REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SIMULTANEOUS LOCALIZATION & MAPPING APPLICATION ON UNMANNED AIREL VEHICLE (UAV)

HAYRETTİN ERTÜRK

MSc. THESIS
DEPARTMENT OF ELECTRIC & ELECTRONICS ENGINEERING
PROGRAM OF CONTROL & AUTOMATION ENGINEERING

ADVISER
ASSOC. PROF. DR. KAYHAN GÜLEZ

İSTANBUL, 2015

REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SIMULTANEOUS LOCALIZATION & MAPPING APPLICATION ON UNMANNED AIREL VEHICLE (UAV)

A thesis submitted by Hayrettin ERTÜRK's partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 25.06.2015 in Department of Electric & Electronic Engineering, Control & Automation Engineering Program.

Thesis Adviser

Assoc. Prof. Dr. Kayhan GÜLEZ

Yıldız Technical University

Approved By the Examining Committee

Assoc. Prof. Dr. Kayhan GÜLEZ

Yıldız Technical University _____

Assist. Prof. Dr. Janset DAŞDEMİR

Yıldız Technical University _____

Assist. Prof. Dr. Tufan KUMBASAR

İstanbul Technical University _____

ACKNOWLEDGEMENTS

Study both theory and implementation in academically works, requires an extraordinary work and motivation to complete a thesis. Without a motivation from the people besides me it was not possible to pass these tough times.

First of all I'm very grateful to my parents and my sister for their love and support to be a person and reach my goals. And I'm also grateful to my uncles that they showed me the value of knowledge based life and directed me to that.

With all my respect, I thank to my thesis advisor Assoc. Prof. Dr. Kayhan GÜLEZ and Research Assistant Dr. T.Veli MUMCU for their scientific motivation and keep me always look through forward. Their support kept my motivation to finish this thesis.

I also would like to thank my valuable friends Alper COPLUGİL and Olcan ŞİMŞEK for their materially and morally supports. Their friendship will always be valuable.

June, 2015

Hayrettin ERTÜRK

TABLE OF CONTENTS

Page

LIST OF SYMBOLS

$\vec{E}$	Earth Reference Frame
$\vec{B}$	Body Reference Frame
$\vec{\xi}$	Position of the Body Respect to Earth Frame
$\vec{\eta}$	Body Orientation Respect to Body Reference Frame
$\vec{\theta}$	Pitch Euler Angle
$\vec{\Phi}$	Roll Euler Angle
$\vec{\Psi}$	Yaw Euler Angle
$\overrightarrow{V^B}$	Translational Velocity Vector
$\overrightarrow{\omega^B}$	Rotational Velocity Vector
R_θ	Translational Rotation Matrix
T_θ	Transformation Matrix
τ	Torque
I	Inertia
$\vec{f}$	Force vector
$\vec{F}$	Force vector
V	Voltage
e	Electro-Motor Force
i	Current
P	Power
K_V	Motor Speed Constant
K_m	Motor Torque Constant
Ω	Angular Speed
J	Rotor Inertia
η	Efficiency
ρ	Air Density
A	Circular Area
$P(x)$	Probability of Variable x
x_k	Vehicle Position State
m	Map
Z_k	Observation at Time k
U_k	Input at Time k
σ	Variance
Q	Zero Mean Gaussian Noise
R	Zero Mean Gaussian Noise

P	Processes Error Covariance
H	Observation Transform Matrix
K	Kalman Gain
S	Innovation Matrix
$f(x,u)$	Non-linear State Function
$h(x)$	Non-linear Observation Function

LIST OF ABBREVIATIONS

UAV	Unmanned Airel Vehicle
UGV	Unmanned Ground Vehicle
PBVS	Pose Based Visual Servo
IBVS	Image Based Visual Servo
EKF	Extended Kalman Filter
PV	Photovoltaic
DOF	Degree-of-Freedom
COG	Center-of-Gravity
BLDC	Brushless DC Motor
PMBLDC	Permanent Magnet Brushless DC Motor
PWM	Pulse Width Modulation
PID	Proportional-Integral-Derivative
RPM	Revolution Per Minute
RMSE	Root Mean Squared Error
SLAM	Simultaneous Localization And Mapping
ROS	Robot Operating System
IMU	Inertial Measurement Unit
MCU	Microcontroller
GPS	Global Positioning System
IEEE	Institute of Electrical & Electronics Engineers
PPM	Pulse Position Modulation
MEMS	Micro Electro-Mechanical Sensor

LIST OF FIGURES

Page

ABSTRACT

SIMULTANEOUS LOCALIZATION AND MAPPING APPLICATION ON UNMANNED AIREL VEHICLES (UAV)

Hayrettin ERTÜRK

Department of Control & Automation Engineering

MSc. Thesis

Adviser: Assoc. Prof. Dr. Kayhan GÜLEZ

In view of today's technological achievements, robotics applications become more important. Increase in a information knowledge and extensive information sharing platforms and life speed nearly beyond the human capability. Within this speed in near future, daily time will not be enough for a mankind. Therefore man needs unmanned vehicles and robots to catch the time.

In this study, a review and an application of Unmanned Vehicle concept is realized. Simultaneous Localization and Mapping Algorithm (SLAM) is currently most important area especially on vehicles. There are various appoaches and implementatation techniques are available. In this work, SLAM Implementation with Unmanned Airel Vehicles (UAV) were studied. Rotary-Wing UAV Structure is reviewed in detail. Also Bayes theory Framework is reviewed to implement Unmanned Algorithm bases. Under Bayes theory, recursive bayes theory is reviewed. Kalman filter based and Particle filter based implementation techniques are discussed. Also Robot Operating System (ROS) reviewed in details. At the end a Particle filter based application is realized with Robot Operating System (ROS) using Gmapping Algorithm and results are discussed.

Key words: Unmanned Vehicle, Bayes Filter, Particle Filter, Mapping, Localization

ÖZET

İNSANSIZ HAVA ARACI (İHA) İLE EŞ ZAMANLI KONUMLAMA VE HARİTALAMA UYGULAMASI

Hayrettin ERTÜRK

Kontrol & Otomasyon Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Doç. Dr. Kayhan GÜLEZ

Günümüz teknolojik gelişmeleri ışığında, robotik uygulamaları gün geçtikçe daha da önemli hale gelmektedir. İnsanoğlunun bilgi birikimindeki hızlı artış, bu bilginin geniş paylaşım platformları ile yayılımı ve günlük hayat hızının artışı insan kapasitesininin sınırlarına doğru ilerlemektedir. Bu hız göz önüne alındığında yakın bir gelecekte günlük zaman dilimleri dahi insanoğluna yetmeyecektir. Bu ve benzeri sebeplerle insanoğlu zaman tasarrufu adına insansız araçlar ve robotlara ihtiyaç duyacaktır.

Bu çalışma kapsamında İnsansız Araç konsepti incelenmiş ve uygulaması gerçekleştirilmiştir. Eş Zamanlı Lokalizasyon ve Haritalama Algoritmaları, İnsansız Araç konseptinde en önemli alandır. Bu konsepte yönelik çeşitli teorik yaklaşımlar ve uygulama teknikleri mevcuttur. Bu tez kapsamında Eş Zamanlı Lokalizasyon ve Haritalamanın, İnsansız Hava Aracı (İHA) üzerinde uygulaması çalışılmıştır. İHA olarak döner kanatlı Quadrotor aracı kullanılmış ve detaylarıyla incelenmiştir. Ayrıca Bayes Teorisi ve Eş Zamanlı Lokalizasyon ve Haritalama Algoritmaları için döngüsel Bayes Teorisi incelenmiştir. Döngüsel Bayes teorisinin uygulamasına yönelik olarak Kalman Filtresi tabanlı ve Parçacık Filtresi tabanlı uygulama metodları tartışılmıştır. Uygulama platformu olarak Robot İşletim Sistemi (ROS) kullanılmış ve detayları incelenilmiştir. Son olarak parçacık filtresi tabanlı bir algoritma olan Gmapping algoritması kullanılarak ROS üzerinde uygulama yapılmış ve sonuçlar incelenmiştir.

Anahtar Kelimeler: İnsansız Araç, Bayes Filtresi, Parçacık Filtresi, Haritalama, Konumlama

CHAPTER 1

INTRODUCTION

1.1 Literature Review

UAV concept is an important technological field that will lead society life to another level. They have many usage areas and the concept will make human life easier. Due to this bringing UAV studies together to one piece is not possible. They have wide variety of usage areas.

For this reasons, researches focused and made important improvements in these different UAV studies. These studies focused on wide variety of fields like Advanced UAV Control Technique Implementations [1],[2],[3], Trajectory tracking methods [4], UAV coordination & cooperation [5], UAV Based Imaging & Monitoring [6],[7], Safety studies [8].

Figure 1.1 Potential threat detection over interest area using airel image obtained from an UAV [6]

UAV's have common usage in imaging and monitoring area. With UAV's it is possible to monitor large areas and scanning whether 2D or 3D images of this areas without human intervention. One of the most used methods for imaging with UAV is, servo controlled camera usage. Basically this method includes a camera mounted on two servo motors at 2-axis. However mounting this system to an UAV is not enough to take clear images. UAV needs a compensation movement so that camera become stabile and clear images can be obtained. Basically researchers focus on two mainstream control technique named Pose-Based Visual Servo (PBVS) and Image-Based Visual Servo (IBVS)[9]. In Image based technique, control realized with visual feature based errors, which are applied to actuators to maneuver on inverse direction of image's Jacobian Matrix[10].

In pose-based technique (PBVS), one needs to formulate the position and orientation estimation problem. In [11],[12], an example formulation of position and orientation estimation problem defined as a tracking problem and solved by using an Extended Kalman Filter (EKF).

Real-time crop management with image based monitoring techniques has been trending topic in last decade [13]. To implement a successful image based monitoring, high resolution images of field are necessary. Some of the studies showed implementations with satellite based monitoring. However these solutions lack of imagery with optimum

spatial and spectral resolutions. Also re-visit time of satellite is longer than desired time for most of the crop stress-detection applications. Usage of manned airborne platforms is possible but operating costs are high. Comparing with these application techniques, an UAV has a capacity of high spatial resolution monitoring, quick turnaround times and low operating costs [7]. Besides UAV usages, combined UAV and UGV work is also desired and studied by researches [14].

Soil moisture and salinity measurement in agricultural fields is another desired usage area of UAV's. Normally to observe soil moisture or salinity in agricultural fields, one can use probe based sensors. But with probe based sensors, measurement is local in field therefore global assumptions are made on this local data of the field. Consequently to make a successful detection of soil moisture, global measurement is necessary. One of the known methods to create a global soil moisture map is radiometer method. In general terms, a radiometer is sending radio on predefined frequency range and receives reflected waves. Mostly radiometers are deployed on airel vehicle [15]. But sometimes usage of ground based vehicles is possible [16]. However deploying a radiometer on airel vehicle is expensive. Hence some authors deployed radiometer on fixed-wing UAV's and successfully created soil moisture and salinity maps of agricultural fields[17].

Fighting with fire in large areas like forests is not easy and lethal for fireman. Ground vehicles cannot move as they like because of the terrain of the forest. An effective way of fire fighting in forest is blocking fire's movement direction. Therefore path planning is vital for fireman. In this situation, deployed UAV's over forest can take infrared images of area. Some researchers developed path planning algorithms basis on infrared images [18].

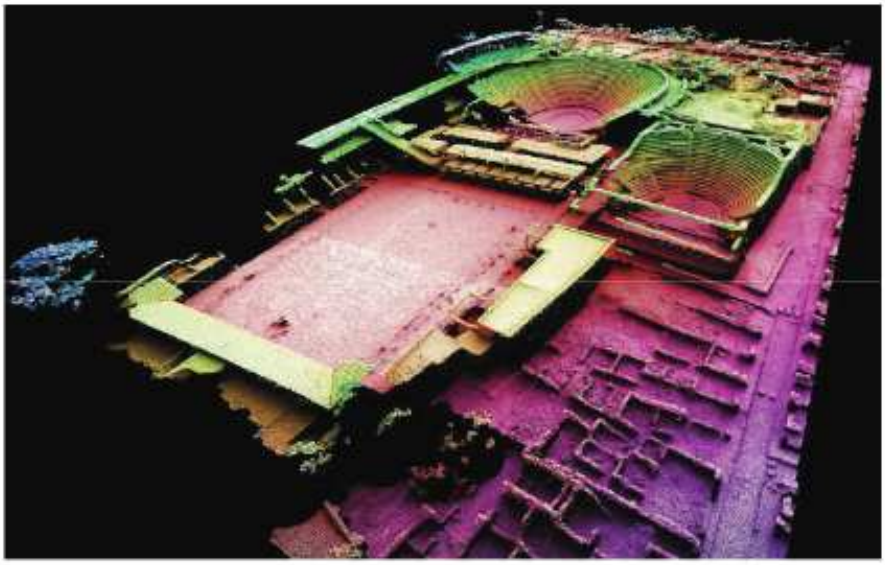

Figure 1.2 Dense point cloud of Pompeii theater which scanned using an UAV [19]

One of the UAV's unique usage area is archeological studies. UAV's are helpful at photogrammetric applications on archeological sites [19]. UAV based 3D imaging can help better evaluating archeological sites.

Traffic is a common problem in urban areas of advanced cities. It causes a lot of time and fuel waste. Traffic analysis plays a key role in solving the problem. Latest studies uses UAV's to evaluate traffic situation by tracking vehicles in real time [20].

Such interesting usage of an UAV is in the photovoltaic (PV) energy plants. At PV plants partial failure and performance detection is important. Different UAV's capable of thermal and regular imaging had used and showed that failure detection can be really easy with proposed methods [5].

It is clear that military applications have been leading role in UAV development history. UAV's used in military applications are managed by a human operator.

Figure 1.3 Nimbus UAV is monitoring PV Plant with thermal imagery and evaluates error conditions[5]

However fully automatic target detection and evaluation is much desired in such applications. In example vehicle detection is important topic in defense. Such study made on pipeline patrol service. Aim of the study was detect the vehicles which heading towards to pipeline and creating an alarm. An UAV is deployed with visual inspection capabilities and target detection made on the supplied data. As a result 85% of the vehicles detected and 1800 false alarms occurred during the one year work period[6]. Such studies play key role in detecting important targets, decrease non-target causalities in military operations.

Basically UAV's split to two main structural concepts. Fixed-wing vehicles and rotary-wing vehicles. Both of these structures have their unique physical models and control methods need to be implemented according to these models. One can use basic control methods like PID to control the vehicle in isolated conditions, but most of the time environment and design requirements are challenging and push researches to use more advanced control methods.

For example one of these challenging environments is landing to a moving platform. Assuming an UAV needs to be operated over ship or moving car, a complex and stabile algorithm is needed. As an example, in [3] controlling of an UAV over moving platform and automatic vertical landing onto that is investigated and optical flow sensor

based algorithm had been implemented. Within the approach, only optical flow data and inertial data (for flow de-rotation) are used. Neither linear velocity nor distance reconstruction is needed.

To accomplish tasks an UAV needs to track a trajectory. Trajectory tracking is one of the most important fields in UAV studies. However in general, a good trajectory tracking algorithm requires sensor fusion, observer models and different control algorithms. Authors made studies on robust control based tracking [1], neural network based tracking [2], Kalman filter based tracking [21] and fuzzy logic based tracking algorithms [4].

In many application coordinated work of UAV's are expected. Especially last years many studies are focused on this field. There are different ways of coordinated UAV work. As an example all UAV's in team can be distribute around a center point. But mostly distributed, decentralized team work is desired. An example, perimeter surveillance application with an expanding-contracting perimeter, requires decentralized team work is proposed [22]. With proposed algorithm one can monitor changing perimeters, account for dynamic insertion and deletion of team members, and operate with a small communication range in a decentralized manner. However algorithm is not efficient comparing to centralized algorithm.

UAV studies still in development phase. However it is clear that they will be used heavily on daily life in many applications at near future. Therefore safety regulations should exists on the usage of UAV. These concerns already motivated some researches. One of the considerations on the topic is interoperability of UAV's and manned airel vehicles. To avoid collisions, inter-compatibility plays important role. Also in development and manufacturing, software and hardware quality and safety standards applied to civil aviation industry should apply to UAV production phase [8].

1.2 Objective of the Thesis

When working with UAV's, most important fact is they do not carry humans to operate. Rather remote controlling of UAV is enough. But most of the time, on an UAV mission, it is desired an UAV to be autonome so that it can realize tasks without human intervention. As an example, on a disaster search and rescue mission, an UAV needs to look for targets, besides it needs to prevent itself from wreckages. It might be hard task for a human operator but if UAV has autonome capability, it would be very easy task to

handle. Also in such task, communication would not be possible all the time, for example when UAV needs to be search trough underground. In any case, autonomy is a necessary feature for an UAV.

To supply such feature to an UAV there are many different techniques exists. But most important topic between these techniques is environmental awareness. Before any act, vehicle needs to know its environment and its position at that environment. After this knowledge, any kind of movement strategy can be applied which is preconfigured at vehicles memory.

In last decades solutions to the environmental awareness problem is collected under "Simultaneous Localization and Mapping (SLAM)" problem. SLAM problem describes the environmentally awareness and self-location knowledge at that environment in the same domain. It is important to understand SLAM problem to implement it on UAV.

1.3 Hypothesis

SLAM algorithm in its basic form, maps the environment and gives the exact location of a vehicle at that environment. Implementing a SLAM algorithm will give self-location and environmental awareness to an UAV. In this work also implemented Gmapping SLAM algorithm on an UAV which is a Quadrotor showed that algorithm gives this location-environment awareness to UAV.

CHAPTER 2

UAV KINEMATICS, DYNAMICS & EXCITATION

Fixed wing and rotary wing UAV's have different kinematic models. Fixed-wing UAV has well understood kinematics. For this reason this study will focus on rotary wing UAV kinematics especially on Quadrotor that is a system with four rotors.

Quadrotor's flight behavior is controlled by rotation speeds of its own four rotors. Therefore to design a good controller and change the position or orientation of Quadrotor, relation between inputs and mathematical model need to be described. A well described model includes coordinate frames, translational and rotational kinematics and dynamics related to desired inputs.

2.1 Coordinate Systems

Before go further in kinematics and dynamics of Quadrotor, coordinate frames, references and transformation between these frames need to be clearly understood. There are two basic form of reference frames are necessary to explain six dimension of freedom (6DOF) kinematics of Quadrotor. A reference inertial frame located on earth (ground) which is known as earth reference frame. Let describe the earth reference frame $\vec{E} = \{E_x, E_y, E_z\}$ fixed with respect to earth. Another reference frame is body fixed frame which is located at center of gravity (COG) of Quadrotor. We can define the body fixed frame $\vec{B} = \{B_x, B_y, B_z\}$.

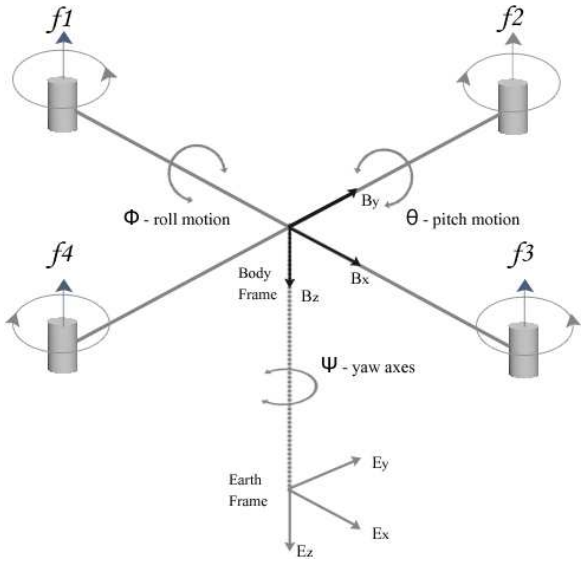

Figure 2.1 Reference Frames of Quadrotor

Using this coordinate frames we can described the kinematics of Quadrotor in six degrees of freedom (6DOF). As example we can describe the translational displacement vector with transformation from $\vec{E}$ to $\vec{B}$.

2.2 Kinematic Equations

As we approach in six degrees of freedom (6DOF). There are two types of state vectors describe the translational and rotational kinematics [24]. Let the vector $\vec{\xi} = \{\xi_x, \xi_y, \xi_z\}$ be the position of quadrotor respect to earth reference frame $\vec{E}$. Then we can describe the body orientation vector respect to body fixed frame $\vec{B}$ as $\vec{\eta} = \{\Phi, \theta, \Psi\}$. In general this vector describes roll, pitch and yaw motions respectively. Orientation vector's angles are defined as euler angles. These angles are bounded as follows:

$$\left\{\Phi \,\middle|\, -\frac{\pi}{2}, \Phi, \frac{\pi}{2}\right\} \quad (2.1)$$

$$\left\{\theta \,\middle|\, -\frac{\pi}{2}, \theta, \frac{\pi}{2}\right\} \quad (2.2)$$

$$\{\Psi \mid -\pi, \Psi, \pi\} \quad (2.3)$$

To extract translational kinematics let us define a $\overrightarrow{V^B} = \{u, v, w\}$ vector which is translational velocity vector of vehicle respect to body fixed frame $\vec{B}$. Assuming $\dot{\vec{\xi}}$ is derivative of position vector respect to earth frame we can write the translational motion as:

$$\dot{\vec{\xi}} = R_\theta \overrightarrow{V^B} \tag{2.4}$$

Through this equation we can find the translational rotation matrix $R_\theta \epsilon R^{3x3}$ Using the vector equations we can write R_θ as :

$$R_\theta = \begin{bmatrix} cos\Psi cos\theta & cos\Psi \sin\theta + sin\Psi sin\theta cos\Phi & -cos\Psi \sin\Phi + sin\Psi sin\theta cos\Phi \\ sin\Psi cos\theta & cos\Psi \cos\Phi + sin\Psi sin\theta sin\Phi & -cos\Psi \sin\Phi + sin\Psi sin\theta cos\Phi \\ -sin\theta & cos\theta sin\Phi & cos\theta cos\Phi \end{bmatrix} \tag{2.5}$$

Then explicit relation between vehicle speed respect to earth frame as follows:

$$\begin{bmatrix} \dot{\xi}_x \\ \dot{\xi}_y \\ \dot{\xi}_z \end{bmatrix} = \begin{bmatrix} cos\Psi cos\theta & cos\Psi \sin\theta + sin\Psi sin\theta cos\Phi & -cos\Psi \sin\Phi + sin\Psi sin\theta cos\Phi \\ sin\Psi cos\theta & cos\Psi \cos\Phi + sin\Psi sin\theta sin\Phi & -cos\Psi \sin\Phi + sin\Psi sin\theta cos\Phi \\ -sin\theta & cos\theta sin\Phi & cos\theta cos\Phi \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.6}$$

Let $\overrightarrow{\omega^B} = \{p, q, r\}$ vector is a rotational velocity vector of vehicle respect to body fixed frame $\vec{B}$. In this case the relation between derivative of body orientation vector $\vec{\eta}$ and $\overrightarrow{\omega^B}$ as follows:

$$\dot{\vec{\eta}} = T_\theta \overrightarrow{\omega^B} \tag{2.7}$$

With this knowledge we can extract the transformation matrix T_θ as follows:

$$T_\theta = \begin{bmatrix} 1 & \sin\Phi\, tan\theta & \cos\Phi\, tan\theta \\ 0 & cos\Phi & -sin\Phi \\ 0 & \frac{sin\Phi}{cos\theta} & \frac{cos\Phi}{cos\theta} \end{bmatrix} \tag{2.8}$$

Then the explicit form can be written as rotational relation of vehicle respect to its own frame as follows:

$$\begin{bmatrix} \dot{\Phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\Phi\, tan\theta & \cos\Phi\, tan\theta \\ 0 & cos\Phi & -sin\Phi \\ 0 & \frac{sin\Phi}{cos\theta} & \frac{cos\Phi}{cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.9}$$

Using these kinematic relations we have the necessary information to create dynamic models of vehicle.

2.3 Dynamics

To understand dynamics movements of Quadrotor; rigid body dynamics of the Quadrotor frame needs to be clearly stated. Through definition, motion equations to create the dynamic model [25] can be written. Also it can be divided as translational dynamics and rotational dynamics. Using this technique each parts of dynamic movement can be studied as isolated.

2.3.1 Rotational Dynamics

Dynamic model implementation of Quadrotor relies on Newton's second law's rotational form which is:

$$\tau = I\dot{\omega}^{B} \tag{2.10}$$

Before explain acting moments on body, inertia matrix of Quadrotors needs to be explained. Vehicle body is three dimensional so that inertia matrix of the body frame can be written as follows:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{2.11}$$

Due to symmetrical structure of rigid body frame and aligned construction in x and y axes, inertia in cross axes are zero and inertia matrix is diagonal and as follows:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.12}$$

2.3.1.1 Roll & Pitch Motions

Assuming free body diagram figure X, length of frame in x and y axes are equal and let us define as l.

With these assumptions, the acting moment on different axes can be defined. Let us start with rolling moment acting on x-axes:

$$\tau_{xx} = I_{xx}\dot{\omega^{B}} \quad (2.12)$$

Quadrotor's main frame on both x-axes and y-axes aligned around center of gravity (COG) thus rolling toque around x-axes τ_{xx} is created by thrust difference of rotors. Referencing the free body diagram, motors M2 and M4 are on the x-axes. This condition leads to following presentation:

$$I_{xx}\ddot{\theta} = l\,(f_2 - f_4) \quad (2.13)$$

Considering y-axes is the same configuration and motors M1 and M3 are on the y-axes. Then pitching motion equation around y-axes become:

$$I_{yy}\ddot{\Phi} = l\,(f_1 - f_3) \quad (2.14)$$

2.3.1.2 Yaw Motion

Yaw motion is around z-axes. Therefore total torque around z-axes determines the yaw motion. Let us start with a torque created by a rotor around its axes. Every rotor creates a moment around its turning direction and we can define it as follows:

$$\tau_{Mi} = b\omega_i^2 + I_M\dot{\omega}_\iota \quad (2.15)$$

Here b is a drag constant of propeller and I_M is inertia of rotor around its axes. In general $\dot{\omega}_\iota$ is small and therefore negligible.

To remark, τ_{Mi} is moment of rotor around its turning axes as can be shown is figure 2.1. Quadrotor's motion model has lot of moment definition and one need to understand each of them very well to learn the system.

Back to yawing motion again, the torque of a rotor already clarified at eq. 2.15. Then total torque around z-axes can be expressed sum of all rotors torques as follows:

$$\tau_\Psi = \sum_{i=1}^{4} \tau_{Mi} \quad (2.16)$$

For a balanced condition, total torque τ_ψ need to be zero. Due to symmetric configuration with a proper setup of turn directions, zero moment around z-axes can be obtainable. Let us consider figure x. Motors on the x-axes M2 and M4's rotors need to turn same directions. In this case on y-axes turning directions of M1 and M3's rotors must opposite direction of x-axes motors turning direction. This kind of setup cancels out the opposite torques and τ_ψ becomes zero as follows:

$$I_{zz}\ddot{\Psi} = \tau_{f2} + \tau_{f4} - \tau_{f1} - \tau_{f3} \quad (2.17)$$

According to this equation, changing each rotors torque it is possible to control yaw motion.

2.3.2 Translational Dynamics

Newton's second law also can be used to explain the translational dynamics. Let us define the law for translational movements:

$$\mathrm{F} = m\dot{V^B} \quad (2.18)$$

where m is the mass of quadrotor and $\dot{V^B}$ is derivative of translational velocity vector respect to body fixed frame.

To explain the motions in three dimensional translational spaces, one needs to be looking for affecting forces on system. Assuming figure 2.1, we have the thrust force in z-axes direction created by four rotors. In this case the force vector in three dimensional spaces can be defined as follows:

$$\acute{\mathrm{F}} = \{0, 0, u\} \quad (2.19)$$

where u is thrust force.

Now necessary information to find out effecting forces in three dimensions is gathered. By using the kinematic translation matrix defined at (5) the translational motion equations can be written as follows:

$$m\ddot{\xi}_x = u(sin\Psi sin\theta cos\Phi - cos\Psi \sin\Phi) \quad (2.20)$$

$$m\ddot{\xi}_y = u(sin\Psi sin\theta cos\Phi - cos\Psi \sin\Phi) \quad (2.21)$$

$$m\ddot{\xi}_z = ucos\theta cos\Phi - mg \quad (2.22)$$

where g denotes the gravitational force.

2.4 Excitation Modeling

UAV's can be excited by different sources like internal combustion engine, jet engine or electrical engine. For fixed wing UAV's combustion engines are preferred since these types do not need fast maneuver to be stabile and engine response time is enough.

However rotary-wing UAV's generally needs fast maneuver to be stabilize especially Quadrotor. In this case, electric motor has enough response time for fast maneuvers. Due to this reason, Quadrotor built with DC Electric Motor based excitation system.

Although mostly electrical motor based excitation systems are used on Quadrotor, there is one disadvantage comes through. Flight time become limited due to energy density of electrical batteries. Comparing to gasoline based combustion engines; electrical batteries still much lower electrical density. Because of this, flight time is still limited on electrical engine based Quadrotor.

To properly excite a Quadrotor, excitation mechanism and physical transformations needs to be well defined. Due to this reason, electrical-mechanical relation and electrical transfer function of motor needs to be known. To understand and extract the transfer function, one needs to understand DC electrical motor and its characteristics.

2.4.1 DC Motor Types

DC Motor, in its simple form, a machine that converts electrical energy to mechanical energy or in the reverse direction. A DC Motor has two main parts called stator and rotor. Stator is the fixed part of the machine. Stator supplies the main electromagnetic field required by DC Motor with permanent magnets or with electrical cable windings which generate electromagnetic field. Other part named rotor is a rotate-free part. Rotor is coupled to main mil of the machine and it can rotate around stator. Basically applying

DC Voltage to rotor windings, a rotary electromagnetic field will be created around rotor and interaction of this field with stator's field will result with electo-mechnically generated force. As a result of these mechanical forces, rotor will start the turn and keep this state as long as voltage applied to rotor.

However not all electrical motors have the same configuration. In recent years, a motor named Brushless DC Motor (BLDC) is becoming popular. In BLDC's rotor has the permanent magnets and stator has the copper windings. Directly applying voltage to all windings is not possible BLDC's. Voltage must be applied to windings in a sequence to rotate the rotor. One of the main advantages of BLDC's is high torque density. Due to this specification, there is not any gear system is needed. Direct drive becomes possible.

2.4.2 DC Motor Dynamics

2.4.2.1 DC Motor Electrical Analysis

Whether standard DC Motor or BLDC Motor, an electrical motor can be modeled with simple resistor, capacitance and inductance load models [25]. Due to geometry of DC Motor, inductance is much higher compared to capacitance as a result capacitance can be eliminated in model.

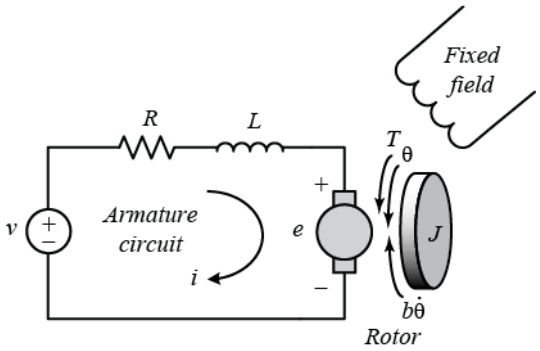

Figure 2.2 DC Motor Electro-Mechanical Model [41]

To model the motor control interface we need to add another voltage source. This source V can be seen on Fig 2.2**.** In real conditions, control voltage source has resistance, inductance and capacitance. But since we use this source as a controller, these values should be low enough to drive efficiently so that we can eliminate in model.

By referencing Fig 2.2, we can write the electrical equation of the motor model as follows:

$$V = V_R + V_L + e \quad (2.23)$$

Where V_R and V_L are voltage across resistor and inductor respectively. And e is the back emf (electromagnetic field).

Considering back-emf e is proportional to angular velocity of shaft, it can be written as follows:

$$e = K_v\Omega \quad (2.24)$$

Here K_v known as speed constant and Ω is angular speed. Since the voltage equation across a resistor and inductor are obvious, model can be written explicitly:

$$V = Ri + L\frac{di}{dt} + K_v\Omega \quad (2.25)$$

In explicit form, i is current in amperes, Ω is motor's angular speed in radian/s and finally K_v is motor constant in (volt*s)/radian unit. K_v is a result of motor structure and cannot be changed in the model. It is clear to see that motor speed is directly related to voltage.

Due to inductance effect is small on the motors used; inductance can be eliminated from the model. Then the equation becomes:

$$V = Ri + K_v\Omega \quad (2.26)$$

Also eq. 2.26 can be written in terms of current which will be used later on:

$$i = \frac{V - K_v\Omega}{R} \quad (2.27)$$

2.4.2.2 DC Motor Mechanical Analysis

Controlling a DC Motor is basically controlling the load with input voltage where load is coupled to motor's shaft. Therefore load relation and the basis electro-mechnical

transformations needs to be well understood. First of all motor's torque with electrical terms needs to be defined as follows:

$$\tau_M = K_M i \qquad (2.28)$$

Here it can be seen that motor torque τ_M is proportional to i which is current on windings with a constant K_M in Nm/Amperes unit. This definition is raw torque definition of DC Motor's rotors self torque. In order to rotate the rotor following rotational Newton equation should be considered:

$$J\dot{\Omega} = \tau_M - \tau_L \qquad (2.29)$$

This equation simply explains the mechanical relation between rotor and external load. If generated torque τ_M is greater than load torque τ_L and inertial torque, then rotor will start to rotate. Well here there is in important point. If the vice-versa is thought and electrical equations are considered then if greater load torque is applied than motor torque to rotor, it will transform mechanical energy to electrical energy. In this case motor will be "generator". This is the main reason why DC Motor's called electro-mechanical transformation machines. But in this thesis only motor mode is in interest.
Finally, electrical terms can be placed into load equations as following:

$$J\dot{\Omega} = K_M i(t) - \tau_L \qquad (2.30)$$

As it seems on equation mechanical load is directly connected to current applied. If we like to go further and put the current i term to it's place, voltage related equation can be written as follows:

$$J\dot{\Omega} = -\frac{K_M K_v \Omega}{R} - \tau_L + \frac{K_M V}{R} \qquad (2.31)$$

2.4.3 Motor Voltage & Propeller Angular Velocity Relation

To lift a Quadrotor from the ground a propeller needs to rotate and flow the air through ground. This enables the necessary force for Quadrotor to lift. In this described excitation system the propeller is coupled to electrical DC Motor's shaft.

Various load types various characteristics. In this case, a propeller exists which is a propeller-type load and it has a characteristic as follows:

$$\tau_L = D\Omega^2 \qquad (2.32)$$

This equation explains the relation between shaft speed also propeller speed. Load torque generated by propeller, is related to square of its speed. It means at higher speeds there will be much more load then lower speeds.

If eq. 2.32 placed into the eq. 2.31 than full relation between voltage and angular velocity by means of motor and propeller parameters will be as follows:

$$V = \frac{RD\Omega^2}{K_M} + \frac{RJ\dot{\Omega}}{K_M} + K_V\Omega \qquad (2.33)$$

2.4.4 Motor Voltage & Thrust Relation

Although in electrical motors, force is directly related to current, we can keep the voltage-thrust relation in order to make controlling easier. So current i from eq. 2.28 needs to be extracted as follows:

$$i = \frac{\tau_M}{K_M} \qquad (2.34)$$

Also power equations to find out voltage – thrust relation is needed. Let be start by calculating the electrical power as follows:

$$P = IV = \frac{\tau_M}{K_M}V \qquad (2.35)$$

Mechanical power output by means of efficiency of electrical power output can be explained as follows:

$$P_M = \eta P = \eta \frac{\tau_M}{K_M} V \quad (2.36)$$

To keep the relation between induced power to the output power on propellers one needs to define a merit f:

$$f = \frac{P_H}{P_M} \quad (2.37)$$

Placing mechanical output power to eq. 2.36 and re-arrange:

$$P_H = \eta f \frac{\tau_M}{K_M} V \quad (2.38)$$

Induced power also can be written as follows:

$$P_H = F v_H \quad (2.39)$$

If we equalize equation 2.38 and 2.39:

$$\eta f \frac{\tau_M}{K_M} V = F v_H \quad (2.40)$$

This equation is the key for the voltage-thrust relation but the air speed needs to be explicitly written using momentum theory:

$$v_H = \sqrt{\frac{F}{2\rho A}} \quad (2.41)$$

where ρ is air density and A is circular area covered by rotating propeller.

Let us define torque by means of F with a constantk_i:

$$\tau_M = k_i F \quad (2.42)$$

If we substitute air speed into equation 2.40 then relation of voltage with thrust will be as follows:

$$F = 2\rho A\left(\frac{\eta f k_i}{K_M}\right)^2 V^2 \qquad (2.43)$$

2.5 Simplified Excitation Modeling

When modeling a Quadrotor excitation system, it is good to use if all parameters of motors are known. However, most of the times, motor's parameters are not well known. Especially for small UAV's, motors are produced by hobby shops. So there is not much technical documents exists. Also to test the motor or propeller to find the parameters not always available and setup test bed for each of them is time consuming.

To overcome this problem, a simplified version of excitation model can be designed as a state of art. Designing such system, first of all requires understanding the place of excitation in Quadrotor control chain.

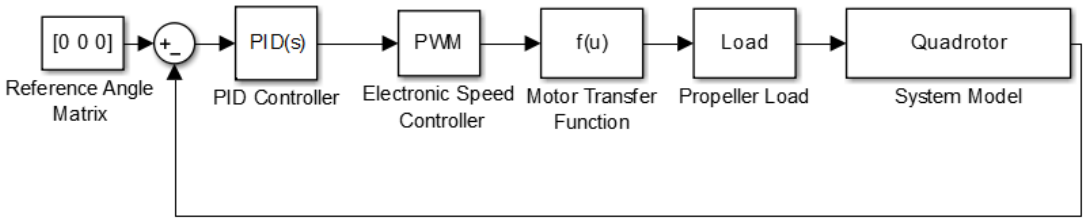

Figure 2.3 Block Diagram of a Quadrotor PID control model with all components

In figure 2.2, full control chain can be seen. In its basic form a PID controller for the system can be assumed. PID controller outputs to a motor voltage driver which is PWM technique based driver in our case and digitally driven. After that driver applies generated output voltage with PWM technique to motor that is modeled as a transfer function. After voltage input to motor transfer function a load force generated at propellers which enables Quadrotor to be controlled.

Now if the control chain is reviewed, it can be seen that whole excitation can be simplified to PWM input – Thrust (or motor shaft angular speed) output. One needs to find the PWM-Thrust and PWM-Angular Speed relations as an equation to use in

model.

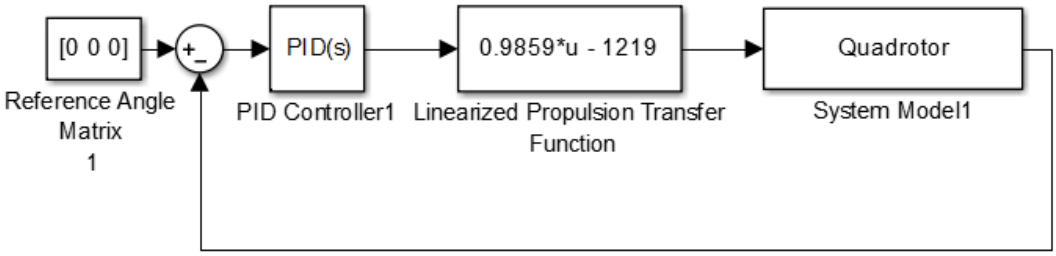

Figure 2.4 Block Diagram of a quadrotor PID control model with linearized propulsion transfer function

For these purposes a specific test bed is prepared that can be seen on figure 2.4. This test bed basically consists of a tool to measure thrust and laser tachometer to measure angular speed of propeller. For PWM driver servo PWM driver is used. Servo PWM driver is specific driver developed by hobby communities and producers. Servo Pulse Width is basically a PWM Signal but it has a 20ms of period and a pulse width between 1ms and 2ms. By changing pulse width between this gap it is possible to change driver's output PWM state.

Figure 2.5 Test setup for simplified excitation models

Test procedure is realized with digital input application to PWM drivers and collecting data from thrust measurement and laser tachometer. Once the procedure is started,

control command applies to PWM driver step by step. In each step start pulse width 1000us increased by 10us. Each step, thrust is measured as gram and angular speed measured as RPM. After data collection to extract a linear transfer function both for Servo Pulse Width-Thrust and Servo Pulse Width -Angular speed, linear regression is used. .

The measured data are imported to the curve fitting toolbox in relation of Servo Pulse Width vs. Thrust and Servo Pulse Width vs. Angular speed. After this step, the linearized functions are obtained as the result of linear regression operation.

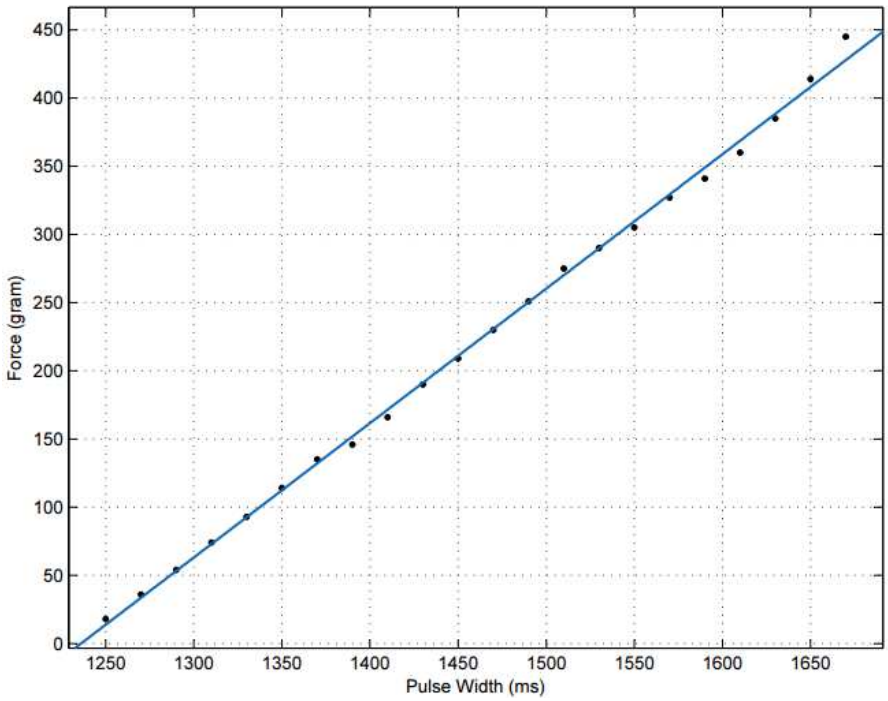

Figure 2.6 Pulse Width – Force relation measurement and linearized function (Dots: Real data, Line: Linearized function)

In Figure 2.5 the measured data for pulse-width vs. thrust force relation and linear transfer function extracted from this data can be seen.

As the measured data itself has a linear characteristic, the error of this linear regression is also small. The linearized function as follows:

$$f(x) = 0.9859x - 1219 \quad (2.44)$$

The relative mean square error (RMSE) in result of this operation is 5,642.

Figure 2.6 represents pulse width-angular speed relation, and the linear transfer function is (4),

$$f(x) = 8.712x - 8546 \quad (2.45)$$

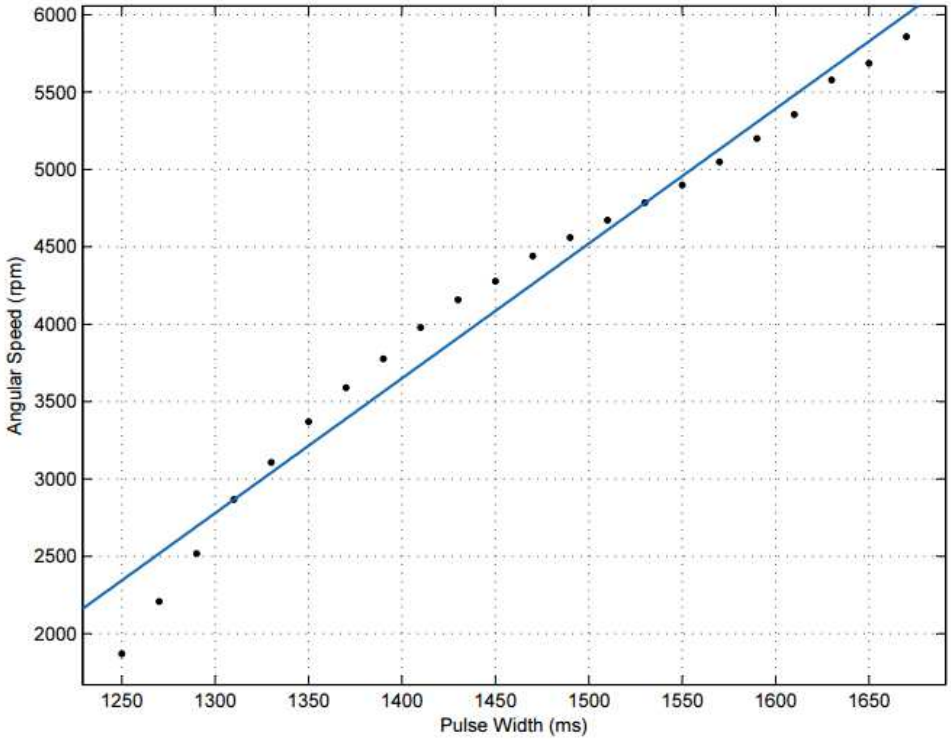

Figure 2.7 Pulse Width – Angular Speed relation measurement and linearized function (Dots: Real data, Line: Linearized function)

The relative mean square error (RMSE) in result of pulse width-angular speed relation is 194, 4.

CHAPTER 3

SLAM

3.1 Theoretical Background of SLAM

SLAM (Simultaneous Localization and Mapping), by definition is a technique to use map an unknown environment and self-localize on that map of an moving target. SLAM is active research field in recent years. From autonomous vehicles to miniature robots, every moving autonome system needs to know the environment and where itself in that environment. Not only for vehicle or robots, also other research fields like archeology, sub-sea monitoring uses SLAM to study on those environments.

SLAM problem has a duality in itself. To map an unknown environment by a vehicle, vehicle needs to know its position. As the same to know its position vehicle needs to see environment map. A solution proposed for SLAM should remove this duality and propose a solution for both mapping and localization problem. Otherwise proposed method will be a solution for only mapping or only localization.

For long time studies kept going on this direction and researches focused only mapping or only localization problems. They thought all errors on landmark and vehicle positions are independent and correlations should be minimized. But afterwards it has been understood that vehicle position errors and landmark estimations have high degree of correlations and this leads to a single solution. Then it has been realized that SLAM problem should be covered in single estimation problem rather than single mapping or localization problem. A study published in International Symposium on Robotics Research at ’95 defined this single estimation problem for the first time.

3.2 Probabilistic Approach

Theoretical solution of SLAM includes probabilistic estimation based methods. In a general form, every sensor or measurement tool have some degree of error and uncertainty margins. So in this perspective, as an example, it is not possible to say a variable "X" is at exactly location "Y". Rather it is more realistic to say variable "X" is has 90% percent of probability to be at location "Y".

Probabilistic approach gives ability to probabilistic definition of errors and uncertainties too. Up to these reasons SLAM concept's solution have been defined under coverage of probability theory.

At probabilistic SLAM's theoretical solution, formulization of solution is as follows:

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) \qquad (3.1)$$

In this definition, x_k refers to vehicle position, m refers to map, $Z_{0:k}$ refers to observations from beginning to time, $U_{0:k}$ refers to control commands which applied to vehicle from beginning to time k and x_0 refers to beginning position of vehicle. This definition gives the probability distribution of vehicle location x and map m. Within this definition there are no exact locations are exists in solution rather probabilistic distributions.

By definition of SLAM, aim is collect and analyzes the data instantly. Due to that, the algorithm should not analyze a collected data, rather than algorithm should run on instant data. So algorithm structure should not rely on bulk data process. Algorithm should have relied on a recursive process that analyze most recently collected data and calculate the probability distribution.

To fulfill these requirements, a special form of Bayes Theory that is Recursive Bayes Theory has been proposed as a general framework solution. It should be noted that probabilistic solution proposed for SLAM is a general framework and application method can vary as long as satisfy the rules of framework.

3.3 Bayesian Algorithm Framework

Let be start with some basics of bayes theory. Let be define two events A and B. Then basic joint probability of these events would be:

$$P(AB) = P(A|B)P(B) \quad (3.2)$$

or,

$$P(AB) = P(B|A)P(A) \quad (3.3)$$

Considering this equality we can re-arrange and write the bayes theory formulation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (3.4)$$

This formulation shows the probability distribution of event A after consideration of event B. To be precise, it should be known that $a\ prior$ knowledge of event A is taken account into event B and $a\ posterior$ knowledge of event A is find out. For this reason $P(A|B)$ named as $posterior$ and $P(A)$ named as $prior$.

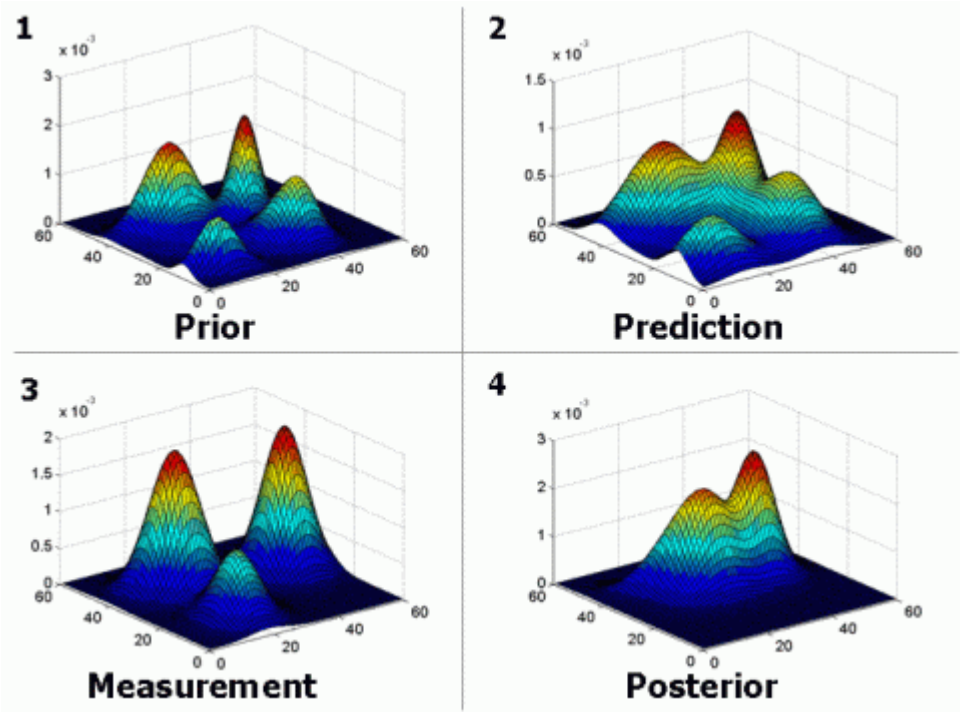

Figure 3.1 3D Gaussian vizualization of Bayes Process [26]

Let expand the definition with an example related to topic. Let be event A is a solid landmark lies on one dimension space and event B is observation of this landmark's

position. Where observation is actually the position of landmark with an error of n as follows:

$$B = A + n \tag{3.5}$$

Here n is Gaussian noise with zero mean. Let go further with definition. One needs to find out $P(B|A)$ which is joint distribution of observations over landmark positions. Also named as likelihood. If we write explicitly:

$$P(B|A) = P(n + A|A) \tag{3.6}$$

This gives exactly the distribution of noise n (Since in our case noise is physical world definition of likelihood between quantities). Noise is already assumed as Gaussian, so that following distribution can be written:

$$P(B|A) = \frac{1}{\sqrt{2\pi\sigma_n}} e^{-\frac{(B-A)^2}{2{\sigma_n}^2}} \tag{3.7}$$

Noise model is defined, but according to bayes theory one need to define $a\ prior$ knowledge in this case past information on landmark positions which is$P(A)$. Let us assume the mean of the landmark position is 10cm and has a variance of 1cm.Than one can fit this information to a Gaussian function (It should be remembered that it could be other function rather than Gaussian but it is easier to show the distribution this way). Then $P(A)$ will be:

$$P(A) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(A-10)^2}{2}} \tag{3.8}$$

Now using these two Gaussian, $a\ posterior$ can be found. In other words, the probability distribution of landmark position estimation based on prior knowledge and recent observations can be specified. To be exact, $P(A|B)$ will be:

$$P(A|B) = e^{-\frac{1}{2}\left[\frac{(B-A)^2}{{\sigma_n}^2} + (A-10)^2\right]} \tag{3.9}$$

Over these probability distributions, to find out “the best guess” of possible landmark positions, maxima of the distribution should be found:

$$\hat{A} = argmax_x P(A|B) \quad (3.10)$$

This simple one dimension landmark position estimation process actually the most basic definition of SLAM and the explanation of why bayes theory is proposed for the probabilistic solution. Simply it gives the most possible position for the landmark.
However one thing should not be overlooked. Raw usage of bayes theorem assumes a prior knowledge set is available. But it has already examined that SLAM solution should be done within most recent data, not with the past bulk data.
To overcome this situation, a special form of bayes theorem which is recursive bayes theorem has been proposed to SLAM solution.
Let be define the recursive bayes theorem. But first let define a set of observations $z = \{z_0,\ldots, z_k\}$ on state x. One needs to make two assumptions when changing bayes filter to recursive bayes filter. These are:

- States on stochastic process, must follow a first order Markov process

$$P(x_k|x_{0:k-1}) = P(x_n|x_{n-1}) \quad (3.11)$$

- Observations z must not have any dependency to states x.

Then we can start with writing basic bayes theorem and re-arrange the equations beyond the knowledge of previous assumptions for recursive process:

$$P(x_k|z_k) = \frac{P(z_{0:k}|x_k)P(x_k)}{P(z_{0:k})} \quad (3.12)$$

$$P(x_k|z_k) = \frac{P(z_k, z_{0:k-1}|x_k)P(x_k)}{P(z_k, z_{0:k-1})} \quad (3.13)$$

$$P(x_k|z_k) = \frac{P(z_k\ |z_{0:k-1}, x_k)P(z_{0:k-1}|x_k)P(x_k)}{P(z_k, z_{0:k-1})P(\ z_{0:k-1})} \quad (3.14)$$

$$P(x_k|z_k) = \frac{P(z_k\ |z_{0:k-1}, x_k)P(x_k|z_{0:k-1})P(\ z_{0:k-1})P(x_k)}{P(z_k, z_{0:k-1})P(\ z_{0:k-1})P(x_k)} \quad (3.15)$$

$$P(x_k|z_k) = \frac{P(z_k|x_k)P(x_k|z_{0:k-1})}{P(z_k, z_{0:k-1})} \quad (3.16)$$

In this formulation, bayes theorem became recursive bayes formulation. Let us redefine the basic definitions of bayes formulation with the knowledge of recursive formulation. It is obvious that $P(z_k|x_k)$ represents observation model. Than prior will be:

$$P(x_k|z_{0:k-1}) = \int P(x_k|x_{k-1})\, P(x_{k-1}|z_{0:k-1})\, dx_{k-1} \quad (3.17)$$

Here $P(x_k|x_{k-1})$ represents the state transition probability density which is one of the basic part of the probabilistic SLAM framework. Also it should be noted $P(x_{k-1}|z_{0:k-1})$ the previous posterior is integrated with state transition model to generate the current prior.

And evidence in other words, the normalization constant $P(z_k, z_{0:k-1})$ can be expanded as following:

$$P(z_k, z_{0:k-1}) = P(z_k|x_k)P(x_k|z_{0:k-1})\, dx_k \quad (3.18)$$

As can be seen, recursive bayes theorem is a general probabilistic framework for SLAM implementations. There are two important probabilities, “observation” and “state transition” is the heart of this framework. Since any implementation technique under this framework should apply these models.

Let us extend the recursive bayes theorem in terms of vehicle location state x, map state m, landmark observations z and control inputs u. Then the observation model can be written as:

$$P(z_k|\ x_k, m) \quad (3.19)$$

And motion model can be defined in terms of vehicle location and control inputs:

$$P(x_k \mid x_{k-1}, u_k) \quad (3.20)$$

An assumption of markov process already defined and the vehicle location x_k only depends on x_{k-1} so it is valid to use this motion model.

Considering the structure of recursive bayes theorem, it can divide into two main parts those Time Update and Measurement Update.

As it is already defined that prior of recursive bayes theorem is actually the time update-part. If we re-arrange in subject to SLAM, than:

$$P(x_k, m|z_{0:k-1}, u_{0:k}, x_0) = \int P(x_k \mid x_{k-1}, u_k)\, P(x_{k-1}, m|z_{0:k-1}, u_{0:k-1}, x_0)\, dx_{k-1} \quad (3.21)$$

Then the posterior calculation of recursive bayes theorem will be the result of measurement updates. Let us re-arrange general formulation in subject to SLAM to represent the measurement update as follows:

$$P(x_k, m|z_{0:k}, u_{0:k}, x_0) = \frac{P(z_k|x_k, m)P(x_k, m|z_{0:k-1}, u_{0:k}, x_0)}{P(z_k \mid z_{0:k-1}, u_{0:k})} \quad (3.22)$$

This is the exact representation of Online SLAM. However it is also possible to split online SLAM solution for offline SLAM solutions. In case of all possible vehicle locations are already known with beginning position, probability density of map can be written as :

$$P(m \mid z_{0:k}, u_{0:k}, x_{0:k}) \quad (3.23)$$

Also in case of all landmarks are deterministic on map, than whole SLAM problem can be reduced to localization problem as follows:

$$P(x_k \mid z_{0:k}, u_{0:k}, m) \quad (3.24)$$

3.4 Implementations of SLAM Framework

3.4.1 Kalman Filter Based SLAM

Since recursive bayes theorem only a general framework, there can be different possible implementation techniques. One of and the most known bayes filter implementation is Kalman Filter.

Kalman Filter is introduced by Rudolf Kalman at late 1950's [27]. Ever since it has been used in a wide range of application fields scales from space to defense industry. Kalman filter is an optimal estimator for a state if all errors are zero mean Gaussian errors. In a case like this Kalman filter will give the best estimate for squared mean error.

Nature of the Kalman Filter very well fits to recursive bayes filter concept. As it is stated for recursive bayes filter, Kalman Filter is recursive two state procedure those known as Time-Update and Measurement Update.

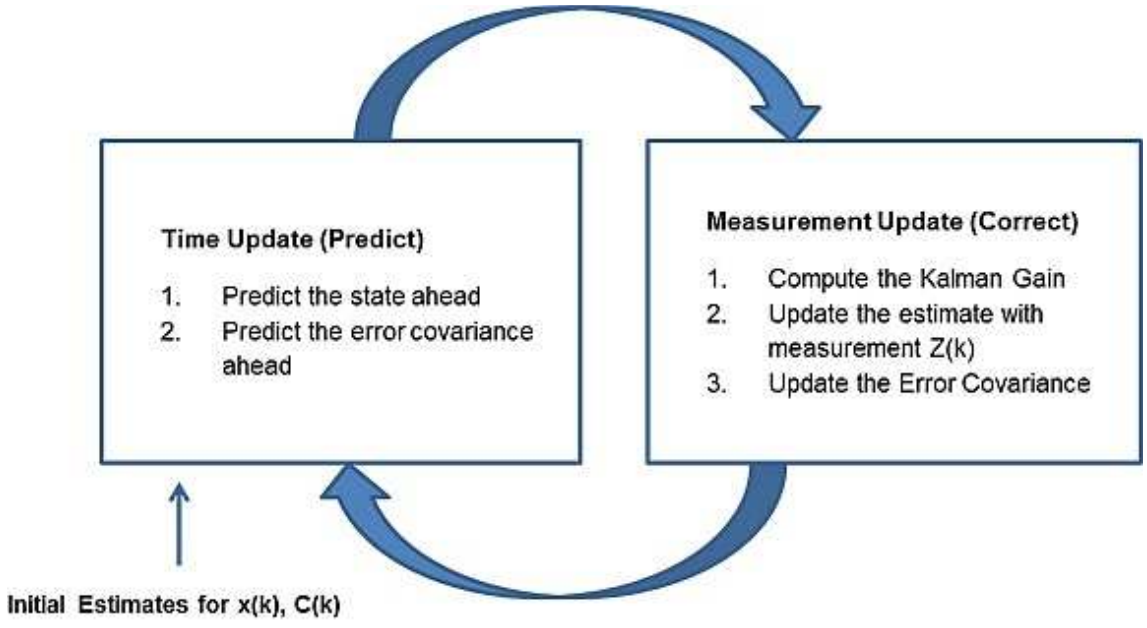

Figure 3.2 Kalman Filter Time & State Update Cycles [28]

3.4.1.1 Kalman Filter

Let us start with a system model definition. Then state transition and observation models will be as following:

$$x_k = Ax_{k-1} + Bu_k + q_{k-1} \quad (3.25)$$

$$z_k = Hx_k + r_k \quad (3.26)$$

Where q assumed as zero-mean Gaussian distributed process noise of system model. Also r_k assumed to be zero-mean Gaussian distributed measurement noise. Here, it is important to understand this "Gaussian" assumption. By declaring Gaussian assumption, for a random variable set, if we could have collect all the data in this set, it would have zero mean and some standard deviation value. This assumption makes linear connections between variables possible in algorithm. However most of the time real data obtained from nature or manmade instruments are not linear. But this will be skipped to be back return again.

So far system and observation models are defined. Than Prediction step of Kalman Filter can be written as follows:

$$\hat{x}_k = Ax_{k-1} + Bu_k \quad (3.27)$$

$$\hat{P}_k = AP_{k-1}A^T + Q_k \quad (3.28)$$

At eq.3.27 next state is predicted through state transition matrix and the current control inputs. It should be considered for a system like noisy sensor filtering there would be no control inputs and only would be state exists. Also P that is error covariance of the state x is predicted through additive process noise covariance$Q$.

Considering the time update (or prediction) step, state (or measurement) update will be defined with the following equations:

$$S_k = H_k\hat{P}_kH_k^T + R_k \quad (3.29)$$

$$K_k = \hat{P}_kH_k^TS_k^{-1} \quad (3.30)$$

$$x_k = \hat{x}_k + K_k(z_k - H_k\hat{x}_k) \quad (3.31)$$

$$P_k = [I - K_kH_k]\hat{P}_k \quad (3.32)$$

State update produce begin with the eq. 3.29 which is predicted error correction. Here R is the measurement noise covariance. Secondly eq. 3.30 gives the main factor of state update, Kalman gain. Afterwards eq. 3.31 updates the predicted step with Kalman gain

multiplied with state error to obtain the current step. Finally eq. 3.32 updates state error covariance to use for the next steps.

The main advantage of Kalman Filter is one or more states can be updated by multiple measurement observers. This gives ability to obtain best state estimate from different observation sources with different error margins, stabilities, etc.

3.4.1.2 Extended Kalman Filter

EKF is a special form of Kalman Filter. It is designed to be overcome non-linearity on Kalman Filter that already discussed on previous section. Although their non-linear nature, for many cases linear assumptions are satisfies the real application with Kalman Filter. But sometimes non-linearities cannot be eliminated. Non-linear states like circular trajectories cannot be well approximated with linear Kalman Filter. EKF overcomes this situations by linearize the non-linear state transition and measurement transition function.

Basically procedure is the same with linear Kalman Filter, but state-transition and observation models will take part as functions in following:

$$\hat{x}_k = f(x_{k-1}, u_k) + q_k \quad (3.33)$$

$$z_k = h(x_k) + r_k \quad (3.34)$$

Here f and h can be any non-linear function.

When approximating a non-linear curve to a linear function; taking the first derivative of non-linear function will give the linearized function at specified point. However mostly non-linearity in a system or observation model is not known. This linear approximation leads the posterior to become Gaussian. Once linearization is done then whole process is same as linear Kalman Filter recursive process. EKF filter uses first order taylor expansion to linearization process.

3.4.2 Particle Filter Based SLAM

It is already reviewed that for state observations and estimations Kalman Filter Based Algorithms are works. But Kalman Filter in its nature a Linear Estimator, it is not that successful on where non-linearity is descent. Even though Extended Kalman Filter which linearize non-linear states, it still based on Gaussian assumptions.

To overcome this situation, a fully non-linear filter by its nature is necessary. That is the time where particle filter come to scene. Particle Filters are computes posteriors of Hidden Markov States from noisy observations. Particle Filters relies on Monte-Carlo sampling technique which is a recursive technique too. This recursion process makes Monte-Carlo method suitable for Bayes filter implementation.

Probability distribution of Monte-Carlo based particle filter is as follows:

$$p(x_{o:t} \mid z_{1:t}) \tag{3.35}$$

Equation 3.35 represents the probability distribution by a finite set of samples which names in our case as particles. Through the definition particle filter can be written as follows [29]:

$$p(x_{o:t} \mid z_{1:t}) = \sum_{i=1}^{N} w_t^i \, \delta(x_{0:t} - x_{0:t}^i) \tag{3.36}$$

Particle filters have many advantages over comparing the other Bayes Filter implementations. Particle filters can approximate a wide range of distributions. Also comparing especially to Kalman Filter, implementation is much easier.

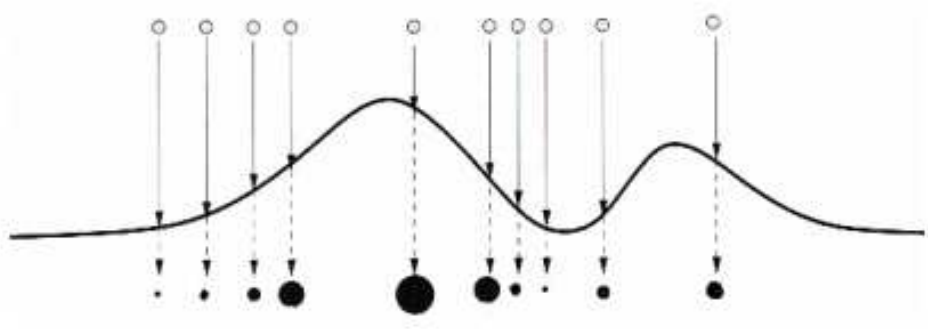

Figure 3.3 Particle Filter Non-Linear Distribution of a Particle Set (Circle diameters correspondent to particle weights)[29]

In figure 3.4 basic particle filter algorithm can be seen. Basically it takes most current state, observations and inputs than computes the sample distributions and updates weights according to probabilities and estimates the state.

1: **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$):
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Figure 3.4 Basic Particle Filter Algorithm [30]

3.4.3 Grid Mapping with Rao-Blackwellized Particle Filters (Gmapping)

Grid mapping is a 2D technique to build maps by occupying landmark areas on grid. It can be generated with particle filters. However when building a map by particle filter there are some challenges [31]. As an example, finding the optimal number of particles in a particle set to build an accurate map is challenging. Rao-Blackwellized approaches

defines complexity in number of particles to build a map. By simply increasing the particle numbers can increase accuracy but decrease computation performance. On the other side when one try to reduce the particle number to find optimal numbers; re-sampling state can eliminate higher probability particles.

In Gmapping algorithm [31], there are two important aims:

- Considering sensor accuracy when creating proposal distribution
- Creating adaptive re-sampling technique that reducing particle number while maintain a lean map

Particle distribution has main importance in particle filter. If there would have chance to distribute samples directly on target, then all particle would become equal and no longer necessary. However in general particles need to be randomly distributed to approximate the target. But some observation data can be used to approximate particles to target more closer.

For the proposal distribution, key idea of Gmapping algorithm to consider both observation likelihoods and motion model which odometry in this case. Algorithm locally approximates the posterior around the maximum of the observation likelihood which supplied by odometry data. In this case odometry data derived from laser scan registration. Once the posterior is calculated, a Gaussian approximation for the next N samples. This method enables the efficiently obtain the next set of particles.

Another key idea of Gmapping algorithm is focus on re-sampling state. Since not only approximating the particles with higher probabilities is enough; also re-sampling these particles with higher probabilities is important. However on basic re-sampling procedures higher probability particles has risk to be eliminated. Gmapping algorithm proposes a criterion to decide which particle to eliminate and which one to keep.

Gmapping algorithm uses a method efficiency variable N_{eff} to represent how well fits the current particles represents the posterior. N_{eff} is calculated as follows:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w^i)^2} \tag{3.37}$$

where w^i refers to normalized weight of particle i.

If N_{eff} drops under the $\frac{N}{2}$ where N is the particle number, algorithm re-sample the state. This reduces the eliminating the correct particles, since number re-sample states is reduced. This method brings to re-sample state when it is needed.

Both these better proposal distribution and adaptive re-sampling technique built more robust map creation method.

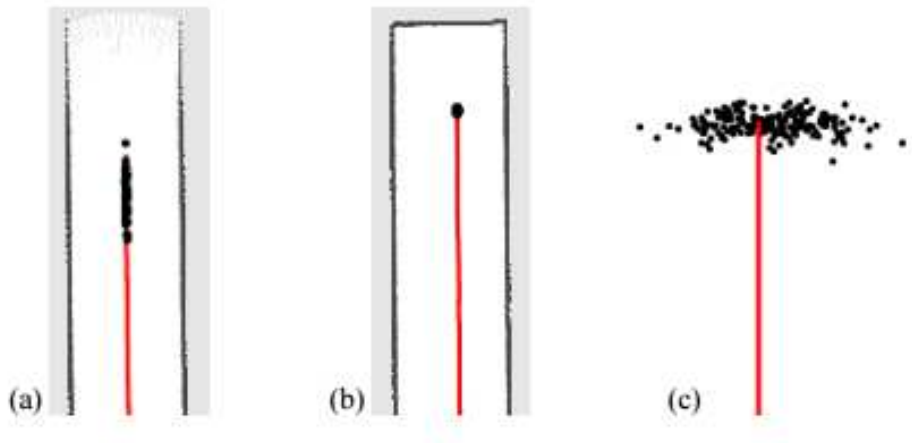

Figure 3.5 a) Proposal Distribution on open corridor b) proposal distribution on dead-end corrridor c) proposal distribution only assuming odometry model. [31]

CHAPTER 4

ROBOT OPERATING SYSTEM (ROS)

ROS (Robot Operating System) is a software platform [32] especially designed for robotics applications. The platform designed as a operating system and provides the services which one would expect from an operating system like drivers, libraries, package management utilities, message passing tools.

ROS had been born because of basic need of a stabile middleware in robotics research. ROS gives an opportunity to researches to skip platform creation and focus on implementation of real study case.

One of the main goals of the ROS platform is supporting the code reuse. Through this, once a code written for a specific purpose, it can be used and shared by others.

4.1 Project Goals

ROS designed thin as possible. It is not wrapping the code's main function through which can be used in another platform. This enables an easy integration with other robotics frameworks.

ROS supports language independency. The concept can be implemented in any modern programming language and had already been implemented on Python, C++ and Lisp. Also preferred development model is ROS-agnostic libraries with clean functional interfaces.

Scalability is important for ROS. It is suitable for large runtime systems and for large development processes.

4.2 Base Platforms

ROS is a conceptual operating system. Thus it must work on a real operating system. For now it only runs on UNIX based systems. Main support of ROS for Ubuntu and Mac OS X. But with contribution of community it has ported to other Linux distributions like Fedora, Gentoo, Arch Linux, Raspbian and many more. Porting to Windows has not realized yet.

4.3 ROS Distributions

ROS is distributed under version control system. From the beginning, following distributions have been made:

- ROS Box Turtle (March 2, 2010)
- ROS C Turtle (August 2, 2010)
- ROS Diamondback (March 2, 2011)
- ROS Electric Emys (August 30, 2011)
- ROS Fuerte Turtle (April 23, 2012)
- ROS Groovy Galapagos (December 31, 2012)
- ROS Hydro Medusa (September 4th, 2013)
- ROS Indigo Igloo (July 22nd, 2014)

4.4 ROS Concepts

ROS has different levels of concept. Most basics are “The File system level”, “The Computation Graph Level” and naming conventions. Following sections will describe these levels in detail.

4.4.1 ROS File System Level

All the ROS concept and applications developed on it, needs resources to manipulate, move and store the data. Inside ROS there is a unique file system exists for this need. This file system concept gives proper data management and application packaging ability.

4.4.1.1 Packages

An application developed in ROS must be organized as package. In other words to create an application, one need to organize its components as a package. After that, building and releasing the software through ROS is possible. A package may contain nodes (the main runnable unit in ROS which will be described in later sections), ROS dependent-independent libraries, configurations, datasets and other things related with application and needs to be organized.

4.4.1.2 Metapackages

Metapackage is basically a virtual package. It does not contain application or any other files. It only serves to reference one or more packages.

4.4.1.3 Package Manifests

Manifest for packages used to describe different properties of packages like its name, version, license information, description and dependencies. Also package build system uses these information provided by package manifests.

4.4.1.4 Message Types

Message type, in its basic form, a file that has a extension ".msg" and a predefined writing notation for included data structures. A message is used for communication with "topic" which is a data source-sink will be described in later sections

4.4.1.5 Service Types

Service type is same as message type as its file format. Excluding that it has a extension ".srv" and its data structure notation includes two parts separated to request and response. It is used at peer to peer service communication through the system

4.4.2 ROS Computation Graph Level

One of the most exciting aspect of ROS, it is a distributed system and not only controlled by one master data controller which extends the system through large networks and make the applications scalable. Communication through all around the network is peer to peer. This peer-to-peer network brings out the "Computation Graph" which is a network chain to process the data together.

Basic computation graph concept consists of the process like Master, nodes, parameter servers, messages, services, topics and data bags.

4.4.2.1 Nodes

Node is the most basic unit that makes the computation in ROS. Every node in a ROS runs a process. Considering an example robot arm application, one node may control the transformations between axes; another one controls the movement commands and the last node controls graphical view of the system. Node units plays important role in ROS's distributed, scalable structure

4.4.2.2 Master

Comparing the master controlled system, "Master" in ROS does not do any computation. Instead it provides name registration and lookup for the rest of the Computation Graph. It controls all the node communication, message exchange and service invocations. Looking from another angle, it is similar to internets famous DNS system

4.4.2.3 Parameter Server

On the ROS network key-value related data sets are stored at parameter server

4.4.2.4 Messages

Basic communication supplied with messages between nodes. A message is compounds with custom fields and corresponding data types to these fields. Basic primitive types (int, string etc.) and complex data types may used in messages.

4.4.2.5 Topics

Topic is a data source-sink system in ROS network. One can subscribe or publish data to topic and exchange predefined message over the network. Multiple nodes can publish or subscribe concurrently (Of course, this is not real time concurrency but it is used to describe the ability) to same topic. For example let us consider a topic names IMU which contains angular positions at x, y, z axes. A node which collects data from hardware can publish the received data to topic. After that many other nodes can

subscribe and receive the IMU data from that topic. It is also same on the reverse direction. Different nodes can feed the IMU topic publishing data. A one node could subscribe to that topic. Another approach to topics, assuming them as a data bus. Anyone can send or receive data from bus.

4.4.2.6 Services

Service is one-to-one communication implementation in ROS. Topics are good for many-to-many communications but there would be some cases where one-to-one communication is more appropriate in distributed systems. Services work with request-reply mechanism. Data type between these request and reply are stored in service type. To make a service request, one node need the provide a service with giving a name to it. Then a client node can fill the request part of the service type and makes the call; when service provider node takes the request, it makes the necessary operations and returns the response with filling the reply part of service type. Services are generally useful for remote commands like drive a motor, move the arm and so on.

4.4.2.7 Bags

Bags are the file format and the method of saving messages in ROS. Any important data shared between nodes as a message may stored in bags. Bags are useful for storing messages in real actions and to use at simulations and reviews after that. As an example, an application using a laser range finder can collect the data in real environment and store it in a bag. Then it can be used in algorithm development phases again.

4.4.3 Names

In order to proper work of ROS computation graph, every resource in graph needs to have a proper name. Though “Graph Resource Names” provides this hierarchical naming structure. Every node, parameter, topic or service needs to name under this hierarchy. Actually this naming conventions similar to namespace conventions at object oriented programming languages. As example, global namespace can be defined as:

- /

any resource name under this global namespace could be the following example:

- /exampleparameter

We can add other resource names under this parameter. In this case:

- /exampleparameter/anotherparameter

can be defined. In this example “anotherparamater” is defined under “exampleparameter” namespace.

One important thing to remember, this is a naming convention of resources. “anotherparameter” can be a node name or can be a parameter name holds a value.

With namespace logic, resources create their own resources under their namespaces. Communication between different namespaces is possible but generally above both namespaces and according to appropriate semantic. Otherwise encapsulation advantage of namespace logic would be pointless.

Resources are not aware of their namespace. So they can be written as if they all would work in same-level. In case of their usage under another graph, one can only need to put it under appropriate subgraph. For example imagine an application named LaserTools and have a node named Scanner. If another one would like to use it in its own application, then simply putting it to subgraph named LaserTools will be enough. Then if top-level application would write a node name Scanner, it would not conflict with other Scanner node. Because naming will be like this:

- /Scanner
- /LaserTools/Scanner

One can think of why this strict naming is needed. But ROS is designed especially for code reuse. Returning to example a researcher can use the LaserTools package in its own application without affecting own application and save time with code reuse.

4.4.3.1 Naming Convention

Names need to be valid according to these rules:

- First character must be alpha character “[a-z|A-Z]”, tilde “~” or forward slash “/”
- Subsequent characters can be alphanumeric “[0-9|a-z|A-Z]”, underscores “_”, or forward slashes “/”

4.4.3.2 Name Resolving

Graph Resource Names split to four types as base, relative, global and private. These types have the following syntax:

- base
- relative/name
- /global/name
- ~private/name

In default configuration, any resource related the node, will resolve the node's namespace. If there is no namespace is defined then name will be base name. Base names have the same namespace resolution with relative names. Generally base names are used to initialize node name.

Any name starts with forward slash "/" will resolve under global namespace. When using global namespace one should be careful. Since it will limit the code portability. If the written nodes become a component then it should have relative names for code reuse.

Private names are starts with tilde "~". Any parameter called with "~" will resolve under node. As example parameter "~foo/bar" will under our Scanner node will resolve to:

- /LaserTools/Scanner/foo/bar

4.4.4 Package Resources Names

Package resource names are actually shortage of files and data types on ROS File system Level. For example "std_msgs/Int" name is a shortage of Int data type under std_msgs package. Again, to remark, "Int" data type is virtual data type which is created on ROS File system Concept.

Main files which can be accessed thorough Package Resource Names;

- Message Types
- Service Types
- Node Types

Package Resource Names are same as with normal file paths except with much shorter way. ROS content search system and its ability about content assumption make

possible to locate exact file from Package Resource Name. For example any message type is located under “msg” folder in package folder. So if we search for “std_msgs/Int”, then ROS will look to “/path_to_package/std_msgs/msg/Int.msg”.

4.4.4.1 Valid Naming

Valid naming of Package Resource Names have very strict rules. For this reason a Package Resource Name must obey the following rules:

- First character is an alpha character ([a-z|A-Z])
- Subsequent characters can be alphanumeric ([0-9|a-z|A-Z]), underscores (_) or a forward slash (/)
- There is at most one forward slash ('/')

4.5 ROS Setup

4.5.1 Environment Setup

Running ROS is requires complete environment setup. All the necessary packages, workspaces and core files need to be properly setup. This setup is basically provided by operating system’s shell commands which are “sh” files under Linux distributions. With proper shell files any package can be easily included or excluded to environment.
Environment setup files can be generated manually as well as automatic package build processes or by packages which installed with package managers.
Main setup file of ROS which introduces core path and necessary core information to operating system is under the path:

- '/opt/ros/<distro>/'

and to source this setup file to operating system which is Linux(Ubuntu) in our case is supplied by:

- # source /opt/ros/<distro>/setup.bash

command. Afterwards operating system will be ready to run ROS and its commands. It is important to notice that this concept gives ability to switch between different ROS versions. Only thing to do is writing another installed version to <distro> field on the path. One thing important is this setup should be added to “.bashrc” file of Linux so that ROS becomes part of boot process of Linux.

4.5.2 Creating Workspace

In the first versions of ROS workspace management is provided by “rosws” and “rosbuild” tools. But afterwards “catkin workspace” concept is initialized. Catkin is also a package build and management system, but more advanced which allows multiple or independent builds and provided workspace concept. “rosws” and “rosbuild” tools still exists in latest versions but catkin is mainstream workspace management tool. Through we need to go more deep to catkin workspace structure.

4.5.2.1 Catkin Workspace

Catkin, in its basic form, a package management system where we can modify, build and install catkin packages. A general layout of catkin workspace can be seen on figure 4.1. Each of the main folders serves for different purposes.

4.5.2.2 Source Space

Source space contains the source files of catkin packages. All the ROS packages which we want to build should be placed under this folder. Each unique folder under source space refers to a package.

4.5.2.3 Build Space

Build space is used by catkin to invoke CMake compile tool to build packages in source space. All the necessary build process information is putting under this folder during the process.

4.5.2.4 Development (Devel) Space

Development space is the first place where build targets are placed after compile process. This is very useful for testing and development before installation.

4.5.2.5 Install Space

Install space is a folder where the build targets are installed. This directory does not have to be under workspace. It can be under anywhere. Location of this directory can be set by “CMAKE_INSTALL_PREFIX” variable inside CMakeLists.txt configuration file.

```
workspace_folder/         -- WORKSPACE
  src/                    -- SOURCE SPACE
    CMakeLists.txt        -- The 'toplevel' CMake file
    package_1/
      CMakeLists.txt
      package.xml
      ...
    package_n/
      CMakeLists.txt
      package.xml
      ...
  build/                  -- BUILD SPACE
    CATKIN_IGNORE         -- Keeps catkin from walking this directory
  devel/                  -- DEVELOPMENT SPACE (set by CATKIN_DEVEL_PREFIX)
    bin/
    etc/
    include/
    lib/
    share/
    .catkin
    env.bash
    setup.bash
    setup.sh
    ...
  install/                -- INSTALL SPACE (set by CMAKE_INSTALL_PREFIX)
    bin/
    etc/
    include/
    lib/
    share/
    .catkin
    env.bash
    setup.bash
    setup.sh
```

Figure 4.1 Catkin Workspace General Overview

4.5.3 Creating Workspace Using Catkin Tools

To create a workspace using catkin following Linux shell commands need to be run:

- $ mkdir -p ~/catkin_ws/src
- $ cd ~/catkin_ws/src
- $ catkin_init_workspace

After running “catkin_init_workspace” command, a workspace is created under “catkin_ws” path and new packages can be added to “src” folder. Yet we created an empty workspace so there are no packages under “src” folder.

Now we have a catkin workspace and have no packages inside. However we can still build the workspace. To build the workspace following commands need to be run in order:

- $ cd ~/catkin_ws/
- $ catkin_make

This command compiles workspaces and creates two other folders “build” and “devel”. As it is explained in section 4.2.1 build folder is where build process occurred and devel is the folder for build targets.

After build process, inside “devel” folder several setup files can be found. Using this setup files we can switch our environment to this workspace. This is possible using the command:

- source devel/setup.bash

By checking “ROS_PACKAGE_PATH” variable in Linux shell we can verify if our workspace is in environment.

4.6 Creating A ROS Package

Package in ROS, by definition, is a simple folder with simple XML file. There is already a tool exists in ROS to create package and can be run as:

- $ catkin_create_pkg

This command creates the folder and xml structure. But to understand the structure we can create the package manually.

4.6.1 Folder Structure

Before any action it is important to remember environment setup should have been done. Let us define a package names as “LaserDriver”. First of all we need to create the folder named as “LaserDriver” under our catkin workspace with following commands:

- mkdir -p src/ LaserDriver
- cd src/ LaserDriver

We created the folder “LaserDriver” and went into that directory. Now inside the folder we need to create a XML file named as “package.xml” and txt file names as “CMakeLists.txt”. Now our package folder structure is ready.

4.6.2 Package Manifest

Package manifest is a simple XML file which includes the information like package name, version, author, date, internal and external dependencies. Example XML file content can be seen on figure X1.

With a proper package manifest, if the environment properly setup, ROS will find newly created package. If we execute the command “rospack”, it will find the example “LaserDriver” Package. It can be run as follows:

- $ rospack find LaserDriver

It should be noted that assuming the XML structure on figure 4.2, LaserDriver package depends on roscpp and std_msgs packages. These should be installed on system to build the package (In this case “roscpp” and “std_msgs” packages are default packages of ROS and comes with installation). Following sections will describe how and when to use these dependency tags in XML file.

```
<package>
  <name>LaserDriver</name>
  <version>1.0.0</version>
  <description>
  Example package for ROS.
  </description>
  <maintainer>Hayrettin Ertürk</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>

  <run_depend>roscpp</run_depend>
  <run_depend>std_msgs</run_depend>
</package>
```

Figure 4.2 Package.xml file structure

4.6.2.1 Build Tool Dependencies

This is the definition of system tool to build the package itself. In our case there is only one build tool which is catkin. This is generally useful for cross-compile purposes. In the XML file it’s tag is “<buildtool_depend>”

4.6.2.2 Build Dependencies

Build dependency defines which packages are needed to build our package. If a packages is defined as a build dependency any resource from that package like header files, libraries etc will be needed at build time. To define a package as a dependency one need to use “<build_depend>” tag.

4.6.2.3 Run Dependencies

Run dependencies are necessary to run the package. Generally run dependencies are shared libraries. To declare a run dependency "<run_depend>" tag must be used.

4.6.2.4 Test Dependencies

Test dependencies are used for unit tests. It should be noted that a dependency which already declared as a build or run dependency must not be duplicated as a test dependency. Only additional dependencies should be added. To add a test dependency one can use "<test_depend>" tag in package manifest XML.

4.6.3 Package Build Configuration

To build a properly prepared package we need a "CMakeLists.txt" file. This file used by catkin and provides CMake build configuration. Example CMakeLists.txt file for our package LaserDriver can be seen on figure X3.

```
cmake_minimum_required(VERSION 2.8.3)
project(LaserDriver)
find_package(catkin REQUIRED roscpp std_msgs)
catkin_package()
```

Figure 4.3 CMakeLists.txt file with basic setup for compile process

With these configurations we have configured a catkin ROS package. It is important to note that "package.xml" is necessary for ROS to find the package and "know" it. Where "CMakeLists.txt" file is necessary for binary compile process. However generally system dependencies declared in package manifest should take part in CmakeLists.txt file, inside "find_package()" function. This information is very useful and should be kept in mind to understand ROS basics very well and to setup and maintain packages easily.

CHAPTER 5

SLAM IMPLEMENTATION

In basis of this thesis, an offline SLAM to acquire a consistent map is implemented through Gmapping Algorithm over ROS platform. However implementation needs carefully configuration of ROS Packages and hardware platforms.

5.1 Hardware Setup

5.1.1 Laser Scanner

One of the most important hardware when implementing the SLAM is using the appropriate circular sensor to learn about landmarks. From the historical perspective of SLAM implementations; ultrasonic sensors widely used on early applications.. However ultrasonic sensors have limited sensing range on angular plane. Also sensing accuracy varies over angular sensing plane. Due to unidirectional sensing, to cover all around the vehicle body, more than one sensor needs to be used. Besides nature of the ultrasonic sensing is based on sending sound waves through air medium and catch back. So it is

Figure 5.1 Hokuyo Laser Scanner [40]

not always possible to sense the distance with enough accuracy to use at SLAM applications.

However in recent years, laser scanners have been initialized. Laser scanning technique is similar to ultrasonic sensing. But with one difference that laser scanners sending laser light and sense back the reflection from a landmark. When a laser scanner is rotated around a circle center, then it is possible to measure distances around the sensor with high accuracy.

For this reasons Hokuyo branded URG-04LX-UG01 laser scanner is used for SLAM purposes. It has a 240 degree of angular measurement range and 20mm to 5600mm linear measurement range. This scanner completes one scan in 100msec which is fast enough to collect distance samples to build a map.

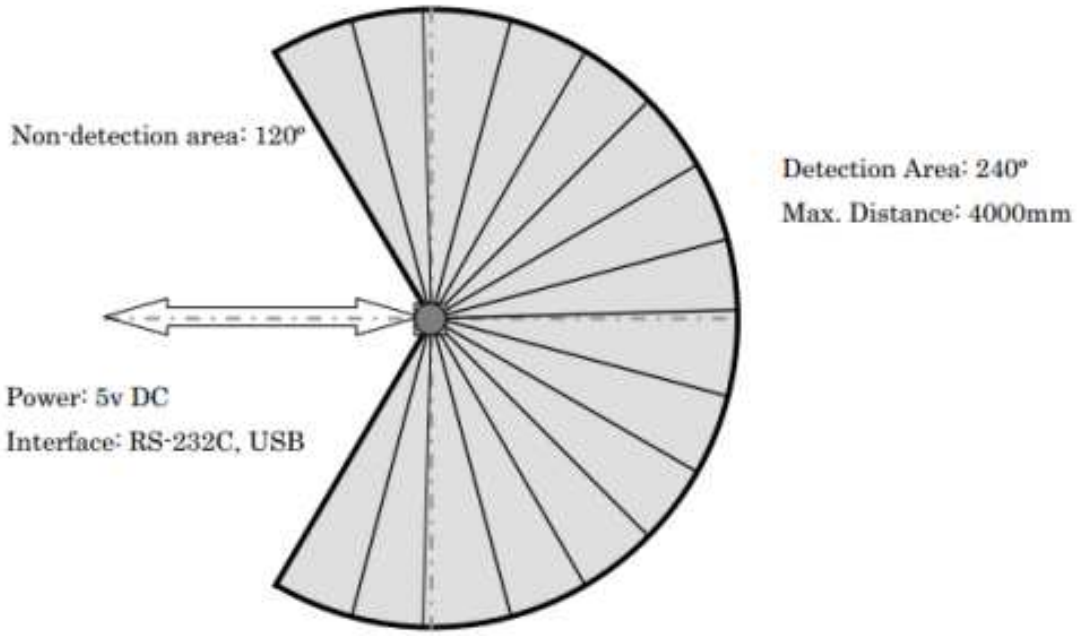

Figure 5.2 Laser Scanner's scan range [33]

5.1.2 Raspberry Pi

Raspberry Pi is a mini pc with embedded functionalities. When running a distribution of Linux on Raspberry Pi, it is possible to open or close I/O ports or communicate with sensors over I2C or SPI protocol. This gives flexibility to when using Linux top level features one can control embedded units. Also Raspberry Pi has HDMI output which can be connected to any monitor that supports HDMI. One can connect the Raspberry Pi

with HDMI and keyboard/mouse interface or through SSH over Ethernet or Wifi using a dongle.

Raspberry Pi is used as a computer to collect Laser Scanner Data. Since it has USB ports and operating system enabled, so that more suitable than embedded controller to transfer laser data. Therefore a special ROS distribution [34] is installed on official Raspberry Pi distribution which is Raspbian.

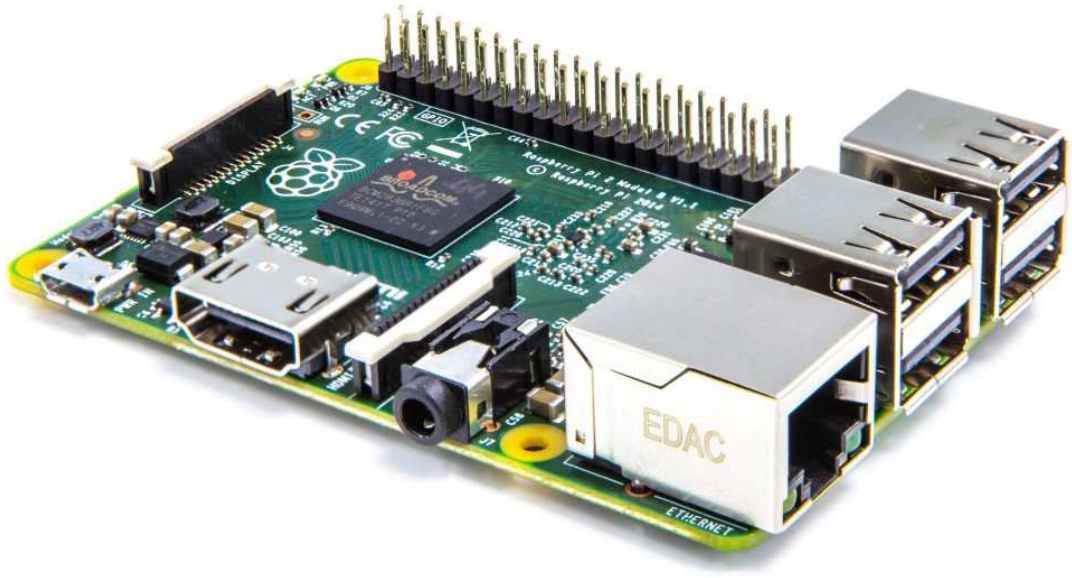

Figure 5.3 Raspberry Pi board upper view

5.1.3 Quadrotor Structure

As an UAV, a Quadrotor has been built for this study. Basically Quadrotor is a vehicle that composed of two rectangle material symmetrically coupled in center of both. In this study a rigid body composed with wooden material. As a requirement Quadrotor needs to be light and solid. Another option was to use carbon fiber coated body structure. But considering the costs using wood material was easy to use and budget friendly option.

Figure 5.4 The Quadrotor that built in this work and Futaba remote controller

5.1.4 Flight Controller Board

To control the Quadrotor, flight controller hardware is used as an APM 2.5 board which is developed for open source flight controller project “Arducopter”. APM 2.5 board is an embedded controller with a MCU, gyroscope sensor, accelerometer sensor, magnetometer, barometric pressure sensors.

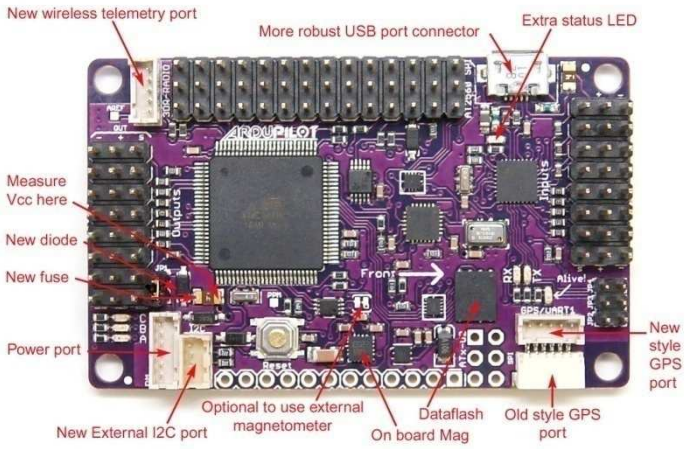

Figure 5.5 APM 2.5 Flight Controller Board [36]

Board also has an external i2c connection port to connected external sensors. A GPS connection socket and a telemetry connection to connect PC interface is available. APM 2.5 board supports up to 8 motor outputs which is enough ports to create Octorotor. In this work only 4 outputs is used since a basic Quadrotor is chosen as an UAV.

5.1.5 Telemetry & Remote Control

For wireless telemetry and tune the Ardupilot software, Xbee radio modules have been used. Xbee radios are produced by Digi Company and works at 2.4 GHz band with IEEE 802.15.4 protocol. Xbee radios are especially used on robotic applications very often due to their easy to use nature. They can communicate same as wired RS-232 serial port with a maximum baud rate of 115200 baud.

Figure 5.6 Xbee Telemetry RF Module

Also to send control commands to Quadrotor, a Futaba branded remote control have been used. Futaba remote controller also works at 2.4 Ghz and communicate with a special communication protocol named PPM. PPM is a simple protocol which sums servo PWM commands that explained in previous chapters. In servo PWM control duty cycle is between 1ms and 2ms. When PPM summing multiple servo PWM commands it basically put a constant duty cycle frame in 20ms period where one commands falling edge and where other one's rising edge meets. This summing enables to send all commands in single channel.

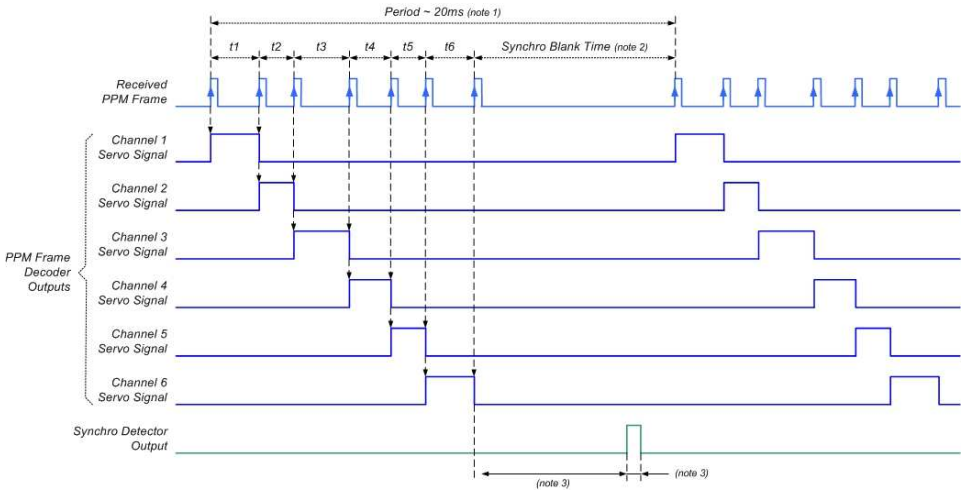

Figure 5.7 Graphical view of PPM signals sum into one channel [35]

5.1.6 Excitation

For the excitation, same configuration with test setup for extracting simplified transfer function model is used. As a motor Mystery motors are used. Also Mystery 20A limited Electronic Speed Controllers are used. These are speed controllers and work with servo PWM protocol too which also compatible with APM 2.5 controller board.

5.2 Software Setup

5.2.1 Arducopter Setup

Arducopter [36] is an open source project to form a general software platform for rotary wing UAV's. It has basic multirotor controller algorithms with PID controllers. Based on this core, it has a lot of feature like Gimbal controlling, altitude hold based on GPS or Barometric pressure sensor, return to station when signal is lost and more.

After loading Arducopter into the APM 2.5 board, it is possible to configure the parameters including PID over Xbee telemetry channel on PC using Mission Planner interface. It is especially built to configure and track almost all parameters available on running Arducopter software.

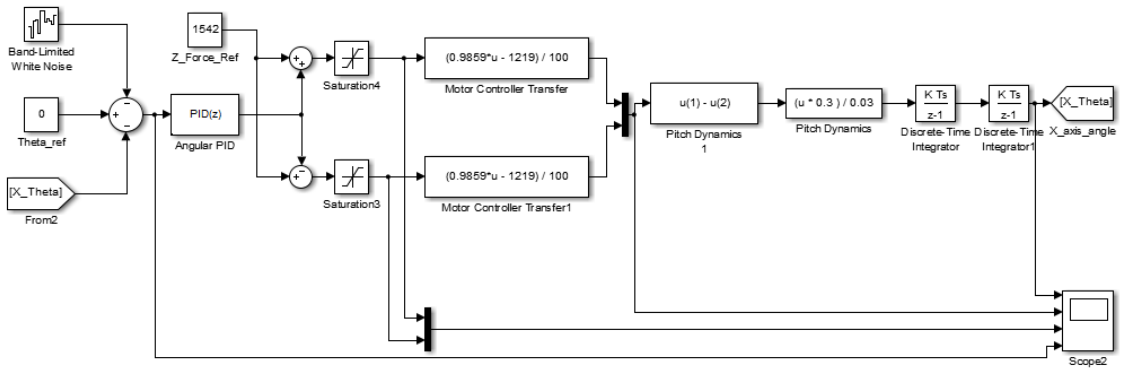

Figure 5.8 Matlab model created for single axes(pitch) to approximate rotational orientation control using simplified transfer function

For this work especially PID parameter's change availability was important. Considering Quadrotor's structure is custom built and custom excitation is used, parameters for flight controller should have been configured. Using the simplified transfer function model in Chapter 2, a Matlab model of Quadrotor simply in one axes has created. Afterwards using the model PID parameters are auto-tuned on Matlab. Using the tuned parameters, real PID values are approximated and applied to Quadrotor using Arducopter.

5.2.2 Preparing ROS Environment for SLAM

As already explained in details at previous chapter, before any attempt ROS environment should be correctly setup for any application. First off all a catkin workspace needs to be configured before any package installation. In this case a workspace with a name "catkin_ws" is created.

To implement SLAM using Gmapping algorithm, following packages are necessary to work on the same time:

- Static transform for laser scanner to vehicle base
- Hokuyo node package to collect laser scan
- Laser Scan Matcher package to estimate odometry data from laser scan
- Gmapping package to implement SLAM

5.2.2.1 Static Transform Publisher

Due to nature of ROS, any joint is considered as a transform on a vehicle at ROS. In this study Laser Scanner is mounted on top of Quadrotor, so there aren't any moving

joint which means there aren't any dynamic transform. But to setup the transform chain it is needed to setup a transform between laser scanner and Quadrotor's base. This is realized with static transform node under "tf" [37] package of ROS.

Static transform publisher is used with the following parameters:

- static_transform_publisher x y z yaw pitch roll frame_id child_frame_id period_ in_ms

Here x, y, z are translational transforms and yaw, pitch, roll transforms. Frame id and child frame id are frames to transform and period is the publish period in ms rate.

Running the following command will start a transform between "base_link" frame which is the Quadrotor and "laser" frame which is laser scanner in our case:

- rosrun tf static_transform_publisher 0 0 0 0 0 0 /base_link /laser 100

This command will define an exact transform between Quadrotor and laser scanner and publish it with 100ms of period.

Or it can be placed in launch file with following notation:

- <node pkg="tf" type="static_transform_publisher" name="Quadrotor_laser_transformer" args="0 0 0 0 0 0 /base_link /laser 100" />

5.2.2.2 Hokuyo Package

To get laser scan into ROS, a special node is necessary to collect scan data and publish to ROS topic. A specific driver [38] has been developed for this purpose by ROS community that named as "hokuyo node" package.

Before running hokuyo node package, especially under Linux, it should be checked that device has the necessary permissions to run. Otherwise running of the package will fail.

To run the hokuyo laser scanner into ROS following command is used:

- rosrun hokuyo_node hokuyo_node

Also it can be placed in launch file as following:

- <node name="hokuyo" pkg="hokuyo_node" type="hokuyo_node"> </node>

After run, hokuyo_node package is publishing the scan data to “/scan” topic.

5.2.2.3 Laser Scan Matcher Package

Gmapping algorithm requires to use of some odometry data to transform vehicle into base world frame. Since wheel based odometry is not possible on a UAV, odometry data should be derived from other sources. For this purpose Laser Scan Matcher [39] is developed by ROS community. Laser Scan Matcher package is estimates the odometry data from laser scan data. In case of odometry or IMU data existence, they can be used to improve the accuracy of the pose estimate. In our case package is used only with laser data since odometry data was necessary at all.

Running the Laser Scan Matcher package requires to run the following command:

- rosrun laser_scan_matcher laser_scan_matcher

Or it can be placed into launch file using the following notation:

- <node pkg="laser_scan_matcher" type="laser_scan_matcher_node" name="laser_scan_matcher" output="screen">
  <param name="fixed_frame" type="string" value="odom"/>
  </node>

Also parameters of the package can be put between node tags and configured. Through this configuration odometry data is available under /odom topic.

5.2.2.4 Gmapping Package

To run the Gmapping package following command should be run on command-line:

- rosrun gmapping slam_gmapping scan:=scan

This command will subscribe Gmapping to /scan topic. Or Gmapping package can be run on launch file using the notation:

- <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen"></node>

This runs the Gmapping package with default configurations which is enough to implement a generic slam.

5.3 SLAM with Gmapping Package

After software and hardware setup, a slam application has been made with manually controlled Quadrotor. As a mapping environment, Yıldız Technical University's Robotic Research Lab at Control Engineering Department has been used. Gmapping algorithm successfully generated maps and therefore located the vehicle inside simultaneously. It has been also showed that algorithm successfully eliminates moving targets and not integrates them to map. The data visualized with ROS visualization tool Rviz.

In figure 5.9, the vehicle at the beginning position and map only consists what it scans around for the first time. In this view, gray area is unexplored area, light gray area is unoccupied area and the black areas are occupied areas as stationary landmarks.

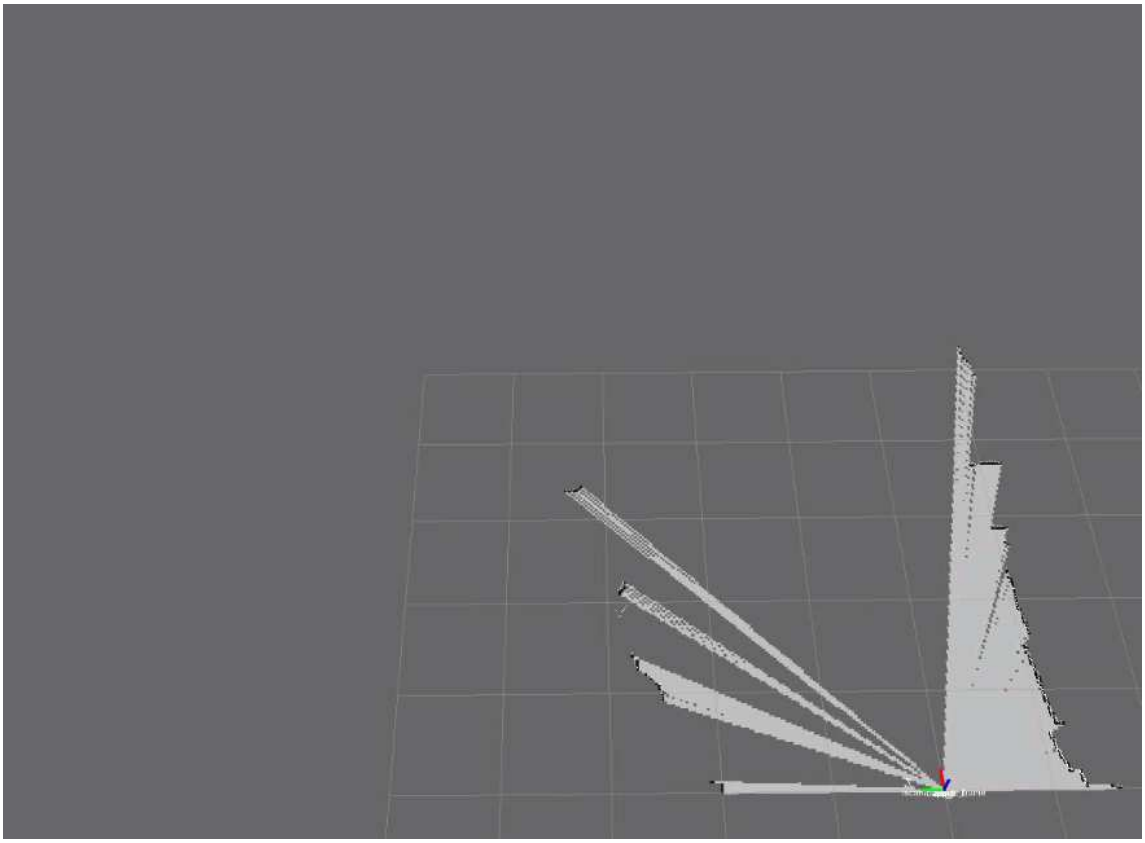

Figure 5.9 Map created at the beginning of exploration

In figure 5.10 after some steps ahead the vehicle explores new fields and map is updated according to recent data. It is clear that algorithm place landmarks for most certain readings, in other words it place the landmarks where belief therefore probability is high. It should be noted that in figure 5.10 range finder actually measure the distance at

left side of the room but probability is much low to place a landmark and it needs more scans to increase the probability that landmarks are actually there.

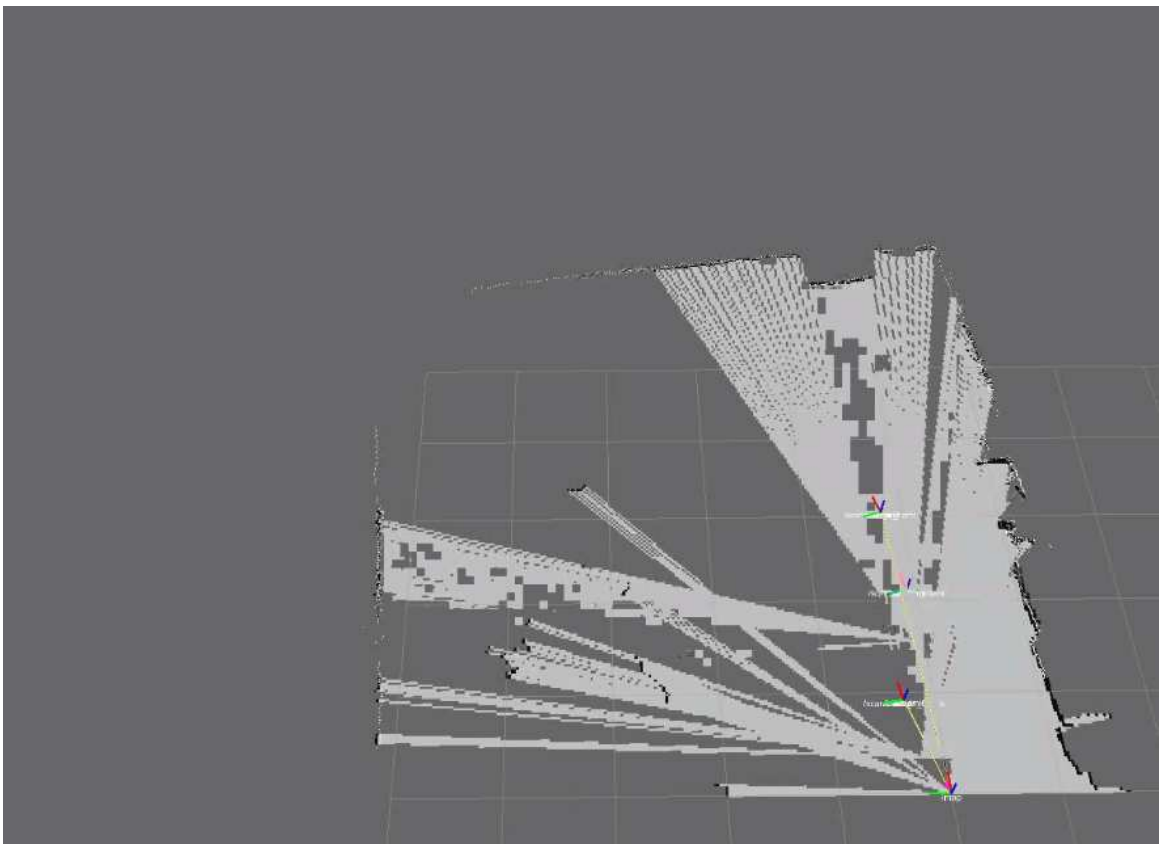

Figure 5.10 After few steps later, landmarks become visible on the map

In figure 5.11 and figure 5.12 as already mentioned, algorithm took more scan on the other side of the room and probabilities are increased. As a result landmarks are placed for the north half of the room. However to complete the room vehicle had to take more scans.

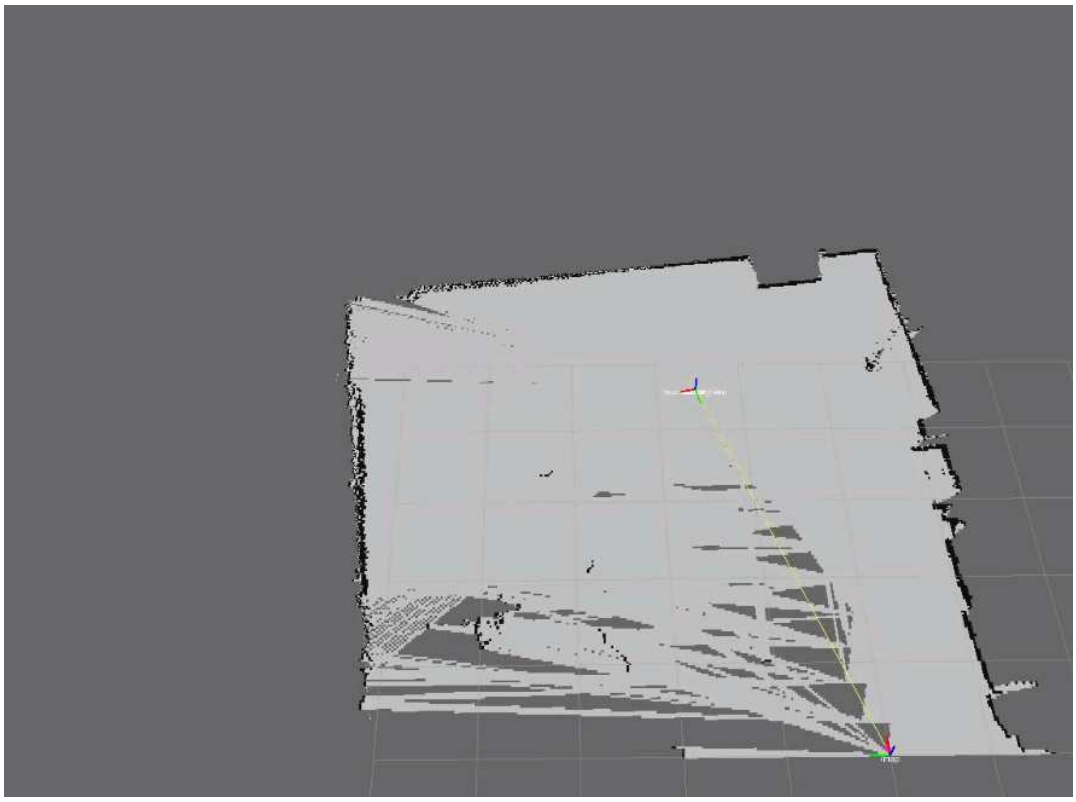

Figure 5.11 North of the room is already become clear

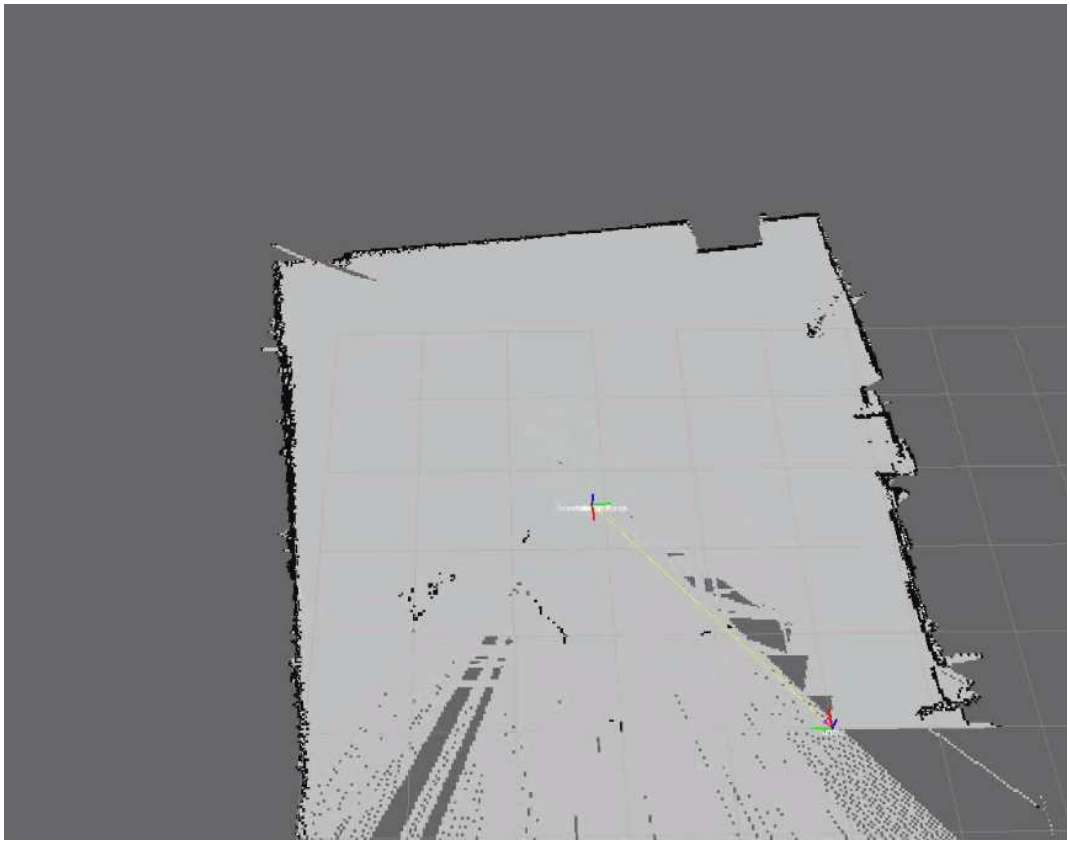

Figure 5.12 Half of the room scanned enough and map is created on the north side of the room

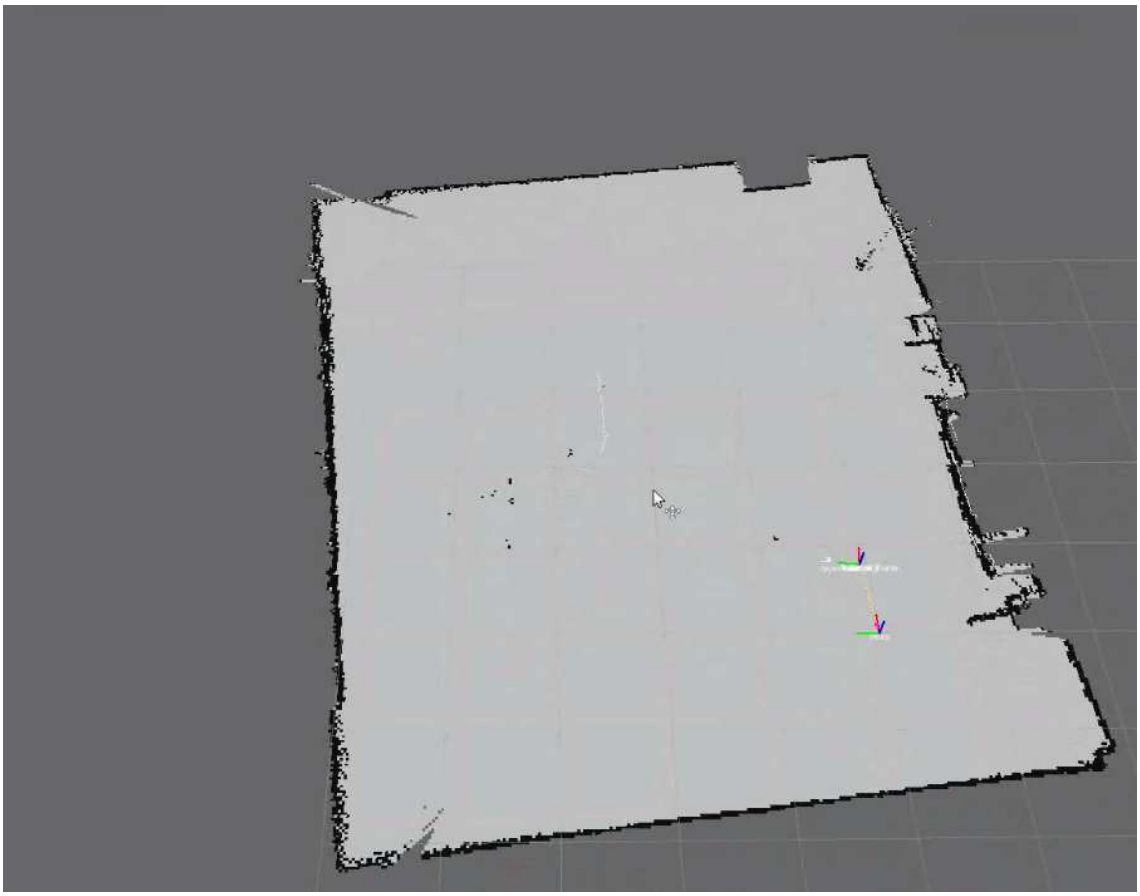

Figure 5.13 Finally scan is completed and map is created. Also vehicle pose is known in this map.

In Figure 5.13 and figure 5.14 it is clear that map algorithm successively created the map. However there are still some unexplored walls and corners but not significant comparing to whole map. Due to Quadrotor's takeoff process, some low landmarks like tables take part in the map. Since this is 2-D process mapping algorithm is not aware of height. But after takeoff it only mapped walls and corners.

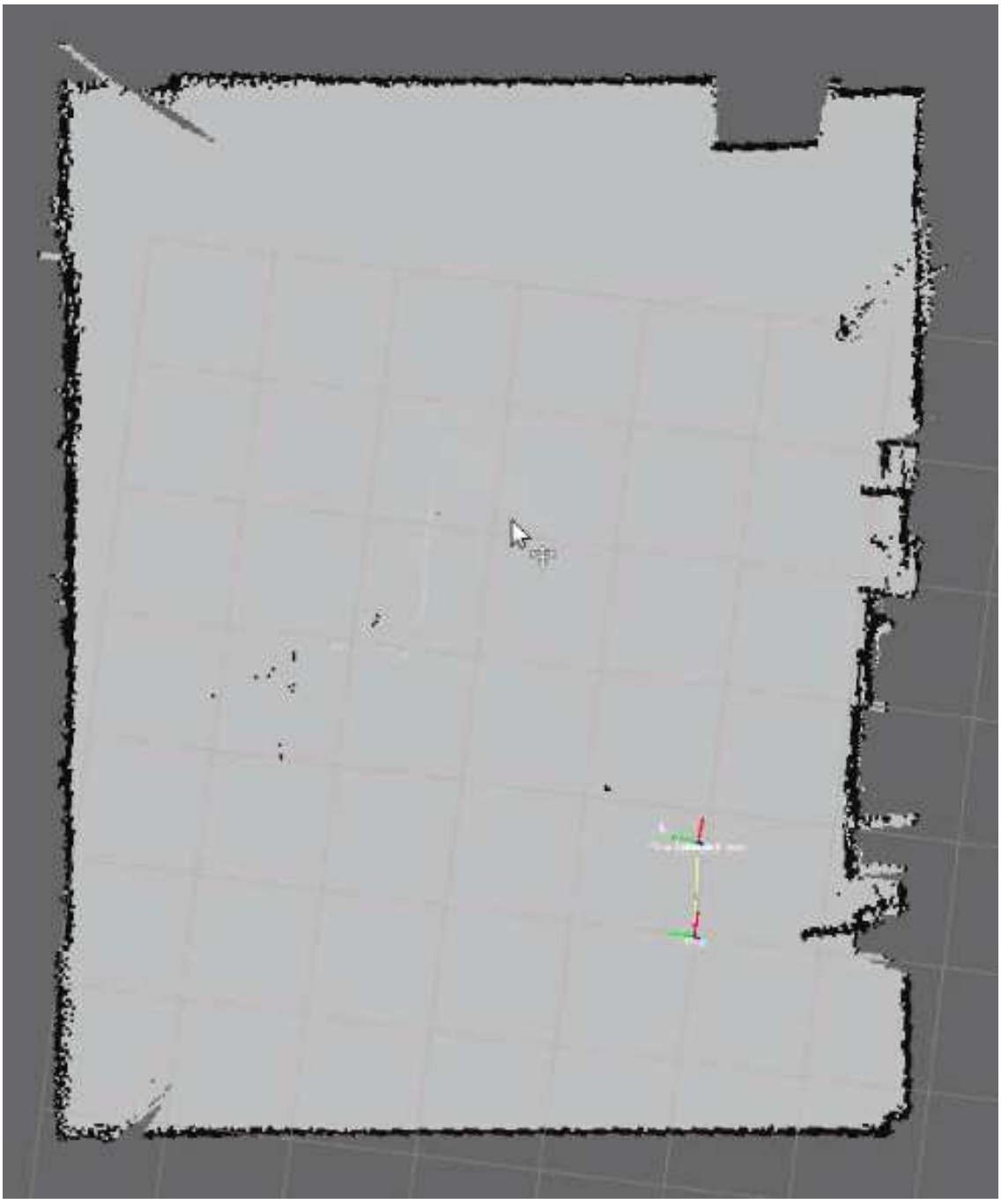

Figure 5.14 Top view of the completed map. The reference at the very bottom of the map is beginning position. Other reference upwards of the beginning is where vehicle is located at the end of mapping process.

CHAPTER 6

CONCLUSION

Unmanned Vehicle concept was upcoming trend in last decades and its importance getting higher every day. Not only for vehicles, also has Unmanned Concept played a big role in robotic applications. Environment recognizing and self-decide abilities for given targets, is necessary to use robotic applications autonomously. Improvements at micro electro-mechanical sensor technologies (MEMS) and embedded electronic devices is already make possible to apply the Unmanned Concept not only for military or space technology but daily life applications. This self-decide ability of vehicles and robots will make human life easier and much safer.

In this work, SLAM, the most recent algorithm framework for unmanned concept is reviewed and a SLAM application is applied using an Unmanned Airel Vehicle (UAV) that is a Quadrotor. Through manually controlling the Quadrotor environment recognized. Map creating and in the same time localizing the robot inside the map has realized with Gmapping SLAM implementation.

As an implementation platform the Robot Operating System (ROS) has been used. ROS is a platform that developed especially for robotics research applications. It has concepts like real operating system but it is a virtual operating system. Most supported base platform of ROS is Ubuntu distribution of Linux. There are many tools in ROS to use at implementation of any robotic algorithm like filters, messages, sensor drivers and so on. Also application in this thesis is built using an implemented package of Gmapping algoritm in ROS.

It is clear that implementation of SLAM framework is not easy that it requires a background of inter-disciplinary background. Topics like image processing, sensor

fusion, advanced software techniques, control algorithms and filters are essential to study on this field.

Through application results it has been showed that Gmapping successfully recognize around and creating maps also locating the vehicle inside. One of the important aspects of Gmapping algorithm that it is not only Online SLAM application. Using the created map for especially static places like buildings, further applications for only localization implementations saved map can be used.

Assuming last century, developed machines were manually controlled or automatically controlled. And application fields were limited to industry or military/space applications. However technological developments and mobile technology developments takes us one step closer to use machines all over the daily life. To make that possible, it is clear that next decades will be autonomy age.

REFERENCES

[1] Juan Du, Xinmin Wang, and Rong Xie Yusong Jiao, (2010). "H∞ State Feedback Control for UAV Maneuver Trajectory Tracking", International Conference on Intelligent Control and Information Processing, 2010, Dalian, 253-257.

[2] H. Zargarzadeh, and S. Jagannathan David Nodland, (2011). "Neural Network-based Optimal Control for Trajectory Tracking of a Helicopter UAV", 50th IEEE Conference on Decision and Control and European Control Conference *(CDC-ECC)*, 2011, Orlando, 3876-3881.

[3] F. Chaumette, and S. Boudet E. Malis, (1999). "Landing a VTOL Unmanned Aerial Vehicle on a Moving Platform Using Optical Flow", IEEE Trans. Robot. Autom., 1999, 15:2.

[4] Tao Dong, X.H. Liao, R. Zhang, and Z. Sun, (2005). "Path Tracking and Obstacles Avoidance of UAVs - Fuzzy Logic Approach", The 14th IEEE International Conference on Fuzzy Systems, 2005, Reno, 43-48.

[5] P.B. Quater, F. Grimaccia, S. Leva, and M. Mussetta, (2014). "Light Unmanned Aerial Vehicles (UAVs) for Cooperative Inspection of PV Plants", IEEE Journal of Photovoltaics, 2014, June, 4:1107-1113.

[6] J. Nefian, A.V. Gleason, X. Bouyssounousse, and T Fong, (2011). "Vehicle detection from aerial imagery", IEEE Conference on Robotics and Automation (ICRA), 2011, Shanghai, 2065-2070.

[7] Pablo J. Zarco-Tejada, Lola Suárez, Elias Fereres Jose A. J. Berni, (2009), "Thermal and Narrowband Multispectral Remote Sensing for Vegetation Monitoring From an Unmanned Aerial Vehicle", IEEE Transactions On Geoscience And Remote Sensing, 2009, March, 47:722-738.

[8] R. Loh, Yi Bian, and T. Roe, (2009), "UAVs in civil airspace: Safety requirements", IEEE Aerospace and Electronic Systems Magazine, 2009, January,

24:5-17.

[9] François Chaumette, and Sylvie Boude Ezio Malis, (1999). "2-1/2-D Visual Servoing", IEEE Transactions On Robotics and Automation, 1999, April, 15: 238-250.

[10] Tarek Hamel, Robert Mahony Nicolas Guenard, (2008). "A Practical Visual Servo Control for an Unmanned Aerial Vehicle," IEEE Transactions On Robotics, 2008, April, 23:331-340.

[11] Y. Wu, W. Liu, X. Chen & J. Zhang, (2010). "Novel Approach to Positionand Orientation Estimation inVision-Based UAV Navigation", IEEE Transactions On Aerospace And Electronic Systems, April, 2010, 46:687-700.

[12] Jinchun Zhou, (2014). "EKF Based Object Detect and Tracking for UAV By Using Visual-Attention-Model", IEEE Informatics and Computing (PIC) International Conference, Shanghai, 2014, May, 168-172.

[13] John V. Stafford, (2000). "Implementing Precision Agriculture in the 21st Century," Journal of Agricultural Engineering Research, 2000, July, 76:267-275.

[14] Joshua Vander Hook, David Mulla, Volkan Isler & Pratap Tokekar, (2013). "Sensor Planning for a Symbiotic UAV and UGV system for Precision Agriculture", 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), 2013, Tokyo, 5321-5326.

[15] Ashutosh S. Limaye, Charles A. Laymon & William L. Crosson, (2005). "Parameter Sensitivity of Soil Moisture Retrievals From Airborne L-Band Radiometer Measurements", SMEX02-IEEE Transactions On Geoscience And Remote Sensing, 2005, July, 43:1517-1528.

[16] F. Jonard, L. Weihermuller, K.Z. Jadoon, and M. Schwank, (2011). "Mapping Field-Scale Soil Moisture With L-Band Radiometer and Ground-Penetrating Radar Over Bare Soil," IEEE Transactions on Geoscience and Remote Sensing, 2011, August, 8:2863-2875.

[17] R. Acevo-Herrera, A. Aguasca, X. Bosch-Lluis and A. Camps, (2009). "On the use of compact L-band Dicke radiometer (ARIEL) and UAV for soil moisture and salinity map retrieval: 2008/2009 field experiments", Geoscience and Remote Sensing Symposium, IEEE International, 2009, Cape Town, 729-732

[18] D.W. Casbeer, R.W. Beard, T.W. McLain, Sai-Ming Li and Raman K. Mehra, (2005). "Forest fire monitoring with multiple small UAVs", American Control Conference, Portland, 2005, June, 3530-3535.

[19] R Saleri, M. Pierrot-Deseilligny, E. Bardiere and V. Cappellini, (2013). "UAV photogrammetry for archaeological survey: The Theaters area of Pompeii", Digital Heritage International Congress, 2013, Marseille, 497 - 502.

[20] F. Heintz, P. Rudol, and P. Doherty, (2007). "From images to traffic behavior - A UAV tracking and monitoring application", 10th International Conference on Information Fusion, 2007, Quebec, 1 - 8.

[21] Qing-Li Zhou, Youmin Zhang, Yao-Hong Qu and C. Rabbath, (2010). "Dead reckoning and Kalman filter design for trajectory tracking of a quadrotor UAV", International Conference On Mechatronics and Embedded Systems and Applications (MESA), 2010, Qingdao, ShanDong, 119 - 124.

[22] D. Kingston, R.W. Beard, and R.S. Holt, (2008). "Decentralized Perimeter Surveillance Using a Team of UAVs," IEEE Transactions on Robotics, 2008, December, 24:1394 - 1404.

[23] M. Rezaei, R. Mohsenipour, H. Nemati, S. M. Smailzadeh and H. Bolandi, (2013). "Attitude Control of a Quadrotor with Optimized PID", Intelligent Control and Automation, 2013, April, 335-342.

[24] M. Mohammadi and A.M. Shahri, (2013). "Modelling and Decentralized Adaptive Tracking Control of a Quadrotor UAV", RSI/ISM International Conference on Robotics and Mechatronics, 2013, Tehran, 293-300.

[25] M. Yasir Amir and V. Abbass, (2008). "Modeling of Quadrotor Helicopter Dynamics", International Conference on Smart Manufacturing Application, 2008, Gyeonggi-do, 100-105.

[26] Bohyung Han, Bohyung Han - POSTECH, http://cvlab.postech.ac.kr/~bhhan/kbf.html , 4 April 2015

[27] Kalman R.E, (1960) "A New Approach to Linear Filtering and Prediction Problems," ASME Transactions, 1960, D:82:33-45.

[28] Kimberly Tuck, http://www.edn.com/design/analog/4413345/2/Using-adaptive-filtering-to-enhance-capacitive-sensing-of-buttons-sliders , 4 April 2015

[29] Micheal Pfeiffer, A brief Introduction to Particle Filters, http://www.igi.tugraz.at/pfeiffer/documents/particlefilters.pdf, 12 Feb 2015

[30] S. Thrun, Burgard W., and D. Fox, Probabilistic Robotics., (1999-2000).

[31] G. Grisetti, C. Stachniss, and W. Burgard, (2007). "Improved Techniques for Grid Mapping," IEEE Transactions on Robotics, 2007, 23:34-46.

[32] ROS, http://wiki.ros.org/ , 3 March 2015

[33] Localize with a hokuyo laser range finder, http://www.generationrobots.com/en/content/52-localize-with-a-hokuyo-laser-range-finder , 16 February 2015

[34] Rospbian, http://wiki.ros.org/groovy/Installation/Raspbian , 13 February 2015

[35] DIY Drones, http://diydrones.com/profiles/blogs/705844:BlogPost:38393 , 19 March 2015

[36] Arducopter, http://copter.ardupilot.com/ , 19 March 2015

[37] Tf Package, http://wiki.ros.org/tf , 16 February 2015

[38] Hokuyo Node, http://wiki.ros.org/hokuyo_node , 15 February 2015

[39] Laser Scan Matcher, http://wiki.ros.org/laser_scan_matcher , 22 March 2015

[40] Hokuyo, https://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx_ug01.html , 17 February 2015

[41] DC Motor Speed: System Modeling, http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling, 10 January 2015

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname : Hayrettin ERTÜRK

Date of birth and place : 25.08.1989 - İzmir

Foreign Languages : English, German

E-mail : hayrettinerturk@gmail.com

EDUCATION

Degree	**Department**	**University**	**Date of Graduation**
Undergraduate	Electrical Engineering	Yıldız Technical University	2012
High School	Applied Sciences	Atakent Anatolian High School	2007

WORK EXPERIENCE

Year	**Corporation/Institute**	**Enrollment**
2014-…	Arneca Technologies	Software Expert
2013-2015	GFortech Engineering	Control System Designer – Cofunder
2011-2014	T-Soft E-Commerce Systems	Software Expert

PUBLISHMENTS

Conference Papers

1. H. Erturk, G. Tuna, T. Mumcu, K. Gulez, “A Performance Analysis of Omnidirectional Vision Based Simultaneous Localization and Mapping”, Intelligent Computing Technology Lecture Notes in Computer Science Volume, 2012, 7389:407-414

2. G.Tuna, T. Mumcu, K. Gulez, V. Gungor, H. Erturk, “Unmanned Aerial Vehicle-Aided Wireless Sensor Network Deployment System for Post-disaster Monitoring”, Emerging Intelligent Computing Technology and Applications Communications in Computer and Information Science, 2012, 304:298-305