.2.2.2 Effective Distance

Using distance metric that is defined before, a new metric can be generated by taking into account for the area of the map.

$$P_d = \frac{d[m]}{A[m^2]} = \frac{1}{[m]} \tag{4.1}$$

d represents the distance of the robot in meters as it is mentioned in previous part. A specifies the size of the area to be discovered in m^2. P_d refers to the performance criterion that results from the ratio of the robot's path to the total area of the map.

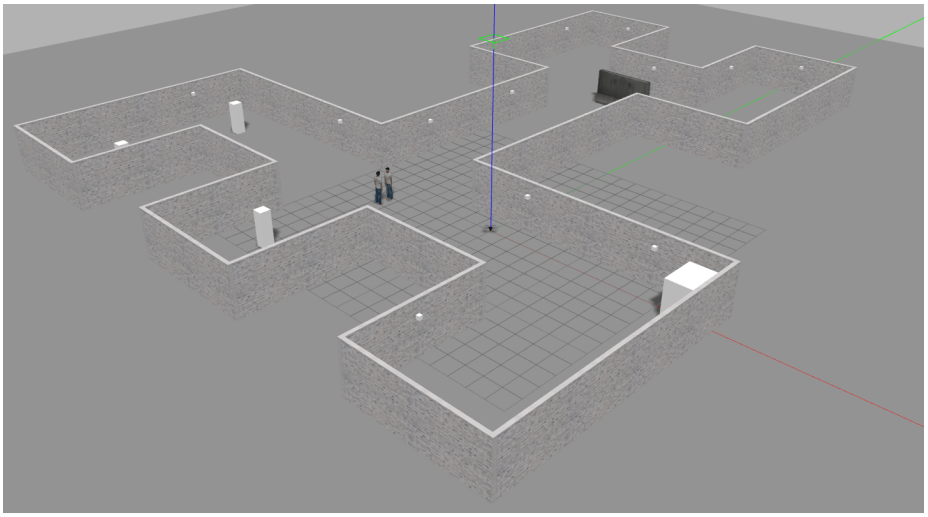


Figure 4.8: Map 2 with Obstacles

4.2.2.3 Time

This criterion is the total time spent throughout the mapping process of an unknown indoor environment. This performance criterion, which is indicated by t and calculated in seconds, can be controlled by means of a three-dimensional simulation tool.

4.2.2.4 Entropy

Shannon Entropy, which is used to measure uncertainty about random variables, is one of the most important criteria of information theory. This concept was introduced in 1948 by Claude E. Shannon, who gave his name to the calculation [3]. Shannon Entropy is formulated as follows,

$$H(x) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \tag{4.2}$$

In the equation, Shannon Entropy is expressed as $H(x)$, while the probability of the source is shown as $p(x_i)$. The logarithm base b is generally considered to be 2 in the calculations.

With the help of a code written on MATLAB, the histograms are obtained with the "hist3 ()" command and the entropies are calculated with the "entropy ()" command.

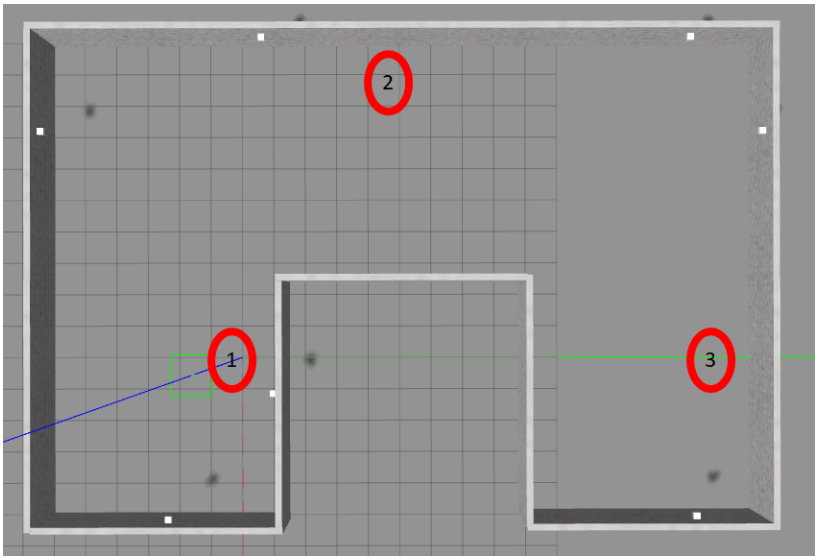


Figure 4.9: Release Numbers for Repeatability Analysis on Map 1

Since the large entropy value indicates that mapping is performed by leaving more traces on the map, the navigation algorithm with a small entropy value is considered more successful. Entropy values of two different autonomous navigation algorithms are shared in "Results" chapter.

In Figure 4.11, the dark blue sections symbolize the map of the enclosed area. The dots of different colors indicate the trace that the robot left behind after autonomous navigation. By calculating the probabilities of different colored squares, entropy information is obtained. This information refers to the disorder in the room. The high entropy value can be achieved by the excess of squares of different colors in the room. In other words, the more traces the robot leaves on the map, the greater the irregularity. Therefore, low entropy value makes the algorithm successful.

4.2.2.5 Repeatability

Repeatability criterion is specified to measure the ability to perform similar performances regardless of the starting point. No units have been identified for this criterion. It would be measured by using other metrics.

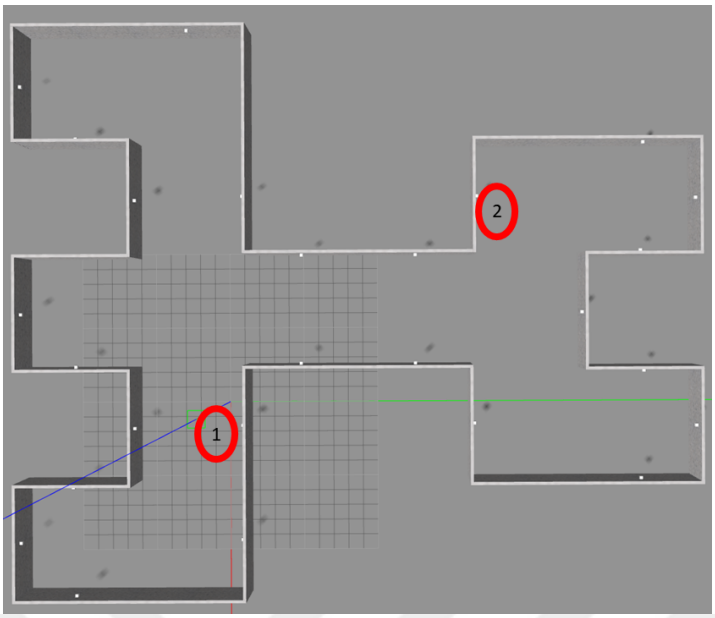


Figure 4.10: Release Numbers for Repeatability Analysis on Map 2

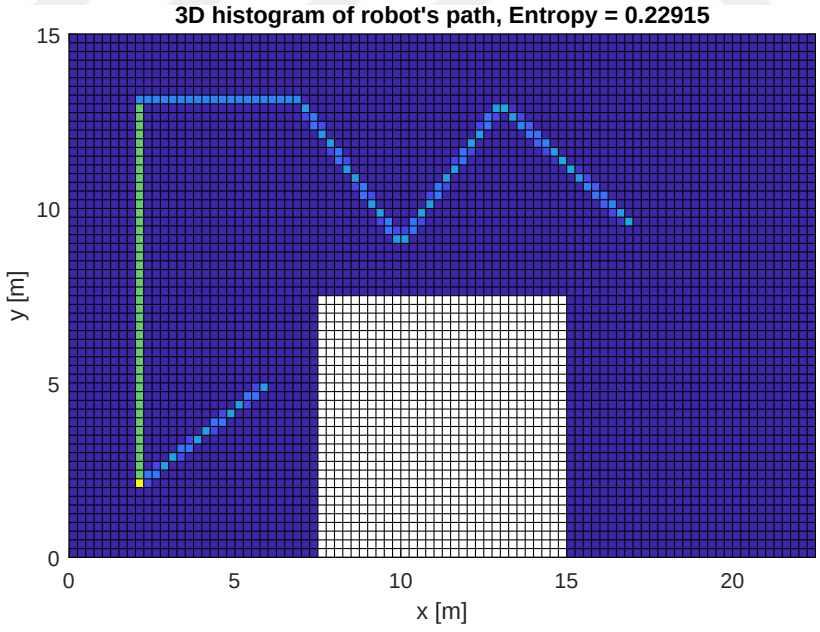


Figure 4.11: Sample 3D Histogram Graph

CHAPTER 5

RESULTS

5.1 Simulations

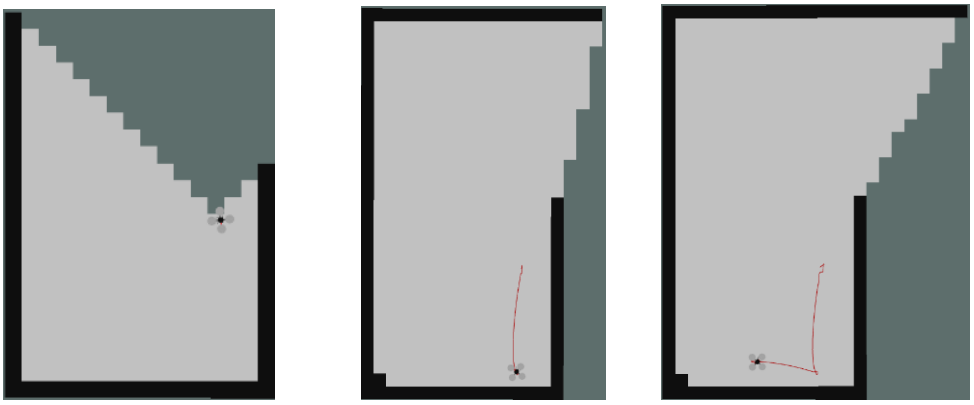
Simulations performed in Map 1 without obstacles are shown in Figures 5.1, 5.2 and 5.3. The map is explored with five stops in Figure 5.1 while it is explored with six stops in Figure 5.2. On the other hand, number of stops is four in Figure 5.3 and it is equal to number of corners since wall following algorithm stops if it encounters with a wall.

The three-dimensional histograms of the map 1 and the trace left by the robot on the map are plotted in Figure 5.4. Wall following algorithm has the smallest entropy value while the rest share similar entropy values.

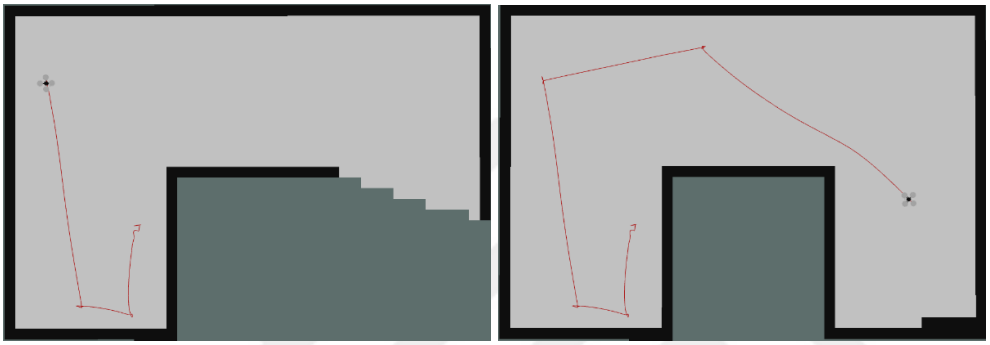
In order to measure repeatability, quadrotors are released at different positions. Results are given in Figures 5.5, 5.6 and 5.7. Entropy values of the target-based navigation algorithm are close to each other for different starting points. Exploration algorithm and wall following algorithm get distant values depending on the initial positions.

Performances of the algorithms on map 1 without obstacles are shown in Table 5.1. Target-based navigation algorithm obtain better results compared with other algorithms except entropy value.

Simulations performed in Map 1 with obstacles are shown in Figures 5.8, 5.9 and 5.10. The map is explored with five stops both in Figure 5.8 and Figure 5.9. Number of stops is four for wall following algorithm and it follows the same path in previous simulation without obstacles in Figure 5.10.



(a) Time step 1 (b) Time step 2 (c) Time step 3



(d) Time step 4 (e) Time step 5

Figure 5.1: Navigation with Exploration Algorithm on Map 1 without Obstacles

Table 5.1: Performance Comparison Table for Map 1 without Obstacles

	Exploration Algorithm	Target-Based Navigation Algorithm	Wall Following Algorithm
$d\ [m]$	37.62	36.32	45
$P_d\ [1/m]$	0.133743	0.12912	0.16
$t\ [s]$	253	247	312
H	0.23142	0.22915	0.21645

The three-dimensional histograms of the map 1 with obstacles and the trace left by the robot on the map are plotted in Figure 5.11. There is no big difference among entropy values. However, wall following algorithm has the smallest entropy value

while the rest share similar entropy values.

In order to measure repeatability, quadrotors are released at different positions. Results are given in Figures 5.12, 5.13, and 5.14. Entropy values of the target-based navigation algorithm are so close to each other for different starting points. Entropy values of exploration algorithm and wall following algorithm deviate more depending on the initial positions.

Performances of the algorithms on map 1 with obstacles are shown in Table 5.2. Target-based navigation algorithm obtain better results compared with other algorithms except entropy value.

Table 5.2: Performance Comparison Table for Map 1 with Obstacles

	Exploration Algorithm	Target-Based Navigation Algorithm	Wall Following Algorithm
$d\,[m]$	32.91	31.22	40
$P_d\,[1/m]$	0.117026	0.110993	0.142222
$t\,[s]$	220	214	277
H	0.20136	0.20544	0.19384

Simulations performed in map 2 without obstacles are illustrated in Figures 5.15 and 5.16. Number of stops in Figure 5.15 is greater although the distance travelled is short. Oppositely, a couple of stops exist in Figure 5.16 while the distance travelled is long.

The three-dimensional histograms of the map 2 without obstacles and the trace left by the robot on the map are plotted in Figure 5.17. There is a significant difference between entropy values. Target-based navigation algorithm obtain the smaller entropy value.

In order to measure repeatability, quadrotors are released at different positions. Results are given in Figures 5.18 and 5.19. Entropy values of both navigation algorithms are so close among themselves for different starting points. However, entropy values of target-based navigation algorithm is smaller than entropy values of wall following

algorithm.

Performances of the algorithms on map 2 without obstacles are shown in Table 5.3. Target-based navigation algorithm is far superior to wall following algorithm in terms of all performance metrics.

Table 5.3: Performance Comparison Table for Map 2 without Obstacles

	Target-Based Navigation Algorithm	Wall Following Algorithm
$d\ [m]$	157.66	181.5
$P_d\ [1/m]$	0.186859	0.215111
$t\ [s]$	1239	1393
H	0.18724	0.23961

Simulations performed in map 2 with obstacles are shown in Figures 5.20 and 5.21. Number of stops in Figure 5.20 decreases compared with previous simulation due to obstacles. Wall following algorithm performs similar performance with in previous simulation without obstacles in Figure 5.21. Similarly, it travels longer distance.

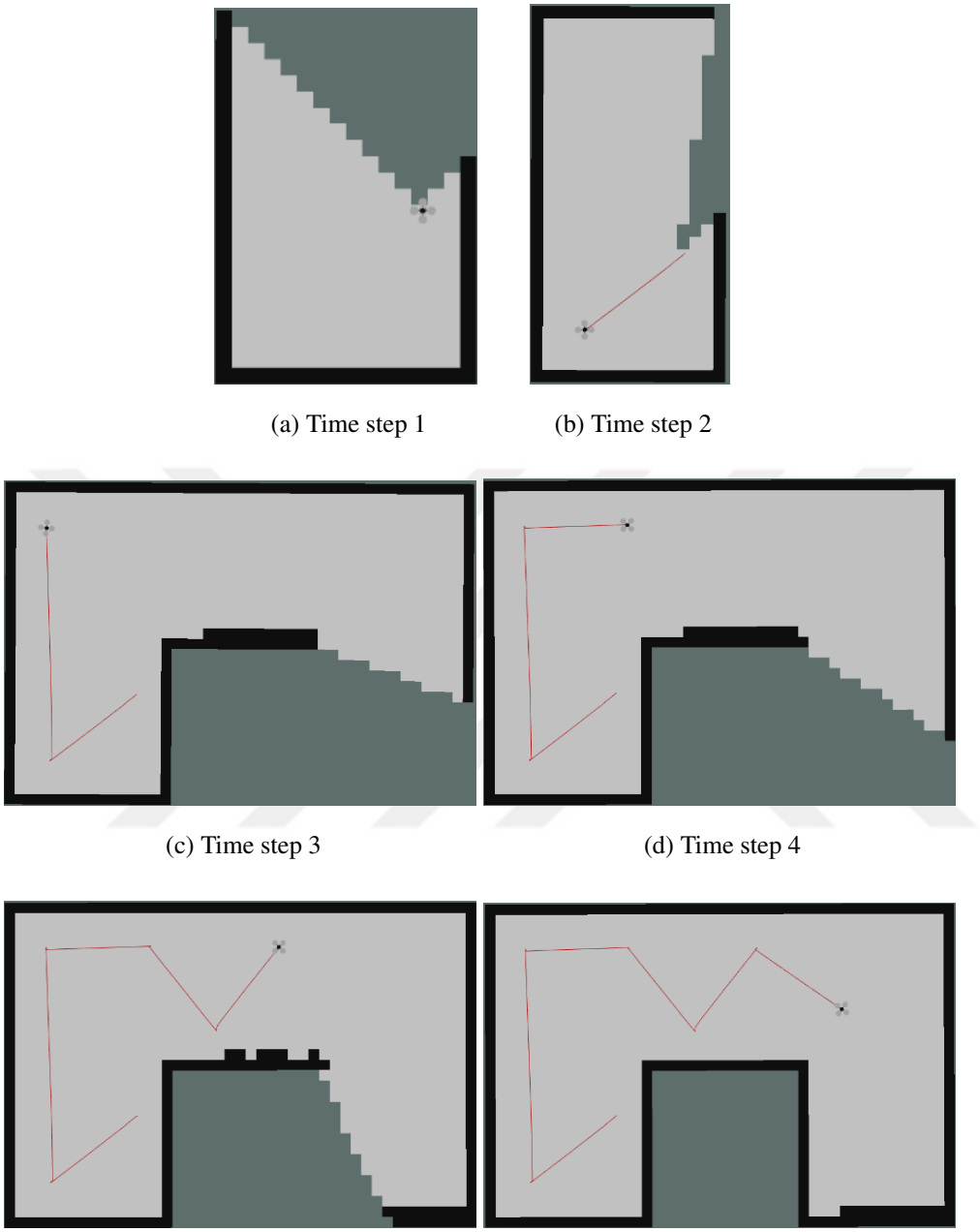
The three-dimensional histograms of the map 2 with obstacles and the trace left by the robot on the map are plotted in Figure 5.22. There is a significant difference between entropy values. Target-based navigation algorithm obtain the smaller entropy value.

In order to measure repeatability, quadrotors are released at different positions. Results are given in Figures 5.23 and 5.24. Entropy values of both navigation algorithms are so close among themselves for different starting points. However, entropy values of target-based navigation algorithm is smaller than entropy values of wall following algorithm.

Performances of the algorithms on map 2 with obstacles are shown in Table 5.4. Target-based navigation algorithm is far superior to wall following algorithm in terms of all performance metrics.

Table 5.4: Performance Comparison Table for Map 2 with Obstacles

	Target-Based Navigation Algorithm	Wall Following Algorithm
$d\,[m]$	103.3	181.5
$P_d\,[1/m]$	0.122436	0.215111
$t\,[s]$	812	1393
H	0.15011	0.23923

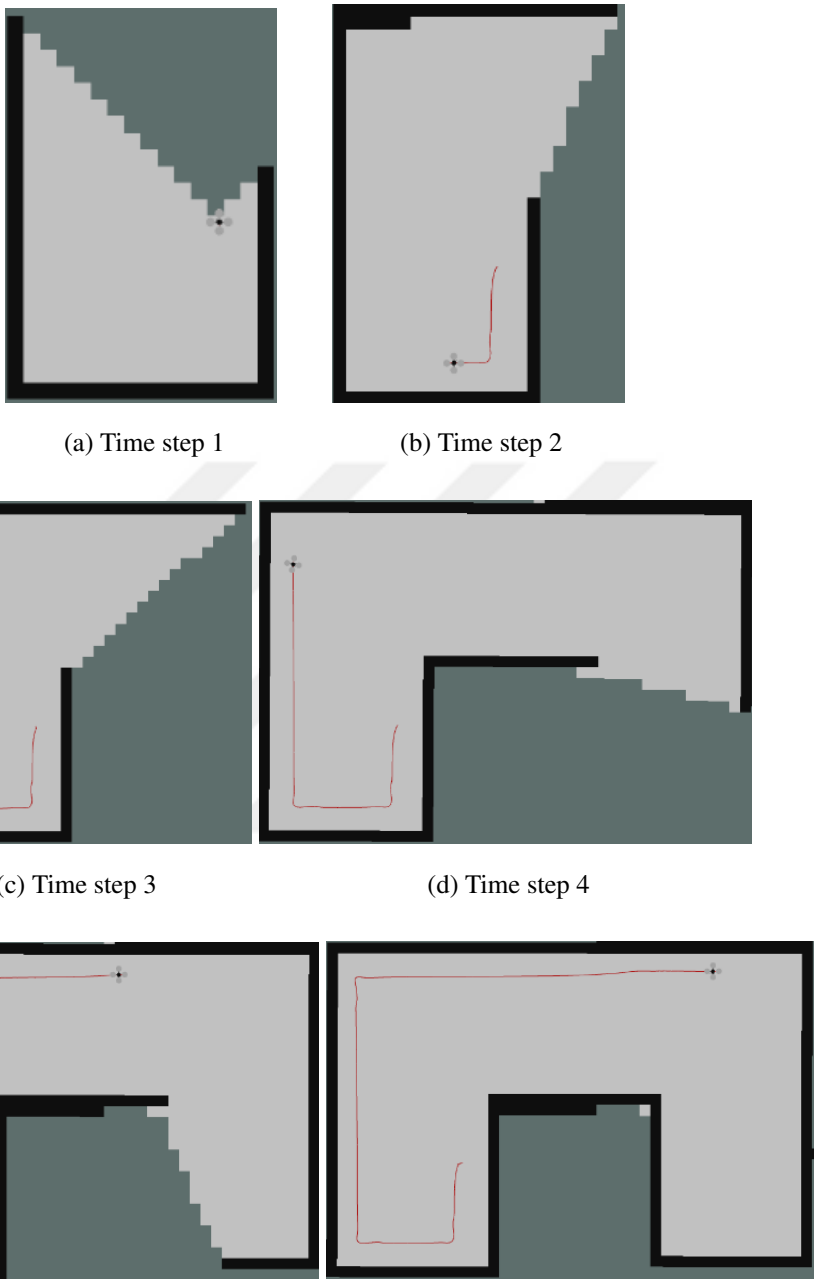


(a) Time step 1 (b) Time step 2

(c) Time step 3 (d) Time step 4

(e) Time step 5 (f) Time step 6

Figure 5.2: Navigation with Target-Based Navigation Algorithm on Map 1 without Obstacles

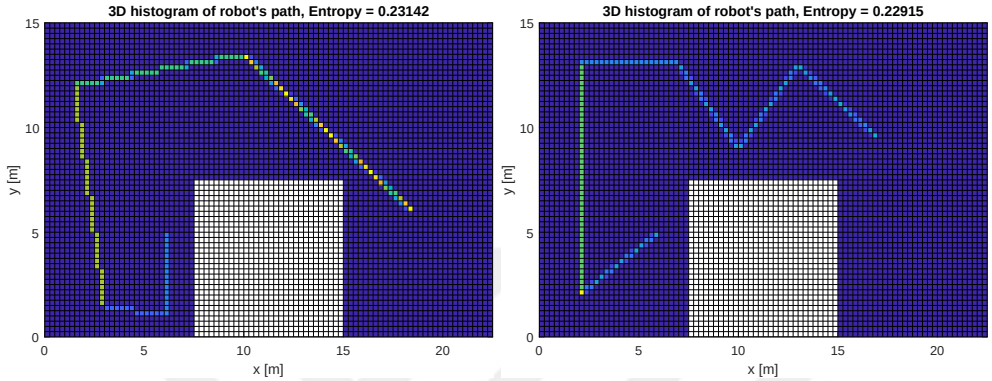


(a) Time step 1 (b) Time step 2

(c) Time step 3 (d) Time step 4

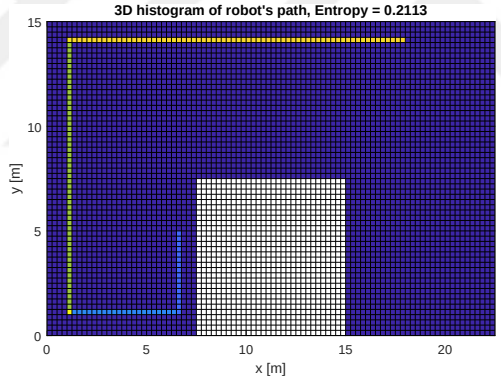
(e) Time step 5 (f) Time step 6

Figure 5.3: Navigation with Wall Following Algorithm on Map 1 without Obstacles



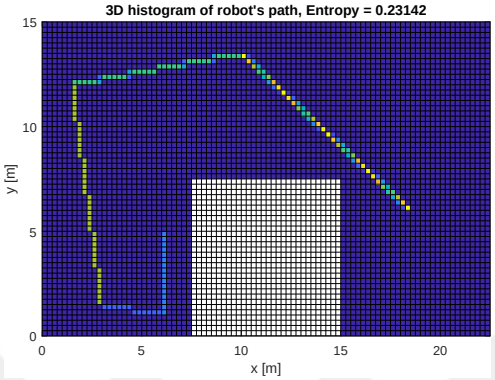
(a) Exploration Algorithm

(b) Target-Based Navigation Algorithm

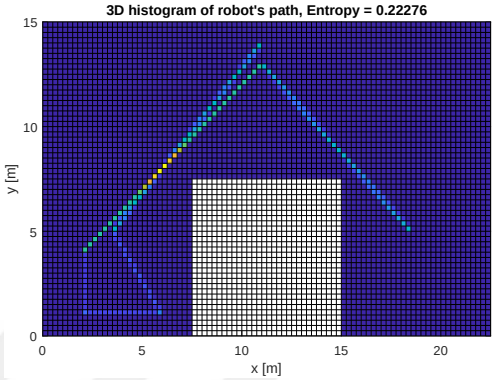


(c) Wall Following Algorithm

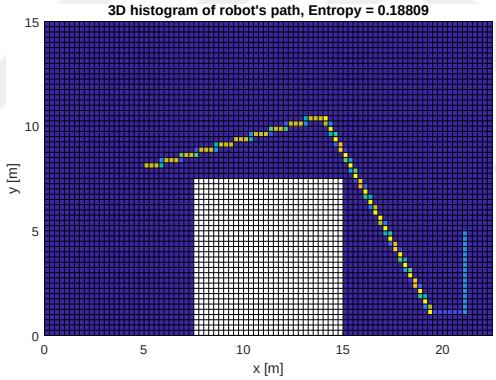
Figure 5.4: 3 Dimensional Histogram Graph of Map 1 without Obstacles



(a) Exploration Algorithm at First Initial Position

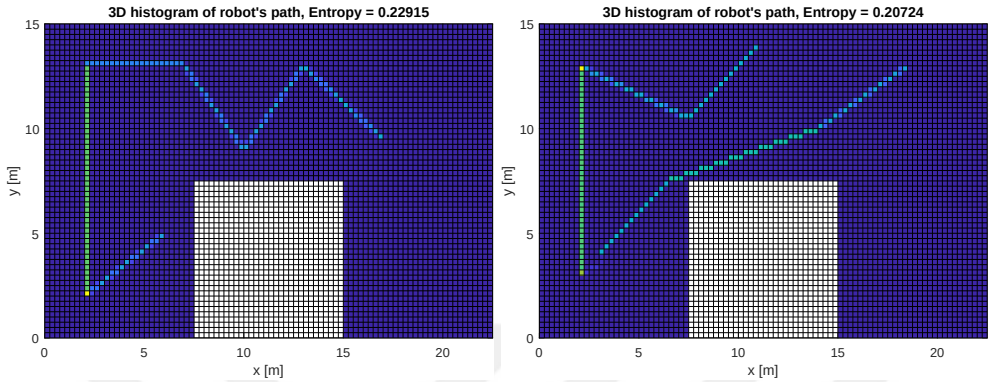


(b) Exploration Algorithm at Second Initial Position



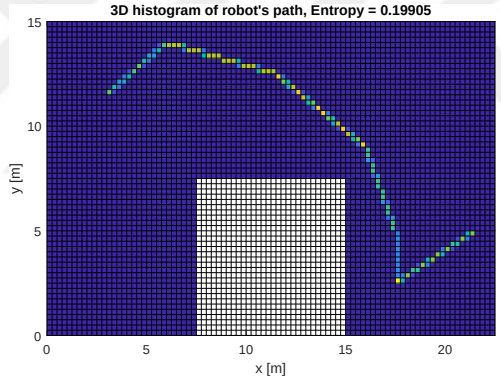
(c) Exploration Algorithm at Third Initial Position

Figure 5.5: Repeatability Analysis for Exploration Algorithm in Map 1 without Obstacles



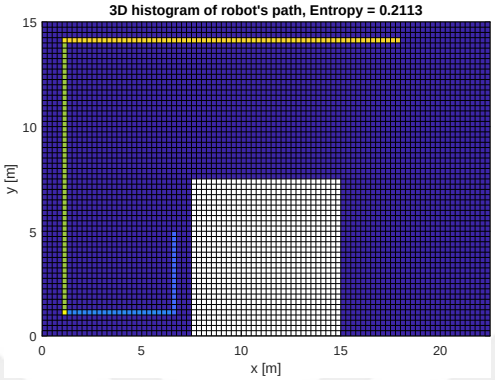
(a) Target-Based Navigation Algorithm at First Initial Position

(b) Target-Based Navigation Algorithm at Second Initial Position

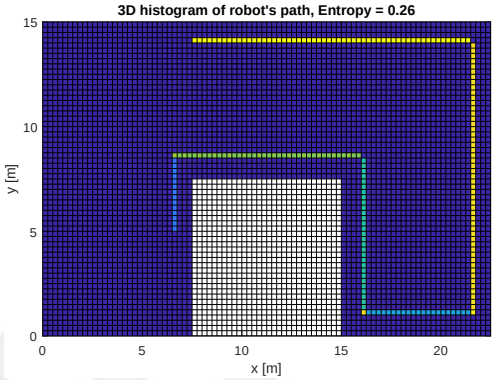


(c) Target-Based Navigation Algorithm at Third Initial Position

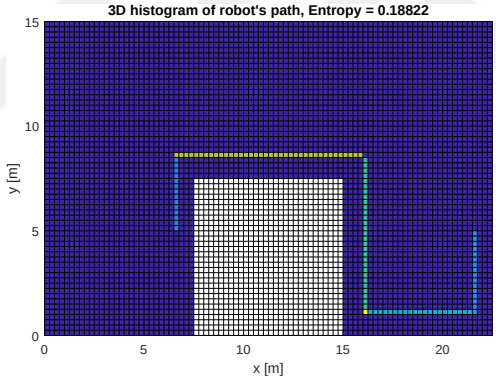
Figure 5.6: Repeatability Analysis for Target-Based Navigation Algorithm in Map 1 without Obstacles



(a) Target-Based Navigation Algorithm at First Initial Position

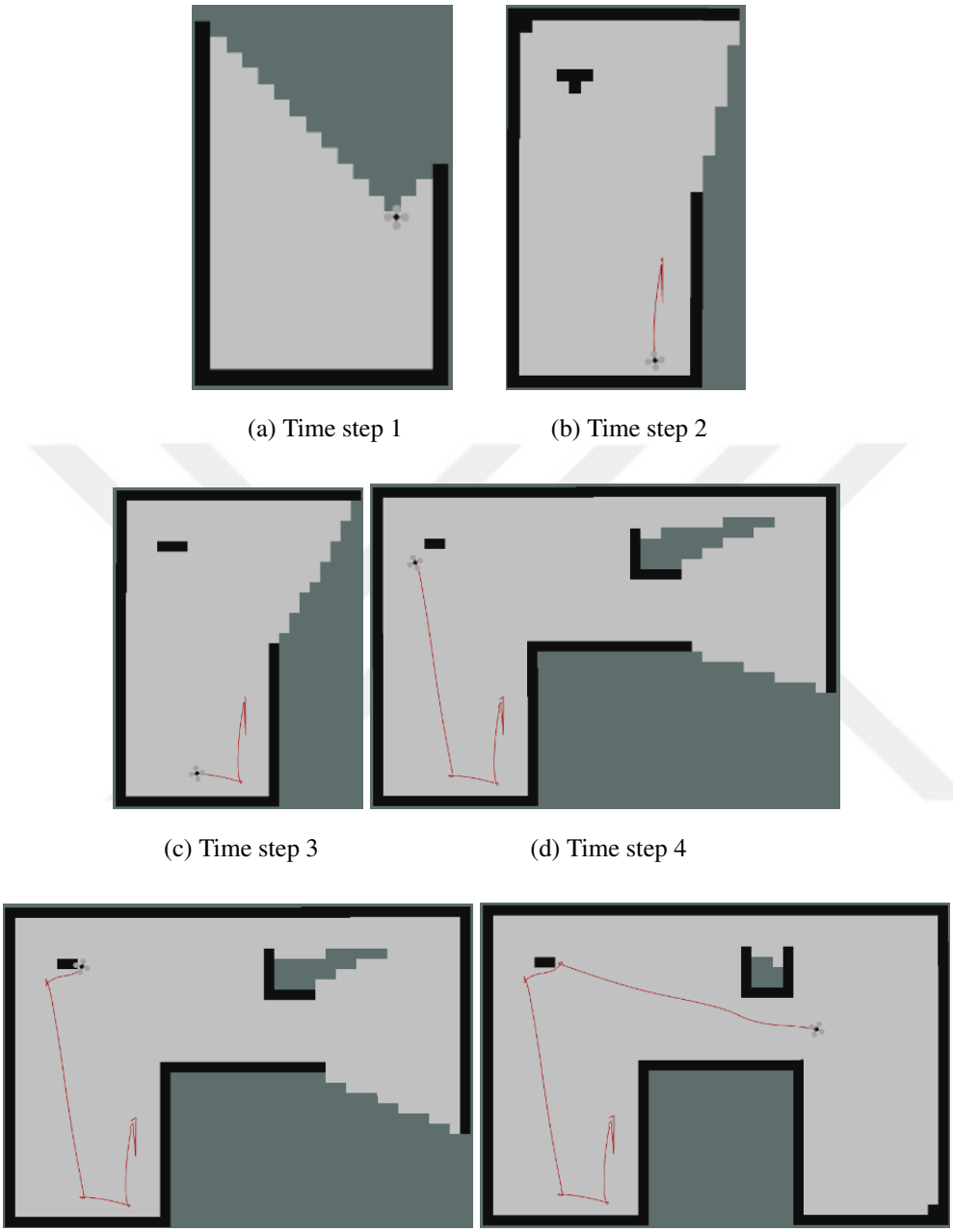


(b) Target-Based Navigation Algorithm at Second Initial Position



(c) Target-Based Navigation Algorithm at Third Initial Position

Figure 5.7: Repeatability Analysis for Wall Following Algorithm in Map 1 without Obstacles

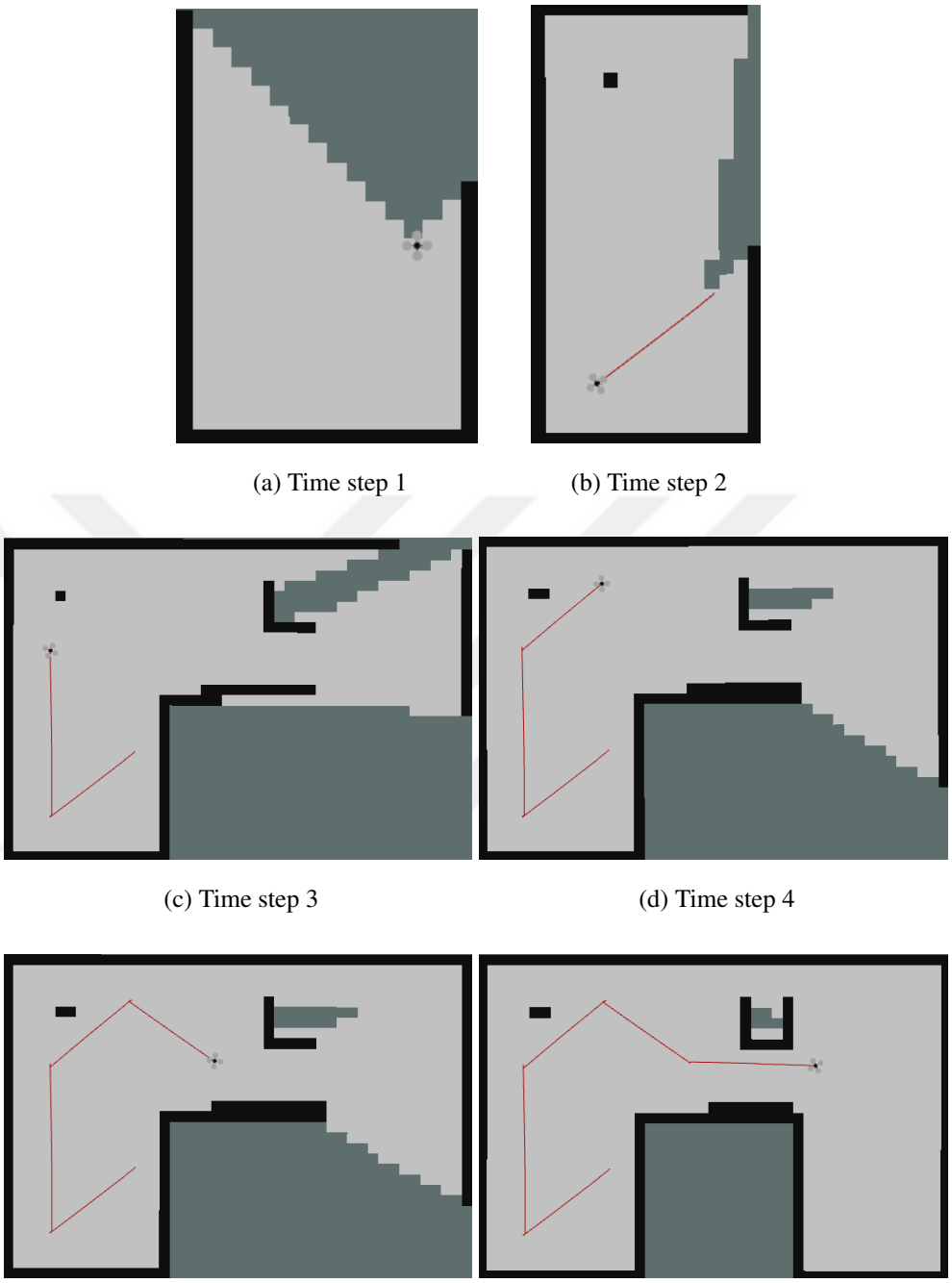


(a) Time step 1 (b) Time step 2

(c) Time step 3 (d) Time step 4

(e) Time step 5 (f) Time step 6

Figure 5.8: Navigation with Exploration Algorithm on Map 1 with Obstacles

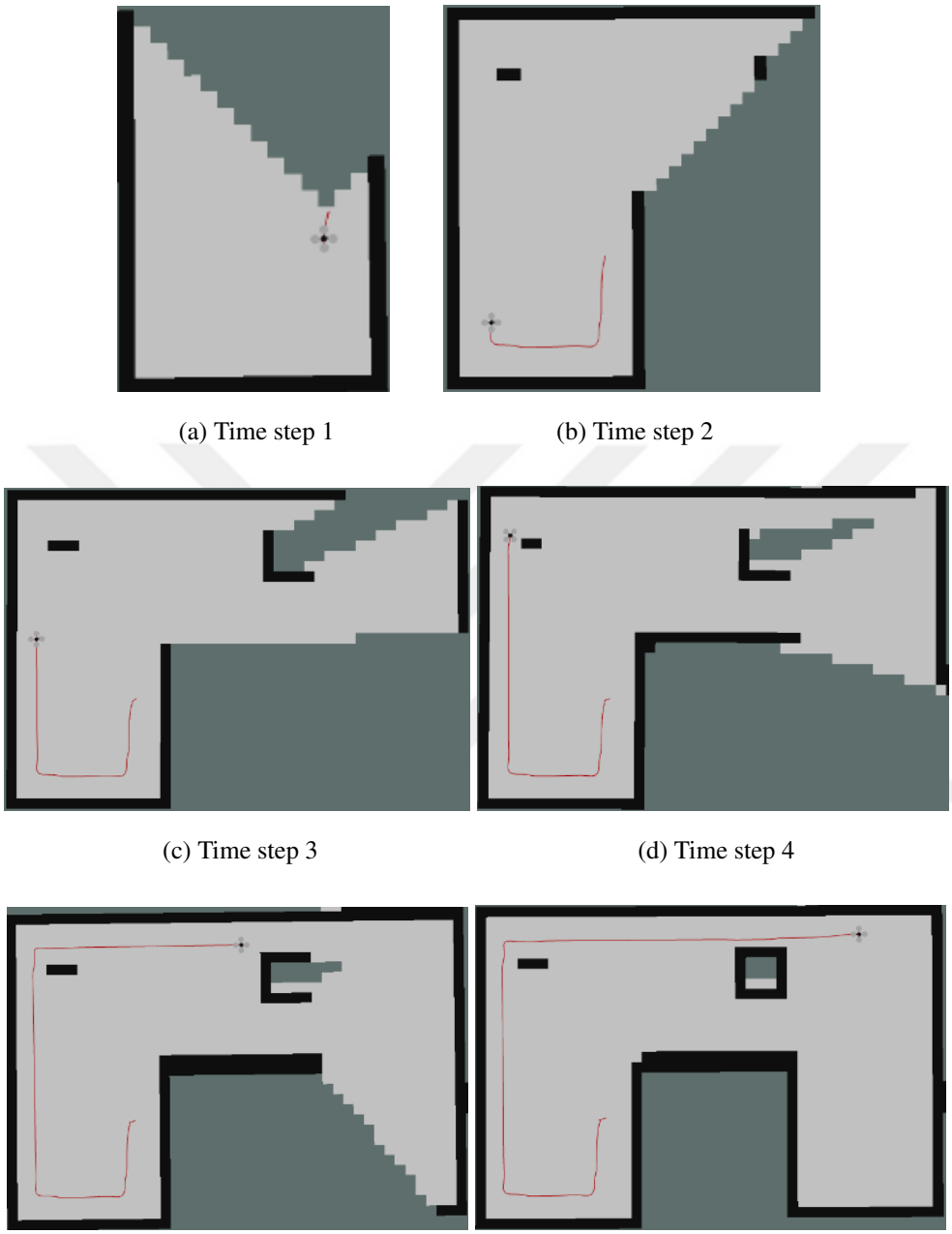


(a) Time step 1 (b) Time step 2

(c) Time step 3 (d) Time step 4

(e) Time step 5 (f) Time step 6

Figure 5.9: Navigation with Target-Based Navigation Algorithm on Map 1 with Obstacles



(a) Time step 1

(b) Time step 2

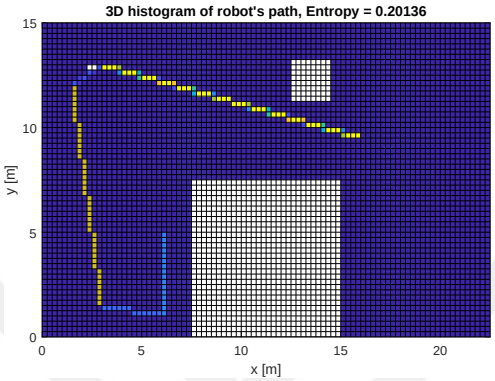
(c) Time step 3

(d) Time step 4

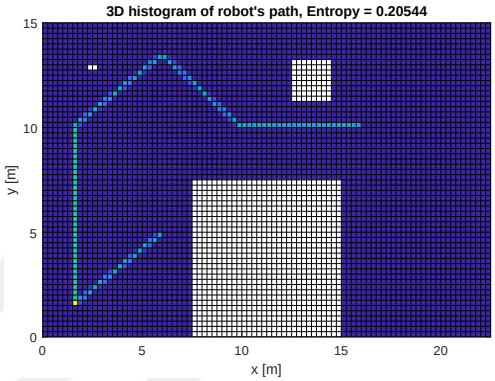
(e) Time step 5

(f) Time step 6

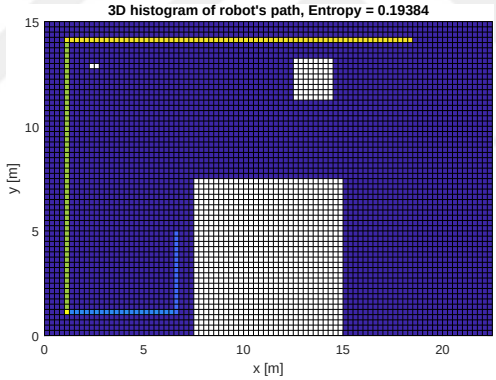
Figure 5.10: Navigation with Wall Fallowing Algorithm on Map 1 with Obstacles



(a) Exploration Algorithm

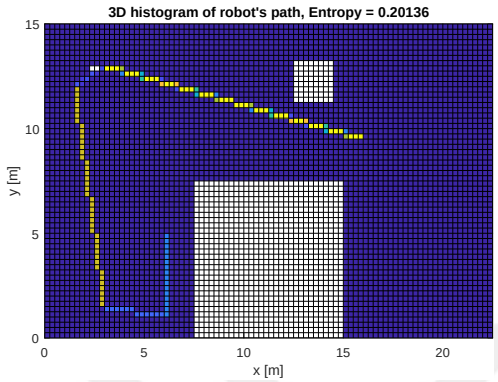


(b) Target-Based Navigation Algorithm

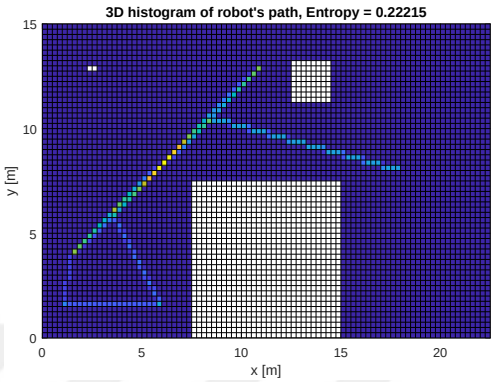


(c) Wall Following Algorithm

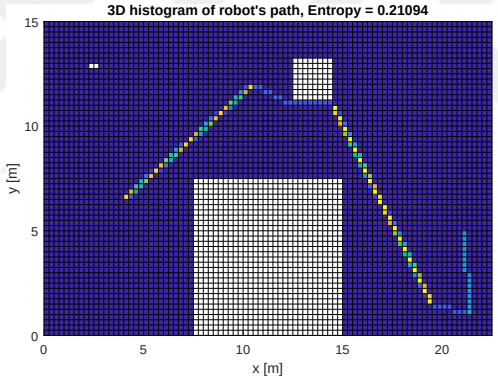
Figure 5.11: 3 Dimensional Histogram Graph of Map 1 with Obstacles



(a) Exploration Algorithm at First Initial Position

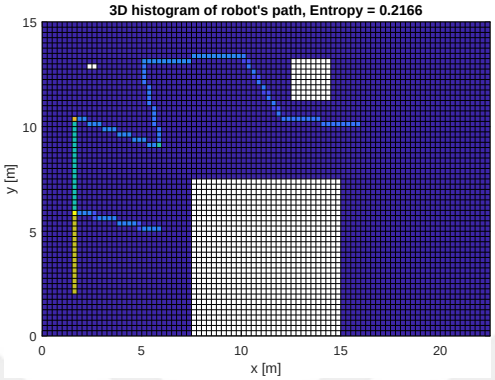


(b) Exploration Algorithm at Second Initial Position

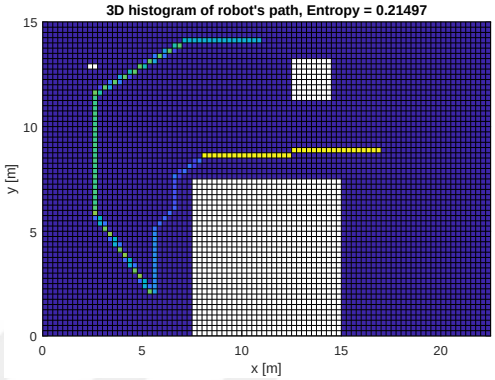


(c) Exploration Algorithm at Third Initial Position

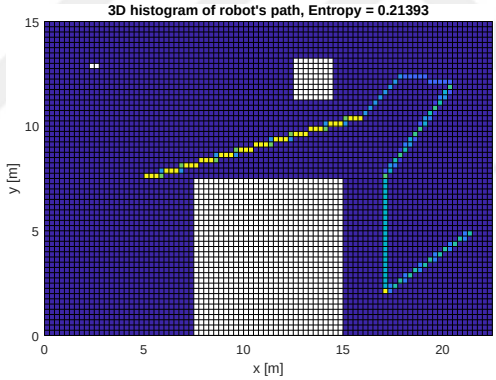
Figure 5.12: Repeatability Analysis for Exploration Algorithm in Map 1 with Obstacles



(a) Target-Based Navigation Algorithm at First Initial Position

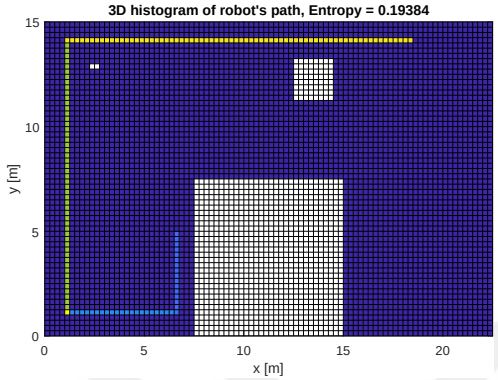


(b) Target-Based Navigation Algorithm at Second Initial Position

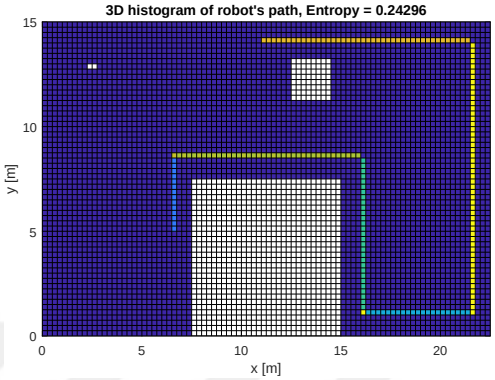


(c) Target-Based Navigation Algorithm at Third Initial Position

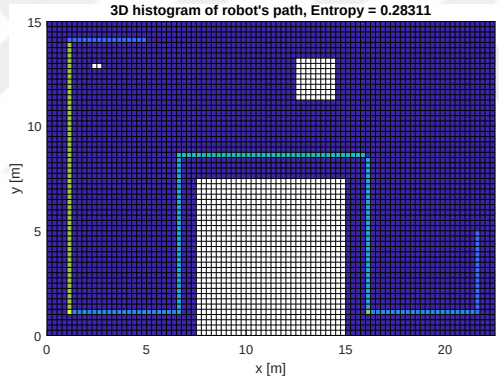
Figure 5.13: Repeatability Analysis for Target-Based Navigation Algorithm in Map 1 with Obstacles



(a) Target-Based Navigation Algorithm at First Initial Position

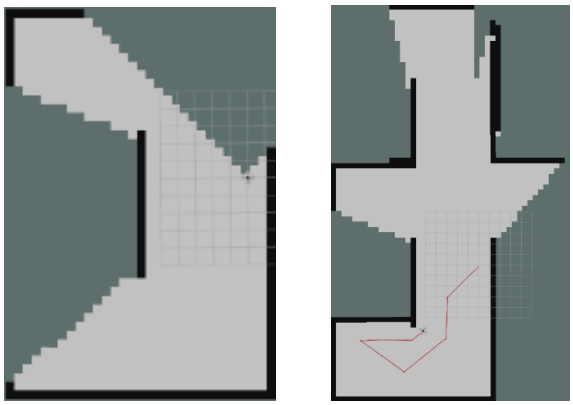


(b) Target-Based Navigation Algorithm at Second Initial Position

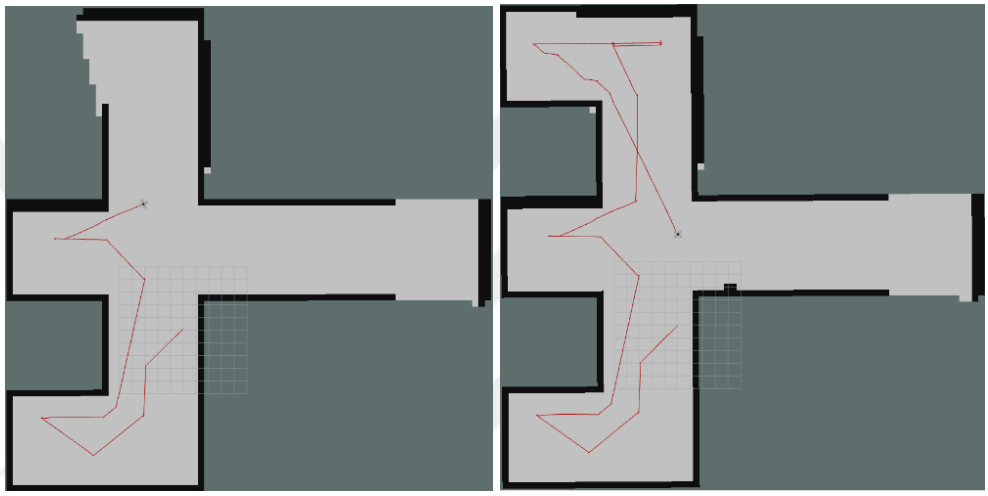


(c) Target-Based Navigation Algorithm at Third Initial Position

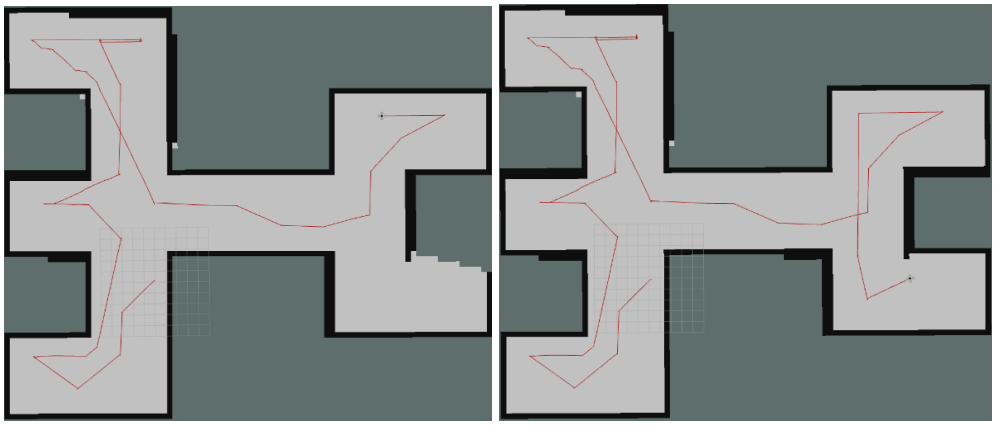
Figure 5.14: Repeatability Analysis for Wall Following Algorithm in Map 1 with Obstacles



(a) Time step 1 (b) Time step 2

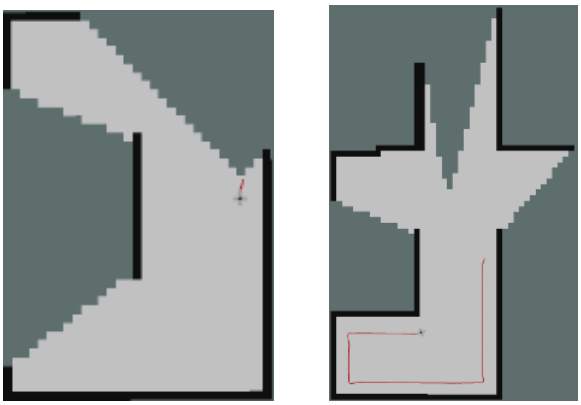


(c) Time step 3 (d) Time step 4

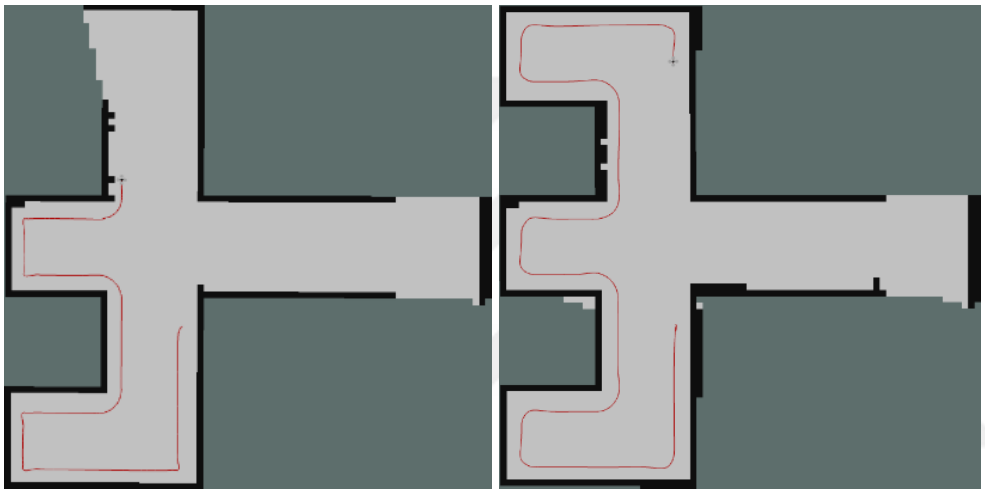


(e) Time step 5 (f) Time step 6

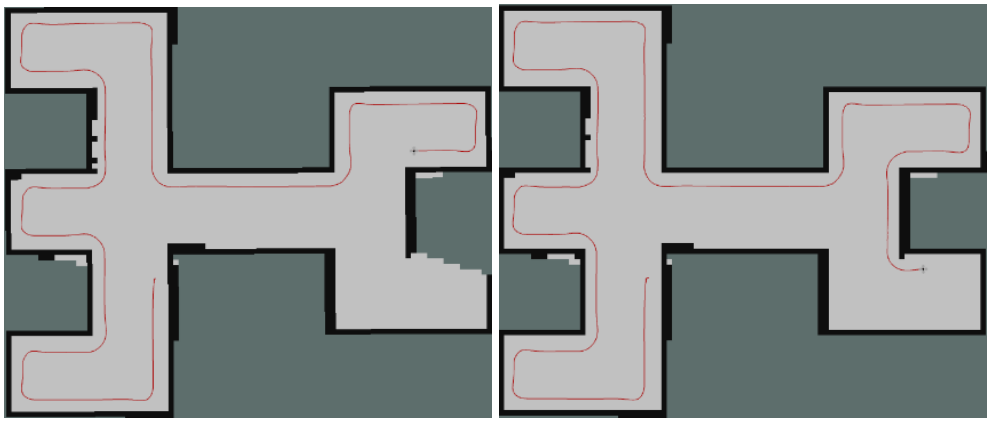
Figure 5.15: Navigation with Target-Based Navigation Algorithm on Map 2 without Obstacles



(a) Time step 1 (b) Time step 2

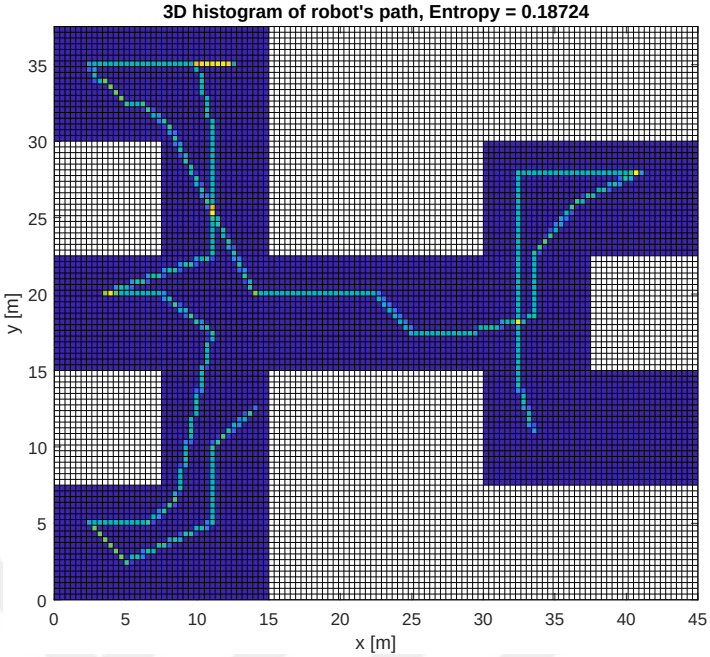


(c) Time step 3 (d) Time step 4

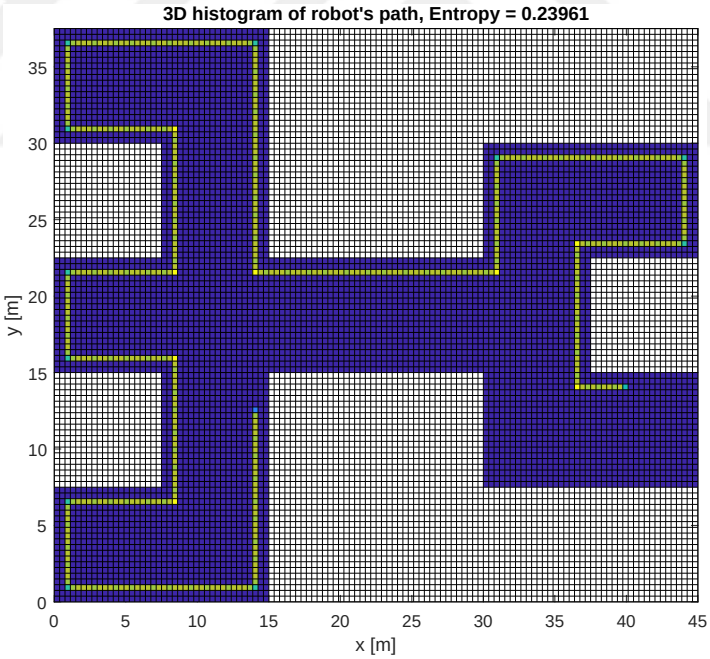


(e) Time step 5 (f) Time step 6

Figure 5.16: Navigation with Wall Following Algorithm on Map 2 without Obstacles

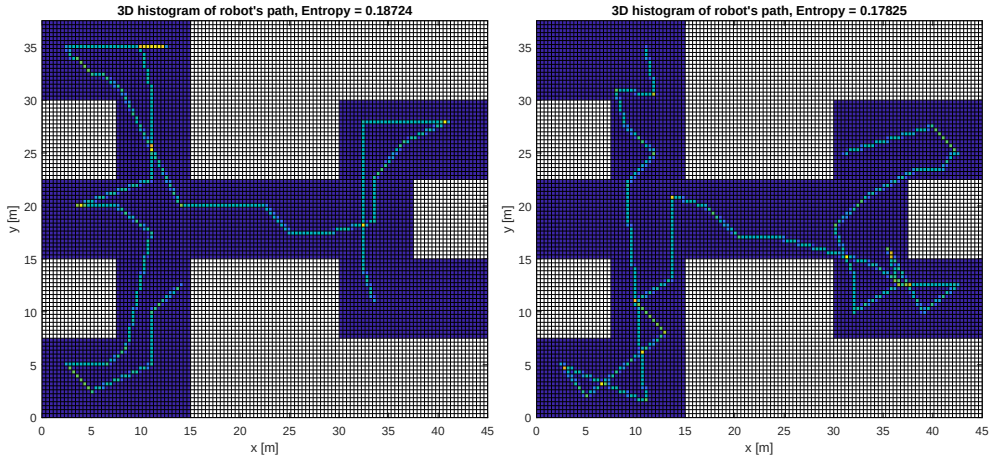


(a) Target-Based Navigation Algorithm



(b) Wall Following Algorithm

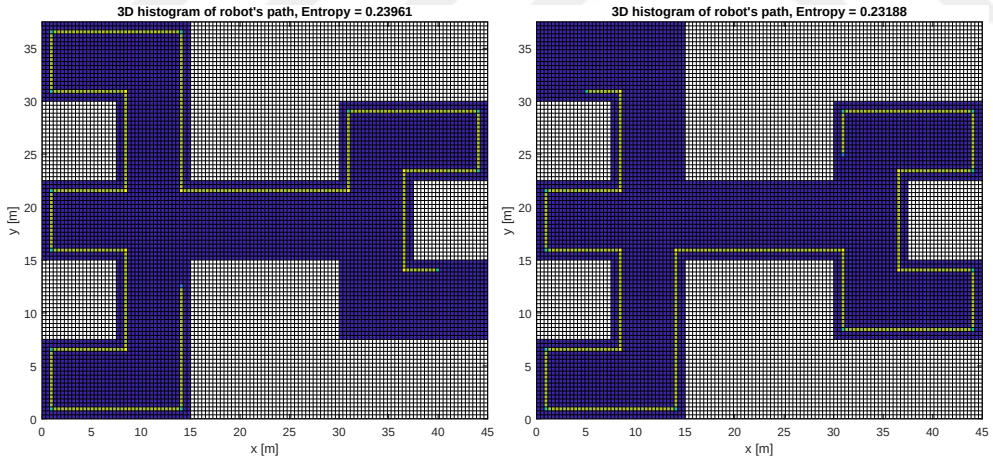
Figure 5.17: 3 Dimensional Histogram Graph of Map 2 without Obstacles



(a) Target-Based Navigation Algorithm at First Initial Position

(b) Target-Based Navigation at Second Initial Position

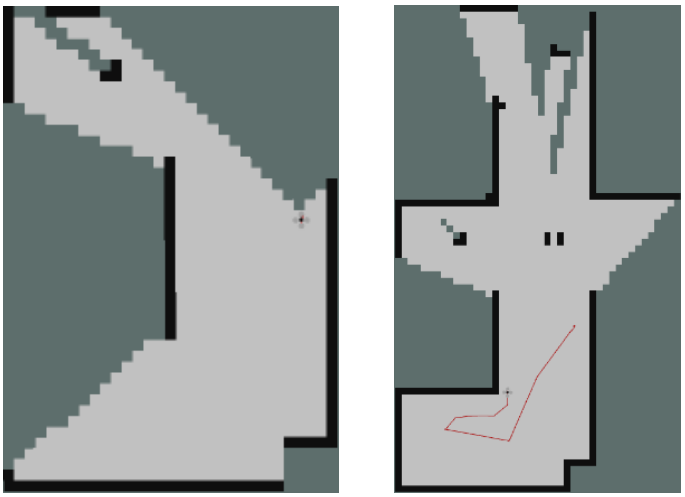
Figure 5.18: Repeatability Analysis for Target-Based Navigation Algorithm in Map 2 without Obstacles



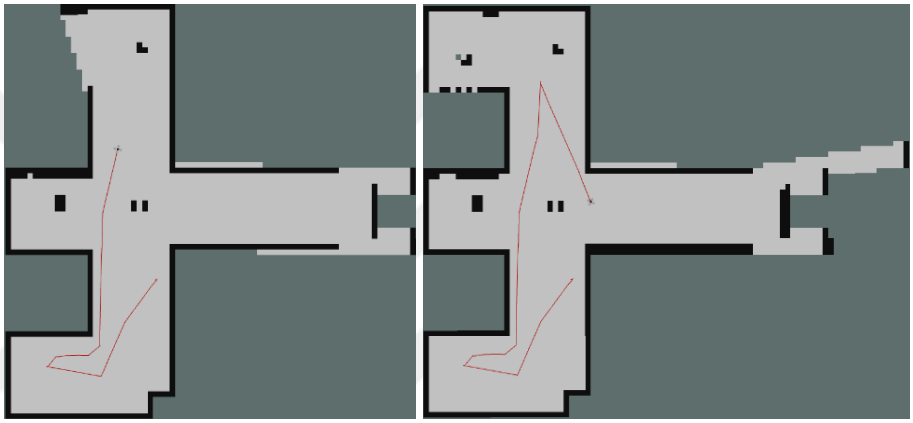
(a) Wall Following Algorithm at First Initial Position

(b) Wall Following Algorithm at Second Initial Position

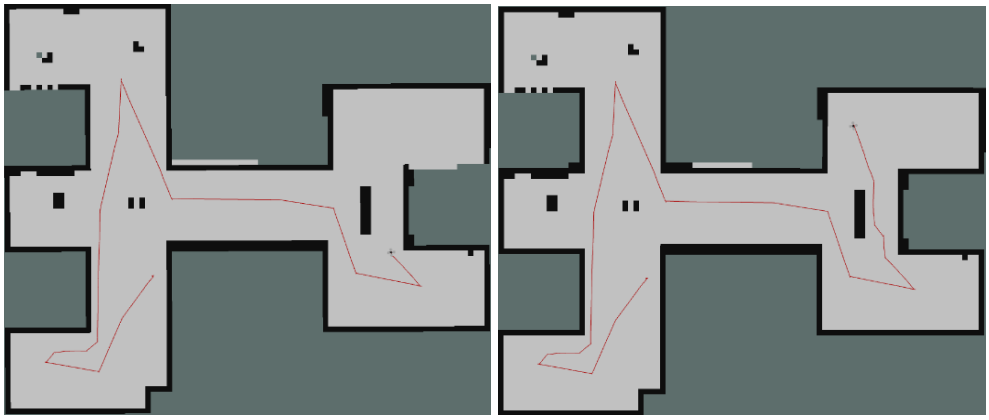
Figure 5.19: Repeatability Analysis for Wall Following Algorithm in Map 2 without Obstacles



(a) Time step 1 (b) Time step 2



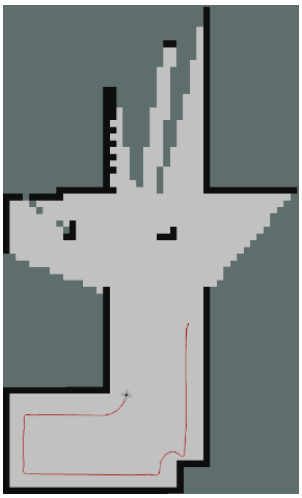
(c) Time step 3 (d) Time step 4



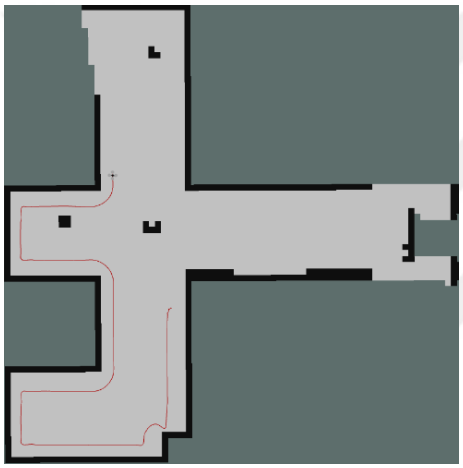
(e) Time step 5 (f) Time step 6

Figure 5.20: Navigation with Target-Based Navigation Algorithm on Map 2 with Obstacles

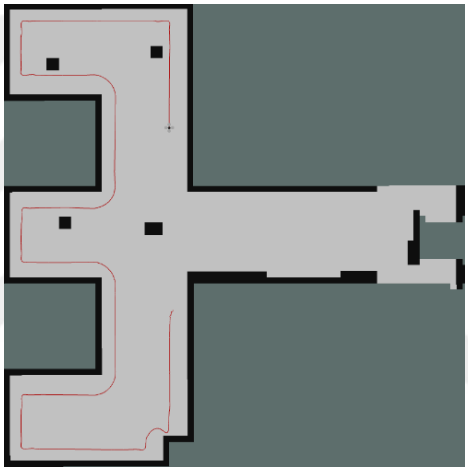
(a) Time step 1



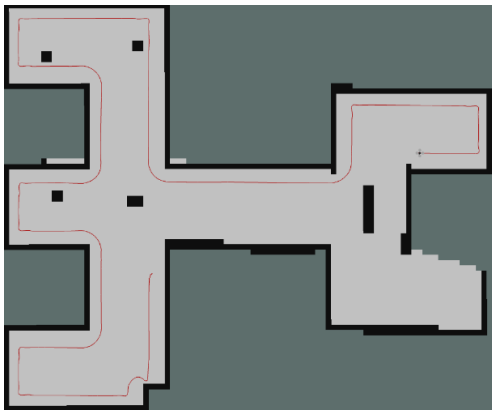
(b) Time step 2



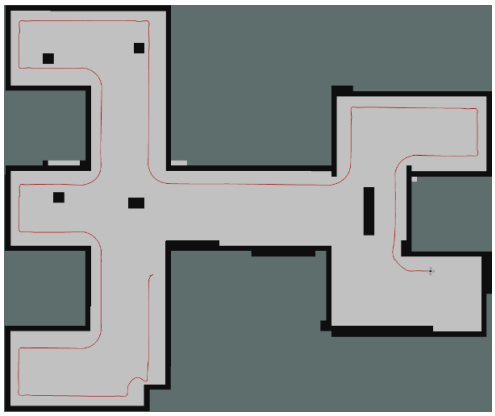
(c) Time step 3



(d) Time step 4

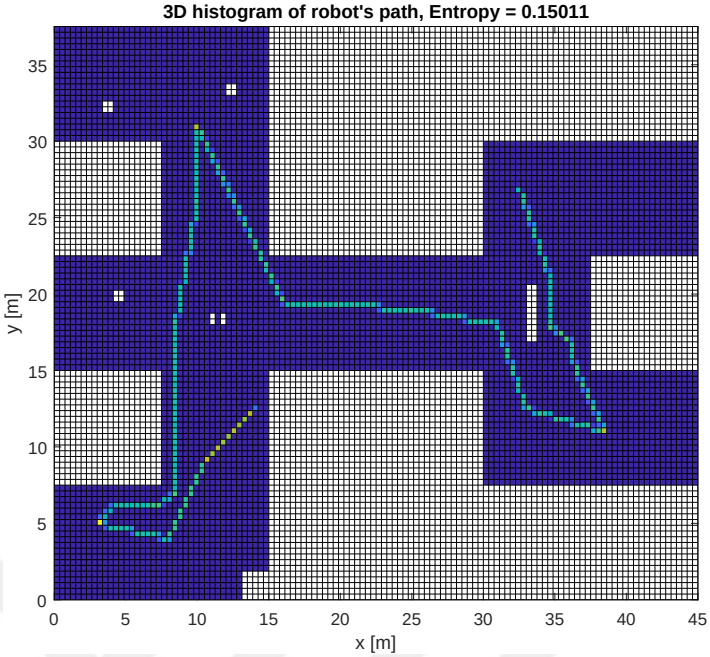


(e) Time step 5

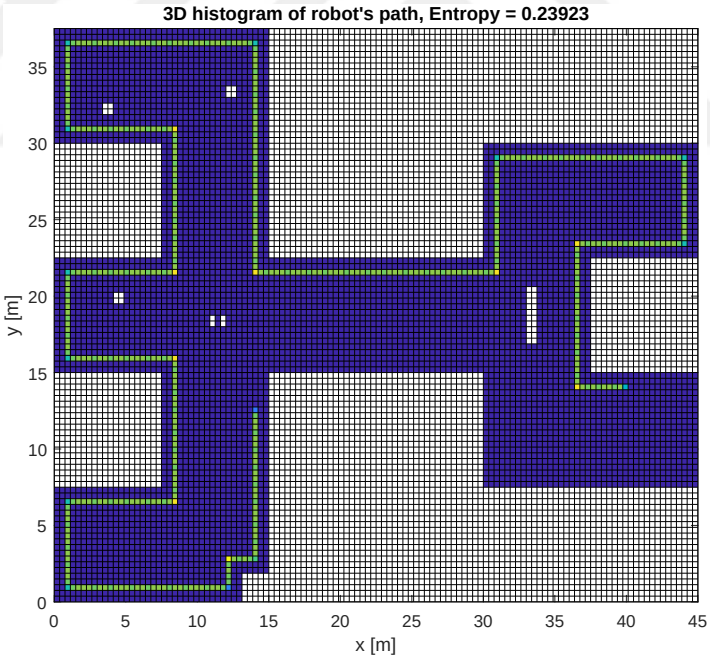


(f) Time step 6

Figure 5.21: Navigation with Wall Following Algorithm on Map 2 with Obstacles

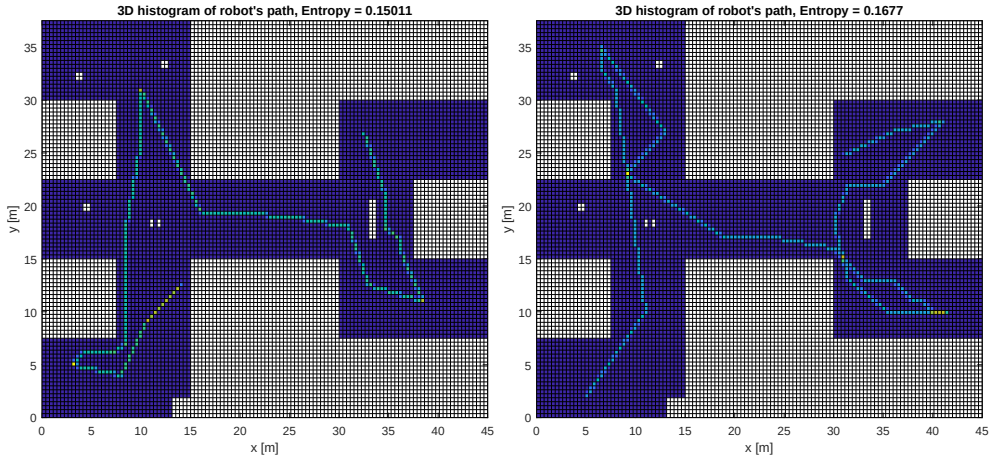


(a) Target-Based Navigation Algorithm



(b) Wall Following Algorithm

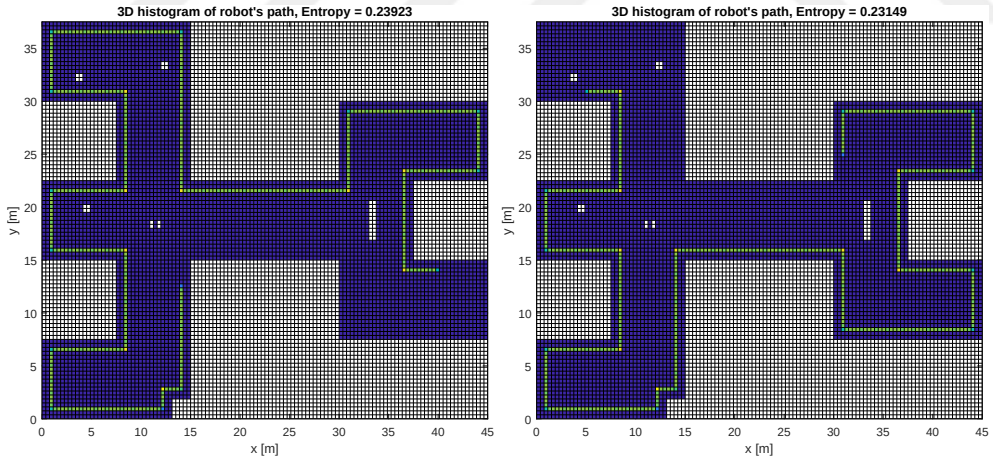
Figure 5.22: 3 Dimensional Histogram Graph of Map 2 with Obstacles



(a) Target-Based Navigation Algorithm at First Initial Position

(b) Target-Based Navigation Algorithm at Second Initial Position

Figure 5.23: Repeatability Analysis for Target-Based Navigation Algorithm in Map 2 with Obstacles



(a) Wall Following Algorithm at First Initial Position

(b) Wall Following Algorithm at Second Initial Position

Figure 5.24: Repeatability Analysis for Wall Following Algorithm in Map 2 with Obstacles

CHAPTER 6

DISCUSSION

In this study, navigation characteristics of different algorithms on various maps have been investigated. Algorithms are classified according to their success.

Exploration algorithm basically moves towards the corner points. It is successful on small indoors. However, it fails on large indoor environments. Therefore, simulation of this algorithm could not be performed in map 2. Since the obstacles are also considered as corners, the number of visiting points increases during navigation. In other words, the number of destination points is directly proportional to the number of obstacles. It can be observed by examining the simulations on map 1 as shown in Figure 5.1 and Figure 5.8. It is possible to re-visit the discovered areas since it does not keep the positions in its memory. Major changes are observed in entropy values when the initial position of the robot is altered (see Figure 5.5 and Figure 5.12). This indicates that the algorithm is poor in repeatability. In addition, Table 5.1 and Table 5.2 state that exploration algorithm obtains moderate results in terms of distance, effective distance, and time compared with other algorithms.

Wall following is a well known algorithm to navigate in maze-like environments. Although it is known as a simple and effective algorithm, it has some drawbacks. If the size of the map increases, the distance travelled and the time spent also increase. In other words, size of the map is inversely proportional to success of the algorithm. It explores the map in similar ways. Thus, starting from different points for navigation does not affect the entropy value much(see Figures 5.7, 5.14, 5.19, 5.24). It does not check whether the map is explored or not since the only reference is wall tracking. It may obtain successful results in the indoor environments covered with continuous walls. However, it fails if there are nested walls in the environment. Moreover, the

center of the indoor may not be discovered if the area is too large. The performance metrics in Table 5.3 and Table 5.4 show that wall following algorithm has worse results than other algorithms.

Target-based navigation algorithm assigns targets and generates safe paths that can be controlled by parameters. This algorithm is able to explore indoor environments in each and every case. Since angle of sight and range of vision of the sensor are high according to the size of small map, all algorithms obtain similar results. The performance and capability of target-based navigation algorithm can be understood easily on large maps with obstacles. If a small clue is obtained about an unknown indoor to be discovered, the performance of the algorithm can be improved by manipulating the parameters. If there is no restriction in terms of time and distance, the value of parameters can be reduced to form more stops within the indoor environment. Low entropy values are obtained because quadrotor does not make irregular movements and the trace is relatively low (see Figures 5.4, 5.11, 5.17, and 5.22). The time and distance values in the Tables 5.1, 5.2, 5.3, and 5.4 also support this argument. In addition, similar results are obtained regardless of the initial position since it has a consistent target generation algorithm as shown in Figures 5.6, 5.13, 5.19, and 5.24.

All in all, exploration and wall following algorithms move by using the details in the indoor environment. Therefore, they do not check whether the map has been discovered. On the other hand, target based navigation algorithm directly uses the explored area to select the next destination. This opportunity guarantees that the map is explored. In addition, comparisons show that target-based navigation algorithm gets better results than other navigation algorithms.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this study, a novel mapping and obstacles avoidance algorithm was developed for a quadrotor in order to obtain the map of unknown indoor environments. Two different navigation algorithms that were presented in the previous studies were used to compare performance of developed novel algorithm with various performance metrics. LIDAR based SLAM method was used in all algorithms to discover indoor areas. UWB localization was applied to exploration algorithm and novel algorithm by using many anchors placed on the walls and a tag mounted on quadrotor. In this way, local positioning system was formed. It was not used for wall following algorithm since it does not deal with localization. Considering different scenarios, different advantages and disadvantages of algorithms may arise. Wall following is an old and well-known navigation algorithm. It is suitable for maze-like environments but it can be used for any indoor environment. However, it may fail when time and distance travelled are crucial. In addition, it does not succeed indoors with large empty spaces in the middle of the place. Exploration algorithm uses corners and open spaces to find destination points. Even if it is considered as successful at small indoors, it fails in large indoor environments with obstacles. Differently, presented novel algorithm guarantees to obtain the whole map since it directly uses explored areas by composing a matrix. Navigation system of the wall following and exploration algorithms is not based on explored map. The map of an indoor environment is explored unconsciously by these algorithms since they just use the features of the field without checking whether the map is explored or not. In addition, novel algorithm may be applied for any robots including aerial robots, ground robots and underwater robots since it is written with

generic programming style. Furthermore, the characteristics of this navigation algorithm may be altered by configuring the parameters.

The performed simulations showed that novel algorithm mostly beats the opponents. Performance criteria such as time, distance, entropy, etc. reveal the superiority of novel algorithm. Exploration algorithm directly failed in large indoor environments including different obstacles. Although wall following algorithm managed to obtain map of indoors, it wasted time and travelled to a longer distance. Moreover, repeatability analysis indicates that performance metrics come out with close values for novel algorithm. In other words, navigation with novel algorithm raises similar results regardless of the robot's initial position.

7.2 Future Work

The aim of this study was to obtain map of unknown indoors in three dimensional simulation environment. In the future, navigation algorithms can be tested in physical environment by setting up the maps in real world. Instead of LIDAR based SLAM method, camera based SLAM method may be used easily by taking advantage of generic novel algorithm. Experiments can be repeated using a couple of UWB anchors by defining soft walls which UWB signals can pass. In addition, several quadrotors can be released from different points simultaneously in order to reduce discovery time of the map and communication among quadrotors can be established using wireless technologies. Furthermore, novel algorithm can be tested using various aerial, ground and underwater robots.

REFERENCES

[1] Gazebosim.Org.

[2] Ros.Org.

[3] J. M. Abatti. Small power: The role of micro and small UAVs in the future. (November):165–197, 2005.

[4] M. Achtelik, J. Williams, M. J. Owen, and M. C. O'Donovan. Autonomous Navigation and Exploration of a Quadrotor Helicopterin GPS-denied Indoor Environments. *First Symposium on Indoor Flight*, 2009.

[5] S. Ahrens, D. Levine, G. Andrews, and J. P. How. Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2643–2648, 2009.

[6] A. Araújo, D. Portugal, M. S. Couceiro, and R. P. Rocha. Integrating Arduino-Based Educational Mobile Robots in ROS. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 77(2):281–298, 2014.

[7] C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit. Instant Outdoor Localization and SLAM Initialization from 2.5D Maps. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1309–1318, 2015.

[8] V. Barral, P. Suárez-Casal, C. J. Escudero, and J. A. García-Naya. Multi-sensor accurate forklift location and tracking simulation in industrial indoor environments. *Electronics (Switzerland)*, 8(10), 2019.

[9] C. Brenneke, O. Wulf, and B. Wagner. Using 3d laser range data for slam in outdoor environments. (October):188–193, 2004.

[10] F. Çakmak, E. Uslu, M. Balcılar, S. Yavuz, and M. F. Amasyalı. ROS Uyumlu Robot Platformu Gerçeklenmesi ROS Compatible Robot Platform Implementation. 2014.

[11] D. M. Cole and P. M. Newman. Using laser range data for 3D SLAM in outdoor environments. *Proceedings - IEEE International Conference on Robotics and Automation*, 2006(May):1556–1563, 2006.

[12] J. R. B. del Rosario, J. G. Sanidad, A. M. Lim, P. S. L. Uy, A. J. C. Bacar, M. A. D. Cai, and A. Z. A. Dubouzet. Modelling and Characterization of a Maze-Solving Mobile Robot Using Wall Follower Algorithm. *Applied Mechanics and Materials*, 446-447(July):1245–1249, 2013.

[13] G. Dissanayake, H. Durrant-whyte, and T. Bailey. A (slam). (April 2000), 2006.

[14] M. W. M. G. Dissanayake and R. A. Jarvis. A New Solution to the Simultaneous Localization and Map Building Problem. *Robotics and . . .*, 17(3):229–241, 2005.

[15] L. Fang, A. Fisher, S. Kiss, J. Kennedy, C. Nagahawatte, R. Clothier, and J. L. Palmer. Comparative evaluation of time-of-flight depth-imaging sensors for mapping and SLAM applications. *Australasian Conference on Robotics and Automation, ACRA*, 2016-Decem:285–291, 2016.

[16] FRANCESCO SABATINO. *Quadrotor control: modeling, nonlinear control design, and simulation*. PhD thesis, 2015.

[17] L. Freda and G. Oriolo. Frontier-Based Probabilistic Strategies for. (April):3892–3898, 2005.

[18] S. Grzonka, G. Grisetti, and W. Burgard. A fully autonomous indoor quadrotor. *IEEE Transactions on Robotics*, 28(1):90–100, 2012.

[19] C. Hegde and N. S. Guptha. Implementation of Mapping Algorithm for SLAM Operation. *Ijetae*, 3(9):235–238, 2013.

[20] HOKUYO. Scanning Rangefinder Distance Data Output/URG-04LX-UG01 Product Details | HOKUYO AUTOMATIC CO., LTD.

[21] N. Johnson. Vision-Assisted Control of a Hovering Air Vehicle in an Indoor Setting. *Engineering and Technology*, (August), 2008.

[22] M. Kara Mohamed, S. Patra, and A. Lanzon. Designing simple indoor navigation system for UAVs. *2011 19th Mediterranean Conference on Control and Automation, MED 2011*, pages 1223–1228, 2011.

[23] B. Kempke, P. Pannuto, and P. Dutta. SurePoint. pages 318–319, 2016.

[24] E. B. Küçüktabak, M. M. Pelit, Z. Ö. Orhan, and A. Emre. Kapalı Bir Alanda Basit Bir İHA ile Keşif Metodu Tasarımı Indoor UAV Exploration Method with UWB Localization. pages 1–6, 2017.

[25] H. Liu, J. Liu, P. Banerjee, and H. Darabi. Survey of Wireless Indoor Positioning Techniques and Systems. *Oftalmologia (Bucharest, Romania : 1990)*, 35(1):39–42, 1991.

[26] G. Mao, B. Fidan, and B. D. Anderson. Wireless sensor network localization techniques. *Computer Networks*, 51(10):2529–2553, 2007.

[27] J. Meyer, A. Sendobry, S. Kohlbrecher, and U. Klingauf. Simulation, Modeling, and Programming for Autonomous Robots. 7628(November), 2012.

[28] V. Nguyen, A. Harati, A. Martinelli, R. Siegwart, and N. Tomatis. Orthogonal SLAM: A step toward lightweight indoor autonomous navigation. *IEEE International Conference on Intelligent Robots and Systems*, pages 5007–5012, 2006.

[29] F. M. Noori, D. Portugal, R. P. Rocha, and M. S. Couceiro. On 3D simulators for multi-robot systems in ROS: MORSE or Gazebo? *SSRR 2017 - 15th IEEE International Symposium on Safety, Security and Rescue Robotics, Conference*, pages 19–24, 2017.

[30] H. I. M. A. Omara and K. S. M. Sahari. Indoor mapping using kinect and ROS. *2015 International Symposium on Agents, Multi-Agent Systems and Robotics, ISAMSR 2015*, pages 110–116, 2016.

[31] O. Oral, A. E. Turgut, and K. B. Arıkan. IHA ile GPS Kullanmadan Kapalı Alanların Haritasının Çıkartılması. *ToRK 2019 - Türkiye Robotbilim Konferansı*, 5(1):105–111, 2019.

[32] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira. Large-scale 6-DOF SLAM with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957, 2008.

[33] L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield. Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10965 LNAI:357–368, 2018.

[34] J. F. Roberts, T. S. Stirling, J.-C. Zufferey, and D. Floreano. Indoor Flight. *European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*, (September):17–21, 2007.

[35] R. Sinekli. Multi-Robot Simülatörü. 2013.

[36] P. Toivanen, V. Imani, and K. Haataja. Three main paradigms of simultaneous localization and mapping (SLAM) problem. (April):74, 2018.

[37] F. Wang, J. Cui, S. K. Phang, B. M. Chen, and T. H. Lee. A mono-camera and scanning laser range finder based UAV indoor navigation system. *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings*, pages 694–701, 2013.