SOME STUDIES ON DIFFERENT NAVIGATION TECHNIQUES OF MOBILE ROBOT

REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF

M.Tech (Robotics & Automation)

Under the supervision of

Dr. PUSHPENDRA S. BHARTI

(Associate Professor)

By

VISHAL GAUTAM

To

University School of Information, Communication and Technology

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY
SEC-16C, DWARKA, NEW DELHI

May 2018

DCLARATION

This is to certify that Report entitled " SOME STUDIES ON DIFFERENT NAVIGATION TECHNIQUES OF MOBILE ROBOT "Which is submitted by me in partial fulfillment of the requirement for the award of degree M.Tech(Robotics and automation) in USICT,GGSIP University ,Dwarka, Delhi comprises only my original work and due acknowledgement has been made in the text to all other material used.

Date: 3/5/2018 **Name of student :VISHAL GAUTAM**

CERTIFICATE

This is to certify that Report entitled " SOME STUDIES ON DIFFERENT NAVIGATION TECHNIQUES OF MOBILE ROBOT" Which is submitted by **VISHAL GAUTAM** in partial fulfillment of the requirement for the award of degree M.Tech(Robotics and automation) in USICT,GGSIP University,Dwarka,Delhi is a record of the candidate own work carried out by him under my supervision. This matter embodied in this thesis is original and has not been submitted for the award of other degree.

Date: **Supervisor**

Dr. PUSHPENDRA S. BHARTI

(Associate Professor)

ACKNOWLEDGEMENT

I take this opportunity to express a deep sense of gratitude for Associate Professor **Dr. Pushpendra S. Bharti**, for providing excellent guidance and their support. It is because of his constant and general interest and assistance that this project has been successful. I would like to thank the member of Robotics Society at USICT, namely Manish kumar, and P.hD Scholar Sahil Dalal for their valuable suggestions and helpful discussions. I would also like to thank my family and friends who have been a source of encouragement and inspiration throughout the duration of this project.

VISHAL GAUTAM

Enrollment No. 00516418716

M.TECH (R & A),USICT

CONTENTS

LIST OF FIGURE

LIST OF TABLES

ABSTRACT

Now a day’s mobile robots are vastly used in many industries for performing different activities such as, manufacturing, security, automated transportation systems, material handling, warehouse management, packet distribution and arrangement ,and in working in accessible and dangerous sites ,etc. The goal of mobile robot is to reach a prescribed destination by using optimal path from all the paths available , and in least time, with certain precision and accuracy. To achieve this goal Navigation becomes one of the most important part for mobile robots. The problem of mobile robot navigation is very wide and complex. In the past years, there has been a tremendous progress in the area of autonomous robot navigation and a large variety of robots ,robot navigation algorithms have been developed who demonstrated robust navigation capabilities indoors and non-urban outdoor environments. In this thesis , we present simulation of different kind of navigation techniques for mobile robots designed to operate in static obstacles environments. We are using math works (MATLAB) software for simulation of navigation techniques . Problem of navigation of mobile robots can be sub divided into subparts i.e. mapping of environment ,localization of mobile robot in its environment, obstacle avoidance, path planning from present position to final (Goal) position. In this work we first describe the technique that is used to generate map of real world. After that we use object recognition technique in MATLAB ,to solve the localization problem of mobile robot.Then we describe and do simulation on different maps of real world environment and using different software, for four different kinds of navigation techniques which are most popular among mobile robots.

1. Artificial potential field based method.
2. Rule based navigation.
3. Probabilistic roadmap technique.
4. Navigation by using artificial neural network.

CHAPTER1

INTRODUCTION

1.1Background

Robotics is the branch of technology that deals with the design, construction, operation, and application of robots, as well as computer systems for their control, sensory feedback, and information processing. These technologies deal with automated machines that can take the place of humans in dangerous environments or manufacturing processes, or resemble humans in appearance, behavior, and/or cognition[6]. The goal of mobile robot is to reach a prescribed destination by using optimal path from all the paths available , and in least time, with certain precision and accuracy. To achieve this goal Navigation becomes one of the most important part for mobile robots. The problem of mobile robot navigation is very wide and complex. Navigation is a field of study that focuses on the process of monitoring and controlling the movement of vehicle from one place to another place. The environment which a mobile robot can encounter, affects the complexity of navigation problem. Static environment with few obstacles has least complex and for fast changing dynamic environment has most complex navigation problem. Many research efforts has been made worldwide on different aspects of mobile robot navigation and many approaches has been proposed . The solution proposed for navigation problem can be classified according to similarities in their problem solving approach. They can be classified as Local navigation and Global navigation.Local navigation by reactive means is employed by techniques such as fuzzy systems, or field-based systems. In this a robot relies only on current or recent sensor data[2]. This is useful for rapid responses to avoid collisions. Local navigation is used mostly in static environment.Global navigation looks at the broader objective of the robot, and identifies long-range paths for the robot to follow. The objective of the “Global Navigation” is to find an optimal path from the robot’s current location to the goal position. The goal position could be fixed or moving **(**as in it docking of two mobile robots).Example: Search Trees, Probabilistic techniques etc.

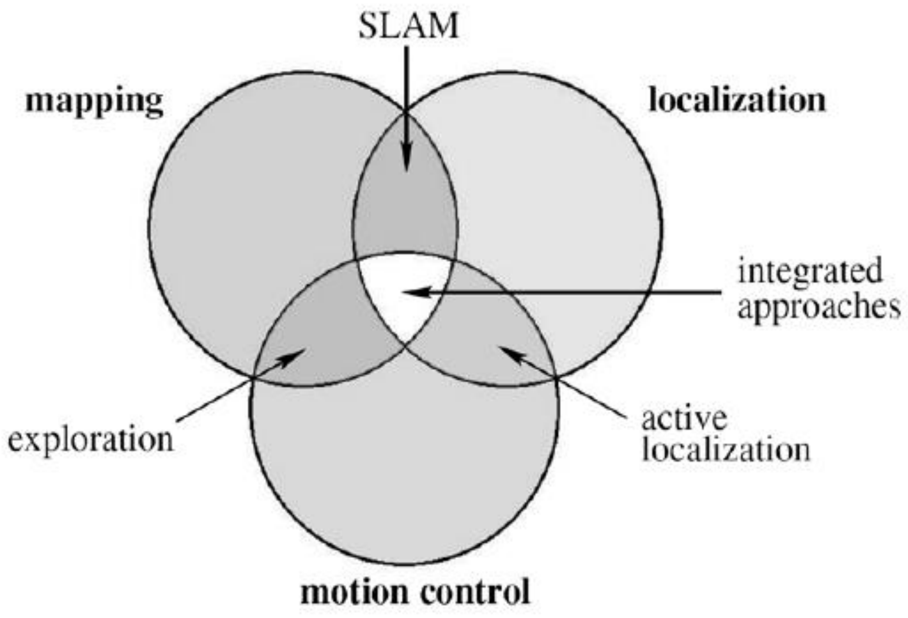


Figure1: Relationship between mobile robot research area[10]

Robot navigation is defined by the ability of a mobile robot to determine its own position in its frame of reference and then to plan a path towards some goal location In order to navigate in an environment, the mobile robot requires representation, i.e. a map of the environment, and the ability to interpret that representation. Therefore navigation can be defined as the combination of the five fundamental abilities.

Task of navigation of mobile robots is broken down into five subtasks .

1. First of all map of environment is constructed based on information gathered from mobile robot sensors.
2. Localization(Identifying the current location of the robot on map),and location of objects in its environment.
3. Path planning of mobile robot from initial position to final position.
4. Obstacle avoidance and changing the path to reach to its goal position .Choosing the optimal path from no of paths available .

5. Motion control, the mobile robot must modulate its motor output to move on optimal path.

These days simultaneous localization and mapping (SLAM) techniques are used, which submerged two subtask of navigation ,mapping and localization into one subtask. For maximum usefulness, the techniques to achieve these tasks should be generalized. They should be usable in a wide variety of situations, work in both static and dynamic environments, and should be applicable to a large class of robots. These days different type of navigation techniques are used for solving navigation problems in mobile robots ,each with its advantages and disadvantages.

1.2 Contributions

.In this thesis first we try to solve the problem of mapping and localization of mobile robot. For this we are using camera to first capture the image of real world environment then image is converted to map of environment, which is readable to simulation software and mobile robots operating system, For this we are converting image to occupancy grid map. For solving localization problem we using array of overhanging camera and object recognition algorithms in software (MATLAB), to localize mobile robot in indoor environment .After that we present simulation for four popular navigation techniques which are:

1. Artificial potential field based method : In this In this technique obstacles and goals are modeled as charged surfaces, and the net potential generates a force on the robot. These forces push the robot away from the obstacles, while pulling it towards the goal.
2. Rule based navigation: In this type of navigation fuzzy rules are used to model the relationship between input information and control output and is distinguished by its robustness with respect to noise and variation of system parameters.

3. .Probabilistic roadmap technique: The aim of the offline roadmap (graph) building stage is to randomly draw a small graph across the workspace. All vertices and edges of the graph should be collision-free so that a robot may use the same graph for its motion planning.
4. Navigation by using artificial neural network: With the use of Artificial Neural Network, self decision making by robot is performed. By using ANN robot is designed for a real time implementation and act as intelligent system. Use of the ANN in system improves performance.

   Testing algorithm using Simulation is much faster than testing with real robot. Most of the time the Robots themselves are quite expansive the experimenting a algorithm on it without testing it is not really the optimal way to design algorithms or test algorithms.

1.3 Thesis overview

The overview in this thesis are broadly divided into five chapters. First the introduction, Chapter 1, after that Chapter 2 provides literature review of navigation, mobile robot, mapping, localization, navigation techniques : Artificial potential field based method, Rule based navigation, Probabilistic roadmap technique, Navigation by using artificial neural network . in chapter 3 methodology and experimental setup are described in detail. In this section methods to construct map and localization algorithms are presented along this robot hardware design ,introduction of MATLAB and their related toolboxes image processing toolbox, robotics toolbox, arduino toolbox etc,software to generate dataset for Ann is also discussed . are also discussed in detail. In Chapter 4 Simulation of different navigation techniques on different real world environment and a detailed report of results and discussion has been given. This chapter summarizes the findings of all chapters discussed before. Finally in Chapter 5 conclusions of this research and future ways for further investigation has been discussed.

CHAPTER 2

LITERATURE REVIEW

2.1 Map building

Robot mapping is concerned with developing techniques that enable a mobile robot to construct and maintain a model of its environment based on spatial information gathered over time. the spatial information comes from directly perceiving the environment through external sensors. however, many more ways of acquiring spatial information, including external representations such as floor plans, sketches, or written descriptions[1], as well as direct communication with other robots or with humans[2]. Robot mapping is a challenging problem because of the uncertainty inherent in the available spatial information and in the model itself, which always is an approximation of the real world. The problem of representing the environment in which the robot moves is a dual of the problem of representing the robot's possible position or positions. Decisions made regarding the environmental representation can have impact on the choices available for robot position representation. Three fundamental relationships must be understood when choosing a particular map representation[10]:

1. The precision of the map must appropriately match the precision with which the robot needs to achieve its goals.

2. The precision of the map and the type of features represented must match the precision and data types returned by the robot's sensors.

3. The complexity of the map representation has direct impact on the computational complexity of reasoning about mapping, localization, and navigation.

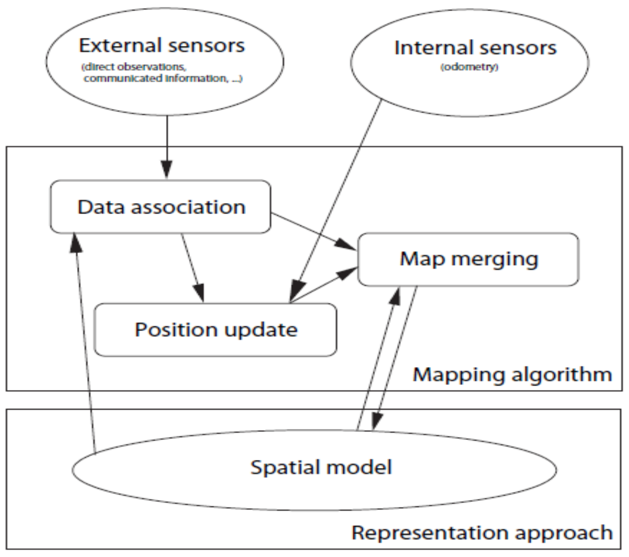


FIG2 : Incremental mapping information flow.

It is not an easy endeavor to compare existing mapping approaches in order to decide which one is most suitable under particular conditions. With regard to the spatial representations used in different mapping systems, we propose three general criteria to evaluate and compare different approaches: (1) extractability and maintainability, (2) information adequacy, and (3) efficiency and scalability. We are going to discuss these criteria in the following.

(1)Extractability and maintainability

It is crucial that a spatial model formulated within the chosen spatial representation approach can be constructed and maintained from the information available to the robot. This means we need to be able to formulate algorithms that take the input information and update the model accordingly. This requires managing the uncertainty in the input information.Generally, the information required by approaches that model the environment in a low-level sensor-near way seems to be much easier to extract and maintain because no sophisticated processing of the sensor data is needed. The effects of imperfect sensors can be explicitly modeled statistically. On the other hand, more abstract representations may have the advantage that the modeled relations can be more reliably derived from the sensor data, e.g., we might be able to reliably tell that a certain object is between two other objects, while it is much harder to determine the exact shape and location of an object.Extractability and maintainability of representation approaches can also vary significantly for different kinds of environments. For instance, an approach may only be suitable for well-structured indoor environments or rely on artificial unique landmarks. We will call representation approaches that can adequately represent arbitrary environments universal.

(2)information adequacy

Another aspect is information adequacy , that the level of detail is sufficient to support the regarded operations. However, demand for low computational costs and low space consumption warrants a certain pursuit of sparseness: A representation should not contain extra information, information that is required neither for any of the operations nor for maintaining the model itself, and required information should only be represented at a level of detail that is really needed.

(3)Efficiency and scalability

The ways in which a certain kind of spatial information can be represented are limitless. As a mobile robot typically is supposed to work in real time, the operations should be as efficient as possible. Efficiency significantly depends on the way the information is represented. Typically, there are trade-offs involved in which one way of representing things favors a certain operation while making other operations more expensive. A good spatial representation, therefore, would be one which optimizes the overall performance over all operations working on the spatial model, which is rather difficult to assess. However, looking at the complete set of operations, and not only at the efficiency of map construction and map maintenance, will give a much better picture. In addition to the efficiency aspect, we will use the term scalability to discuss how well a representation approach scales with the size of the represented environment. This concerns efficiency of operations as well as space consumption.

2.1.1Map representation

In the following section, we review the literature on robot mapping regarding the spatial representation approaches employed. We distinguish between different basic spatial representation approaches, which are the elementary representation formalisms to describe an environment in a homogeneous way, and different organizational forms, which describe different ways of combining basic spatial representation approaches to form more complex representation structures.

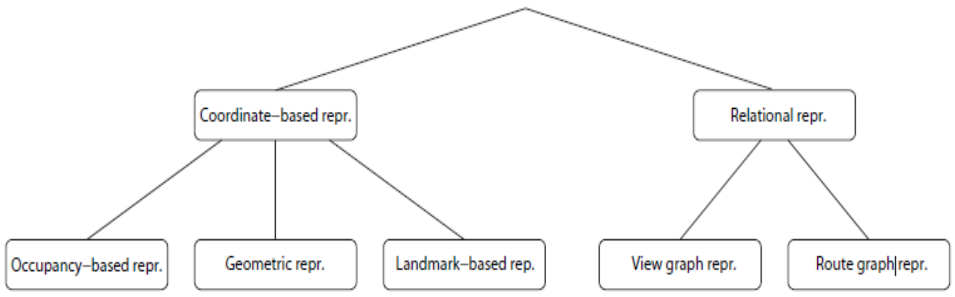


FIG 3 Types of map representation[10]

In this thesis we are using only occupancy grid map ,so we discuss only occupancy grid map. Occupancy Grid creates a 2-D occupancy grid object, which you can use to represent and visualize a robot workspace, including obstacles. The integration of sensor data and position estimates create a spatial representation of the approximate locations of the obstacles. Occupancy grids are used in robotics algorithms such as path planning. They are also used in mapping applications, such as for finding collision-free paths, performing collision avoidance, and calculating localization. You can modify your occupancy grid to fit your specific application. Each cell in the occupancy grid has a value representing the occupancy status of that cell. An occupied location is represented as true (1) and a free location is represented as false (0). The two coordinate systems supported are world and grid coordinates. The world coordinates origin is defined by Grid Location In World, which defines the bottom-left corner of the map. The number and size of grid locations are defined by the Resolution. Also, the first grid location with index (1,1) begins in the top-left corner of the grid. Occupancy-based representations represent occupied and free parts of space equitably by decomposing space into cells and storing for each cell whether it is (at least partially) occupied or (entirely) free.

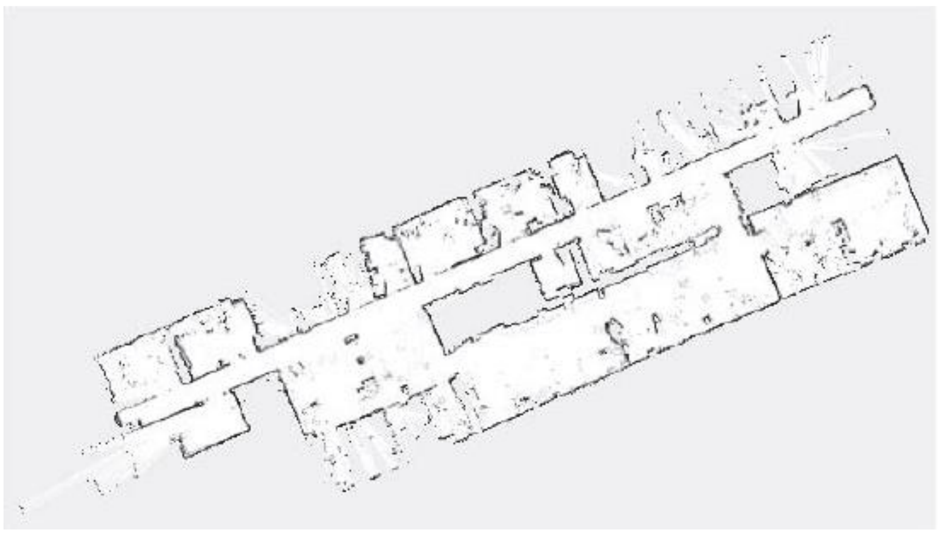


FIG 4 Occupancy grid map constructed using ROS[5].

2.2 Localization of mobile robot

localization has received the greatest research attention in the past decade and, as a result, significant advances have been made on this front.. Localization has been a hot topic in mobile robotics since the information of the robot pose (position and orientation) is an essential demand for many applications. In order to deal with general application, SLAM (simultaneous localization and mapping) has been proposed to study the environment expression and localization problem. Some excellent algorithms using 2D laser range finder [40], [41] have been proposed. Since dynamic obstacles appearance generally because of human activities, the robot pose error is accumulated gradually. However, cumulative error is difficult to eliminate

if the mobile robots move fast [43]. What's more, the position of some static obstacles could also be changed, so the robustness of mobile robots localization is of great significance. To achieve good performance in localization, it is very common to use map-based localization approaches for indoor mobile robots. And some methods need more than one sensor or modification of the environment. For example, cameras, lasers, and sonars are used in [42], [45] for working stably; the working space of mobile robots is limited [40], and mobile robots only required to be able to accurately reach several positions in industrial environments First, describes how sensor and effectors uncertainty is responsible for the difficulties of localization. Then, describes two extreme approaches to dealing with the challenge of robot localization: avoiding localization altogether, and performing explicit map-based localization.

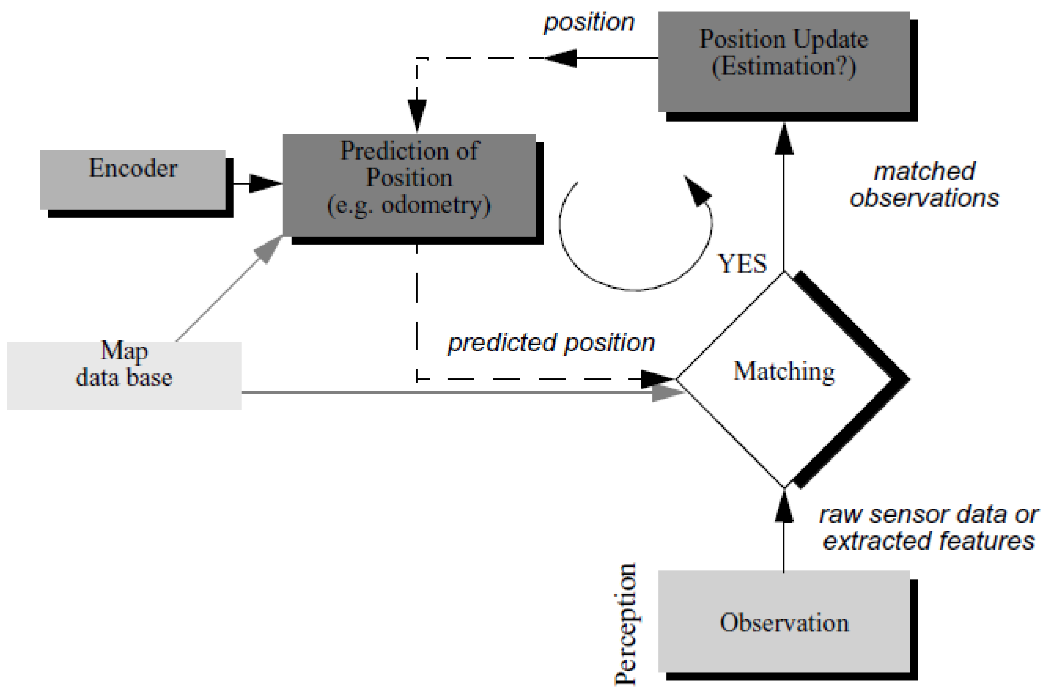


FIG 5.LOCALIZATION REPRESENTATION [33]

To achieve good performance in localization, it is very common to use map-based localization approaches for indoor mobile robots. And some methods need more than one sensor or modification of the environment. For example, cameras, lasers, and sonar are used for working stably; the working space of mobile robots is limited , and mobile robots only required to be able to accurately reach several positions in industrial environments; even the environment need to be modified with some markers such as QR code[45]

2.3Artifical potential field

The “Artificial Potential Fields” (AFP) method involves modelling the robot as a particle in space, acted on by some combination of attractive and repulsive fields. In this technique obstacles and gods are modeled as charged surfaces, and the net potential generates a force on the robot. These forces push the robot away from the obstacles, while pulling it towards the goal. The robot moves in the direction of greatest negative gradient in the potential. Despite its simplicity this method can be effectively used in many simple environments. However, the simple APF method described above have several key problems . The field may contain local minima, especially when there are many obstacles in the environment. This can trap the robot, its a gradient-descent algorithm cannot escape from it local minimum. The robot may find itself unable to pass through small openings such as through doors. The robot may exhibit oscillations in its motions. The robot may be guided away from the goal by a moving obstacle which creates a moving local minimum. Being stuck in this ”shadow“ means that the robot cannot move around the obstacle. There have been various attempts to address these issues in the APF. These have included virtual obstacles . created to offset minima, alternate field functions including harmonic functions and distance transforms with no local minima, by decreasing the repulsion of obstacles not in the robots direction of motion. This reduces the amount of oscillation, while still allowing the robot to avoid obstacles in its path. These are successful in the static environment, but may not be as suitable for the dynamic environment as the computational complexity is very high. The following methods attempt to utilise the simplicity of a field-based approach, but avoid the drawbacks listed above.[33]

There are number of methods and algorithm initially developed for single mobile robots working in environment containing static obstacles .Artificial potential field based navigation method is from one of the initially developed navigation method .Due to its simplicity this method can be effectively used in many simple environments and still popular in mobile robot navigation. This method was originally invented for robot manipulator path planning and now is used often and under many variants in the mobile robots. This method is a reactive planning technique, where the immediate distances from obstacles are considered to compute the immediate move, without much bothering about the future. In such a manner immediate actions lead to motion of the robot, ultimately reaching to the goal position. This methods solves the sub problems of navigation like localization, path planning ,and control of mobile robots .[34] The resulting artificial potential field is a control law for the robot. Due to this field the robot can localize its position with respect to map, it can always determine its next required action based on the field.

In this method artificial potential field is created across the robot's map and this field directs the robot to goal position .In this robot is modeled as a particle in space, acted on by superposition of attractive and repulsive fields. In this technique obstacles and goal are modeled as charged surfaces, and the net potential generates a force on the robot. These forces push the robot away from the obstacles, while pulling it towards the goal. The robot moves in the direction of greatest negative gradient in the potential The basic idea behind all potential field approaches is that the robot is attracted toward the goal, while being repulsed by the obstacles . In the simplest case, we assume that the robot is a point, thus the robot's orientation is neglected and the resulting potential field is only 2D . If we assume a differentiable potential field function $U(q)$, we can find the related artificial force) F(q) acting at the position q = (x, y)

. $F(q) = -\nabla U(q)$

where $\nabla U(q)$ denotes the gradient vector of at position .

The potential field acting on the robot is then computed as the superposition of the attractive field of the goal and the repulsive fields of the obstacles.

$U(q) = U\ \text{attractive}(q) + U\ \text{repulsive}(q)$

Similarly, the forces for attracting and repulsing part in given map

$F(q) = F\ \text{attractive}(q) - F\ \text{repulsive}(q)$

$F(q) = -\nabla U\ \text{attractive}(q) - \nabla U\ \text{repulsive}(q)$[33]

2.4 Fuzzy based navigation

A sensor-based navigation algorithm, combines two types of obstacle avoidance behaviors, each for the convex obstacles and the concave ones is proposed byBarret, Benreguieg, and Maaref [14] . To avoid the convex obstacles the navigator uses either fuzzy tuned artificial potential field (FTAPF) method or a behavioural agent, however an automatically online wall-following system using a neuro-fuzzy structure has been designed for the concave one. a virtual target approach for resolving the limit cycle problem in navigation of a behaviour-based mobile robot. The real target has been switched to a virtual location so that robot can navigate according to the virtual target until it detects the opening. The efficiency and effectiveness of the refined fuzzy behaviour-based navigation are demonstrated by means of both simulation and physical experiments. Aguirre Eugenio and Gonzalez Antonio [14] dealt with a hybrid deliberative-reactive architecture for mobile robot navigation for integrating planning and reactive control, and attention is focused on the design, coordination and fusion of the elementary behaviours. Saade and Khatib [19] have developed a data-driven fuzzy approach to provide a general framework for solving the Dynamic motion problem (DMP) problem of a mobile robot under some constraints[17]. The main advantage of the current approach over recent fuzzy-genetic one is that the robot can navigate successfully in the presence of moving obstacles and independently of the number of these obstacles. The proposed approach has also reveals the reduction in the travel time. The proposed algorithm has shown good results as compared to ANFIS on robot trajectory in terms of their length and the time required by the robot to reach the goal. The superiority of the new algorithm can be helpful in building fuzzy models without any compulsion of planting effort in gaining accurate and enormous number of data points. Li and Hseng (2003) have designed and implemented a new fuzzy controller for a car-like mobile robot (CLMR) that holds autonomous garage-parking and parallel-parking capacity by using real time image processing. The system consists of a host computer, a communication module, a CLMR, and a vision system. Fuzzy garage parking control (FGPC) and fuzzy parallel parking control (FPPC) have been used in order to control the steering angle of the CLMR. have presented a new method for the intelligent control of the nonholonomic vehicles. Fuzzy perception has been directly used, both in design of each reactive behaviour and solving the problem of behaviour combination in order to implement a fuzzy behaviour based control architecture. The capabilities of the control system have been improved by considering teleoperation and planned behaviour, together with their combination with reactive ones. Experimental results have shown the robustness of the suggested technique. Abdessemed Foudil, Benmahammed Khier, and Monacelli Eric (2004) have used the fuzzy logic controller in the development of complete navigation procedure of a mobile robot in a messy environment. An evolutionary algorithm has been implemented in order to solve the problem of extracting the IF-THEN rule base. The validity of the proposed method has been demonstrated through simulation results. Demirli and Molhim (2004) have presented a new fuzzy logic based approach for dynamic localization of mobile robots. The proposed approach uses sonar data obtained from a ring of sonar sensors mounted around the robot. The angular uncertainty and radial imprecision of sonar data are modelled by possibility distributions.

The information received from the adjacent sonar sensors are united, which helps in the reduction in the uncertainty in sonar impressions. In the beginning a local fuzzy map has been constructed with help of reduced models of uncertainty, and then fitted to the given global map of the environment to identify robot"s location. This fit offers either a unique fuzzy location or multiple candidate fuzzy locations. Since the coordinates (x, y) and orientation of the identified locations are represented by possibility distribution, these locations are referred to as fuzzy locations. To reduce the number of candidate locations, a new set of candidate fuzzy location is obtained by moved the robot to a new position. By considering the robot"s movement, a set of hypothesized locations is identified from the old set of candidate locations. The hypothesized locations are matched with the new candidate locations and the candidates with low degree of match are eliminated. This process is continued until a unique location is obtained. The matching process is performed by using the fuzzy pattern matching technique. The proposed method is implemented on a Nomad 200 robot and the results are reported. Parhi [22] has described a fuzzy logic based control system for the navigation of multiple mobile robots in a cluttered environment, such that the robots do not collide to each other. For this he has used fuzzy logic controller to combine the fuzzy rules in order to direct the steering of the robot to avoid the obstacles present in its path. Moreover Petri Net model has been used by implementing crisp rules to avoid the collision between the different mobile robots. Simulation and test results validate the system functions by enabling the robots to reach their goal without hitting the static obstacles or colliding with other robots[7]. Fatmi *et al.* [19] have demonstrated a successful way of constructing the navigation task in order to deal with problem of autonomous navigation of mobile robot. Issues of individual behaviour design and action coordination of the behaviours were addressed using fuzzy logic.

Rule based navigation is an alternative to field-based methods. Rule based method use simple rules like - “if left sensor active, turn right”, or they can be generalized by fuzzy logic and machine learning techniques. Simple rules are easy to use, but are often very limited to the environments for which they are built. Fuzzy control systems, employing fuzzy set theory, have been proven to produce better performance than simple rules. However, these too are limited to environments highly similar to the one in which they were constructed. To overcome this limitation of fuzzy rule based system this method is used mostly in hybrid with other navigation problem solving approach. the Fuzzy Logic System by using membership functions that takes input from mobile robot sensors and gives output as left and right motor speed that implicitly controls the robot turn . Membership functions have a given pre-determined range of values that control the state of an input or an output. The fuzzy rule is a Iinguistic set of if-then statements. The fuzzy system is a result of a lot of manual tuning of the rules and membership functions, over a wide variety of scenarios

2.5 Probabilistic roadmap

A probabilistic roadmap (PRM) is a network graph of possible paths in a given map based on free and occupied spaces. The robotics.PRM class randomly generates nodes and creates connections between these nodes based on the PRM algorithm parameters. Nodes are connected based on the obstacle locations specified in Map, and on the specifiedConnectionDistance. You can customize the number of nodes, NumNodes, to fit the complexity of the map and the desire to find the most efficient path. The PRM algorithm uses the network of connected nodes to find an obstacle-free path from a start to an end location. To plan a path through an environment effectively, tune the NumNodes andConnectionDistance properties.

When creating or updating the robotics.PRM class, the node locations are randomly generated, which can affect your final path between multiple iterations. This selection of nodes occurs when you specify Map initially, change the parameters, or update is called. To get consistent results with the same node placement, use rng to save the state of the random number generation.

Use the NumNodes property on the PRM object to tune the algorithm. NumNodes specifies the number of points, or nodes, placed on the map, which the algorithm uses to generate a roadmap. Using the ConnectionDistance property as a threshold for distance, the algorithm connects all points that do not have obstacles blocking the direct path between them.

Increasing the number of nodes can increase the efficiency of the path by giving more feasible paths. However, the increased complexity increases computation time. To get good coverage of the map, you might need a large number of nodes. Due to the random placement of nodes, some areas of the map may not have enough nodes to connect to the rest of the map. In this example, you create a large and small number of nodes in a roadmap.[46]

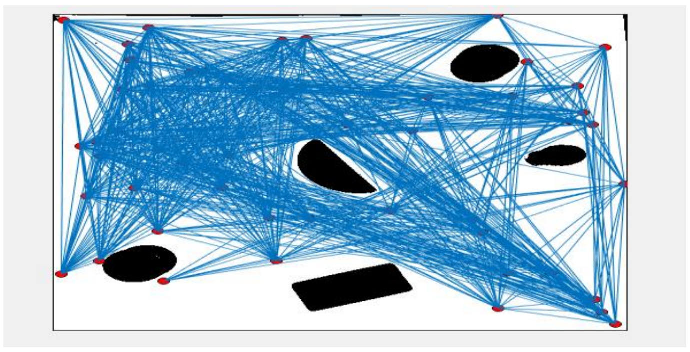


FIG6 : probablistic road map

2.6 Artifical neural network

Artificial neural networks are information-processing systems which have certain performance characteristics in common with biological neural networks [30]. Artificial neural networks have been evolved as generalizations of mathematical models of human cognition or neural biology, based on the following four assumptions [33]:

1. "Information processing occurs at many simple elements called neurons."

2. "Signals are passed between neurons over connection links."

3. "Each connection link has an associated weight, which, in a typical neural net,

multiplies the signal transmitted."

4. "Each neuron applies an activation function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal." A neural network can be characterized, firstly, by its structure of connections between the neurons (known as its architecture), additionally by its method of determining the weights on the connections (called its training, or learning, algorithm), and finally, its activation function.

Neural networks are structured from a large number of simple processing components called neurons, units, cells, or nodes. Each neuron is connected to other neurons through directed communication links, each with a weight associated to it (as shown in Figure 1). The weights correspond to information being processed by the network to solve a problem. Neural networks can be applied to a wide selection of problems, such as storing and recalling data or patterns, grouping similar patterns, performing general mappings from input patterns to output patterns, classifying patterns, or finding solutions to constrained optimization problems

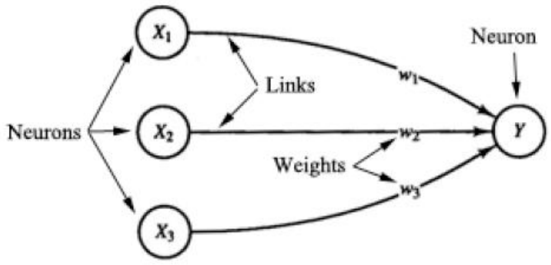


FIG 7 ann basic network[30]

The internal state of a neuron is known as its activation or activity level, which is a function of the inputs it has received. Typically, activation is sent as a signal from one neuron to several other neurons. However, only one signal can be sent from each neuron at the same time, although that signal can be broadcast to several other neurons. For example, consider neuron Y, shown in Figure 1, that receives inputs from neurons $X1$, $X2$, and $X3$. The activations (output signals) of these neurons are $x1$, $x2$, and $x3$, respectively. In addition, the weights on the connections from $X1$, $X2$, and $X3$ to neuron Y are $w1$, $w2$, and $w3$, respectively. The net input, y_in, to neuron Y is the sum of the weighted signals from neurons $X1$, $X2$, and $X3$, that is:

$y\_in = w1x1 + w2x2 + w3x3$

Further, suppose that neuron Y is connected to neurons $Z1$, and $Z2$, with weights $v1$, and $v2$, respectively, as depicted in Figure 2. neuron Y sends its signal y to each of these units. However, generally, the values received by neurons $Z1$, and $Z2$ will be different. Since each signal is scaled by the appropriate weight, $v1$ or $v2$. As shown in this simple example, in a typical network, the activations $z1$ and $z2$ of neurons $Z1$, and $Z2$ would depend on inputs from several neurons and not just one. [33] Even though the neural network in Figure is very simple, the presence of an intermediate unit Y (also known as the hidden unit), together with a nonlinear activation function, gives the network the capability to solve many more problems than can be solved by a network with only input and output units. However, the difficulty to train (i.e., find optimal values for the weights) a net with hidden units is more than a network

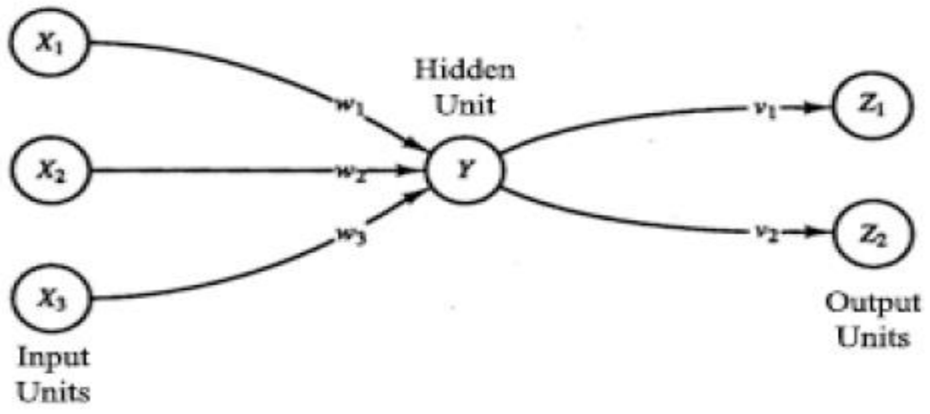


FIG 8 neural network[30]

CHAPTER 3

METHODOLOGY AND EXPERIMENTAL SETUP

3.1.1 Mapping

In this work we use a robot arena with an overhead camera as shown in Figure 30 . The camera can be easily calibrated and the image coming from the camera can be used to create a robot map, as shown in the figure 31. This is a simplistic implementation of the real life scenarios where multiple cameras can be used to capture different parts of the entire workspace, and their outputs are fused to create an overall map used by different kinds of motion planning algorithms .In this section we are using image processing to create the map. Image processing is a discipline of computer science and applied mathematics that studies digital images and their transformations in order to improve their quality or extract information.

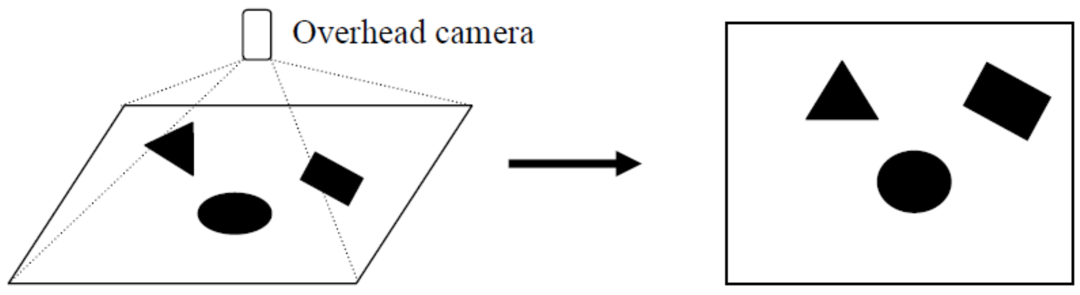


figure 10 : Over head camera system for creation of robot map[30]

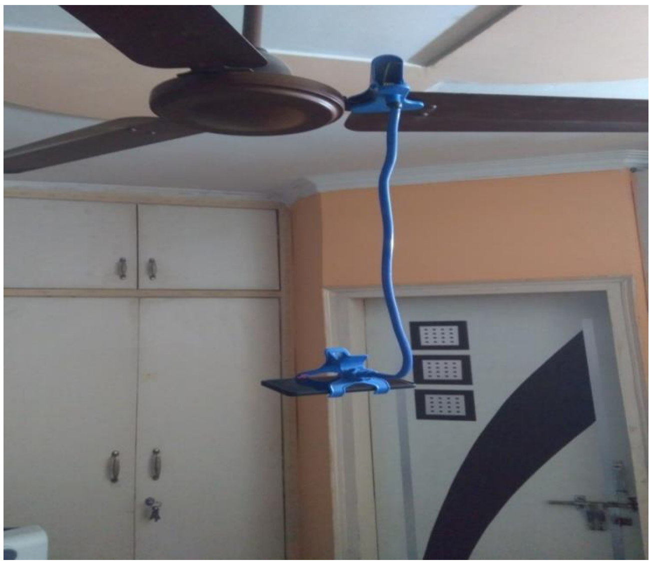


Figure 11over head mobile camera to capture images

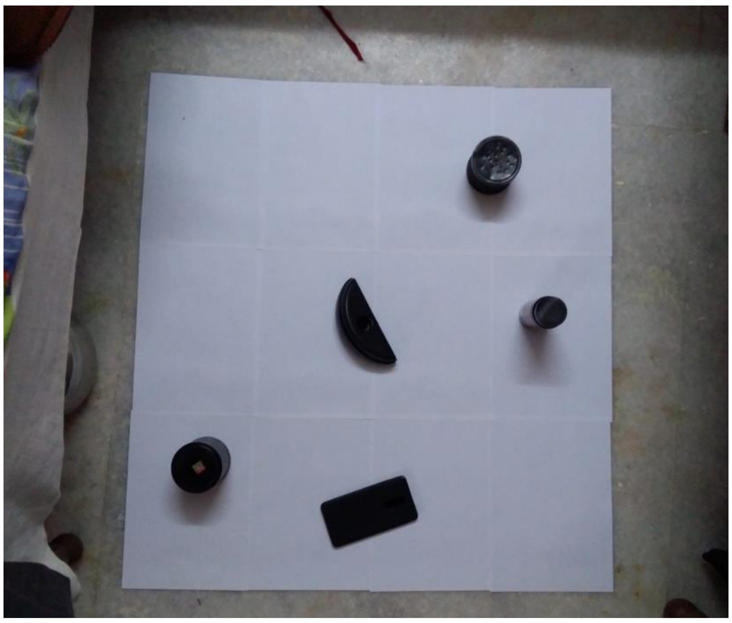


Figure 12 image of robot real environment

STEPS TO CREATE OCCUPANCY GRID MAP BY USING OVERHEAD CAMERA:

1. First an overhead camera captures a color image (RGB image).

2.By using MATLAB image processing toolbox ,image processing is done which has following steps.

2.1.RGB image is converted to gray image. The gray level of an image is simply designed to have colors that are all expressed in gray. In fact, the "gray" is a color in which the components: red, green and blue have the same intensity in space RGB (Red, Green and Blue). The easiest method to convert the color image to another gray level is to calculate the luminance of a pixel using the following equation[45]:

Gray = $0.299 \times 0.587 \times$ Red + Green + $0.114 \times$ Blue (1)

2.2 Histogram equalization of gray image is done.

2.3 Then the image is converted into binary image.

2.4 After histogram equalization thersholding of the image is done. we must convert the image to grayscale input format and then converts this image grayscale by binary threshold. The output binary image from 1 (white) for all the pixels of the input image with higher luminance as the level and from 0 (black). Therefore it with the matlab command[45].

bw_ima = im2bw(RGB1,level_ima); (2)

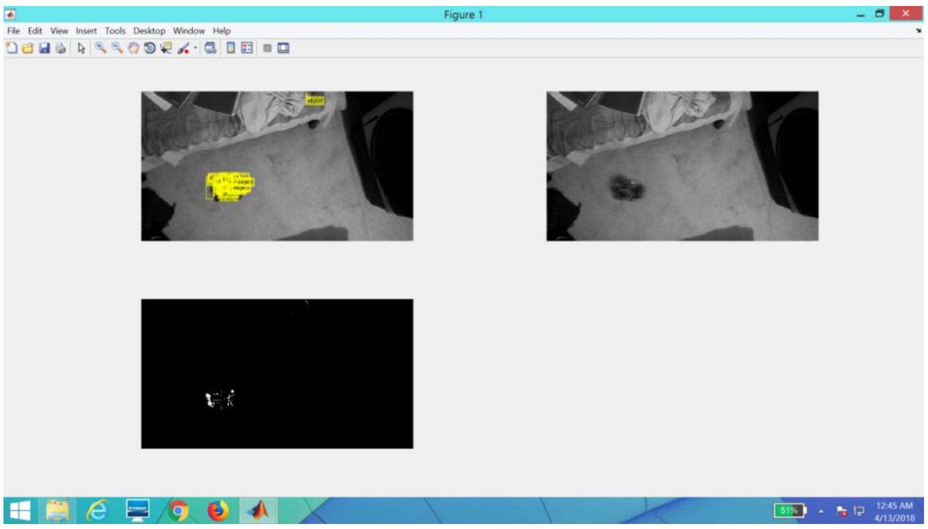
3.Then this .jpg image (mostly) is converted into .bmt image. or .png according to need of software using.

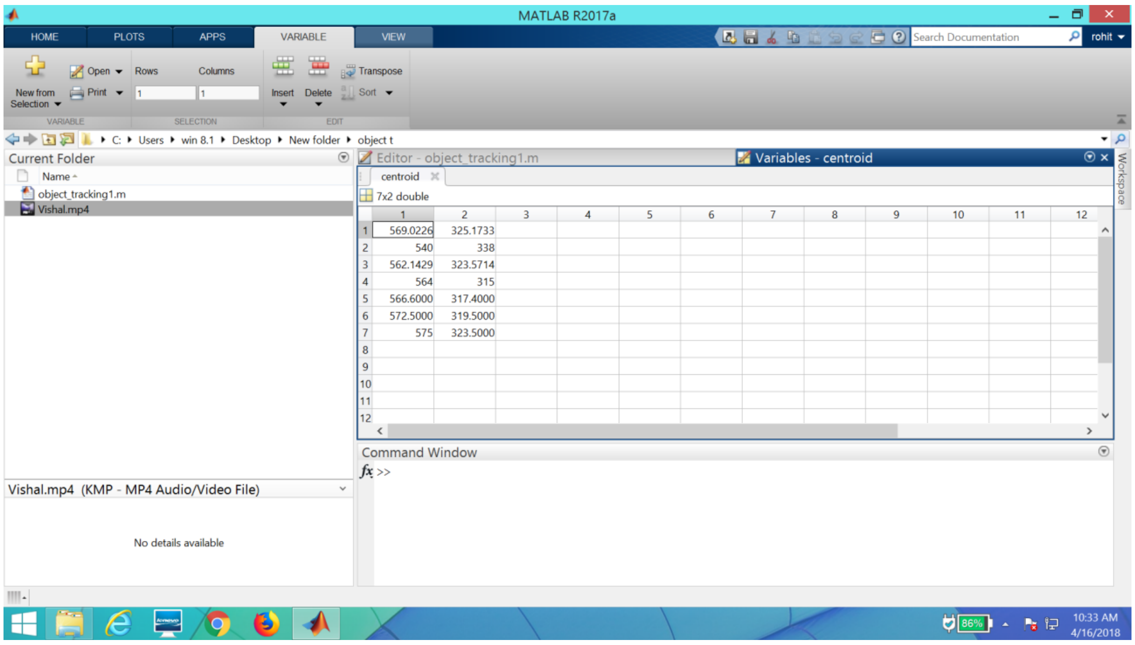
4. Then this .bmt image or .png image can be used as occupancy grid map in simulation of navigation of mobile robot.

3.1.2 LOCALIZATION

The camera can be easily calibrated and the image coming from the camera can be used to detect robot as shown in the figure. This is a simplistic implementation of the real life scenarios where multiple cameras are used to capture different parts of the entire workspace, and their outputs are fused to locate an overall map used by the motion planning algorithms.

The same camera can also be used to capture the location of the robot at the start of the planning and also as the robot moves. This solves the problem of localization. An interesting looking region of interest becomes the goal of the robot to be used in the motion planning algorithms.As the robot passes from the range of overhead camera its start capturing video and with the help of MATLAB computer vision toolbox motion of mobile robot is tracked and mat file is generated which contain pixal value(coordinate value)all the values from where robot passes.In this way problem of localization for indoor mobile robot can be solved. x and y coordinate vale is shown in workspce of matlabas shown in fig 12





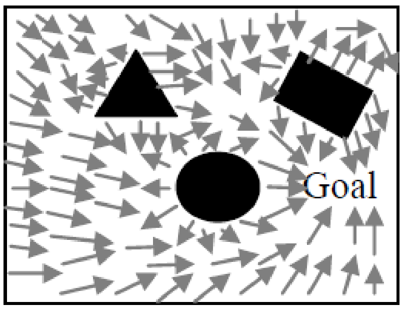
	1	2	3	4	5	6	7	8	9	10	11	12
1	569.0226	325.1733										
2	540	338										
3	562.1429	323.5714										
4	564	315										
5	566.6000	317.4000										
6	572.5000	319.5000										
7	575	323.5000										
8												
9												
10												
11												
12												

```
fx >>
```

FIG12 :mobile robot tracking by using matlab.

3.1.3 POTENTIAL FIELD NAVIGATION

Artificial Potential Field based navigation is a reactive planning technique, where the immediate distances from obstacles are considered to compute the immediate move, without much bothering about the future. In such a manner immediate actions lead to motion of the robot, ultimately leading to the goal. All obstacles repel the robot with a magnitude inversely proportional to the distance. The goal attracts the robot. The resultant potential, accounting for the attractive and repulsive components is measured and used to move the robot. The potential field for a sample scenario is shown in Figure 2. Directions indicate the direction of the potential vector. The distance of the obstacles at all angles from the robot is measured. In this work we only use 5 distances at specific angles are measured to compute the repulsive potential. These are forward, left side, right side, forward left diagonal and forward right diagonal. The different inputs are summarized in Figure 10.



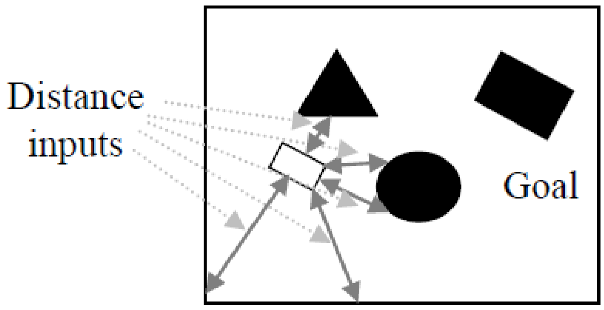


FIG13 :representation of potential field

3.1.4 FUZZY RULE BASED NAVIGATION

Fuzzy based navigation is a reactive planning technique, where the immediate position and distances from obstacles is considered to compute the immediate move, without much bothering about the future. In such a manner immediate actions lead to motion of the robot, ultimately leading to the goal. In order to solve the problem using fuzzy logic, we first need to select a few inputs which best represent the situation that the robot is currently placed in. The decision of motion is made purely on the basis of these inputs and not the actual scenario. For this problem 6 inputs are selected. These are distance from the obstacle in front, distance from the obstacle at the front left diagonal, distance from the obstacle at the front right diagonal, angle between the heading direction of robot and the goal, distance of the robot from the goal and preferred turn. The different inputs are summarized in Figure 11. The last input, preferred turn indicates whether it would be beneficial to turn clockwise or anti-clockwise, all other inputs ignored. A simple rule is used to set the parameter. If the front obstacle is far away, turn is so as to more face the goal. If the front obstacle is close and a new front obstacle is encountered, turn using the side of the goal is preferred. If the front obstacle is close and the same obstacle as encountered in the previous step is found, the same turn as made previously is repeated.

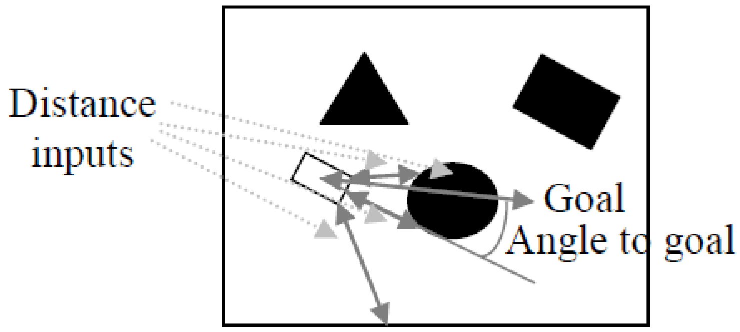


FIG 14: representation for direction of fuzzy logic program

3.1.5 PRBALABISTIC ROADMAP NAVIGATION TECHNIQUE

The algorithm has two stages: an offline roadmap (graph) building stage and an online planning/query stage. The aim of the offline roadmap (graph) building stage is to randomly draw a small graph across the workspace. All vertices and edges of the graph should be collision-free so that a robot may use the same graph for its motion planning. The PRM selects a number of random points (states) in the workspace as the vertices. In order to qualify being a vertex, a randomly selected point (state) must not be inside some obstacle. Let there be *k* number of states which is an algorithm parameter. Higher are the number of vertices or *k*, better would be the results with a loss of computational time. The algorithm then attempts to connect all pairs of randomly selected vertices. If any two vertices can be connected by a straight line, the straight line is added as an edge. The concept is shown in Figure 12.

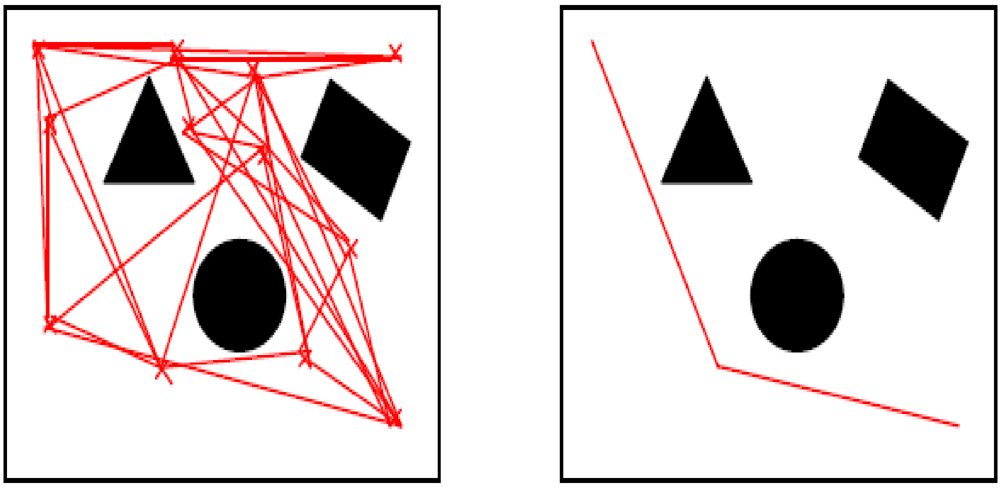


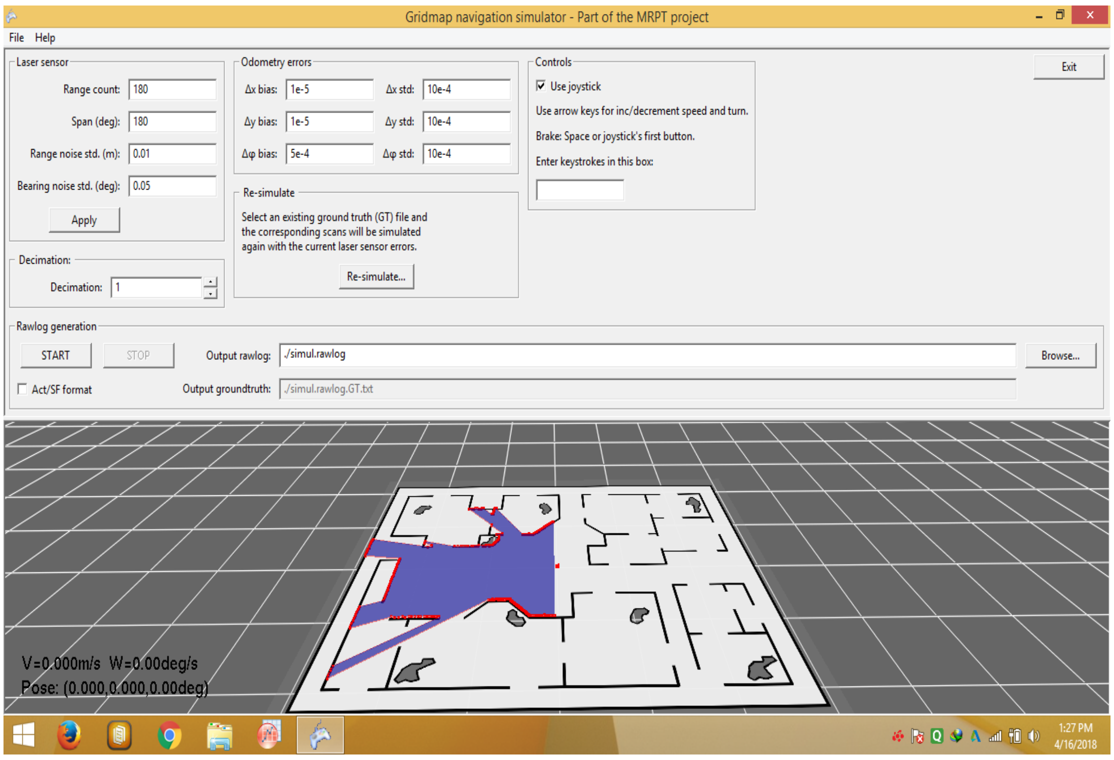
FIG 15: prm representation with a* algo computed path

The online planning/query stage aims to use the roadmap (graph) developed earlier for planning the path of a robot. Since a graph is already known, any graph search algorithm can be used. The code uses A* algorithm for the same. The weights of the edges is taken as the Euclidian distance between the connecting points, and the heuristic function (denoting the nearness of the point to the goal) is taken as the Euclidian distance to the goal. Both the functions are given as separate files from where they can be changed.

3.6 NAVIGATION BY USING ARTIFICIAL NEURAL NETWORK

Navigation by using artificial neural network is a global navigation technique. For this technique we can use mobile robot hardware along with MATLAB or Mrpt (Mobile Robot Programming Toolkit) along with MATLAB can be used.Description of mrpt is presented in next section.First step in this technique is to generate dataset for training and testing of neural network technique .Dataset is nothing but information gathered by running the robot several times in known environment condition .After getting dataset matfile is generated and this matfile which consists of raw data from mobile robot sensors is used for training of neural network and testing by using neural network toolbox in MATLAB.After that neural network takes input from mobile robot sensors and in output it gives control action to robot .For this technique we are using mrpt to generate the dataset by using known static environment map generated previously, inputs are selected in mobile robotics programming toolkit and move the robot in simulation using 4 keys of laptop keyboard i.e. up ,down ,left ,right.

Environment condition :


Gridmap navigation simulator - Part of the MRPT project

Laser sensor
Range count: 180
Span (deg): 180
Range noise std. (m): 0.01
Bearing noise std. (deg): 0.05

Decimation:
Decimation: 1

Odometry errors
Δx bias: 1e-5   Δx std: 10e-4
Δy bias: 1e-5   Δy std: 10e-4
Δφ bias: 5e-4   Δφ std: 10e-4

Re-simulate
Select an existing ground truth (GT) file and the corresponding scans will be simulated again with the current laser sensor errors.

Controls
[✓ Use joystick]
Use arrow keys for inc/decrement speed and turn.
Brake: Space or joystick's first button.
Enter keystrokes in this box:

Rawlog generation
Act/SF format
Output rawlog: ./simul.rawlog
Output groundtruth: ./simul.rawlog.GT.txt

V=0.000m/s W=0.00deg/s
Pose: (0.000,0.000,0.00deg)

FIG 16 ENVIRONMENT FOR ANN SIMULATION [31]

The mobile robot designed has an laser sensor (a distance meter) and can scan the space in front from -90 to +90 degrees. The sensing has been made at , 0 to 180 degree angles. The signal of the sensor at these angle stored .rawlog file.Which can be converted to. .txt file . Files "vis..rawlog" are binary files generated through the MRPT library. They can be visualized, edited, and managed in many ways through the GUI application "RawLog Viewer".

The text files "vis_LASER.txt" and "vis_ODO.txt" have been automatically generated from the binary rawlog by "RawLogs Viewer".

The "xxx_LASER.txt" file contains one scan range per line. Values are in meters. The "xxx_ODO.txt" contains robot pose increments as measured by the odometry. Each

line contains 3 values:

-Ax: Increment in "x", in meters.

-Ay: Increment in "y", in meters.

-Aphi: Increment in "phi", in radians. Phi=0 is in the direction of "+X" axis.

After generating the dataset .MATLAB 2017 neural network toolbox is used to train ing and testing of dataset . It is trained using actual data and generates weights of connections for optimized condition using feedforward neural network Algorithm and with ‘tanh’ sigmoid threshold function.

3.2 EXPERMINTAL SETUP

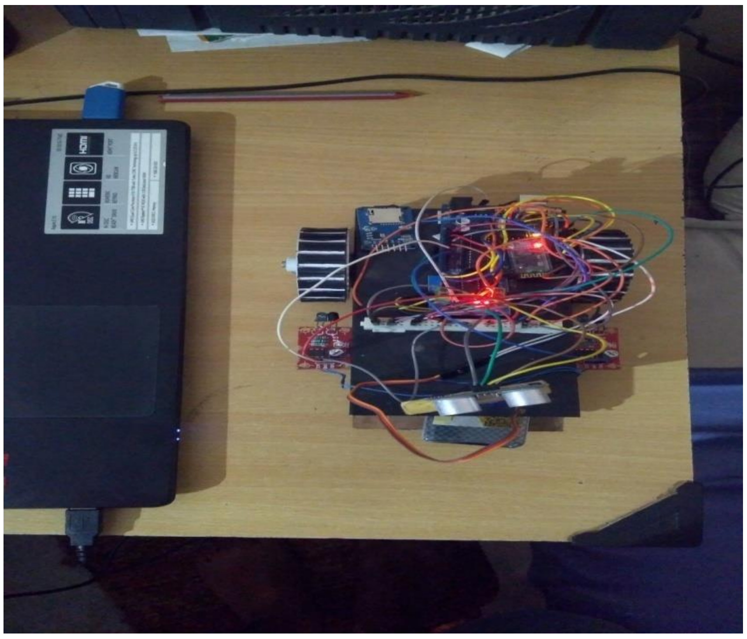


FIG 17MOBILE ROBOT

For experimental setup MATLAB 2016 , MATLAB 2017 are used for simulation of navigation technique, and mobile robot is also used for simulation of localization problem.Mobile Robot Programming Toolkit provides developers with portable and well-tested applications and libraries covering data structures and algorithms employed in common robotics research areas. It is open source,software.

Mobile robot has following components.

1. Arduino Uno which is based on Atmega 328p microcontroller which can process data in both digital and analog for input as well as output purpose.
2. servo motors are arranged and mechanically fixed.
3. Mobile robot setup is wirelessly controlled by an app developed on MIT App Inventor which has four sliders which individually controls the position of each servo motor via HC-05 Bluetooth module. The Bluetooth module serially communicates with the microcontroller via TX and RX pins.
4. Motor controler (h bridge).
5. 2 standred wheel and one omni wheel.
6. wheel encoder.
7. dc motor

8. lipo battry
9. utlrasonic sensor
10. lidr sensor
11. memory card module to store sensors data which is used to generate dataset.
12. bread board and jumper wires for connection.

CHAPTER 4

SIMULATION AND RESULT ANALYSIS

Simulation of navigation techniques are done in MATLAB 2017(a) by using BIT map imageor .png image according to software used ,of real life scenario .In this simulation parameters like execution time, path length, learning rate ,error rate is compared for different real maps there are 4 maps of real environment are used in this thesis

SIMULATION PARAMETER

All results on AMD A10-7300, 3.2 GHz with 8GB RAM.

For all results:

source=[50 50]

goal=[450 450]

resolution of BIT map image used: 500×500

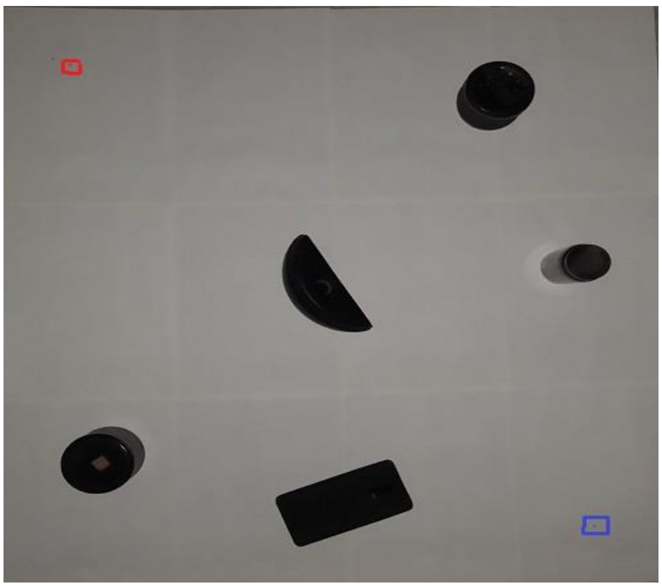


FIG 18 : Real map used for simulation of mobile robot in MATLAB for first 3 navigation technique with red dot as intial robot poistion and blue dot as goal poistion

TABLE 1:SIMULATION RESULT OF POTENTIAL FIELD TECHNIQUE

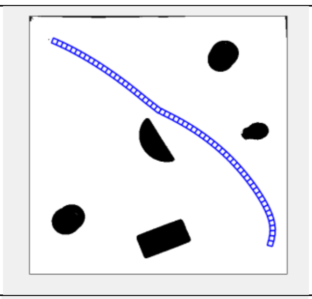
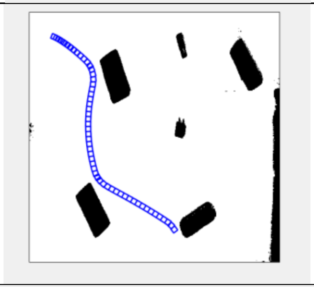
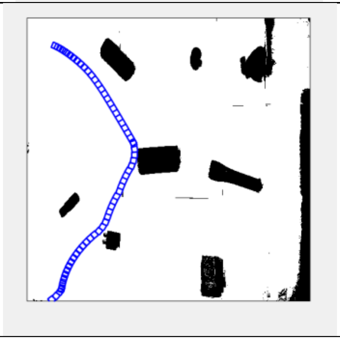
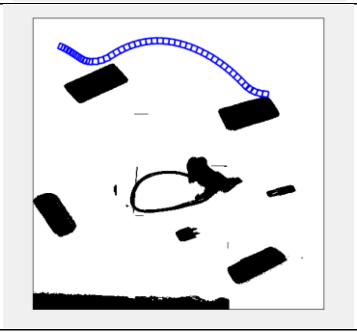
S.NO	MAP	PATH LENGTH	EXECUTION TIME IN SEC
1		808	5.2
2		785	2.67
3		COLLISION	
4		COLLISION	

TABLE 2:SIMULATION RESULT OF FUZZY LOGIC BASED TECHNIQUE

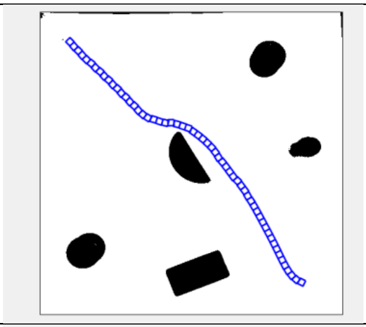
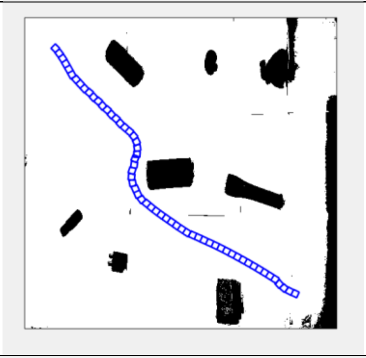
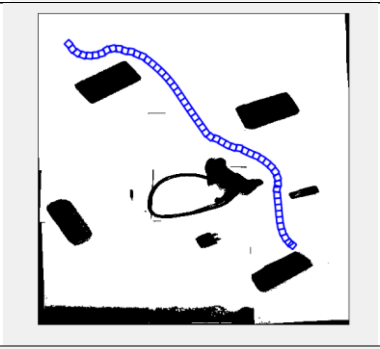
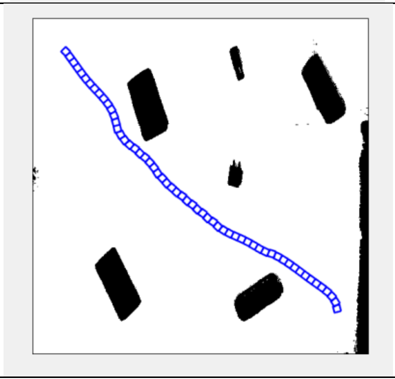
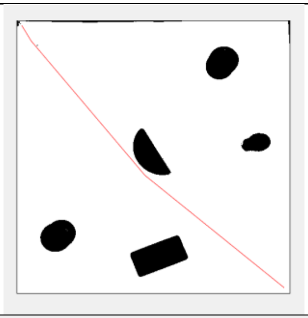
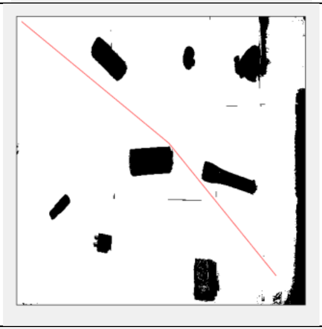
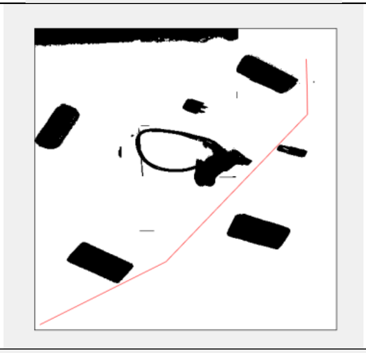
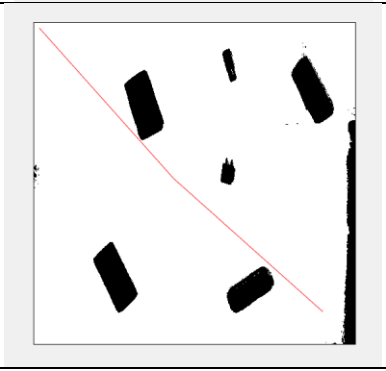
S.NO	MAP	PATH LENGTH	EXECUTION TIME
1		890	6.5
2		785	3.05
3		ROBOT STOPED	
4		777	2.86

TABLE 3:SIMULATION RESULT PROBABLISTIC ROADMAP BASED TECHNIQUE

S.NO	MAP	PATH LENGTH	EXECUTION TIME
1		910	**6.82**
2		**885**	**3.62**
3		**574**	**6.26**
4		**632**	**3.71**

FOR ANN simulation is done using neural network toolbox of matlab 2017 with following conditions

input :180

output :3

Netwok used :feedforward neural network

result :correctlty classfied rate in trainig = 78.56

error in testing =0.2244

CHAPTER 5

CONCLUSION , AND FUTURE SCOPE

CONCLUSION

In this thesis 4 navigation technique has been studied and simulation is done on Matlab 2017 .Ann has been studied only and result has been analyzed from research paper [31].following are the observation of study :

1. artificial potential field is the simplest from all ,but it is very basic it can be only used for static environmentt when using alone .its has one major drawback i.e local minima.
2. fuzzy is better then potential field but only suitable till the condition not come that is not defined in fuzzy. It mostly suitable for know environment ,and has less time and space complexity then PRM.
3. PRM is better then both of previous in many sense it always converges if path is available are available but it has greater time and space complexity ,so it can be used in small indoor environments.
4. ANN is better then all and can better used with noisy sensor data and in dynamic environment also it can be used .but training and testing data generation is main drawback of this technique ,and this data is not general .only can be used for same robot and same environment.

   final conclision , these technique is depend on kind of robot and environment condition,So we should choose navigation technique according to environment condition and type of robot used.

FUTURE SCOPE

There are clearly a number of robust techniques for various key sub-problems in robot navigation. There are also wide variety of techniques which are well developed while not completely robust. However, there is still no known technique or combination of techniques which will result in a robust, generalized performance. The possibility of combining some of the more powerful techniques from each category, to result in a. general technique suitable to a wide variety of environments is still open. It is proposed that research be undertaken to combine of some of these techniques, in an effort to develop general robust navigation system for mobile robot .

APPENDICES A

MATLAB PROGRAM

MATLAB CODE FOR ARTIFICAL POTENTIAL NAVIGATION

```
map=int16(im2bw(imread('map10 (2).bmp'))); % input map read from a bmp file. for new maps write the file name here
source=[50 50]; % source position in Y, X format
goal=[450 450]; % goal position in Y, X format
robotDirection=pi/8; % initial heading direction
robotSize=[10 10]; %length and breadth
robotSpeed=10; % arbitrary units
maxRobotSpeed=10; % arbitrary units
S=10; % safety distance
distanceThreshold=30; % a threshold distace. points within this threshold can be taken as same.
maxAcceleration=10; % maximum speed change per unit time
maxTurn=10*pi/180; % potential outputs to turn are restriect to -60 and 60 degrees.
k=4; % degree of calculating potential
attractivePotentialScaling=300000; % scaling factor for attractive potential
repulsivePotentialScaling=300000; % scaling factor for repulsive potential
minAttractivePotential=0.5; % minimum attractive potential at any point

%%%%% parameters end here %%%%%

currentPosition=source; % position of the centre of the robot
currentDirection=robotDirection; % direction of orientation of the robot
robotHalfDiagonalDistance=((robotSize(1)/2)^2+(robotSize(2)/2)^2)^0.5; % used for distance calculations
pathFound=false; % has goal been reached
pathCost=0;
t=1;
imshow(map==1);
```

```
rectangle('position',[1 1 size(map)-1],'edgecolor','k')
pathLength=0;
if ~plotRobot(currentPosition,currentDirection,map,robotHalfDiagonalDistance)
     error('source lies on an obstacle or outside map');
end
M(t)=getframe;
t=t+1;

if ~feasiblePoint(goal,map), error('goal lies on an obstacle or outside map'); end

tic;
while ~pathFound

    % calculate distance from obstacle at front
    i=robotSize(1)/2+1;
    while true
        x=int16(currentPosition+i*[sin(currentDirection) cos(currentDirection)]);
        if ~feasiblePoint(x,map), break; end
        i=i+1;
    end
    distanceFront=i-robotSize(1)/2; % robotSize(1)/2 distance included in i was inside
the robot body

    % calculate distance from obstacle at left
    i=robotSize(2)/2+1;
    while true
        x=int16(currentPosition+i*[sin(currentDirection-pi/2)        cos(currentDirection-
pi/2)]);
        if ~feasiblePoint(x,map), break; end
        i=i+1;
    end
    distanceLeft=i-robotSize(2)/2;

    % calculate distance from obstacle at right
    i=robotSize(2)/2+1;
    while true
        x=int16(currentPosition+i*[sin(currentDirection+pi/2)
cos(currentDirection+pi/2)]);
        if ~feasiblePoint(x,map), break; end
        i=i+1;
    end
    distanceRight=i-robotSize(2)/2;

    % calculate distance from obstacle at front-left diagonal
    i=robotHalfDiagonalDistance+1;
    while true
        x=int16(currentPosition+i*[sin(currentDirection-pi/4)       cos(currentDirection-
pi/4)]);
        if ~feasiblePoint(x,map), break; end
        i=i+1;
    end
    distanceFrontLeftDiagonal=i-robotHalfDiagonalDistance;

    % calculate distance from obstacle at front-right diagonal
    i=robotHalfDiagonalDistance+1;
    while true
        x=int16(currentPosition+i*[sin(currentDirection+pi/4)
cos(currentDirection+pi/4)]);
        if ~feasiblePoint(x,map), break; end
        i=i+1;
    end
    distanceFrontRightDiagonal=i-robotHalfDiagonalDistance;

    % calculate angle from goal
     angleGoal=atan2(goal(1)-currentPosition(1),goal(2)-currentPosition(2));

     % calculate diatnce from goal
     distanceGoal=( sqrt(sum((currentPosition-goal).^2)) );
     if distanceGoal<distanceThreshold, pathFound=true; end

     % compute potentials
     repulsivePotential=(1.0/distanceFront)^k*[sin(currentDirection)
cos(currentDirection)] + ...
     (1.0/distanceLeft)^k*[sin(currentDirection-pi/2) cos(currentDirection-pi/2)] + ...
     (1.0/distanceRight)^k*[sin(currentDirection+pi/2) cos(currentDirection+pi/2)] + ...
     (1.0/distanceFrontLeftDiagonal)^k*[sin(currentDirection-pi/4)
cos(currentDirection-pi/4)] + ...
     (1.0/distanceFrontRightDiagonal)^k*[sin(currentDirection+pi/4)
cos(currentDirection+pi/4)];

     attractivePotential=max([(1.0/distanceGoal)^k*attractivePotentialScaling
minAttractivePotential])*[sin(angleGoal) cos(angleGoal)];
     totalPotential=attractivePotential-repulsivePotentialScaling*repulsivePotential;

     % perform steer

preferredSteer=atan2(robotSpeed*sin(currentDirection)+totalPotential(1),robotSpeed
*cos(currentDirection)+totalPotential(2))-currentDirection;
     while preferredSteer>pi, preferredSteer=preferredSteer-2*pi; end % check to get
the angle between -pi and pi
     while preferredSteer<-pi, preferredSteer=preferredSteer+2*pi; end % check to get
the angle between -pi and pi
     preferredSteer=min([maxTurn preferredSteer]);
     preferredSteer=max([-maxTurn preferredSteer]);
     currentDirection=currentDirection+preferredSteer;

     % setting the speed based on vehicle acceleration and speed limits. the vehicle
cannot move backwards.
     preferredSpeed=sqrt(sum((robotSpeed*[sin(currentDirection)
cos(currentDirection)] + totalPotential).^2));
     preferredSpeed=min([robotSpeed+maxAcceleration preferredSpeed]);
     robotSpeed=max([robotSpeed-maxAcceleration preferredSpeed]);
     robotSpeed=min([robotSpeed maxRobotSpeed]);
     robotSpeed=max([robotSpeed 0]);

     if robotSpeed==0, error('robot had to stop to avoid collission'); end

     % calculating new position based on steer and speed
     newPosition=currentPosition+robotSpeed*[sin(currentDirection)
cos(currentDirection)];
     pathCost=pathCost+distanceCost(newPosition,currentPosition);
     currentPosition=newPosition;
     if ~feasiblePoint(int16(currentPosition),map), error('collission recorded'); end

     % plotting robot
     if ~plotRobot(currentPosition,currentDirection,map,robotHalfDiagonalDistance)
       error('collission recorded');
     end
     M(t)=getframe;t=t+1;
end
fprintf('processing time=%d \nPath Length=%d \n\n', toc,pathCost);
```

MATLAB CODE FOR FUZZY RULE BASED NAVIGATION

```
map=int16(im2bw(imread('map6.bmp'))); % input map read from a bmp file. for new maps write the file name here
source=[50 50]; % source position in Y, X format
goal=[450 450]; % goal position in Y, X format
robotDirection=pi/4; % initial heading direction
robotSize=[10 10]; %length and breadth
robotSpeed=10; % arbitrary units
maxRobotSpeed=10; % arbitrary units
S=10; % safety distance
distanceThreshold=30; % a threshold distace. points within this threshold can be taken as same.
maxAcceleration=10; % maximum speed change per unit time
directionScaling=60*pi/180; % fuzzy outputs to turn are restriect to -1 and 1. these are magnified here. maximum turn can be 60 degrees

%%%%% parameters end here %%%%%

fuz=readfis('fuzzyBase.fis'); % fuzzy inference system used. to read/edit use fuzzy(readfis('fuzzyBase.fis')) at the command line
distanceScaling=(size(map,1)^2+size(map,2)^2)^0.5; % all inputs are scaled by this number so that all distance inputs are between 0 and 1. maximum distance can be distanceScaling
currentPosition=source; % position of the centre of the robot
currentDirection=robotDirection; % direction of orientation of the robot
robotHalfDiagonalDistance=((robotSize(1)/2)^2+(robotSize(2)/2)^2)^0.5; % used for distance calculations
pathFound=false; % has goal been reached
prevTurn=0; % preffered turn at the previous time step, used for turning heuristic, see variable turn being set below.
prevDistanceLeftDiagonal=distanceScaling; % diagonal distance at the previous time step, used for tracking obstacles, used for turning heuristic, see variable turn being set below.
```

```
prevDistanceRightDiagonal=distanceScaling; % diagonal distance at the previous time step, used for tracking obstacles, used for turning heuristic, see variable turn being set below.
pathCost=0;
t=1;
imshow(map==1);
rectangle('position',[1 1 size(map)-1],'edgecolor','k');
pathLength=0;
if ~plotRobot(currentPosition,currentDirection,map,robotHalfDiagonalDistance)
    error('source lies on an obstacle or outside map');
end
M(t)=getframe;
t=t+1;

if ~feasiblePoint(goal,map), error('goal lies on an obstacle or outside map'); end

tic;
while ~pathFound

   % calculate distance from obstacle at front
   for i=robotSize(1)/2+1:distanceScaling
      x=int16(currentPosition+i*[sin(currentDirection) cos(currentDirection)]);
      if ~feasiblePoint(x,map), break; end
   end
   distanceFront=(i-robotSize(1)/2)/distanceScaling; % robotSize(1)/2 distance included in i was inside the robot body

   % calculate distance from obstacle at front-left diagonal
   for i=robotHalfDiagonalDistance+1:distanceScaling
d         x=int16(currentPosition+i*[sin(currentDirection-pi/4) cos(currentDirection-pi/4)]);
      if ~feasiblePoint(x,map), break; end
   end
   distanceFrontLeftDiagonal=(i-robotHalfDiagonalDistance)/distanceScaling;
```

```
    % calculate distance from obstacle at front-right diagonal
    for i=robotHalfDiagonalDistance+1:distanceScaling
        x=int16(currentPosition+i*[sin(currentDirection+pi/4)
cos(currentDirection+pi/4)]);
        if ~feasiblePoint(x,map), break; end
    end
    distanceFrontRightDiagonal=(i-robotHalfDiagonalDistance)/distanceScaling;

    % calculate angle deviation to goal
     slopeGoal=atan2(goal(1)-currentPosition(1),goal(2)-currentPosition(2));
     angleGoal=slopeGoal-currentDirection;
     while angleGoal>pi, angleGoal=angleGoal-2*pi; end % check to get the angle
between -pi and pi
     while angleGoal<-pi, angleGoal=angleGoal+2*pi; end % check to get the angle
between -pi and pi
     angleGoal=angleGoal/pi; % re-scaling the angle as per fuzzy modelling

     % calculate diatnce from goal
     distanceGoal=( sqrt(sum((currentPosition-goal).^2)) )/distanceScaling;
     if distanceGoal*distanceScaling<distanceThreshold, pathFound=true; end

     % calculate preferred turn.
     % this indicates, if the front obstacle is far away, turn so as to more face the goal
     % if the front obstacle is close and a new front obstacle is encountered, turn using
the side of the goal is preferred
     % if the front obstacle is close and the same obstacle as encountered in the
previous step is found, same turn is made
     if (prevTurn==0 || prevTurn==1) && distanceFront<0.1 &&
(distanceFrontLeftDiagonal-
prevDistanceLeftDiagonal)*distanceScaling<maxRobotSpeed, turn=1;
     elseif prevTurn==-1 && distanceFront<0.1 && (distanceFrontRightDiagonal-
prevDistanceRightDiagonal)*distanceScaling<maxRobotSpeed, turn=-1;
     else turn=(angleGoal>=0)*1+(angleGoal<0)*(-1);prevTurn=turn;
     end
     prevDistanceLeftDiagonal=distanceFrontRightDiagonal;
     prevDistanceRightDiagonal=distanceFrontLeftDiagonal;

     % pass all computed inputs to a fuzzy inference system
     computedSteer=evalfis([distanceFront                    distanceFrontLeftDiagonal
distanceFrontRightDiagonal angleGoal turn distanceGoal],fuz);
     currentDirection=currentDirection+computedSteer*directionScaling;

     % speed is set based on the front and diagonal distance so as not to make the robot
collide, but make it slow and even stop before possible collission
     % distances here include additional safety distance of S
     distanceFrontSafety=max([distanceFront*distanceScaling-S 0]);

distanceFrontLeftDiagonalSafety=max([distanceFrontLeftDiagonal*distanceScaling-
S 0]);

distanceFrontRightDiagonalSafety=max([distanceFrontRightDiagonal*distanceScalin
g-S 0]);

     % maximum speeds admissible as per the above safety distance
     maxSpeed1=min([sqrt(2*maxAcceleration*distanceFrontSafety)
maxRobotSpeed]);
     maxSpeed2=min([sqrt(maxAcceleration*distanceFrontLeftDiagonalSafety)
maxRobotSpeed]);
     maxSpeed3=min([sqrt(maxAcceleration*distanceFrontRightDiagonalSafety)
maxRobotSpeed]);
     maxSpeed=min([maxSpeed1 maxSpeed2 maxSpeed3]);

     % setting the speed based on vehicle acceleration and speed limits. the vehicle
cannot move backwards.
     preferredSpeed=min([robotSpeed+maxAcceleration maxSpeed]);
     robotSpeed=max([robotSpeed-maxAcceleration preferredSpeed]);
     robotSpeed=min([robotSpeed maxRobotSpeed]);
     robotSpeed=max([robotSpeed 0]);

     if robotSpeed==0, error('robot had to stop to avoid collission'); end

     % calculating new position based on steer and speed
     newPosition=currentPosition+robotSpeed*[sin(currentDirection)
cos(currentDirection)];
     pathCost=pathCost+distanceCost(newPosition,currentPosition);
     currentPosition=newPosition;
     if ~feasiblePoint(int16(currentPosition),map), error('collission recorded'); end

     % plotting robot
     if ~plotRobot(currentPosition,currentDirection,map,robotHalfDiagonalDistance)
       error('collission recorded');
     end
     M(t)=getframe;t=t+1;
end
fprintf('processing time=%d \nPath Length=%d \n\n', toc,pathCost);
```

PRM MATLAB PROGRAM

```
map=im2bw(imread('MAP E-crop.bmp')); % input map read from a bmp file. for new maps write the file name here
source=[10 10]; % source position in Y, X format
goal=[450 450]; % goal position in Y, X format
k=50; % number of points in the PRM
display=true; % display processing of nodes

%%%%% parameters end here %%%%%

if ~feasiblePoint(source,map), error('source lies on an obstacle or outside map'); end
if ~feasiblePoint(goal,map), error('goal lies on an obstacle or outside map'); end

imshow(map);
rectangle('position',[1 1 size(map)-1],'edgecolor','k')
vertex=[source;goal]; % source and goal taken as additional vertices in the path planning to ease planning of the robot
if display, rectangle('Position',[vertex(1,2)-5,vertex(1,1)-5,10,10],'Curvature',[1,1],'FaceColor','r'); end
if display, rectangle('Position',[vertex(2,2)-5,vertex(2,1)-5,10,10],'Curvature',[1,1],'FaceColor','r'); end
tic;
while length(vertex)<k+2 % iteratively add vertices
x=double(int32(rand(1,2) .* size(map)));
if feasiblePoint(x,map),
vertex=[vertex;x];
if display, rectangle('Position',[x(2)-5,x(1)-5,10,10],'Curvature',[1,1],'FaceColor','r');
end
end
end
if display
disp('click/press any key');
waitforbuttonpress;
```

```
end
edges=cell(k+2,1); % edges to be stored as an adjacency list
for i=1:k+2
for j=i+1:k+2
if checkPath(vertex(i,:),vertex(j,:),map);
edges{i}=[edges{i};j];edges{j}=[edges{j};i];
if display, line([vertex(i,2);vertex(j,2)],[vertex(i,1);vertex(j,1)]); end
end
end
end
if display
disp('click/press any key');
waitforbuttonpress;
end

%structure of a node is taken as index of node in vertex, historic cost, heuristic cost,
total cost, parent index in closed list (-1 for source)
Q=[1 0 heuristic(vertex(1,:),goal) 0+heuristic(vertex(1,:),goal) -1]; % the processing
queue of A* algorihtm, open list
closed=[]; % the closed list taken as a list
pathFound=false;
while size(Q,1)>0
[A, I]=min(Q,[],1);
n=Q(I(4),:); % smallest cost element to process
Q=[Q(1:I(4)-1,:);Q(I(4)+1:end,:)]; % delete element under processing
if n(1)==2 % goal test
pathFound=true;break;
end
for mv=1:length(edges{n(1),1}) %iterate through all edges from the node
newVertex=edges{n(1),1}(mv);
if length(closed)==0 || length(find(closed(:,1)==newVertex))==0 % not already in
closed
historicCost=n(2)+historic(vertex(n(1),:),vertex(newVertex,:));
heuristicCost=heuristic(vertex(newVertex,:),goal);
```

```
totalCost=historicCost+heuristicCost;
add=true; % not already in queue with better cost
if length(find(Q(:,1)==newVertex))>=1
I=find(Q(:,1)==newVertex);
if Q(I,4)<totalCost, add=false;
else Q=[Q(1:I-1,:);Q(I+1:end,:);];add=true;
end
end
if add
Q=[Q;newVertex historicCost heuristicCost totalCost size(closed,1)+1]; % add new
nodes in queue
end
end
end
closed=[closed;n]; % update closed lists
end
if ~pathFound
error('no path found')
end

fprintf('processing time=%d \nPath Length=%d \n\n', toc,n(4));
path=[vertex(n(1),:)]; %retrieve path from parent information
prev=n(5);
while prev>0
path=[vertex(closed(prev,1),:);path];
prev=closed(prev,5);
end

imshow(map);
rectangle('position',[1 1 size(map)-1],'edgecolor','k')
line(path(:,2),path(:,1),'color','r');
```

NEURAL NETWOK CODE

```
%getting laser data values

l=dlmread('D:\4th sem\path_planning\dataset_intel.tar\intel_LASER_.txt');

%getting odometry data values

o=dlmread('D:\4th sem\path_planning\dataset_intel.tar\intel_ODO.txt');

%training the data

TrainData = l(1:2:end,:);

%testing the data

TestData = l(2:2:end,:);

%training the target

TrainTarget = o(1:2:end,:);

%testing the target

TestTarget = o(2:2:end,:);



TrainData=(mapminmax(TrainData'))';

% target_test = uint8(target_test);

TestData=(mapminmax(TestData'))';



% setdemorandstream(pi);



net = feedforwardnet([340,120,350], 'traingdx');



net.performFcn = 'mse';
net.trainParam.goal=1e-2;

net.trainParam.min_grad=1e-50;

net.trainParam.epochs=1000;

net.trainParam.mc=0.95;

net.trainParam.mu=0.01;


% net.trainParam.mu_max=1e20;




TrainData = TrainData';

TestData = TestData';


TrainTarget = TrainTarget';

TestTarget = TestTarget';


net = train(net,TrainData,TrainTarget);


% Train_error_rate

err_tr=0;

for p=1:size(TrainData,2)

    y = sim(net, TrainData(:, p));

    actual_class = find(y==max(y));

    desired_class = find(TrainTarget(:,p)==max(TrainTarget(:,p)));
    if (desired_class~=actual_class)

        err_tr=err_tr + 1;

    end

end

err_rate_tr=err_tr/(size(TrainData,2))

Correctly_classified_rate_tr=100-(err_rate_tr*100)


% Test_error_rate

err_tst=0;

for p=1:size(TestData,2)

    y = sim(net, TestData(:, p));

    actual_class = find(y==max(y));

    desired_class = find(TestTarget(:,p)==max(TestTarget(:,p)));

    if (desired_class~=actual_class)

        err_tst = err_tst + 1;

    end

end

err_rate_tst=err_tst/size(TestData,2)

Correctly_classified_rate_tst=100-(err_rate_tst*100)
```

APPENDICES B

ARUDINO PROGRAM FOR ROBOT CONTROL

PROGRAM FOR ULTRASONIC

```
#include <Servo.h>



Servo myservo;  // create servo object to control a servo



int pos = 0;    // variable to store the servo position

int v1 = 0;

int v2 = 0;

int v3 = 0;

int v4 = 0;

int v5 = 0;



float duration, distance;



#define trigPin 13 //Sensor Echo pin connected to Arduino pin 13

#define echoPin 12 //Sensor Trip pin connected to Arduino pin 12





void setup() {

pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);


myservo.attach(8);  // attaches the servo on pin 9 to the servo object

Serial.begin(9600);


}


void loop() {




for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

myservo.write(pos);              // tell servo to go to position in variable 'pos'


Serial.println(pos);


digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) / 29.1;
```

```
Serial.println(distance);


if(pos==0&&distance<10.00)

{ v1=1;

}

if(pos==45&&distance<10.00)

{ v2=1;

}

if(pos==90&&distance<10.00)

{ v3=1;

}

if(pos==135&&distance<10.00)

{ v4=1;

}

if(pos==180&&distance<10.00)

{ v5=1;

}


Serial.println(v1);

Serial.println(v2);

Serial.println(v3);

Serial.println(v4);

Serial.println(v5);
```

```
delay(15);

// waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

myservo.write(pos);              // tell servo to go to position in variable 'pos'

//  Serial.println("angle");

Serial.println(pos);

//Defining both the parameters in float

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) / 29.1;

Serial.println(distance);



if(pos==0&&distance<10.00)

{ v1=1;

}

if(pos==45&&distance<10.00)

{ v2=1;
}

if(pos==90&&distance<10.00)

{ v3=1;

}

if(pos==135&&distance<10.00)

{ v4=1;

}

if(pos==180&&distance<10.00)

{ v5=1;

}


Serial.println(v1);

Serial.println(v2);

Serial.println(v3);

Serial.println(v4);

Serial.println(v5);


delay(15);

}


}
```

PROGRAM FOR BLUETOOTH CONTROL

```
#include <SoftwareSerial.h>


SoftwareSerial BT(0, 1); //TX, RX respetively

String readdata;


void setup() {

 BT.begin(9600);

 Serial.begin(9600);

  pinMode(3, OUTPUT);

  pinMode(7, OUTPUT);

  pinMode(5, OUTPUT);

  pinMode(6, OUTPUT);


}

//-----------------------------------------------------------------------//

void loop() {


  Serial.println(analogRead(A0));

            delay(100);

Serial.println(analogRead(A1));

            delay(100);

  while (BT.available()){  //Check if there is an available byte to read
```

```
delay(10); //Delay added to make thing stable

char c = BT.read(); //Conduct a serial read

readdata += c; //build the string- "forward", "reverse", "left" and "right"

}

if (readdata.length() > 0) {

  Serial.println(readdata);


if(readdata == "1")

{

  digitalWrite(3, HIGH);

  digitalWrite (7, HIGH);

  digitalWrite(5,LOW);

  digitalWrite(6,LOW);

  delay(100);

}


else if(readdata == "2")

{

  digitalWrite(3, LOW);

  digitalWrite(7, LOW);

  digitalWrite(5, HIGH);

  digitalWrite(6,HIGH);

  delay(100);
```

```
 }


 else if (readdata == "3")
 {
   digitalWrite (3,HIGH);
   digitalWrite (7,LOW);
   digitalWrite (5,LOW);
   digitalWrite (6,LOW);
   delay (100);

 }


else if ( readdata == "4")
{
  digitalWrite (3, LOW);
  digitalWrite (7, HIGH);
  digitalWrite (5, LOW);
  digitalWrite (6, LOW);
  delay (100);
}


else if (readdata == "5")
{
  digitalWrite (3, LOW);

  digitalWrite (7, LOW);

  digitalWrite (5, LOW);

  digitalWrite (6, LOW);

  delay (100);

}
```

BIBLIOGRAPHY

[1] Daniele De Gregorio and Luigi Di Stefano," SkiMap: An Efficient Mapping Framework for Robot Navigation" . IEEE International Conference on Robotics and Automation (ICRA) Singapore, May 29 - June 3, 2017.

[2] M. Yousef Ibrahim, Allwyn Fernmdes," Study on Mobile Robot Navigation Techniques". IEEE International Conference on Industrial Technology (ICIT).2004.

[3] Jayasree K R, Jayasree P R, Vivek A," Dynamic target tracking using a four wheeled mobile robot with optimal path planning technique" International Conference on circuits Power and Computing Technologies [ICCPCT,.2017.

[4] Aykut O¨ zdemir, Volkan Sezer," A Hybrid Obstacle Avoidance Method: Follow the gap with dynamic window approach". First IEEE International Conference on Robotic Computing,2017.

[5] Matthew Klingensmith, Siddartha S. Sirinivasa, and Michael Kaess," Articulated Robot Motion for Simultaneous Localization and Mapping (ARM-SLAM)", IEEE robotics and automation letters, vol. 1, no. 2, july, 2016.

[6] Akshay A. Mane, Mahesh N. Parihar, Sharad P. Jadhav, Bhavesh B. Digey," Robotics Based Simultaneous Localization And Mapping of an Unknown Environment using Kalman Filtering". 5th Nirma University International Conference on Engineering,2015.

[7] Lijun Zhao and Guanglei Huo Ke Wang and Ruifeng Li," A Multi-feature Localization Algorithm for Mobile Robots Indoor Environment Mapping", Proceedings of 2016 IEEE International Conference on Mechatronics and Automation, Harbin, China,2010.

[8] Jin Cheng, Bing Wang, Yuan Xu, Ke Wang," Construction Method of Line-Segments Based Map From 2D Laser Sensor Data for Mobile Robot", Proceedings of the 36th Chinese Control Conference, Dalian, China, July 26-28, 2017.

[9] SONG Jianchao, ZHANG Xuebo, SUN Lei, LIU Jingtai," Map-based Robust Localization for Indoor Mobile Robots", Proceedings of the 36th Chinese Control Conference, Dalian, China, July 26-28, 2017.

[10] J.O. Wallgrün, Hierarchical Voronoi Graphs," Spatial Representation and Reasoning for Mobile Robots", DOI 10.1007/978-3-642-10345-2_2, Springer-Verlag Berlin Heidelberg, 2010.

[11] J.Parthasarathy," positioning and navigation system using gps" International Archives of the Photogrammetric, Remote Sensing and Spatial Information Science, Tokyo Japan 2006.

[12] Rainer K¨ummerle, Michael Ruhnke," Autonomous Robot Navigation in Highly Populated Pedestrian Zones", published in final form at http://dx.doi.org/10.1002/rob.21534.

[13] Wolfram Burgard," Probabilistic Approaches to Robot Navigation", Digital Object Identifier 10.1109/MRA.2008.925681, IEEE Robotics & Automation Magazine, JUNE 2008.

[14] Velappa Ganapathy, Member, IEEE, Soh Chin Yun, Student Member, IEEE and Jefry Ng," Fuzzy and Neural Controllers for Acute Obstacle Avoidance inMobile Robot Navigation", 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics Suntec Convention and Exhibition Center Singapore, July 14-17, 2009.

[15] Bassel Abou Merhy, Pierre Payeur, Member, IEEE, and Emil M. Petriu, Fellow, IEEE," Application of Segmented 2-D Probabilistic Occupancy Maps for Robot Sensing and Navigation ", IEEE transactions on instrumentation and measurement, vol. 57, no. 12, december 2008.

[16] Juan Carlos Vega Oliver - Graduate, Pedro Freddy Huamaní Navarrete ," Fuzzy Control to Simulate 4 Autonomous Navigation Behaviors in a Differential-Drive Mobile Robot", 978-1-5386-3279-6/17/$31.00 ©2017 IEEE.

[17] Robotics toolbox ,for MATLAB, http://www.cat.csiro.au/cmst/staff/pic/robot.

[18] Prof. A. G. Andurkar, Ms. Rupali Tankar, Ms. Suvarna Patil," Path Navigation for Robot Using Matlab", International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 4 .

[19] Razif Rashid, I. Elamvazuthi, Mumtaj Begam, M. Arrofiq," Fuzzy-based Navigation and Control of a Non-Holonomic Mobile Robot", journal of computing, volume 2, issue 3, , issn 2151-9617, march 2010.

[20] Valeri Kroumov1 and Jianli Yu," Neural Networks Based Path Planning and Navigation of Mobile Robots". *www.intechopen.com.*

[21] J. Borenstein and Y. Koren. The Vector Field Histogram–Fast Obstacle Avoidance for Mobile Robots. IEEE Transactions on Robotics and Automation, 7:278–288, 1991.

[22] "Reactive Obstacle Avoidance for Mobile Robots that Operate in Confined 3D Workspaces",5th Nirma University International Conference on Engineering,2015.

[23] Hugh durrant-whyte and tim bailey," Simultaneous Localization and Mapping: Part I", , IEEE Robotics & Automation Magazine, 1070-9932/06/$20.00©2006 .

[24] István Engedy, Gábor Horváth," Artificial Neural Network based Mobile Robot Navigation",6th IEEE International Symposium on Intelligent Signal Processing • 26–28 August, 2009 Budapest, Hungary.

[25] Anish Pandeya, Saroj Kumarb, Krishna Kant Pandeya,Dayal R," Mobile robot navigation in unknown static environments using ANFIS controller", 29 January 2016; accepted 9 April 2016Available online 29 April 2016.Perspectives in Science (2016) **8**, pp-421—423.

[26] Arindam Singha, Anjan Kumar Ray, Arun Baran Samaddar," Navigation of Mobile Robot in a Grid-based Environment using Local and Target Weighted Neural Networks"8th ICCCNT ,IIT Delhi, India, July 3 - 5, 2017.

[27] Danica Janglová,"Neural Networks in Mobile Robot Motion", pp. 15-22, Inernational Journal of Advanced Robotic Systems, Volume 1 Number 1 2004.

[28] Rui Araújo," Prune-Able Fuzzy ART Neural Architecture for Robot Map Learning and Navigation in Dynamic Environments", IEEE transactions on neural networks, vol. 17, no. 5, september 2006.

[29] Asako Kanezaki, Jirou Nitta, and Yoko Sasaki," GOSELO: Goal-Directed Obstacle and Self-Location Map for Robot Navigation Using Reactive Neural Networks", IEEE Robotics and automation letters, vol. 3, no. 2, april 2018.

[30] Dezfoulian, Seyyed Hamid, "A Generalized Neural Network Approach to Mobile Robot Navigation and Obstacle Avoidance",https://scholar.uwindsor.ca/etd/102/,2012

[31] G. N. Tripathi and V.Rihani," motion planning of an autonomous mobile robot using artificial neural network", WISP ,. IEEE International Symposium,2009.

[32] Farhad shamsfakhr, bahram sadeghibigham," A neural network approach to navigation of a mobile robot and obstacle avoidance in dynamic and unknown environments", doi:10.3906/elk-1603-75, Turkish Journal of Electrical Engineering & Computer Sciences,2017.

[33] Introduction to autonomous mobile robots by Roland Siegwart, illah R. nourbakash.

[34] Autonomous robots modeling ,path planning, and Control by Farbod Fahimi.

[35] Fuzzy sets and fuzzy logic Theory and application by George J. Klir and Bo Yuan.

[36]] L. Tang, S. Dian, G. Gu, K. Zhou, S. Wang and X. Feng, "A Novel Potential Field Method for Obstacle A voidance and Path Planning of Mobile Robot," in 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, 2010.

[37] John Paolo C. Tuazonl , Ken Gilfed v. Prado , Neil John A. Cabia , Reeann L. Enriquez4, Francesca Louise c. Rivera, and Kanny Krizzy D. Serrano," An Improved Collision A voidance Scheme using Artificial Potential Field with Fuzzy Logic",IEEE Region 10 Conference (TENCON) - Proceedings ofthe International Conference,2016.

[38] Ronald J. Williams, David Zipser," A Learning Algorithm for Continually Running Fully Recurrent Neural Networks",Neural Computation 1,pp. 270-280 Massachusetts Institute of Technology. 1989.

[39] PI-Kyun sung, Ki-Bum Hong, Suk-Kyo Hong ,Soon-Chon Hong,"Path planning of mobile robot using neural network".IEEE 0-7803-5662-4/99/$10.00 01999 .

[40] W. Hess, D. Kohler, H. Rapp, D. Andor," Real-Time Loop Closure in 2D LIDAR SLAM", IEEE International Conference on Robotics and Automation, 2016, pp. 1271-1278.

[41]Grisetti G, Stachniss C, Burgard W," Improved techniques for grid mapping with rao-blackwellized particle filters", IEEE transactions on Robotics, 2007, 23(1): 34-46.

[42] Kohlbrecher S, Von Stryk O, Meyer J, et al." A flexible and scalable slam system with full 3d motion estimation, Safety, Security, and Rescue Robotics", 2011 IEEE International Symposium on. IEEE, 2011.

[43] BurgardW, Cremers A B, Fox D, et al, "The interactive museum tour-guide robot", Aaai/iaai. 1998: 11-18.

[44] Marder-Eppstein E, Berger E, Foote T, " The office marathon: Robust navigation in an indoor office environment", IEEE International Conference on Robotics and Automation, 2010 IEEE International Conference on. IEEE, 2010

[45] Hajer Omrane , Mohamed Slim Masmoudi and Mohamed Masmoudi," Intelligent Mobile Robot Navigation", International Conference on Smart, Monitored and Controlled Cities (SM2C), Kerkennah, Tunisia, 2017.

[46] MATLAB2017 help

[47]Programming in MATLAB, a problem solving approach byRam n.patel,ankush mittal.