International Master's Thesis

Mobile Robot Navigation using potential fields and market based optimization

Muhammad Abdullah
Technology

Studies from the Department of Technology at Örebro University
Örebro 2013

Mobile Robot Navigation using potential fields and market based optimization

Studies from the Department of Technology at Örebro University

Muhammad Abdullah

Mobile Robot Navigation using potential fields and market based optimization

Supervisors: **Abdelbaki Bouguerra**
Rainer Palm

Title: Mobile Robot Navigation using potential fields and market based optimization

ISSN 0000-0000

Abstract

A team of mobile robots moving in a shared area raises the problem of safe and autonomous navigation. While avoiding static and dynamic obstacles, mobile robots in a team can lead to complicated and irregular movements. Local reactive approaches are used to deal with situations where robots are moving in dynamic environment; these approaches help in safe navigation of robots but do not give optimal solution. In this work a 2-D navigation strategy is implemented, where a potential field method is used for obstacle avoidance. This potential field method is improved using fuzzy rules, traffic rules and market based optimization (MBO). Fuzzy rules are used to deform repulsive potential fields in the vicinity of obstacles. Traffic rules are used to deal situations where two robots are crossing each other. Market based optimization (MBO) is used to strengthen or weaken repulsive potential fields generated by other robots based on their importance. For the verification of this strategy on more realistic vehicles this navigation strategy is implemented and tested in simulation. Issues while implementing this method and limitations of this navigation strategy are also discussed. Extensive experiments are performed to examine the validity of MBO navigation strategy over traditional potential field (PF) method.

Acknowledgements

First and foremost, I would like to thank my supervisors Abdelbaki Bouguerra and Rainer palm for their much appreciated guidance and help during this thesis work. I wish to thank all my friends and classmates for providing support and friendship that I needed. I especially thank my parents and my entire family for their unconditional support and encouragement throughout my study. Lastly, I offer my regards to all of those who supported me during the completion of this thesis work.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1
Introduction

1.1 Motivation

Recently mobile robots have started to work in the real world scenarios. Applications of mobile robots are immense and acquiring importance. These applications include agricultural robotics (fertilizing and planting), support to medical services (transportation of medication), client supports (museum tour, exhibition guides) and military missions (surveillance and monitoring) etc [25].

A team of mobile robots can do work in parallel and has advantages over single robot systems. Multi mobile robot systems can complete a given task faster as compared to a single robot. In such tasks where multi mobile robots are involved, there is a requirement that all robots navigate and avoid each other to reach their goal positions. Multi mobile robot systems can be used for material transportation in factories, defense, agricultural robotics and service support.

Material handling involves driver-less vehicles transporting materials and goods to storage sites and workstations of the production facilities. For material handling AGVs (Automatic guided vehicles) are used and there are different navigation methods for AGVs [3]. Commercially used AGVs are guided by fixed buried cables or chemical strips painted on the factory floor [3]. These methods have limitations e.g. designing of these tracks can take more time and modifications can increase the expenses. In other alternative approaches placement of retro-reflector landmarks can be used for localization [6] and navigation can be done independently in the local frame of robots. In [23] an approach is discussed where flexible AGVs can operate in FMSs (Flexible manufacturing system), as flexible AGVs can react to small changes in environment.

Application of material transportation in a factory, the basic functionality of the system can be divided into task allocation and safe navigation. The approach used in [23] allocates tasks through a centralized system (base station) but uses a decentralized approach for collision-free motion. Every vehicle updates its position on the base station, gets task information from the base sta-

tion and uses a searching algorithm for calculating its trajectory. For collision free motion of vehicles in case of multiple vehicles a decentralized architecture is used in [23], where vehicles with low priority give way to other vehicles. In our work, the main focus is on the navigation of mobile robots in a shared area. Some other considerations are that every robot can accurately localize itself and can get other robots positions from the base station.

Several methods have been proposed for navigation of robots and obstacle avoidance. The potential field method is one of the famous methods for obstacle avoidance [15]. Advantages and disadvantages of the potential field method with respect to steadiness and dead-locks are discussed in [16]. A popular method to deal with obstacle avoidance is the fuzzy logic approach, where fuzzy rules are formulated based on the common sense of humans. In [19] and [27] fuzzy rules are implemented on system of mobile robot for obstacle avoidance.

In a team of mobile robots there is a need of coordination between team members while navigating in a shared area. Decentralized coordination methods like multi-agent control are expected to function on systems having large number of local systems more effectively than centralized methods [21]. For large decentralized systems market-based (MB) optimization is one of the most attractive and capable approaches. The MB approach is very similar to economical system where consumers and producers both compete and collaborate on a market of commodities.

Our main focus is in navigation of multiple mobile robots in a shared area. Several methods can be used to address this problem, as it needs some coordination. One method is proposed by [21] where potential field method is used for safe navigation of robots. This potential field method is optimized using a combination of fuzzy rules and MB optimization.

1.2 Problem statement

This work aims at improving the work proposed by Palm and Bouguerra [21] titled as 'Navigation of mobile robots by potential field methods and market based optimization'. Palm and Bouguerra proposed an approach for the autonomous navigation of team of mobile robots. The scope of their work is basically theoretical and in the simulation the weight, height, engine and wheels of the vehicles and the sensors have been neglected.

This thesis will help to answer the question 'whether the navigation using potential field methods and market-based optimization can be effectively applied on more realistic scenarios (i.e. where there are many robots acting in a shared area)?' For this purpose, implementation of this method can be tested in simulation using Gazebo [11]. As Gazebo is a 3D multi robot simulator, it provides an accurate model of reality.

Another question is 'How can we evaluate this method for navigation of a team of mobile robots?' or 'which parameters can be used for the evaluation of

multi-robot navigation?' In [5] some of the parameters are discussed that can be used for evaluation of motion tasks, but it's only for the motion evaluation of a single mobile robot. The same parameters can be used to find the combined result of a group of mobile robots while navigating.

The aim of this thesis work is also to discuss what problems and limitations we can face while applying this method? Another aim is the evaluation of this method when team of mobile robots is moving in a small area.

1.3 Contributions

- Implementation of a package of proposed algorithm in ROS (Robot operating system) that can be used on a real robot. It's a generic package in ROS and can be used for multiple robots. Issues while implementing the proposed method on more realistic simulation environment are discussed and limitations of this navigation method are exposed.
- Implementation of traffic rules for the purpose of safe navigation. In the proposed navigation method two traffic rules (slow down, turn right) are introduced, but the issue was that how we can implement these traffic rules if a vector is given as an input? For this purpose a traffic rule method is implemented in this work that deforms the vector based on these two traffic rules. It is discussed in detail in section 3.2.3.
- Implementation of a simple solution for the local minima problem inherent in potential field methods.
- Extensive experimental evaluation of multi-robot navigation method in a small shared area.

1.4 Thesis outline

This thesis is organized as follows.

-Chapter 2 provides the background to this work, discussing the basic concepts of motion planning, potential fields, fuzzy logic systems and MB approach.

-Chapter 3 gives the details about the implementation of the proposed approach. Limitations and problems are discussed.

-Chapter 4 presents and discusses the experimental results.

-Chapter 5 contains the end conclusion from the results and some ideas for future improvements.

Chapter 2
Background

This chapter provides the foundation to this work and also discusses the potential field concepts in the field of robotics, as well as fuzzy logic concepts and market based optimization (MBO) technique.

2.1 Mobile Robots

Mobile robots are vehicles with the ability to change their positions. These robots can move on grounds, on the surface of water, under water and in the air. Two modes can be used to operate mobile robots [8]. One is tele-operated mode where movement instructions are given externally. Another mode is autonomous where robots operate on the information that these get from sensors and no external instructions are given.

Wheeled mobile robots are one of the types of mobile robots extensively used in research and industry, as wheel is the most popular locomotion mechanism in mobile robotics [25]. One of the advantages of wheeled robots is that balancing is not a problem as robots are designed in such a way that all wheels are on the ground.

2.2 Holonomic and Non-holonomic drive robots

Wheeled robots can be further classified into holonomic and non-holonomic drive robots. A brief introduction to understand holonomic and non-holonomic robots is given in [25].

The total DOF that a robot can have is called the degree of maneuverability 'δ_M', and can be defined as equation (2.1) [25].

$$\delta_M = \delta_m + \delta_s \tag{2.1}$$

Here δ_m is the degree of freedom that can be controlled directly through the wheels by setting different velocities, while δ_s is the degree of freedom that the robot indirectly controls by steering control and moving. Let us take an

example of three different types of robots i.e. a differential drive robot, a tricycle robot and an omni-directional robot. Both differential drive and tricycle robots have the same $\delta_M = 2$. But these robots are not similar. δ_M in case of differential drive robot comes from $\delta_m = 2$ and $\delta_s = 0$, as there is no steering involved, so translation and rotation is done through wheels as both wheels can have independent velocities. In case of tricycle robots $\delta_M = 2$ comes from $\delta_m = 1$ and $\delta_s = 1$. As in tricycle robots steering is also involved in movement.

An omni-directional robot has $\delta_M = \delta_m + \delta_s = 3 + 0 = 3$, as omni-directional robots can also move sideways. So omni-directional robots can set its 3 position variables (x, y, θ) independently.

Let little talk about workspace of wheeled mobile robots. In a workspace mobile robot can achieve any position and require DOF = 3 (2 translations along (x-axis,y-axis) and 1 rotation around z-axis) [25].

Holonomic drive robots can be described based on the relationship between controllable DOF of robot and total degrees of freedom of its workspace. *"A robot is holonomic if and only if Controllable DOF = Total DOF"* [25]. Omni-directional robot is a holonomic drive robot.

If the controllable DOF is less than the total DOF, then these mobile robots have a non-holonomic drive. Diffrential drive and tricycle robots are non-holonomic drive robots.

In this thesis, non-holonomic drive robots are considered. Differential drive robots with car like motions (discussed in 3.3) are used for testing purpose.

2.2.1 Kinematic model of the differential drive robot

Kinematics is the study of the mechanical system of robot [25]? In mobile robotics there is a need to learn the mechanical behavior of a robot to design suitable mobile robots for tasks and to study how to design the control software to run the mobile robot hardware [25].

Forward kinematics is the study of robot motion if the robot geometry is given knowing the speed of wheels. Differential drive robots have two wheels, each with diameter r. Point P is the center point between wheels axis and each wheel is distance l from P. See figure (fig. 2.1),

where

d_l = distance traveled by left wheel

d_r = distance traveled by right wheel

$d = (d_r + d_l)/2$ distance covered by robot

$\Delta = (d_r - d_l)/D$, central angle, where $D = 2l$

$R = d/\Delta$, turning radius

Robot motion

Motion control of non-holonomic vehicles is not an easy task [25]. Consider robot's goal position is (x_g, y_g) in a global frame as shown in figure (fig. 2.2).

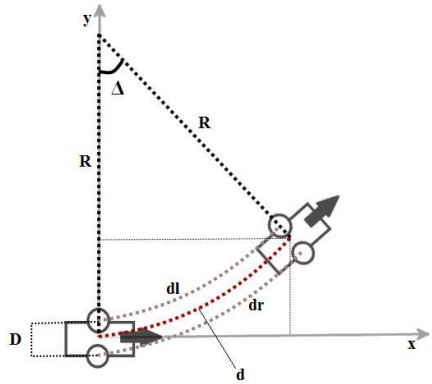


Figure 2.1: Schematic Diagram of a Differential Drive Robot

The error in position is E_{pos} and error in orientation is E_θ. Aim is to reduce E_{pos}, it can be done by controlling linear and angular velocities. A proportional strategy is used in our implementation

$$\begin{bmatrix} v_t \\ w \end{bmatrix} = \begin{bmatrix} k_p & 0 \\ 0 & k_\theta \end{bmatrix} \times \begin{bmatrix} E_{pos} \\ E_\theta \end{bmatrix} \tag{2.2}$$

Where

$E_{pos} = \sqrt{{D_x}^2 + {D_y}^2}$ and $D_x = x_g - x_r$, $D_y = y_g - y_r$

$E_\theta = atan2(D_y, D_x) - \theta_r$

Here k_p and k_θ are proportional terms.

Now to move robot there is a need to convert (v_t, w) to (v_r, v_l). It can be done using equation (2.3) [8].

$$\begin{aligned} v_r &= v_t + (D/2)w \\ v_l &= v_t - (D/2)w \end{aligned} \tag{2.3}$$

where
v_t = linear velocity
w = angular velocity
v_r = linear velocity of right wheel
v_l = linear velocity of left wheel

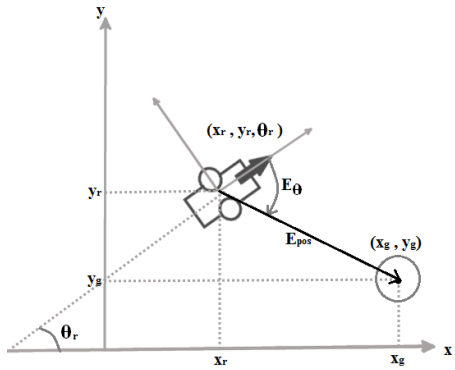


Figure 2.2: Position error in achiving goal

2.3 Multi mobile robot navigation

When in a team of mobile robots a member is given a task in a large multi-robot environment it has to move from its current position to its target position without colliding with other robots. In such multi robot applications there is a need of cooperation between the members of the team while moving.

In [9], two multi robot coordination approaches are discussed. The first option is the centralized approach, where planning for all robots is done by a stationary system or by one robot of the team. This master robot/system has the complete information of all team members and plans the task for each team member. This approach requires an intense communication between team members but provides optimal solutions. On the other hand perfect results cannot be generated for large teams.

The second option is a decentralized approach where all the team members rely on the local information and performs actions independently. So each team member acts according to its surroundings. This approach is fast and every robot can find a good local solution but it does not provide an optimal global solution [9].

To deal with multiple robots in a small area an approach named prioritized planning can be used [30]. In this approach robot priorities are selected where the robot with highest priority plans first. Robots with less priority plans afterwards and avoid the paths planned by earlier robots. A drawback of this approach is that planning cannot be performed simultaneously and can take much computing time if the number of robots is increased. Another approach [23] can be used where robots with lower priorities give way to other robots. Priorities are set by a central computer but the decision to give way is made in

the local frames of robots. A weakness of this approach is that robots should know the priorities of other robots.

A '*cocktail party model*' is proposed in [18] which is similar to a navigation technique applied by guests in a cocktail party. Each robot moves to its target position viewing others as obstacles and without communicating. This approach is simple and communication free. The disadvantage is that this approach cannot handle the situation where two robots are coming in opposite direction in a narrow corridor.

Another concept of using traffic rules is discussed in [14] where predefined rules are used for mobile robot motion coordination. There is an assumption in using traffic rules that every robot knows and follows the same rules.

2.4 Potential fields

In mobile robotics, reaching a goal position without colliding with obstacles is a difficult task. The potential field algorithm is very simple and easy to implement for avoiding obstacles. The concept of potential fields is that some virtual potential fields act on a robot [15]. A potential field is a reactive architecture as it reacts to the environment [20]. It can be implemented offline, when complete information of a map or an environment is known or can be generated using real-time sensor data. In our implementation we are generating potential fields using the real-time sensor data.

The idea for obstacle avoidance by potential field methods was first introduced in 1985 by Khatib [15]. Khatib introduced potential fields having two parts, attractive and repulsive potentials. An attractive potential is produced by a goal and pulls the robot to the goal position. A repulsive potential is produced in the region of obstacles and keeps the robot at safe distance from obstacles. So by combining these two parts a virtual force is induced that moves the robot towards goal and keeps it away from obstacles.

A drawback of potential field algorithms is that a robot may be trapped in a position due to local minima. Different methods were introduced to avoid local minima problems in case of potential-field-based navigation. Wall following [32] and adding virtual obstacles [7] are methods that work fine to escape from local minima situations. Another approach was introduced [17], where an additional force is added when a robot stops in local minima. Once a robot has escaped from local minima, the traditional potential fields can be reused. In our implementation, we have used this approach with little changes to avoid the local minima problem. (It is discussed in 3.2.6 in chapter 3).

2.4.1 Visualizing potential fields

A potential field is a collection of vectors. In mathematics a vector has a magnitude and a direction. A vector is represented geometrically as an arrow in 2-space or 3-space, where the length of an arrow is the magnitude and the

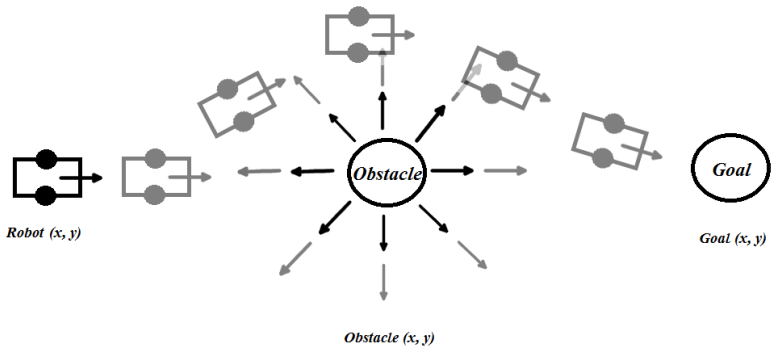


Figure 2.3: Robot moving towards goal and avoiding obstacle using PF

angle of arrow is the direction. Vectors can be added or subtracted. A brief introduction to vectors is given in [28].

Two dimensional maps are used in most of the mobile robot applications [20]. To understand potential fields consider the (x, y) positions of a robot. Obstacle and goal are known in this map and the robot is moving towards the goal. In figure (fig. 2.3) it shows how that obstacle is applying force on robot. If a robot is in the range of an obstacle, it "feels" some force that makes it to move away from the obstacle. If the robot is out of the obstacle range it moves towards the goal position.

2.4.2 Generating potential fields

A basic understanding is given in this section that how potential fields are generated. An idea of generating attractive and repulsive potential fields is discussed in detail in [2]. In an attractive field a vector is computed from the robot point towards goal position. In repulsive fields, a vector is computed from obstacle position towards robot position. By addition of both vectors a resultant vector is computed. The robot moves in the direction of this resultant vector with a velocity proportional to the length of this vector.

To find the attractive potential field, the (x, y) positions of robot and goal should be known. Let (x_r, y_r) be the position of the robot and (x_g, y_g) the position of the goal in a 2D map.

The attractive potential field vector is given by

$$\begin{aligned} &\vec{v}_{atrr} = \begin{bmatrix} x_{attr} \\ y_{attr} \end{bmatrix} \\ &where \\ &x_{attr} = (x_g - x_r) \\ &y_{attr} = (y_g - y_r) \end{aligned} \tag{2.4}$$

Similarly to find the repulsive potential field, let (x_o, y_o) be the position of the obstacle and (x_r, y_r) the position of the robot in a 2D map. The distance and angle between the obstacle and the robot are

$$d_o = \sqrt{(x_r - x_o)^2 + (y_r - y_o)^2} \tag{2.5a}$$

$$\theta_o = \tan^{-1}((y_r - y_o)/(x_r - x_o)) \tag{2.5b}$$

So the repulsive potential field vector is given by

$$\begin{aligned} &\vec{v}_{rep} = \begin{bmatrix} x_{rep} \\ y_{rep} \end{bmatrix} \\ &where \\ &x_{rep} = \begin{cases} \frac{1}{d_o}\cos(\theta_o), & \text{if } d_o \leqslant s \\ 0, & \text{otherwise} \end{cases} \\ &y_{rep} = \begin{cases} \frac{1}{d_o}\sin(\theta_o), & \text{if } d_o \leqslant s \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{2.5c}$$

Here, s is the radius of the effective region of the obstacle centered at (x_o, y_o).

In robot navigation problems there are goals and obstacles. So there is a need for combining the fields generated by obstacles and goals. By simply addition of these fields a final potential field can be calculated.

$$\vec{v}_{final} = \vec{v}_{atrr} + \vec{v}_{rep} \tag{2.6}$$

In this work attractive and repulsive potential fields are generated as discussed in [21].

2.4.3 Types of potential fields

There are some basic types of potential fields which can be combined to construct more complex fields. These fields are well reviewed in [20].

The first field is the attractive field. In figure (fig. 2.4(a)) the center point is applying an attraction force on the robot. So in an attractive field the robot tries to move towards that point. This field is very useful to move a robot to a goal position.

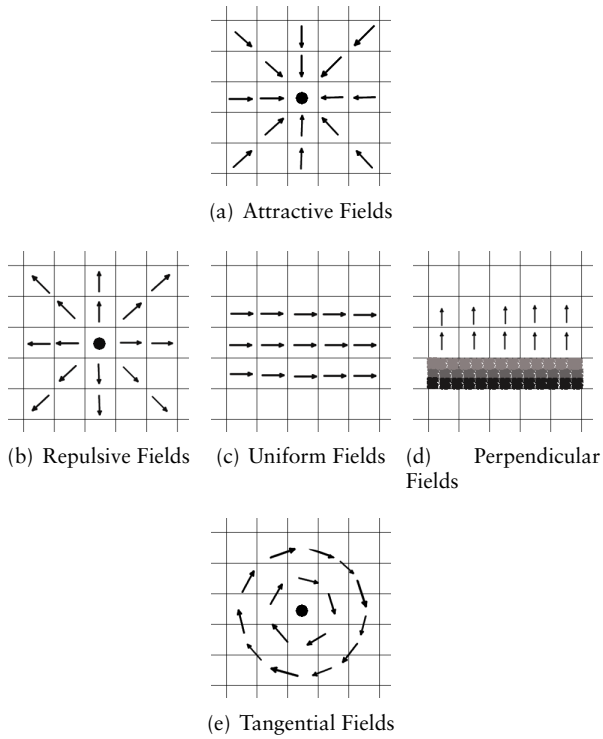


Figure 2.4: Types of Potential Fields

The second field is a repulsive field shown in figure (fig. 2.4(b)). This field is opposite to the attractive field and it is related with obstacles. This field is helpful to keep the robot away from obstacles.

The third field is the uniform potential field shown in figure (fig. 2.4(c)). In this field, the robot senses always the same force, and tries to move in the direction of the force, with constant velocity. This field is helpful when it is needed to move the robot in a specific direction.

The fourth field is the perpendicular field as shown in figure (fig. 2.4(d)). The robot senses this field when an obstacle or a wall is present and perpendicular to the robot. This field is useful for avoiding walls or boundaries.

The fifth type of potential field is the tangential field shown in figure (fig. 2.4(e)). This field is determined by finding the magnitude and direction in a similar way as in the repulsive field. But to have the direction of the vector tangent to the obstacle, theta (θ) is set to ($\theta = \theta + 90$). This field is useful for robot to move around an obstacle.

In our implementation we are dealing only with attractive and repulsive potential fields.

2.4.4 Limitations of the potential field approach

The potential field algorithms are popular due to their simplicity and easy implementation. For simple scenarios, it gives satisfactory results. But there are some drawbacks in implementing it for real time applications. In 1991, Y.Koren and J.Borenstein discussed the limitations of potential field algorithm for mobile robot navigation [16].

The local minima problem is the most common problem, where after adding multiple potential fields a new vector with zero magnitude is calculated [20]. As a result the robot remains motionless. Another drawback is that, the robot cannot pass among two closely spaced obstacles [16]. While moving in narrow passages the robot is effected by repulsive forces, which causes unstable movement of the robot [16].

2.5 Fuzzy control systems

The fuzzy logic approach is one of the famous methods that can be used for obstacle avoidance. A fuzzy controller is used for obstacle avoidance in [19], where a reactive control system is developed using fuzzy logic. To avoid a collision a combination of driving and steering commands are generated using a fuzzy controller depending on the environmental changes. In [27], potential fields are used for navigation where two layers of fuzzy logic control are used for obstacle avoidance. The first layer is used for merging sensor information to provide directions of obstacles i.e. front, back, left and right. The second layer uses three inputs, the 1st is the output of the first layer of the fuzzy control, the 2nd is the output of the potential fields, the 3rd is the robot speed and generates the control output i.e. the linear velocity and steering angle.

While using the potential field method, repulsive fields generated by obstacles can affect the movement of a robot, two different cases are discussed in [21]. One case is that when a goal is in the repulsive range of an obstacle, which keeps the robot away from the goal. The second case is when obstacles that are not in the way to the goal are influencing the motion of the robot. In these situations obstacles can cause unwanted repulsive fields and there is a need to deform these repulsive fields. Following [21] in this thesis work, we are deforming obstacle fields using fuzzy rules. An introduction to fuzzy logic system is given in detail in this section.

2.5.1 Fuzzy sets

Our world is full of uncertainties. In real world there are some statements we cannot make with certainty [4]. For example a statement about weather *'The weather is not so hot'* has some uncertainty. This statement cannot be strictly True/False, e.g. one cannot say that it is hot *True* or not hot *False*. In fuzzy logic this statement can be *True/False* to a certain level. Zadeh in 1965 laid the

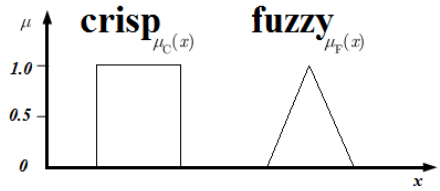


Figure 2.5: Membership functions: $\mu_C(x)$ is a membership function of *crisp set* and $\mu_F(x)$ is a membership function of *fuzzy set*

foundation of fuzzy set theory [33]. Zadeh stated that in this world there are many sets that have non-separate borders. For example *'the set of old people'*, here a person with 80 years is the member of the set and a 10 year old boy will not be the member of this set. But what about people at the age of 30, 50 and 60? Zadeh introduced the "membership function"; this function allocates a degree of membership to each member of set. Those sets in which the degree of membership can be assigned to elements are called "fuzzy sets" [13]. In case of crisp sets a membership function returns either 0 (*not a member*) or 1 (*is a member*) and is denoted by μ. But in case of fuzzy sets the membership function returns a value that is in the interval $[0, 1]$. It can be easily seen in figure (fig. 2.5). Here $\mu_C(x)$ is a membership function of a *crisp set* and $\mu_F(x)$ is a membership function of a *fuzzy set*.

There are three basic fuzzy logic operations AND, OR and NOT [31]. There are several definitions of these operations, one widely used definition is

$\mu_A(x) \cap \mu_B(x) = \min(\mu_A(x), \mu_B(x))$
$\mu_A(x) \cup \mu_B(x) = \max(\mu_A(x), \mu_B(x))$
$\sim (\mu_A(x)) = 1 - (\mu_A(x))$

2.5.2 Fuzzification

Transformation of crisp values to fuzzy values is called fuzzification [13]. Different membership functions can be used for fuzzification, triangular or trapezoidal membership functions are easy to represent and mostly used [26].

To understand fuzzification, let us take an example of a robot's distance from its traget d_t (how far the target is). The maximum distance d_t is 12 meters in our case. For the fuzzification distance $d_t = 9$ meters, the two membership functions μ_M and μ_B used which characterize a *medium* or *big* distance fuzzy sets. It can be easily understood from figure (fig. 2.6) that $\mu_M = 0.75$ and $\mu_B = 0.25$.

After fuzzification it becomes very simple to express these values in linguistic terms. These linguistic terms help us to apply rules in an easy manner [26]. The common sense of a human is applied on a computer controlled system using

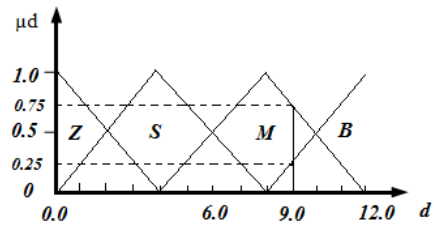


Figure 2.6: Example: Fuzzy membership functions

these rules which is a great attractiveness of fuzzy logic [26]. Fuzzy rules are of the form IF-THEN rules. Mamdani fuzzy rules and Takagi-Sugeno fuzzy rules are two types of fuzzy rules used in fuzzy systems [31]. A simple mamdani fuzzy Rule is

IF Target Distance is *Big* **AND** Obstacle distance is *Small* **THEN** Speed is *Small*. (Rule)

Here '*Target Distance*' and '*Obstacle distance*' are input variables and '*Speed*' is the output variable. *Big* and *Small* are fuzzy sets. The first two variables are called input fuzzy sets and last one is the output fuzzy set. The **IF** part is called "rule antecedent" and **THEN** part is called "rule consequent" [31]. The combination of a set of rules is called a *rule base* [22]. In our implementation mamdani fuzzy rules are used.

2.5.3 Inference engine

The mechanism that produces an output from a rule-base is called inference mechanism also known as "inference engine" [22]. Different inference mechanisms can be used based on the types of fuzzy rules [31]. Applying the **AND** fuzzy operator in (Rule), we get a membership value which is the outcome of the rule antecedent. Now the question is how to compute **THEN** in (Rule)? Calculating **THEN** is called "fuzzy inference". The Mamdani method is mostly used due to its simple structure of '*min-max*' operations. To take an output against each rule fuzzy implication operator can be used. In this thesis, the minimum operator is used to obtain an output against each rule.

All the outputs obtained against each rule are combined into a single fuzzy set using a fuzzy aggregation operator [1]. In our work maximum aggregation operator is used to combine outputs.

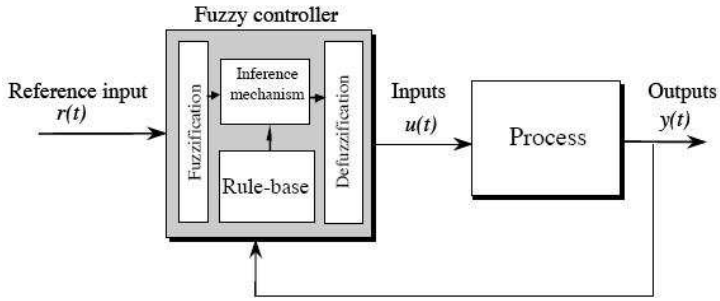


Figure 2.7: Fuzzy control system [22]

2.5.4 Defuzzification

After getting an output from the inference mechanism, this output has to be converted into a crisp value as an output of the fuzzy controller [22], and this mechanism is called defuzzification. There are different methods for defuzzification [26]. In our implementation, we have used center of gravity (*CoG*) method for defuzzification, which is computed using center of area and area of each implied fuzzy set. The formula for CoG reads,

$$\mathrm{CoG} = \frac{\sum_{x=a}^{b} x\mu(x)}{\sum_{x=a}^{b} \mu(x)} \tag{2.7}$$

where x is the sample value and $\mu(x)$ is the membership value at x.

2.5.5 Fuzzy control system

A block diagram of a fuzzy control system is shown in figure (fig. 2.7). How a fuzzy controller works is described in detail in [26]. The fuzzy controller has four blocks; the input crisp information is converted into fuzzy values by fuzzification block. An inference mechanism determines the output against each fuzzy rule in a rule base. Using a defuzzification block, these outputs are combined and converted into a crisp value.

2.6 Market based (MB) approach

Dealing with multiple mobile robots, there is a need for coordination between team members. In section 2.3, different methods for coordination are discussed. For multi-robot coordination tasks, the Market based approach got

popularity by researchers in the last few years. In robot teams, there are some resources and tasks whereas there is a need to distribute these resources efficiently among team members to accomplish tasks [9]. *"Market based approach is a decentralized resource allocation method"* [29]. MB approach is very similar to economical systems where a production firm produces goods in limited resources and distributes these goods and wants to increase its profit [9]. In this thesis work we are using the potential field method for navigation of multi mobile robots, and according to [21] the MB approach for coordination between team members. The MB approach helps us in giving preferences to other robots and the potential field can be strengthened or weakened based on these preferences.

A basic concept to understand economic system is given in this paragraph. In [12] a detailed introduction is given to understand the economic system. According to [12] *'Market'* is a place where commodities are traded between producers and consumers. Commodities are the goods or services offered by producer and demanded by consumers. *"The price acts as a regulator of the quantity of commodities that are offered by producers and that are demanded by consumers"*. [12]. In an economic system, goods are generated by producers in given resources (labor, technology etc.) and are distributed among consumers. Consumers have some demands in purchasing goods where their main goal is to maximize their satisfaction, while the goal of production firm is to maximize the profit. An economic system is called in equilibrium state when an equilibrium price is selected in such a way that producers are ready to sell goods with the quantities that consumers are ready to buy and as a result there is no surpluses or shortages [12].

Economic system principles can be implemented for multi-robot coordination [9]. So for the implementation of such technique one has to have a model of an idealized economic system [29]. In [29], an idealized model for economic system is discussed in detail. This system consists of a price system, commodities, producers and consumers. Every commodity has a limited quantity and a price is set for each quantity in this system.

Producers take production factors as input and generate commodities as output. Production factors are the inputs that are needed in the production of commodities; these factors could be labor, land, technology etc. Producers have to buy these factors on certain price. The main goal of the producer is to maximize its profit. On the other hand, consumers choose their demands in buying commodities. The demand of a consumer depends on two questions: one is how much money he has? and second is what preferences he has? Every consumer has some limited wealth and its preference is to maximize its satisfaction using this wealth. Answering the two questions above, the goal of the consumer is in choosing the demands. E.g. if two demands are given, the consumer has to choose which one he prefers. In this idealized model each consumer takes commodities as inputs and performs some task as an output. The goal of the consumer is to choose the best preference.

In [29] the economic model is simplified by considering that consumers and producers are dealing with only one commodity, where u_{iD} (quantity of commodity demanded) and u_{jS} (quantity of commodity supplied) are used. *"The set u_{iD}, u_{jS} of all demands and supplies is called the state of economy"* [29]. The state of economy is called in equilibrium if (2.8).

$$\sum_{i=1}^{n} u_{iS} = \sum_{j=1}^{n} u_{jD} \tag{2.8}$$

The distribution of resources can be made in an optimal way when the state of the economy is in equilibrium [29]. In [21] weights are used to strengthen or weaken the repulsive fields generated by other robots and these weights are equivalent to '*quantity of commodity*' in [29]. The market based approach discussed in this thesis work is explained in [21] in detail. An overview of using market based multi-robot coordination in different areas is given in [9].

Chapter 3
Potential field and market based optimization for navigation of mobile robots

3.1 Introduction

Navigation of mobile robots with potential field method is popular due to its simplicity and easy implementation. The performance of the potential field method in the case of multi-robot navigation can be improved by using some optimization methods. Multi-robot navigation in a shared area can sometimes accidently leads to situations when there is a requirement to avoid other robots from a certain distance and to generate a smooth movement of robots. To handle such situations, the MBO is proposed in [21] where some potential fields are strengthened and some are weakened based on the local situation. Every robot decides its control action in the local frame by using its own sensors. In this chapter, the potential field method with fuzzy rules and MBO [21] is discussed together with what issues we can face while implementing this method.

We have divided our work into two parts that can be easily understood from figure (fig. 3.1). In the first part, we have generated potential fields and these fields are optimized with fuzzy rules and the MBO technique proposed in [21]. As a result we find a velocity vector. The second part is providing this velocity vector to the robot. The movement depends on the properties of the robot (omni-directional robot, differential-drive robot or car-like robot). We have used differential-drive robot in our implementation. The idea is that each mobile robot will be using same algorithm for navigation. All calculations discussed in this chapter are formulated in the local frame of the robot.

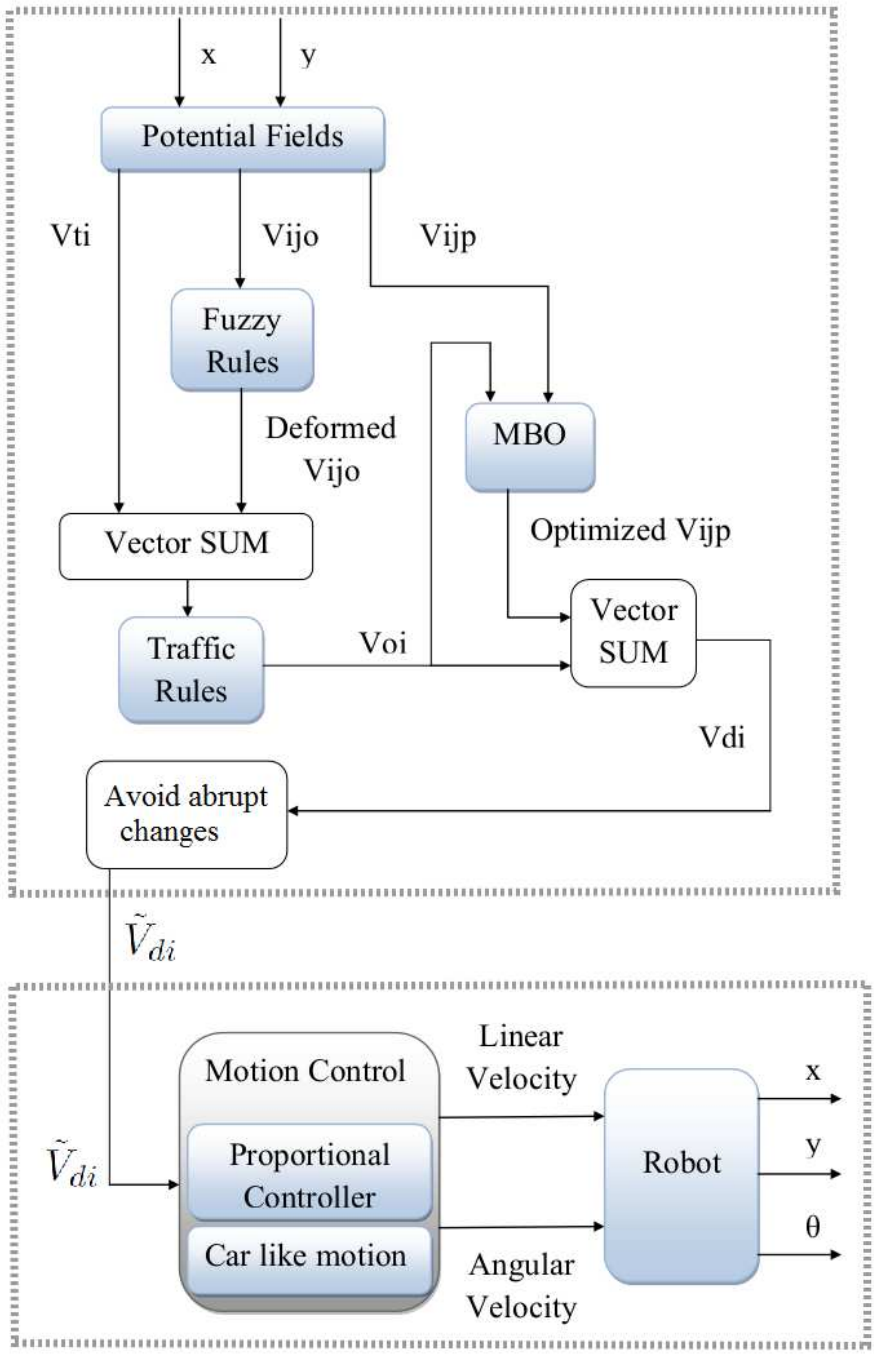


Figure 3.1: Flow diagram for the entire navigation method, implemented in this thesis work. (x,y) = Robot position, v_{ti} = tracking velocity vector of robot, v_{ijo} = repulsive velocity vector between obstacle j and robot i, v_{ijp} = repulsive velocity vector between robots i and j

3.2 Navigation algorithm implementation

3.2.1 Potential field method

Attractive and repulsive potential fields are computed as described in [21] to safely reach a goal position. Let us consider a robot platform P_i, its tracking velocity vector is computed as

$$\nu_{ti} = k_{ti}(x_i - x_{ti}) \tag{3.1}$$

$x_i \in \mathbb{R}^2$: position of platform P_i,
$x_{ti} \in \mathbb{R}^2$: position of target T_i,
$\nu_{ti} \in \mathbb{R}^2$: tracking velocity vector,
$k_{ti} \in \mathbb{R}^{2x2}$: gain metrix (diagonal)

This tracking velocity vector is very similar to attractive potential field vector computed in (2.4.2). But here a gain k_{ti} diagonal matrix is used, where values on the diagonal are negative.

The repulsive potential field vector is computed between robot platform P_i and obstacle O_j, using

$$\nu_{ij_o} = -c_{ij_o}(x_i - x_{jo})d_{ij_o}^{-2} \tag{3.2}$$

$x_{j_o} \in \mathbb{R}^2$: position of obstacle O_j,
$\nu_{ij_o} \in \mathbb{R}^2$: repulsive velocity vector between platform P_i and obstacle O_j,
$d_{ij_o} \in \mathbb{R}$: Euclidian distance between platform P_i and obstacle O_j,
$c_{ij_o} \in \mathbb{R}^{2x2}$: gain metrix (diagonal), having negative values along its diagonal

Similarly repulsive PF between P_i and other platform P_j is computed using (3.3)

$$\nu_{ij_p} = -c_{ij_p}(x_i - x_j)d_{ij_p}^{-2} \tag{3.3}$$

$x_j \in \mathbb{R}^2$: position of other platform P_j,
$\nu_{ij_p} \in \mathbb{R}^2$: repulsive velocity vector between platforms P_i and P_j,
$d_{ij_p} \in \mathbb{R}$: Euclidian distance between platforms P_i and P_j,
$c_{ij_p} \in \mathbb{R}^{2x2}$: gain metrix (diagonal), having negative values along its diagonal

As all calculations are formulated in the local frame of platform (i) so in equations (3.1), (3.2) and (3.3) $x_i = (0, 0)$ and target, obstacle and platform (j) positions will be in local frame of platform (i) as well. Platform P_i can detect obstacles and other robots from its sensor. Here, it is considered that platform P_i is continuously getting the updated positions of other platforms P_j and can differentiate between obstacles and other platforms. A laser range finder is used to sense obstacles and other platforms and the maximum detecting range is

Table 3.1: Repulsive velocity vector on diffrent obstacle distances.

Platform position	Obstacle Position	distance from obstacle	v_{ij_o}
$(0,0)$	$(9,0)$	$9m$	$[\ -0.1111; 0\ ]$
$(0,0)$	$(7,0)$	$7m$	$[\ -0.1429; 0\ ]$
$(0,0)$	$(5,0)$	$5m$	$[\ -0.2000; 0\ ]$
$(0,0)$	$(3,0)$	$3m$	$[\ -0.3333; 0\ ]$
$(0,0)$	$(1,0)$	$1m$	$[\ -1.0000; 0\ ]$
$(0,0)$	$(0.7,0)$	$0.7m$	$[\ -1.4286; 0\ ]$
$(0,0)$	$(0.5,0)$	$0.5m$	$[\ -2.0000; 0\ ]$
$(0,0)$	$(0.3,0)$	$0.3m$	$[\ -3.3333; 0\ ]$
$(0,0)$	$(0.1,0)$	$0.1m$	$[\ -10.000; 0\ ]$

selected. Table (3.2) shows how field is changing depending on the distance of obstacle from platform, C_{ij} is selected as $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$.

In our implementation a cylindrical shaped obstacle is used with radius $r = 0.15$ meters and height $h = 0.62$ meters. The laser range finder has a range of 9 meters and angle range is from -135to135 degrees and the angle step size is 0.3953 degree, shown in 3.2(a). Laser range finder detects the outer points of the obstacle and in our implementation, the center point of the obstacle is not computed. Therefore the repulsive velocity vector is computed against each detected point of the obstacle shown in 3.2(b). Based on these settings of the laser range finder and the obstacle type, the sensor is detecting different obstacle points at different distances, shown in (fig. 3.2(c)). In (fig. 3.3) it is shown how a cylindrical obstacle will apply a repulsive force on the robot at different distances.

The final velocity vector V_{di} is the sum of

$$v_{di} = v_{ti} + \sum_{i=1}^{m_o} v_{ij_o} + \sum_{i=1}^{m_p} v_{ij_p} \tag{3.4}$$

where m_o and m_p are the number of obstacles and platforms, respectively. In (3.4), the repulsive velocities will not have an effect if the length of the vector v_{ti} is very big. So there is a need to limit the length of v_{ti}. (Algorithm- 1) is used to limit v_{ti}.

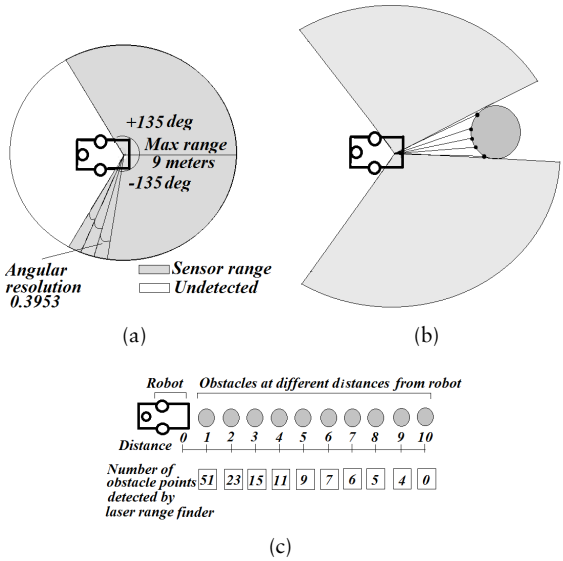


Figure 3.2: Laser range sensor: (a) sensing area (b) obstacle points detected by laser range finder (c) number of obstacle points detected at different distances

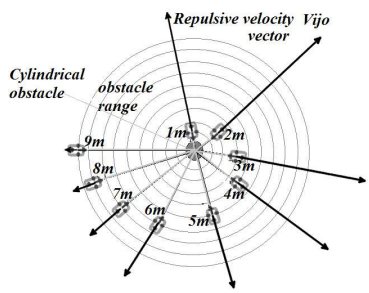


Figure 3.3: Effect of obstacle distance from the platform on v_{ij_o} (repulsive velocity vector)

Algorithm 1 Limiting tracking velocity vector v_{ti}

$limit = \text{input}$

$v_{ti} = \begin{bmatrix} v_{ti_x} \\ v_{ti_y} \end{bmatrix}$

$where$

$v_{ti_x} = \text{x-component of } v_{ti}$

$v_{ti_y} = \text{y-component of } v_{ti}$

Length (magnitude) $| v_{ti} |$ and direction $\theta_{v_{ti}}$ of vector v_{ti} is computed

$| v_{ti} | = \sqrt{{v_{ti_x}}^2 + {v_{ti_y}}^2}$

$\theta_{v_{ti}} = atan2(v_{ti_y}, v_{ti_x})$

Condition

if $| v_{ti} | > limit$ **then**

$| v_{ti} | = limit$

end if

$v_{ti_x} = | v_{ti} | \cos(\theta_{v_{ti}})$

$v_{ti_y} = | v_{ti} | \sin(\theta_{v_{ti}})$

3.2.2 Fuzzy rules to optimize obstacle potential fields

The obstacle potential field is independent of an attractive potential field and can cause unwanted repelling forces when the target position is in the effective region of the obstacle. A method is introduced in [21] to deform potential fields, which makes a decision to strengthen or weaken the repulsive potential field. This decision is made based on two considerations (see figure fig. 3.4) [21]. First consideration is to strengthen the repulsive potential field if obstacle is hiding the target and to weaken it if target "can be seen" from the platform. This can be done by calculating an angle α_{ij}, which is the angle between V_{ij_o} and Vti. The second consideration is the length of the tracking velocity vector, so repulsive potential field should be strong if this length is big (target is far) and weak if the length is small (target is near). It can be done by simply calculating the length of V_{ti}.

A coefficient $coef_{ij}[0,1]$ is introduced to fullfill the above requirements which is multiplied with v_{ij_o} to obtain a new v_{ij_o}.

$$v_{ij_o} = -coef_{ij} c_{ij_o} (x_i - x_{jo}) d_{ij_o}^{-2} \tag{3.5}$$

This cooeficient is calculated from a set of 16 fuzzy rules

$$\text{IF } v_{ti} = B \text{ AND } \alpha_{ij} = M \text{ THEN } coef_{ij} = M \tag{3.6}$$

These rules are summarized in table (3.2), where Z-zero, S-small, M-medium and B-big are fuzzy sets [21]. The fuzzy controller is implemented in the same way as discussed in section (2.5). The fuzzy membership functions (μ_α, μ_{vt}, μ_{coeff}) used here are shown in figure (fig. 3.5).

Table 3.2: Table of fuzzy control rules to calculate the fuzzy cooeficient

v_{ti}	α_{ij}			
	Z	S	M	B
B	Z	S	M	B
M	Z	S	M	M
S	Z	M	M	S
Z	Z	Z	Z	Z

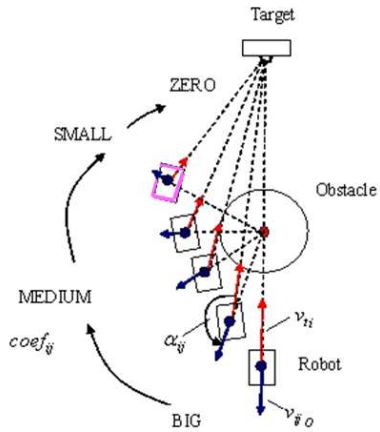


Figure 3.4: Deformation of obstacle potential fields based on v_{ti} (tracking velocity vector) and angle α_{ij} (angle between v_{ij_o} and v_{ti})

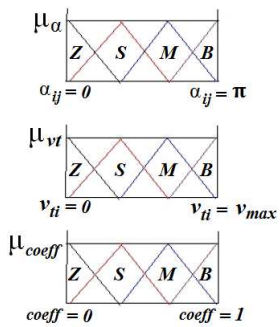


Figure 3.5: Fuzzy membership functions: μ_α (membership function to fuzzify input variable α_{ij}), μ_{vt} (membership function to fuzzify input variable v_{ti}), μ_{coeff} (membership function to compute output variable 'coeff')

3.2.3 Traffic rules

Common rules for movements are required in a working area of mobile robots. In [14] a method was proposed in which traffic rules were used to run mobile robots in an environment. Elements and classification are discussed in [14] to construct traffic rules. Elements of traffic rules are the information about the environment and the mobile robots. Information about the environment is the type of obstacles present in the working area e.g. static or dynamic. The information about the mobile robots is the sensor range, running speed and minimum turning radius etc. In classification of traffic rules three points are discussed. First, applying traffic rules on the current situation of the robot, e.g. the robot is moving at high speed, turning right or left. Second is if the robot detects other mobile robots, what actions should be choosen in this situation? Avoiding or overtaking are possible actions. Third is the guarantee of a safe movement in the case when there is chance of accident.

For smooth and safe movements of mobile robots, traffic rules are applied in [21]. In our environment we have only static obstacles and mobile robots. Static obstacles are of cylindrical shape and of the same size. Furthermore mobile robots are of the same type and all off them are using the same algorithm for navigation. So the concept is that all these mobile robots will use common traffic rules. Rules are as follows.

1. Decrease the speed if vehicle comes from right side.
2. Move to the right if angle β of two approaching platforms is $\beta < \beta_0$. (β_0 is a small angle)

Now the problem is that how we can implement these traffic rules in the case, when the potential field vector is given as an input? In [21] tracking velocity, obstacle PF and traffic rules are combined. Here the tracking velocity and the obstacle PF are vector quantities that can be combined (using vector addition) equation (3.7a). But the problem is that how to combine traffic rules with vector quantities. It can be done by introducing 'traffic rules' function; this function will take a vector quantity as an input and give a vector quantity as an output (equation (3.7b)).

$$v_{oi} = v_{ti} + \sum_{i=1}^{m_o} v_{ij_o} \tag{3.7a}$$

$$v_{oi} = \mathtt{trafficRule}(v_{oi}) \tag{3.7b}$$

As we have a vector input and we want to implement traffic rules on this vector. This implementation is done as follows.

Slow speed

"Decrease speed if vehicle comes from the right side".

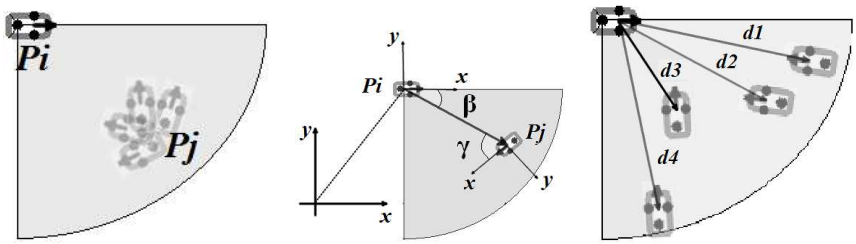


Figure 3.6: Trafic rule slow speed: (a) vehicle detection on right side (b) Computing angles β and γ (c) Nearest vehicle on right side

Let us assume that we have two vehicles (P_i and P_j) and we want to implement the above rule on vehicle P_i. The implementation of this rule can be divided into two parts. One part is 'vehicle comes from right side', it raises the question how we can determine that P_j is on the right side and coming towards P_i? The second part is 'Decrease speed', it raises another question how it's possible to decrease the speed if we can only control the value of velocity vector?

To find that 'vehicle comes from right side' three conditions have to be fulfilled. First condition is that angle between P_i and P_j is $-90 < \beta < 0$, it means P_j is on the right side of P_i. Second is the $range < 3m$, it means P_j is very near. Now to determine that P_j is coming towards P_i an angle 'γ' is introduced that will help to find the direction of 'P_j'. As it's not possible to find such a fixed value that can tell about 'P_j coming towards P_i', range of angle 'γ' is set. So the third condition is that angle between P_j and P_i is $-90 < \gamma < 90$, it means P_j is coming towards P_i. So combining these three conditions means that P_j is coming towards P_i from the right side at a $distance < 3$ meters. It can be easily understood from figure (fig. 3.6(a)) and (fig. 3.6(b)).

In our implementation 'speed' of robot depends on the length of velocity vector, which means that the robot will move at a slow speed if the vector length is small discussed in [section 2.2.1 Chapter 2]. Considering equation (3.8), the robot speed can be decreased if the length of input vector is shortened to some value.

$$v_{oi} = \text{setSpeed}(v_{oi}) \tag{3.8}$$

The distance from the nearest platform P_j coming from right side is calculated, and taken as the new length of vector V_{oi}. By using angles β and γ as discussed above, it is determined that platform P_j is coming from the right side. It can be seen from figure (fig. 3.6(c)), that the nearest platform P_j approaching from right side is at distance d3.

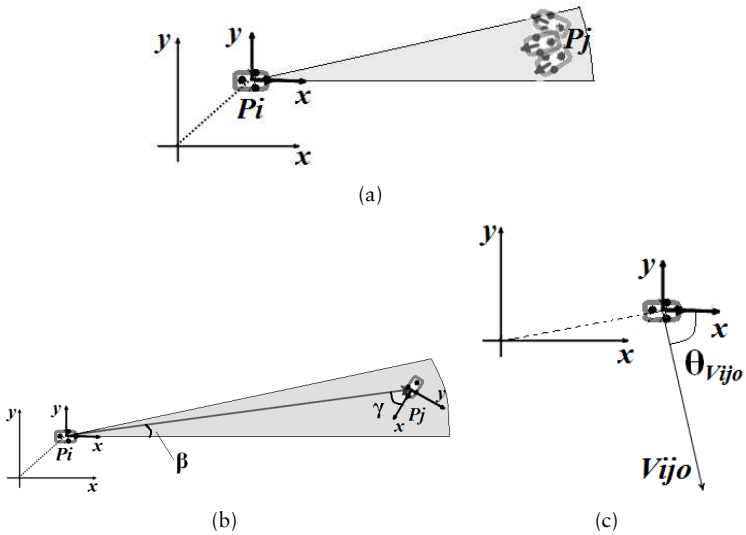


Figure 3.7: Trafic rule Turn right: (a) vehicle on left side (possible orientations of P_j telling that P_j comming towards P_i) (b) Computing angle β and γ (c) Check if vehicle is already turning right ($\theta_{v_{oi}} < -45^o$)

Turn right

"Move to the right if angle β of two approaching platforms is $\beta < \beta_0$".

Consider two vehicles (P_i and P_j), we want to implement the above rule on P_i. In this rule we have two questions. One question is what are the conditions to be fulfilled to use this rule? The second question is how it's possible for a robot to turn right if we can only control the value of the velocity vector?

$$v_{oi} = \text{TurnRight}(v_{oi}) \tag{3.9}$$

This rule is implemented when two conditions are fulfilled. The first condition is that angle $0 < \beta < \beta_0$ (figure fig. 3.7(b)). The second condition is that $-45^o < \gamma < 45^o$, it means that P_j is coming towards P_i (It can be easily understood from figure (fig. 3.7(b))). In figure (fig. 3.7(a)) P_j can have different orientations at different values of γ that can tell P_j coming towards P_i.

Robot P_i can be turned right by changing the direction of input vector v_{oi} i.e. $\theta_{v_{oi}} = -45$. The direction will not be changed if the robot is already turning right, means $\theta_{v_{oi}} < -45$ (fig. 3.7(c)).

As a result we get a new v_{oi}, which is a combination of the tracking velocity, obstacle PF and traffic rules.

3.2.4 Market based approach to optimize potential fields

In this section market based approach proposed in [21] is discussed. In [21] economic market mechanism is used for the optimization of robot potential fields. There are two requirements for this technique to implement on multi-robot system, first model the system to be optimized (discussed in section 3.2.1) and second model the optimization strategy itself (model of idealized economic system discussed in MBO section 2.6). The desired velocity vector is described as

$$v_{di} = v_{Oi} + \sum_{j=1, i\neq j}^{m} w_{ij} v_{ij_p} \tag{3.10}$$

Where
v_{Oi}: combination of
-tracking velocity v_{ti}
-v_{ij_o} repulsive velcoity between platform i and obstacle j
-Traffic rules
m: number of platforms
v_{ij_p}: repulsive velocity between platform i and j
w_{ij}: weighting factors for repulsive velocities where $\sum_{j=1,i\neq j}^{m} w_{ij} = 1$

Here the aim is to adjust weights in such a manner that contributing platforms show a smooth movement while avoiding each other. "One possible option for tuning the weights w_{ij} is to find global optimum over all contributing platforms". It becomes a difficult task when many platforms are interacting. Therefore for such a multi-agent system problem a market based approach is preferred in [21]. The economic system discussed here is same as discussed in [29], and weights are used instead of commodities.

Imagine that a local system S_i (platform) belongs to m producer agents $P_{ag_{ij}}$ and m consumer agents $C_{ag_{ij}}$. Producers sell weights and consumers buy weights on a common price p_i. Producer agents $P_{ag_{ij}}$ supply weights and the aim of $P_{ag_{ij}}$ is to maximize a local profit function ρ_{ij}. Here "local" means belonging to system S_i. While consumer agents $C_{ag_{ij}}$ demand for weights and try to maximize related local utility functions U_{ij}. The system S_i is called at the equilibrium state if

$$\sum_{j=1}^{m} w_{ij_p}(p_i) = \sum_{j=1}^{m} w_{ij_c}(p_i) \tag{3.11}$$

Where
w_{ij_p}: supplied weights by producer agent
w_{ij_c}: utilized weights by consumer agent
p_i: common price

Producer and consumer agents trade with each other depending on the definitions of a cost function for consumer and producer agents. According to [21] the definition of these functions is as follows

Local utility function for consumer agent is

$$\begin{gathered} \mathrm{Utility} = \mathrm{benefit} - \mathrm{expenditure} \\ U_{ij} = \tilde{b}_{ij} w_{ij_c} - \tilde{c}_{ij} p_i (w_{ij_c}^2) \end{gathered} \tag{3.12}$$

where $\tilde{b}_{ij}, \tilde{c}_{ij} \geqslant 0, p_i \geqslant 0$.
Local profit function for producer agent is

$$\begin{gathered} \mathrm{profit} = \mathrm{income} - \mathrm{costs} \\ \rho_{ij} = g_{ij} p_i (w_{ij_p}) - e_{ij} p_i (w_{ij_p})^2 \end{gathered} \tag{3.13}$$

where $g_{ij}, e_{ij} \geqslant 0$.
The same price p_i will be used for cost functions, because weights w_{ij} are calculated based on this common price. A local energy function between platforms P_i and P_j is defined in [21], using (3.10). Aim is to minimize this local energy function.

$$\begin{aligned} \tilde{J}_{ij} &= v_{d_i}^T v_{d_i} \\ &= a_{ij} + b_{ij} w_{ij} + c_{ij} (w_{ij})^2 \to \min \end{aligned} \tag{3.14}$$

Now the mapping is required between (3.12) and (3.14), the question is how the parameters in (3.12) can be selected? Setting these values as follows (3.15) assure $w_{ij} \geqslant 0$.

$$\tilde{b}_{ij} = |b_{ij}|, \quad \tilde{c}_{ij} = c_{ij} \tag{3.15}$$

Equations (3.12) and (3.14) are quadratic equations. So at $v_{d_i} = 0$, energy function (3.14) reaches its minimum and at $p_i = 1$ utility function (3.12) reaches its maximum.
Using values in (3.15), utility function (3.12) becomes

$$U_{ij} = |b_{ij}| w_{ij_c} - c_{ij} p_i (w_{ij_c})^2 \tag{3.16}$$

Maximization of (3.16) gives

$$\frac{\partial U_{ij}}{\partial w_{ij_c}} = |b_{ij}| - 2 c_{ij} p_i w_{ij_c} = 0 \tag{3.17}$$

Local w_{ij_c} acquired is

$$w_{ij_c} = \frac{|b_{ij}|}{2 c_{ij}} \cdot \frac{1}{p_i} \tag{3.18}$$

Similarly the maximization of the local profit function (3.13) is

$$\frac{\partial \rho_{ij}}{\partial w_{ij_p}} = g_{ij} p_i - 2 e_{ij} w_{ij_p} = 0 \tag{3.19}$$

Local w_{ij_p} acquired is

$$w_{ij_p} = \frac{p_i}{2\eta_{ij}} \quad \text{where} \quad \eta_{ij} = \frac{e_{ij}}{g_{ij}} \tag{3.20}$$

The equlibrium requirement is that the sum of produced w_{ij_p} and the sum of demanded w_{ij_c} should be equal, which gives the balance equation (3.21).

$$\sum_{j=1}^{m} w_{ij_c} = \sum_{j=1}^{m} w_{ij_p} \tag{3.21}$$

Substituting (3.18) and (3.20) into (3.21) gives the price p_i.

$$p_i = \sqrt{\frac{\sum_{j=1}^{m}(\frac{|b_{ij}|}{c_{ij}})}{\sum_{j=1}^{m}(\frac{1}{\eta_{ij}})}} \tag{3.22}$$

Substituting (3.22) into (3.18) gives the final weights w_{ij}, to be used in each local system. After finding new weights, these weights are normalized with respect to $\sum_{j=1}^{m} w_{ij} = 1$.

Market based optimization for collision avoidance between mobile platforms

In this section the MBO technique for optimization of repulsive forces between platforms is addressed. The system equation of mobile robots is as follows

$$v_{d_i} = v_{O_i} + \sum_{j=1, i\neq j}^{m} w_{ij} v_{ij_p} \tag{3.23}$$

"Global" energy function given in [21] is as follows

$$\begin{aligned} \tilde{J}_i &= v_{d_i}^T v_{d_i} \\ &= v_{O_i}^T v_{O_i} + 2v_{O_i}^T \sum_{j=1, i\neq j}^{m} w_{ij} v_{ij_p} \\ &+ \Big(\sum_{j=1, i\neq j}^{m} w_{ij} v_{ij_p}\Big)^T \Big(\sum_{j=1, i\neq j}^{m} w_{ij} v_{ij_p}\Big) \end{aligned} \tag{3.24}$$

The local energy function (3.14), reflects only the energy of two coordinating platform P_i and P_j. Given

$$
\begin{aligned}
\tilde{J}_{ij} &= v_{d_i}^T v_{d_i} \\
&= v_{O_i}^T v_{O_i} + \Big(\sum_{k=1,k\neq i,j}^{m} w_{ik} v_{ik_p}\Big)^T \Big(\sum_{k=1,k\neq i,j}^{m} w_{ik} v_{ik_p}\Big) \\
&+ 2\sum_{k=1,k\neq i,j}^{m} w_{ik} v_{O_i}^T v_{ik_p} \qquad (3.25) \\
&+ 2w_{ij}\Big(v_{O_i}^T + \sum_{k=1,k\neq i,j}^{m} w_{ik} v_{ik_p}^T\Big) v_{ij_p} \\
&+ w_{ij}^2 \big(v_{ij_p}^T v_{ij_p}\big)
\end{aligned}
$$

Now comparing (3.14) and (3.25) we obtain

$$
\begin{aligned}
b_{ij} &= 2\Big(v_{O_i}^T + \sum_{k=1,k\neq i,j}^{m} w_{ik} v_{ik_p}^T\Big) v_{ij_p} \\
c_{ij} &= v_{ij_p}^T v_{ij_p}
\end{aligned} \qquad (3.26)
$$

where a_{ij} is neglected, as a_{ij} does not contribute to the MBO process.

From (3.26), b_{ij} and c_{ij} are calculated and then price p_i is found from (3.22). By using this price in (3.18), w_{ij} is calculated. After calculating all new weights, these are normalized to satisfy

$$\sum_{j=1}^{m} w_{ij} = 1 \qquad (3.27)$$

3.2.5 Issues

Previous weights

In equation 3.26, it is considered that P_i platform is getting information of previous weights assigned to different robots. In our implementation we are assuming that every robot knows positions of other robots in its local frame, so previous weights can be saved associated with robot names. But the previous weights cannot be saved if the robot names are not known. In future work of this thesis saving information of previous weights is not possible, if the positions of other robots are unknown and if positions of static and dynamic obstacles will be provided only depending on the laser range finder.

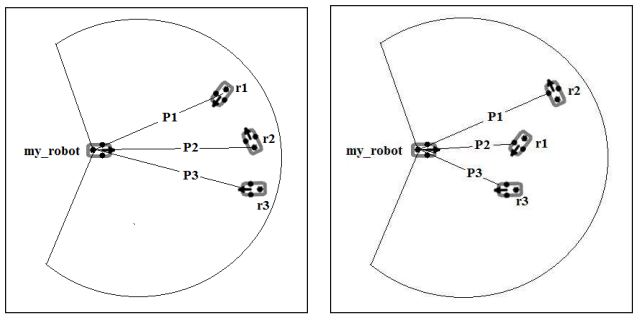


Figure 3.8: Example to understand limitation of using previous weights: (a) At first time step my_robot detects $P_1 = r_1$, $P_2 = r_2$ and $P_3 = r_3$ (b) At second time step my_robot detects $P_1 = r_2$, $P_2 = r_1$ and $P_3 = r_3$

Let us take an example as shown in figure (fig. 3.8), where four robots are moving in a shared area. In figure (fig. 3.8(a)) my_robot detects three mobile robots using its laser range finder. In first time step, r_1 is at left, r_2 is in front and r_3 is at right as shown in figure (fig. 3.8(a)). my_robot detects P_1, P_2 and P_3 without knowing the names of those robots and computes weights for each robot. Where $P_1 = r_1$, $P_2 = r_2$ and $P_3 = r_3$. In the second time step, r_2 is at left, r_1 is in front and r_3 is at right as shown in figure (fig. 3.8(b)). Again my_robot detects P_1, P_2 and P_3 without knowing the names of these robots but this time $P_1 = r_2$, $P_2 = r_1$ and $P_3 = r_3$. Here robots can change their positions and my_robot cannot differentiate between them. It is a consideration in MBO that my_robot will be using previous weights. So if in the first time step weights w_{i1_p}, w_{i2_p}, w_{i3_p} are assigned to P1, P2 and P3 respectively; these previous weights cannot be used in equation (3.26) for second time step as robots have changed their positions.

The solution provided here is to start the optimization in each time step with the same weights (in our case it is 0.5), and then to let the optimization simply run in a loop either with a defined number of loops or until a certain threshold is reached. This solution is shown in figure (fig. 3.9), and is working fine in our simulation.

Weights are normalized

After implementing this optimization, robot P_i assigns different weights to other robots depending on their importance. In equation 3.27, weights are normalized to 1. This can decrease the effect of $\sum_{j=1}^{m} w_{ij} v_{ij_p}$ in equation 3.23, if population of interacting robots is increased (It is explained in detail in section (Limitations) chapter 4).

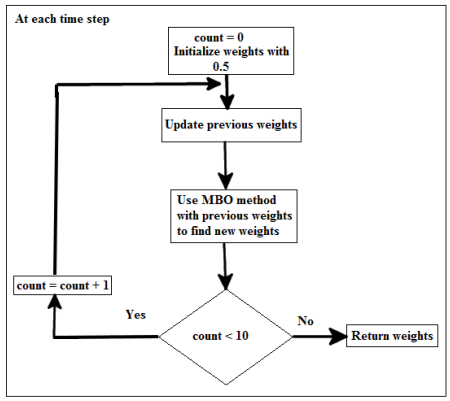


Figure 3.9: Flow chart to solve previous weights problem

3.2.6 Solution to local minima problem

One of the drawbacks of potential field algorithms is that the robot can get stuck in a local minima (section 2.4.4 in chapter 2). In the presence of obstacles, a mobile robot moves towards the target with decreasing attractive force and increasing repulsive force [17]. A local minima problem occurs when sum of overall forces is zero. Normally this problem occurs when an obstacle occurs directly between mobile robot and target position. In our implementation a local minima problem may occur when equation 3.4 gives a vector v_{d_i} with zero magnitude.

In [17] an additional force is introduced to escape from the local minimum. This additional force has the same size as that of the attractive force but an included angle θ to the attractive force is added as shown in figure (fig. 3.10) [17]. This additional force is used only when robot is in a local minimum. Once robot is out of this problem it starts again using traditional potential fields. In [17] a point robot is used for experimentation. In our implementation considering the orientation of robot, the same method with little changes is used to avoid local minima.

How we can check that robot is in local minima or not? We do it by checking the magnitude of v_{d_i} in equation (3.23). If it is very small and the robot is far from its goal position it means that the robot is stuck at a local minima. Here robot can stuck in local minima with any orientation it has (it is not necessary that robot is moving straight towards target position). This problem is solved by adding a $new\_v_{t_i}$ and removing the $old\_v_{t_i}$ in equation (3.23). $new\_v_{t_i}$ will have the same size as that of $old\_v_{t_i}$ but its direction will be different. In our implementation the selection of angle θ for $new\_v_{t_i}$ is different as that

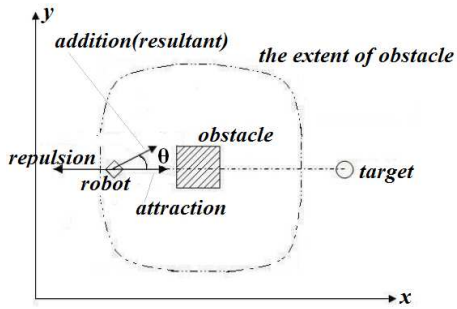


Figure 3.10: Local minima solution: Additional force is added with magnitude same as of attraction force but with different direction

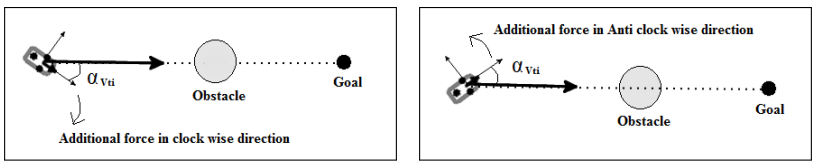


Figure 3.11: Additional force to solve the local minima problem considering the orientation of robot: (a) if $\alpha_{V_{ti}} < 0$ then new_V_{ti} will be in clock wise (b) if $\alpha_{V_{ti}} > 0$ then new_V_{ti} will be in anti-clock wise

compared to θ discussed in [17]. But the concept is the same. $new\_v_{t_i}$ will be oriented in clock-wise or anti-clock-wise direction depending on the robot orientation. In figure (fig. 3.11), this can be easily understood if $\alpha_{v_{ti}} < 0$, direction of $new\_v_{t_i}$ will be clock-wise, otherwise it will be anti-clock-wise. Where $\alpha_{v_{ti}}$ is computed as

$$\alpha_{v_{ti}} = \theta_{robot} - \theta_{old\_v_{ti}}$$

IF $\alpha_{v_{ti}} < 0$, For clock wise direction of $new\_v_{t_i}$, a $\theta_{new\_v_{ti}}$ is computed
$\theta_{new\_v_{ti}} = \theta_{robot} + (-45)$;

IF $\alpha_{v_{ti}} \geqslant 0$, For anticlock wise direction
$\theta_{new\_v_{ti}} = \theta_{robot} + (45)$;

$new\_v_{t_i}$ is calculated using
$new\_v_{t_i}\_x = d * \cos(\theta_{new\_v_{ti}})$
$new\_v_{t_i}\_y = d * \sin(\theta_{new\_v_{ti}})$
where d is the magnitude of $old\_v_{t_i}$.

A new v_{d_i} is calculated by removing $old\_v_{t_i}$ and adding $new\_v_{t_i}$, given in equation 3.28

$$v_{d_i} = v_{d_i} - old\_v_{t_i} + new\_v_{t_i} \tag{3.28}$$

This method is not very robust but it is helpful in simple scenarios when local minima problem occurs due to only one obstacle. As in our experiments we are using cylindrical obstacle of the same size. So a local minima problem caused by this obstacle is easily solvable.

3.2.7 Emergency stop

We are working with non-holonomic vehicles and in our implementation, vehicles can only move forward. A reverse movement is not considered. This can lead vehicles to situations where there is a chance of a collision if a vehicle wants to move forward. So another rule is implemented to avoid collisions when other obstacles/robots are very near. In this rule, the vehicle will stop if the angle from obstacle/robot is $-45^o < \theta < 45^o$ and the distance is less than 2 meters. The robot can be stopped by setting

$Vdi = 0$

3.2.8 Avoid abrupt changes

As we are dealing with velocity vector a change occurs for this vector at every time step. Hence there is a possibility of abrupt changes in this velocity vector. For example a robot is moving towards its goal and at every time step a very little change occurs for velocity vector. Suddenly, the robot detects an obstacle and this causes an abrupt change in the velocity vector. This kind of abrupt changes can lead to unsmooth motion of the robot. So there is a requirement to evade these abrupt changes. The filter equation (3.29) is used to avoid abrupt changes in direction and magnitude of v_{d_i}. Here $\tilde{v}_{d_i}(t+1)$ is computed depending on the previous $\tilde{v}_{d_i}(t)$ and computed v_{d_i}. At time $t = 0$, $\tilde{v}_{d_i} = 0$.

$$\tilde{v}_{d_i}(t+1) = [1 - a(\Delta t)]\tilde{v}_{d_i}(t) + (a\Delta t)v_{d_i} \tag{3.29}$$

where the range of a is $[0, 1]$. We have used $a = 0.7$ and $\Delta t = 0.1$ seconds in equation (3.29).

3.3 Motion control of differential drive robots

Now the question is how to move a robot if the velocity vector (equation 3.29) is given as an input to robot? It depends on the type of robot we are using. Motion control of omni-directional, differential drive and car like robots is different. In our work a differential drive robot is used because it is easily controllable and a car like motion is easily implementable.

To find linear and angular velocities from the velocity vector (3.29), a proportional controller is used (discussed in section 2.2.1 chapter 2).

$$\begin{bmatrix} v_t \\ w \end{bmatrix} = \begin{bmatrix} k_p & 0 \\ 0 & k_\theta \end{bmatrix} \times \begin{bmatrix} E_{pos} \\ E_\theta \end{bmatrix} \tag{3.30}$$

Where

$$E_{pos} = \sqrt{D_x{}^2 + D_y{}^2}$$

$$E_\theta = atan2(D_y, D_x)$$

Equation (3.30) (already explained in section 2.2.1) is used, where values of the proportional terms are selected as $k_p = 0.09$ and $k_\theta = 0.3$ after experimentation. E_{pos} and E_θ are computed by setting values

$D_x = \tilde{v}_{di_x}$
$D_y = \tilde{v}_{di_y}$
in equation (3.30).

Car like movement

One of the aims was to test the proposed algorithm using robots having car like motions. Here we are working with differential drive robot whose controllability is easy as compared to car like robots. Both robots are non-holonomic robots, the only difference between motions is that car like robots can move in a circular path but there is a limited turning radius, while differential drive robots have the ability of zero degree rotation (it can rotate at a point). Differential drive robots can achieve the motion of car like robots by simply limiting the turning radius.

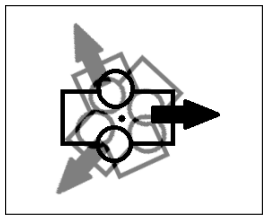
Three different cases are possible in the case of a differential drive robot

i). $v_l = v_r$ moving on a straight line and R becomes infinite. In this case the angular velocity will be $w = 0$

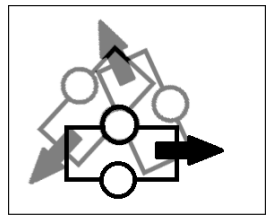
ii). $v_l = 0$ rotation about the left wheel figure (fig. 3.12(b)) and $v_r = 0$ rotation about right wheel see figure (fig. 3.12(b)).

iii). $v_l = -v_r$, In this case $R = 0$ and zero degree rotation is performed. The robot rotates at a point see figure (fig. 3.12(a)).

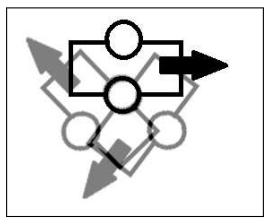
In the 3rd case the turning radius is $R = 0$. And in the 2nd case the turning radius is very small as shown in figure (fig. 3.12). Differential drive robots can give a car like movement if the turning radius R is limited to R_{min}, when the



(a) At $v_l = -v_r$, Turning radius $R = 0$



(b) At $v_l = 0$, Turning around left wheel



(c) At $v_r = 0$, Turning around right wheel

Figure 3.12: Turning behaviour when turning radius R is small

robot is turning. The angular velocity of a car like robot can be calculated using equation (3.31) [8].

$$\dot{\theta} = \frac{v}{l} \tan(\phi) \tag{3.31}$$

Where

$\dot{\theta}$ = angular velocity of car
ϕ = steering angle of car
v = linear velocity
l = length of car

From geometrical considerations we have [8]

$$\dot{\theta} = v/R \tag{3.32}$$

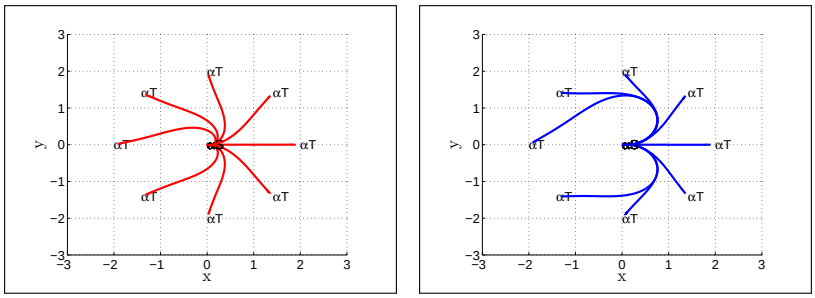
Comparing equations 3.31 and 3.32, R_{min} can be found

$$R_{min} = l/\tan(\phi_{max})$$

Erratic robot (differential drive robot) has a length $l = 0.4$ meters and considering a maximum steering angle $\phi_{max} = 30$ degrees

$$R_{min} = 0.6928$$

After calculating Rmin, (Algorithm- 2) is implemented to limit the value of angular velocity w, when robot is turning. In figure (fig. 3.13) differential drive



(a) Normal motion of differential drive robot

(b) Car like motion

Figure 3.13: Comparison of diffrential drive and car like motions of a differential drive robot. (a) no restriction on the turning radius (b) with a restriction on the turning radius

motion and car like motion are shown where the difference between them can be seen.

Algorithm 2 Limiting angular velocity w of a differential drive robot, for car like motion

$\mathrm{R}_{min} = 0.6928$
$w = \frac{v}{\mathrm{R}_{min}}$
angular velcoity w is computed
$\mathrm{sign} = (w < 0.0\ ?\ -1.0\ :\ 1.0)$
Condition
if $\frac{v}{|w|} < \mathrm{R}_{min}$ **then**
$w = \frac{v}{\mathrm{R}_{min}} \times \mathrm{sign}$
end if

Chapter 4
Results

4.1 Introduction

Extensive experiments are performed to check the validation of the MBO navigation strategy. Results of these experiments are presented in this chapter. Limitations of MBO navigation strategy are also discussed by showing some scenarios where mobile robots run into problems. Experiments which are described here were performed mainly in one basic experimental setup.

First some experiments were performed to test the fuzzy rules for obstacle avoidance. Second, experiments were performed to test MBO for different numbers of mobile robots moving in a small area. In these experiments simple scenarios were considered and detailed results are described. Experiments were performed for 3, 4 and 5 robots.

Third, for extensive experimentation to check the validation of MBO, an experimental setup is used where random scenarios were generated. Experiments were performed with and without obstacles; for 3, 4 and 5 robots, 20 random scenarios were generated for each experiment.

4.2 Software framework and packages

Our implementation was carried out using ROS (Robot operating system). Gazebo (3D simulator) and erratic-robot (differential drive robot) packages in ROS were used for testing.

4.2.1 Robot Operating System (ROS)

Robot Operating System (ROS) is a software framework, having an operating system like functionality. ROS provides operating system services including hardware abstraction, low level device control, implementation of commonly-used functionality, message passing between processes, and package management [10]. ROS also provides device drivers, libraries, visualizers and more

to help software developers to produce robot applications. ROS is a platform where more capable robot applications can be quickly and easily built.

4.2.2 Gazebo

Gazebo is a 3D multi-robot simulator for outdoor and indoor environments [11]. It provides an accurate model of reality. Gazebo provides simulation of standard robot sensors, models for used robot types and simulation of rigid-body physics etc. It also provides plug-in modules, so users can develop their own robot or sensor models. Gazebo is compatible with ROS and Player [24].

4.2.3 Erratic robot

An erratic robot is a differential drive robot. It is selected for testing, as the 'erratic_robot' stack is available in ROS. This stack has basic configuration, hardware drivers, sensor drivers and simulation for erratic robot in Gazebo. A differential drive plug-in for Gazebo simulator is available in the 'erratic_robot' stack.

4.3 Experimental setup

Traditional PF algorithm is implemented in ROS package 'potentialfields'. ROS node can be started on any robot with different options. Traffic rules TF and MBO are options and can be selected with the following 4 combinations given in table (4.1).

	MBO	TF	Algorithm behavior
1	OFF	OFF	Traditional PF
2	OFF	ON	Traditional PF with TF
3	ON	OFF	Traditional PF with MBO
4	ON	ON	Traditional PF with TF and MBO

Table 4.1: possible options combination of algorithm

The navigation strategy is tested in Gazebo simulator in ROS, using different scenarios having different numbers of erratic robots. Each 'erratic robot' is equipped with a laser range finder for obstacle detection. The idea is that every robot will be using its local frame to compute its velocity vectors and will try to avoid other robots to reach its goal position.

Experiments are performed in different scenarios and in every scenario we are logging the data (linear and angular velocities, position in map at time t) of each robot. In our work, five parameters are used for the performance evaluation of each robot. These are accuracy, time, path length, curvature change (CC) and lateral stress (LS).

4.3.1 Evaluation parameters

Performance evaluation of mobile robot motion methods is a difficult problem. The difficulty is that such methods cannot be tested offline. When evaluation is done online, these methods behave differently on different tasks and environments. In [5], an evaluation framework is given for the testing of motion methods of mobile robots. This framework can be used to evaluate and compare different motion methods but only on a single robot.

On the other hand, MBO navigation strategy can be evaluated only for multiple robots navigating in a shared area where every robot uses same method for navigation. Which makes it more difficult to evaluate multi-robot navigation methods. In this thesis work, an evaluation framework proposed by [5] is used and evaluation parameters are computed for every robot taking part in multi-robot navigation in a shared area. The evaluation parameters used are as follows.

Accuracy

Accuracy is the distance between robot's final position and goal position. And this parameter tells the robot was able to reach its goal position and how far it was from its goal? Accuracy can be calculated by using equation

$$\mathrm{Accur_i} = \|\mathrm{x_{ti}} - \mathrm{x_{finalpos}}\| \tag{4.1}$$

Completion Time

It is the time in seconds that a robot will take to complete its navigation task. For a good performance a robot should have low execution time. (ROS simulation time is considered as we are working in simulation)

Path length

Path length is the total distance covered by the robot to reach its goal position. Shorter path lengths are desirable for a better performance.

Curvature change (CC)

This parameter is associated with car-like robots and it is helpful to find the oscillations in trajectory [5]. Since we are getting the information of linear and angular velocities of robots, curvature can be found using equation (4.2) [5].

$$\mathrm{k(i)} = \frac{w(\mathrm{i})}{v(\mathrm{i})} \tag{4.2}$$

Where $w(i)$ and $v(i)$ are angular and linear velocities respectively. The parameter "curvature change" is computed as

$$CC = \frac{\sum_{i=1}^{N}(|k(i)'| \,.\, \Delta t)}{N} \tag{4.3}$$

where
N is total number of values

Small value of CC is desirable for smooth robot motion.

Lateral stress (LS)

This parameter helps to find the behavior of robot while turning. It is computed using equation (4.4) [5]

$$LS = \sum_{i=1}^{N}(\frac{v(i)^2}{|r(i)|} \,.\, \Delta t) \tag{4.4}$$

where $r(i) = \frac{1}{k(i)}$

Where $v(i)$ and $r(i)$ are linear velocity and curvature radius respectively. If the robot is turning at high speed it means LS will be high. Less value of LS is desirable for safe turning.

All these parameters can not achieve required values at the same time. Its possible that a robot having less completion time can have higher curvature change (same for other parameters).

4.4 Working of 'mboNavigation' stack in ROS and some assumptions

In this section, a very basic understanding of 'mboNavigation' stack is described. Detailed information of this stack is available in the appendix. The 'mboNavigation' ROS stack has three ROS packages.

`tf_map`

In this package one node is implemented named '`robotPositionsInmap`' figure (fig. 4.1). This node has one output named '`robotPositionsInmap`', This output data has information of exact positions of robots in world map. One of the assumptions is that every robot knows other robot's position in world map.

`Obstacle_detection`

Consider we are working with '`alpha`' robot. In this package one node is implemented named '`alpha_obstacleInfo`' figure (fig. 4.1). This node has two

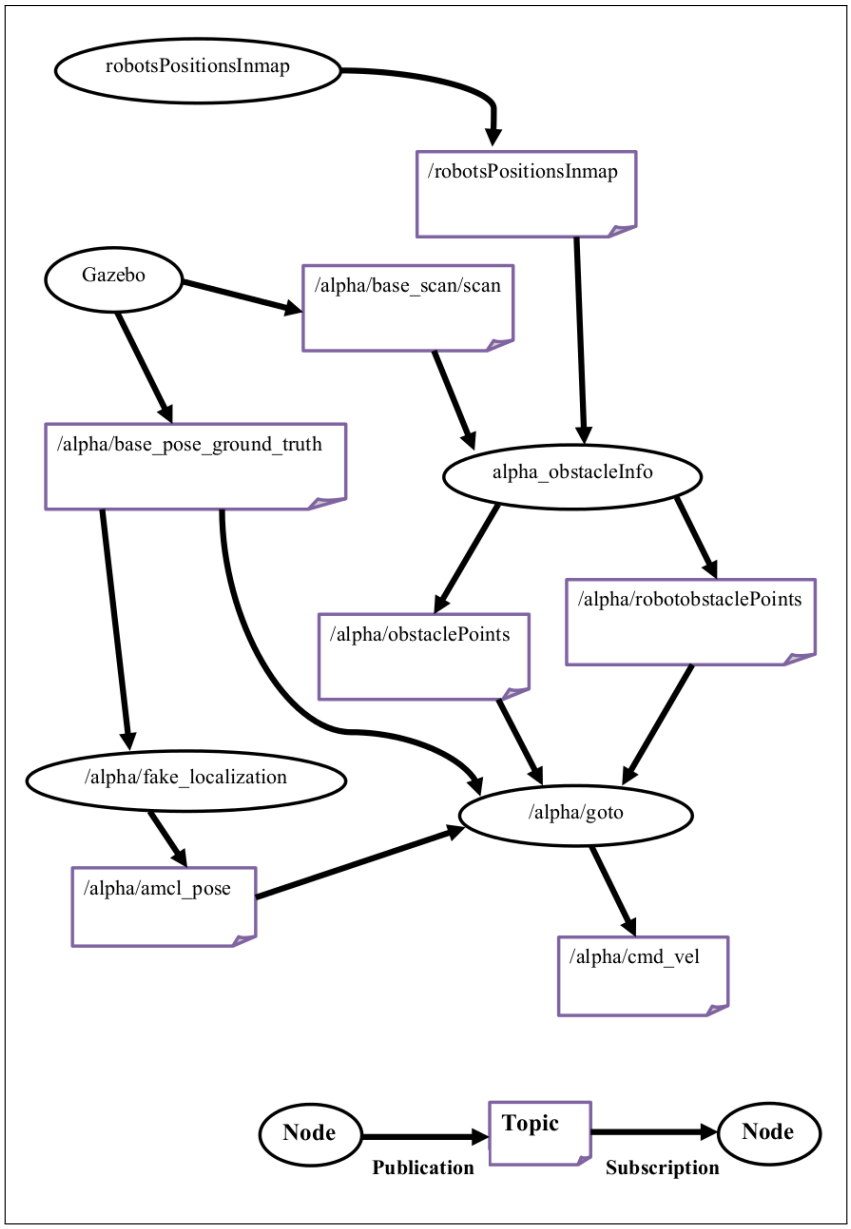


Figure 4.1: visualization of messages publish/subscribe on topics between ROS nodes

inputs '/alpha/base_scan/scan' and 'robotsPositionsInmap'. Input data '/alpha/base_scan/scan' provides the information of sensor laser range finder. After differentiating other robot points from obstacle points, this node has two outputs '/alpha/obstaclePoints' and '/alpha/robotobstaclePoints'. These two outputs provide positions of obstacles and other robots in the local frame of 'alpha' robot. Another assumption is that we are using a cylindrical obstacle of same diameter and height as mentioned in chapter (3). We are not considering walls and obstacles of different shapes.

potentialFields

Consider we are working with 'alpha' robot. In this package one node is implemented named 'alpha_goto'. Navigation strategy is implemented in this node. This node takes four inputs as shown in figure (fig. 4.1), and gives one output '/alpha/cmd_vel'. '/alpha/cmd_vel' output data provides linear and angular velocities to the 'alpha' robot to move.

4.4.1 ROS launch File

To run the robots a launch file is written that helps to run all the nodes associated with every robot. In this launch file initial and target positions of robots are set. The initial position, goal position and orientation of each robot is selected in map frame. To run a scenario, the launch file needs

Initial position of each robot in map (x, y, θ)
Orientation (θ) of robot in map is selected in the direction of goal
Target position of each robot in map (x_t, y_t)

4.5 Results

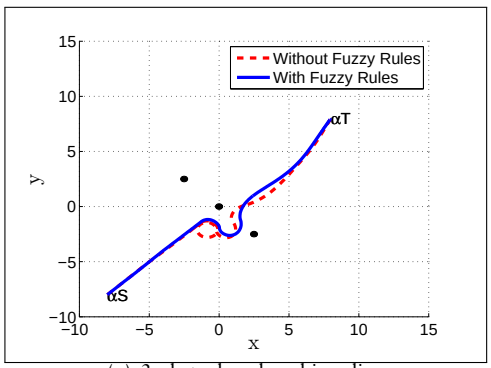
4.5.1 Obstacle avoidance (Comparison with and without fuzzy rules)

Experiments were run to check how fuzzy rules can optimize obstacle fields? Three different tests were run.

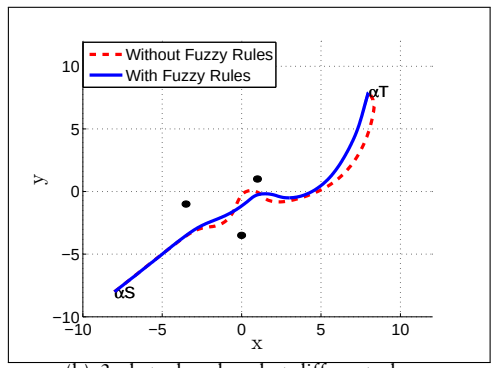
In the first test, 3 obstacles were placed between robot initial and target positions. The robot's movement was tested with and without fuzzy rules as shown in figure (fig. 4.2(a)). In figure (fig. 4.2(a)), it can be seen that without fuzzy rules due to unwanted rotation of the robot near obstacles, the robot took more time to reach its target while in the case of using fuzzy rules, robot can pass between two obstacles as shown in figure (fig. 4.2(a)).

In the second test, 3 obstacles were placed with different configuration (non collinear). It can be seen in figure (fig. 4.2(b)) that without fuzzy rules, the robot path is strangely effected by obstacle potential fields.

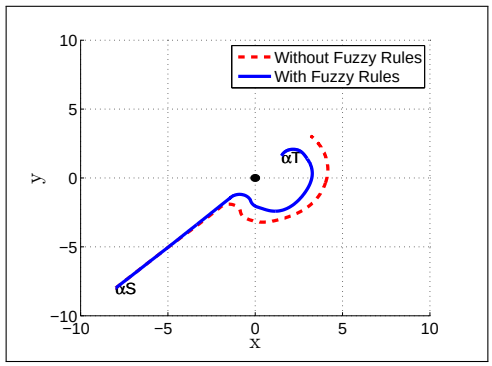
In the third test, the goal point was in the effective range of obstacle. In figure (fig. 4.2(c)), it can be seen that without fuzzy rules the robot was un-



(a) 3 obstacles placed in a line



(b) 3 obstacles placed at different places



(c) Goal in obstacle range

Figure 4.2: Performance comparison with and without fuzzy rules. α_S (start position) and α_T (target position)

able to reach this goal point, while in case of using fuzzy rules robot reached successfully at its target position.

To understand that how fuzzy rules can affect the movement of robot, an example is shown in figure (fig. 4.3) where detail information of fuzzy coefficients is gathered at two different points. At point P1, 'α' robot sensed three obstacles. 'α' robot detected 12 points on obstacle 1, 7 points on obstacle 2 and 16 points on obstacle 3. After that 'α' robot calculated fuzzy coefficients for the repulsive PF generated by these obstacle points. It can be analyzed from the assigned fuzzy coefficient values that robot had less effect of repulsive PF generated by obstacle 1 and 3 as these obstacles were not in the way to the target position.

At point P2, 'α' robot detected 22 points on obstacle 1, 15 points on obstacle 2, 12 points on obstacle 3 and calculated fuzzy coefficients for the repulsive PF generated by all points detected. From the fuzzy coefficient values in this case it can be analyzed that effect of obstacle 1 and 3 was reduced while obstacle 2 generated strong repulsive PF as it was in the way to the target.

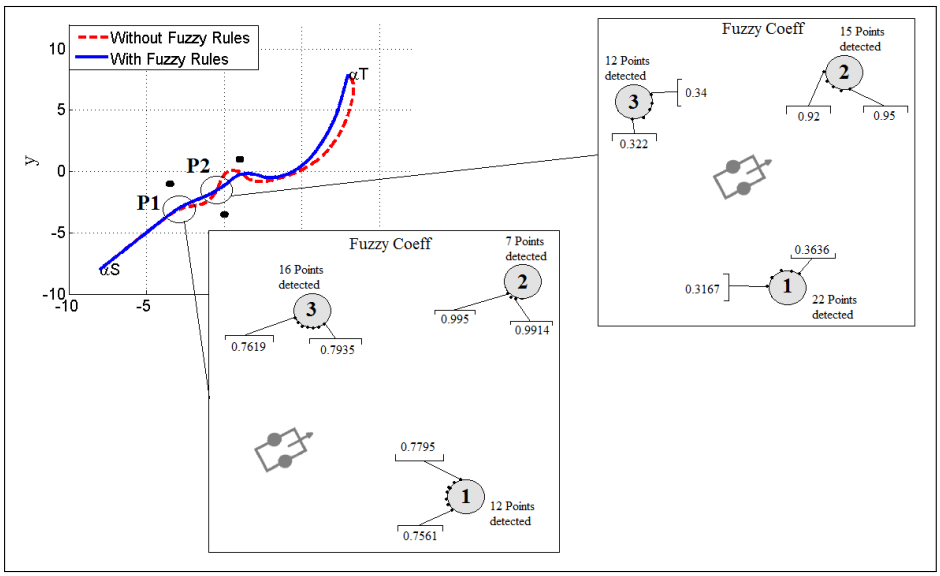


Figure 4.3: Detail information of fuzzy coefficients at two different positions. At P1 and P2 'α' robot is less effected by repulsive PF of obstacles 1 and 3 as these obstacles are not in the way to the target.

4.5.2 Results simple scenarios

MBO navigation algorithm was tested on simple scenarios. In these scenarios, robots interact in a small area while reaching at their goals. Experiments are performed with and without obstacles.

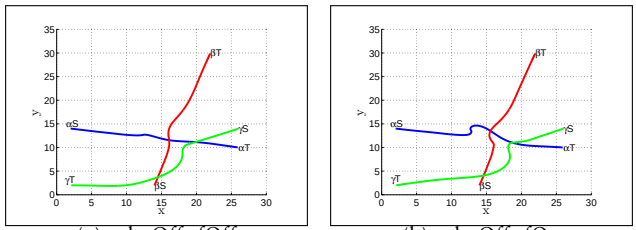
Movement of 3 Robots

Without obstacles In the case of 3 robots, initial and target positions were selected for robots 'α', 'β' and 'γ'. These positions were selected in such a way that while moving, these robots were interacting with each other. 3 robots reached their target positions under the four combination of traffic rules and MBO (given in table (4.1)). Navigation results can be seen in figure (fig. 4.4). Detailed results of each robot can be seen in table (4.2).

In case of 'mbo-OFF and tf-OFF' normal potential fields were affecting each other (fig. 4.4(a)). In case of option 'mbo-OFF and tf-ON', traffic rules were turned on. In figure (fig. 4.4(b)) it can be analyzed that 'α' robot is influenced by 'β' and 'γ' robot that force 'α' robot to move on the left side. At the same time 'α' robot found 'β' robot at its right side and applied 'slow down' traffic rule that caused 'αlphα' robot to move left side at a slow speed. For 'γ' robot it can be seen that it detected 'β' robot on its right side and applied 'slow down' traffic rule. Same for 'β' robot that it detected 'γ' robot on its right side and applied 'slow down' traffic rule. As all the robots were applying 'slow down' traffic rule so these took more time to reach at their target positions.

In case of 'mbo-ON and tf-OFF', MBO option is turned on. In figure (fig. 4.4(c)) 'β' robot showed a smooth motion as it assigned weights to 'α' and 'γ' robot using MBO method. These weights decreased the influence of repulsive PF generated by 'α' and 'γ' robots, and helped in smooth motion of 'β' robot. 'γ' robot showed smooth motion by assigning weights to 'α' and 'β' robots, which helped to reduce the influence of repulsive velocity vectors of 'α' and 'β' robots. In case of 'mbo-ON and tf-ON', both options were turned on. In figure (fig. 4.4(d)) navigation results can be seen where every robot used MBO method to assign weights but traffic rules were not applied as conditions to apply traffic rules were not fulfilled.

Table (4.2(d)) is showing the combined (averaged) result of 3 robots. It can be seen from (4.2(d)) that in case of MBO navigation strategy, the robots took less time and covered less distance to complete the navigation task. CC and LS parameters are showing that movement was smooth when this algorithm was used.



(a) mboOff-tfOff (b) mboOff-tfOn

(c) mboOn-tfOff (d) mboOn-tfOn

Figure 4.4: Movement of 3-Robots without obstacles: Navigation results of different options of the navigation algorithm

(a) Result of alpha robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.100	0.100	0.099
Time	68.100	71.400	63.000	63.900
Length	24.370	26.861	24.269	24.350
CC	0.317	0.992	0.355	0.425
LS	2.270	3.345	2.085	2.185

(b) Result of beta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.100	0.099	0.099
Time	67.000	69.100	65.600	67.000
Length	29.311	29.624	29.125	29.192
CC	1.681	2.375	1.459	1.558
LS	1.106	1.293	1.073	1.031

(c) Result of gamma robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.100	0.099	0.099
Time	68.900	70.200	65.800	66.500
Length	29.380	29.757	27.668	27.926
CC	0.735	0.731	0.557	0.480
LS	2.873	3.201	2.553	2.683

(d) Average result of all robots

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Avg Accur	0.100	0.100	0.099	0.099
Avg Time	68.000	70.233	64.800	65.800
Avg Length	27.687	28.748	27.021	27.156
Avg CC	0.911	1.366	0.790	0.821
Avg LS	2.083	2.613	1.904	1.966

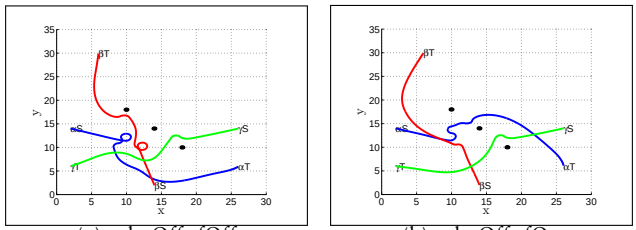
Table 4.2: Movement of 3-Robots without obstacles: Performance comparison of different options of the navigation algorithm

With obstacles In this experiment, the navigation of 3 robots was tested in a shared area where static obstacles were also placed. Starting and target positions were selected for robots 'α', 'β' and 'γ'. Three obstacles were placed in shared area. These positions were selected in such a way that while moving, these robots were interacting with each other and sensing static obstacles at the same time. With different options (from table 4.1) of the algorithm, all the robots were able to reach their target positions.

Navigation results can be seen in figure (fig. 4.5). When option 'mbo-OFF and tf-OFF' is selected, robots used traditional potential fields to reach their target positions, it can be seen in figure (fig. 4.5(a)) that robots 'α' and 'β' showed unwanted rotational movements. From the movement of 'γ' robot it can be analyzed that 'γ' robot first turned right because of the obstacle and then turned left because of the presence of 'α' and 'β' robots.

When MBO method was introduced in option 'mbo-ON and tf-OFF', repulsive potential fields of other robots were optimized. Figure (fig. 4.5(c)) is showing the smooth motion of all the robots. In case of 'mbo-ON and tf-ON', both the options were turned on. Every robot used MBO method to assign weights to other robots. It can be analyzed in figure (fig. 4.5(d)) that 'α' robot showed smooth motion as it assigned less weights to robots 'β' and 'γ'. Same was the case for 'β' and 'γ' robots. When 'α' and 'β' robot came closer to each other traffic rules were turned on. 'β' robot detected 'α' robot on left side and applied 'turn right' traffic rule while 'α' robot detected 'β' robot on right side and applied 'slow down' traffic rule.

Detailed results of each robot can be seen in tables (4.3). The accuracy parameter shows that all robots got success while reaching their target positions. 'α', 'β' and 'γ' robots took less time in case of option 'mbo-ON and tf-ON'. Distance covered by each robot was also less when MBO method was used. In case of using the MBO method the CC value was less for 'α' and 'β' robots but for 'γ' robot it was almost same as compared to traditional PF. The value of LS parameter was also less when the MBO method was used showing the safe turning of robots. A combined (averaged) result of this scenario is shown in table (4.3(d)), a better performance due to MBO can be clearly observed from the different values of the metrics.



(a) mboOff-tfOff (b) mboOff-tfOn

(c) mboOn-tfOff (d) mboOn-tfOn

Figure 4.5: Movement of 3-Robots with obstacles: Navigation results of different options of the navigation algorithm

(a) Result of alpha robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.100	0.100	0.099
Time	93.600	97.600	70.000	71.300
Length	36.626	34.636	28.171	28.690
CC	1.762	1.547	0.703	0.513
LS	5.378	4.880	2.573	3.228

(b) Result of beta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.099	0.100	0.099
Time	93.300	81.900	77.300	76.999
Length	36.548	33.017	32.249	32.314
CC	3.459	1.092	2.153	2.374
LS	4.406	1.711	2.188	2.481

(c) Result of gamma robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.100	0.100	0.099
Time	90.700	77.100	70.100	70.900
Length	28.402	28.646	26.213	26.254
CC	0.413	0.434	0.381	0.428
LS	3.003	2.874	2.770	2.748

(d) Average result of all robots

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Avg Accur	0.100	0.099	0.100	0.099
Avg Time	92.533	85.533	72.467	73.066
Avg Length	33.859	32.099	28.878	29.086
Avg CC	1.878	1.024	1.079	1.105
Avg LS	4.263	3.155	2.511	2.819

Table 4.3: Movement of 3-Robots with obstacles: Performance comparison of different options of the navigation algorithm

Movement of 4 Robots

Without obstacles In this experiment, the algorithm was tested with 4 robots 'α', 'β', 'γ' and 'δ'. Starting and target positions were selected in such a way that robots were interacting with each other while moving towards their target positions.

Navigation results with different options (from table 4.1) are shown in figure (fig. 4.6). In case of using traditional PF, it can be seen in figure (fig. 4.6(a)) and (fig. 4.6(b)) that repulsive potential fields of robots are affecting each other. When 'mbo-ON and tf-ON' option was turned on, every robot used MBO method to assign weights. These weights helped each robot to reduce the effect of repulsive PF generated by other robots and showed smooth motion while moving towards its target position (fig. 4.6(d)).

Detailed results of each robot can be analyzed from tables (4.4). Combined (averaged) result (table 4.4(e)) of this scenario is showing that all the robots took less time and covered less distance when the MBO navigation strategy was used. Smoothness of each robot can be analyzed from CC and LS parameters.

It can be analyzed from the results in figures (fig. 4.6(a)) and (fig. 4.6(b)) have similar trajectories and same for figures (fig. 4.6(c)) and (fig. 4.6(d)). However it raises the question that why traffic rules have no effect on the trajectories. The reason is that in this scenario the robots act far away from eachother for using traffic rules.

In this scenario in case of option tf-ON, robots were not so near and conditions for using traffic rules were not fulfilled, that's why results are same in cases when traffic rules were turned OFF or ON.

(a) Result of alpha robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.100	0.100	0.099
Time	70.400	70.300	66.900	66.900
Length	30.694	30.560	29.156	29.108
CC	0.496	0.428	0.503	0.418
LS	3.435	3.451	3.169	3.138

(b) Result of beta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.099	0.099	0.100
Time	71.400	71.100	67.600	67.700
Length	31.397	31.390	30.049	30.079
CC	1.863	1.860	2.038	1.864
LS	1.505	1.522	1.375	1.369

(c) Result of gamma robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.100	0.099	0.099
Time	69.700	69.600	68.700	68.600
Length	30.213	30.120	28.880	28.808
CC	0.412	0.462	0.388	0.392
LS	3.284	3.302	2.774	2.608

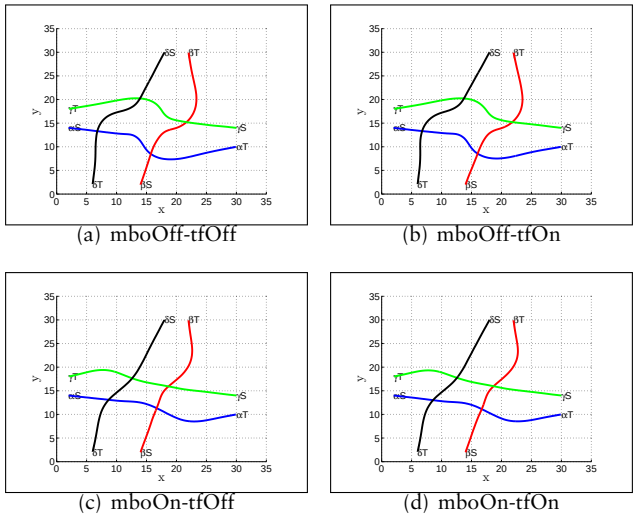
(d) Result of delta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.099	0.100	0.100
Time	69.100	68.900	67.200	67.100
Length	32.318	32.194	30.934	30.937
CC	2.842	2.052	1.386	1.678
LS	2.096	1.984	1.739	1.705

(e) Average result of all robots

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Avg Accur	0.100	0.100	0.099	0.099
Avg Time	70.150	69.975	67.600	67.575
Avg Length	31.155	31.066	29.755	29.733
Avg CC	1.403	1.200	1.079	1.088
Avg LS	2.580	2.565	2.264	2.205

Table 4.4: Movement 4-Robots without obstacles: Performance comparison of different options of the navigation algorithm



(a) mboOff-tfOff (b) mboOff-tfOn

(c) mboOn-tfOff (d) mboOn-tfOn

Figure 4.6: Movement of 4-Robots without obstacles: Navigation results of different options of the navigation algorithm

With obstacles In this case, the algorithm was tested with 4 robots navigating in a shared area where three static obstacles were also placed. The placement of starting and target positions of each robot was selected in such a way that while movement, robots were sensing the presence of static obstacles and other robots at the same time.

The navigation results in figure (fig. 4.7) show the motion behavior of each robot when different options (from table 4.1) were selected. In case of 'mbo-OFF and tf-OFF' it can be analyzed in figure (fig. 4.7(a)) that superimposing potential fields from obstacles and robots led to unwanted motions of the robots. 'α' robot showed rotational movement while avoiding from obstacles and other robots.

In case of 'mbo-ON and tf-ON' option, MBO method was used to assign weights. In figure (fig. 4.7(d)) it can be analyzed that all the robots showed smooth motion. Using MBO method 'α' robot was less influenced by the repulsive velocity vectors of other robots. In this scenario conditions for applying traffic rules were not fulfilled that's why results are same when traffic rules option was turned OFF or ON.

Tables (4.5) show the detailed results of each robot. In this experiment the MBO method can be compared with traditional PF from combined (averaged) result in table (4.5(d)).

(a) Result of alpha robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.100	0.099	0.099
Time	95.600	95.700	86.697	86.300
Length	37.153	37.562	33.681	33.029
CC	1.352	0.771	0.945	1.632
LS	6.535	6.300	4.430	4.324

(b) Result of beta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.099	0.100	0.099
Time	81.100	79.900	74.100	73.800
Length	29.350	29.500	30.039	29.977
CC	2.041	2.738	1.891	1.985
LS	1.081	1.231	1.054	1.069

(c) Result of gamma robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.099	0.099	0.099
Time	78.800	78.500	72.400	72.400
Length	33.253	33.362	31.200	31.365
CC	0.360	0.536	0.425	0.493
LS	3.302	3.297	2.877	3.005

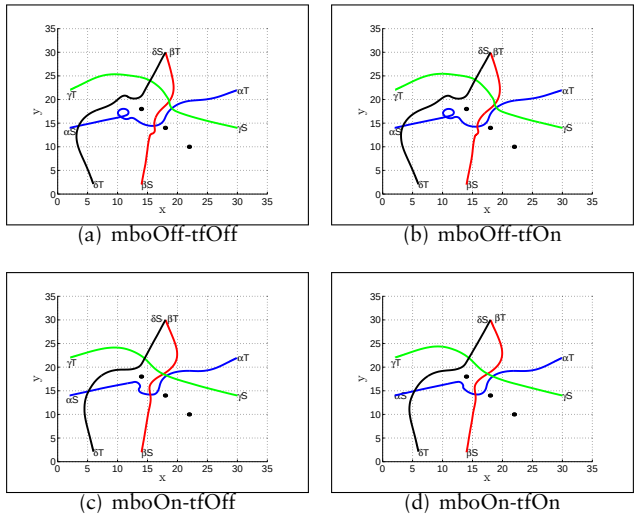
(d) Result of delta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.099	0.099	0.099
Time	83.100	82.500	74.897	74.100
Length	36.269	36.359	34.592	34.217
CC	0.967	0.991	1.581	1.389
LS	2.784	2.818	2.279	2.373

(e) Average result of all robots

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Avg Accur	0.099	0.099	0.099	0.099
Avg Time	84.650	84.150	77.023	76.650
Avg Length	34.006	34.196	32.378	32.147
Avg CC	1.180	1.259	1.211	1.375
Avg LS	3.426	3.412	2.660	2.693

Table 4.5: Movement of 4-Robots with obstacles: Performance comparison of different options of the navigation algorithm



(a) mboOff-tfOff (b) mboOff-tfOn

(c) mboOn-tfOff (d) mboOn-tfOn

Figure 4.7: Movement of 4-Robots with obstacles: Navigation results of different options of the navigation algorithm

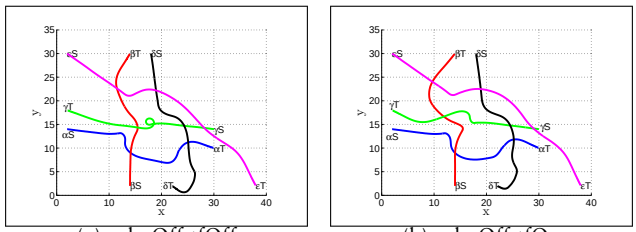
Movement of 5 Robots

Without obstacles In this experiment, the algorithm was tested with 5 robots, 'α', 'β', 'γ', 'δ' and 'ϵ'. Navigation results with different options (from table 4.1) in figure (fig. 4.8(c)) show that the MBO navigation strategy is far better than the traditional PF, because the MBO strategy was reducing the unwanted effects of superimposing potential fields.

When 'mbo-OFF and tf-OFF' option was used all the robots safely reached at their target positions but showed unwanted movements (fig. 4.8(a)). In case of 'mbo-OFF and tf-ON', traffic rules option was turned on (fig. 4.8(b)). In this case conditions for applying traffic rules were fulfilled by 'β' and 'γ' robots. 'γ' robot detected 'β' robot on left side and applied 'turn right' traffic rule. And at the same time 'β' robot detected 'γ' robot on the right side and applied 'slow down' traffic rule. Combining this rule and repulsive velocity vectors of other robots, 'β' robot moved to the left at slow speed. That's why 'β' robot took more time to reach its target position.

When 'mbo-ON and tf-ON' option was used, all the robots used MBO method to assign weights to other robots. Repulsive velocity vector generated by robots are optimized using these weights and all the robots showed smooth motions and reached at their target positions safely (fig. 4.8(d)). Conditions for applying traffic rules were not full filled in this case that's why results in figures (fig. 4.8(c)) and (fig. 4.8(d)) are similar.

Detailed results of each robot can be seen from tables (4.6). It can be analyzed from combined (Averaged) results (table 4.6(f)) of this scenario that the MBO navigation strategy performs really well compared to traditional PF.



(a) mboOff-tfOff (b) mboOff-tfOn

(c) mboOn-tfOff (d) mboOn-tfOn

Figure 4.8: Movement of 5-Robots without obstacles: Navigation results of different options of the navigation algorithm

(a) Result of alpha robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.100	0.099	0.100
Time	87.000	86.800	81.600	80.900
Length	33.851	33.378	29.201	29.097
CC	0.605	0.394	0.335	0.354
LS	4.154	3.834	2.810	2.817

(b) Result of beta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.099	0.099	0.099
Time	72.400	74.900	67.300	70.400
Length	29.751	32.625	28.797	29.706
CC	2.165	3.513	2.674	2.814
LS	0.920	1.974	0.850	0.842

(c) Result of gamma robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.099	0.099	0.100
Time	77.399	78.200	67.400	72.100
Length	33.555	30.457	28.891	29.280
CC	0.757	0.484	0.458	0.625
LS	5.725	3.525	3.303	3.154

(d) Result of delta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.099	0.099	0.100
Time	82.000	76.600	74.099	67.900
Length	35.772	34.036	33.121	31.092
CC	1.466	2.008	1.318	1.220
LS	2.228	2.055	1.627	1.407

(e) Result of epsilon robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.099	0.100	0.099
Time	89.000	89.500	84.700	84.200
Length	48.636	49.071	46.237	46.180
CC	0.616	0.730	0.694	0.708
LS	5.305	5.205	4.579	4.779

(f) Average result of all robots

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Avg Accur	0.099	0.099	0.099	0.099
Avg Time	81.560	81.200	75.020	75.100
Avg Length	36.313	35.913	33.250	33.071
Avg CC	1.122	1.426	1.096	1.144
Avg LS	3.667	3.319	2.634	2.600

Table 4.6: Movement of 5-Robots without obstacles: Performance comparison of different options of the navigation algorithm

With obstacles In this test, 5 robots were moving in a shared area in the presence of static obstacles. Placement of starting and target positions of robots were selected in such a way that while movement, robots were sensing static obstacles and other robots at the same time. In this experiment robots movement was tested with different options (from table 4.1) of algorithm. Navigation results can be seen in figure (fig. 4.9).

In case of 'mbo-OFF and tf-OFF' option, traditional PF were used that helped every robot to reach its target position safely but it caused unwanted movement of robots (fig. 4.9(a)). 'α' and 'β' robots showed unwanted rotations while moving towards their target positions. In this case each robot was affected by repulsive velocity vectors of other robots and showed unnecessary motion. In case of 'mbo-OFF and tf-ON' option, traffic rules were turned on. In figure (fig. 4.9(b)) it can be analyzed that how motion of 'β' and 'δ' robots are affected by using traffic rules. 'β' robot detected 'α' robot on the left side and applied 'turn right' traffic rule. 'δ' robot detected 'γ' robot on the right side and applied 'slow down' traffic rule that caused 'δ' robot to take more time to reach at its target position.

When 'mbo-On and tf-On' option was used, MBO method helped each robot to assign less weights to the robots that were not in the way to its target. So optimizing repulsive velocity vectors generated by other robots, each robot showed smooth motion (fig. 4.9(d)) . Conditions for using traffic rules were not full filled in this case that's why figures (fig. 4.9(c)) and (fig. 4.9(d)) are showing similar results.

Detailed results of each robot are shown in tables (4.7). Table (4.7(d)) shows the combined (averaged) result of this scenario, and the performance of the MBO method can be compared with traditional PF.

(a) Result of alpha robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.129	0.099	0.100	0.099
Time	106.500	93.700	83.000	85.300
Length	37.963	38.011	30.623	31.547
CC	2.182	1.749	0.629	1.451
LS	6.477	5.547	3.575	3.919

(b) Result of beta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.099	0.100	0.099
Time	88.700	79.400	72.600	71.800
Length	37.010	30.110	29.915	30.142
CC	2.505	4.349	2.124	2.119
LS	4.049	1.725	1.021	1.137

(c) Result of gamma robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.100	0.100	0.099
Time	82.400	78.400	71.200	71.200
Length	34.008	33.524	31.185	31.283
CC	1.294	0.838	0.427	0.543
LS	3.290	3.588	3.655	3.760

(d) Result of delta robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.099	0.125	0.099	0.100
Time	91.100	98.500	79.700	80.300
Length	36.522	34.891	31.202	31.502
CC	1.239	0.982	1.662	1.480
LS	2.905	2.435	1.321	1.444

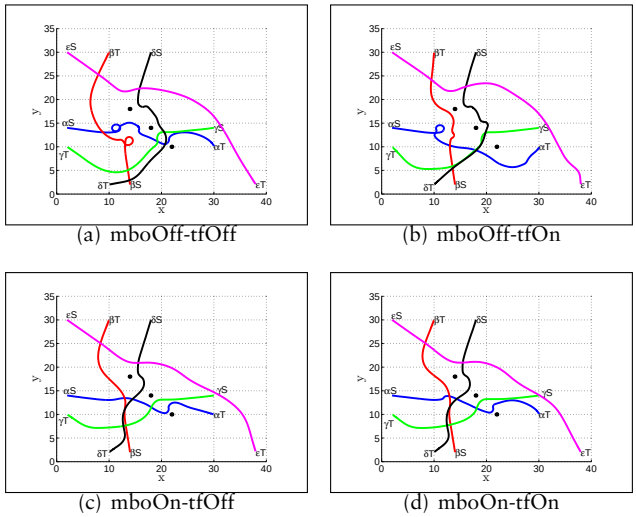
(e) Result of epsilon robot

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Accur	0.100	0.100	0.191	0.099
Time	93.900	93.300	83.200	89.200
Length	49.098	50.403	48.249	47.893
CC	0.543	1.237	0.731	0.677
LS	4.309	4.744	4.816	4.556

(f) Average result of all robots

	mboOFF-tfOFF	mboOFF-tfON	mboON-tfOFF	mboON-tfON
Avg Accur	0.105	0.104	0.118	0.099
Avg Time	92.520	88.660	77.940	79.560
Avg Length	38.920	37.388	34.235	34.473
Avg CC	1.553	1.831	1.115	1.254
Avg LS	4.206	3.608	2.877	2.963

Table 4.7: Movement of 5-Robots with obstacles: Performance comparison of different options of the navigation algorithm



(a) mboOff-tfOff (b) mboOff-tfOn

(c) mboOn-tfOff (d) mboOn-tfOn

Figure 4.9: Movement of 5-Robots with obstacles: Navigation results of different options of the navigation algorithm

4.5.3 Results of random scenarios

Experimental setup for random scenarios

Navigation algorithms can behave differently for different environments and situations. So there is a possibility that a navigation algorithm showing good results in one scenario can give unwanted results in other scenarios. MBO navigation strategy is tested on random scenarios to analyze how this method will behave for different scenarios. An experimental setup is implemented where random scenarios are generated and the MBO navigation strategy is tested on these scenarios. As discussed in [4.4.1], a launch file needs initial and target position of each robot in map frame. In the following we have computed random goal positions of robots for experimentation.

A simple program is written that computes random goal positions in a small area for each robot. For this purpose a grid map is considered where each grid cell has a size (4×4) meters, as shown in figure (fig. 4.10). The initial position of each robot is fixed every time, but the goal position of the robot is selected randomly. After selecting random grid cells, these are translated into positions of continuous map. This conversion is done by using the equation (4.5).

$$\begin{aligned} x &= 2 + M \times 4 \\ y &= 2 + N \times 4 \end{aligned} \tag{4.5}$$

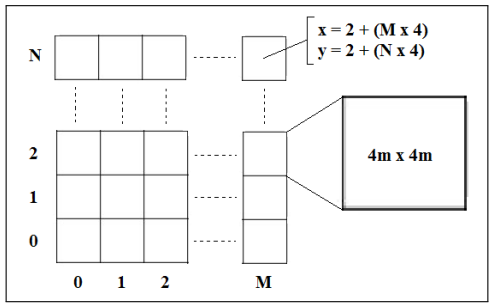


Figure 4.10: Grid map to assign robot's start and target positions

These random positions are then provided to a launch file to run a random scenario in simulation. We have tested MBO navigation strategy for different numbers of robots with random scenarios. For each scenario a combined (averaged) result was found and was compared with other scenarios results.

A specific time period was given as an upper bound before stopping the simulation. Different number of robots required different time periods for simulation. 7, 10 and 12 minutes were given to test 3 robots, 4 robots and 5 robots scenarios, respectively. Failures can occur when one/many robots are not able to reach their goal positions within a given time.

Results for 20 random scenarios are shown in graphs separately for time, length, CC and LS. For every scenario an average result (results of all robots was averaged) was taken against different options (from table 4.1).

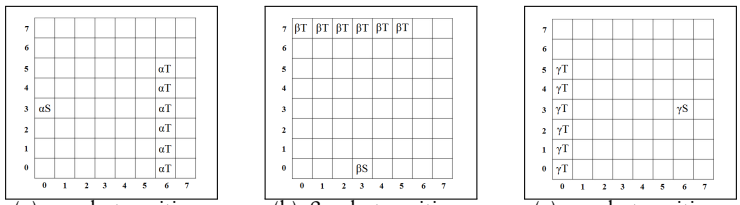
Combined results of 20 random scenarios are calculated as shown in figure (fig. 4.13). The average result of each option in every scenario is normalized with respect to non-optimized option (mbo-OFF and tf-OFF) in the respective scenario. It can be easily understood from table (4.8) how each option is normalized. Here options with normalized values < 1 are better than the non-optimized option (mbo-OFF and tf-OFF). By doing this normalization, the results of every scenario are equally scaled. Median, mean and standard-deviation of normalized results of 20 random scenarios are calculated as shown in figure (fig. 4.13) where comparison of different options of algorithm is quite easy to understand. These results will help us to understand how MBO navigation strategy behaves on random scenarios.

Movement of 3 Robots in random scenarios

Without obstacles The MBO navigation strategy was tested with the 3 robots 'α', 'β' and 'γ' in random scenarios. Starting positions of these robots were fixed, but target positions were selected randomly. It can be easily understood from figure (fig. 4.11) where 'α_S', 'β_S' and 'γ_S' are fixed starting positions and target positions from 'α_T', 'β_T' and 'γ_T' can be selected randomly.

	MBO	TF	Normalized result Result=(Time/Length/CC/LS)
Opt_1	OFF	OFF	(Opt_1/Opt_1) = 1
Opt_2	OFF	ON	(Opt_2/Opt_1) = Opt_2 Normalized Result
Opt_3	ON	OFF	(Opt_3/Opt_1) = Opt_3 Normalized Result
Opt_4	ON	ON	(Opt_4/Opt_1) = Opt_4 Normalized Result

Table 4.8: Normalization of results against each option of algorithm with respect to result of non-optimized option (mbo-OFF and tf-OFF)

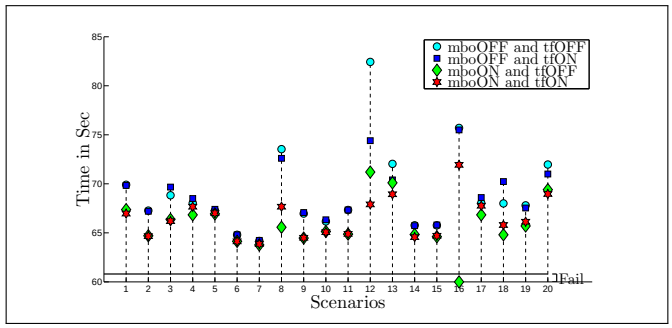


(a) α robot positions (b) β robot positions (c) γ robot positions

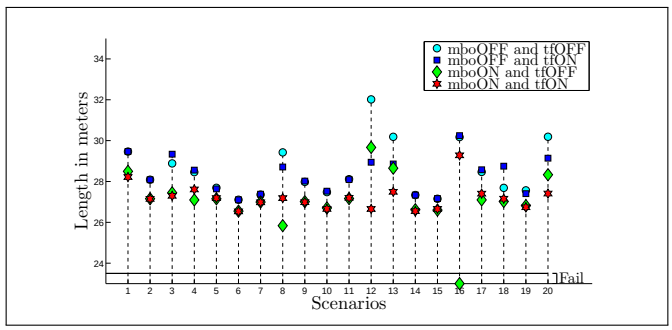
Figure 4.11: Placement of 3 robots in random scenarios without obstacles: possible grid map positions

Results of 20 random scenarios in figure (fig. 4.12) shows what the effects of different options of the algorithm on each performance metric are. Figure (fig. 4.12(a)) shows the time comparison of 20 random scenarios. It can be analyzed that time taken by the MBO method in each scenario was less as compared to traditional PF. Only one failure occurred when option 'mbo-ON and tf-OFF' was used in scenario 16, where one/many robots were not able to reach their target positions. From figure (fig. 4.12(b)) it can be seen that the distance covered in case of using the MBO method was less in all random scenarios. But from CC values of random scenarios in figure (fig. 4.12(c)) it can be analyzed that the MBO method was not able to reduce CC. As only in some scenarios the CC value was less when the MBO method was used. CC values computed against different options of algorithm in case of scenario 8, are totally opposite to the requirement. In figure (fig. 4.12(d)) LS values in all random scenarios is less when the MBO method was used.

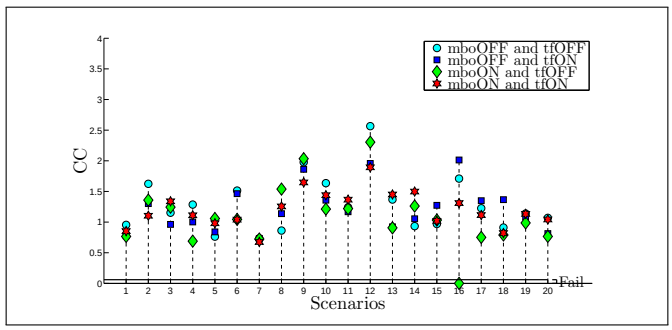
Combined results can be seen in figure (fig. 4.13), where for all 20 random scenarios median and mean values in case of (mbo-ON and tf-ON) are better as compared to other options. Standard deviation is showing different behavior with different evaluation parameters. In results of CC parameter against each option, large standard deviation can be analyzed.



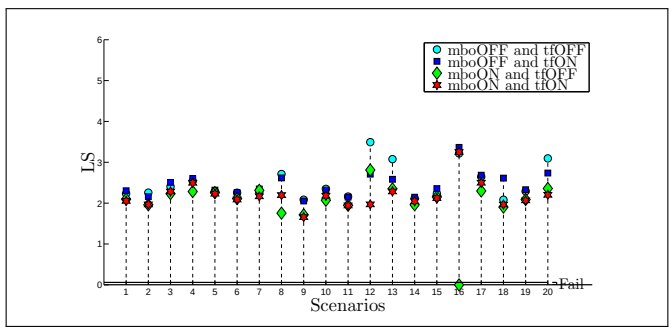
(a) Random scenarios: time comparison



(b) Random scenarios: path length comparison



(c) Random scenarios: CC comparison



(d) Random scenarios: LS comparison

Figure 4.12: Movement test of 3 Robots in random scenarios without obstacles: Comparison results of different options of algorithm

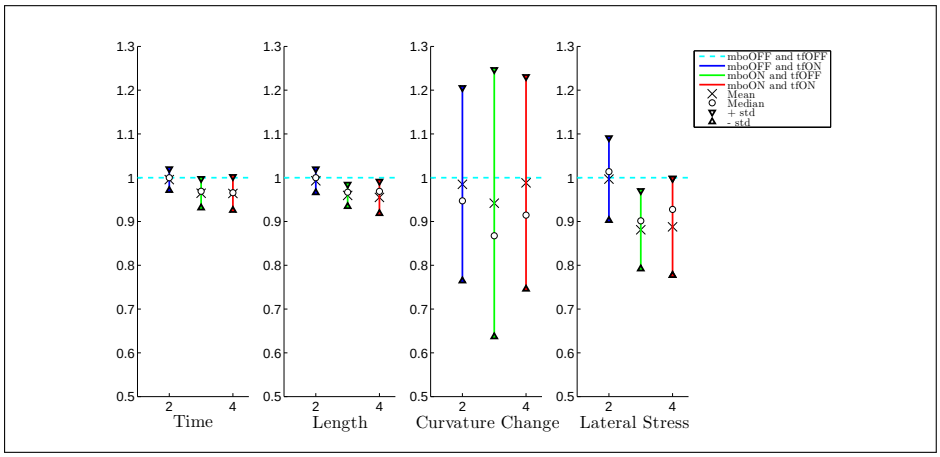


Figure 4.13: Comparison: Median, Mean and standard deviation 20 random scenarios after normalizing cases w.r.t. non-optimized case

With obstacles In this experiment, the algorithm was tested with 3 robots in a shared area where obstacles were placed. Experiment was performed with 20 random scenarios, where target positions of robots were selected randomly. Possible positions of all robots are shown in figure (fig. 4.14).

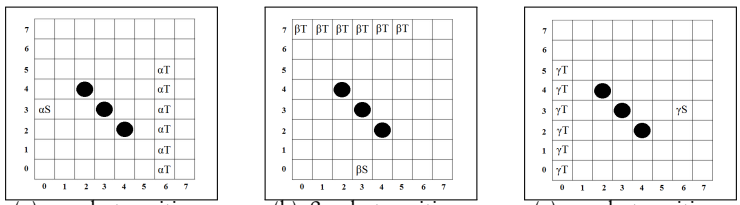
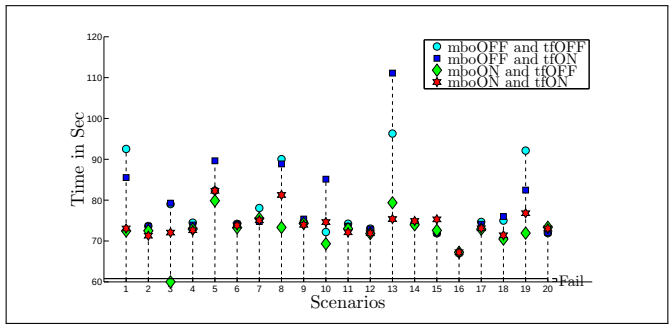
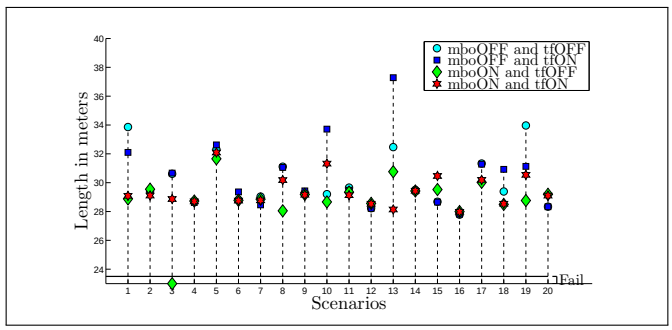


Figure 4.14: Placement of 3 robots in random scenarios with obstacles: possible grid map positions

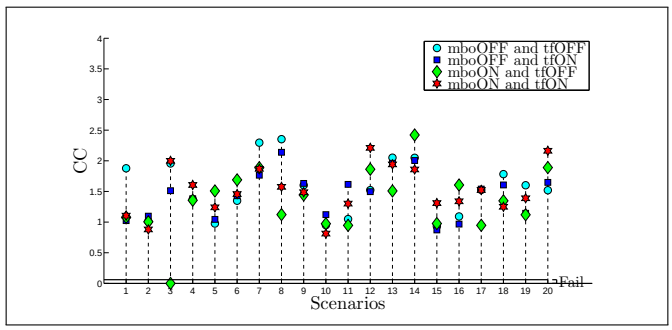
Results are shown in figure (fig. 4.15), where parameters time, length, CC and LS are shown for different scenarios. It can be seen in figure (fig. 4.15(a)) that time taken in case of the MBO method was less as compared to traditional PF in most scenarios. One failure occurred in scenario 3 when option 'mbo-ON and tf-OFF' was used. In figure (fig. 4.15(b)) distance covered in the case of the MBO method was less as compared to traditional PF in most of the scenarios. In figure (fig. 4.15(c)), it can be seen that CC values are different for different scenarios. Only in some scenarios the MBO method gave good results for CC.



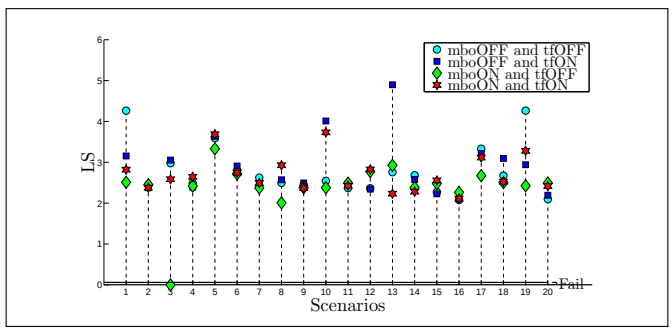
(a) Random scenarios: time comparison



(b) Random scenarios: path length comparison



(c) Random scenarios: CC comparison



(d) Random scenarios: LS comparison

Figure 4.15: Movement of 3 robots in random scenarios with obstacles: Comparison results of different options of algorithm

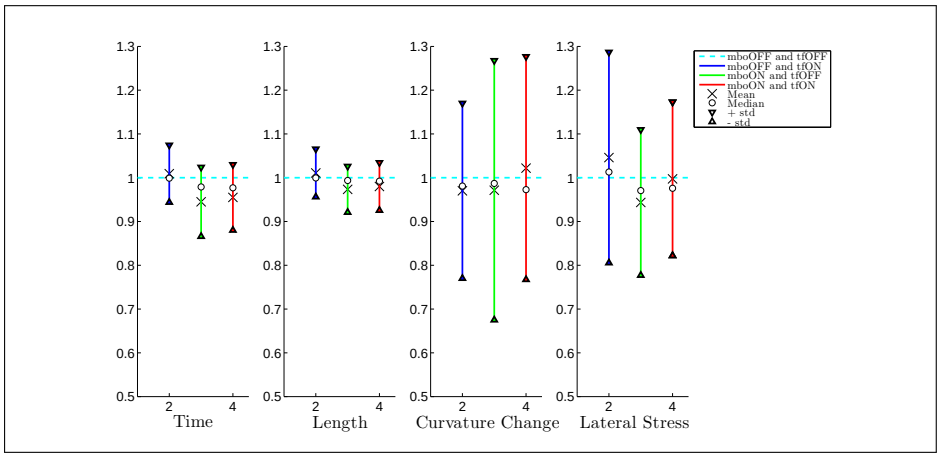


Figure 4.16: Comparison: Median, Mean and standard deviation 20 random scenarios after normalizing cases w.r.t. non-optimized case

The result of LS for random scenarios in figure (fig. 4.15(d)) shows that in most of the scenarios the MBO method has good results compared to traditional PF.

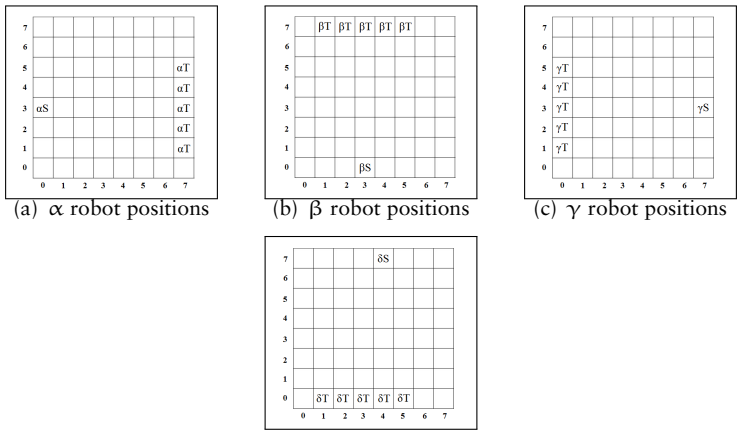
Along with good results, MBO method was giving bad results in some of the scenarios. It can be seen from figures (fig. 4.15) that the MBO method was not good in scenario 15 as time, length, CC and LS parameters are showing unwanted results for option 'mbo-ON and tf-ON'. Similarly in scenarios 9, 12 and 20 all methods took same time to accomplish a task. The reason behind such results is that the navigation method can behave differently on different environments, and there is a chance that one algorithm working efficiently in one scenario can behave differently by changing the environment. So comparing two methods, there is a possibility that in a scenario the first method can have advantage on the second method and in any other scenario the second method can have advantage on the first one.

In figure (fig. 4.16) combined results show that MBO navigation method is little better compared to traditional PF. We got small standard deviation values for time and length parameters but large standard deviation for CC and LS parameters.

Movement of 4 Robots in random scenarios

Without obstacles In this experiment 4 robots were involved in testing of the MBO navigation strategy. 'α', 'β', 'γ' and 'δ' are the four robots. The starting position was fixed for these robots but the target positions were selected randomly, possible positions of all robots can be seen in figure (fig. 4.17).

Results of 20 random scenarios in figure (fig. 4.18) shows what are the effects of different options of algorithm on each performance metric. 3 failures



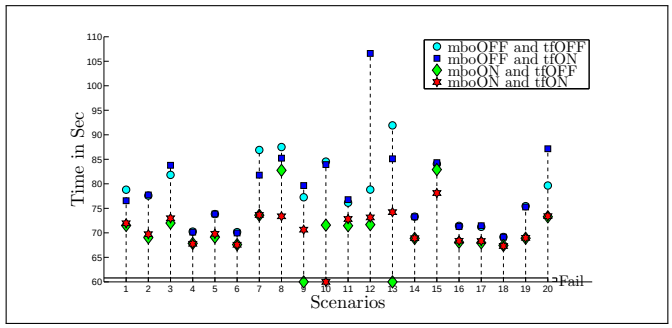
(a) α robot positions (b) β robot positions (c) γ robot positions

(d) δ robot positions

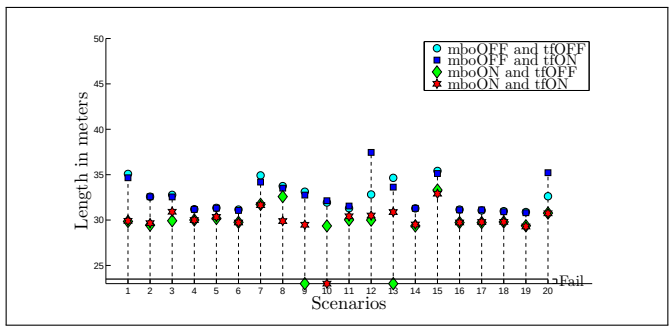
Figure 4.17: Placement of 4 robots in random scenarios without obstacles: possible grid map positions

(one/more robots were not able to reach their target positions) occurred when the MBO method was used. In scenarios 9 and 13, option 'mbo-ON and tf-OFF' was responsible for failures and in scenario 10 option 'mbo-ON and tf-ON' was not able to accomplish the task. But for all other scenarios it can be seen that the MBO method showed reasonable results. In figure (fig. 4.18(a)) the time taken by the MBO method is less as compared to traditional PF in all random scenarios. The distance covered for the MBO method was also less and can be analyzed in figure (fig. 4.18(b)). From figures (fig. 4.18(c)) and (fig. 4.18(d)), values of CC and LS can be compared between the MBO method and the traditional PF.

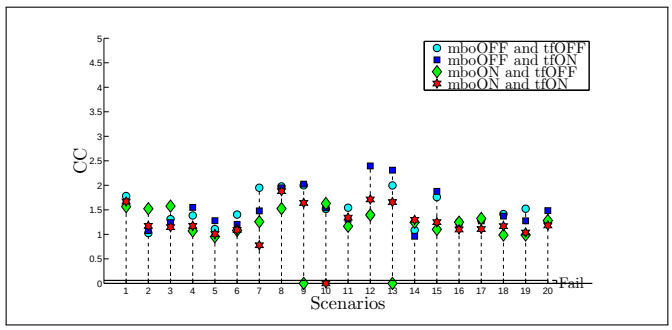
In figure (fig. 4.19) the combined results of 20 random scenarios can be seen, where the MBO method has quite better results compared to traditional PF. Time and length parameters show small standard deviation against each option. But we got large standard deviation for CC and LS parameters.



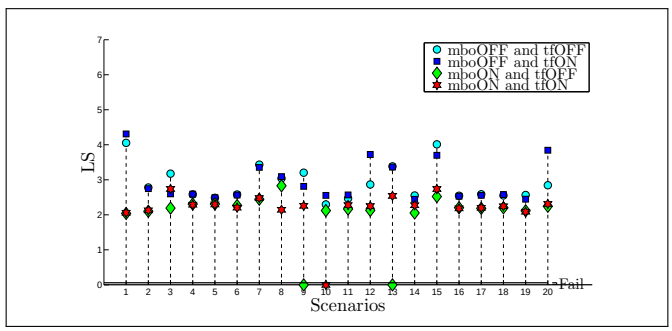
(a) Random scenarios: time comparison



(b) Random scenarios: path length comparison



(c) Random scenarios: CC comparison



(d) Random scenarios: LS comparison

Figure 4.18: Movement of 4 robots in random scenarios without obstacles: Comparison results of different options of algorithm

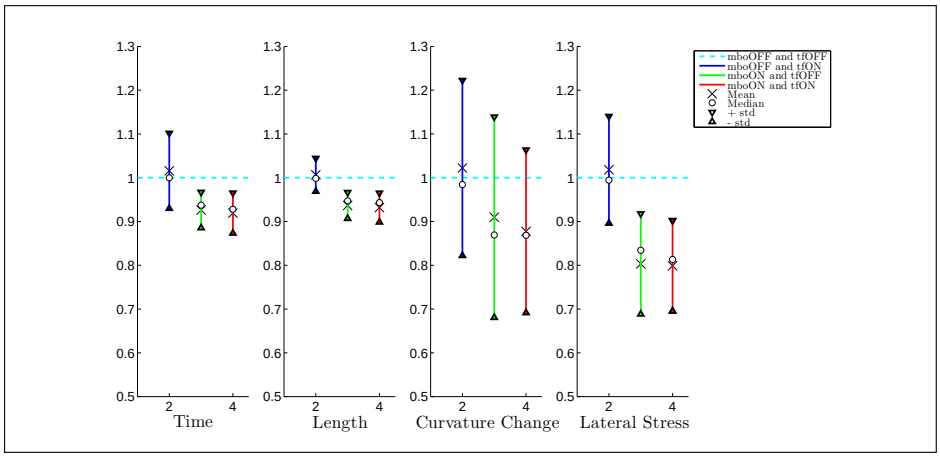
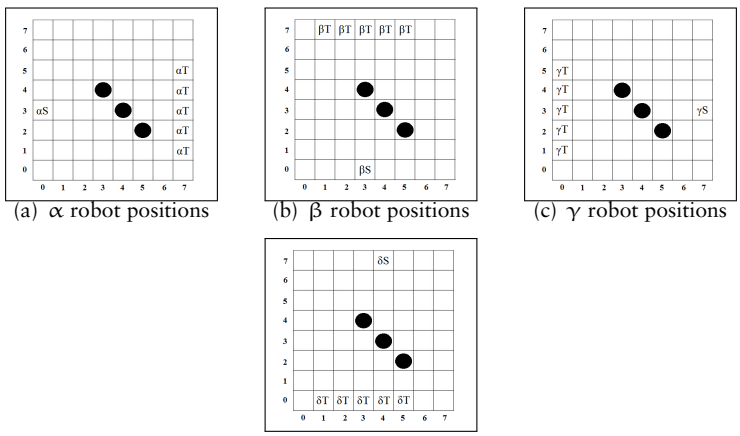


Figure 4.19: Comparison: Median, Mean and standard deviation 20 random scenarios after normalizing cases w.r.t. non-optimized case

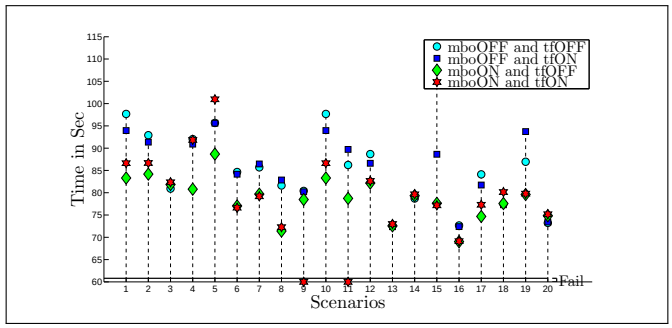
With obstacles In this experiment, the algorithm was tested with 4 robots moving in a shared area when static obstacles were placed. 20 random scenarios were run with different options (from table 4.1)) of the algorithm. Starting positions for all robots were fixed but target positions were selected randomly. Possible positions of robots and static obstacles can be seen in figure (fig. 4.20).



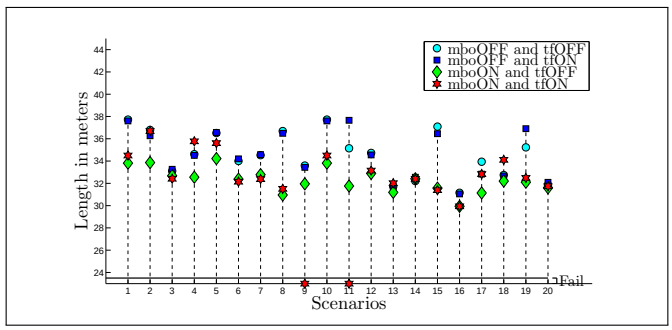
(a) α robot positions (b) β robot positions (c) γ robot positions

(d) δ robot positions

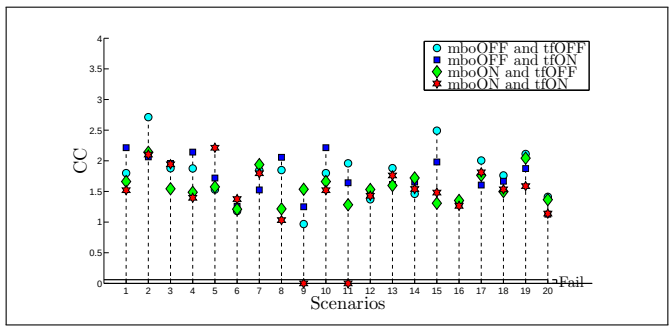
Figure 4.20: Placement of 4 robots in random scenarios with obstacles: possible grid map positions



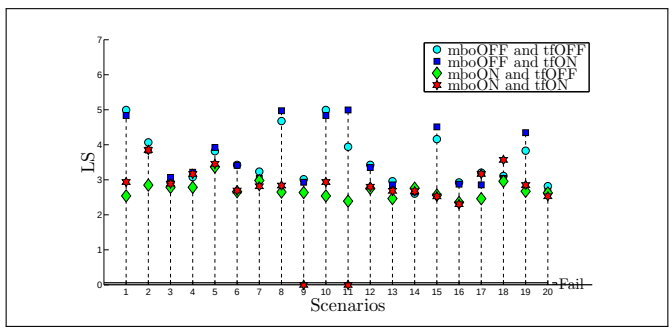
(a) Random scenarios: time comparison



(b) Random scenarios: path length comparison



(c) Random scenarios: CC comparison



(d) Random scenarios: LS comparison

Figure 4.21: Movement of 4 robots in random scenarios with obstacles: Comparison results of different options of algorithm

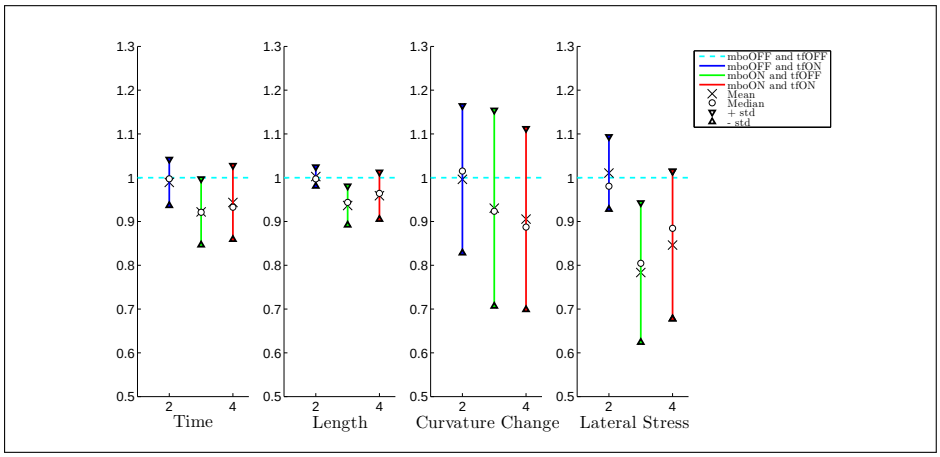


Figure 4.22: Comparison: Median, Mean and standard deviation 20 random scenarios after normalizing cases w.r.t. non-optimized case

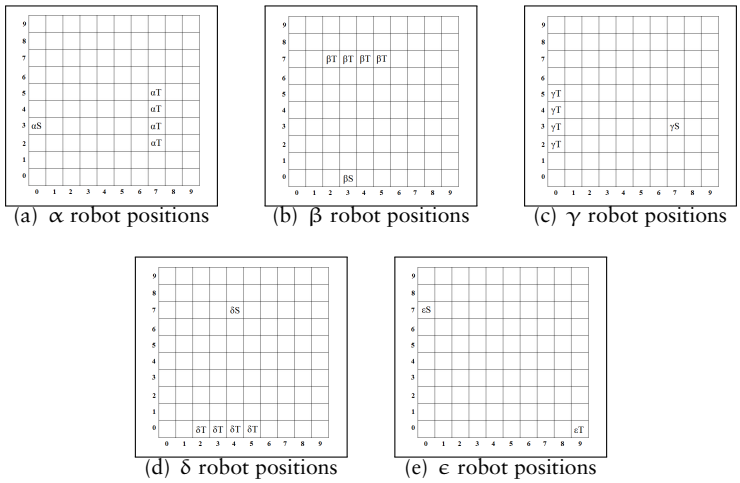
The results of 20 random scenarios in Figure (fig. 4.21) show the effects of different options of algorithm on each performance metric. 2 failures occurred in option 'mbo-ON and tf-ON' in scenario 9 and 11. In most of the scenarios, the MBO method was giving better results as compared to traditional PF. In figures (fig. 4.21(a)) and (fig. 4.21(b)) it can be seen that in most scenarios the time taken and the distance covered were less when the MBO method was used. Results for CC and LS in figures (fig. 4.21(c)) and (fig. 4.21(d)) show that in most of the scenarios the MBO method helped for smooth motion of robots.

Combined results of 20 random scenarios in figure (fig. 4.22) show that results for MBO method are better compared to traditional PF. Again time and length parameters show small standard deviation while CC and LS parameters show large standard deviation values.

Movement of 5 Robots in random scenarios

Without obstacles In this test, 5 robots were used for the evaluation of algorithm. 'α', 'β', 'γ', 'δ' and 'ϵ' are the five robots. Start and end positions for 'ϵ' robot were fixed, but target positions of other robots were selected randomly. Possible positions of all robots can be seen in figure (fig. 4.23).

Figure (4.24) is showing results for time, length, CC and LS for 20 random scenarios. Total 4 failures occurred when the MBO method was used. 3 failures occurred when option 'mbo-ON and tf-OFF' was used in scenarios 8, 9 and 14. In scenario 7, option 'mbo-ON and tf-ON' was responsible for failure. In all other scenarios the MBO method showed reasonable results compared to traditional PF. In figures (fig. 4.24(a)) and (fig. 4.24(b)) it can be seen that the MBO method took less time and covered less distance as compared to tradi-



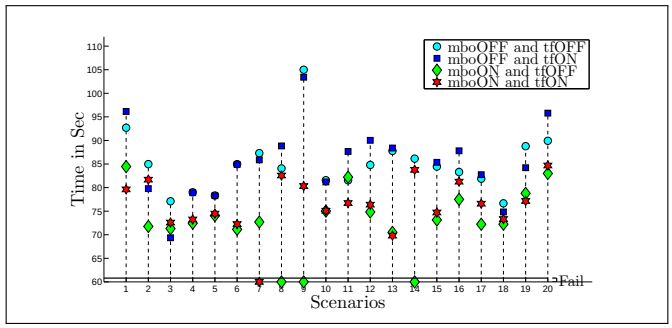
(a) α robot positions (b) β robot positions (c) γ robot positions

(d) δ robot positions (e) ϵ robot positions

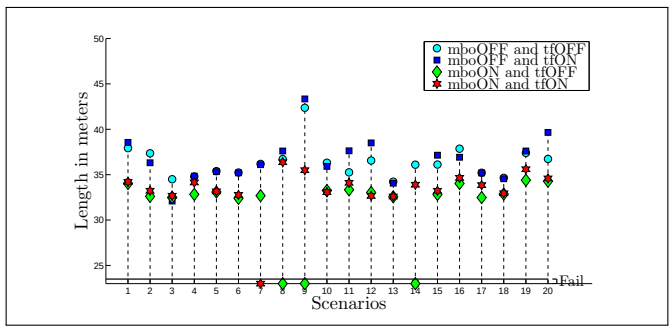
Figure 4.23: Placement of 5 robots in random scenarios without obstacles: possible grid map positions

tional PF. But in figure (fig. 4.24(c)), the result for CC were different, because in some scenarios the MBO method was good and in some scenarios traditional PF showed reasonable results. From LS results in figure (fig. 4.24(d)) it can be analyzed that the MBO method was reasonable compared to traditional PF.

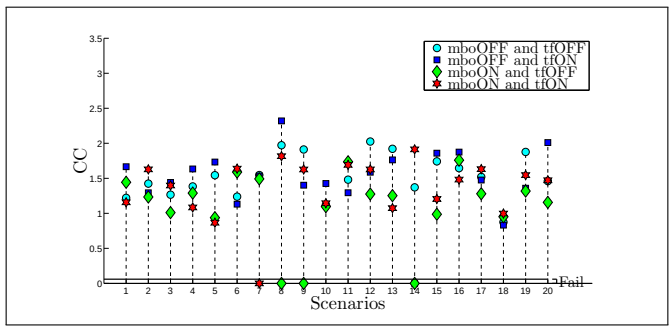
In figure (fig. 4.25) it can be analyzed that results for time, length and LS parameters are better in case of using the MBO method. We got large standard deviation for CC parameter against each option.



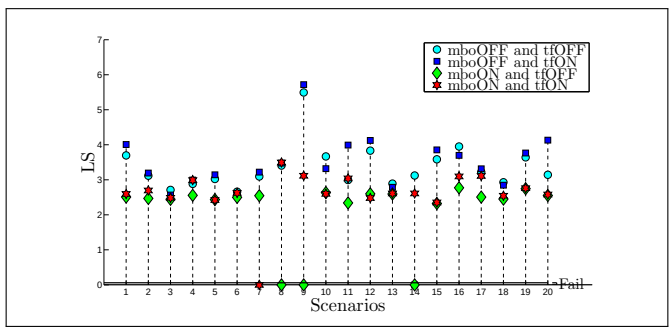
(a) Random scenarios: time comparison



(b) Random scenarios: path length comparison



(c) Random scenarios: CC comparison



(d) Random scenarios: LS comparison

Figure 4.24: Movement of 5 robots in random scenarios without obstacles: Comparison results of different options of algorithm

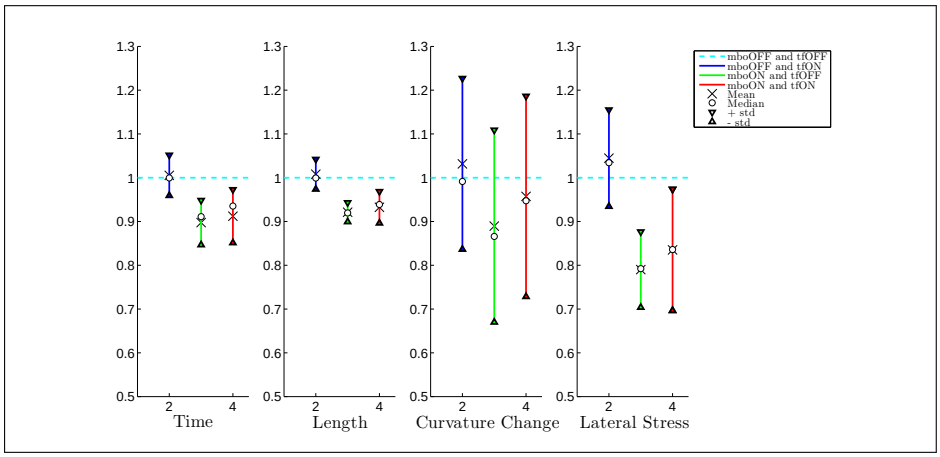


Figure 4.25: Comparison: Median, Mean and standard deviation 20 random scenarios after normalizing cases w.r.t. non-optimized case

With obstacles In this experiment, the algorithm was tested with 5 robots in a shared area where static obstacles were placed. 20 random scenarios were run with different options (from table 4.1) of algorithm. Possible positions for all robots and static obstacles can be seen in figure (fig. 4.26).

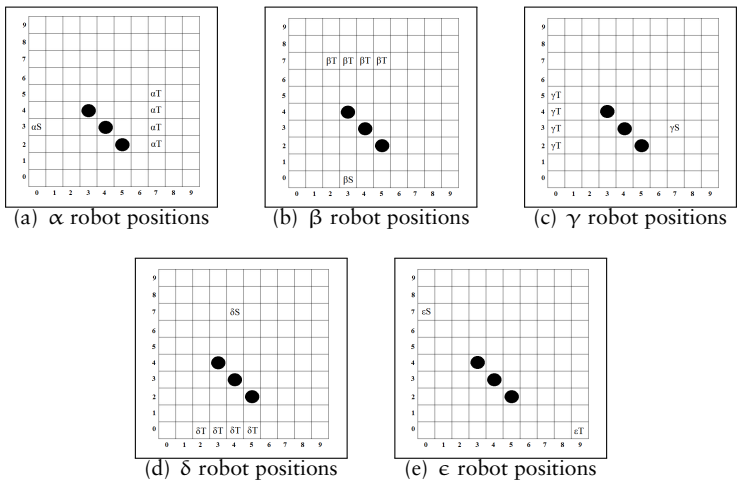
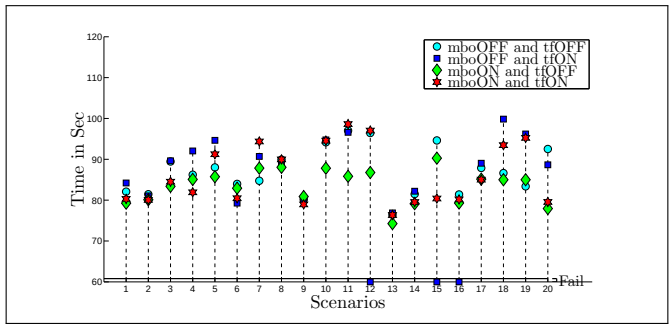
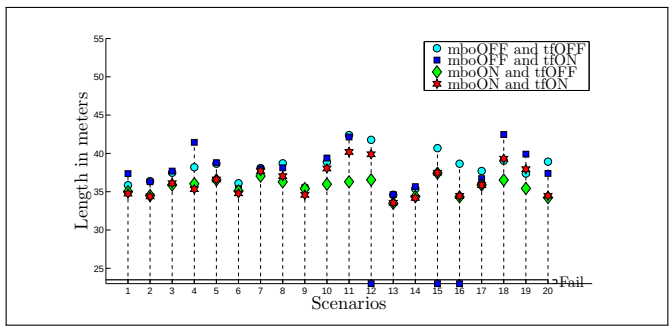


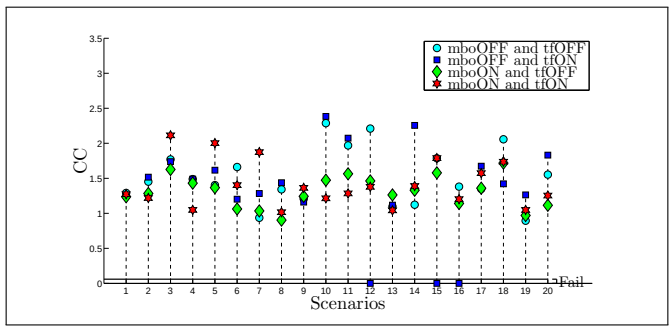
Figure 4.26: Placement of 5 robots in random scenarios with obstacles: possible grid map positions



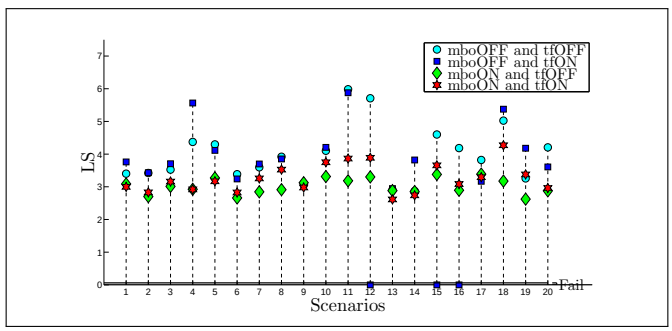
(a) Random scenarios: time comparison



(b) Random scenarios: path length comparison



(c) Random scenarios: CC comparison



(d) Random scenarios: LS comparison

Figure 4.27: Movement of 5 robots in random scenarios with obstacles: Comparison results of different options of algorithm

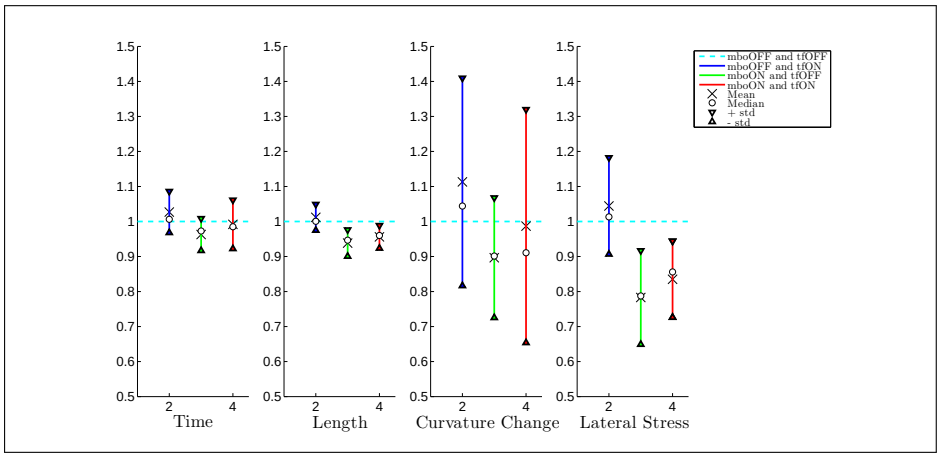


Figure 4.28: Comparison: Median, Mean and standard deviation 20 random scenarios after normalizing cases w.r.t. non-optimized case

Results of 20 random scenarios in figure (fig. 4.27) show what the effects of different options of algorithm on each performance metric are. Total 3 failures occurred in scenarios 12, 15 and 16 when option 'mbo-OFF and tf-ON' was used. In figure (fig. 4.27(a)) results show that in some cases the MBO method took less time compared to traditional PF, and in some cases the results for traditional PF were reasonable. But results in figure (fig. 4.27(b)) show that the MBO method covered less distance in most scenarios compared to traditional PF. CC results in figure (fig. 4.27(c)) are showing different results, as in some scenarios MBO method showed good results and in some scenarios traditional PF showed reasonable results. In figure (fig. 4.27(d)) LS results show that the MBO method have reasonable results compared to traditional PF.

In figure (fig. 4.28) combined results of 20 random scenarios show that MBO method is little better compared to traditional PF. Time, length and LS parameters show better results against MBO method. We got large standard deviation for CC against each option.

It is analyzed in the combined results of each experiment that MBO method has more effect on time and length parameters as compared to CC and LS. Another result is analyzed that on increasing number of robots, MBO method performs well compared to traditional PF.

4.5.4 Limitations of MBO navigation strategy in some special cases

There are some limitations when the proposed method was used for multi-robot navigation. Two cases are discussed here.

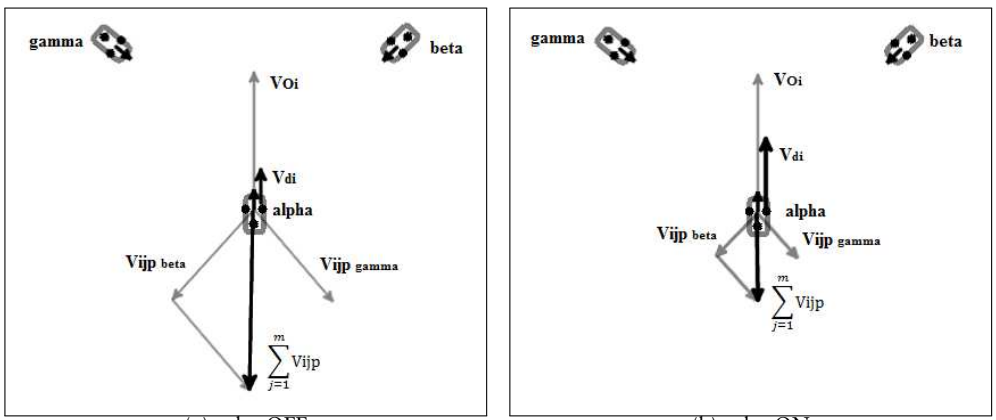
Case 1

MBO navigation strategy reduces the repulsive potential field of other robots based on their importance. But it can also cause problems in some of the cases. One example is discussed here.

We can easily understand the MBO limitation when we have such scenarios shown in figure (fig. 4.29). In this scenario α, β and γ robots are heading towards their goals and are at same distance from each other. Here we will only talk about α robot and how potential fields affect it in case of mbo-ON and mbo-OFF. β and γ robots are applying equal repulsive forces on α robot. Let's talk about what will happen in case of mbo-OFF and mbo-ON? In figure (fig. 4.29) we have

$$\begin{aligned} V_{ij_{p(\beta)}} &= w_{ij} v_{ij_{p(\beta)}} \\ V_{ij_{p(\gamma)}} &= w_{ij} v_{ij_{p(\gamma)}} \\ \sum_{j=1}^{m} V_{ij_p} &= V_{ij_{p(\beta)}} + V_{ij_{p(\gamma)}} \end{aligned} \tag{4.6}$$

mboOFF: In this case, α will assign $w = 1$ to each robot and we will get a vector $\sum_{j=1}^{m} V_{ij_p}$ as shown in figure (fig. 4.29(a)). This vector is a bigger vector and effects a lot and tries to stop the α robot heading towards other robots. Here v_{d_i} is the desired velocity vector.



(a) mbo-OFF (b) mbo-ON

Figure 4.29: Limitation in symmetric situations: (a) In case of mbo-OFF, small Vdi will slow down the speed from a distance (b) In case of mbo-ON, bigger value of Vdi will take robots near to each other and can cause collision

mboON: In this case α will assign $w = 0.5$, to each robot as robots are at same distance (Symmetry). We will get a vector $\sum_{j=1}^{m} V_{ij_p}$ as shown in figure (fig. 4.29(b)). This vector is small compared to the vector computed in the case of mbo-OFF. Here v_{d_i} is bigger as compared to v_{d_i} computed in the case of

mbo-OFF and tries to take robots closer to each other. This can cause collision or result in longer time spent in trying to avoid each other.

Case 2

Another limitation of the MBO navigation strategy is that, sometimes robots are not giving importance to other robots while navigating. This can bring robots so near that they can collide. In our implementation 'emergency stop' traffic rule is implemented so robots stop when these cases occur. One example is given here to understand this limitation.

Consider four robots moving in a small area, these robots are 'α', 'β', 'γ' and 'δ'. While moving they make a configuration as shown in figure (fig. 4.30(a)). In figure (fig. 4.30(b)), 'α' robot detects other robots in its local frame and target position of α robot is also shown here. Weights were calculated by 'α' robot in this case.

Weights calculated by 'α' robot:

$w_0 = 0.6243$ γ robot
$w_1 = 0.2394$ δ robot
$w_2 = 0.1362$ β robot

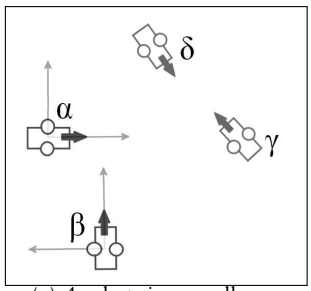
In figure (fig. 4.30(c)), 'β' robot detects other robots in its local frame and target position of beta robot is shown. In this case 'β' robot calculated weights given as follows

Weights calculated by 'β' robot:

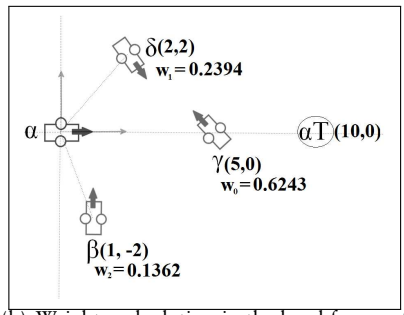
$w_0 = 0.5155$ δ robot
$w_1 = 0.2903$ α robot
$w_2 = 0.1940$ γ robot

It can be seen from the weights calculated by 'α' and 'β' that 'α' robot is giving less importance to 'β' robot. The same for 'β' robot as it is giving less importance to 'α' robot. This can lead robots in such a situation that they can collide. That was the common problem in some of the cases when MBO navigation strategy was used and failures occurred.

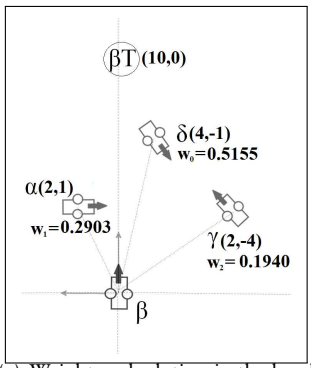
It is observed during experiments that the MBO navigation strategy assigns weights depending on the positions of other robots and does not consider the orientation of other robots. So robots using the MBO strategy are not able to assign higher weights to robots coming towards them which can result in unsafe situations.



(a) 4 robots in a small area



(b) Weights calculation in the local frame of α robot



(c) Weights calculation in the local frame of β robot

Figure 4.30: Case2: Limitation in some cases when robot using MBO assigns less weights to other robots coming towards it

4.6 Results Summary

The proposed algorithm is a combination of multiple methods. Potential fields are used for safe movement of robots from starting position to target position. Fuzzy rules are used to reduce the effect of repulsive potential fields in the vicinity of obstacles. Traffic rules are used to handle the situations where robots are crossing each other. The MBO navigation strategy is used to optimize repulsive potential fields of other robots. After using all these methods a final velocity vector is computed and provided to the robot to move.

It has been observed that the navigation of mobile robots can be effectively implemented using a combination of these methods. In this chapter first fuzzy rules were tested for obstacle avoidance and it was observed that fuzzy rules can reduce unnecessary potential fields in the effective region of obstacles. In later experiments the validation of the MBO method was tested and results were compared with traditional PF. It has been observed that while using traditional PF method, repulsive potential fields were effecting and causing unwanted motions of robots, but when we activate the MBO option in those experiments, we obtained smoother results. From experiments, it has been examined that robots using the MBO method take less time and cover less distance, while values of LS parameters in different scenarios showed that MBO method helps in safe turnings of robots while moving. During experiments it has been observed that the MBO method has more effect on time and length parameters, while CC and LS are much related with robot control. In the results of random scenarios it is observed that MBO method showed better results in most of the scenarios but in some scenarios MBO navigation method caused failures. These failures occurred due to the limitations of MBO method. Another point is that of an increased number of robots effect performance of MBO strategy. It becomes better as compared to traditional PF but due to limitations of MBO method, still some risk of collision exists.

As the proposed algorithm is a combination of different methods, the final movement of robots can be effected by all these methods. Since every method has some limitations e.g. potential fields have a local minima problem, traffic rules try to add some additional vectors, etc which can cause unwanted movements of the robots.

During experiments it is observed that the MBO method has two limitations while assigning weights to other robots. One limitation is due to the normalization of weights to 1, it can cause problems if we increase the number of robots in a small area as it will assign very little weights to some of the robots. The second limitation of the MBO method is that while computing the weights it only considers the positions and does not care about orientations of other robots.

Chapter 5
Conclusion

5.1 Summary

In this thesis we focused on mobile robots which share their workspace with other robots. Considering the problem of navigation for teams of mobile robots, a mixture of different methods is used for safer and effective navigation. The idea was that multiple robots will be moving in a small area and every robot will use the same algorithm for navigation. The implemented navigation strategy uses the potential field algorithm for navigation and some additional layers are implemented to optimize repulsive potential fields. Fuzzy rules are used to optimize repulsive potential fields generated by obstacles. Traffic rules are used to handle the situations where robots are crossing each other and the MBO method is used to optimize the repulsive potential fields generated by other moving robots. MB optimization is very similar to economic systems where consumers and producers deal with commodities on the basis of a common price. So by using MB optimization, repulsive potential fields generated by other vehicles are strengthened or weakened depending on their importance. As a result every robot computes a final velocity vector that helps to achieve a smooth motion and reduce cost of movement of each robot in a small area.

The main aim of this thesis was to implement and test extensively the MBO navigation strategy on more realistic simulation environment. In this thesis, the 'mboNavigation' stack in ROS is implemented which can be used to control a real robot. But before using this method with robots in real environments some extensive experiments are performed in the gazebo simulator for evaluation purposes. For evaluation of the MBO navigation strategy an experimental setup was implemented where random scenarios were generated and in each scenario robots were run with different options of the navigation algorithm. During these experiments data (linear and angular velocities, position in map) of every robot was logged in and was processed in Matlab at a later time to generate results.

During experiments it was observed that the MBO navigation strategy is effective in most cases and the robots took less time and covered less distance to reach their goal positions as compared to traditional PF. Smooth movement of robots was evaluated using CC (curvature change) and LS (lateral stress) parameters. It was observed that the results for CC were more or less the same for the MBO navigation strategy and traditional PF. Results against LS parameter showed that the robots had safe movements while turning when the MBO strategy was used. Another finding was that the proposed method has more effect on time and length parameters while CC and LS parameters are more related to the control of the robots. From the results of random scenarios it is observed that the average evaluation parameters are better with MBO navigation method but there is also a risk with current implementation due to limitation of this navigation method. The results showed that with higher number of robots in a small area the MBO strategy gives better results as compared to traditional PF but a risk of collision also increases due to the limitations of the MBO strategy. To avoid this collision risk, a safe stopping rule is introduced.

Despite these encouraging results of MBO method, some limitations exist for improvements. For example one limitation of assigning weights is that weights are normalized to 1 and in case of many robots assigning normalized weights can decrease the influence of these robots. This issue of assigning small weights can be solved by changing the normalization of weights. Another limitation is that the MBO method assigns weights considering the positions of other robots and does not care about the orientation of other robots. So a robot using the MBO method can cause assigning less weight to robots that are not in the way to the target but are coming towards it. This can lead robots in such a situation that these can come very close to each other and can block each other's way. This issue can be overcome by considering the orientations of other robots in MBO method.

The MBO navigation method has been implemented in more realistic simulation environment and has been extensively tested in this thesis. The navigation method deals with the complicated configuration of team of mobile robots moving in a shared area. The navigation method has been successful in navigating multiple mobile robots in a shared environment. This method has provided quite reasonable results as compared to traditional PF. Some limitations of this navigation method has been discussed that can be overcome by introducing some methods.

5.2 Future work

'mboNavigation' stack implemented in this thesis work is at its initial phase, where it is a big assumption that every robot can localize itself very accurately and knows positions and orientations of other robots. Based on this information every robot calculates final velocity vector and moves. In future, some

packages can be added where a robot can differentiate between static and dynamic obstacles using laser range finder and can localize itself in the map.

Here the final velocity vector calculated in case of MBO strategy is used only to move a differential drive robot. In future, the same 'mboNavigation' stack can be used with omni-directional robots but motion control will be different. The MBO navigation strategy can be tested on omni-directional robots with little changes, but the parameters for evaluation will be time and length. CC and LS will not be considered as these parameters are associated with car like robots. It's a consideration that omni-directional robots can provide good results as compared to non-holonomic vehicles against MBO navigation strategy and limitations of MBO can be easily handled.

Another possible future work is to develop methods to deal with small weight issues of the MBO approach. Moreover, it is an open question, how the MBO approach can be used to take into account the velocity of perceived vehicles and not just their positions.

References

[1] Mamdani's fuzzy inference method. `http://www.dma.fi.upm.es/java/fuzzy/fuzzyinf/` (Cited on page 15.)

[2] Michael A. Goodrich. Pf tutorial. http://students.cs.byu.edu/ cs470ta/goodrich/fall2004/lectures/Pfields.pdf. (Cited on page 10.)

[3] Ronald C. Arkin and Robin R. Murphy. Autonomous navigation in a manufacturing environment. *Robotics and Automation, IEEE*, 6(4):445 – 454, 1990. (Cited on page 1.)

[4] Valluru B. Rao. *C++ Neural Networks and Fuzzy Logic*. MIS-press, 1995. (Cited on page 13.)

[5] Daniele Calisi and Daniele Nardi. Performance evaluation of pure-motion tasks for mobile robots with respect to world models. *Autonomous Robots*, 27(4):465 – 481, 2009. (Cited on pages 3, 43, and 44.)

[6] Chih-Fu Chang, Ching-Chih Tsai, Jui-Cheng Hsu, Shoei-Chuen Lin, and Chu-Chen Lin. Laser self localization for a mobile robot using retro-reflector landmarks. In *Still to Find*, Find. (Cited on page 1.)

[7] Liu Chengqing, Marcelo Hang Jr, Hariharan Krishna, and Lim Ser Yong. Virtual obstacle concept for local minimum recovery in potential field based navigation. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference*, 2000. (Cited on page 9.)

[8] Gerald cook. *Mobile robots: Navigation, control and remote sensing (Ch.1)*. Wiley-IEEE Press, 2011. (Cited on pages 5, 7, and 39.)

[9] M. Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survay and analysis. *Proceedings of the IEEE*, 94(7):1257 – 1270, 2006. (Cited on pages 8, 17, and 18.)

[10] Willow Garage. Ros. `http://ros.org/wiki/ROS/Introduction`. (Cited on page 41.)

[11] Gazebo. Gazebo user guide. `http://gazebosim.org/user_guide/overview__intro.htm` (Cited on pages 2 and 42.)

[12] H. Robert Heller. *The economic system (Ch.16)*. Macmillan, 1972. (Cited on page 17.)

[13] Timothy J. Ross. *Fuzzy Logic with engineering applications*. John Wiley and Sons, 2004. (Cited on page 14.)

[14] Shin kato, Sakae Nishiyama, and Jun'ichi Takeno. Coordinating mobile robots by applying traffic rules. In *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference*, 1992. (Cited on pages 9 and 27.)

[15] O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference*, 1985. (Cited on pages 2 and 9.)

[16] Y. koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference*, 1991. (Cited on pages 2 and 13.)

[17] Qing Li, Lijun Wang, Bo Chen, and Zhou Zhou. An improved artificial potential field method for solving local minimum problem. In *Intelligent Control and Information Processing (ICICIP), 2011 2nd International Conference*, 2011. (Cited on pages 9, 35, and 36.)

[18] V.J. Lumelsky and K.R. Harinarayan. Decentralized motion planning for multiple mobile robots: The cocktail party model. In *Autonomous Robots*, 1997. (Cited on page 9.)

[19] Angelo Martinez, Eddie Tunstel, and Mo. Jamshidi. Fuzzy logic based collision avoidance for a mobile robot. In *Robotica*, 1994. (Cited on pages 2 and 13.)

[20] R. R. Murphy. *Introduction to AI robotics*. The MIT press, 2000. (Cited on pages 9, 10, 11, and 13.)

[21] Rainer Palm and Abdelbaki Bouguerra. Market-based optimization for the navigation of mobile robots. In *Proceedings of the 5th European Conference on Mobile Robots ECMR 2011, Orebro*, 2011. (Cited on pages 2, 11, 13, 17, 18, 19, 21, 25, 27, 30, 31, and 32.)

[22] Keving M. Passino and Stephen Yurkovich. *Fuzzy Control*. Addison-Wesley, 1998. (Cited on pages 15 and 16.)

[23] D. Herrero Perez and H. Matinez Barbera. Decentralized coordination of autonomous agvs in flexible manufacturing systems. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference*, 2008. (Cited on pages 1, 2, and 8.)

[24] Player. The player project. `http://playerstage.sourceforge.net/`. (Cited on page 42.)

[25] Ronald Siegwart and Illah R. Nourbakhsh. *Introduction to autonomous mobile robots*. The MIT press, 2004. (Cited on pages 1, 5, and 6.)

[26] Marcelo Godoy Simoes. Introduction to fuzzy control. `http://inside.mines.edu/~msimoes/documents/Intro_Fuzzy_Logic.pdf`. (Cited on pages 14, 15, and 16.)

[27] Petru Emanuel Stingu and Frank L. Lewis. Motion path planning for mobile robots. `arri.uta.edu/acs/ee5322/lectures/Motion%20path%20planning.pdf`. (Cited on pages 2 and 13.)

[28] Gilbert Strang. *Inreoduction to linear algebra*. Wellesley-cambridge press, 2009. (Cited on page 10.)

[29] Holger Voos and Lothar Litz. A new approach for optimal control using market based algorithms. In *Proceedings of the European Control Conference ECC99, Karlsruhe*, 1999. (Cited on pages 17, 18, and 30.)

[30] Charles W. Warren. Multiple robot path coordination using artificial potential fields. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference*, 1990. (Cited on page 8.)

[31] Hao Ying. *Fuzzy Control and Modeling Analytical Foundations*. IEEE press, 2000. (Cited on pages 14 and 15.)

[32] Xiaoping Yun and KO-Cheng Tan. A wall-following method for escaping local minima in potential field based motion planning. In *Advanced Robotics, 1997. ICAR '97. Proceedings., 8th International Conference*, 1997. (Cited on page 9.)

[33] L. A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338 – 353, 1965. (Cited on page 14.)

Appendix A
mboNavigation stack in ROS

A navigation stack is implemented in ROS (Robot operating system). Nodes and topics are discussed in detail in this appendix.

A.1 Used ROS packages

In this section ROS packages and messages are discussed that are used while implementing 'mboNavigation' stack.

sensor_msgs/LaserScan Message

It's a laser range finder sensor message that stores detected ranges at different angles. Sensor's settings (min/max angle, angle increment, max/min range etc) can also be achieved from this message.

nav_msgs/Odometry Message

It's a navigation message contains information of position (x, y, theta) and velocity (linear, angular) of robot.

Visualization_msgs/Marker Message

This ROS message is used by high level packages like rviz visualizer. Different markers (Arrow, cube, sphere etc) can be visualized by setting values for this message and providing it to rviz. In our implementation Marker::Arrow is used to visualize velocity vectors generated by each robot.

tf

Tf is a package that helps user to keep track of different coordinate frames. Relation between coordinate frames can be easily understood and points can be easily transformed from one coordinate frame to another.

Fake_localization

From the name of this package it can be understood that it helps to provide fake localization of robot. This package is used in simulations to provide perfect localization. In our work, we are considering that every robot can accurately localize itself; fake_localization package is used for this purpose.

Simulator_gazebo

Using this stack ROS user can work and test their robots in gazebo (3D simulator). In our implementation gazebo simulator is used to test navigation strategy on multiple mobile robots.

Erratic_robot

Erratic robot is a differential drive robot; complete stack for this robot is available in ROS. An `erratic_gazebo_plugin` is also available. In our work navigation strategy is tested on multiple erratic robots in gazebo simulator.

rviz

In ROS rviz is a 3D visualizer. Different ROS messages can be visualized in rviz; for example odometry data can be visualized, coordinate frames and their relations with other frames can be visualized, new markers can be generated against new messages and can be visualized. In our work rviz is used to visualize trajectories and velocity vectors generated by each mobile robot.

A.2 mboNavigation stack implementation in ROS

A 2D navigation stack is implemented that produces velocity commands using potential field algorithm, fuzzy rules, traffic rules and market based optimization. Differential drive robots can use this stack for safe multi-robot navigation.

Following packages are included in this stack.

tf_map

A package '`tf_map`' is implemented that uses '`tf`' package to find position of every robot in '`map`' frame. These positions are then published using '`PositionofRobots`' message on topic '`/robotsPositionsInmap`'.

obstacle_detection

In this package a node '`robotName_obstacleInfo`' is implemented to find obstacle points and positions of other robots in local frame of robot. This node

subscribes '/robotsPositionsInmap' and 'robotName/base_scan/scan' topics and differentiate between obstacle points and other robot points. After separating other robots from obstacles this node publishes topics '/robotName/obstaclePoints' and '/robotName/robotobstaclePoints'.

potentialFields

A node 'robotName_goto' is implemented in this package. MBO navigation strategy is implemented in this node. Target position is provided as an input. This node subscribes four topics. '/robotName/obstaclePoints' and '/robotName/robotobstaclePoints' topics provide information of obstacles and other robots and this information is used to generate repulsive potential field. '/robotName/base_pose_ground_truth' topic provides perfect linear and angular velocities of robot (this information is logged in for evaluation purpose). '/robotName/amcl_pose' topic is subscribed that provides perfect position of robot in map. Based on all this information 'robotName_goto' node produces velocity commands (linear and angular velocities) and publishes a topic '/robotName/cmd_vel' for movement of robot.

'robotName' is set by user.