

# **HEMOCARE HUMAN TRACKING ROBOT USING POMDP TECHNIQUE**



## **Author**

Muhammad Armaghan Akhtar  
NUST 2012 61238 MCEME 35512F

## **Supervisor**

Dr. Kunwar Faraz Ahmed

DEPARTMENT OF MECHATRONICS ENGINEERING  
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY  
RAWALPINDI

JUNE, 2016

# **HEMOCARE HUMAN TRACKING ROBOT USING POMDP TECHNIQUE**

## **Author**

Muhammad Armaghan Akhtar  
NUST 2012 61238 MCEME 35512F

A thesis submitted in partial fulfillment of the requirements for the degree of  
MS Mechatronics Engineering

## **Thesis Supervisor:**

Dr. Kunwar Faraz Ahmed

Thesis Supervisor's

Signature: \_\_\_\_\_

DEPARTMENT OF MECHATRONICS ENGINEERING  
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,  
RAWALPINDI

JUNE, 2016

## **Declaration**

I certify that this research work titled “*Homecare Human Tracking Robot using POMDP technique*” is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Muhammad Armaghan Akhtar

NUST 2012 61238 MCEME 35512F

## **Language Correctness Certificate**

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

Muhammad Armaghan Akhtar

NUST 2012 61238 MCEME 35512F

Signature of Supervisor

## **Copyright Statement**

- Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST College of E&ME. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the author.
- The ownership of any intellectual property rights which may be described in this thesis is vested in NUST College of E&ME, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the College of E&ME, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST College of E&ME, Rawalpindi.

## **Acknowledgements**

All the praise for Almighty “Allah” who bestowed me the opportunity and potential to make contribution in already existing ocean of knowledge. I firmly believe that “Allah” will never spoil my effort. Every piece of work is rewarded according to the nature and degree of devotion put in.

My complete love for Hazrat Muhammad (Peace Be Upon Him) who has given us a code of life and may each and every moment of my life devoted for his praise.

Thesis usually falls short of its expectation unless it is aided and guided by the right person at the right time. I humbly state that this thesis is not entirely a fruit of my individual effort but of a number of persons, who have guided me throughout the span of this research.

Sincere thanks and Special appreciation goes to my supervisor, Dr. Kunwar Faraz, for his supervision and constant support. His keen academic supervision, valuable constructive suggestions, constructive comments, kind cooperation, support and inspiration in planning and execution of the research throughout the academic and thesis works have contributed to the success of this project.

I immensely extend my thanks to Dr. Khurram Kamal, Dr. Umar Shahbaz and Dr. Adnan Masood for being on my thesis guidance and evaluation committee.

I would also like to pay special thanks to my mother Iram Akhtar, father Muhammad Akhtar, wife Sidra Yaseen, first sister Sundus Akhtar and 2<sup>nd</sup> sister Kanza Akhtar for her tremendous support and cooperation. Each time I got stuck in something, she came up with the solution. Without her help I wouldn't have been able to complete my thesis. I appreciate her patience and guidance throughout the whole thesis.

Last but not least I do have no words at command to adequately offer my great fullness to my adoring parents and my family members for their constant support endless love, prayers whose hands always raise for me to achieve higher goals of life.

I record my indebtedness & heartfelt gratitude to my family members for their constant support & encouragement.

*Dedicated to my exceptional parents and adored siblings whose  
tremendous support and cooperation led me to this wonderful  
accomplishment.*

## Abstract

Robots are progressively becoming more and more useful in bringing comfort to human life. One of the important new roles of robots is caring for elderly people who want to stay at home due to physical or cognitive difficulties. Navigation algorithm of a robot is a basic constituent that is to be programmed for its desired motion. Autonomous navigation of robots is tough to be planned in uncertain and dynamic environment. It becomes a complicated task to navigate a robot when the human's movement is uncertain. Partially Observable Markov Decision Process (POMDP) is one of the techniques used in navigation of robots in uncertain environment. POMDP is a complicated technique that requires more processing and computational power. In this work a Localized POMDP technique is introduced. This technique will enable robot to reduce computational power and allow it to calculate results in lesser time. Furthermore, in this work a robot learning phase is defined. In this phase a robot learns the positions of human with respect to time and creates a probability distribution function that help the robot to navigate human more accurately. An algorithm in MATLAB software with simulation is developed for its implementation and results.

**Key Words:** *Partially Observable Markov Decision Process, Probability Distribution Function, Robot Navigation, Path Planning, Motion Planning of Mobile Robot*



## Table of Contents

<b>Declaration.....</b>	<b>i</b>
<b>Language Correctness Certificate .....</b>	<b>ii</b>
<b>Copyright Statement.....</b>	<b>iii</b>
<b>Abstract.....</b>	<b>vi</b>
<b>Table of Contents .....</b>	<b>vii</b>
<b>List of Figures.....</b>	<b>ix</b>
<b>List of Tables .....</b>	<b>xi</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>2</b>
<b>CHAPTER 2: LITERATURE REVIEW .....</b>	<b>8</b>
2.1 Potential functions.....	8
2.2 Visibility Graph.....	11
2.3 Cell decomposition method.....	12
2.4 Sampling based algorithm.....	13
2.5 Robot Navigation with Bayesian Filtering.....	15
2.6 Markov Decision Process.....	17
2.7 Partially Observable Markov Decision Process (POMDP).....	19
<b>CHAPTER 3: METHODOLOGY .....</b>	<b>21</b>
3.1 Map initialization .....	21
3.2 Defining Target path .....	22
3.3 Path segregation for localization .....	23
3.4 Defining Policy Tree .....	23
3.5 Dealing with uncertainty while moving .....	26
3.6 Localization to optimize the movement of robot .....	27
3.7 Call interrupt by Human.....	28

<b>CHAPTER 4: IMPLEMENTATION &amp; EXPERIMENTAL RESULTS .....</b>	<b>32</b>
4.1 Computing Belief Space.....	35
4.2 Defining Localization, Optimum Policy and Value Function .....	36
4.3 Comparison of Simple POMDP & Localized-POMDP .....	42
<b>CHAPTER 5: CONCLUSION &amp; FUTURE RECOMMENDATIONS.....</b>	<b>43</b>
<b>APPENDIX A - MATLAB CODE.....</b>	<b>45</b>
<b>REFERENCES.....</b>	<b>54</b>

## List of Figures

Figure 1: Robot Control System .....	2
Figure 2: Robot Arm for Welding (Complex Tasks).....	3
Figure 3: Assembly Line Robots (Repetitive Tasks).....	3
Figure 4: Robot Navigation .....	4
Figure 5: Possible Robot Moves .....	5
Figure 6: Potential Field Example .....	9
Figure 7: Parabolic Function.....	9
Figure 8: Repulsive graph.....	10
Figure 9: Sum of Potentials.....	11
Figure 10: Visibility Graph Navigation .....	11
Figure 11: Voronic Diagram .....	12
Figure 12: Cell Decomposition Method .....	12
Figure 13: Path formed by cell decomposition method .....	13
Figure 14: Random Samples .....	13
Figure 15: Sample Selection .....	14
Figure 16: Straight Path linkage .....	14
Figure 17: Collision Free Links .....	14
Figure 18: Start to Goal Path .....	15
Figure 19: Probability Map based Localization.....	15
Figure 20: Robot One dimension Location.....	16
Figure 21: Expected location .....	16
Figure 22: Robot Exact Location.....	16
Figure 23: Basic Navigation Flow Chart .....	17
Figure 24: Example of MDP.....	18
Figure 25: Finite Horizon (POMDP) .....	19
Figure 26: POMDP Structure.....	20
Figure 27: Initial MAP.....	22
Figure 28: Target Path and Rooms .....	22
Figure 29: Robot Map.....	23
Figure 30: Policy Tree (Human Tracking Robot).....	24
Figure 31: Human Tracking Navigation Policy.....	26
Figure 32: Distribution Function .....	27

Figure 33: Simulation GUI in MATLAB .....	29
Figure 34: Interrupt Flow chat .....	30
Figure 35: GUI Interface (MATLAB) .....	32
Figure 36: Initial map after running the Environment .....	33
Figure 37: Probability Distribution Function.....	34
Figure 38: Robot default location changes w.r.t human location for optimization .....	34
Figure 39: Robot Actions.....	38
Figure 40: Example.....	40

## List of Tables

Table 1: Pseudocode MDP.....	19
Table2: Probability Distribution Function Code.....	27
Table 3: Optimization Coding .....	28
Table 4: MATLAB code for Human Uncertainty.....	33
Table 5: Grey scale for visual differentiation .....	34
Table 6: Creating Belief Space .....	35
Table 7: Code Iterations.....	36
Table 8: Localization .....	37
Table 9: Policy Chart .....	39
Table 10: Policy Code.....	39
Table 11: Comparison Table.....	42

## CHAPTER 1: INTRODUCTION

Robotics holds tremendous potential for benefiting every domain of human life. Although this benefit has been limited to very specialized environments such as factories, technology has matured to integrate robotic technologies into the human environment for everyday use. However, this integration cannot be successful without understanding the interaction between robots and humans. Robot is a machine consisting of three major things sensing, movement & intelligence. Sensing the environment is the initial thing that the robot has to do. It is similar to the way that human sense the surroundings like eyes (light sensors), hands (touch and pressure sensors), nose (chemical sensors), ears (sonar sensors) and tongue (taste sensors). These types of sensors give awareness to robot. The second aspect of the robot after sensing is the movement. Movement or the actuation of the robot in the environment is the basic outcome of the robot. Movement of the robot includes rolling on the wheels, walking on legs, moving some parts like arms. The third factor of the robot is the smartness and intelligence while performing action in the environment. Program is installed in the robot on which the robot plans to maneuver or act in the environment. With all the above discussion it concludes that the robot is a system that contains sensors, control system, manipulators, power supplies and software all working together to perform a task. [1]

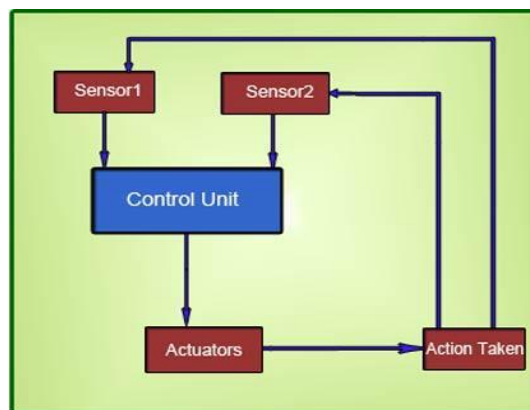


Figure 1: Robot Control System [1]

Robots are mechanical devices programmed to perform specific repetitive functions. They are used routinely to carry out many tasks that people don't want to do because they are boring, dirty and dangerous. Moreover robots are used where the tasks are too complex to do for humans. In few countries robots are used in restaurants kitchens for serving. They are also important earlier in food production, planting rice and tending growing crops. Robots also work as receptionists and cleaning and help look after the elderly in care homes. [2]



Figure 2: Robot Arm for Welding (Complex Tasks) [2]

Robots are used in variety of applications like space robotics, underwater robotics, Electric mobility, Logistics, search & rescue, security robotics, assistance & rehabilitation system and agricultural robots. In assistance & rehabilitation field of robotics, robots can support human in complex, exhausting or repeated tasks. Application areas are both helping in daily life activities and medical rehabilitation. [3]



Figure 3: Assembly Line Robots (Repetitive Tasks) [3]

For mobile robots, navigation is the basic thing that needs to be programmed before acting in an environment. Map based is one of the techniques that is used for mobile navigation. Map based is a technique in which a mobile robot creates a map of its local environment. This local map is then compared with the global map that is initially saved in the memory of mobile robot. Robot creates a map with the help of attached sensors. Sensor detects the hurdles like wall and obstacles and updates its map according to that. Error and

uncertainty analyses play an important role in accurate estimation and map building. It is very essential to cater for the uncertainty in the map e.g. by modeling it with probability distribution functions. The representation used for the map should provide a way to incorporate newly sensed information into map. It should also provide the necessary information for path planning and obstacle avoidance. [4]



Figure 4: Robot Navigation [4]

Now a days robots are used to help elderly people who want to stay at home remain independent past the point they'd usually be unable to live alone due to physical or cognitive difficulties. [5]

For mobile devices navigation is very important. In navigation mobile robot avoids different obstacles to achieve its target. Robot navigation is the ability of robot to determine its own location with reference to a frame then plan a path towards some target. Navigation can be divided into three categories: self-localization, Path planning and map interpretation. [6] Self-localization is the process of figuring out where the robot is located. Planning is the way to achieve the target is best possible way. Map interpretation is the logic that robot creates to overcome the obstacles and achieving goals. [7]

Navigation of robot in uncertain environment in which the obstacles are moving randomly is very tricky. Furthermore if the target is also random then in this case the planning of the path becomes complicated. There is variety of techniques to overcome this criticality. Robot updates the path according to the current dynamic obstacle and moves by creating a probabilistic function to finds its target.

One of the methods to solve this uncertainty is Markov Decision Process (MDP) [8]. MDP provides a model of decision making in the situation of uncertainty where the target is partly random. This technique is used to solve problems with dynamic programming for example if a robot tracks a target, its own motion is un-deterministic and the target that the



robot is chasing is also not fully specified. These factors increase the dynamicity while tracing target. There are four basic components of MDP: States, actions, effect of the action and immediate value of the action [10]. Moreover solution of an MDP is called a policy and it specifies the best policy. Policy is described by a value function [11].

MDP technique is quite helpful to interpret when target is uncertain. But if robot's own position is uncertain, then a different technique is used i.e. Partially Observable MDP (POMDP). In POMDP we add a set of observations to a model. The current state gives us an observation which provides a guess about the state. The observation can be probabilistic which tell us the probability of each observation for every state in the model [12].

POMDP technique takes a lot of computational power during problem solving. As in this technique the robot manipulates possibilities from each possible node and finds its target from each node and after moving a step generates another similar tree of nodes until target is achieved. During its movement robot also updates the values of probability distribution function and assign every possible node a value. To resolve such issue of computational power a Localized POMDP technique is introduced by which computational power will be reduced as well as by defining a localized technique robot unusual movement while target searching will also be reduced [13].

Now a day's robots are used for health/care and for social use beyond the traditional scope of surgical. It is a growing research field based on human perception centered on supervised learning. Our application is to implement this technique in which the robot will optimally find its way in uncertain environment at home to help elderly people who want to stay at home due to physical or cognitive difficulties [11].

The robot and the target operate in a grid environment defined at home. In one step robot can either stay or move to one of the eight adjacent positions in home environment as shown below:

3	2	1
4	Move	8
5	6	7

Figure 5: Possible Robot Moves (MATLAB GUI Reward Values Show Box)

The robot pays a cost for each move and receives a reward every time it arrives in the same position as that of the target. The robot makes observations that generate actions by this an internal belief state are created. When robot move it update values the cells of grid according to observations and save in its memory. A state estimator in programming is used for updating the belief state based on last action, the current observation and previous belief state or values to cells of grid. After computing different states a policy is defined that the robot will avoid walls and will travel to the shortest path towards its target i.e. human. To implement this technique a simulation on MATLAB software was built in GUI (Graphical User Interface) environment [10].

Our goal in this project is to track robot optimally after it is called by a human. The robot will stays closer to human in order to get maximum award and reach to human in lesser time. This technique is implemented in this project by defining different default locations of robot. Robot automatically selects and travels towards the default locations in the home map to stay closer to elder person. Moreover, in this project robot learning phase is also defined that helps robot to learn the positions of the robot with respect to time and update the values each time when it observes the human.

In Chapter 2, different navigation methods of mobile robot are explained that includes potential field method, visibility graph mapping and tracking method, sampling based target tracking method and robot navigation by Bayesian filtering method. In this chapter uncertainty solving methods described are Markov Decision Process (MDP) and Partially Observable Decision Process (POMDP). MDP is used when the target is un-deterministic and robot motion is known. Methodology that is adopted to solve this uncertainty of target is described in this chapter. Furthermore POMDP technique is also described for un-deterministic motion of robot.

In Chapter 3, the methodology of localized POMDP technique is simulated in the MATLAB software. In this methodology, initialization of map is carried out, after that target defining and path segregation for localization is implemented. Furthermore policies are defined to solve the uncertainty while chasing the target. Optimization technique is combined with POMDP technique to optimize the movement of robot.

In Chapter 4, MATLAB coding and simulation results are presented to understand full picture of human tracking robot navigation. In this chapter, MATLAB interface is described step by step with the experimental results. At the end of this chapter, comparison of simple POMDP technique and localized POMDP technique is tabulated to get a clear idea of this efficient technique.

In the last Chapter 5, conclusions and future recommendations of this localized technique is discussed that will be helpful for the robot and the target operated in a dynamic environment. The Localized POMDP technique has some limitations. This method is incomplete as when the probability of human varies drastically then the robot will be unable to find its way correctly towards the human when it calls. This issue can be solved by guiding robot about the presence of human.

Robots are useful in bringing comfort to human's life, for example caring for elderly people who want to stay at home due to physical or cognitive difficulties, help in monitoring elder person, offers endless patience to elder person, preserve dignity and promote independence and serve as a communication tool. For a Mobile Robot the ability to navigate in its environment is important, when human calls the robot, it will track the human and avoids different dynamic and fixed obstacles.

Navigation algorithm of a robot is a basic constituent that is to be programmed for its desired motion. Autonomous navigation of robots is tough to be planned in dynamic environment. Navigation Uncertainty Factors are:

- Target movement is uncertain
- Robot movement is not fully accurate
- Dynamic uncertain Obstacles in the way while chasing target

The technique that we are using in this project to solve uncertainty is POMDP. It is a decision process in which the agent cannot directly observe the states. Instead, it maintains a probability distribution over the set of possible states, based on a set of observations and observation probabilities. POMDP is a complicated technique that requires more processing and computational power. Our objective is to introducing a new Localized-POMDP technique that enable robot to reduce computational power and allow it to calculate results in lesser time. Our second objective is to merge robot learning phase in this technique. In this phase a robot learns the positions of human with respect to time and creates a probability distribution function that help the robot to navigate human more accurately. Our achievement in this project is that we have carried out the above mentioned objective through MATLAB software simulations detailed in chapter 3 and chapter 4.

## CHAPTER 2: LITERATURE REVIEW

Navigation of mobile robot is basically to find a collision free motion for the robot from one configuration space to another configuration space. Sensors detect the hurdles in the way while robot is in motion. Robot explores and senses an unknown environment to construct a representation that is useful in navigation. Once the task and the robotic system is defined, selection of algorithm is being carried out. Algorithm is selected in such a way that the execution time is optimal with the problem of motion planning. Optimality, completeness and computational complexity naturally trade off with each other. Some methods of motion planning are very efficient but it lacks the completeness and vice versa [10].

### 2.1 Potential functions

A potential field method of tracking an object is basically through an attractive and repulsive gradient. The goal location generates an attractive potential that pulls the robot towards the goal. The obstacles generate a repulsive potential that pushes the robot far away from obstacles. The negative gradient of the total potential is treated as an artificial force applied to the robot. The sum of the forces is shown in the following equation [32].

Artificial Potential

$$U(q) = U_{goal}(q) + \sum U_{obstacles}(q) \text{ ----- (Eq: 1.1)}$$

In the above Eq 1.1,  $U_{goal}(q)$  is the attractive potential &  $\sum U_{obstacles}(q)$  is the repulsive potential

Artificial Force Field

$$F(q) = -\nabla U(q) \text{ ----- (Eq: 1.2)}$$

In above Eq 1.2,  $-\nabla U(q)$  is the negative gradient

Repulsive Potential creates a potential barrier around the obstacle region that cannot be traversed by the robot's configuration. It is usually desirable that the repulsive potential does not affect the motion of the robot when it is sufficiently far away from obstacles. For example in the following figure 6:

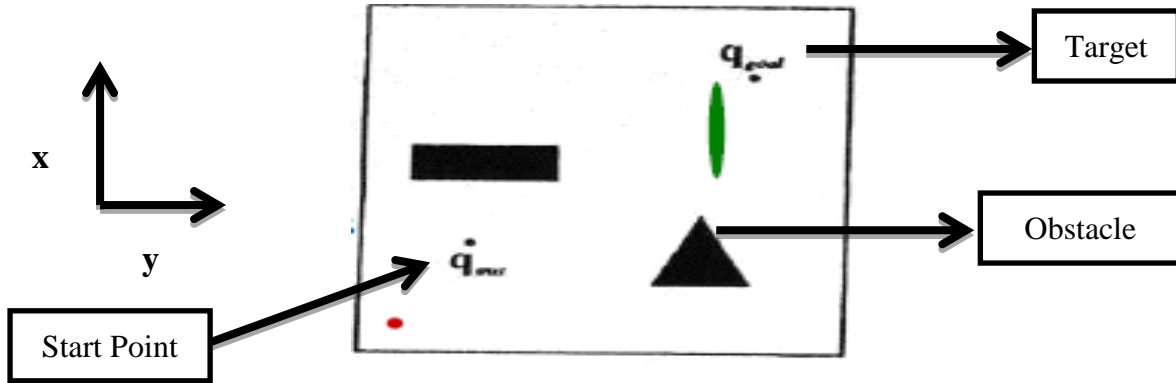


Figure 6: Potential Field Example [32]

In the above figure 6 the attractive potential is given as follows in a parabolic function:

$$U_{goal}(q) = \frac{1}{2} \xi \|q - q_{goal}\|^2 \text{ ----- (Eq:2.0)}$$

The parabolic shape equation graph of attraction is shown below which is minimum at goal:

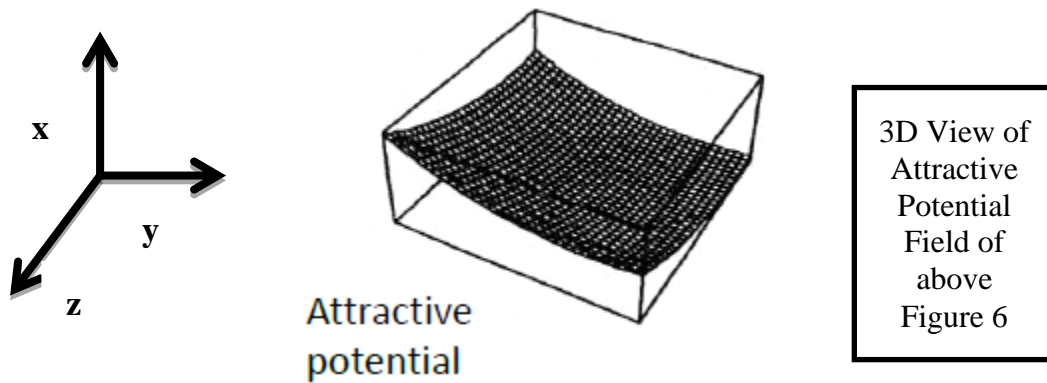


Figure 7: Parabolic Function [32]

The attraction force of the above function is:

$$F_{att}(q) = -\xi(q - q_{goal}) \text{ ----- (Eq:3.0)}$$

The repulsive force calculations are as follows:

$$U_{rep}(q) = \frac{1}{2}\eta \left( \frac{1}{\rho(q)} - \frac{1}{\rho_o} \right)^2 \text{ if } \rho(q) \leq \rho_o \text{ -----(Eq: 4.1)}$$

$$U_{rep}(q) = 0 \text{ if } \rho(q) > \rho_o \text{ -----(Eq: 4.2)}$$

Where:

-  $\eta$  is the scaling factor

$$- \rho(q) = \min_{q=CB} \|q - q'\|$$

-  $\rho_o$  is the positive constant (distance of influence) of the obstacles.

The force is given as follows:

$$F_{rep}(q) = \eta \left( \frac{1}{\rho(q)} - \frac{1}{\rho_o} \right) \frac{1}{\rho^2(q)} \nabla \rho(q) \text{ if } \rho(q) \leq \rho_o \text{ ----- (Eq: 5.1)}$$

$$F_{rep}(q) = 0 \text{ if } \rho(q) > \rho_o \text{ ----- (Eq: 5.2)}$$

The repulsive graph as shown as follows, where z-axis is the force magnitude:

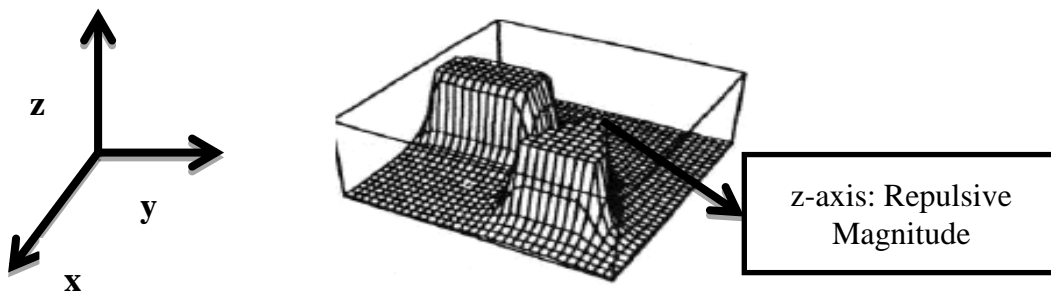


Figure 8: Repulsive graph [32]

By adding the attractive and repulsive force we obtain the following potential graph:

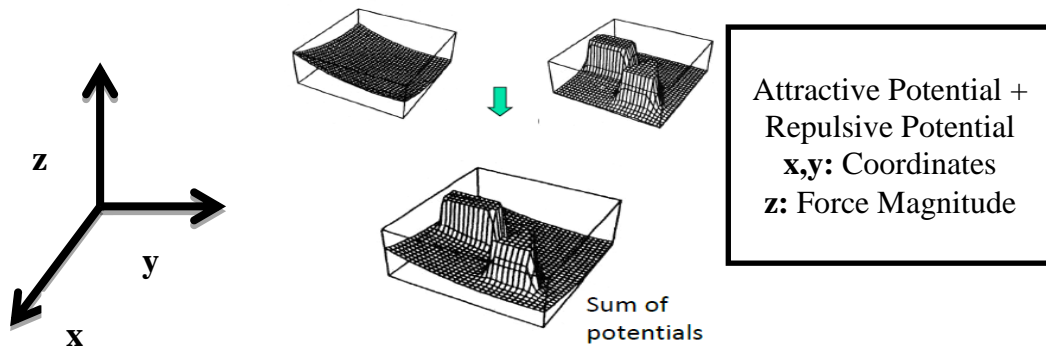


Figure 9: Sum of Potentials [32]

## 2.2 Visibility Graph

Visibility is the type of roadmap navigation. In this navigation method the polygonal configuration space for a robot is segregated in such a way that vertices of each object is connected to one another. It is formed by connecting all visible vertices, start point and the end point to each other. For two points to be visible no obstacle can exist between them. Path exists on the perimeter of obstacles. The following is the method of drawing the lines between vertices [32].

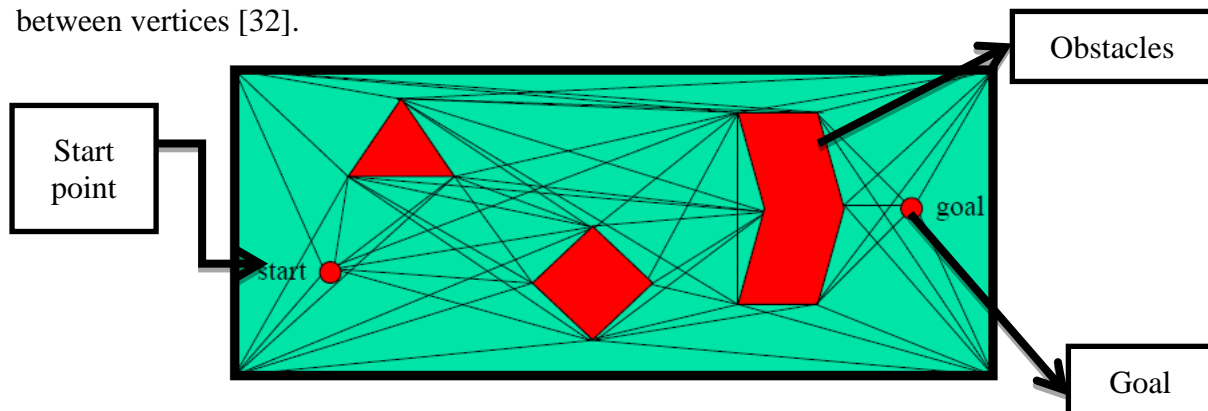


Figure 10: Visibility Graph Navigation [32]

Each line in the above figure 10 represents part of a path from the start to the goal. When the robot travels on the path made by the above method the clearance from the objects is zero that is not a proper navigation for a robot. This problem is can be solved by using another roadmap method of navigation i.e. Voronoi diagram. In this method the path that is created has a maximum clearance between the point and obstacles. Locus of the points in this diagram is equidistant from the closest two or more obstacle boundaries including the workspace boundary [32]. The Voronoi diagram is shown in the figure 11:

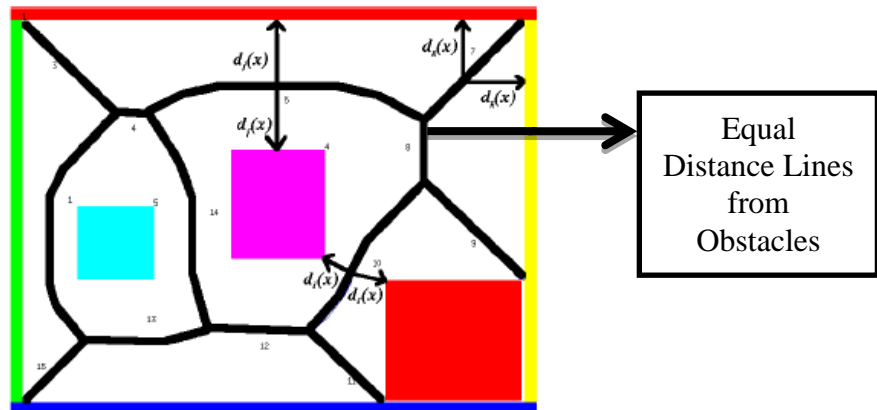


Figure 11: Voronoi Diagram [32]

The advantage of this method is that the roadmap that is created avoids obstacles as much as possible.

### 2.3 Cell decomposition method

In this method of navigation the cells are made and are decomposed into small segments. There are two types of methods exact cell decomposition and approximate cell decomposition. Exact cell decomposition is a trapezoidal decomposition in which the free space is decomposed into trapezoidal and triangular cells. The decomposition is shown in the following diagram [32].

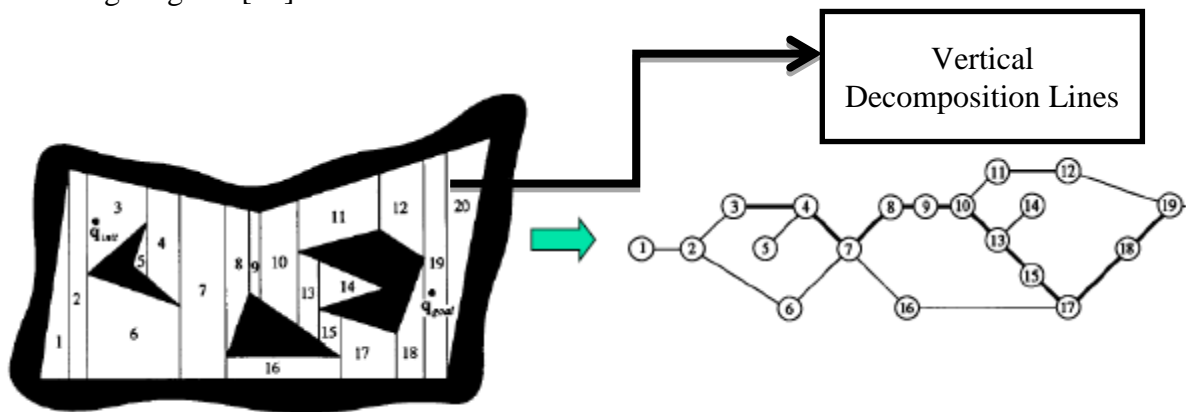


Figure 12: Cell Decomposition Method [32]

Connectivity graph representing in the above figure 12 is representing the adjacency relation between the cells. For the path the mid points of the intersection of two consecutive cells are connected as shown in the figure 13:



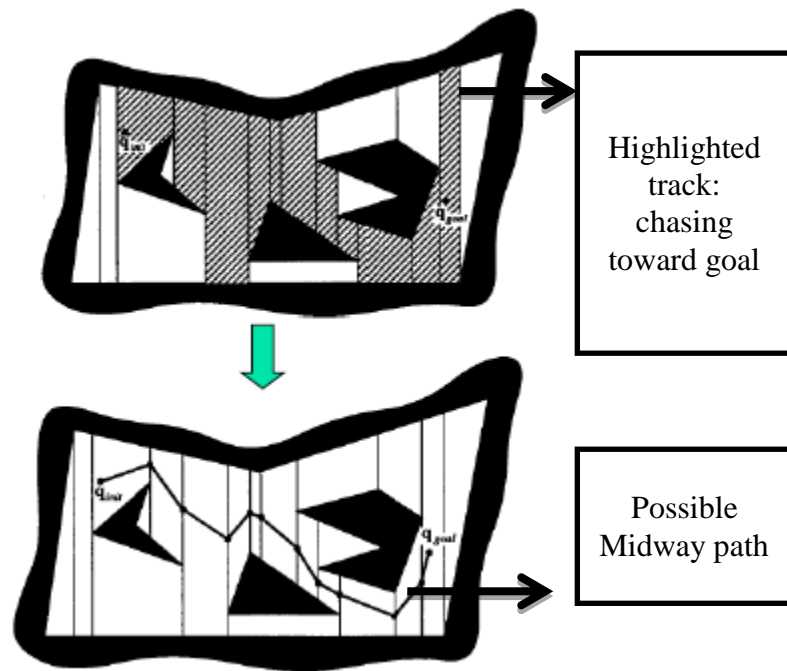


Figure 13: Path formed by cell decomposition method [32]

## 2.4 Sampling based algorithm

In this method, the random samples are scattered in the map as shown in the following figure 14: [32]

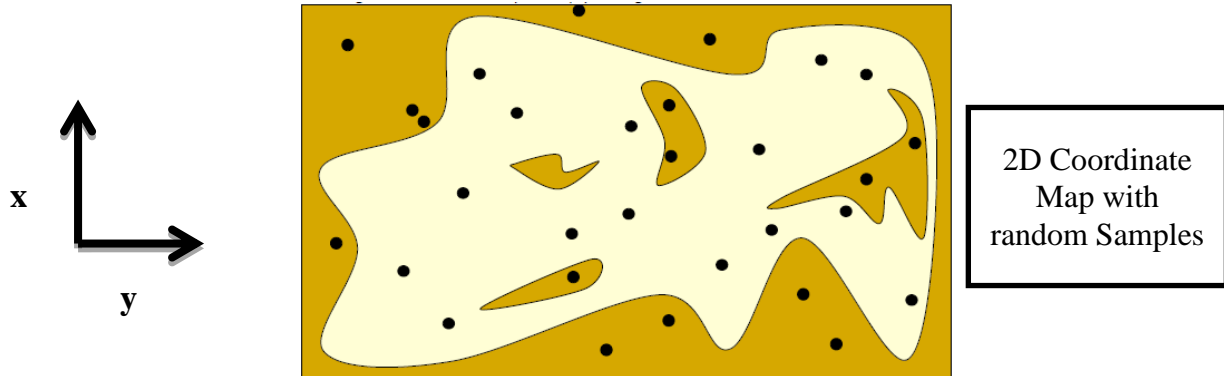


Figure 14: Random Samples [32]

After that the sampled configurations are tested for collision. The samples which are inside obstacles are deleted. The highlighted black dots in the following figure 15 are deleted.

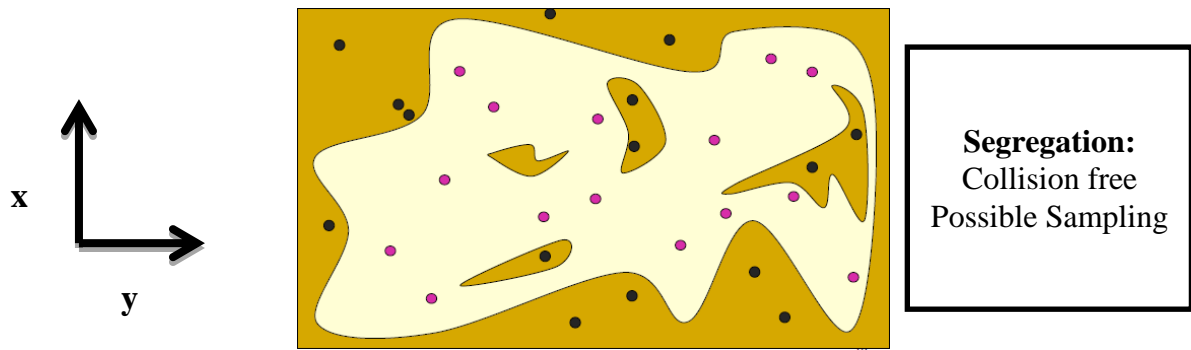


Figure 15: Sample Selection [32]

The collision free configurations are retained as milestones. Each milestone is linked by straight paths to its nearest neighbors as shown in the figure 16.

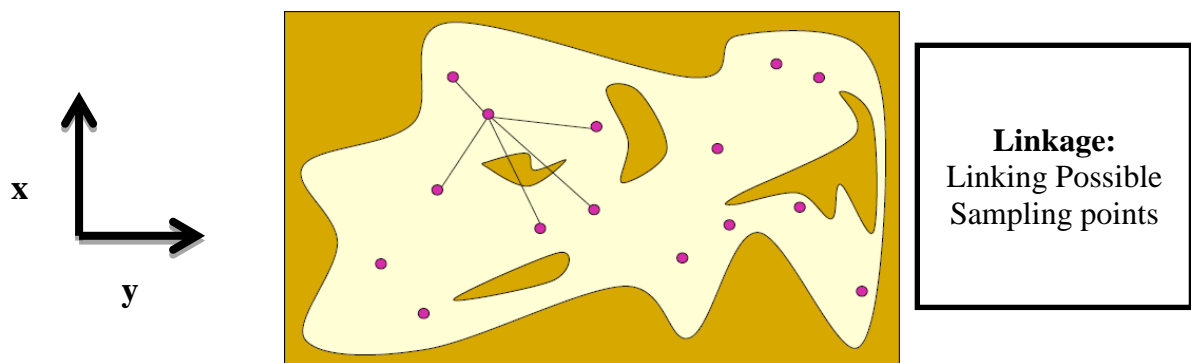


Figure 16: Straight Path linkage [32]

The collision free links are retained as local paths form the probabilistic sample based road map as shown below:

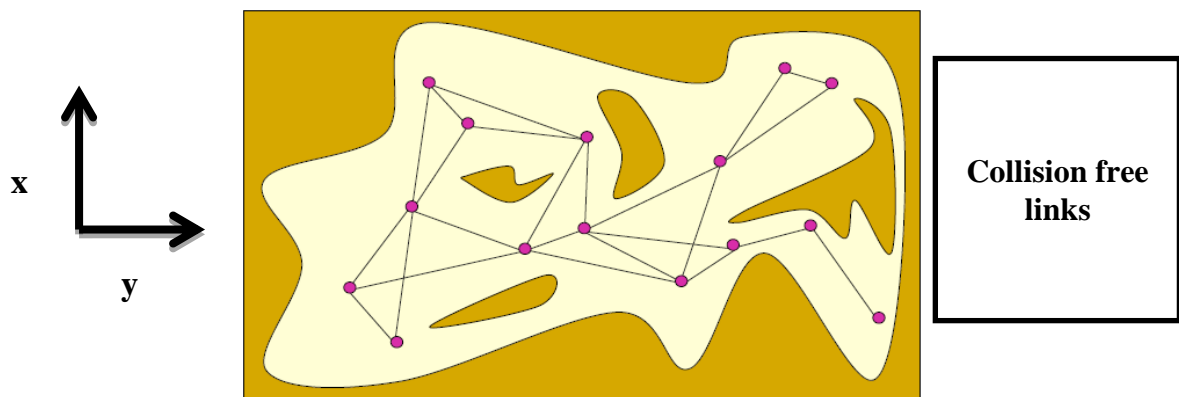


Figure 17: Collision Free Links [32]

Suppose that the above method is applied to search for a path from 's' point to 'g' goal as shown in the figure 18.

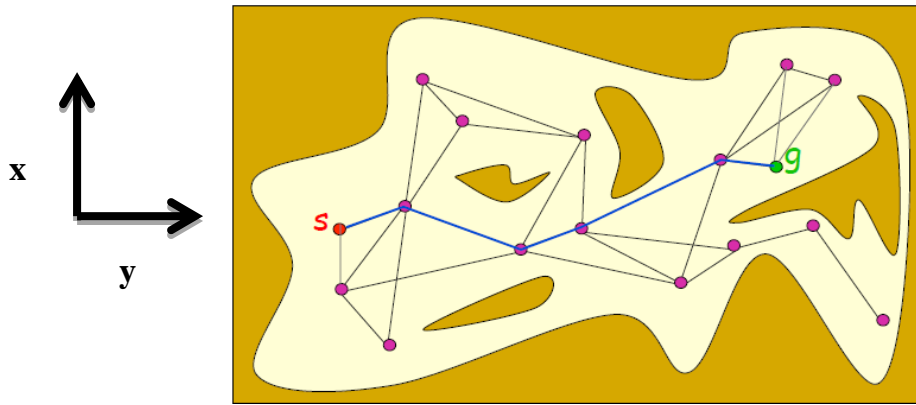


Figure 18: Start to Goal Path [32]

## 2.5 Robot Navigation with Bayesian Filtering

In robot navigation the path planning is essential to optimize the movements of robot. Sensors are used to create an environmental map. Robot owns location and the target location estimation is very important to navigate in that particular environment. The probability map based localization is the basic theme that is used for determining the robot own position. In the following figure 19 robot location estimation is defined in the following diagram [32].

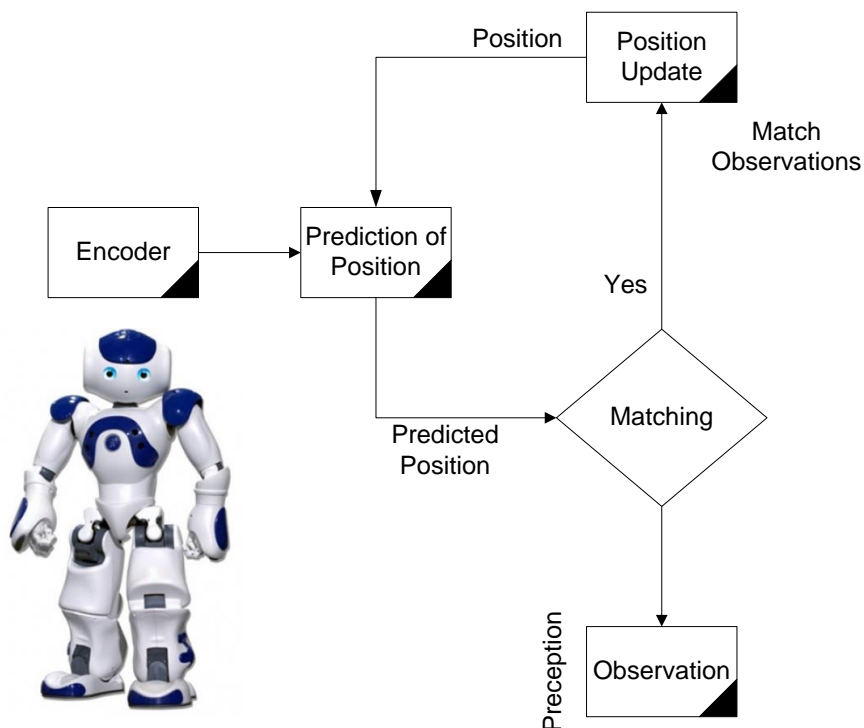


Figure 19: Probability Map based Localization [32]

In the above figure 19 the encoder senses the movement of the robot in real space and gives feedback to a state estimator that predicts the position of robot. Then this predicted position of robot is matched with the map data base to estimate exact position [11].

State estimation of mobile robot can be carried out by Markov Localization. In this method the robot's belief is represented by a probability distribution function over possible locations. This method uses Bayer's Rule and convolution to update the belief whenever the robot senses or moves. During each update, the probability for each state of the entire space is updated [32]. Assume the robot position is one-dimensional as shown in the following figure:

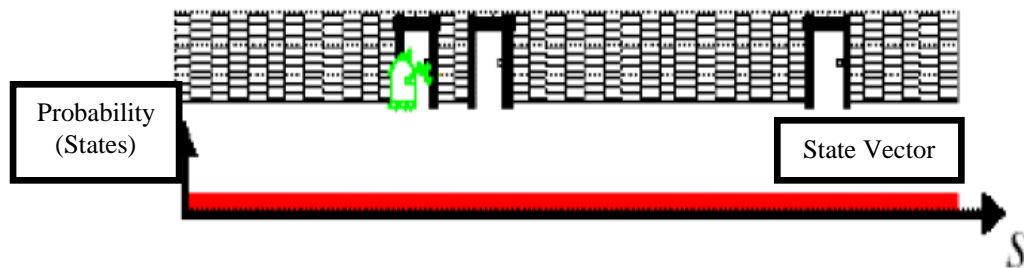


Figure 20: Robot One dimension Location [11]

The robot queries its sensors and finds out its next to a door as shown below:

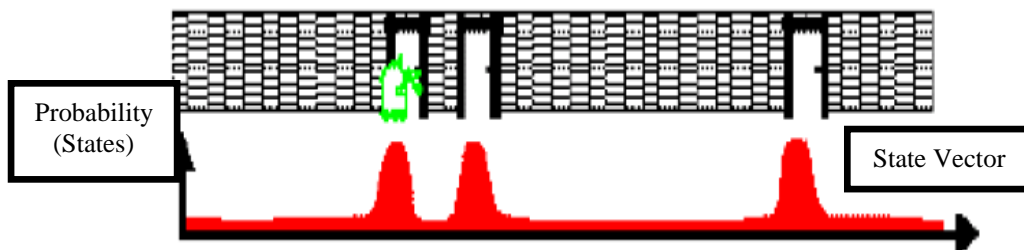


Figure 21: Expected location [11]

After that the robot moves one meter forward. To account for inherent noise in robot motion the new belief is smoother. The robot queries its sensors and again it finds its next to the door as shown below:

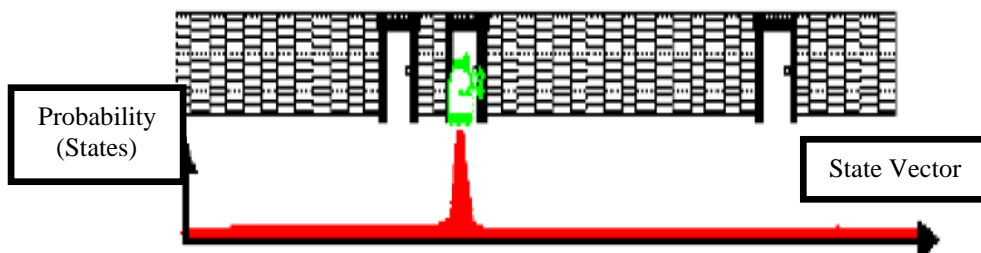


Figure 22: Robot Exact Location [11]

## 2.6 Markov Decision Process

The basic motion planning of a mobile robot is to produce a continuous motion from a start point to target. Navigation of robot can be decomposed into three tasks: Mapping, Path planning and Collision avoidance [29]. Hierarchy of mobile robot planning is defined in the figure 23:

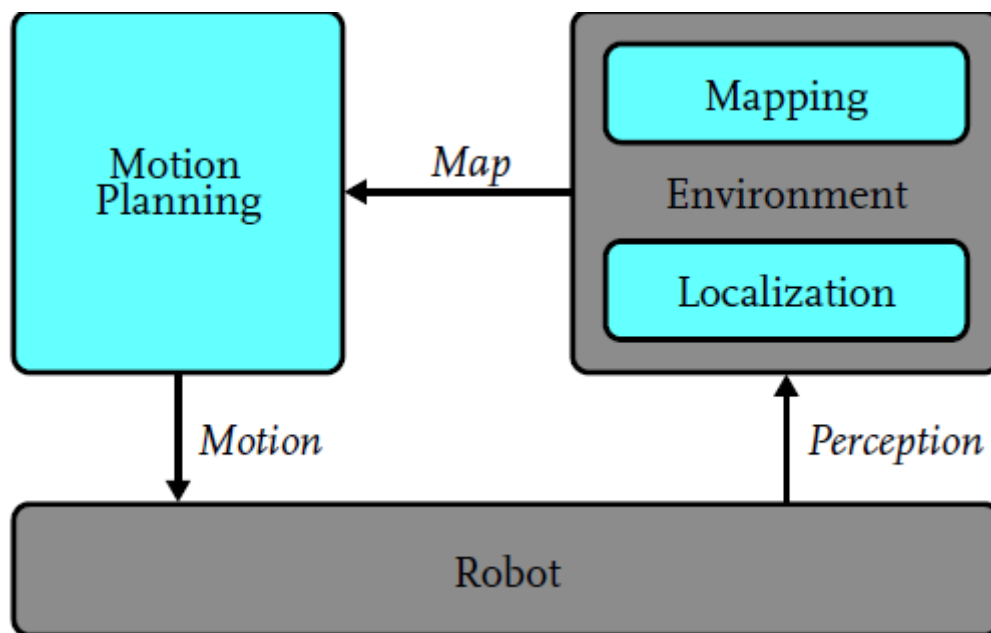


Figure 23: Basic Navigation Flow Chart

Robot perceives form the environment from its sensors. Mapping is then processed by the robot according to the observed environment. After that the rules and regulations that are programmed in the robot is processed [29].

In the most of the cases the target is uncertain; the uncertainty in the target can be solved with different techniques. Target uncertainty means that the target position is not fully deterministic. One of the techniques to solve such problem is Markov Decision Process (MDP). Moreover if the robot's own position is un-deterministic then another advance technique is used named as Partially Observable MDP. Following is the detail of these two techniques [29].

MDP is the mathematical model of an uncertain environment, where the results are partially random. It is very helpful for reinforcement learning and dynamic programming. It is a basic technique to solve complicated partially observable problems. Furthermore, it is a discrete time control process. In the Figure 24 an MDP example is shown in which there are three states  $s_1$ ,  $s_2$  &  $s_3$  with two actions  $a_1$  &  $a_2$  [21].

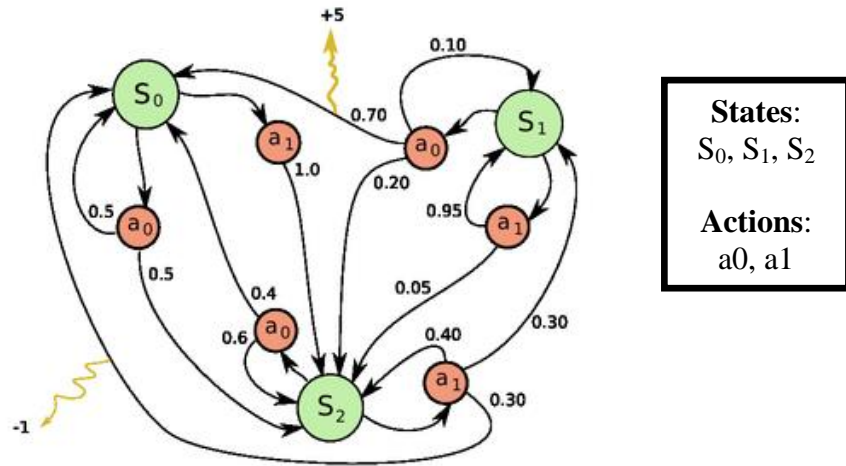


Figure 24: Example of MDP

This technique is described as states, actions, state transition function and reward function. The expected reward only depends on previous state and the action taken. Agent acts in such a way to maximize the expected sum of reward [12].

For a given policy let ' $V_t(s)$ ' is the expected sum of reward starting from state 's' to target in 't' steps.

$$V_t(s) = R(s) + DEV \text{ ----- (Eq: 6.0)}$$

Where: DEV (Discounted expected value) =  $\gamma \sum T(s, s')V_{t-1}(s')$

This DEV is basically the future expected value for all resulting states  $s'$ . Immediate value of being in a state 's' is  $R(s)$ . Where  $T(s, s')$  is the state transition function which gives probability function over each world state and agent action. For a greedy policy an action is taken to maximize the expected immediate reward plus the expected discounted value of the next states [26].

$$\Pi(s) = \max[R(s) + \gamma \sum T(s, s')V_{t-1}(s')] \text{ ----- (Eq: 7.0)}$$

We will get values of policies as  $\Pi_1(s), \Pi_2(s), \dots, \Pi_n(s)$ . The optimum policy is the maximum attain value as defined in Equation 8:

$$V^*(s) = \max[\Pi_1(s), \Pi_2(s), \dots, \Pi_n(s)] \text{ ----- (Eq:8.0)}$$

Table 1: Pseudocode MDP

Pseudocode
$V_1(s) = 0 \forall s$ $t=1$ <b>while</b> $ V_t(s) - V_{t-1}(s)  < \epsilon$ $t = t + 1$ <b>for</b> $\forall s$ $Q_t(s) = R(s) + \gamma \sum T(s, s') V_{t-1}(s')$ $V_t(s) = \max Q_t(s)$ <b>end</b> <b>end</b>

## 2.7 Partially Observable Markov Decision Process (POMDP)

MDP only compute the optimal policy for the current state ‘s’ but lack the capability to cater for the agent whose current position is not completely deterministic. POMDP framework provides systematic way to solve this. A POMDP is a generalization of a MDP technique in which probability distribution function is maintained over the set of possible states. Results of POMDP yield the optimal action for each possible belief over the world states. The optimal action maximizes the expected reward of the agent over expected infinite horizon [26].

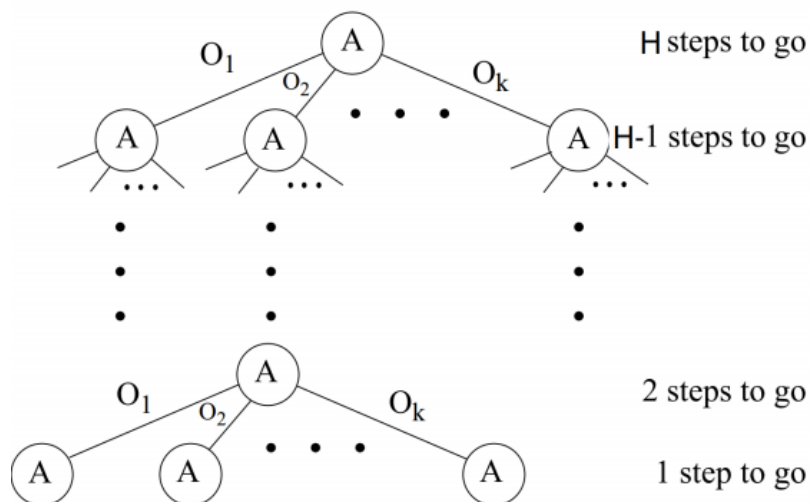


Figure 25: Finite Horizon (POMDP)

The framework of POMDP contains  $\{S, A, T, R, \Omega, O\}$ ; where  $\Omega$  is observation function which gives a probability distribution for each action and resulting state.  $O(s', a, o)$  is the probability of making observation  $o$  given that the agent took action 'A' and present in next state  $S'$  [30].

The basic structure of POMDP can be represented as follows: [31]

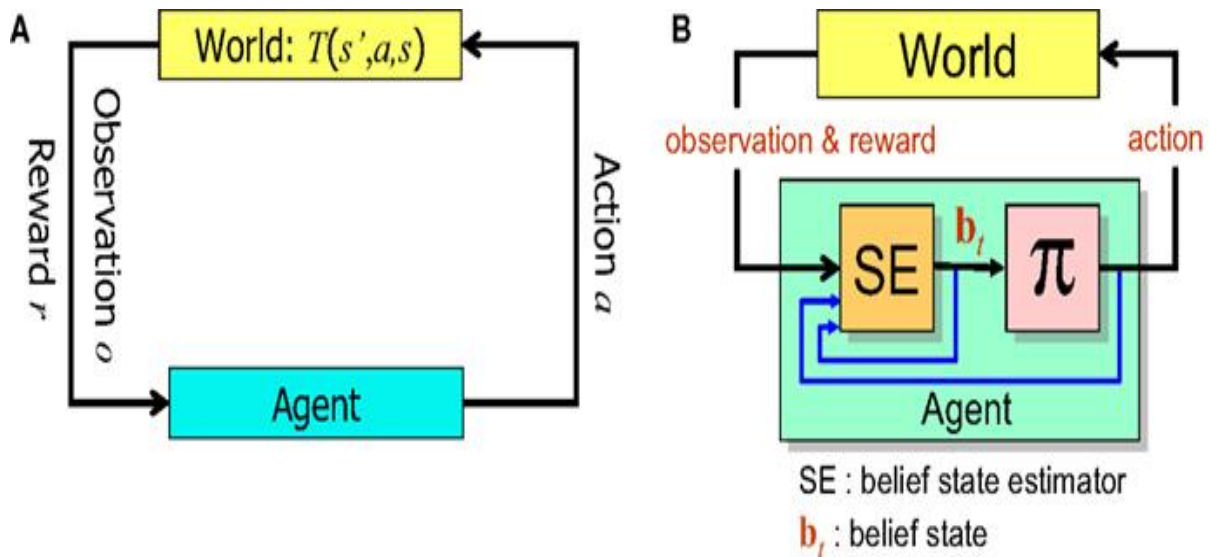


Figure 26: POMDP Structure

In the above Figure 26; SE is the state estimator that determines the state from the observations taken from the environment. “ $\pi$ ” is the policy of move / action that defines move according to the estimated state. After taking an action the robot again estimates its location and afterwards changes the state estimator [23].

In this research we have taken an example of human tracking robot navigation at home. In this the robot will follow the human and helps the human in his / her daily life activities. Robot will search and track the human when human needs it. In this application we have implemented a Localized way of implementing POMDP technique [22].

Robot senses the human with sensors attached on the body of robot. It may be vision based tracking or some other way. In this research we will not discuss the methods of sensing human rather we will discuss regarding navigation of robot in a given map of home [24].



## CHAPTER 3: METHODOLOGY

To implement Localized POMDP technique in navigation of mobile robot we have simulated it in MATLAB. Following are the programming steps to implement this technique. Note that these following steps are described in a generalized way so that a programmer can easily program the technique. Our goal is to implement the POMDP technique for navigation of human tracking robot in less time. The flowchart of methodology is describes as follows:

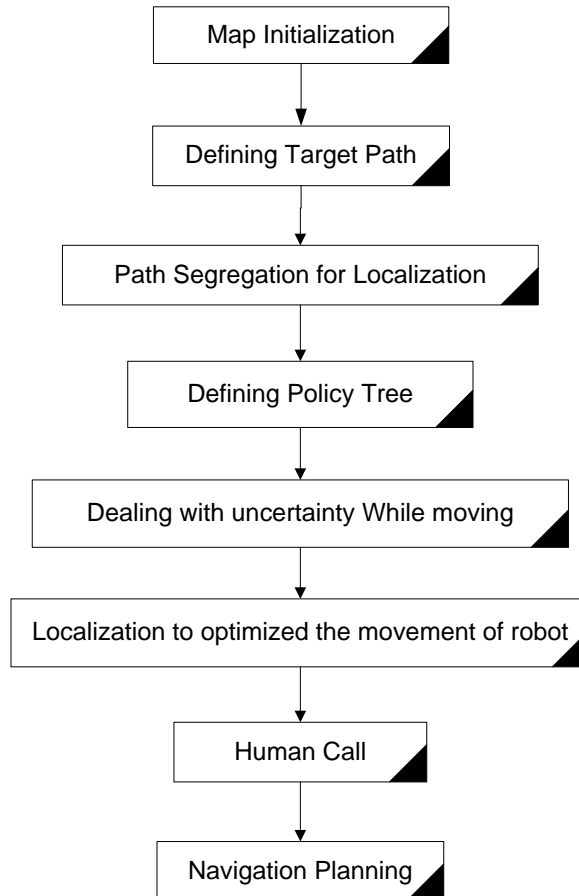


Figure 27: Methodology Flow chart

### 3.1 Map initialization

Robot must have an idea of the path in which it has to travel and to track a target. For this, robot learns the map initially. In programming of robot the map is initially defined. In the following Figure 27 the black lines are the walls and remaining is the space for robot to move.

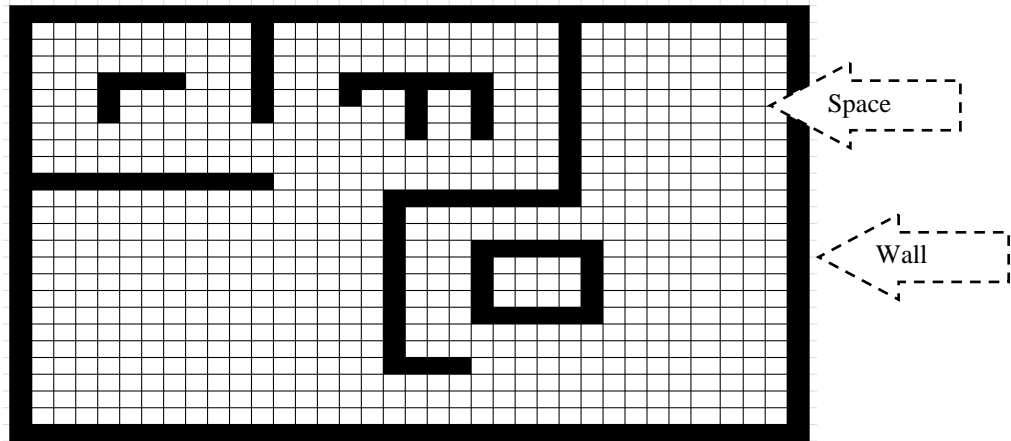


Figure 28a: Initial MAP

### 3.2 Defining Target path

The second step is to define the path of the target. Purpose of defining the target track is to tell the robot to trace the target on that path while tracking phase. In the Figure 28, target first and last point is shown from 1 to 198. For the sake of simplicity in the calculations and understanding; the path of the human is specified form 1 to 198 points, which means the target will be available in any of these points from 1 to 198.

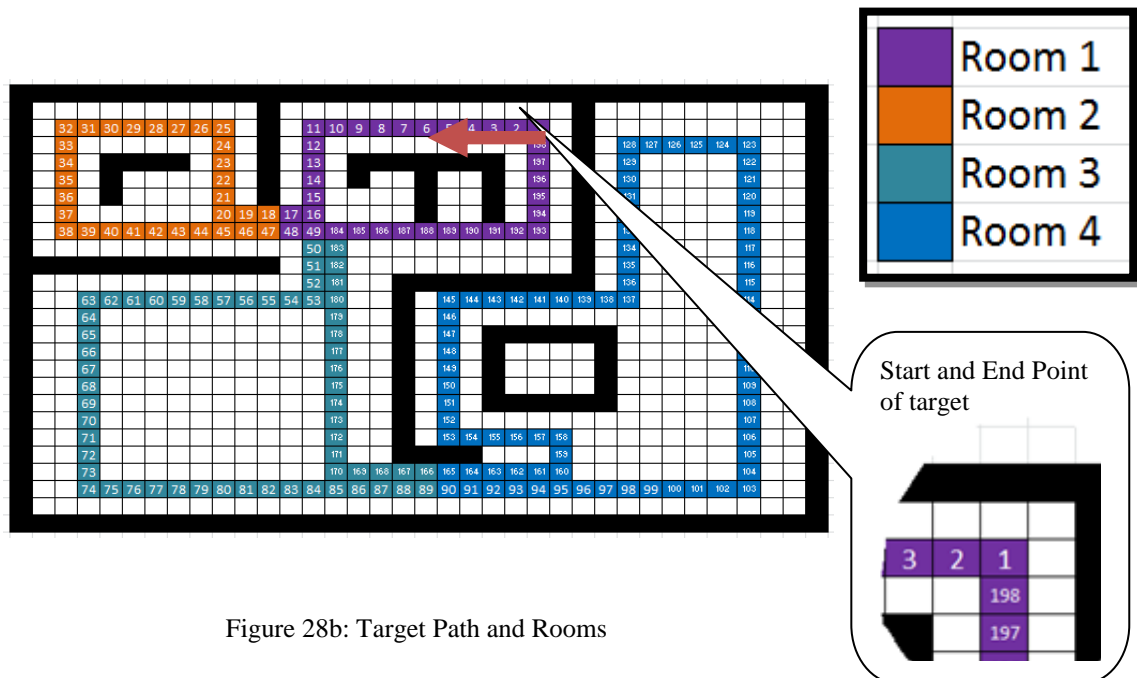


Figure 28b: Target Path and Rooms

### 3.3 Path segregation for localization

To optimize the computation power in POMDP technique we have implemented a path segregation technique. In the Figure 29 the rooms are specified with different colors. Robot changes its location automatically after determining the expected location of target. Default locations of robot are shown in Figure 29 by ‘Red’ color (1 ~ 4). Methodology that is adopted to optimize the robot movement will be discussed afterwards in this chapter.

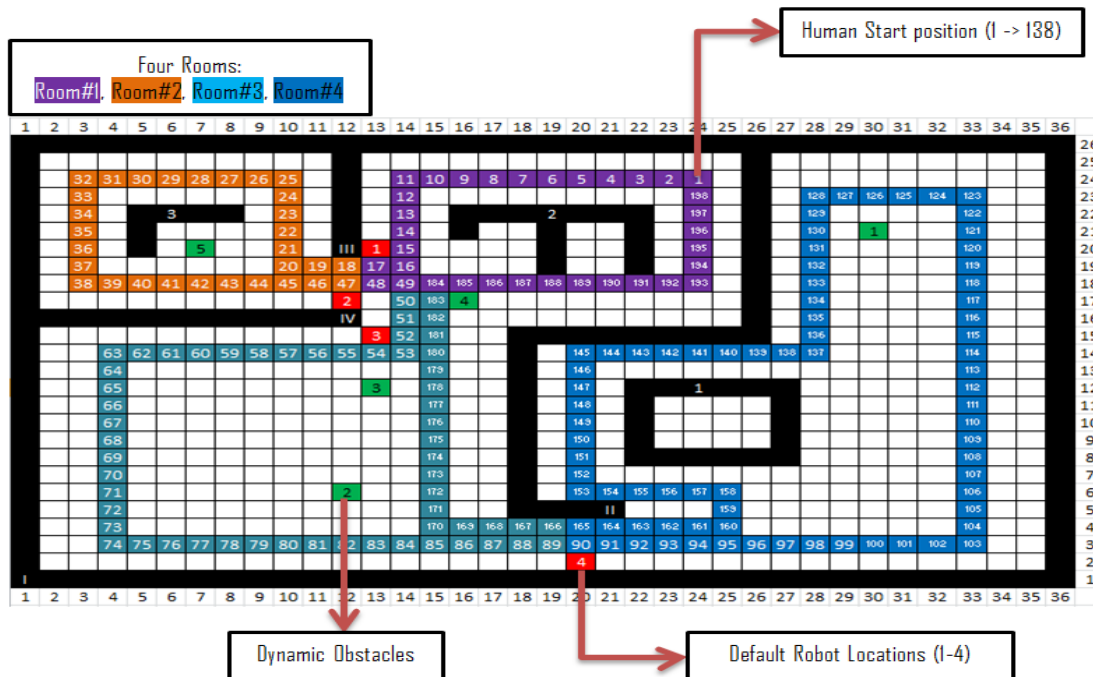


Figure 29: Robot Map

### 3.4 Defining Policy Tree

POMDP technique can be decomposed into a state estimator and a policy. Moreover agent is unable to observe current state but by making observation based on the actions and resulting state. The next step after segregation is to define a policy tree. Policy is basically the brain of POMDP in which the rules and regulations of the robot are defined. Robot makes decisions according to this policy tree. In our application of human tracking robot the policy tree is shown in the Figure 30.

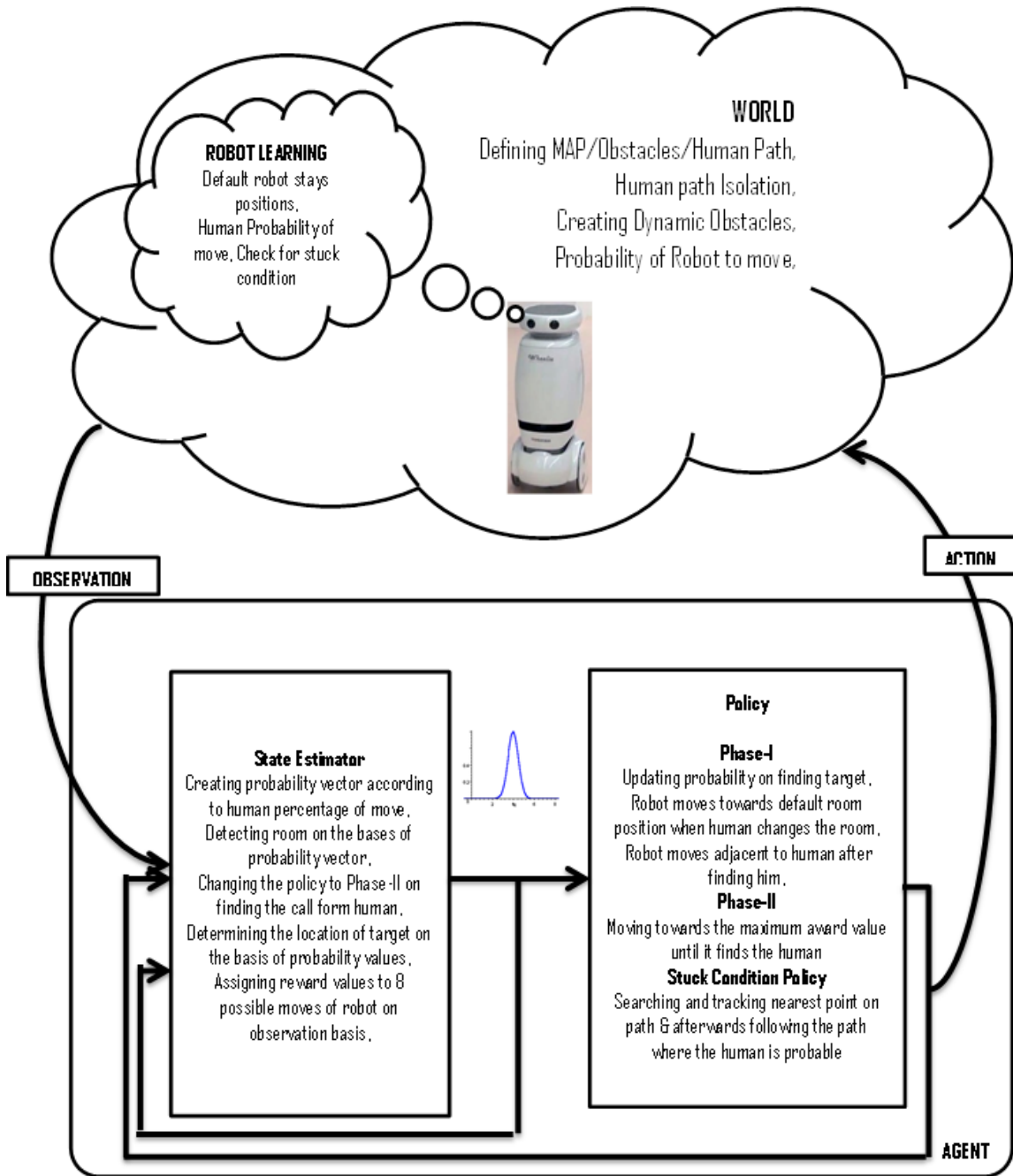


Figure 30: Policy Tree (Human Tracking Robot)

In the above Figure 30, the environment of robot is defined. In the world of the robot; map, information about obstacles and human path are defined. With this information the robot will be able to observe its surroundings and learn the environmental changes. Robot learns the environment by learning the probability of target / human and updating its state values in grid.

SE (state estimator) in the figure 30 acquires data from environment by observing it and generates vectors on adjacent grid's cells. Robots observe the previous human behavior (percentage of move) and generate probability vector values according to that and determine the location of the target.

After determining the state, policies are to be defined. In our application the policy is categorized in two parts. In first category the robot after determining the location of human in a specified room, moves to default room location. When human calls the robot it moves towards the human probable position in that specified room defined in category policy second. While finding the target robot meanwhile updates the probability values. In category second policy, the robot moves towards the maximum award value until it finds the human/target.

If the robot stuck while searching human then a stuck policy is defined to achieve full traceability of human. The stuck condition policy will be described later on in chapter 4.

The policy tree flow chart is shown in the Figure 31. After the initialization of GUI, map obstacles and path of target is defined in the output function. After defining these above mentioned parameters simulation will begin. Robot will move according to the information initially programmed in the robot. Initial information contains default locations of robot perspective to room locations. Robot will gather the information, manipulate the initial data and process it through probability function defined in the policy tree. In probability distribution function robot creates a function that generates probability vectors in eight different directions. Human's probability of move is learned by the robot and robot will generate a data according to that. Meanwhile robot also updates the data if any human movement variation is observed. The probability vector will also help the robot to detect the specific room in which human is probably present.

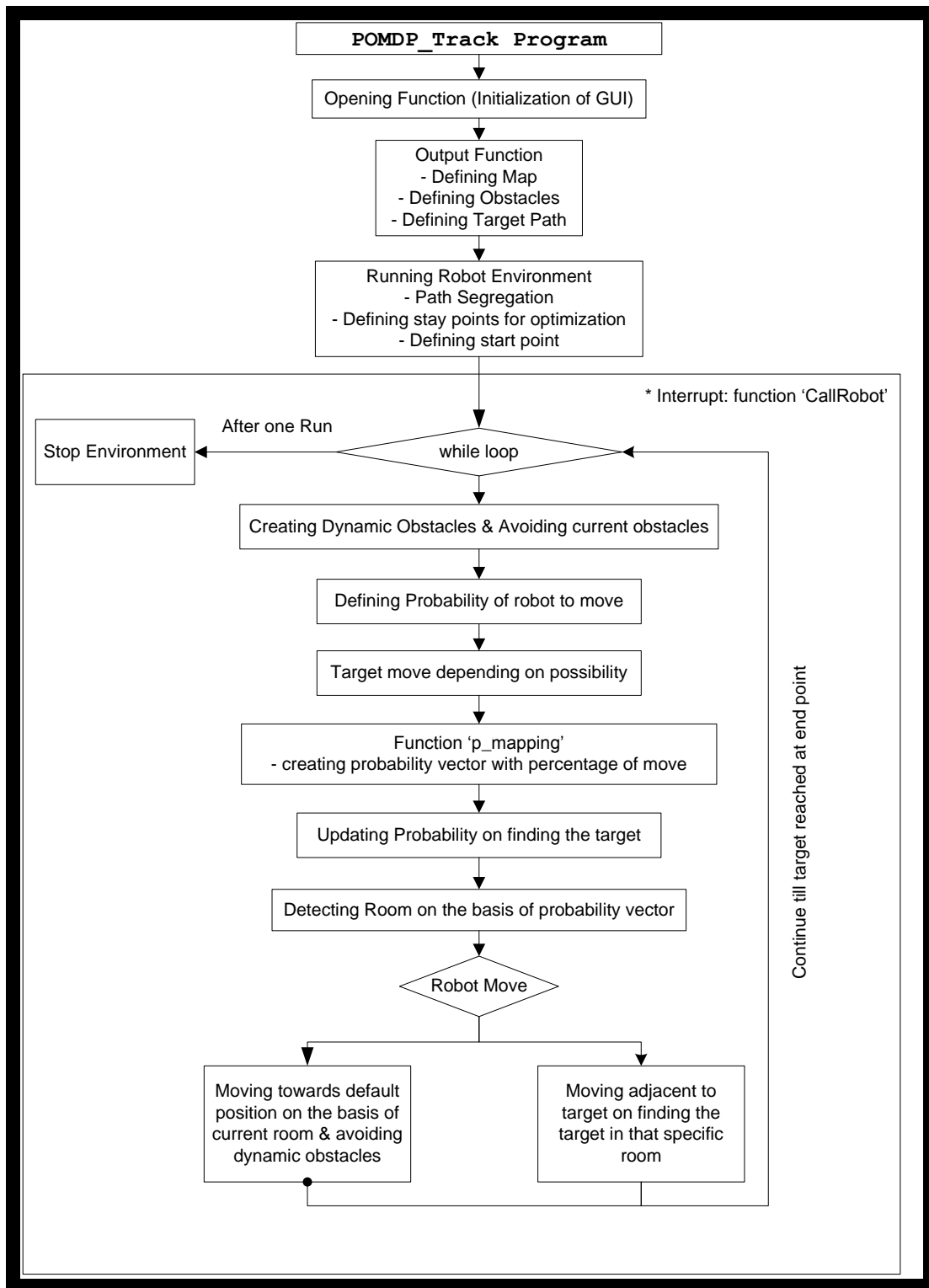


Figure 31: Human Tracking Navigation Policy

### 3.5 Dealing with uncertainty while moving

Robot movement in real life is always uncertain. Robot moves in environment in a grid structure but there is always a probability that robot might have taken that step correctly

or not. This makes the robot to be uncertain in the environment. This factor of uncertainty is reflected in our simulation by defining a probability distribution function (Figure 32). This function creates a grid structure around a robot and assigns values to adjacent cells. The code of the above function in MATLAB is given table 2.

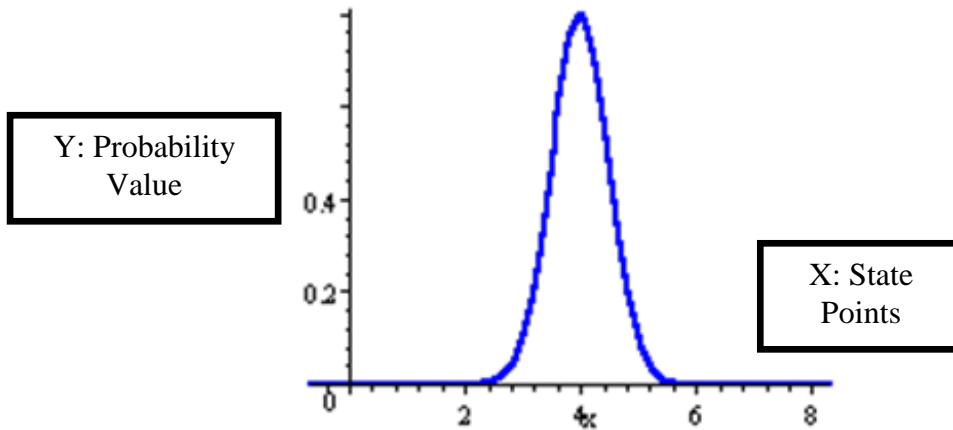


Figure 32: Distribution Function

Table2: Probability Distribution Function Code

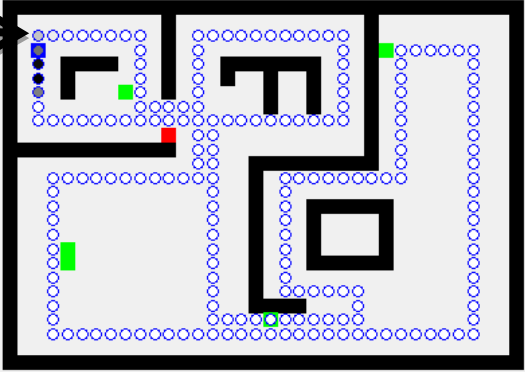
MATLAB CODE
<pre> function [A] = p_mapping(P,A) %P: Probability of move &amp; A: Initial/Final vector a=[0 A]; b=a; for n=2:1:length(a)     a(n)=(b(n-1)*P)+(b(n)*(1-P)); end if A(length(A))&gt;0     a(length(a))=1-sum(a(1:(length(a)-1))); end A=a(2:length(a)); </pre>

### 3.6 Localization to optimize the movement of robot

The methodology that is adopted for optimization is by segregating the whole map in to small portions. In our application whole home map is divided into separate rooms. The probability of human in each room is calculated and based on that robot moves towards

defined default location. This technique is implemented so that when human calls the robot it reaches to human in less time. The code that is used for optimization is given in table 3.

Table 3: Optimization Coding

MATLAB CODE	
<pre>function [Room] = RoomLocation (p_vector)  %p_vector is the path of human in terms of probability  %Defining rooms R1=[17 16 15 ... 184 49 48]; R2=[47 46 45 ... 20 19 18]; R3=[50 51 52 ... 181 182 183]; R4=[90 91 92 ... 163 164 165];  sum1=sum(p_vector(R1)); sum2=sum(p_vector(R2)); sum3=sum(p_vector(R3)); sum4=sum(p_vector(R4));  sumsum=[sum1 sum2 sum3 sum4];  %Defining the most probable location of human [V,Room]=max(sumsum);</pre>	
<p><b>Note:</b> Robot will move towards default room location after determining the probable location of target.</p>	

The above function determines the room location; Probability vector of human is the input of this function. Rooms are defined in this function initially, afterwards the sum of all the probability in the specified rooms are determined. The maximum value of the sum indicates the location of human. After determining the room, robot moves to default room location to achieve maximum award by reaching towards human in less time.

### 3.7 Call interrupt by Human

In order to call robot human commands the robot to come; this scenario is represented in simulation by a button as shown in the following Figure 33.



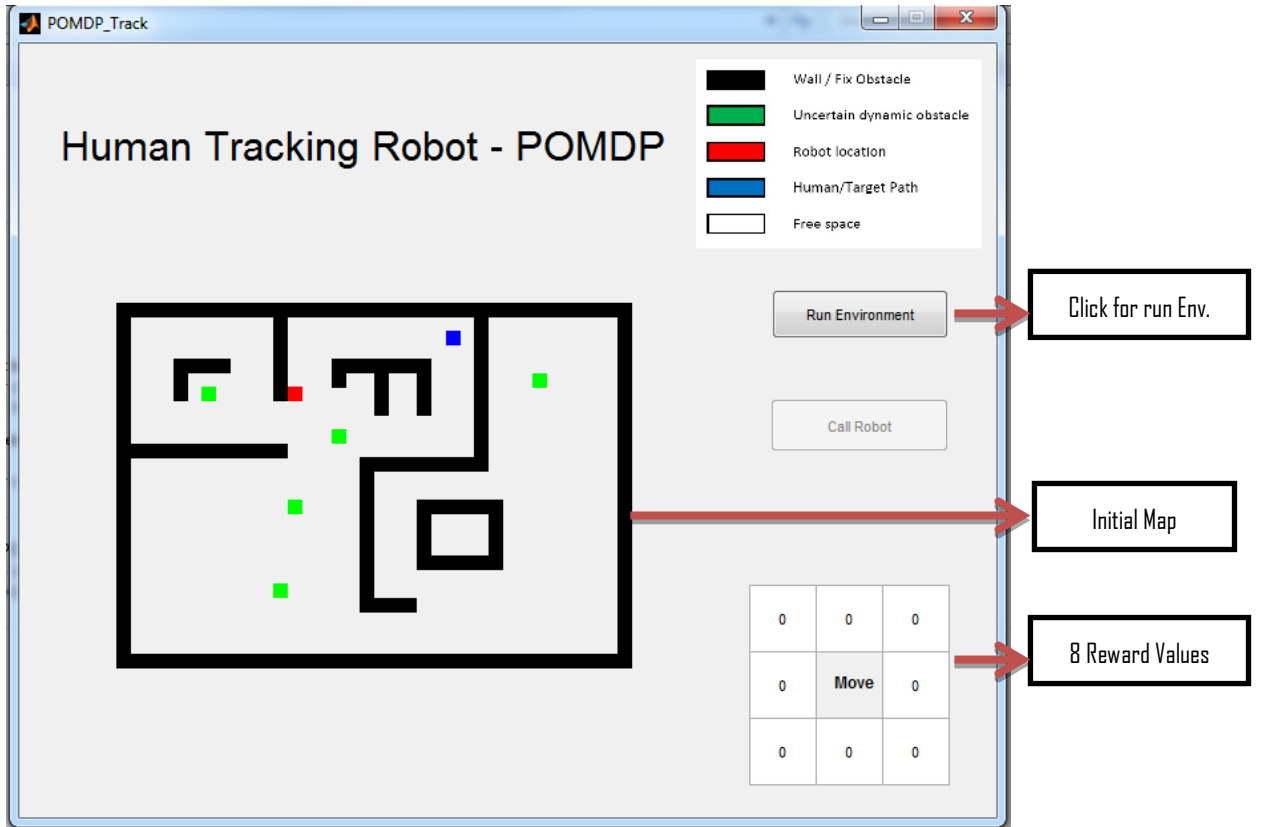


Figure 33: Simulation GUI in MATLAB

After pressing button to run the environment, a probability grid is created. Values are assigned on the basis of probability distribution function. The methodology adopted while human calls the robot is described in the form of flow chart as shown in the Figure 34.

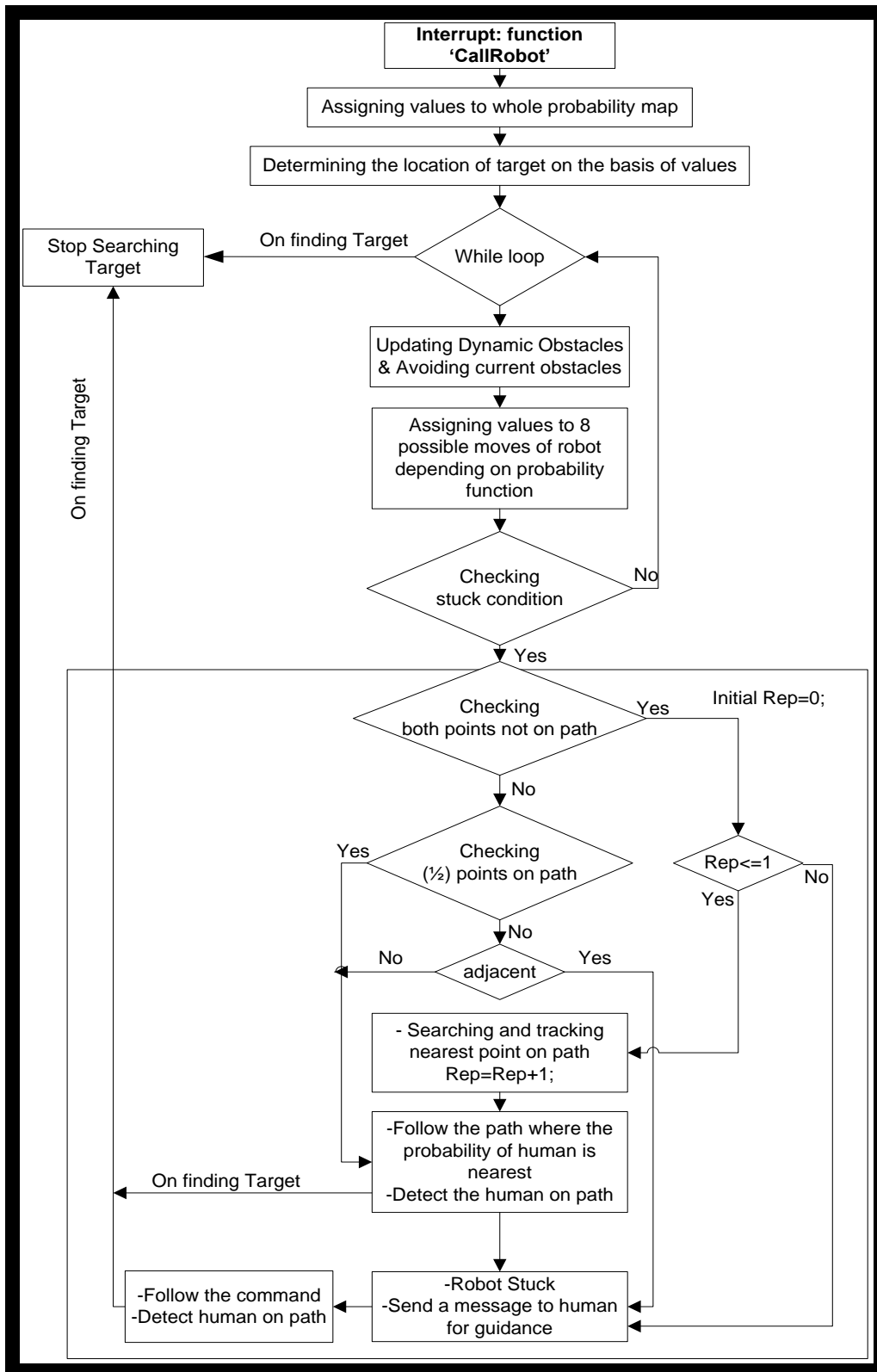


Figure 34: Interrupt Flow chat

In the Figure 34, when human calls the robot; a probability map is created. Values are then assigned to each element in the probability map that will determine the location of the target. After determining the location of the target, a while loop is configured that will end on

finding the target. Robot will move while avoiding the obstacles, in its each move it will assign values to eight adjacent cells of the grid depending on the probability function. Furthermore, robot will also check the stuck condition. In stuck condition, the Robot will find the nearest point on the human track and divide the track in two portions. After dividing the path in two portions, robot will determine the human location based on probability and follow the path where the probability of human is greater.

## CHAPTER 4: IMPLEMENTATION & EXPERIMENTAL RESULTS

In this chapter MATLAB coding and simulation results are presented to understand full picture of human tracking robot navigation implementation.

The GUI interface of the robot environment is shown in the following figure 35:

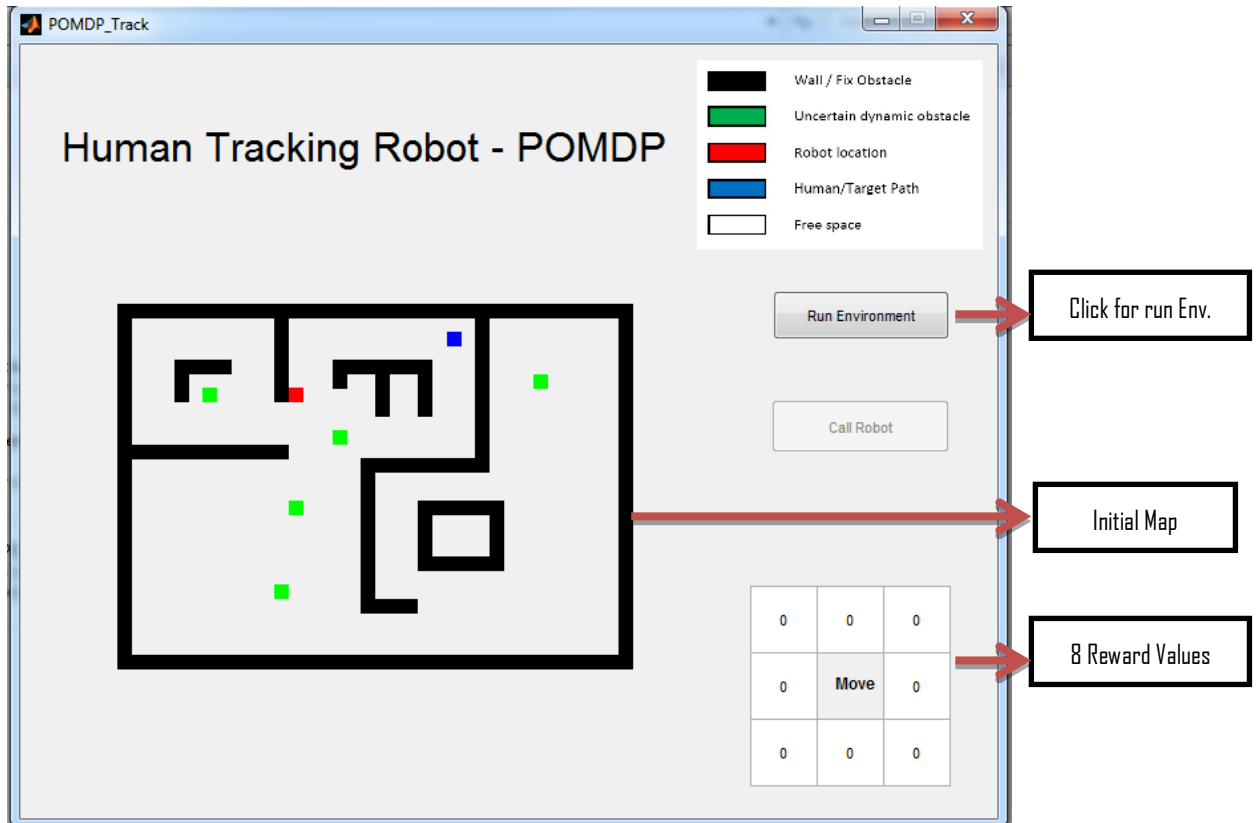


Figure 35: GUI Interface (MATLAB)

The figure 35 showed the initial map of a home. Black indicates the walls of the home, green are the dynamic obstacles, red indicates the robot location and blue dot indicates the human actual position. Rest of the grid in light grey is the free space for human and robot to travel. By clicking the “Run Environment” button as shown in the Figure 35, the program will execute and the human will move in the environment randomly in free space. After clicking this above mentioned button, another button “Call robot” is enabled so that user can click the button if human needs the robot. In actual environment the human can give a command to robot to come by a wireless connection or some other source. In this simulation the method of obtaining the command signals from the human is not mentioned rather focusing on the navigation technique that is implemented in this research. Moreover, the

values / rewards obtained on adjacent cells of the current robot's cell will appear in GUI while robot is tracking / searching the human on call.

When "Run Environment" is clicked the following GUI will appear:



Figure 36: Initial map after running the Environment

The human (In blue dot) moves randomly on the path (empty dots) are the path grid points in which the human moves (Figure 36). Each step of the human is not deterministic and uncertain; the uncertainty of the human in MATLAB simulation is defined by the formula as shown in the Table 4.

Table 4: MATLAB code for Human Uncertainty

```
%Defining percentage of robot move i.e. 90 percent
rmov=randi([1 10],1,1)>1;
%Defining percentage of robot move i.e. 50 percent
%rmov=randi([1 2],1,1)>1;
```

Table 4 depicts the MATLAB keyword command "randi" is the random integer command that generates a random integer between 1 and 10. After that "<" command will returns a value of 1 or 0 depending on the integer that MATLAB has selected randomly. In this whole syntax there is 0.9 probability of human to move ahead in next cell. Furthermore, for the human probability to move in the next grid point with the probability 0.5 is also shown in of the MATLAB code in the last two lines.

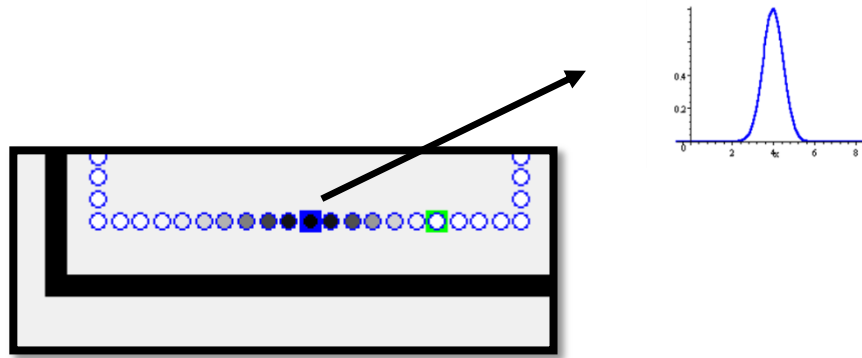


Figure 37: Probability Distribution Function

In the Figure 37, a probability distribution function of the human in the whole track / path is defined. The darker the dot, the more will be the probability of human in that cell. The grey scale coloring is carried out through MATLAB commands as shown in Table 5. The purpose of grey scale implementation in coding is for the visual differentiation so that the user can easily see the probability values changings.

Table 5: Grey scale for visual differentiation

```
greyscale = abs(p_vector./max(p_vector));
%Plotting visually points of probability function
for points=1:1:198
plot(xt(points),yt(points),'o','markerfacecolor',[1 1 1]*...
...(1-greyscale(points)))
End
```

The localization technique that is used in this research is to optimize the movement of robot in order to achieve maximum award value. In this technique the robot moves to defined robot locations with respect to the movement by human in perspective room. As the human changes the room the robot moves from one default location to other default location. This movement does not depends weather the human calls the robot or not.

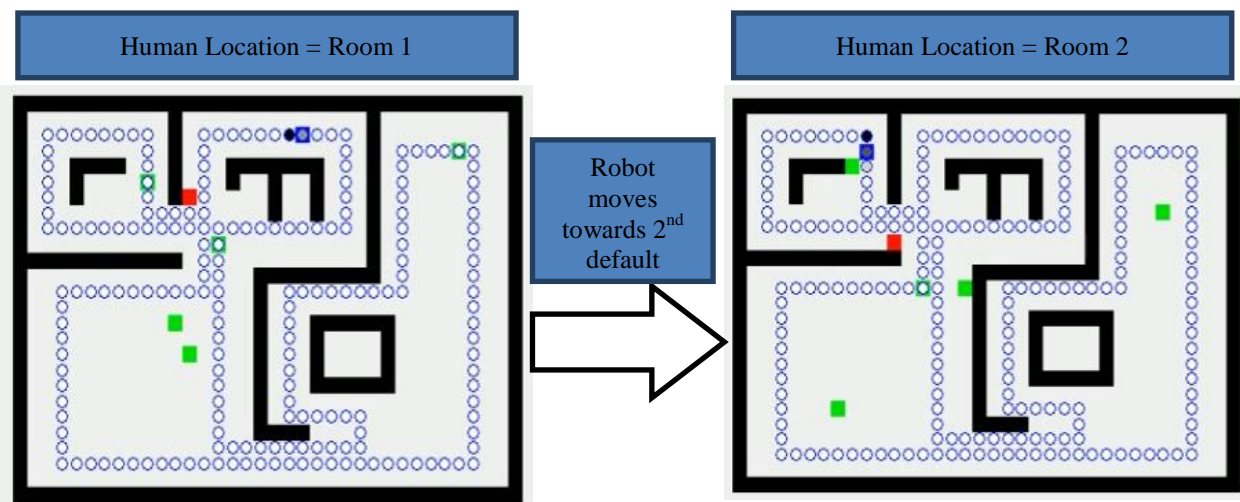


Figure 38: Robot default location changes w.r.t human location for optimization

In the Figure 38, the picture on the left is showing the human in Room 1 location and robot is in its first default location. In the right Figure 38, the robot changes its default position when the human travels from Room 1 location to Room 2 location. The default location of the robot changes with respect to human location in rooms. The probability distribution function of human's path decides the probable room where human is located. The methodology of the probability distribution function is already defined in the chapter 3.

#### 4.1 Computing Belief Space

Belief state 'b' is basically a probability distribution function 'S'. Let b (s) is the probability assigned to the human track.

Where  $0 \leq b(s) \leq 1$  for all  $s \in S$

$$\sum_{s \in S} b(s) = 1 \quad \text{----- (Eq: 9.0)}$$

$b'$  is the new belief state that originated from old belief state through state estimator SE with an action "a" and an observation "o".

$$b'(s') = \Pr(s' | o, a, b)$$

$$\begin{aligned} &= \frac{\Pr(o | s', a, b) \Pr(s' | a, b)}{\Pr(o | a, b)} \\ &= \frac{\Pr(o | s', a) \sum_{s \in S} \Pr(s' | a, b, s) \Pr(s | a, b)}{\Pr(o | a, b)} \\ &= \frac{O(s', a, o) \sum_{s \in S} T(s, a, s') b(s)}{\Pr(o | a, b)} \quad \text{----- (Eq: 10.0)} \end{aligned}$$

The above formula is implemented in MATLAB to create probability distribution function as shown in the Table6.

Table 6: Creating Belief Space

```

for n=2:1:length(a)
    a(n)=(b(n-1)*P)+(b(n)*(1-P));
end
if A(length(A))>0
    a(length(a))=1-sum(a(1:(length(a)-1)));
end

```

The step by step results generated by the above MATLAB code of the probability distribution function are as follows:

```
A = [1 0 0 0 0 0]
>> A=p_mapping (0.7, A )
A =[ 0.3000 0.7000 0 0 0 0]
>> A=p_mapping (0.7, A )
A = [0.0900 0.4200 0.4900 0 0 0]
>> A = p_mapping (0.7, A )
A = [0.0270 0.1890 0.4410 0.3430 0 0]
```

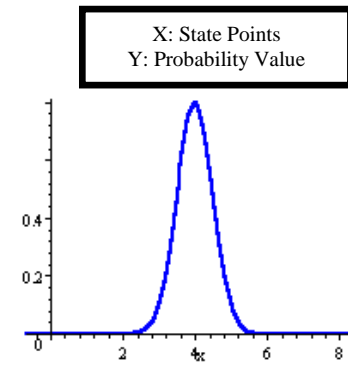


Figure 32: Distribution Function

Table 7: Code Iterations

<b>Iteration 1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>Iteration 2</b>	<b>0.3</b>	<b>0.7</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>Iteration 3</b>	<b>0.09</b>	<b>0.42</b>	<b>0.49</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>Iteration 4</b>	<b>0.027</b>	<b>0.189</b>	<b>0.441</b>	<b>0.343</b>	<b>0</b>	<b>0</b>
...	...	...	...	...	...	...

In the above table 7 the iterations are shown in which human's probability at a cell is initially 100%. After that the probability of human varies on each step as described above.

## 4.2 Defining Localization, Optimum Policy and Value Function

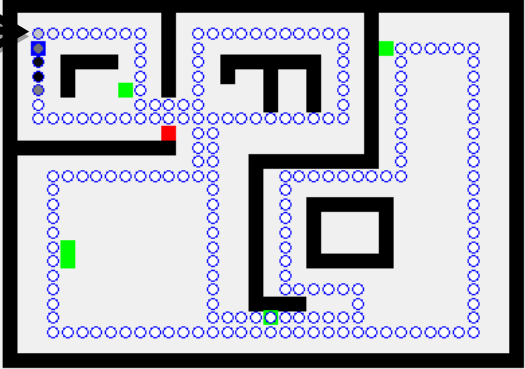
The optimization policy and Localization is the basic phenomena that help the robot to achieve maximum award in less time. For localization the technique that is implemented is describe in the MATLAB code in the Table 8. Robot first analyzes the probability of whole grid of the human path and then sums the probabilities of the grid within the room. Where the probability sum comes out to be one is the probable location of human in that room. The formula of calculating the probability sum is as shown as follows:

$$\sum \Pr(b'|a, b, o) \quad \text{----- (Eq: 11.0)}$$

The MATLAB code is shown in the Table 8 with the results obtained by the above formula is also shown after the table in the Example. The example that is shown after the Table 8 is when the human location is probably in room 2.



Table 8: Localization

MATLAB CODE	
<pre>function [Room] = RoomLocation (p_vector)  %p_vector is the path of human in terms of probability  %Defining rooms R1=[17 16 15 ... 184 49 48]; R2=[47 46 45 ... 20 19 18]; R3=[50 51 52 ... 181 182 183]; R4=[90 91 92 ... 163 164 165];  sum1=sum(p_vector(R1)); sum2=sum(p_vector(R2)); sum3=sum(p_vector(R3)); sum4=sum(p_vector(R4));  sumsum=[sum1 sum2 sum3 sum4];  %Defining the most probable location of human [V,Room]=max(sumsum);</pre>	
<p><b>Note:</b> Robot will move towards default room location after determining the probable location of target.</p>	

**Example:**

```
sum1 = 0
sum2 = 1
sum3 = 0
sum4 = 0
sumsum = [0 1 0 0]
V = 1
Room = 2
```

0
0
0
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0002
0.0010
0.0052
0.0218
0.0700
0.1680
0.2835
0.3002
0.1501
0
0
0

Actions that the robot can take with respect to human location and on calling are eight. The eight robots actions are shown in the Figure 39.

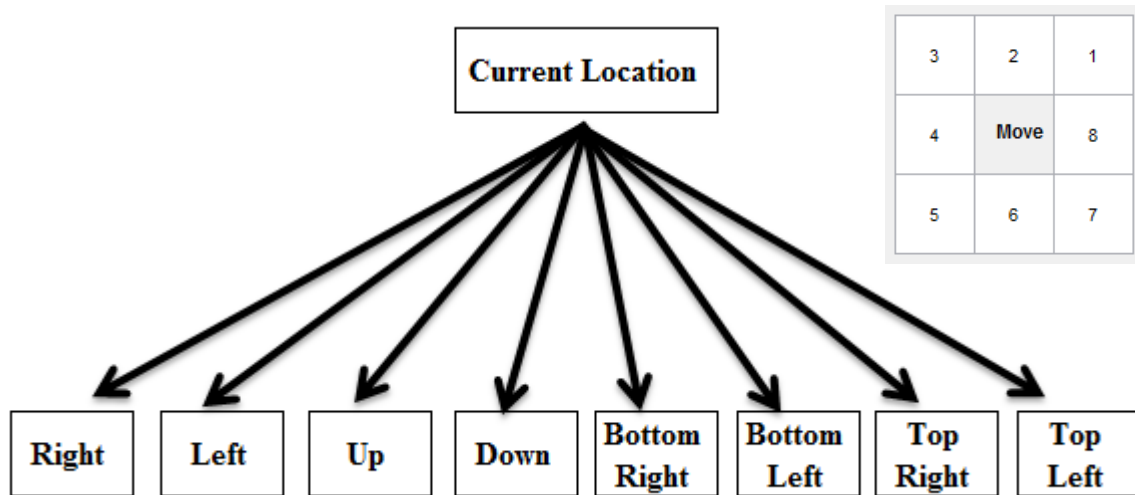


Figure 39: Robot Actions

In these adjacent grid points the robot will only move towards the maximum reward value. The policy that is made for the movement of robot is based on distance probability. Distance probability means that robot will calculate the distance from its own probable position and human probable position with a linear function. In initial parameters the robot learning is carried out so that the map of the home is recognized by robot. With this map the robot finds the distance between human and robot probable locations. In MATLAB this distance command is implemented by “pdist” as shown in the Table9.

For initialization the some initial reward is given to whole grid points in order to differentiate the reward value of moving towards obstacles or side walls. Here the initial award is taken as follows:

$$R ( s , a ) = 0.01 \text{ (All empty grid points)}$$

$$R ( s , a ) = 0 \text{ (Obstacles / Walls)}$$

The reward optimum policy of this human tracking robot is defined as follows:

1. If the probable calculated distance between the robot and human is less as compared to previous calculated probable distance then the reward value is 0.04 (4%).
2. If the probable calculated distance between the robot and human is greater as compared to previous calculated probable distance then penalty of 4% will be given. i.e. “-0.04”.
3. Furthermore, if the probable calculated distance between the robot and human is same then a less penalty is given i.e. 3% (-0.03).

This policy is summarized in a Table 9 as follows:

Table 9: Policy Chart

Probable Distance “Pdist”	Policy	
	Reward	Penalty
Less	4 % (+0.04)	0 %
Greater	0 %	- 4 % (-0.04)
Equal	0 %	- 3 % (-0.03)

This above value is further sum with the belief state space calculated in the previous section. The combine formula of the reward value with the above mentioned policy is as follows:

$$V = R ( s , a ) + \gamma \sum \text{Pr} ( b' | a , b , o ) \quad \text{----- (Eq: 12.0)}$$

The code for the implementation of the above formula is shown in the Table 10 as follows:

Table 10: Policy Code

MATLAB Code
<pre> t_map=zeros([26,36])+0.01; p_vector = 0.01+evalin('base','p_vector'); .... for i=1:1:198 t_map(xt(i),yt(i))= p_vector(i); end .... elseifpdist([RxN(i) RyN(i);PORxPORy]) &lt;pdist([Rx Ry;PORxPORy])  PN(i)=t_map(RyN(i),RxN(i))+0.04; <b>% Near Human +4% Reward</b> elseifpdist([RxN(i) RyN(i);PORxPORy]) &gt;pdist([Rx Ry;PORxPORy]) PN(i)=t_map(RyN(i),RxN(i))-0.04; <b>% Away from Human -4% Penalty</b> elseifpdist([RxN(i) RyN(i);PORxPORy]) == pdist([Rx Ry;PORxPORy]) PN(i)=t_map(RyN(i),RxN(i))-0.03; <b>% May be Stuck Condition -3% Penalty</b> </pre>

After obtaining the eight reward values the robot will take an action on the maximum awarded value as shown in the following formula:

$$V^*(s) = \max [ R(s, a) + \gamma \sum \Pr(b' | a, b, o) ] \text{ ----- (Eq: 13.0)}$$

Where  $V^*(s)$  is the optimum value that is the maximum value among the eight adjacent cell values of the robot. The values variations and the step by step implementation of the above algorithm are shown in an example below:

Suppose a human call a robot at the location shown in Figure 40. In this figure the robot is present at the location mentioned as '1'. Blue dot is the location of the human. The algorithm runs and the robot travel from location 1 to location 10. Step by step variation according to the algorithm defined above is shown after the Figure 40.

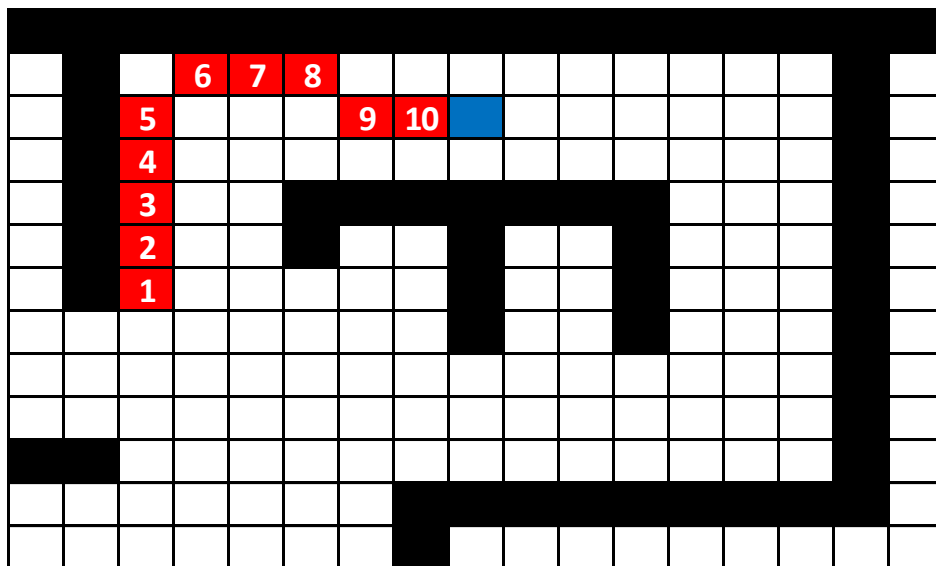
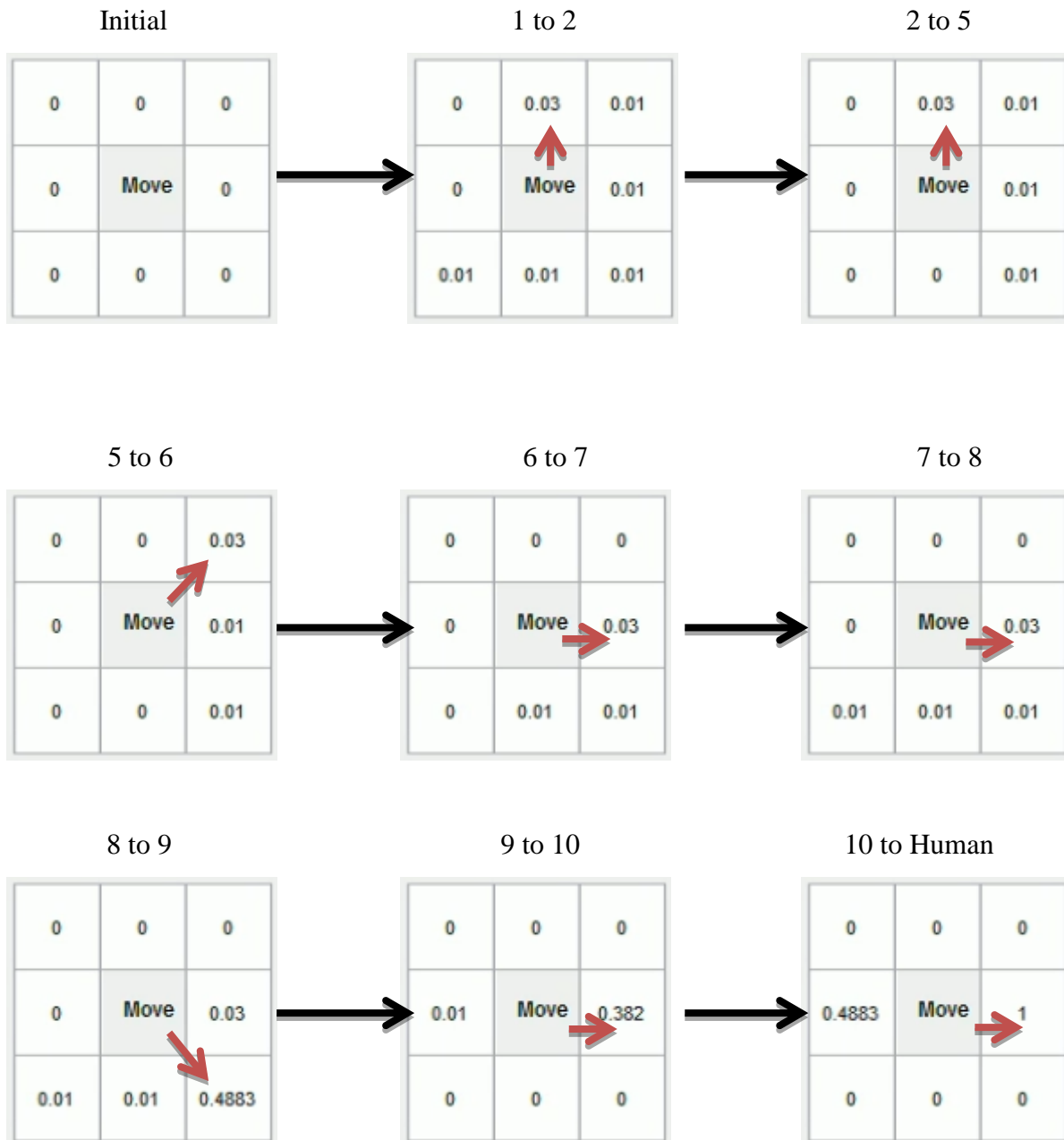


Figure 40: Example



It is clear from the above flow chart that initially before starting the values of the eight adjacent cells are zero. After that initial award is assign as 0.01 except walls and obstacles. The POMDP algorithm runs and the robot will move towards the maximum value of the cell to reach the human. Finally when robot reaches the human, detects the human and get maximum award as 1.

### 4.3 Comparison of Simple POMDP & Localized-POMDP

POMDP technique required lot of computational power. In this technique policy tree that is created to track the target contain all the probabilistic points of the map. In Localized-POMDP technique the probabilistic points are reduced. Reduction is based on number of sections in a map. Simple POMDP provides a complete tracking of target to some extent, while the localized POMDP is not a complete solution if the probability varies drastically. Following in a comparison table the computational power is roughly calculated to give an idea of its optimality.

Table 11: Comparison Table

S. No.	Aspects	Methods	
		POMDP	Localized-POMDP
1.	Formula (Value Function)	$V=R(s,a)+r \sum_{i=1}^n \text{Pr}(b' a, b, o)$ Where n = Total map probabilistic points	$V=R(s,a)+r \sum_{i=1}^{(n1,n2...ns)} \text{Pr}(b' a, b, o)$ Where n = no of points in 'x' region
2.	Computation Power	More data points, more computational power	Less data points, less computational power
3.	Coverage	Full Coverage (Completeness)	In complete (If the probability varies drastically)
4.	Algorithm Computational power calculations	Depends on total map probabilistic points. For example: 36 x 26	If human is in room 1: 14 x 11 Enhancement in computational power = $\frac{36 \times 26 - 14 \times 11}{36 \times 26} \times 100$ 83.5% More efficient

## CHAPTER 5: CONCLUSION & FUTURE RECOMMENDATIONS

This Localized POMDP technique is helpful for the robot and the target operated in a grid environment efficiently. Furthermore in this technique, navigation is combined with the robot learning phase that helps robot to learn and update data to determine expected location with less error. As the time spends the robot action judgments becomes accurate by its observations with respect to time and the human belief also flourished. This is the major and important technique by which the uncertainty of mobile robots can be solved. This penalty and reward game helps robot to navigate accurately in the given map. The robot pays a cost for each move and receives a reward every time it arrives in the same position as that of the target. This technique continues until the robot reaches the required human target. Moreover it is useful for the uncertain changings in the environment and even human's motion is non-deterministic. The factors that add the uncertainty in the area of mobile robot are the uncertain movement of target, robot's accuracy while moving in steps and dynamic obstacles that comes in the way while tracking. Moreover, it is an optimized way in which the robot stays closer to the person in order to track his position well and improve the chance of receiving rewards. The robot automatically moves and changes it position with respect to the human observed location to get maximum value in lesser time. Other advantage of using this technique is to minimize movement in order to reduce power consume by the robot. Battery consumption for mobile robot is very important in today's life because it is very time consuming to charge batteries before it goes to work again in a given environment.

To enhance the generalization of this algorithm and navigation optimization of the human tracking robot a generalized algorithm can be defined with this project that can auto generate robot's default locations perspective to different rooms in the given map. The default locations of the rooms should be taken at minimum distance to avoid the unnecessary movements for the implementation of the above mentioned generalization technique. Any technique can be merged with this technique to auto calculate the default locations of the robot. This project methodology is generalized that can easily be merged with any other navigation technique. To make this methodology more efficient the default location can be defined in such a way that the distance between two consecutive locations are least.

The POMDP technique provides a basic approach of planning under non deterministic environment with observations from surroundings. It gives a best way of taking an action after gaining information from the world. Our current work explores the use of localized

POMDP method for representing best way of obtaining values/awards and the use of simulation in order to concentrate the approximations on rapid changing environment through belief space. The results of this work have allowed us to have a good way of optimizing robot navigation. We are hopeful to extend this technique to get good solution to our variety of daily life applications.



## APPENDIX A - MATLAB CODE

```
function varargout = POMDP_Track(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @POMDP_Track_OpeningFcn, ...
'gui_OutputFcn',  @POMDP_Track_OutputFcn, ...
'gui_LayoutFcn',  [], ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function POMDP_Track_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

% Display pic on GUI
axes(handles.DetailofMap)
assignin('base','im_name' , 'DetailofMAP.jpg');
assignin('base','im_type' , 'jpg');
evalin('base','[IMG,MAP] = imread(im_name,im_type);');
IMG = evalin('base','IMG');
image(IMG);
axis off

guidata(hObject, handles);

function varargout = POMDP_Track_OutputFcn(hObject, eventdata, handles)

%Defining environment
axes(handles.AxesMap)
axis off
hold off;

%Creating Wall
xw1(1:36)=1:1:36;
xw1(37:61)=36;
xw1(62:96)=35:-1:1;
xw1(97:120)=1;

yw1(1:36)=1;
yw1(37:61)=2:1:26;
yw1(62:96)=26;
yw1(97:120)=25:-1:2;

%Creating Fix Obstacle
xo1(1:6)=22:1:27;
xo1(7:10)=27;
xo1(11:15)=26:-1:22;
xo1(16:18)=22;
```

```

yo1(1:6)=8;
yo1(7:10)=9:1:12;
yo1(11:15)=12;
yo1(16:18)=11:-1:9;

%Defining all obs excluding dynamic obs
xw=[xw1 xw2 xw3 xw4 xo1 xo2 xo3];
yw=[yw1 yw2 yw3 yw4 yo1 yo2 yo3];

plot(xw,yw,'s','MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize',9)
axis([-3 40 -3 30])
axis off

hold on

%Creating Uncertain dynamic obstacle
xdo1=30;
ydo1=21;

%All uncertain dynamic obs
xdo=[xdo1 xdo2 xdo3 xdo4 xdo5];
ydo=[ydo1 ydo2 ydo3 ydo4 ydo5];

do_plot=plot(xdo,ydo,'s','MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',9);
axis([-3 40 -3 30])

%Defining robot location
Rx=13;
Ry=20;
rplot=plot(Rx,Ry,'rs','MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',9);
assignin('base','rplot',rplot);

%Defining Current Obstacles overall
CObs=[xw xdo Rx; yw ydo Ry];

% Target Random probabilistic path
xt= [24 23 22 21 20 19 18 17 16 15 14 14 14 14 14 14 13 12 11 10 10 10 10
...
yt= [ 24 24 24 24 24 24 24 24 24 24 24 24 23 22 21 20 19 19 19 19 19 20 ...

t_plot=plot(xt(1),yt(1),'s','MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',9);

%Sending variables to base
assignin('base','xw',xw);
assignin('base','yw',yw);

assignin('base','xdo',xdo);
assignin('base','ydo',ydo);

assignin('base','CObs',CObs);

assignin('base','xt',xt);
assignin('base','yt',yt);

```

```

assignin('base','do_plot',do_plot);
assignin('base','t_plot',t_plot);

assignin('base','Rx',Rx);
assignin('base','Ry',Ry);

varargout{1} = handles.output;

function Run_Callback(hObject, eventdata, handles)

pm1=0;pm2=0;pm3=0;pm4=0;pm5=0;
sr1=0;sr2=0;sr3=0;sr4=0;sr5=0;

Room=1;TransRoomCase=1;
assignin('base','TransRoomCase',TransRoomCase);

set(handles.Run,'Enable','off')

set(handles.CallRobot,'Enable','on')
% Defining Path for robot travel to optimize battery power consumption...
% ...and obtaining maximum reward
% Obtaining values from base

xw=evalin('base','xw');
yw=evalin('base','yw');

xdo=evalin('base','xdo');
ydo=evalin('base','ydo');

CObs=evalin('base','CObs');

xt=evalin('base','xt');
yt=evalin('base','yt');

t_plot=evalin('base','t_plot');

Rx=evalin('base','Rx');
Ry=evalin('base','Ry');

pos=1;step=1;
p_vector=[1 zeros([1,197])];
assignin('base','p_vector',p_vector);

axes(handles.AxesMap)

while pos < 198

% Dynamic Obs avoiding current obs
[xdo, ydo]=UpdateDynamicObs (xdo, ydo,CObs);

%Deleting previous dynamic obs
do_plot=evalin('base','do_plot');
delete(do_plot)

do_plot=plot(xdo,ydo,'s','MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',9);

```

```

    assignin('base','do_plot',do_plot);

%Updating current Obs
CObs=[xw xdo Rx; yw ydo Ry];
assignin('base','CObs',CObs);
assignin('base','xdo',xdo);
assignin('base','ydo',ydo);

%Defining percentage of robot move i.e. 90 percent
%   pv=100/(100-Percent)
    pv=6;
    rmov=randi([1 pv],1,1)>1;
%Defining percentage of robot move i.e. 50 percent
%   rmov=randi([1 2],1,1)>1;

    pos=pos+rmov;
    assignin('base','pos',pos);

%Checking the possibility of target move
    TMove=find((xdo == xt(pos) & ydo == yt(pos)),1);
if any(TMove)
    pos=pos-1;
    assignin('base','pos',pos);
end

    Rx=evalin('base','Rx');
    Ry=evalin('base','Ry');

    pos=evalin('base','pos');

%Deleting Target plot and then updating it
delete(t_plot)
t_plot=plot(xt(pos),yt(pos),'s','MarkerFaceColor','b','MarkerSize',9);
assignin('base','pos',pos);
assignin('base','t_plot',t_plot);

    p_vector=evalin('base','p_vector');

%creating pobability vector with percentage of move
p_vector=p_mapping(0.7,p_vector);
assignin('base','p_vector',p_vector);

%If robot find its target it update the probability vector to 1
if (xt(pos) == Rx && yt (pos)== Ry) || (xt(pos) == Rx+1 && yt (pos)== Ry+1)
|| (xt(pos) == Rx && yt (pos)== Ry+1)|| (xt(pos) == Rx-1 && yt (pos)==
Ry+1) || (xt(pos) == Rx-1 && yt (pos)== Ry) || (xt(pos) == Rx-1 && yt
(pos)== Ry-1) || (xt(pos) == Rx && yt (pos)== Ry-1) || (xt(pos) == Rx+1 &&
yt (pos)== Ry-1) || (xt(pos) == Rx+1 && yt (pos)== Ry)
    p_vector=zeros([1,198]);
    p_vector(pos)=1;
    assignin('base','p_vector',p_vector);
end

PreRoom=Room;

[Room]=RoomLocation (p_vector);

NewRoom=Room;

```

```

    TransRoomCase=evalin('base','TransRoomCase');

    PreCase=TransRoomCase;

if PreRoom ==1 && NewRoom ==2
    TransRoomCase=2;
    assignin('base','TransRoomCase',TransRoomCase);
end

if PreRoom ==2 && (NewRoom ==3 || NewRoom == 1 )
    TransRoomCase=3;
    assignin('base','TransRoomCase',TransRoomCase);
end

if PreRoom ==3 && NewRoom ==4
    TransRoomCase=4;
    assignin('base','TransRoomCase',TransRoomCase);
end

if PreRoom ==4 && NewRoom ==3
    TransRoomCase=5;
    assignin('base','TransRoomCase',TransRoomCase);
end

if PreRoom ==3 && NewRoom ==1
    TransRoomCase=6;
    assignin('base','TransRoomCase',TransRoomCase);
end

if PreCase==7 && PreCase ~= TransRoomCase
    TransRoomCase=8;
    assignin('base','TransRoomCase',TransRoomCase);
end

    greyscale = abs(p_vector./max(p_vector));

%Plotting visually points of probability function
for points=1:1:198
    plot(xt(points),yt(points),'o','markerfacecolor',[1 1 1]*(1-
greyscale(points)))
end

% Robot new location programming
switch TransRoomCase
case 2
if pml < length(xRobOneToTwoPath)
    rmov=randi([1 10],1,1)> 1;
    pml=pml+rmov;
if pml ~= 0
    RMove=find(xdo== xRobOneToTwoPath(pml) & ydo ==
yRobOneToTwoPath(pml),1);
if any(RMove)
    pml=pml-1;
end
    Rx=xRobOneToTwoPath(pml);
    Ry=yRobOneToTwoPath(pml);
end
end

```

```

case 3
if pm2 < length(xRobTwoToThreePath)
    rmov=randi([1 10],1,1)> 1;
    pm2=pm2+rmov;
if pm2 ~= 0
    RMove=find(xdo== xRobTwoToThreePath(pm2) & ydo ==
yRobTwoToThreePath(pm2),1);
if any(RMove)
    pm2=pm2-1;
end
    Rx=xRobTwoToThreePath(pm2);
    Ry=yRobTwoToThreePath(pm2);
end
end

case 4
if pm3 < length(xRobThreeToFourPath)
    rmov=randi([1 10],1,1)> 1;
    pm3=pm3+rmov;
if pm3 ~= 0
    RMove=find(xdo== xRobThreeToFourPath(pm3) & ydo ==
yRobThreeToFourPath(pm3),1);
if any(RMove)
    pm3=pm3-1;
end
    Rx=xRobThreeToFourPath(pm3);
    Ry=yRobThreeToFourPath(pm3);
end
end

case 5
if pm4 < length(xRobFourToThreePath)
    rmov=randi([1 10],1,1)> 1;
    pm4=pm4+rmov;
if pm4 ~= 0
    RMove=find(xdo== xRobFourToThreePath(pm4) & ydo ==
yRobFourToThreePath(pm4),1);
if any(RMove)
    pm4=pm4-1;
end
    Rx=xRobFourToThreePath(pm4);
    Ry=yRobFourToThreePath(pm4);
end
end

case 6
if pm5 < length(xRobThreeToOnePath)
    rmov=randi([1 10],1,1)> 1;
    pm5=pm5+rmov;
if pm5 ~= 0
    RMove=find(xdo== xRobThreeToOnePath(pm5) & ydo ==
yRobThreeToOnePath(pm5),1);
if any(RMove)
    pm5=pm5-1;
end
    Rx=xRobThreeToOnePath(pm5);
    Ry=yRobThreeToOnePath(pm5);
end
end

```

```

case 7
    Rx=xt(pos-1);
    Ry=yt(pos-1);

case 8
if Rx == 14 && Ry == 19
    SubRoute=1;
end
if (Rx == 11 || Rx==12) && Ry == 18
    SubRoute=2;
end
if (Rx == 18 || Rx==19) && Ry == 3
    SubRoute=3;
end
if Rx == 21 && Ry == 4
    SubRoute=4;
end
if Rx == 15 && Ry == 16
    SubRoute=5;
end

assignin('base','Rx',Rx);
assignin('base','Ry',Ry);

rplot=evalin('base','rplot');
delete(rplot)
rplot=plot(Rx,Ry,'rs','MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',9);
assignin('base','rplot',rplot);

step=step+1;

%Time delay for visual inspection units:sec
pause(1)
end
set(handles.Run,'Enable','on')

function CallRobot_Callback(hObject, eventdata, handles)
t_map=zeros([26,36])+0.01;
PN=zeros(1,8);
% U=0;L=0;D=0;R=0;
RxPath=[0 0 0 0 0];
RyPath=[0 0 0 0 0];
var=1;
p_vector = 0.02+evalin('base','p_vector');
xt=evalin('base','xt');
yt=evalin('base','yt');
Rx=evalin('base','Rx');
Ry=evalin('base','Ry');
CObs=evalin('base','CObs');
rplot=evalin('base','rplot');
pos=evalin('base','pos');
xdo=evalin('base','xdo');
ydo=evalin('base','ydo');
do_plot=evalin('base','do_plot');
xw=evalin('base','xw');
yw=evalin('base','yw');

%Updating values of t_map w.r.t p_vector

```

```

for i=1:1:198
    t_map(xt(i),yt(i))= p_vector(i);
end

% Finding Probability Location of Robot POR
[p_vector_value , p_vector_indx]= max(p_vector);
PORx=xt(p_vector_indx);
PORy=yt(p_vector_indx);

Rep=0;

while max(PN)<1

% Dynamic Obs avoiding current obs
[xdo, ydo]=UpdateDynamicObs (xdo, ydo,CObs);

%Deleting previous dynamic obs
delete(do_plot)

do_plot=plot(xdo,ydo,'s','MarkerEdgeColor','g','MarkerFaceColor','g','Marke
rSize',9);
    assignin('base','do_plot',do_plot);

    assignin('base','xdo',xdo);
    assignin('base','ydo',ydo);

%Updating current Obs
CObs=[xw xdo Rx; yw ydo Ry];
    assignin('base','CObs',CObs);

%Creating 8 points
RxN=[Rx+1 Rx Rx-1 Rx-1 Rx-1 Rx Rx+1 Rx+1];
RyN=[Ry+1 Ry+1 Ry+1 Ry Ry-1 Ry-1 Ry-1 Ry];

%Assigning rewards for eight locations
for i=1:1:8
if RxN(i)==xt(pos) && RyN(i)==yt(pos)
    PN(i)=1; %Human
    p_vector=zeros([1,198]);
    p_vector(pos)=1;
    assignin('base','p_vector',p_vector);
    t_map=zeros([26,36]);
    t_map(RxN(i),RyN(i))= 1;
elseif find((CObs(1,:) == RxN(i) & CObs(2,:) == RyN(i)),1);
    PN(i)=0; %Wall or Obstacle
    t_map(RxN(i),RyN(i))= 0;
elseif pdist([RxN(i) RyN(i);PORx PORy]) < pdist([Rx Ry;PORx PORy])
    PN(i)=t_map(RxN(i),RyN(i))+0.04; % Near Human
elseif pdist([RxN(i) RyN(i);PORx PORy]) > pdist([Rx Ry;PORx PORy])
    PN(i)=t_map(RxN(i),RyN(i))-0.04; %Away from Human
if PN(i) <= 0
        PN(i)=0.01;
end
elseif pdist([RxN(i) RyN(i);PORx PORy]) == pdist([Rx Ry;PORx PORy])
    PN(i)=t_map(RxN(i),RyN(i))-0.03;% May be Stuck Condition
if PN(i) <= 0
        PN(i)=0.01;
end
end
end

```



```

end
set(handles.one, 'String', round(PN(1)*10000)/10000)
set(handles.Two, 'String', round(PN(2)*10000)/10000)
set(handles.Three, 'String', round(PN(3)*10000)/10000)
set(handles.Four, 'String', round(PN(4)*10000)/10000)
set(handles.five, 'String', round(PN(5)*10000)/10000)
set(handles.Six, 'String', round(PN(6)*10000)/10000)
set(handles.Seven, 'String', round(PN(7)*10000)/10000)
set(handles.Eight, 'String', round(PN(8)*10000)/10000)

[PN_value , move]=max(PN);

if PN_value ~=1
    Rx=RxN(move);
    Ry=RyN(move);
else
    Rx=xt(pos-1);
    Ry=yt(pos-1);
end

t_map(RxN(move),RyN(move))= PN_value;

RxPath(5)=RxPath(4);
RxPath(4)=RxPath(3);
RxPath(3)=RxPath(2);
RxPath(2)=RxPath(1);
RxPath(1)=Rx;

RyPath(5)=RyPath(4);
RyPath(4)=RyPath(3);
RyPath(3)=RyPath(2);
RyPath(2)=RyPath(1);
RyPath(1)=Ry;

```

## REFERENCES

- [1] Galileo Educational Network GENA 1999-2003: “Robot introduction”
- [2] Robots Used in Everyday Life How by “Peter Staples”
- [3] German Research Center for Artificial Intelligence DFKI GmbH Robotics Innovation (RIC) : Fields of Application
- [4] Mobile Robot Navigation Final Report by Jonathan Dixon Oliver Henlich 10 June 1997, Imperial College, London
- [5] Robots are caring for elderly people in Europe written by Victoria Turk May 6, 2014.
- [6] DeSouza, G.N.; Kak, A.C., "Vision for mobile robot navigation: a survey," in Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.24, no.2, pp. 237–267, Feb 2002.
- [7] Probabilistic Navigation by Andrew Howard R. Bellman. A Markovian Decision Process. Journal of Mathematics and Mechanics 6, 1957
- [8] R. E. Bellman. Dynamic Programming. Princeton University Press, Princeton, NJ, 1957. Dover paperback edition (2003), ISBN 0-486-42809-5.
- [9] Ronald A. Howard Dynamic Programming and Markov Processes, The M.I.T. Press, 1960.
- [10] R. Bellman. A Markovian Decision Process. Journal of Mathematics and Mechanics 6, 1957.
- [11] R. E. Bellman. Dynamic Programming. Princeton University Press, Princeton, NJ, 1957. Dover paperback edition (2003), ISBN 0-486-42809-5.
- [12] Ronald A. Howard Dynamic Programming and Markov Processes, The M.I.T. Press, 1960.
- [13] D. Bertsekas. Dynamic Programming and Optimal Control. Volume 2, Athena, MA, 1995.
- [14] Burnetas, A.N. and M. N. Katehakis. "Optimal Adaptive Policies for Markov Decision Processes, Mathematics of Operations Research, 22,(1), 1995.
- [15] E.A. Feinberg and A. Shwartz (eds.) Handbook of Markov Decision Processes, Kluwer, Boston, MA, 2002.
- [16] C. Derman. Finite state Markovian decision processes, Academic Press, 1970.
- [17] M. L. Puterman. Markov Decision Processes. Wiley, 1994.
- [18] H.C. Tijms. A First Course in Stochastic Models. Wiley, 2003.

- [19] Sutton, R. S. and Barto A. G. Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA, 1998.
- [20] J.A. E. E van Nunen. A set of successive approximation methods for discounted Markovian decision problems. *Z. Operations Research*, 20:203-208, 1976.
- [21] S. P. Meyn, 2007. Control Techniques for Complex Networks, Cambridge University Press, 2007. ISBN 978-0-521-88441-9. Appendix contains abridged Meyn&Tweedie.
- [22] S. M. Ross. 1983. Introduction to stochastic dynamic programming. Academic press
- [23] X. Guo and O. Hernández-Lerma. Continuous-Time Markov Decision Processes, Springer, 2009.
- [24] M. L. Puterman and Shin M. C. Modified Policy Iteration Algorithms for Discounted Markov Decision Problems, *Management Science* 24, 1978.
- [25] *Jaulin, L. (2001). "Path planning using intervals and graphs".*
- [26] *Delanoue, N.; Jaulin, L.; Cottenceau, B. (2006). "Counting the Number of Connected Components of a Set and Its Application to Robotics". Applied Parallel Computing.*
- [27] *Hsu, D.; J.C. Latombe, J.C.; Motwani, R. (1997). "Path Planning in Expansive Configuration Spaces". Proc. IEEE Int. Conf. on Robotics and Automation.*
- [28] *Shvalb, N.; Ben Moshe, B.; Medina, O. (2013). "A real-time motion planning algorithm for a hyper-redundant set of mechanisms."*
- [29] *Zhang, Y. (2011). "UCAV path planning based on FSCABC". Information-An International Interdisciplinary Journal 14.*
- [30] Motion Planning “Maria Isabel Ribeiro Pedro Lima” Instituto de Sistemas e Robótica (ISR)
- [31] Planning and Acting in Partially Observable Stochastic Domains Michael L. Littman
- [32] Principles of Robot Motion Theory, Algorithms, and Implementations, Howie Choset, Kevin