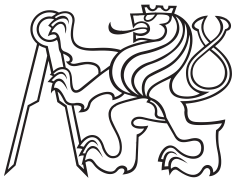
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Global Localization of Mobile Robot Using Randomly Placed Artificial Markers

Bc. Can Gundogdu

Supervisor: Ing. Karel Košnar, Ph.D.
December 2021

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Gundogdu Can** Personal ID number: **490535**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Global localization of mobile robot using randomly placed artificial markers

Master's thesis title in Czech:

Globální lokalizace mobilního robotu náhodně umístěnými umělými značkami

Guidelines:

1. Study different visual SLAM approaches
2. Study properties of RealSense T265
3. Design and implement global localization method using artificial markers randomly placed in environment
4. Verify functionality and measure presicion under reference localization system(vicon)
5. Evaluate properties and precision of the designed method

Bibliography / sources:

[1] Raúl Mur-Artal and Juan D. Tardós. 2017. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. IEEE Transactions on Robotics 33, 5 (2017), 1255–1262.
[2] Dominik Schlegel, Mirco Colosi, and Giorgio Grisetti. 2018. ProSLAM: Graph SLAM from a Programmer's Perspective. In Proceedings of IEEE International Conference on Robotics and Automation (ICRA). 1–9.
[3] Rafael Muñoz-Salinas and Rafael Medina Carnicer. 2019. UcoSLAM: Simultaneous Localization and Mapping by Fusion of KeyPoints and Squared Planar Markers. arXiv:1902.03729.
[4] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. 2010. A Tutorial on Graph-Based SLAM. IEEE Transactions on Intelligent Transportation SystemsMagazine 2, 4 (2010), 31–43.

Name and workplace of master's thesis supervisor:

Ing. Karel Košnar, Ph.D., Intelligent and Mobile Robotics, CIIRC

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **03.02.2021** Deadline for master's thesis submission: **04.01.2022**

Assignment valid until:
by the end of winter semester 2022/2023

Ing. Karel Košnar, Ph.D. Supervisor's signature	prof. Ing. Michael Šebek, DrSc. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt

Student's signature

Acknowledgements

I would like to express my gratitude and appreciation to my supervisor Ing. Karel Košnar, Ph.D. in preparation for the thesis. His experience, knowledge and professionalism in the robotics field is the reason I wanted to work with him. I couldn't have completed the thesis without his efforts and brilliant ideas which lead me to the solution. The moments I faced adversity he helped and supported me which I never will forget.

The hospitality of the Intelligent and Mobile Robotics department also enhanced my performance. I would like to thank everyone who helped me in the department.

I couldn't have conducted this work in the Czech Technical University without the extreme support and help from my mother Zehra Hayal Gundogdu and my father Ahmet Salih Gundogdu. They have been supporting me all my life in any way. I'm certainly grateful to have them.

Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography. Prague, December 2021

Prohlašuji, že jsem předloženou prácivypracoval samostatně, a že jsem uvedlveškerou použitou literaturu.V Praze, . prosinec 2021

Abstract

The thesis is based on implementing a localization technique using randomly placed artificial markers. In order to supplement the localization part, the obstacle avoidance techniques are studied, developed and finally tested in simulation. The state of art localization is presented and various simultaneous localization and mapping (SLAM) techniques are examined in detail. In addition, a localization method using artificial markers is then developed. The localization method consists of two stages: artificial marker detection and Extended Kalman Filter (EKF) to improve the precision of the implemented method. The localization system is tested in both simulation and with a real mobile robot utilizing an Intel RealSense camera while ground truth positions are provided by the Vicon motion capture system.

Keywords: artificial marker,localization, obstacle avoidance, SLAM, EKF, Intel RealSense, Vicon

Supervisor: Ing. Karel Košnar, Ph.D. CIIRC B:322, Jugoslávských partyzánů 1580/3 160 00 Praha 6, Dejvice, Czech republic

Abstrakt

Práce je založena na implementaci lokalizační techniky pomocí náhodně umístěných umělých markerů. Pro doplnění lokalizační části jsou studovány, vyvíjeny a nakonec testovány v simulaci techniky vyhýbání se překážkám. Je prezentován stav lokalizace a podrobně jsou zkoumány různé techniky simultánní lokalizace a mapování (SLAM). Dále je pak vyvinuta metoda lokalizace pomocí umělých markerů. Metoda lokalizace se skládá ze dvou fází: detekce umělých markerů a rozšířeného Kalmanova filtru (EKF) pro zlepšení přesnosti implementované metody. Lokalizační systém je testován jak v simulaci, tak se skutečným mobilním robotem využívajícím kameru Intel RealSense, zatímco pozemní pravdivé pozice poskytuje systém Vicon motion capture.

Klíčová slova: umělý fix,vyhýbání se překážkám, lokalizace, SLAM, EKF, Intel RealSense, Vicon

Překlad názvu: Globální lokalizace mobilního robotu náhodně umístěnými umělými značkami

Contents

Figures

Tables

Chapter 1

Introduction

In the past few decades, the conventional applications of mobile robots expanded on an immense scale. There are a variety of fields utilizing mobile robots such as medical facilities, customer service, rescue missions, agriculture, warehousing and delivery resulting in the creation of numerous mobile robot models. Localizing and navigation is a crucial task for most mobile robots. There is a wide range of developed techniques and implementations to solve the localization and navigation issues in mobile robots. These are mentioned as SLAM (Simultaneous Localization and Mapping) algorithms where each technique consists of different approaches. In this work, localization of the mobile robot using artificial markers will be implemented and proposed. Artificial markers are placed on the walls of the environment and global positions of the markers are known. The mobile robot is expected to localize itself in the environment while moving, which is achieved by estimating the position of the robot in the environment within the global coordinate frame. The simulation scene is illustrated below.

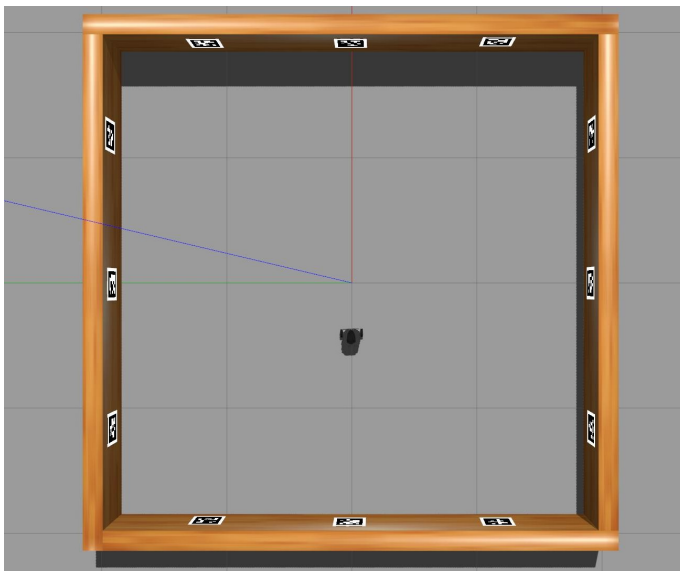


Figure 1.1: Map utilized for testing implemented localization technique in simulation.

The proposed work is tested with TurtleBot mobile robot depicted in Fig. 1.2 and on the Gazebo simulation. The algorithms and localization nodes are developed in Gazebo simulation. The final testing is executed on the real TurtleBot within the Vicon [28] equipped environment. The environment is an arena equipped with Vicon motion capture camera system, off-board computing workstations for design, tests and verification using aerial and ground robots. Vicon is one of the leaders in optoelectronic motion capture systems based on reflective markers. The main advantage of the Vicon setup is the accuracy of the motion detection system allowing to obtain very precise ground truth positions.

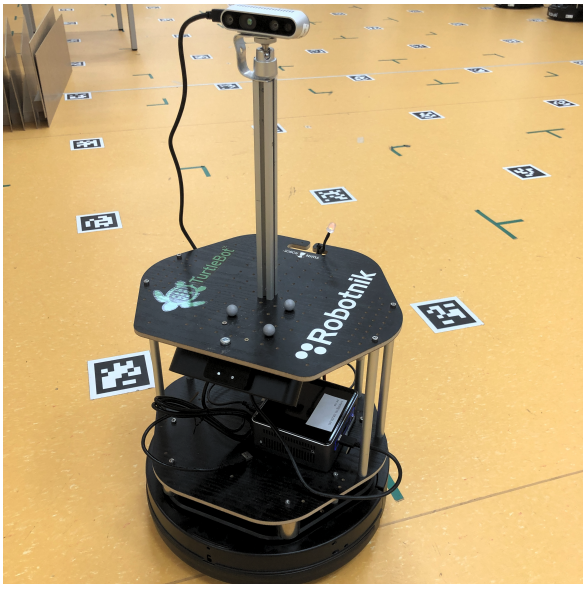


Figure 1.2: TurtleBot mobile robot used in the experiments.

Detection and identification while using artificial landmarks referred to as fiducial markers is a common utilization in Computer Vision applications and Augmented Reality applications [27]. There are a variety of fiducial markers available for developers such as ARToolkit, ARTags and AprilTags. The visualizations of the different artificial markers (tags) are shown below in Fig.1.3.

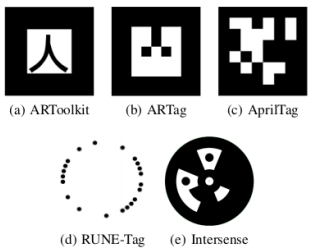


Figure 1.3: Different types of artificial markers [27].

The circular tags are less efficient than square tags because having four corners establishes efficient, robust and reliable localization of the markers using camera vision. Due to AprilTags being up to date in terms of software and its compatibility with the ROS (Robotics Operating System), the use of AprilTags is decided for the proposed work.

The mobile robot used for this work was equipped with Intel RealSense T265 [2] a high technology camera designed for computer vision application which is a tracking camera using a combination of MU and fish eye lenses. The field of view provided by the camera is even greater than the human field of view at 163°. Additionally, the T265 camera is equipped with an Intel RealSense V-SLAM visual processing unit. This unit allows the camera to synchronize the information provided by the IMU and the lenses in real-time. The processor runs V-SLAM algorithms for environment recognition and trajectory calculations. The T265 camera can be used simultaneously with other cameras in the Intel D400 series to develop various robotic vision applications such as 3D mapping, SLAM and obstacle avoidance applications. With the advice from my Supervisor, the camera is alternated to Intel RealSense D435. The key features of the T265 camera, such as V-SLAM and the wide-angle, will be not be utilized in the project. The intentional exclusion of the camera is due to the complications it causes in the fiducial marker detecting system because too many markers are detected in the same frame resulting in inconsistent localization of the markers.

Figure 1.4: Intel RealSense T265 camera [2].

The proposed work is prepared in the ROS (Robot Operating System), which is a set of software libraries and tools that help build robot applications [1]. It was first developed by Ph.D. students Eric Berger and Keenan Wyrobe at Stanford University. Later on, they joined Willow Garage to continue developing ROS. Despite the platform's name, it is not a conventional operating system, it requires Linux Operating System to install ROS. After the installation of ROS, it achieves low-level device control, message interface for communication between nodes, package management, visualization tools,

debugging and hardware extraction similar to other operating systems. ROS is a language agnostic, which means that the subprograms written (nodes) could be implemented in programming languages such as Python and C++ applications. In the proposed work, Python language is utilized.

Chapter 2

State of Art Obstacle Avoidance in Mobile Robots

Safe navigation is one of the significant aims of a mobile robot. In recent years, mobile robots are expected to autonomously navigate in the environment they are placed in. Autonomous navigation could be generalized in four stages: sensing, localization, planning and actuation. In the sensing stage, the mobile robot's sensors collect data from the surroundings and process the data in a way that this information becomes vital for the next stages. The localization stage consists of using the information from the sensing stage to determine the position of the robot in the global frame within a prior map. Once the robot position has been calculated, then the sequence of actions to reach the desired destination can be computed in the planning phase. Finally, the actuation stage executes the plan.

Local obstacle avoidance techniques frequently use a parallel process of the planning phase. The primary objective is to modify the global plan in order to avoid obstacles while stationed in an unknown map. In this work the obstacle avoidance usage is slightly different, generally the prior map of the environment is provided for autonomous navigation although prior map is not available for the obstacle avoidance implemented. Thus, there isn't any global planning method in this work. Local planning is utilized to avoid the obstacles within the map.

There are plenty of various algorithms for local obstacle avoidance meant for mobile robots. Each technique has a different approach to the obstacle avoidance concept. Some of them are barely reactive while others use the information provided by the global planner to alternate the path followed by the mobile robot. In this work, different types of local planning algorithms have been studied and compared.

2.1 Various Obstacle Avoidance Techniques

This section is intended to analyze and describe three techniques of obstacle avoidance for mobile robots. The first technique is the Artificial Potential Field Algorithm, the second is the Vector Field Histogram and the last is the Smooth Nearness Diagram. These techniques have the common attributes of

using the on-board sensors to locate the obstacles around the close proximity of the mobile robot and afterward utilizing this information to navigate to the destination point. In the following subsections, the obstacle avoidance techniques are described.

2.1.1 Artificial Potential Field Algorithm

The local obstacle avoidance by using artificial potential fields was first proposed by O.Khabib [4] in 1995. This method is characterized by a simple and powerful approach. The mobile robot (MR) is considered a particle engaged in a potential field generated by the obstacles and the destination goal. The goal establishes an attractive potential for MR. On the other hand the obstacles detected from the environment establish a repulsive potential for MR. This potential field can be interpreted as an energy field and thus it is gradient, meaning at each point is a force. The MR immersed in the potential, drives to the goal due to the attractive potential results from the gradient of attractive potential field caused by the goal. Meanwhile, the obstacles generate gradient repulsive potential field to repel the robot away from the obstacles leading MR to a safe path to the goal. The gradients can be thought of as forces acting on the positively charged MR, therefore the negatively charged goal is attracting the MR despite positively charged obstacles resulting in repulsive forces for MR.

The MR is considered as a moving particle in an n-dimensional space R^n. For the sake of simplicity, the MR is moving in a 2-axis coordinate frame (x,y) and the position of the robot is represented by q. Then the artificial potential field where the robot moves is a scalar function denoted as $U(q)$. Thus the generated superposition of attractive and repulsive potentials can be described by [6];

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{2.1}$$

The Artificial Potential Field Algorithm's (AFPA) attractive potential is assumed to be zero at the goal point and to increase as the robot is far away from the goal. The repulsive potential, associated with each obstacle, is very high (infinity) in the close vicinity of the obstacles and decreases when the distance to the obstacle increases [5].

$$F(q) = F_{att}(q) + F_{rep}(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q) \tag{2.2}$$

$U(q)$ is assumed to be differentiable thus at each q, the gradient of the potential field can be described as $\nabla U(q)$ which is a vector that points in a direction that $U(q)$ maximally increases. F_t is also a vector, points in a direction that each q point maximally decreases the $\nabla U(q)$.

$$U_{att}(q) = \frac{1}{2} k_{att}(q) d^2_{goal}(q) \tag{2.3}$$

$U_{att}(q)$ is the standard parabolic that increases quadratically while the distance to the goal decreases. Where k_{att} and q_{goal} is a scaling factor, $d_{goal}(q) = \|q - q_{goal}\|$ parameter is the Euclidean distance from the robot to the goal.

$$\nabla U_{att}(q) = k_{att}(q - q_{goal}) \tag{2.4}$$

$\nabla U_{att}(q)$ is a vector depending on the difference of q and q_{goal} that is pointing away from the q_{goal}. When the distance from the goal is expanding, then the magnitude of the attractive potential field is also increasing.

$$U_{rep_i}(q) = \begin{cases} \frac{1}{2}k_{obst_i}(\frac{1}{d_{obst_i}(q)} - \frac{1}{d_0})^2 & if: \ d_{obst_i}(q) < \ 0 \\ 0 & if: \ d_{obst_i}(q) \geqslant \ 0 \end{cases} \tag{2.5}$$

Minimal distance from q to the obstacle i is denoted as $d_{obst_i}(q)$, d_0 is the obstacle influence threshold and k_{obst_i} is the scaling factor. The negative of the gradient of the repulsive potential is $F_{rep_i}(q)$ given as:

$$F_{rep_i}(q) = \begin{cases} k_{obst_i}(\frac{1}{d_{obst_i}(q)} - \frac{1}{d_0})\frac{1}{d^2_{obst}(q)}\frac{q-q_{obst}}{d_{obst_i}} & if: \ d_{obst_i}(q) < \ d_0 \\ 0 & if: \ d_{obst_i}(q) \geqslant \ d_0 \end{cases} \tag{2.6}$$

The APFA is a simple, robust technique for local obstacle avoidance. The main principle behind this approach is energy fields types. When the MR is using this technique in an unknown environment for obstacle avoidance it might not be ideal depending on the environment. The major drawback of this technique is in some situations MR can get stuck in local minima. These issues may arise from the symmetry of the environment and the concave obstacles as well as the MR oscillation while traveling in narrow passages depicted in Fig. 2.1.

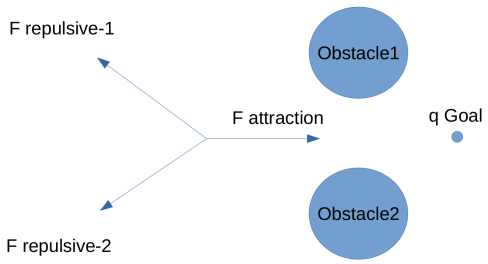


Figure 2.1: Local minimum of the potential field caused by symmetry.

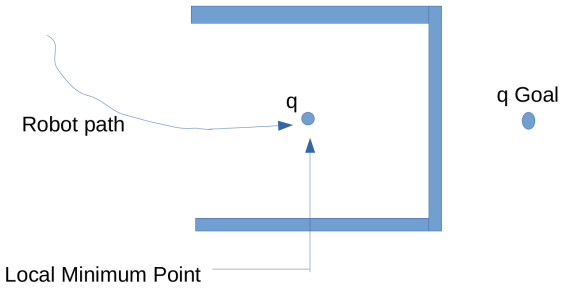


Figure 2.2: Local minimum of the potential field caused by a concave obstacle.

In Fig. 2.2 the MR is approaching the concave obstacle and the goal is attracting the MR. Then at the point q the attractive force to the goal is symmetric to the repulsive force generated from the obstacle. This occasion results in MR achieving local minima and MR can't reach the goal.

2.1.2 Vector Field Histogram

The Vector Field Histogram (VFH) is an on-line obstacle avoidance technique that prevents collision within unknown obstacles and environments. The technique takes advantage of the measurement instruments mounted on the mobile robot (MR) and with the information forwarded from the sensors such as a laser scanner that will be used in detecting the objects around the MR. Meanwhile the MR is steering to reach the destination point and avoiding the obstacles in the environment. This technique was first mentioned by Borenstein and Korem [7] in 1991.

The VFH technique utilizes the two-dimensional Cartesian histogram grid as a world model only considering the x and y axis. The world model is continuously updated according to the range data sampled from the MR's on-board laser scanner. The VFH uses a two-step process for data reduction in order to compute the velocity commands to be forwarded to the robot to reach the goal point. The first step consists of establishing a constant sized two-dimensional histogram grid from the current location of the robot which is then reduced to one-dimensional polar histogram establishing computational advantage. Each sector of the polar histogram is a value served as polar obstacle density in that direction. In the second step, the VFH method selects the most suitable sector in all polar histograms with a low polar density, meaning the safest direction to the goal according to the data received from the sensors.

The application of the VFH is generally finalized in three stages: the first stage is building the two-dimensional Cartesian histogram grid representation

for the obstacles retrieved from the sensors. The data from sensors such as laser scanners or ultrasound sensors used to construct the occupancy of the cells that correspond to the distance d is incrementally updated, increasing the certainty value of the cells. This phenomenon is depicted in Fig. 2.3 below.

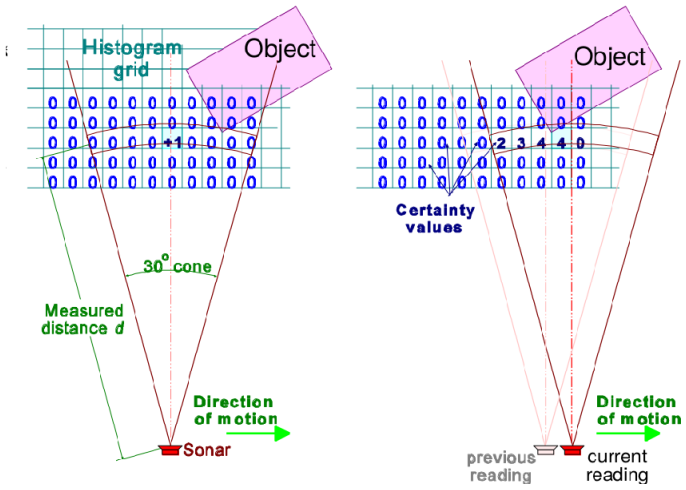


Figure 2.3: Building the two-dimensional histogram grid map [7].

The second stage of the VFH exploits the two-dimensional Cartesian histogram derived from the previous stage and transforms this histogram grid map to a one-dimensional structure for simple computational purposes. Utilizing the entire grid map for computation is therefore unnecessary to preserve and to secure the information gathered from sensor data. This data should be restricted to a window C which is called an active window.

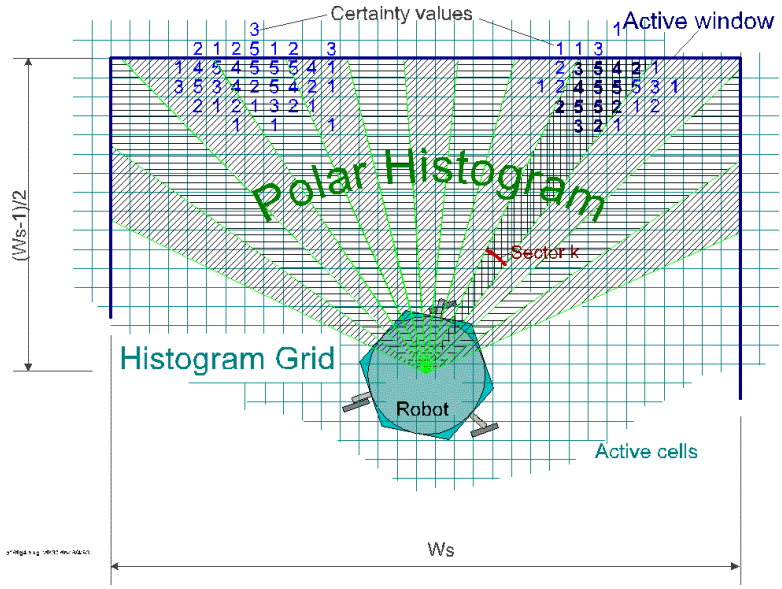


Figure 2.4: Active cell mapping on the Polar Histogram[7].

The active window then can be transformed into a one-dimensional polar histogram. Thus the local map of the environment could be represented as C^* and it is constantly updated according to the incoming inputs from the sensor data. This occasion is depicted in Fig. 2.4. Active grid C^* is mapped compromising n angular sections with the width α. Partition of angular sections is necessary for the evaluation of the one-dimensional polar histogram.

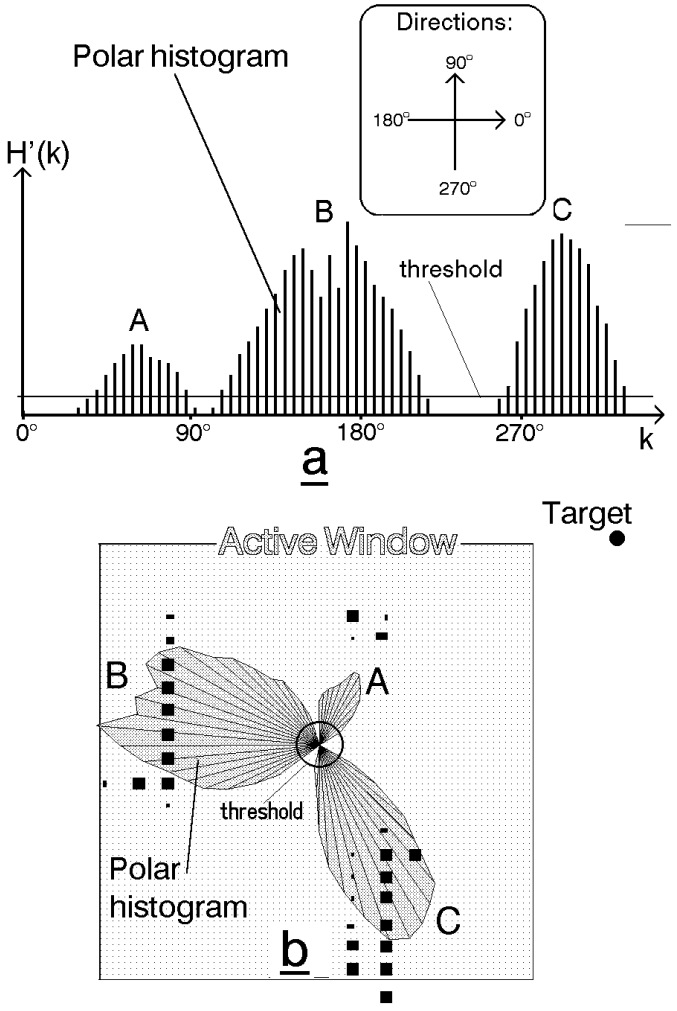


Figure 2.5: Representation of one-dimensional polar histogram [7].

The last stage of the VFH consists of determining the steering angle of the MR according to the one-dimensional polar histogram. This is used to reach the destination while simultaneously adjusting the velocity of the MR from the obstacle polar density. The polar histogram contains "valleys" which are the low polar density zones while "peaks" represent the high polar density zones. The valleys with enough space for the robot to pass are considered as candidate valleys which can be computed by a threshold value S_{max}. If the valley is higher than the threshold value then it is wide enough

for the robot to pass through. This threshold can be set according to the dimensions and kinematic constraints of the MR. The threshold should be selected accurately and should be tuned, otherwise, this method will not be optimal. The VFH method can overcome some issues that are present in Potential Field Algorithm (PFA). This technique solely relies on the on-line input from the sensor data due to non-existent attractive or repulsive forces.

2.1.3 Smooth Nearness Diagram

The Smooth Nearness Diagram (SND) is a local obstacle avoidance technique that was inspired by and improved version of the Nearness Diagram (ND) presented [9] in 2004. ND was the first reactive navigation approach based on gaps. The ND approach is simple and not computationally demanding. ND avoids collision and local trap occasions without determining which areas of the environment are connected. It reduces the local minima problem which some local obstacles avoidance techniques tend to have.

The SND is an improved edition of the ND and they both utilize the concept of gaps. The method was first introduced in 2008 by J. W. Durham and F. Bullo [10]. SND corrects the oscillatory motion of the robot which was present on the ND algorithm. The gaps mentioned are the discontinuities in the depth of obstacles in the close vicinity of the robot which illustrates the potential free paths in the blocked areas of the environment.

A gap is created at an angle where two adjacent depth measurements are separated by r the diameter or one of the measurements can't detect obstacles in the range. In Fig. 2.6 it's observed that the (a) and (b) are the gaps in the environments. The left gap is indicating the obstacle measured is on the left side of the robot, also there might be a blocked area on the left gap. The opposite is declared for the right gap.

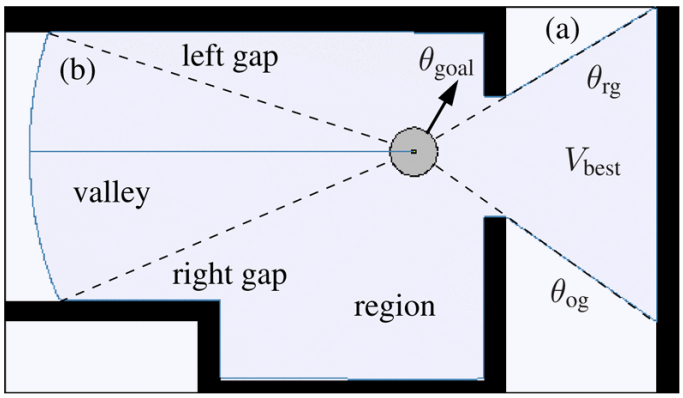


Figure 2.6: Representation of regions, valleys and gaps for SND [10].

The different regions of the environment can be defined by the consecutive gaps, the objective is to select the "valley" that represents the open space for the robot to traverse. The valley can be located on the left gap, that is left

side of the robot or can be located in the right gap that is the right side of the robot. Then all the regions are assembled, after that comparison of regions for the selection of the optimal path is considered. The valley containing the gap with the best heading to reach the goal is selected for non-collision path. The major difference between ND and SND is, SND considers the safety distance from the robot meaning the threat of hitting obstacles will arise while getting closer. In return, this threat factor measured from obstacles will allow the SND to compute the desired heading angle to follow the obstacle free path.

2.2 Implemented Obstacle Avoidance Techniques

For the proposed work two obstacle avoidance techniques are described and implemented. The first method is the Dynamic Window Approach and the other one is Bug-0 Algorithm. They both are popular and effective techniques for obstacle avoidance. This section is meant for explanations and illustrations of the obstacle avoidance techniques implemented. The techniques are tested and results are compared in the next section.

2.2.1 Dynamic Window Approach

The Dynamic Window Approach (DWA) is a robust and efficient technique for local obstacle avoidance purposes. This method was first mentioned in 1997 by D. Fox, W. Burgard and S. Thrun [11]. The DWA is based on the dynamic motion of the MR. The technique utilizes kinematic constraints to determine the optimal velocities and the accelerations of the robot. The algorithm is summarized two by phases, the first is the generation of the search space and the second phase is selecting the optimal path from the search space retrieved from the first phase.

In order to begin DWA it's necessary to define various parameters that will be used throughout the algorithm. These time-invariant parameters are robot radius, maximum speed of the robot and other robot dependent information. After that, inputs from the sensor data (laser scanner data) should be obtained also including the global position. Then the program goes to an infinite loop until it's terminated or reached the goal. Meanwhile, with each reading from the sensors: the laser data, position and current velocity is updated.

After setting the initial parameters the next step is to generate a search space of angular velocity and linear velocity pairs. For this step equations below are proposed in order to compute the minimal and maximal velocities of the robot which is a restriction of the minimum and maximum speed of the mobile robot used.

$$v_{max} = v_c + v_a, v_{min} = v_c - v_b \tag{2.7}$$

$$w_{max} = w_c + w_a, w_{min} = w_c - w_b \tag{2.8}$$

The parameters in the equations above (Eq. 2.7-2.8) are denoted as maximal translation v_a, rotational acceleration w_a, current linear velocity v_c and angular velocity w_c and maximal deceleration v_b executable by the motors. v_{max} and w_{max} are the maximum linear velocity and angular velocity, on the other hand v_{min} and w_{min} are the minimum linear velocity and angular velocity, all of the parameters are derived from the robot dynamics.

The DWA takes directly the dynamics of the robot into account. Thus the parameters should be tuned precisely for different robot models used or for different maps used. The tuning of the parameters is mandatory as long as optimal results are expected from the robot motion.

After the parameter initialization, the motion model of the MR is computed from the Eq. 2.9-2.11 below, allowing to predict the robot position in time with the robot dynamics defined. This robot position later provides the active dynamic window of the robot which describes robot motion in the future time, allowing to compare the possible trajectories in order to select the optimum trajectory to reach the destination. Meanwhile, dynamic window is computed from the dynamic specifications of the robot from the Eq. 2.7-2.8 then these dynamic windows are restricted to robot capabilities so the final dynamic window result is the admissible speed of the robot that can be reached within a short time interval given the limited accelerations of the robot. Which is then compared with obstacles retrieved from the sensors to find a safe trajectory and label trajectories as collision path or free path.

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t).cos\theta(t).dt \tag{2.9}$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t).sin\theta(t).dt \tag{2.10}$$

$$\theta(t_n) = \theta(t_0) + \int_{t_0}^{t_n} w(t).dt \tag{2.11}$$

The obstacles are retrieved from a 360-degree scanner and it's partitioned to different regions incremented by 16 degrees. The average range value of the regions is computed for simplicity. Furthermore, the angles of the obstacles are calculated with respect to the global frame. The angle and the range information of the obstacle allow predicting the position of the obstacles in terms of point in a two-dimensional coordinate frame . A naive but effective

approach is implemented for obstacle avoidance, the obstacles are perceived as a point thus the MR radius is compared to the distance to obstacles ensuring if the difference is smaller than the robot radius then the collision is inevitable then the trajectory is calculated and labeled as collision path then discarded. This process is recursive thus each trajectory is compared to each obstacle generated from the laser scanner for labeling.

The collision free paths are appended to the list. Then paths are characterized by the three parameters: velocity (υ_{vel}) , clearance ($\upsilon_{clearance}$) and target heading (υ_{ang}). The velocity refers to the speed of the trajectory, if fast motion is expected then the velocity parameter should be higher on the other hand clearance is a parameter that describes the lack of obstacles in that path thus for a safer path the parameter should be kept higher. And the last parameter to be considered is the target heading of the robot, this parameter ensures the robot is moving towards the goal location to decrease the detouring of the robot. It is maximal if the robot moves directly towards the target and the distance to the goal is not prolonged in this case. The parameters are mentioned in the paragraph are utilized to select the optimal path for the desired motion of the MR, for each safe trajectory the parameters are calculated then the objective function is generated to integrate the parameters for optimal trajectory selection. The objective function is shown below.

$$Score(v, w) = \lambda_{ang}\upsilon_{ang} + \lambda_{vel}\upsilon_{vel} + \lambda_{clearance}\upsilon_{clearance} \tag{2.12}$$

With the highest sum, the optimal trajectory is chosen w_{best} and v_{best} and the robot starts moving with these parameters. Fig. 2.7 describes the selection of the optimal path of DWA in a very simple simulation. The yellow lines are possible trajectories and the red trajectory is determined as the optimal trajectory to reach the green goal.

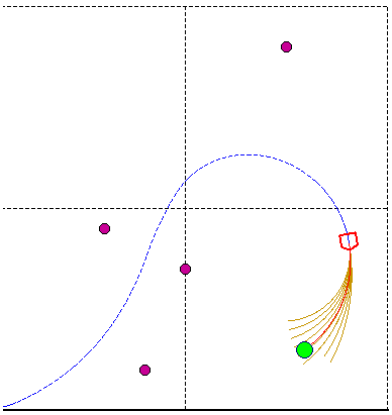


Figure 2.7: DWA trajectory selection while moving toward the green goal (simple robot simulation).

2.2.2 Bug Algorithms

Bug algorithms are inherited from the classical Piano Mover's problem which consists of expecting the robot to arrive at the destination point without colliding with the obstacles in a known environment [12]. But when the environment is unknown the approach is not sufficient to resolve the complexity of the environment. At this point Pledge algorithm proposed by John Pledge succeeds to resolve the issues with Piano Mover's problem nevertheless still needs improvements due to its poor functionality caused by infinite loops (traps).

The Bug algorithms come in handy to correct its ancestor's approach. The name is inspired by how bugs (cockroaches, etc.) in nature navigate in an environment using their antennas and other senses to collect information about their surroundings and navigate to their destination point. The concept of Bug-0, Bug-1 and Bug-2 algorithms were introduced by Lumelski and Stepanov [13].

The Bug Algorithms are simple non computationally demanding methods that solely rely on the local sensor data to determine a safe path to the destination point in two-dimensional space. The method utilizes small memory space allowing robust and practical computation. Bug-0, Bug-1 and Bug-2 algorithms are an efficient solution to reaching the global destination point that uses the local sensor information (laser scanner) from the sensor mounted on the MR. All of the Bug algorithms are based on the same principle that the robot moves towards its destination when the obstacle is visible within the sensors then the robot circulates around the obstacles without any collisions to continue to drive towards the goal.

The main objective of the Bug algorithm is to follow the boundaries of the obstacle without colliding and to arrive at the global destination point. In addition, the Bug algorithms generally hold these three assumptions: the MR achieves perfect localization, the MR is considered as a point in two-dimensional space, the data from the sensors are impeccable [14].

Bug-0 Algorithm

The Bug-0 algorithm is a primitive planner, which doesn't demand any memory space. The mobile robot (MR) initially has the knowledge of its starting point $S_{MR}(x, y)$ and the destination point $G_{MR}(x, y)$ which are points in a two-dimensional Cartesian coordinate system. The Bug-0 algorithm can be summarized in the steps described below.

Step 1: If the following conditions are not met, go directly to G_{MR}

Step 1.1 : Destination reached, stop the process

Step 1.2 : Obstacle encountered, then apply Step 2

Step 2: If the following conditions are not met, then follow the wall in the selected direction (clockwise or counter-clockwise)

Step 2.1 : Destination reached, stop the process

Step 2.2 : Mobile robot's heading is towards the goal point and no obstacles in the way then apply Step 1

The robot's motion behavior when the Bug-0 algorithm is present is depicted below in Fig. 2.8, The red lines imply the path followed by the robot while avoiding the obstacles when encountered.

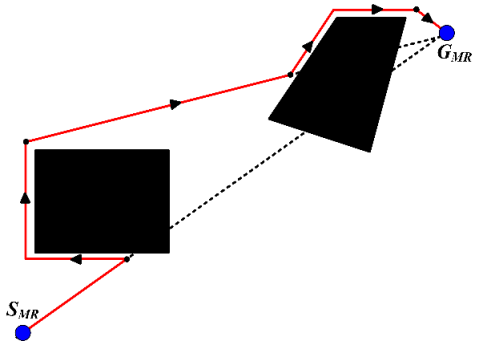


Figure 2.8: Bug-0 Algorithm behavior Algorithm with 2 unknown obstacles.

The black lines indicate a path that might cause the robot to get stuck in an infinite loop. The concave obstacles in the map might be problematic for the robot to navigate towards the goal. Labyrinths and complex maps are not ideal for this algorithm because they can get stuck also. However, the map used in this work is not as complicated as Fig. 2.9 thus implemented algorithm works properly as intended.

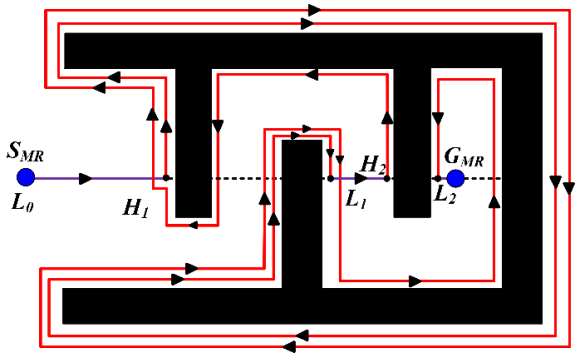


Figure 2.9: Bug-0 Algorithm behavior inside a complex map.

2.3 Experimental setup for Obstacle Avoidance

This section consists of describing the results and compares results obtained from the tests conducted. Both algorithms are tested under the same circumstances the time is tracked for a mutual comparison. The behavior of the mobile robot with two different obstacle avoidance technique are also described in this section.

The implemented local obstacle avoidance algorithms are tested in the same environment and scene using the Gazebo Simulation which is one of the software backbones of Robot Operating System (ROS). The Gazebo simulation utilizes a robust physics engine contributing realistic results allowing the users to easily adapt the robot to real-world testing.

The robot that will be used is the TurtleBot Burger model. The dimensions of the robot are a small size robot (Size: 138mm x 178mm x 192mm) but indeed equipped with a laser scanner sensor allowing the robot to depict the close vicinity environment from the laser readings and the odometry information is provided by the Gazebo simulation.

Figure 2.10: Turtlebot Burger used in Simulation.

The experiments are based on placing the MR in an unknown environment without a prior map. The MR is expected to reach a global destination point selected by the user while avoiding any collisions with obstacles. A laser scan capable of 360-degree measurement, is used for sensor data. Rviz is a 3D visualizer for the Robot Operating System (ROS) framework. It is convenient to utilize Rviz to debug or at least visualize the sensor data

for better comprehension of the simulation. Also, the destination point is selected from the Rviz tool (2D-nav goal) by clicking a point inside the grid map. Fig. 2.11 demonstrates laser scanner data from the MR visualized by the Rviz tool.

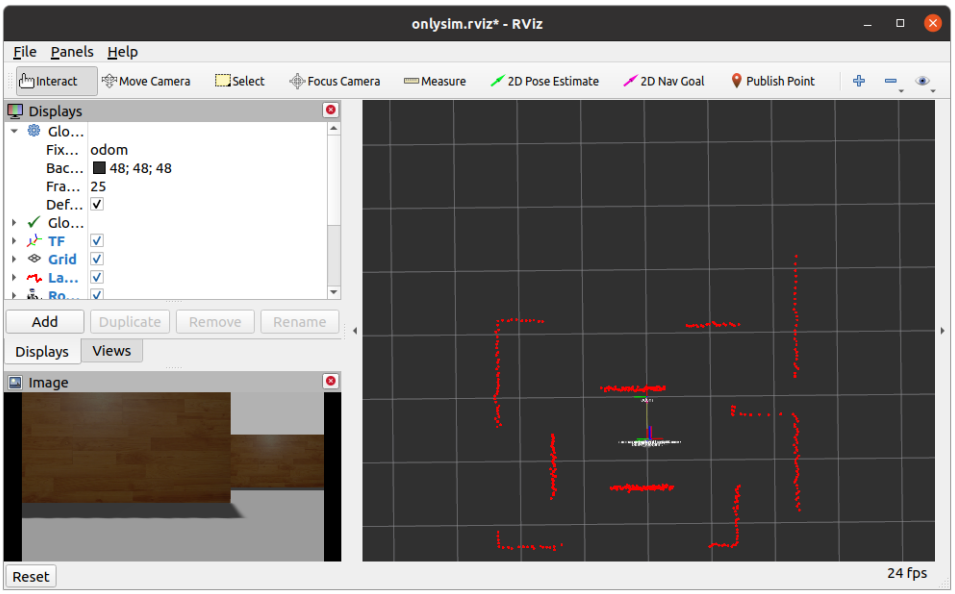


Figure 2.11: Rviz tool: Laser scanner and Camera visualization.

The environment below is used for the experiments on the Gazebo Simulation for the purpose of testing local obstacle avoidance techniques developed in this proposed work. The map dimensions is 5m by 5m.

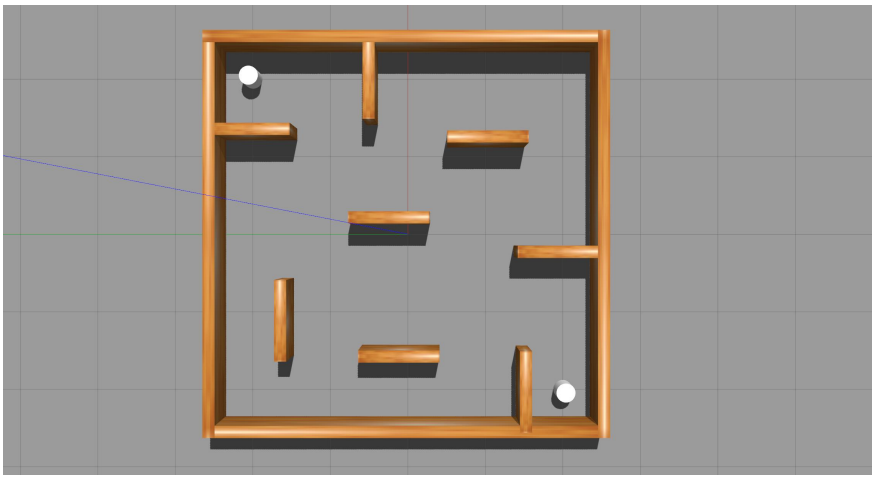


Figure 2.12: Utilized map for obstacle avoidance testing.

Obstacle avoidance techniques implemented are tested in the experiment setup described which is constant for the rest of the tests.

2.4 Experimental results of Obstacle Avoidance methods

The implemented Bug-0 and DWA techniques are tested under the same circumstances in the Gazebo Simulation. The same robot model and map, are preserved for better comparison of the methods. The robot motion behavior is explained for both algorithms. The results are discussed and the tables of results are presented at the end of the chapter.

2.4.1 Bug- 0 Algorithm Results

The implemented Bug-0 algorithm is tested according to the setup explained previously. The Bug-0 algorithm consists of two general methods. The first method is moving towards the destination and if the obstacle is encountered then the second method prevails which is the "follow the wall" method. Until the followed wall is cleared the robot will continue to circulate around the obstacle detected. After bypassing the obstacle the robot will integrate the first method "move towards goal". Switching between the two methods covers the main loop of the algorithm and MR is able reach the goal by interchanging between the methods. This is a brief description of the method implemented.

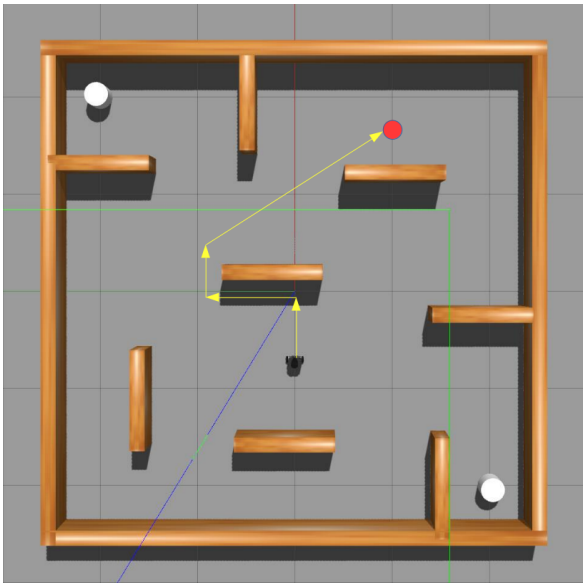


Figure 2.13: Bug-0 Algorithm behavior inside the utilized map.

Fig. 2.13 above is presenting MR motion with Bug-0 algorithm integrated. The MR is placed at its starting position as in the figure then the red point is selected for the destination. The yellow lines are trajectory followed by the MR. The robot moves directly until the obstacles are noticed then the "follow

the wall" method is activated. It's clear that the MR rotated left instead of right, which is certainly is not optimal considering the right side has a shorter distance to the goal. Nevertheless, the solution converges to one referring to, solution exist thus time is not a massive factor as long as solution exists in the Bug-0 algorithm.

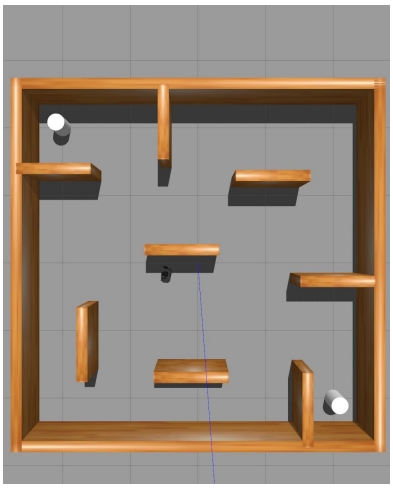


Figure 2.14: Bug-0 Algorithm "follow the wall" stage is active.

The Fig. 2.14 above represents the robot motion while "follow the wall" method is activated.

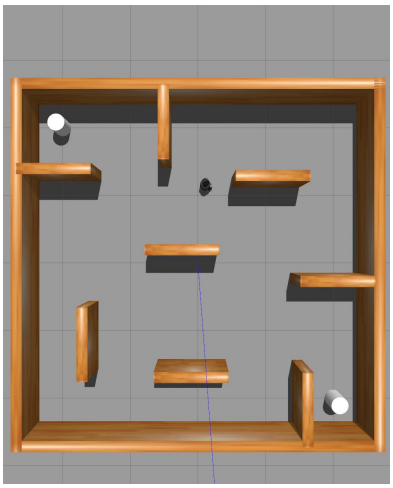


Figure 2.15: Bug-0 Algorithm "move towards goal" stage is active.

The Fig. 2.15 above represents the robot motion while "move towards goal" method is present.

2.4.2 DWA Algorithm Results

The DWA algorithm is also tested in the same map as the Bug-0 algorithm, the robot is placed in the same starting position and asked to move to the designated goal point without collisions. The Fig. 2.16 below is depicting the selected trajectory (blue) by the MR in action.

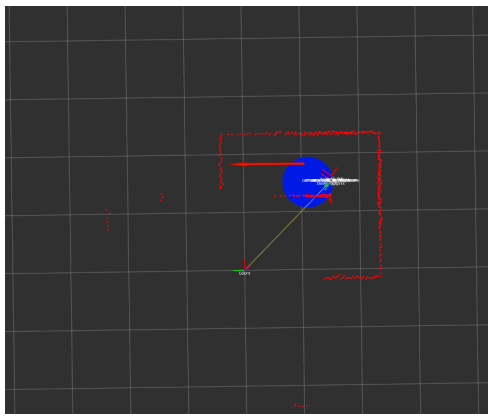


Figure 2.16: DWA Algorithm trajectory selection (blue circle) captured from Rviz tool.

The DWA algorithm is based on many parameters as explained earlier and they need adjusting in order for MR to have optimal and non-collision motion. The Fig. 2.17 below illustrates the path followed by MR while DWA algorithm is present.

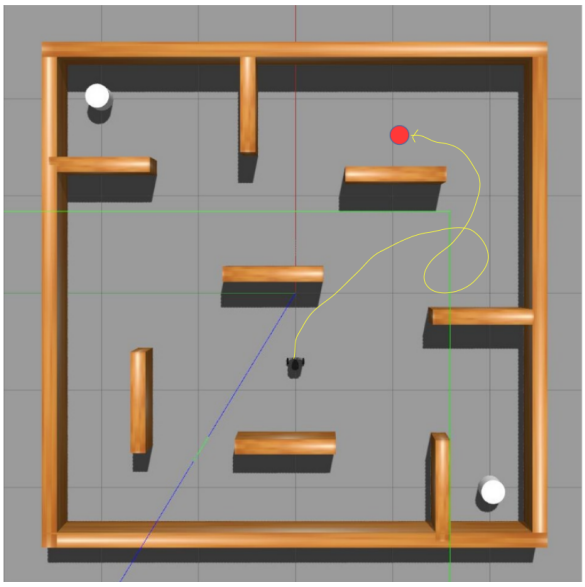


Figure 2.17: DWA algorithm behavior inside the utilized map.

The DWA is an effective obstacle avoidance method. The solution converges to one although the path is obviously not optimal caused by the loop motion of the MR. The reason behind that unnecessary motion is the MR can't find

a clear path to the target even though there is enough space for robot to pass. Thus MR continues the motion with admissible safe trajectories until the robot rotated enough to confirm the clear path and drives towards it. The desired agile and smooth motion of the robot can be achieved by improving the obstacle recognition approach which then will lead to flawless obstacle avoidance technique.

2.4.3 Obstacle Avoidance Algorithms Comparison

The DWA and the Bug-0 algorithms are configured in the same map with the same robot model. Fig. 2.18 illustrates the three destination points denoted as A, B and C. The robot is expected to reach the destination and the duration are collected for each run and described in Table 2.1-2.3. The destination points are tested with each algorithm numerous runs to gather time data.

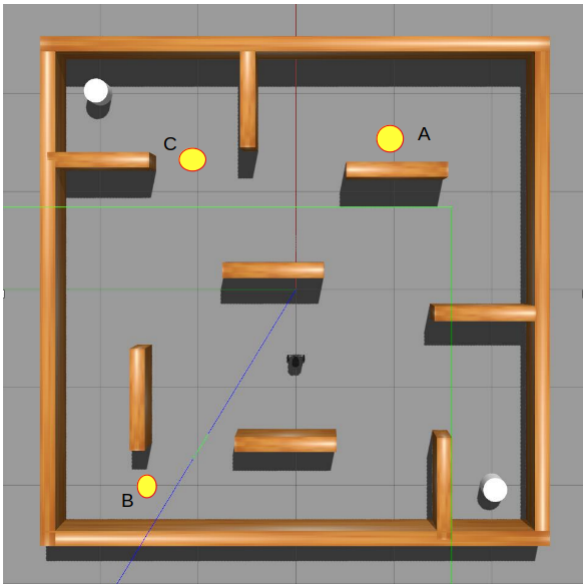


Figure 2.18: Points A, B and C are the designated destinations points for evaluation.

The DWA is a more complex and technical approach to the Bug-0 algorithm due to the dependency on the dynamics of the MR. Thus DWA is expected to be more agile and robust although only in destination A the DWA algorithm prevails. B and C destination points are reached with Bug-0 faster and more reliable than DWA algorithm. The DWA algorithm is unpredictable and sometimes inconsistencies occur in the motion of the robot which leads to massive changes in the duration. Nevertheless, both methods provided two different solutions to obstacle avoidance techniques, the Bug-0 algorithm is perceived as the more reliable and robust method due to inconsistencies of the DWA algorithm.

	Destination A			
	t min	**t max**	**t exp**	**Success Rate**
DWA	15.2 s	22.7 s	17.8 s	80%
Bug-0	38.1 s	44.0	40.3 s	100%

Table 2.1: Results of Bug-0 and DWA algorithm arriving to the destination point A.

	Destination B			
	t min	**t max**	**t exp**	**Success Rate**
DWA	21.8 s	40.9 s	36.3 s	70%
Bug-0	17.1 s	20.5 s	18.4 s	100%

Table 2.2: Results of Bug-0 and DWA algorithm arriving to the destination point B.

	Destination C			
	t min	**t max**	**t exp**	**Success Rate**
DWA	44.8 s	55.0 s	48.1 s	95%
Bug-0	29.4 s	33.6 s	30.9 s	100%

Table 2.3: Results of Bug-0 and DWA algorithm arriving to the destination point C.

Chapter 3

State of Art Localization in Mobile Robots

Localization is one of the backbones of mobile robot (MR) applications. For the MR to operate safely and effectively, an accurate and robust localization system is critical. The main objective of localization is determining the MR position with respect to a global reference frame generally referred to as "map". The localization of the robot is essential for deciding what to do further. The following motion of the robot is depending on the localization data and if it isn't precise enough then the motion is not executed as planned. Thus for the majority of MR, the MR applications rely on a robust localization.

Generally, information for estimating the robot location is collected from on-board sensors of the MR which are then utilized to perceive the surroundings and its own motion. There are plenty of possibilities for sensor selection and algorithm selection to localize the robot. Moreover, sensors include GPS (Global Positioning System), camera, laser scanner, ultrasonic sensor, wheel encoder, IMU (Inertial Measuring Unit), etc. The MR that is equipped with one or more of the sensors previously mentioned is able to compute an estimate of its location relative to where it's started if a mathematical model of the motion is known. The term is called odometry or dead reckoning.

The errors present in the sensor measurements and the motion model, generates robot location estimates obtained from dead reckoning more and more unreliable as the robot navigates in its environment [15]. The formulation of the localization problem depends on the sensors equipped by the MR or the map of the environment. In one example, the map is an form of occupancy grid that allows to differentiate between occupied or free spaces of the map. Alternative example is, partial map of the environment is used such as feature map or landmarks present in environment, then used to localize the robot from the readings transferred by the sensors to find accurate positions.

In this chapter, three different localization techniques are studied and described such as GraphSLAM, ICP-SLAM and ORB SLAM 2 also localization types are explained. Further on a localization technique is developed that incorporates artificial markers that are presented in the next sections.

3.1 Various Localization Techniques

There are multiple techniques available for localizing the MR. The following subtitles explain the types of localization techniques. The localization implemented using artificial markers is a static environment passive localization technique.

- Passive or active localization

  The active localization is achieved by on-board sensors (sensing) of the MR being able to alternate the robot motion in the environment. On the other hand, passive localization relies on sensors for estimating robot position although it doesn't affect robot motion.

- Known map or unknown Map localization

  If the MR is given a known map of the environment prior to localizing it is referred as a known map. It allows utilizing the map to compute estimated position by extracting features of the map and also the sensor inputs. In unknown map which is generally a SLAM (Simultaneous Localization and Mapping) problem. The MR is expected to generate a map of the environment while it's exploring the map by its sensor data.

- Dynamic or static environment

  The dynamic environment consists of moving objects and obstacles which the MR must perceive then react with the alternating trajectory of MR to achieve non-collision motion. The static environment is more predictable and generally the map is constant. In rare cases of moving objects in a static environment, the motion of the object should be known beforehand.

3.1.1 Iterative Closest Point SLAM

One of the most commonly utilized methods is the Iterative Closest Point (ICP) SLAM developed by Mckay and Besl [16] in 1992. The technique takes advantage of transformation between the two overlapping point clouds in estimation of MR position. It recursively computes the registration by finding transform parameters that minimize the Euclidean distance of corresponding points, which are assumed to be the nearest neighbor points [17]. The major drawback of this technique is its high demanding computational power and speed caused by the closest point detection.

The method consist of three stages. At first as an input point set of unregistered scan $P = \{p_1, p_2, p_n\}$ $p_i \in R^3$ and $X = \{x_1, x_2, x_n\}$ $x_i \in R^3$ which refers to the base scan retrieved to align P against.

The first stage is finding corresponding point x, in X for every p_i to find the minimal distance from point set X. This process can be explained in the formulation below.

$$d(p, A) = min(d(p, a_i)) : i \in \{1, 2, ..N\} \tag{3.1}$$

The second stage integrates the set of correspondences achieved previously to obtain a transformation A(q,t) , "q" referring to quaternion and "t" for translation vector. The transformation obtained must minimize the distance between the corresponding points. Mean square error function implemented to ensure the minimization using Eq. 3.2. N_p refers to the number of correspondences.

$$e(q, t) = \frac{1}{N_p} \sum_{i=1}^{N_p} |x_i - R(q)p_i - t|^2 \tag{3.2}$$

The third stage utilizes the Eq. 3.3 to acquire the optimal transformation (alignment) between point sets.

$$A(q, t) = \underset{q,t}{\operatorname{argmin}}(e(q, t)) \tag{3.3}$$

The transformation matrix obtained is represented as below. T is a four by four matrix while t is one by three, and R is three by three rotational matrix.

$$T = \begin{bmatrix} R & t \\ 0^t & 1 \end{bmatrix} \tag{3.4}$$

The ICP algorithm is finalized by merging the stages one through three and loops until the program stops executing. The Fig. 3.1 below depicts the before and after results of the ICP matching.

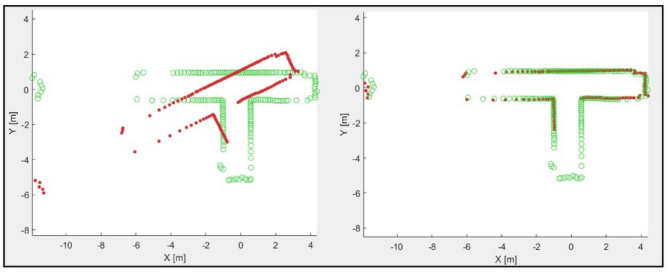


Figure 3.1: Two registered point cloud scans, ICP matching [18].

Algorithm 1 ICP Algorithm

INPUT: input $X = \{x_i\}_i$ to align, reference points $Y = \{y_j\}_j$to align to and R, t are the initial alignment.

1: Establish correspondences $C \leftarrow \left\{ \left(x, \underset{y \in Y}{\operatorname{argmin}} \left\| \mathrm{Rx_i} + \mathrm{t} - \mathrm{y} \right\| \right) \right\} | x \in X.$

2: Compute distance threshold for inliers $d_{inl} \leftarrow rQ_d(p)$ where $Q_d(p)$ is the p-quantile of distances $d_i = \|x_i - y_i\|, (x_i, y_i) \in C$ amd $r > 0$ is a multiplier.

3: Construct set of inliers $C' \leftarrow \{(x_i, y_i)|, (x_i, y_i) \in C \wedge d_i \leq d_{inl}\}$.

4: Solve absoulute orientation $\leftarrow [1]\, C'$, for the inliers.

5: Go to line 1 if not done.

OUTPUT: Alignment R,t average distance for inliers , inlier indicators $a_i = [d_i \leq d_{inl}]$.

3.1.2 Oriented FAST and rotated BRIEF 2 SLAM (ORB-SLAM 2

The ORB-SLAM 2 is an enhanced version of the ORB-SLAM which is a feature-based monocular SLAM system operating in real-time. The ORB-SLAM was developed by Raul Mur-Artal, J. M. M. Montiel and Juan D. Tardos in 2015 [19]. The method is proven to be robust and efficient localization via monocular SLAM.

The ORB-SLAM 2 is proposed by Raul Mur-Artal and Juan D. Tardos in 2017 in order to improve the first ORB-SLAM technique. The method simultaneously works on tracking, local mapping, loop closing. The technique consists of two main components, the FAST key feature detector and the BRIEF feature descriptor [20]. These components are the backbones of the technique and are explained further.

FAST is a robust corner detection implementation utilized in ORB-SLAM 2. Assume p is a pixel to be analyzed by FAST feature, then Bresenham circle radius is defined as four. Then, all pixels enclosed by the Bresenham circle are

sorted by three variations according to the estimated intensities. Variations are intensity higher than p, intensity lower than p and finally similar intensity to pixel p. The results obtained from analyzing the pixel, depends on the eight consecutive pixels, if the pixels have higher or lower intensity than p, the pixel is conceived as FAST feature.

The original is image transformed into a pyramid allowing to resolve the scale variance which is prone to errors. Each level of the image is then down sampled (zoomed) of the previous iteration. Thus the FAST features are extracted from the constructed pyramid rather than features extracted from the original image. This method annihilates the problems caused by scale variance and the computation time is decreased. The Fig. 3.2 illustrates the ORB-SLAM 2 feature detection of the close and far points from an image captured.

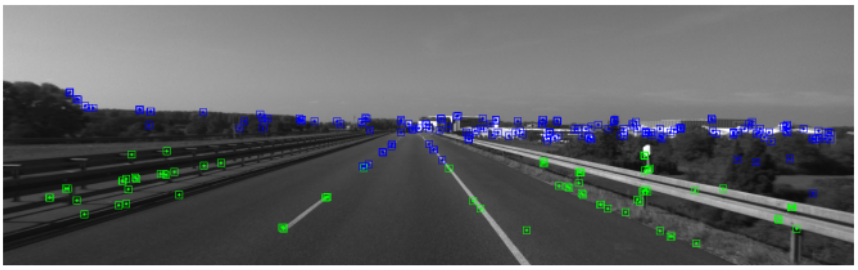


Figure 3.2: Feature detection of far points (blue) and the close points (green) [20].

The computation of the corner orientation pixel density is at the moment of $p + q$ order is described by the Eq. 3.5 below. The u and v are the pixel coordinates and $I(u, v)$ is the pixel density of the pixel analyzed. After that pixel orientation is computed with ease via the Eq. 3.6 below.

$$m_{pq} = \sum_{u=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} u^p v^q \tag{3.5}$$

$$\theta = atan2(m_{01}, m_{10}) \tag{3.6}$$

The BRIEF component utilizes the FAST feature output to transform it to a binary feature vector denoting the N-bit binary string. The binary string analysis is performed as BRIEF selects N random pixels pairs p_x and p_y in the close neighborhood of pixels. The formulation is described in Eq. 3.7 below. After that, the entire binary string is recursively created from the Eq. 3.8.

$$\tau(p_x, p_y) = \begin{cases} 1 & : I(p_x)I(p_y) \\ 0 & : I(p_x) \geq I(p_y) \end{cases} \tag{3.7}$$

$$f(n) = \sum_{i=1}^{N} 2^{i-1}\tau(p_x, p_y) \tag{3.8}$$

The optimization problem is formulated below. Assuming m is the number of points in a three-dimensional space, x_i is a point and n is the number of the camera views [21]. The camera matrix is represented by M_j therefore the Bundle adjustment is obtained from the Eq. 3.9.

$$\min_{x_i, M_j} \sum_{i=1}^{m} \sum_{j=1} \left[(\frac{m_1^j x_i}{m_3^j x_i} - u_j^j)^2 + (\frac{m_2^j x_i}{m_3^j x_i} - u_j^j)^2 \right] \tag{3.9}$$

The ORB-SLAM 2 process and steps are illustrated in the Fig. 3.3-3.4. The technique takes advantage of three parallel processes. The first process is tracking which allows finding key features matching of each frame via local Bundle adjustment. The second process constructs a local map with the local Bundle adjustment and the third process achieves global Bundle adjustment.

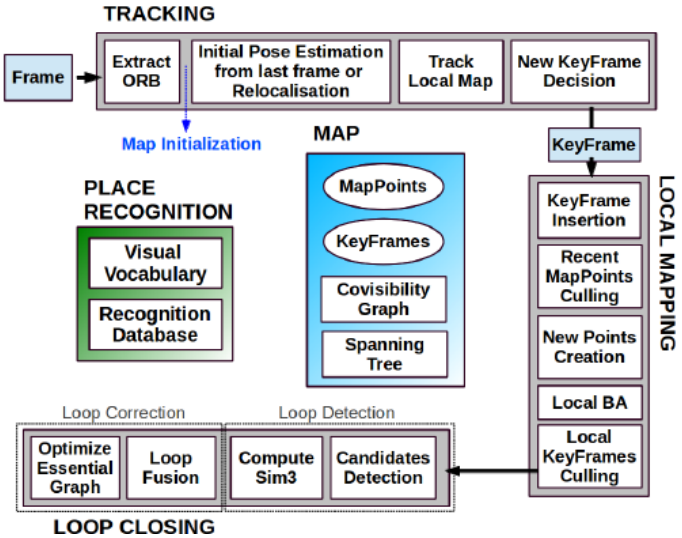


Figure 3.3: ORB-SLAM 2 architecture [20].

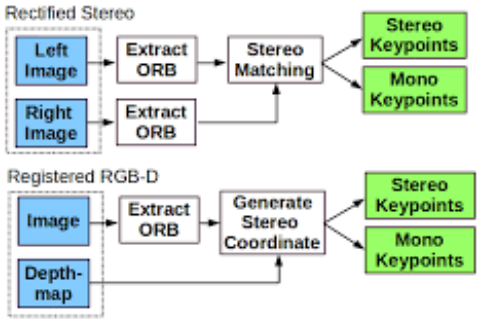


Figure 3.4: ORB-SLAM 2 architecture [20].

3.1.3 GraphSLAM

In recent years there is a variety of SLAM techniques have been developed with different approaches. One of the significant robust techniques is the GraphSLAM presented by Sebastian Thrun and Michael Montemerlo [22] in 2006. Even though there are some techniques implementing graph-like representation, GraphSLAM is an optimized and enhanced version. The technique has similarities to the literature on Lu and Milios's (1997) seminal algorithm. The GraphSLAM provides an immense scale of mapping the environment thus proving the robustness and effectiveness in the urban environment.

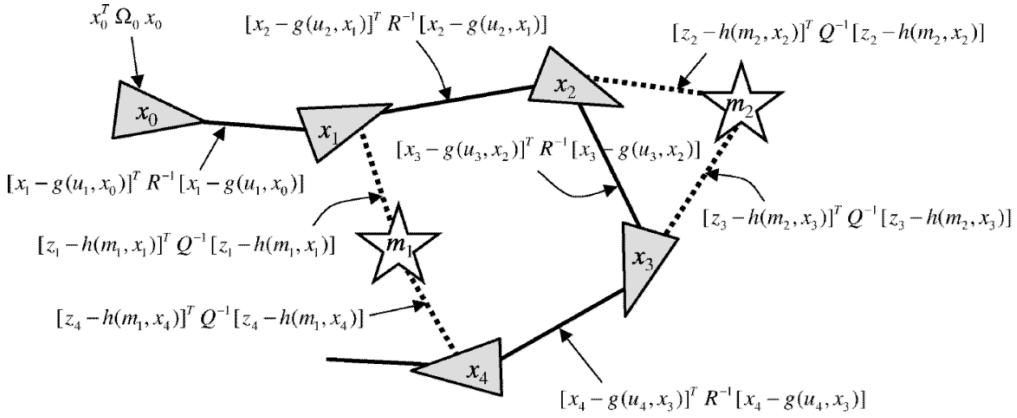


Figure 3.5: GraphSLAM illustration [22].

The Fig. 3.5 is illustrating the algorithm. The graph is extracted from the four poses of the MR denoted as $x_1, ..., x_4$ and the two map features denoted as m_1, m_2. There are two ares utilized for this approach. First is motion ares, links any consecutive poses. The second is the measurement ares which links the features measured in the particular location. The edges in the graph is representing the non-linear motion constraint. The non-linear constraints are assumed to be the negative log-likelihood of the motion model and the measurements. The constraints are then added to the graph because of the of low computational demand. The sum of the constraints results in the non-linear least-squares problem. The Eq. 3.10 for the sum of all constraints is depicted below.

$$J_{GraphSLAM} = x_0^T \Omega_0 x_0 + \sum_t \left[x_t - g(u_t, x_{t-1})\right]^T R^{-1} \left[x_t - g(u_t, x_{t-1})\right] + \sum_t \left[z_t - h(m_{c_t}, x_t)\right]^T Q^{-1} \left[z_t - h(m_{c_t}, x_t)\right] \tag{3.10}$$

The estimate of the map posterior linearizes the constraints. The result obtained is a sparse information matrix and information vector [22]. After this information, variable elimination is conducted resulting in a much small graph that consists of mobile robot poses. Finally, the path posterior map is estimated from the standard inference techniques.

3.2 Implemented Localization Method

Artificial markers are becoming significantly popular in the past couple of years. Due to the fact, there are a variety of methods and approaches it can be integrated into. In this proposed work the artificial markers are utilized for the localization of a MR. The implemented localization technique is different from SLAM techniques previously mentioned considering there is no mapping process, unlike SLAM definition. The position of the markers in the global map frame are known. The robot is expected to move around the constructed map meanwhile localizing itself effectively in the unknown environment.

State of art localization techniques consists of two integrated methods to achieve robust localization. The first method is the localization of the MR via placed fiducial markers in the unknown environment. The second technique is fusing the information from the artificial marker localization and applying an Extended Kalman Filter. The final result obtained is the estimated position of the MR in the global frame which the is the definition of the localization.

The localization technique described above is tested in simulation and

also in physical MR. The following sections of the proposed work explain implemented localization technique in-depth and the components of the localization technique are clarified.

3.2.1 AprilTag Markers

There is a variety of fiducial markers generated for integration in vision-based applications. After conducting research on the different fiducial markers, the ideal marker for the proposed work is decided as AprilTag [25] due to the robustness and agility of the visual fiducial system compared to other artificial markers.

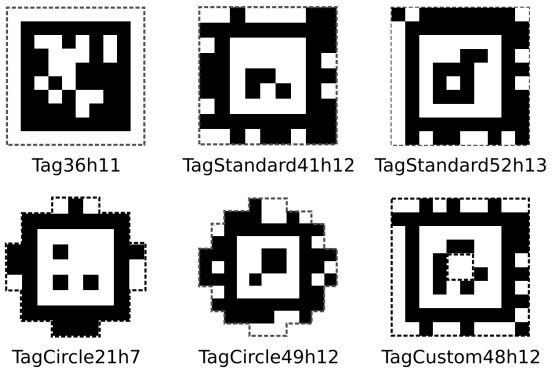


Figure 3.6: AprilTag generated tag family [25].

The AprilTag is an advanced visual fiducial system developed by E.Olson [23] in 2011 which is still supported and updated. The major advantages of the AprilTag to its opponents are the vast quantities of different pre-generated tags, the ability to detect more than one tag in the same frame, preciseness of square planers orientation detection, high-speed computation and outstanding compatibility with ROS.

The AprilTag detection system can be summarized in two steps, the first step is the detection of the marker which involves estimating the relative position of the marker with respect to the camera frame, second is extracting information from the payload.

Fig. 3.8 illustrates the detection scheme in stages. Fig. 3.8(a) is the input image observed from the camera. The process begins with detection of line segments in the image. The gradient direction and magnitude are computed at every pixel referring to Fig. 3.8(b)-(c). Further on, a graph based method is integrated to find similar gradient magnitude and direction for clustering components as shown Fig. 3.8(d). The second stage is finding AprilTag square and line segments [24]. The weighted least squares are implemented

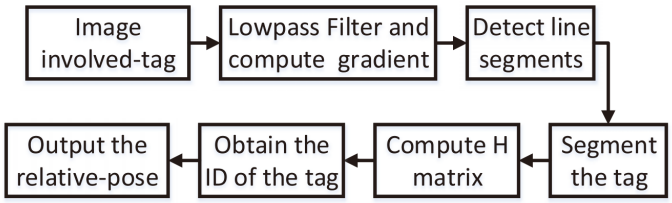


Figure 3.7: AprilTag process flowchart [24].

in Fig. 3.8(e) to fit line segments in the pixels of each component formed in the previous step. In the final Fig. 3.8(f), the intersection of lines allows obtaining pixel's coordinates which are at the center of every minor quad in the AprilTag marker. The process described is already implemented in the AprilTag library and has compatibility with ROS.

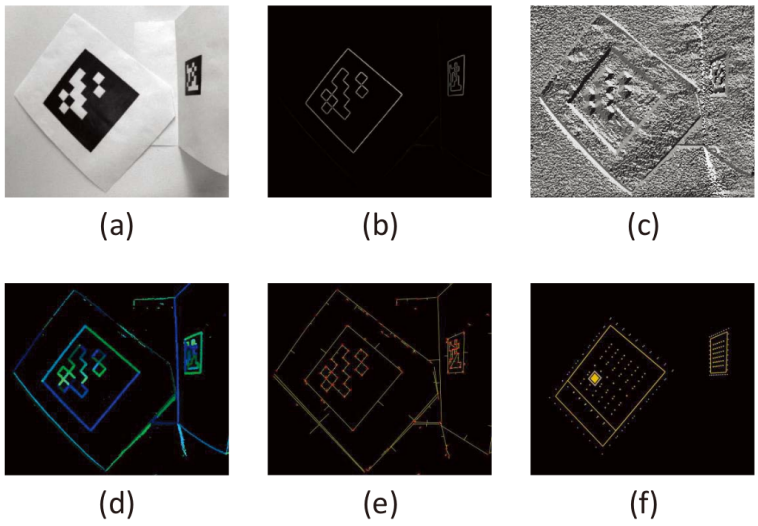


Figure 3.8: AprilTag image processing steps [24].

The AprilTag is an easy-to-use agile library that conducts the image processing and the resulting information is then published to ROS topics. The resulting information includes tag ID, position in 6-DOF (Degrees of Freedom), detected image, etc. In order to use the library some configurations are required to be set such as size of the tags ,number of tags used, specify the family of tag utilized (Tag36h11 is selected), assign camera intrinsic parameters, and the subscribe to ROS topic of camera image output. After acquiring the configurations of the library the entire computation is handled by

the "*apriltag_continuous_localization_node*" then the results are published to preset topics. The illustration below is depicting detected markers from the camera image.

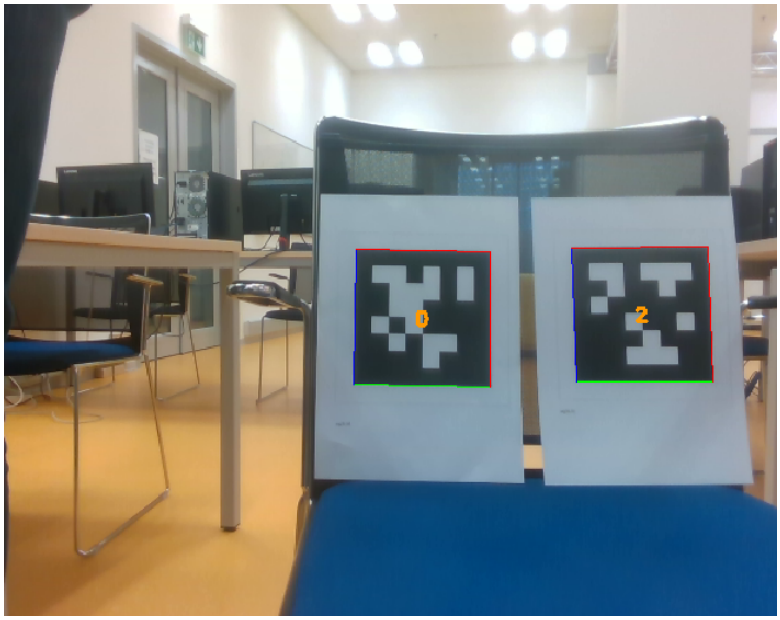


Figure 3.9: AprilTag detection via Intel RealSense D435 camera.

3.2.2 Experimental Setup

The experiments of the implemented Apriltag localization method are tested in both simulation and real environments. The setup for the experiments is explained in the following subsections.

Simulation Setup

The implemented localization technique is tested in the simulation environment provided by Gazebo Software. The exact mobile robot model is used as mentioned in the Section 2.3 however the map is modified by placing the AprilTag markers on the wall and also a smaller map (size 2m by 2m) was generated for the purpose of improving the accuracy of the AprilTag detection system.

The MR is placed inside the unknown map, it's expected to move around the map while achieving exceptionally accurate localization results. The maps used are depicted in Fig. 3.10-3.11.

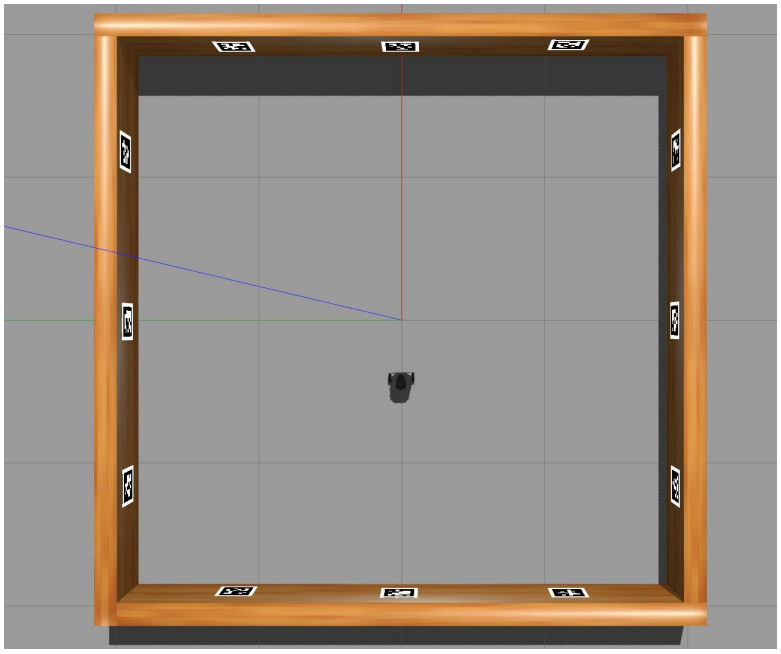


Figure 3.10: Visualization of small map utilized for localization testing.

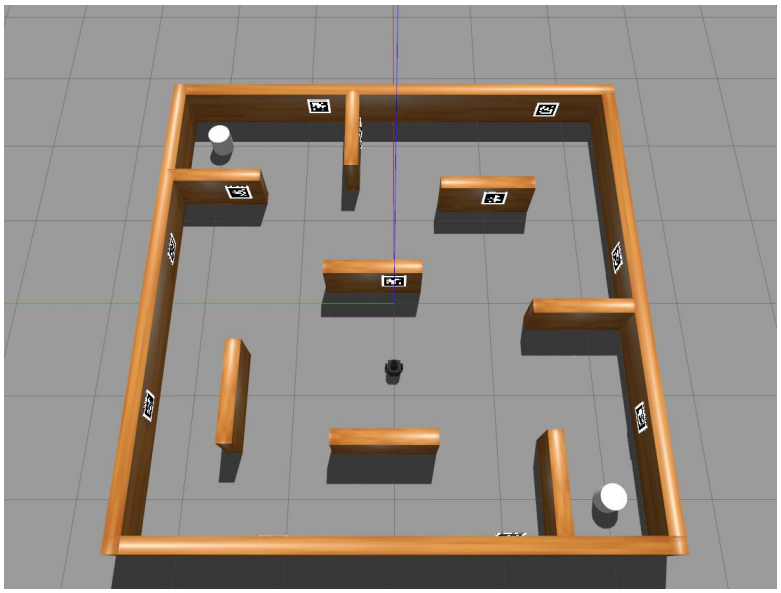


Figure 3.11: Visualization of larger map utilized for localization testing.

Real Mobile Robot Setup and Components

Real testing involves many components and devices to conduct the experiments. In the following subsections, the different types of key components of the experiment are explained in depth.

Turtlebot

The real mobile robot used in the experiment is TurtleBot 2. Which is a popular robot model for education and research due to, open-source software, low cost and compatibility with ROS. The Turlebot was manufactured at Willow Garage and designed by Melonee Wise and Tully Foote in 2010. The robot is a cylindrical shape similar to robotic vacuum cleaner models on the market. The size of the model is 354 mm x 354 mm x 420 mm (Length x Width x Height) , allowing the ideal size for navigation in small environments. The maximum transnational velocity generated by TurtleBot 2 is 0.65 m/s and the maximum rotational velocity is 3.14 rad/s. The maximum payload weight it is able to carry is 5 kg on hard surfaces 4 kg on soft surfaces such as carpet. Fig. 3.12 illustrates the factory production dimensions of TurtleBot 2.

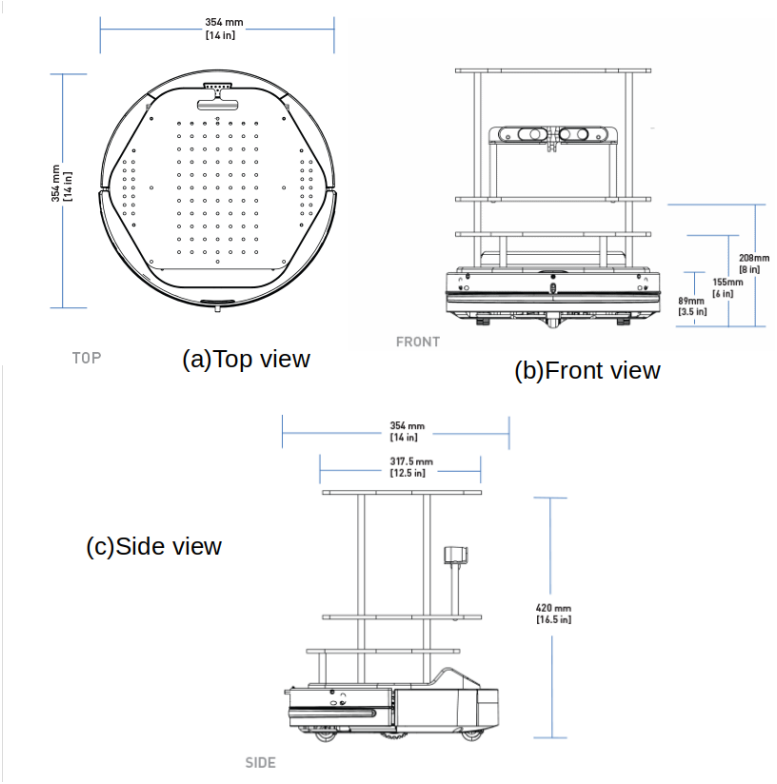


Figure 3.12: TurtleBot 2 Factory Dimensions [29].

Intel NUC

Intel NUC mini PC is integrated onto the TurtleBot 2. The processing and all of the computation is depending on the Intel NUC. It is the backbone of the robot setup. The compact size and high-performance components of the device resulting in the one of the best options for CPU (Central Processing Unit) in MR. The ROS is installed on the Intel NUC which handles the ROS threading and communication between the nodes. The Table below presents the specifications of the Intel NUC model used.

Processor	Intel(R) Core(TM) i5-7260U CPU 2.20GHz
RAM	8 GB
GPU	Intel Iris Plus Graphics 640 (Kaby Lake GT3e)
Storage	100 GB
ROS Version	Kinetic
Ubuntu Version	16.04 LTS

Table 3.1: Intel NUC specifications.

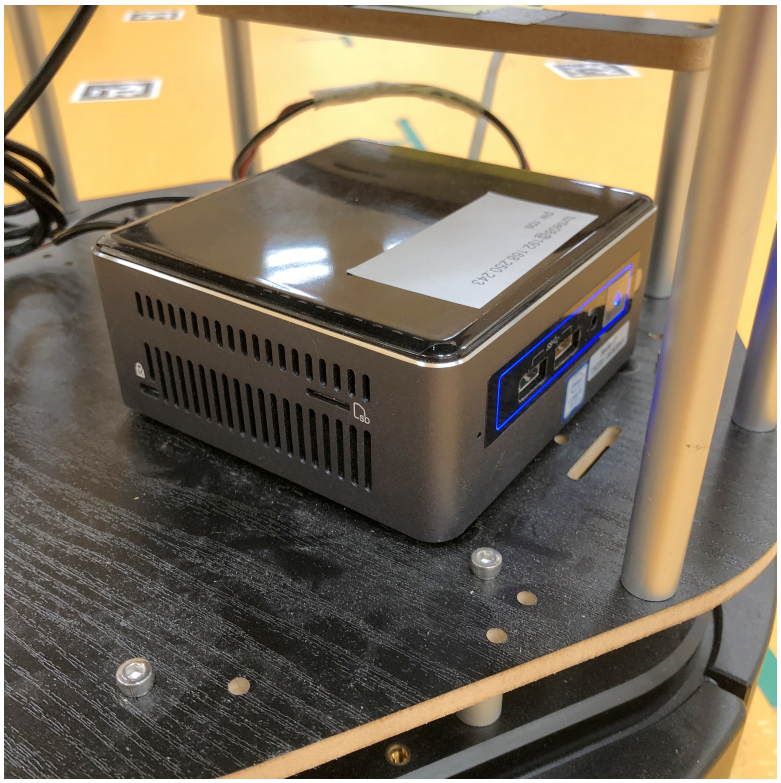


Figure 3.13: Intel NUC integrated to TurtleBot.

Intel RealSense-D435 Camera

Initially the Intel Realsense T265 [2] was the decided camera for the experimental setup. Although by taking the advice of my Supervisor it is changed to RealSense D435 [2] which provided the ideal results for the experiment.

Intel RealSense D435 is high technology stereo depth camera with a built-in IMU. The camera is integrated with a color image of output 640x360 pixels on the other hand the depth image output is 848x480 pixels. The depth image is visualized as a depth map, meaning the pixels represents the distance instead of color and intensity thus it is significant convenience to utilize the camera for obstacle avoidance techniques instead of using lidar. The software is easy to use and the compatibility with ROS is outstanding.

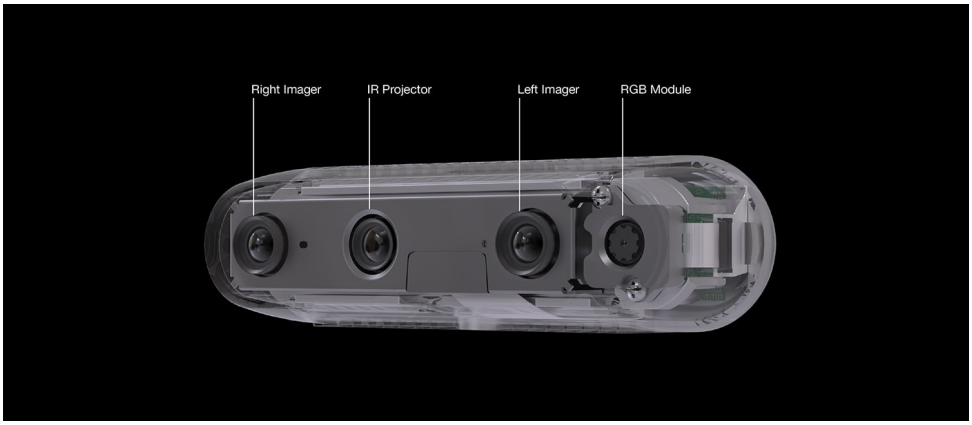


Figure 3.14: Intel RealSense D435 camera [2].

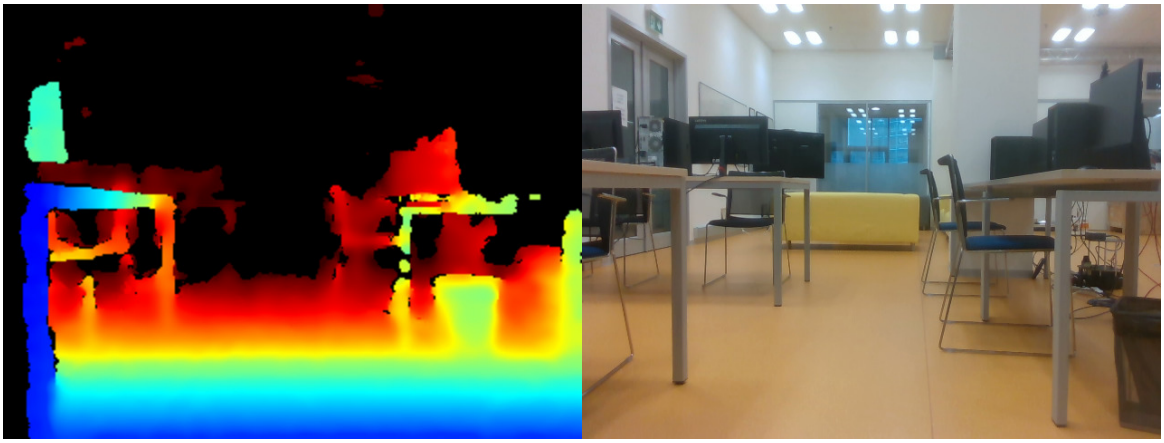


Figure 3.15: Depth Image (left) and Color Image (right) captured via Intel RealSense D435.

VICON Motion Capture System

The Vicon Motion Capture System [28] is a high-end motion tracking system specially developed for robot, gaming and film sectors. The objective of the system is to track and record any desired component inside the detection area of the Vicon System. The synchronized special light-emitting cameras are placed throughout the Vicon System allowing 360-degree coverage of the environment, conducting detection of reflective Vicon Markers which can be attached to any surface. After detection of the reflective markers, the system is able to estimate the global position and orientation of each reflective marker inside the Vicon System. The tracking is extremely precise thus system can be used as a comparison of almost perfect localization (error $<$ 1cm). With the help of a ROS node, the communication between Vicon Motion Capture System and ROS Master is activated allowing to receive the estimated position and orientation of markers obtained from Vicon Motion Capture System. The TurtleBot contains four reflective markers allowing for full range

detection and orientation of the MR. The reflective markers are also utilized to estimate the position and orientation of the artificial markers placed on the wall. Fig. 3.16 below is depicting on the left side is small reflective markers that can be placed on any surface in order to track the desired objects, and on the right side special camera of Vicon is present.

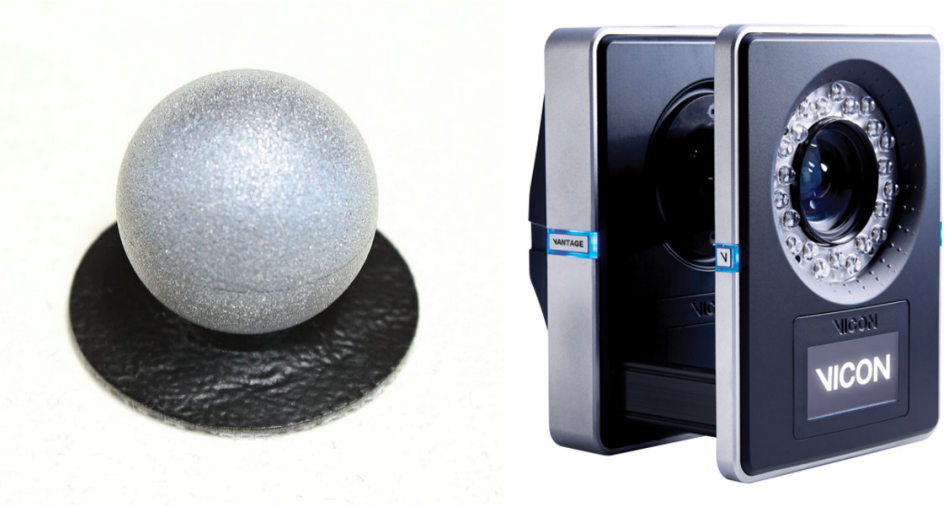


Figure 3.16: Reflective marker (left) and Vicon tracking camera (right).

Fig. 3.17 below illustrates the area which is equipped with the Vicon Motion Capture System. The mobile robot's motion is tracked by the cameras attached to the metal frame on the ceiling.

Figure 3.17: Area equipped with Vicon Motion Capture System.

3.2.3 Apriltag Localization

The Apriltag Lozalization is the implemented technique to achieve precise MR localization. The implemented technique requires the AptilTag library to be installed in order to detect and identify the fiducial tags. Subsequently, this technique takes advantage of the ROS libraries and tag detection system to transform the information and to estimate the position and orientation of the robot in the global coordinate frame. The technique is described in-depth in the following paragraphs.

Before Apriltag Localization technique begins, the artificial markers are placed around the testing environment. The position and orientation of the markers need to be measured with respect to the global coordinate frame. After placing the AprilTags on the wall, the positions in three-dimensional space (x,y and z axis) and the rotation are measured in roll, pitch and yaw notations. This configuration is then imported to a YAML file. The YAML file is utilized by the Apriltag Localization technique to reach the global parameters of the markers used in the experiment. Then to broadcast the information stored in the YAML file, a script is created "tag_broadcaster.py" which reads the YAML file data and statically broadcasts the "map" and "tag_ID" transformation according to the global position of the tags included in the YAML file data. Subsequently, by executing the command "rqt_tf_tree" this leads to Fig. 3.18 below. The obtained transformation tree is an independent component at first although later on will be connected to the general transformation tree. The rest of the method relies on this transformation tree to conduct a process to compute an estimated position of the MR in the environment.

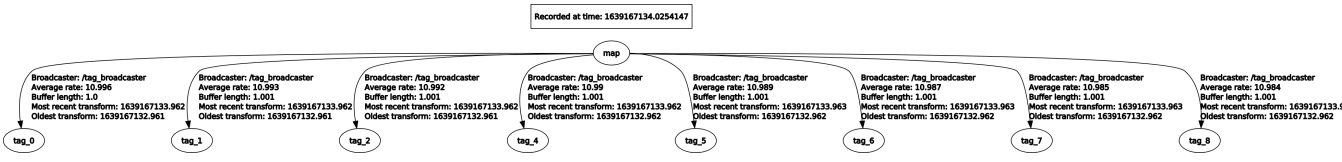


Figure 3.18: Transformation tree visualization after static broadcasting of artificial markers.

The rest of the estimation processes in the method are handled by the python script "robot_pose_broadcaster.py". At the beginning of the script is the initialization of the fundamental parameters necessary for pose estimation such as different coordinate frames. Subsequently, the transformation between the frames is observed and tracking of transformation is achieved by the "tf" ROS package. The package provides the relationship between the coordinate frames in a tree structure buffered in time and allows the user to transform points, vectors, etc. between two coordinate frames for any point at any time. The package delivers massive convenience due to geometry

computations that are implemented in the package therefore the user doesn't have to compute matrix transformations. Nevertheless, comprehending the concept of transformation between the frames is crucial because if the desired transformation between frames is not clearly understood then the results might be trivial. The concept behind the transformations is explained further on.

The homogeneous coordinates are utilized in order to compute the transformation matrix. Any point P in the three-dimensional space R_3 can explicitly be described in the homogeneous coordinate notation, the formulation below is the mathematical representation. The first three rows are x, y and z axis coordinates of the point and the last row is denoted as the scale factor which is 1 in the proposed work.

$$P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.11}$$

Consecutively the rotation matrix (R) and translation matrix (t) are the key components of the transformation matrix. The rotation and the translation matrix are described below in the general form. By using the two matrix forms the transformed coordinates of any rotated point in the three-dimensional space can be obtained. The general rotation matrix is formulated in the quaternion (q) representation in Eq. 3.13 .

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, t = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.12}$$

$$R(q) = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(zx + wy) \\ 2(xy + wz) & 1 - 2(z^2 + x^2) & 2(zy - wx) \\ 2(zx + wy) & 2(zy + wx) & 1 - 2(y^2 + x^2) \end{bmatrix} \tag{3.13}$$

$$q = x_i + y_j + z_k + w \tag{3.14}$$

While the x, y and z variables are the imaginary components of the quaternion and the w is the real component of the quaternion representation in Eq. 3.14 . After setting all the notations above one can obtain the transformation matrix T as illustrated below.

$$T_{4x4} = \begin{bmatrix} R(q)_{3x3} & t_{3x1} \\ 0^T & 1 \end{bmatrix} \tag{3.15}$$

Once the rotation and translation matrix are configured between frames then with the Eq. 3.16 below it's possible to find the transformed vector ($P^{'}$) coordinates in three-dimensional space as :

$$P^{'} = TP \tag{3.16}$$

The concepts and notations introduced are utilized in the Apriltag Localization method. The localization is initiated soon as the detections from the AprilTags are received. First, the ID of the tag is stored and the relative position of the tag with respect to the camera is stored. The transformation between the camera frame and the tag frame of the robot is estimated by the AprilTag detection system. After that series of operations are conducted to describe the MR pose in the global coordinate frame.

The transformation between the base of the robot and the camera frame is used to transform the position of the marker in the base frame of the robot by obtaining the transformation matrix T and by using the Eq. 3.16 the transformed marker pose is estimated with respect to the robot base frame. The transformation matrix between the frames is retrieved from the "lookup_transform" function.

The next transformation is the base frame with respect to the marker frame. Which is the inverse of the last achieved transformation. The inverse of the transformation matrix is the desired parameter.

There on, the transformation matrix of the robot base with respect to the map frame is constructed at the beginning of the section by broadcasting the ground truth positions of the markers. Thus the connection of the transformation tree is satisfied. The result gives the coordinates of the robot base in the map frame. Although the results are not in desired number format it's stored in an array. The same process applies to every detected marker. Nevertheless, the localization process is ongoing in the further steps.

The array is filled with tag poses with respect to the map frame. The mean average of the x and y axis coordinates of the poses are calculated for an estimated position since the MR is placed on the ground, the z-axis is assumed to be zero. The orientation of the base frame is calculated from the first detected artificial marker. Finally, the stored poses array is reduced to a pose to be transformed last time and the transformation leads to highly accurate estimation of the robot pose in the global frame. The standard deviation of the errors is computed as: 0.0959 cm for x-axis, 0.0524 for y-axis and 0.5388 for the yaw angle in the simulation system. For testing, the MR executes a circular trajectory the results of the Apriltag Localization method are represented by the Fig. 3.19-3.24 .

Figure 3.19: The comparison of ground truth (blue line) and Apriltag Localization method (red line) for the x-axis in simulation.

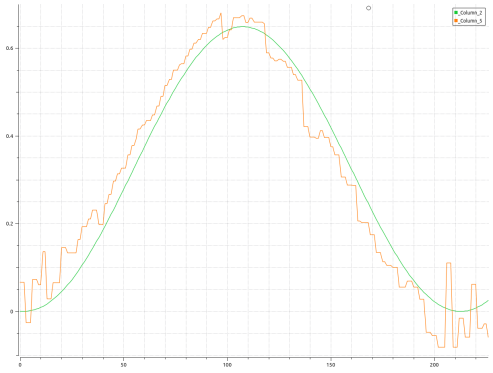


Figure 3.20: The comparison of ground truth (green line) and Apriltag Localization method (orange line) for the y-axis in simulation.

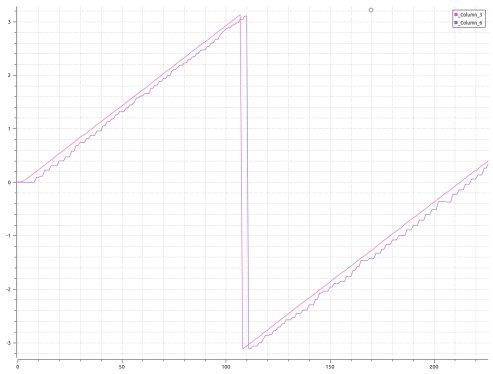


Figure 3.21: The comparison of ground truth (pink line) and Apriltag Localization method (purple line) for yaw in simulation.

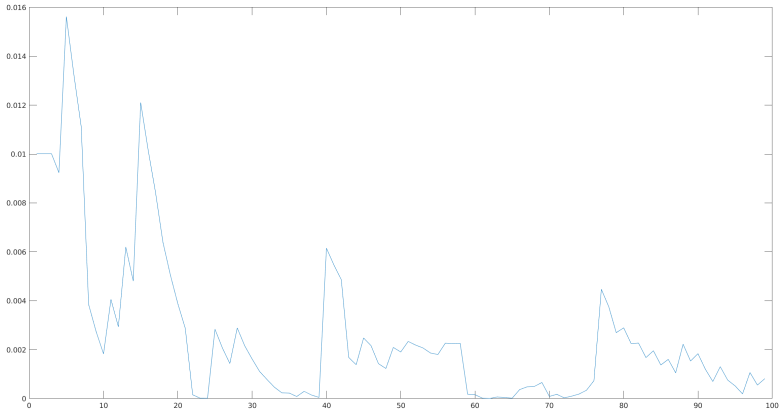


Figure 3.22: Root Mean Square Error (RMSE) computed for the x-axis in simulation.

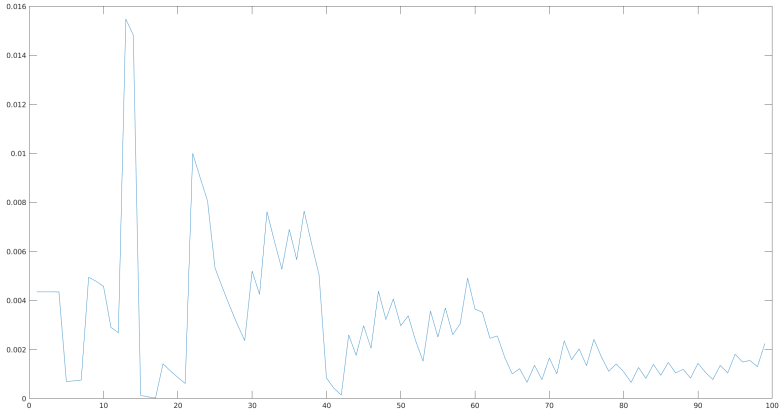


Figure 3.23: Root Mean Square Error (RMSE) computed for the y-axis in simulation.

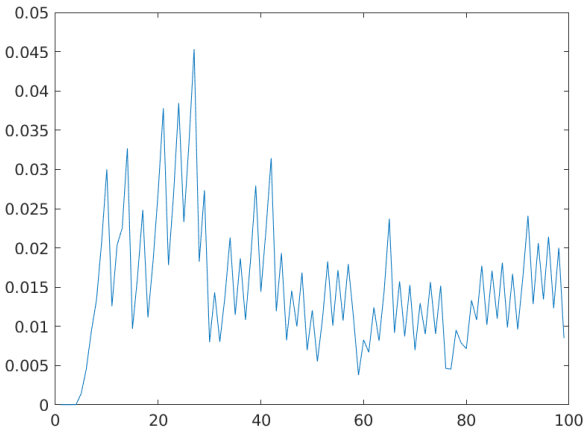


Figure 3.24: Root Mean Square Error (RMSE) computed for yaw in simulation.

The process after this is not available for the real robot but it's a visualization modification via the Rviz software in the simulation. Meanwhile, the processes described are ongoing the transformation of the map to odometry frame is being broadcasted and updated each loop. Thus the robot pose and orientation can be viewed in the map frame. The visualization shows the robot pose is fluctuating and unstable therefore the localization via Apriltag needs to be improved even though the estimated pose is highly accurate.

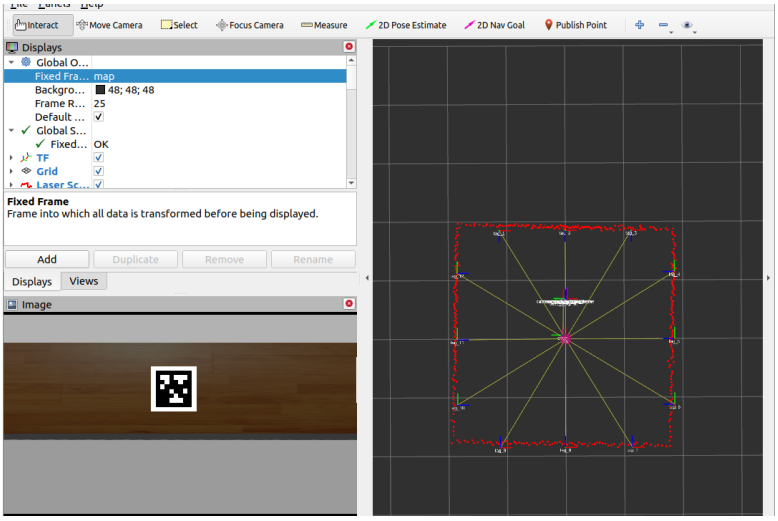


Figure 3.25: Visualization of estimated robot pose via Rviz software tool.

3.2.4 EKF Fusion

The previously implemented localization technique standalone can't provide a precise localization. The irregularities are eliminated by applying an Extended Kalman Filter (EKF). The previous technique's output is one of the inputs for the EKF. The EKF technique is one most popular filtering for pose estimation which is the nonlinear version of the Kalman Filtering that linearizes an estimate of current mean and covariance. Even though EKF is not guaranteed to achieve optimal results. The MR motion in the experiment is nonlinear and the Kalman Filter is not sufficient. Thus the EKF is the ideal selection of filtering for this proposed work.

It is assumed that the x and y axis are the global coordinate system for the filtering. The TurtleBot 2 consists of two independent motorized differentials wheels and two dummy wheels for stability. The motors work independently thus the kinematic model equation of the robot can be estimated as:

$$v_{mr} = v_1 + v_2 = r(\frac{v_r + v_l}{2}) \tag{3.17}$$

$$w_{mr} = w_1 + w_2 = r(\frac{v_r - v_l}{d}) \tag{3.18}$$

The v_{mr} is denoted as linear velocity , v_r and v_l are the right and left wheel speed, r is the wheel radius, d is the distance between two wheels. and finally w_{mr} is the angular velocity of the robot. Then the current state of the MR can be calculated as :

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta s_t \cos(\theta_{t-1} + \Delta\theta_t) \\ \Delta s_t \sin(\theta_{t-1} + \Delta\theta_t) \\ \Delta\theta_t \end{bmatrix} \tag{3.19}$$

The Δs_t is the difference in translational distance and $\Delta\theta_t$ is the angular difference.

The system and measurement equations for discrete-time Extended Kalman Filter as follows [26]:

$$x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_k) \tag{3.20}$$

$$y_k = h_k(x_k, u_k, v_k) \tag{3.21}$$

The f_{k-1} and h_k are non linear functions of state transition vectors. The w_k and v_k are the process noise and measurement noise the values are considered to be zero mean Gaussian. The EKF consists of two stages. The first stage is the prediction of the estimated state and the second stage is updating the estimated state according to the measurement observed. The stages are explained further on.

Prediction Stage

The stage is activated soon as the robot motion is executed. The callback functions are implemented in the script which directs the input for the f_{k-1} function. The current state prediction is computed from the dynamic constraints of the robot via Eq. 3.22. Then the state covariance matrix is estimated as in Eq. 3.23.

$$\widehat{x}_k = f_{k-1}(x_{k-1}, u_{k-1}, w_k) \tag{3.22}$$

$$P_k = F_{k-1} P_{k-1} F_{k-1}^T + L_{k-1} Q_{k-1} L_{k-1}^T \tag{3.23}$$

The parameters referred as L_{k-1} is the Jacobi-an Matrix of the function f_{k-1} which is differentiated with the respect to the input variable, Q_{k-1} is the input covariance matrix.

Update Stage

The update stage takes advantage of the predicted state estimation in the previous stage then utilizes the information each time the measurements are conducted. Thus in order to apply the update stage, measurements are necessary. The stage initializes with computing the update coefficient which is obtained from the Eq. 3.24 Where z_k is the observation state acquired from the measurements.

$$e_k = z_k - h_k(x_k, u_k, v_k) \tag{3.24}$$

Then the the kalman gain K_k is computed below. The H_k is Jacobi-an Matrix of function h_k which is differentiated with the respect to the state variables.

$$K_k = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1} \tag{3.25}$$

After the calculations, the new state equations are updated in Eq. 3.26 thus the covariance matrix is also updated as Eq. 3.27 thus concluding the EKF iteration.

$$\widehat{x}_k = x_k + e_k K_k \tag{3.26}$$

$$P_k = (I - K_k H_k) P_{k-1} \tag{3.27}$$

3.2.5 Results

The results are obtained according to the experimental setup described in the Section 3.2.2. The robot is moving in a circular trajectory while the Apriltag Localization method is estimating the pose of MR, the EKF is using that information for the final accurate pose computation. The graphs below are illustrating the experiment scheme. The robot ground truth pose is compared with EKF and Apriltag Localization method results. The results are shown in two categories: simulation environment and real environment. In the real environment tests, the ground truth position is provided by the Vicon motion capture system.

Simulation Results

The Gazebo simulation's trajectory results are depicted below in Fig. 3.9. The blue line indicates the Apriltag Localization method, the green line is for the EKF result and the red line which is overlapping with the green line is ground truth. The Apriltag Localization method achieves very high accuracy

in a stable position meaning no force should be exerted from the motors. The method solely relies on the image provided by the camera. The motion of the MR causes some complications such as the screen capture time is significantly dropped while moving thus the image is less clear for marker detection leading to inconsistent estimate positions. The EKF fusion comes in handy to resolve the data inconsistencies. The estimated position computed from dynamics of MR and Apriltag Localization method results are fed to the EKF concluding with an accurate estimate of positions compared to the ground truth.

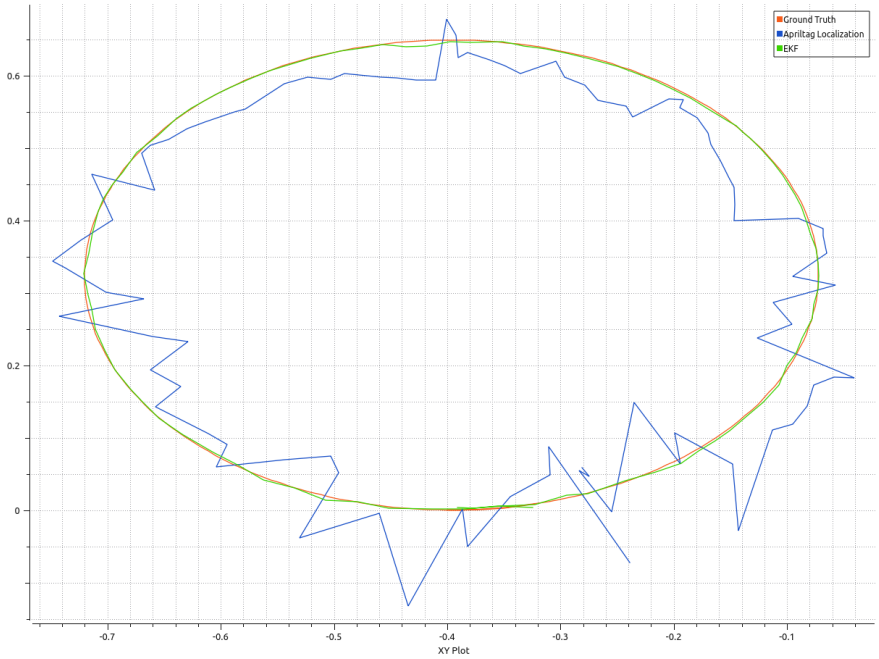


Figure 3.26: The two-dimensional trajectory (x,y) results from the Gazebo simulation.

Real Mobile Robot Results

The tests for the mobile robot are conducted in two different conditions: constant small circular trajectory in the small environment (Fig. 3.27) and larger trajectory in the larger environment (Fig. 3.28). The small environment consists of nine AprilTags (14 cm Tag36h11) placed around the environment for precise AprilTag detection and for smoother EKF results, the markers are detected continuously meaning there isn't a camera frame without an AprilTag marker detected thus AprilTag localization is estimated constantly within the small environment. The large environment consists of six AprilTags (9 cm Tag41h12) and they are placed further from each other, the robot is moved around arbitrary in the area by the user controlling the robot motion, the AprilTags are detected discretely meaning the EKF is computed constantly when the marker is detected then EKF measurement is updated according to the measurement observed. In the following paragraphs, the experiments will

be discussed and the results are displayed in the Fig. 3.29-3.30 .

Figure 3.27: The small testing area with the Apriltags placed (approximately $4m^2$).

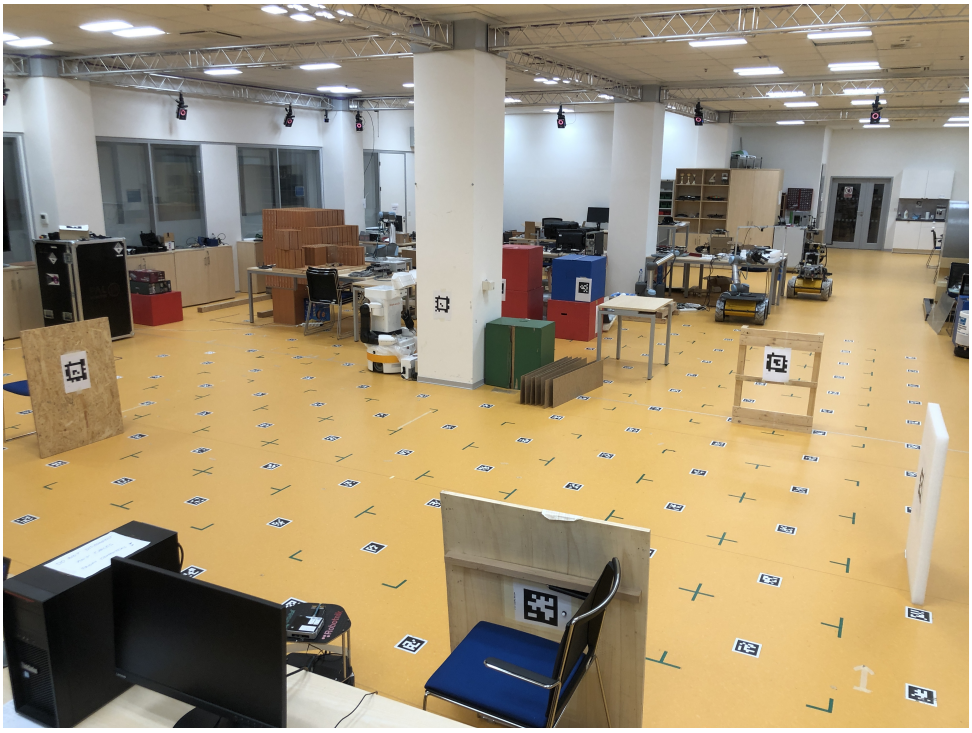


Figure 3.28: The large testing area with the Apriltags placed (approximately $30m^2$).

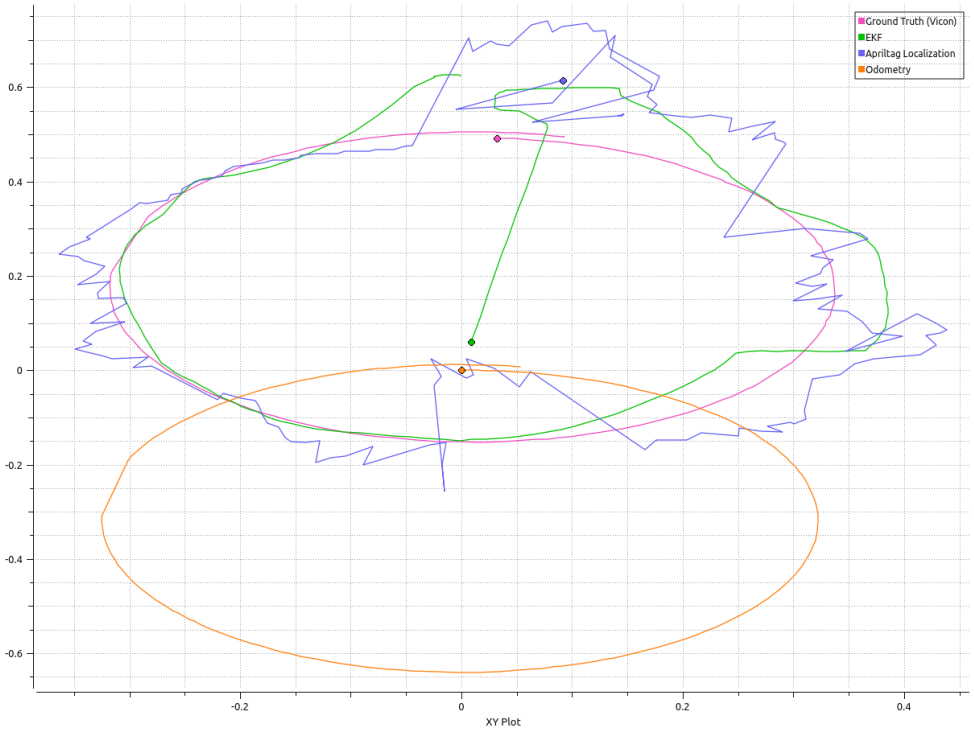


Figure 3.29: The two-dimensional trajectory (x and y axis) results from the small environment test.

The Fig. 3.29 depicts the results obtained in the small environment, the pink line refers to ground truth provided by the Vicon System, the green line is EKF results, the orange line is the odometry calculated by the TurtleBot, the blue line is the Apriltag Localization results which is extremely noisy due to the vibrations in the camera frame caused by the bumps on the surface or sudden acceleration and declaration lead to blur in the camera output image. Thus the detection is not conducted in the optimal conditions resulting in fluctuations in the measurement data. Although this factor was not apparent in the simulation. Nevertheless, Apriltag Localization data is then smoothed via EKF, the results of the EKF are the combination of measurements of AprilTag and the robot pose is estimated from the motion model kinematic equations. The EKF results are less accurate compared to simulation results due to the Apriltag Localization result's inconsistencies. Nevertheless, the EKF is filtering the results obtained from the Apriltag Localization method for precise estimation of the position.

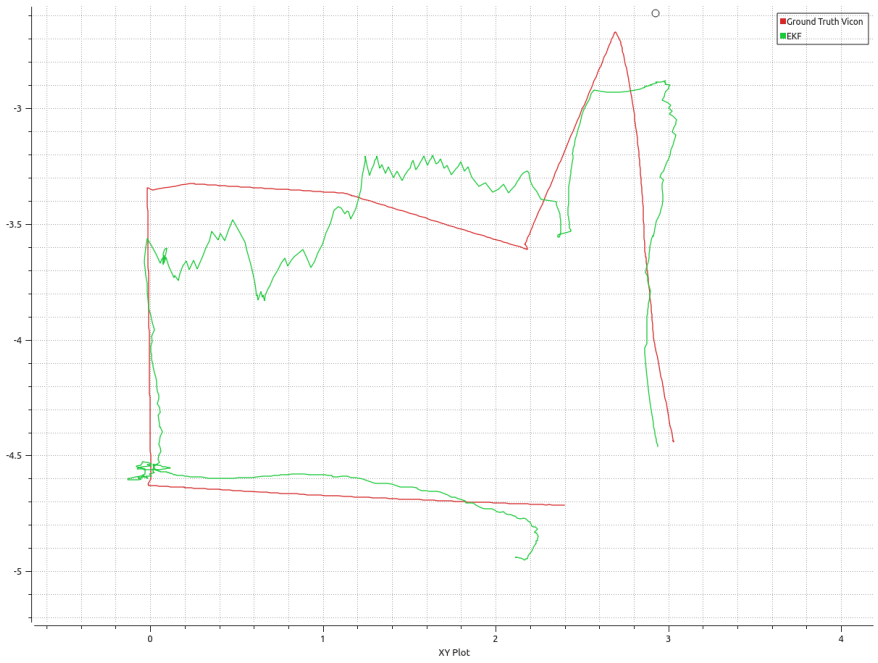


Figure 3.30: The two-dimensional trajectory (x and y axis) results are obtained from the larger environment test.

Fig. 3.30 above, illustrates the results acquired in the larger environment testing and larger trajectory. The robot is steered by the user in an arbitrary trajectory. The red line refers to the ground truth computed from Vicon System, while the green line refers to the EKF result. The noise of the EKF is increased compared to the previous small environment tests due to the reason, the markers are placed at a larger distance away from each other thus the effective working range of the AprilTag markers is exceeded which is approximately 1.8 meters in the experimental setup described earlier. The EKF results are depending on the results obtained from the Apriltag Localization method and the results are fluctuating more than the previous tests causing the EKF results to deteriorate as well. Fig. 3.31 displays the gold lines that refer to Apriltag Localization results conducted in the larger environment. Nevertheless, the EKF is able to smooth the inconvenient results provided by the Apriltag Localization method.

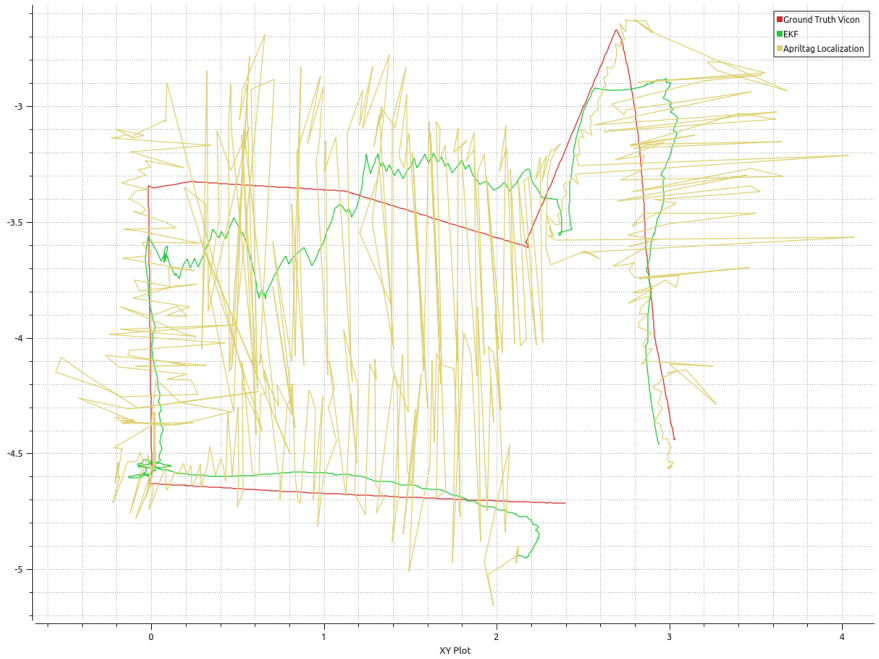


Figure 3.31: The two-dimensional trajectory (x and y axis) results are obtained from the large environment test.

The real environment tests have a tendency for noise, lighting in the environment is a indeed major factor. Even though Intel Realsense D435 is an outstanding device, that has adequate resolution optimal for this task but slight movements in the camera frame lead to inconsistent EKF results. Enlarging the AprilTag markers might improve the accuracy or decreasing the distance to the marker can enhance the overall detection accuracy.

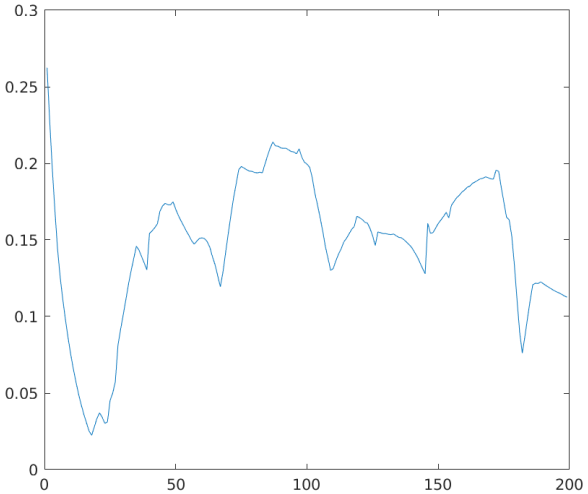


Figure 3.32: The graph of the total root mean square error of the x and y axis is acquired from the small environment test.

The general accuracy is determined by the average error of the axes and the overall root mean square error for x and y axis. The average error is computed as 7.22 cm for x-axis and 3.47 cm for y-axis in the total average of 5.34 cm in the small environment. The larger environment error is estimated as 16.38 cm for the x-axis and 9.82 cm for the y-axis leading to a 13.1 cm total average error. The Fig. 3.32 is retrieved from the small environment representing the overall root mean square error (RMSE) which is the sum of root mean square error of x and y axis divided by two. The error significantly increased when the larger environment test is conducted the reason for it is the increased distance of the artificial markers in the environment resulting in inaccurate position readings from the visual AprilTag markers.

Chapter 4

Conclusion and Future Work

The main objective of the thesis is to implement a localization method for a mobile robot using artificial markers. To supplement the localization part, studies of obstacles avoidance are initially presented and two methods are implemented and tested in the simulation environment. The results of the DWA and Bug-0 algorithms are illustrated in Tables 2.1-2.3. Interestingly the results propose that the Bug-0 algorithm is more reliable and efficient even though it is developed to be simple and computationally efficient. The DWA algorithm on the other hand relies on the motion kinematic of the mobile robot thus expected to be more precise due to extensive calculation compared to Bug-0 algorithms. DWA also demands to adjust plenty of parameters. Nevertheless, the results could have been alternated if a different map is utilized.

In the second part, state of art localization and SLAM techniques are described and a method utilizing artificial marker is implemented. The general localization system consists of two processes. The first being Apriltag Localization which solely relies on image detection. Eventually, this method needed improvement in accuracy thus the localization result is fused with the second process, EKF, to achieve high-level precision. The system is tested in both simulation and with a real robot (TurtleBot-2). The results are presented in Fig. 3.29-3.32 and the average accuracy is determined. The final EKF results differed from the simulation results because the Apriltag Localization method was affected by the blur in the image due to vibrations caused by the motion; therefore, the real environment testing is noisier and has more unpredictable factors. Nevertheless, the mobile robot localization via artificial markers is established and improved, final results prove the system to be highly correlating with the thesis goal.

For future work, the obstacle avoidance algorithms developed could be implemented in a real robot that relies on the results of the localization system developed. Since algorithms are tested in the simulation environment with few alternations of parameters, the script can be easily adapted to the TurtleBot 2. For the future, it is a good idea to integrate localization method and obstacle avoidance techniques.

Bibliography

[1] Robotic Operating official website (ROS).
https://wiki.ros.org/ROS/Introduction

[2] Intel RealSense official web-site.
https://www.intelrealsense.com

[3] Daneshmand, Morteza. (2013). Path Planning and Obstacle Avoidance of Mobile Robots via Potential Field Based Fuzzy Controlling.

[4] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” International Journal of Robotics Research, vol. 5, no. 1, pp. 90–98,1995.

[5] Maria Isabel Ribeiro,1 Navigation/Collision Avoidance 1:1 Introduction Obstacle Avoidance, 2005.

[6] Oroko J. A., Nyakoe G. N. Obstacle avoidance and path planning schemes for autonomous navigation of a mobile robot: a review, in Proc. 2012 Mechanical Engineering Conference on Sustainable Research and Innovation, vol. 4, 2012, pp. 314-318.

[7] J. Borenstein and Y. Koren, “The vector field histogram - fast obstacle avoidance for mobile robots,” IEEE Transaction on Robotics and Automation, vol. 7, no. 3, pp. 278 – 288, 1991.

[8] Zohaib, M., Pasha, Mustafa, Riaz, Raja Ali, Javaid, Nadeem, Ilahi, M.,Khan, Rahim. (2013). Control Strategies for Mobile Robot With Obstacle Avoidance. 3. 1027-1036.

[9] J. Minguez and L. Montano, “Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios,” IEEE Transactions on Robotics and Automation, vol. 20, no. 1, pp. 45–59, 200.

[10] J. W. Durham and F. Bullo, "Smooth Nearness-Diagram Navigation," 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 690-695, doi: 10.1109/IROS.2008.4651071.

[11] D. Fox, W. Burgard and S. Thrun, The Dynamic Window Approach to Collision Avoidance.=, Robotics and Automation Magazine, IEEE, 1997.

[12] Choset H, Lynch KM and Hutchinson S. Principles of Robot Motion: Theory, Algorithms, and Implementations. ABD: M.I.T. Press, 2005.

[13] Lumelsky VJ and Stepanov A. Dynamic path planning for a mobile automaton with limited information on the environment. IEEE Transaction Automatic Control 1986; 31: 1058-1063.

[14] Jom Kandathil, Robins Mathew, and SomashekharHiremath. "Modified bug-1 algorithm based strategy for obstacle avoidance in multi robot system".MATEC Web of Conferences144 (Jan. 2018).

[15] Webster, J.G., Huang, S. and Dissanayake, G. (2016). Robot Localization: An Introduction. In Wiley Encyclopedia of Electrical and Electronics Engineering, J.G. Webster (Ed.). https://doi.org/10.1002/047134608X.W8318

[16] P. Besl and H. McKay, "A method for registration of 3-d shapes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, 1992, pp. 239 –256.

[17] Tiar, R. and Lakrouf, Mustapha and Azouaoui, O.. (2015). FAST ICP-SLAM for a bi-steerable mobile robot in large environments. 1-6. 10.1109/ECMSM.2015.7208683.

[18] Pawel Słowak, Piotr Kaniewski, "LIDAR-based SLAM implementation using Kalman filter," Proc. SPIE 11442, Radioelectronic Systems Conference 2019, 114420N (11 February 2020); https://doi.org/10.1117/12.2564818.

[19] Mur-Artal, Raul, J. M. M. Montiel and Juan D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System." IEEE Transactions on Robotics 31 (2015): 1147-1163.

[20] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," in IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, Oct. 2017, doi: 10.1109/TRO.2017.2705103.

[21] Milan Sonka, Vaclav Hlavac, and Roger Boyle. Image processing, analysis, and machine vision. Thompson Learning, 3 edition, 2008.

[22] Thrun, Sebastian and Montemerlo, Michael. (2006). The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. I. J. Robotic Res.. 25. 403-429. 10.1177/0278364906065387.

[23] E. Olson, AprilTag: A robust and flexible visual fiducial system. Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011: 3400-3407.

[24] Guo, Zhenglong and Fu, Qiang and Quan, Quan. (2018). Pose Estimation for Multicopters Based on Monocular Vision and AprilTag. 4717-4722. 10.23919/ChiCC.2018.8483685.

[25] AprilTag Official Website.
https://april.eecs.umich.edu/software/apriltag

[26] Dan Simon. Optimal state estimation: Kalman, H and nonlinear approaches. Wiley-Interscience, pp. 407-410, 2006.

[27] Jin, Pengju, Pyry Matikainen and Siddhartha S. Srinivasa. "Sensor fusion for fiducial tags: Highly robust pose estimation from single frame RGBD." 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017): 5770-5776.

[28] Vicon Motion Capture System Official Website.
https://www.vicon.com

[29] Turtlebot Official Website.
https://www.turtlebot.com