Design and development of Intelligent Mobile Robots (IMRs) for disaster mitigation and firefighting

First Deliverable

Report submitted on 14th July 2016

PI: Dr. Muhammad Bilal Kadri

Co-PI: Dr. Tariq MairajRasool Khan

Table of Contents

Chapter 1. Introduction

This report is the first deliverable for the National ICTR&D funded research project titled "Design and development of Intelligent Mobile Robots (IMRs) for disaster mitigation and firefighting".

Chapter 2. Literature Review and basic algorithm design

2.1 Introduction

This chapter contains the modeling, control strategies and simulation results of the following four modules of a mobile robot.

a) Control Strategy
b) Networked Control and Mobile Sensor Network
c) Image Processing Module and SLAM
d) Path Planning/Obstacle Avoidance

2.2 Modeling of Mobile Robots

Developing an accurate model of the underlying process is the core requirement for any control system to work perfectly. Many models for wheeled robots have been proposed in the literature ranging from very simple to extremely complicated models. In the following paragraphs some models for wheeled robots have been discussed.

2.2.1 Differential Drive Mobile Robot

The exact location (x,y) and the orientation of the robot are directly related to the left and right wheel velocities v_L and v_R.

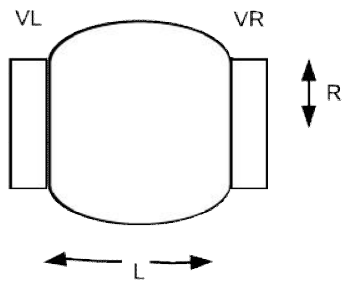


Figure 1 Block diagram of differential drive mobile robot

The state space model of a differential drive mobile robot is given below:

$$\begin{aligned}\dot{x} &= \frac{R}{2}(v_R + v_L)\cos\theta \\ \dot{y} &= \frac{R}{2}(v_R + v_L)\sin\theta \\ \dot{\theta} &= \frac{R}{L}(v_R - v_L)\end{aligned} \tag{1}$$

2.2.2 Unicyle Model

Unicycle model is a more realistic model as compared to the differential drive mobile robot. The position and orientation of the mobile robot are directly related to the linear and angular velocities. Design of controller is much easier unicycle model. For implementing the control scheme on the hardware which requires PWM signal for the left and right wheel motors, a transformation from the unicycle model to the differential drive model is required.

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega\end{aligned} \tag{2}$$

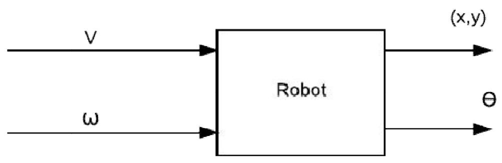


Figure 2 Basic unicycle model for a mobile robot

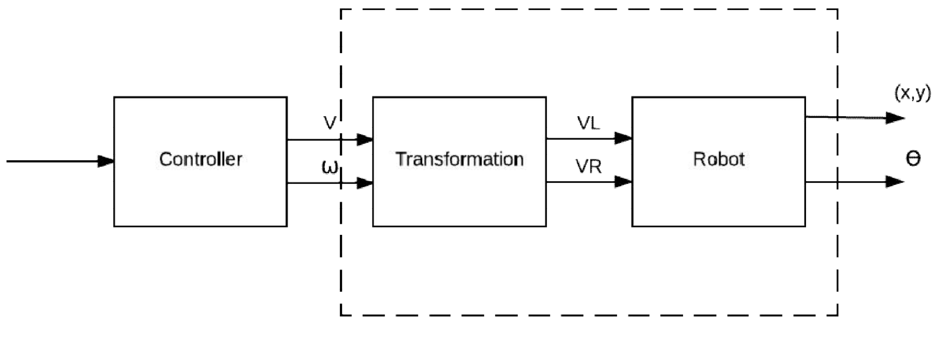


Figure 3 Basic control scheme for a robot

Transformation from linear and angular velocity to right and left wheel velocities

$$v_R = \frac{2v + \omega L}{2R} \tag{3}$$

$$v_L = \frac{2v - \omega L}{2R} \tag{4}$$

2.2.3 Spring-Mass-Damper system

Mobile robot can also be modeled as a simple spring, mass and damper system. All the mobile robots are modeled without any motion constraint. The dynamical equations of the kth mobile robot is given in Equation(5)

$$\begin{aligned} M\ddot{x}_k + B\dot{x}_k &= f_{xk} - k_d\dot{x}_k \\ M\ddot{y}_k + B\dot{y}_k &= f_{yk} - k_d\dot{y}_k \end{aligned} \tag{5}$$

Where ‘M’ is the mass of the robot, B is the damper coefficient, f_{xk}, f_{yk} are the control forces applied on the robot and k_d controls the transient response of the robot.

2.3 Formation Control Strategies

There are three main strategies for the formation control of mobile robots which are listed below:

a) Leader-follower
b) Virtual structures
c) Behavioral structure

The main disadvantage with the leader-follower approach is that there is no feedback from the leader to the follower. If the leader malfunctions or any of the follower(s) fail then there is no guarantee of formation keeping. In the virtual structure all the mobile robots follow a virtual leader and maintain a strong geometric relationship among them.In the behavioral structure, behaviors are defined for each sub-problem i.e. obstacle avoidance, go-to- goal are formation keeping. Behavioral structures are generally modeled as a finite state machine. Although behavioral structures are much closer to how human works, it can destabilize the complete system if transitions between different states are not properly modeled.

2.3.1 Formation Control based on virtual leader

We have implemented a potential based formation control strategy based on virtual structure(Rezaee & Abdollahi, 2014).All the mobile robots irrespective of their current location will start from their initial position and will move towards a circle of radius ‘α’.

$$\begin{aligned}\dot{x} &= -(x-x_c)((x-x_c)^2+(y-y_c)^2-\alpha^2)\\ \dot{y} &= -(y-y_c)((x-x_c)^2+(y-y_c)^2-\alpha^2)\end{aligned} \tag{6}$$

Where x_c and y_c is the coordinate of the virtual robot and ‘x’ and ‘y’ are the coordinates of the kth robot.Each mobile robot acts as a positive charge and exert a force on all the other mobile robots operating within the region. The virtual robot is placed at the centre ofthe formation and acts a negative charge. The virtual robot pulls all the robots closer together until a certain equilibrium point is reached where the attractive force from the virtual robot and repulsive force from other robots cancels each other.The force between any two robots is modeled as Coloumb’s electrostatic force and is given by:

$$F_{ki} = \frac{k_r q_k q_i}{r_{ki}^2} \tag{7}$$

Where k_r is a positive constant depending on the robot configuration and space in which the robots are operating. q_k and q_i are the charges assigned to each robot and ‘r’ is the distance between the robots. The total force exerted by N-1 robots on a single robot ‘k’ is given by:

$$F_k = k_r q_k \sum_{i=1, i\neq k}^{N} \frac{q_i}{r_{ki}^2} \tag{8}$$

Irrespective of the number of participating robots, all the robots will converge on a circular path. Ideally they should form a polygon but due to non linearities they form an irregular polygon with all the vertices i.e. robots lying on the circle.

2.3.2 Simulation results

The charge on each robot is assumed to be 1 Coulomb and the value of the ‘k_r’ is assumed to be 10. Results from three simulations consisting of 3,4 and 5 mobile robots are presented in the following figures. It can be observed from all the figures that irrespective of the staring position of the robot on the grid, all robots converge on the circular path.

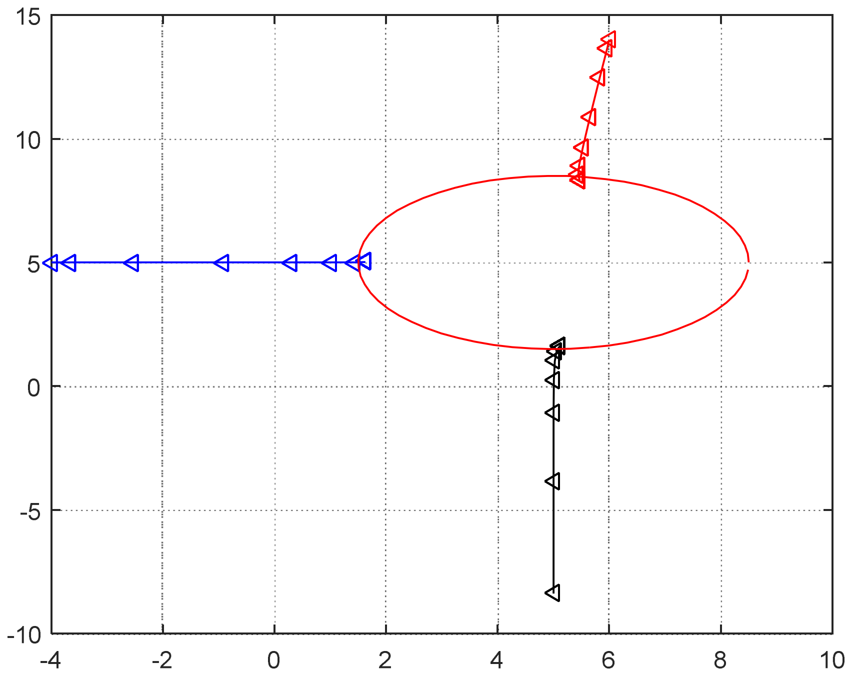


Figure 4 Formation of three robots starting from arbitrary positions

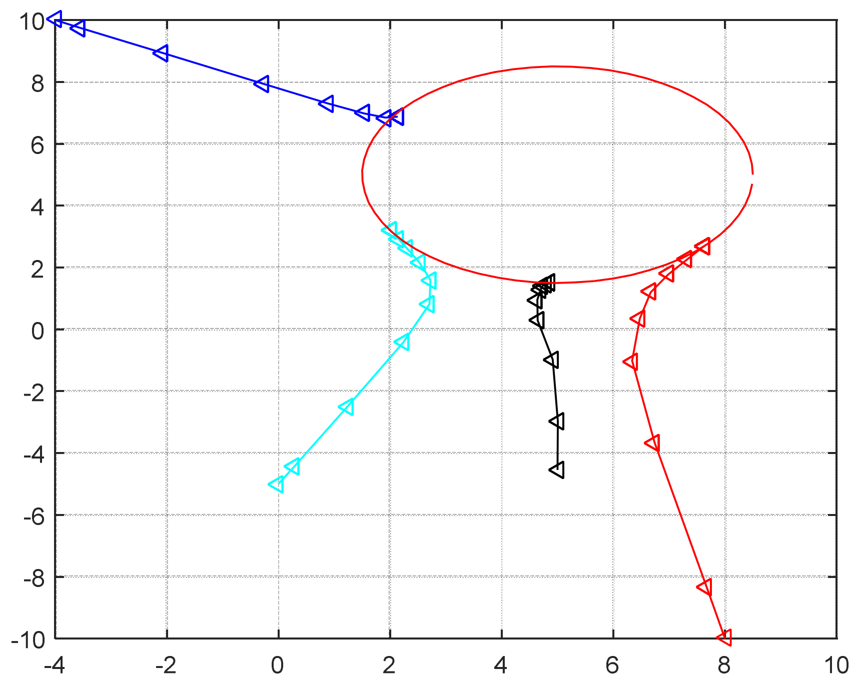


Figure 5 Formation of four robots starting from arbitrary positions

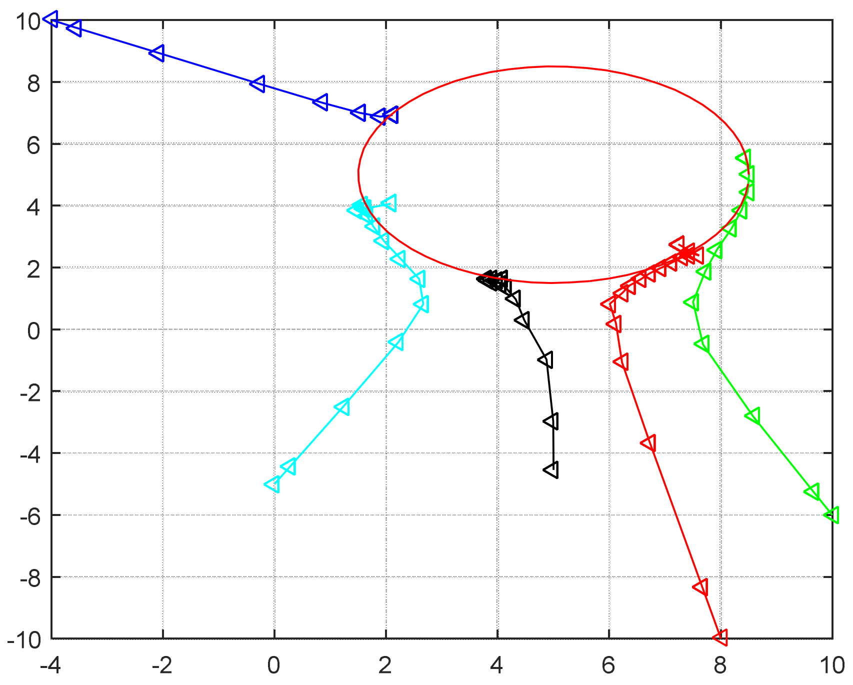


Figure 6 Formation of five robots starting from arbitrary positions

2.4 Control Strategy

The control structure used for the cooperative control of multi mobile robots has been proposed by (Mehrjerdi, Saad, & Ghommam, 2011)is shown in Figure 7. The control strategy has been divided into two parts

1. High level Fuzzy Cooperative Control
2. Low Level PID Control

The trajectory planner generates the trajectories for each robot and gives the reference points to the High level fuzzy controller which then generates an output signal for the Low level PID controllers. The Low level PID controller is responsible for accurate tracking of robot's left and right motor speeds while the High level controller is responsible for the cooperation of robots as well as path following of individual robots. The feedback is collected from the robot's wheel motors in the form of speed and then a localization algorithm is used to locate each robot's position and orientation which is then sent back to the fuzzy controller.

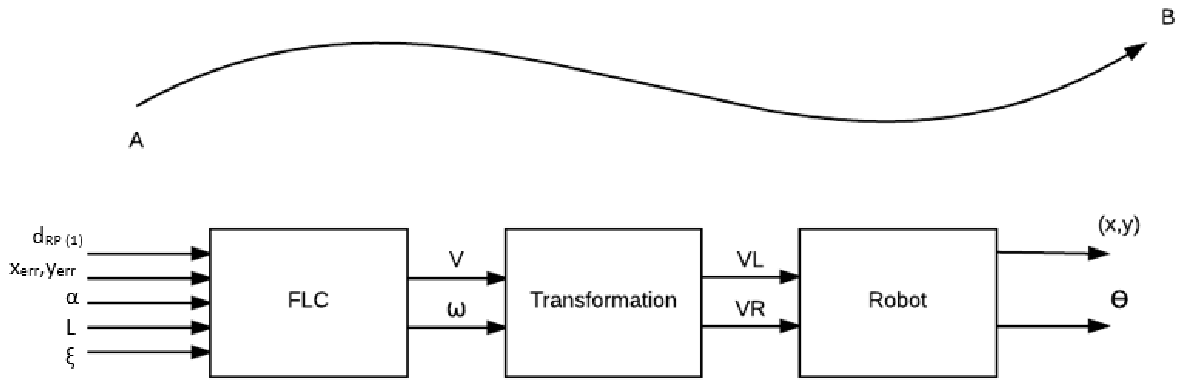


Figure 7 High level control structure for robot navigation

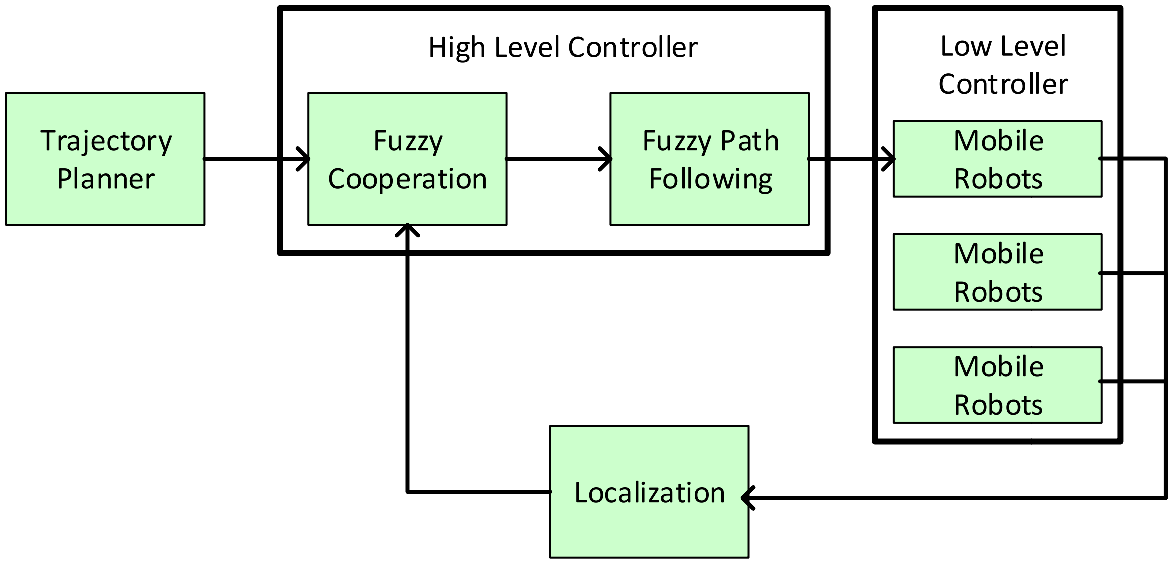


Figure 8 Multi Mobile Robot Cooperative Control Structure(Mehrjerdi et al., 2011)

The Low level control is a PID controller which is responsible for the accurate tracking of the Left and right wheel motors velocities $\omega_{r(i)}, \omega_{l(i)}$. Figure 9shows the structure of the low level PID controller.

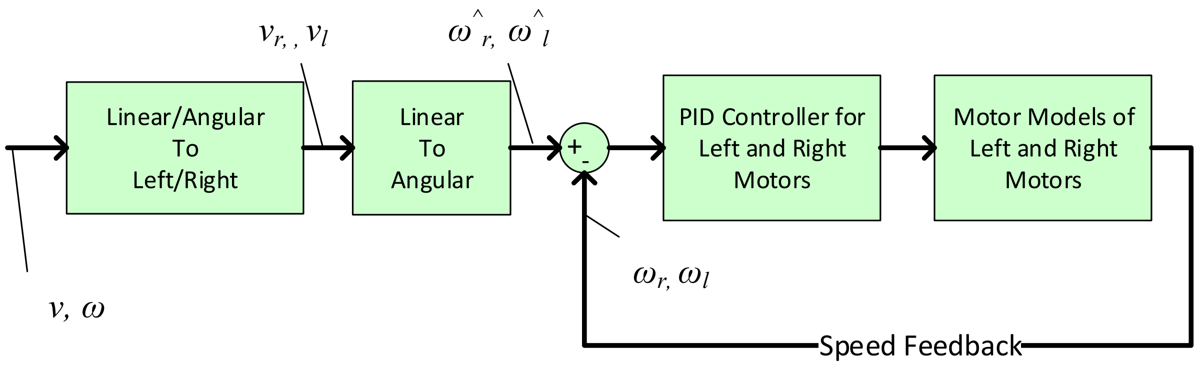

$v_{r,}, v_l$
$\hat{\omega}_r, \hat{\omega}_l$
Linear/Angular To Left/Right
Linear To Angular
+
-
PID Controller for Left and Right Motors
Motor Models of Left and Right Motors
ω_r, ω_l
v, ω
Speed Feedback

Figure 9 Low Level PID Control Structure(Mehrjerdi et al., 2011)

The output from the fuzzy controller is in the form of linear and angular velocities $v_i, \omega_i$of therobot therefore the velocities must first be converted to left and right wheel motor velocities $v_{r(i)}, v_{l(i)}$. This conversion is carried out using equation (3)and(4).

The above conversion gives the linear velocities of the left and right wheel but the PID controller needs the angular velocity as reference. Therefore the angular velocities $\omega_{r(i)}, \omega_{l(i)}$ of the right and left motors can be obtained using following conversion.

$$v_i = r\omega_i$$

Where r is the radius of the right and left wheels of the robot.The velocities $\omega_{r(i)}, \omega_{l(i)}$ are given to the PID controller as reference signal. The PID controller tracks these velocity references with desired accuracy and within certain time limits. The whole functionality of the High Level Fuzzy Cooperative Control is dependent on the assumption that the inner PID controller accurately tracks these velocity references.

2.5 Networked Control and Mobile Sensor Network

Cooperative control can only be achieved if there is wireless communication link between all the participating robots. Network control strategies are required to achieve good control performance . As discussed in the preceding section the cooperation between robots is achieved with the high level fuzzy logic controller. In this work it has been assumed that iealcommunication link between all the robots exists. A wireless connection can be establishee with either aXbee, ZigBee or NRF module.

2.5.1 High Level Fuzzy Cooperative Control

The High Level Fuzzy Controller is a Takagi sugeno type fuzzy controller which is responsible for the path following of the individual mobile robots as well as the group cooperation between the robots.Figure 10shows the Fuzzy Control structure.

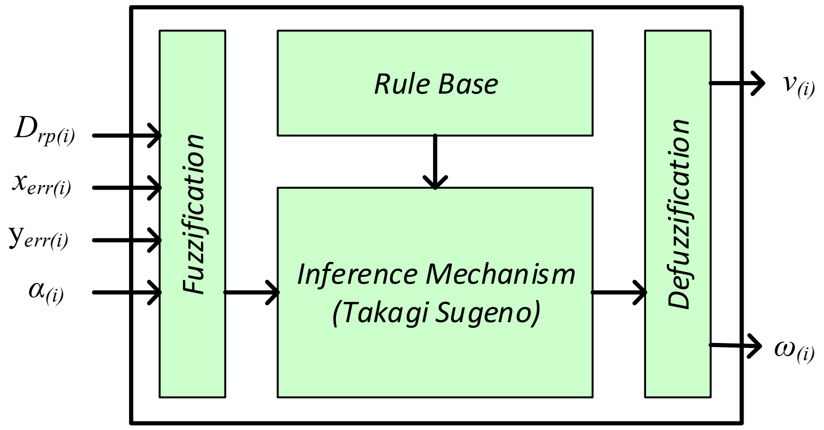


Figure 10 Fuzzy Control Structure

To explain the inputs to the fuzzy controller refer toFigure 11. From Figure 11$i = 1,\dots,k$ is the robot number. The position and orientation of the robot can be described by the vector $P_i = [x_i, y_i, \varphi_i]^T$.The path to be followed by the robots is divided into n discrete set of points, where $n = 0,\dots,f$. Each point on the path can be described by the vector$Q_{di(n)} = [x_{di(n)}, y_{di(n)}, \xi_{di(n)}]^T$. Where $Q_{di(0)}$ being the starting point of the path and $Q_{di(f)}$ being the final point. Whereas$Q_{di(n)}$ represent the nth sample point on the path.

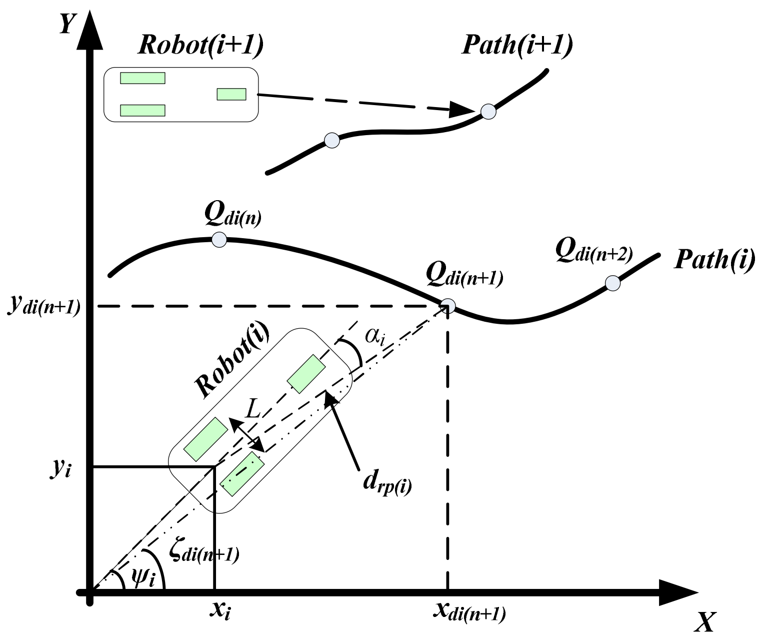


Figure 11 Robot path following parameters(Mehrjerdi et al., 2011)

The inputs to the fuzzy controller are $D_{rp(i)}$, $x_{err(i)}$, $y_{err(i)}$ and $\alpha_{(i)}$ where

- is the distance of the actual position of the ith robot from its next desired point on the path.
- $x_{err(i)}$is the error of the ith robot's position the X direction.
- $y_{err(i)}$is the error of the ith robot's position in the Y direction.
- $\alpha_{(i)}$is the error in orientation of ith robot from the next desired point.

$D_{rp(i)}$is calculated using the distance formula as shown in the following equation.

$$D_{rp(i)} = \sqrt{(x_{di} - x_i)^2 + (y_{di} - y_i)^2} \tag{9}$$

Wherex_{di}, y_{di} are the coordinates of the next desired point on the path while x_i, y_i are the coordinates of the actual position of the robot.

$x_{err(i)}, y_{err(i)}$and $\alpha_{(i)}$are in the robot's reference frame and are calculated from the following equation. Refer to equation (10) for the rotation matrix.

$$\begin{bmatrix} x_{err(i)} \\ y_{err(i)} \\ \alpha_{(i)} \end{bmatrix} = \begin{bmatrix} \cos\varphi_{(i)} & \sin\varphi_{(i)} & 0 \\ -\sin\varphi_{(i)} & \cos\varphi_{(i)} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{di} - x_i \\ y_{di} - y_i \\ \zeta_{di} - \varphi_i \end{bmatrix} \tag{10}$$

Where φ_i is the robots orientation which is adjusted to $\pm\pi$radians.$\zeta_{di}$is the orientation of the next desired point which can be calculated via using the following equation.

$$\zeta_{di} = \tan^{-1}\left(\frac{y_{di} - y_i}{x_{di} - x_i}\right) \tag{11}$$

The fuzzy controller generates the required linear and angular velocities v_i, ω_i of the robot. The output of the fuzzy controller is used as a set point the low level PID controller.

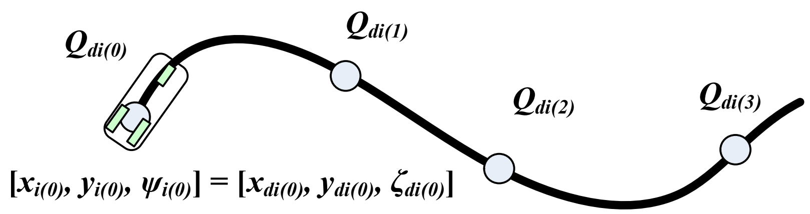
2.6 Path Planning/Obstacle Avoidance

2.6.1 Path Following and Cooperation Problem

The two main tasks of the robot is to follow a desired path as well as maintain a formation with other robots along the journey. The main idea behind the robot following a continuous path is to analyze the path in a set of discrete points. Each point serves as an intermediate destination to the robot, tracking each point will make the robot appear to be moving smoothly in a continuous path. The paths are being generated by the trajectory generator, the paths are modeled using a fifth order polynomial to generate reasonably difficult curved paths for the robots to follow.

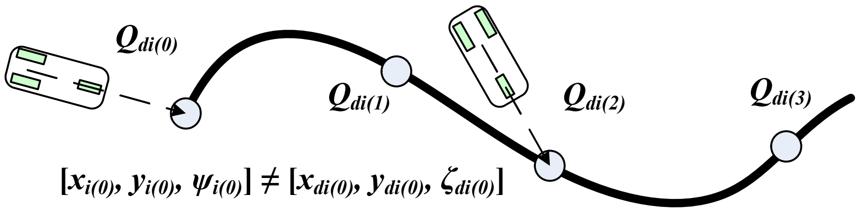
The task of the robot is to move smoothly in a continuous path with best precision possible, passing through every point is not necessary but the robot should pass within the vicinity of a sampling point. The path following is divided into two categories.

1. When the robot is initially placed on the predefined path it then follows the path (see Figure 12).
2. When the robot is initially placed away from the predefined path as shown in Figure 13, it then moves forward to reach the path and track it.



$Q_{di(0)}$
$Q_{di(1)}$
$Q_{di(2)}$
$Q_{di(3)}$
$[x_{i(0)}, y_{i(0)}, \psi_{i(0)}] = [x_{di(0)}, y_{di(0)}, \zeta_{di(0)}]$

Figure 12 Robot Placed on the Path(Mehrjerdi et al., 2011)



$Q_{di(0)}$
$Q_{di(1)}$
$Q_{di(2)}$
$Q_{di(3)}$
$[x_{i(0)}, y_{i(0)}, \psi_{i(0)}] \neq [x_{di(0)}, y_{di(0)}, \zeta_{di(0)}]$

Figure 13 Robot not placed on path(Mehrjerdi et al., 2011)

As mentioned previously P_i being the robot's position and Q_{di} being the next desired point we have considered v_i and ω_i as robot's linear and angular velocities. The objective of the path following controller is to generate an output such that the robots velocity$u_i = [v_i, \omega_i]^T$tracks the desired reference velocity$u_{di} = [v_{di}, \omega_{di}]^T$.As the robot's velocity u_i tracks the desired reference velocity $u_{di}$the error minimizes i.e. $\|u_i - u_{di}\| \to 0$ which will eventually make $\|P_i - Q_{di}\| \to 0$

Now consider a team of mobile robots each of which having its own path following controller like the one described above, hence every individual robot will follow its desired path. For the cooperation problem every robot must follow their respective path in such a way that they maintain an inter robot formation along their journey as well as they must reach their final goal at the same time regardless of their path lengths.

2.6.2 Fuzzy Path Following and Cooperative Controller

The fuzzy controller of Figure 10 is responsible for the path following and cooperation of multiple mobile robots it has two outputs v_i being the linear velocity and ω_i being the angular velocity for the robot to track. The fuzzy controller is based on Takagi - Sugeno Fuzzy Model. The control law equations are of the form.

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = \begin{bmatrix} f_1(D_{rp(i)}, x_{err(i)}, y_{err(i)}, \alpha_{(i)}) \\ f_2(D_{rp(i)}, x_{err(i)}, y_{err(i)}, \alpha_{(i)}) \end{bmatrix} \tag{12}$$

As mentioned previously, the task of the fuzzy controller is to make the robot pass within the vicinity of the desired sampling point if not through it. The controller is designed such that if the sampling points are placed close to each other, then the robot will move at a slower speed but with higher precision. However if the sampling points are placed far from each other, then the robot movement will be less precise but with higher speed.

2.6.2.1 Membership Function $D_{rp(i)}$

The membership functions of $D_{rp(i)}$is shown in Figure 14.$D_{rp(i)}$is the distance of the robots actual position to the next desired point, the input has five membership functions namely **VeryClose**,**Close**, **Medium**, **Far** and **VeryFar**. it is measured in meters (m). The membership functions **VeryClose**, **Close**, **Medium** are placed close to each other this helps the robot maneuver more precisely but with lower speed when the desired sampling point is in close vicinity, however the **Far** and **VeryFar** membership functions covers large region and are placed far from each other this helps the robot move at higher speed but with less precision. Note that all the membership functions have overlapping regions this helps the robot to move smoothly during the transition from one membership function to another.

2.6.2.2 Membership Function for α_i

The membership function of α_i is shown in figure Figure 15. $\alpha_i$is the error in orientation of the robot to that of the next sampling point. It has five membership functions namely **BigNegative**, **mNeg**, **Small**, **mPos**, **BigPositive** it is measured in degrees. The membership functions **mNeg**, **Small** and **mPos** are placed closely to make fine adjustments to the robots orientation when the error is small, while **BigNegative** and **BigPositive** are placed far to compensate for the large errors in orientation and quickly overcome the orientation error. Here also the membership functions have overlapping regions to have a smooth transition between membership functions.

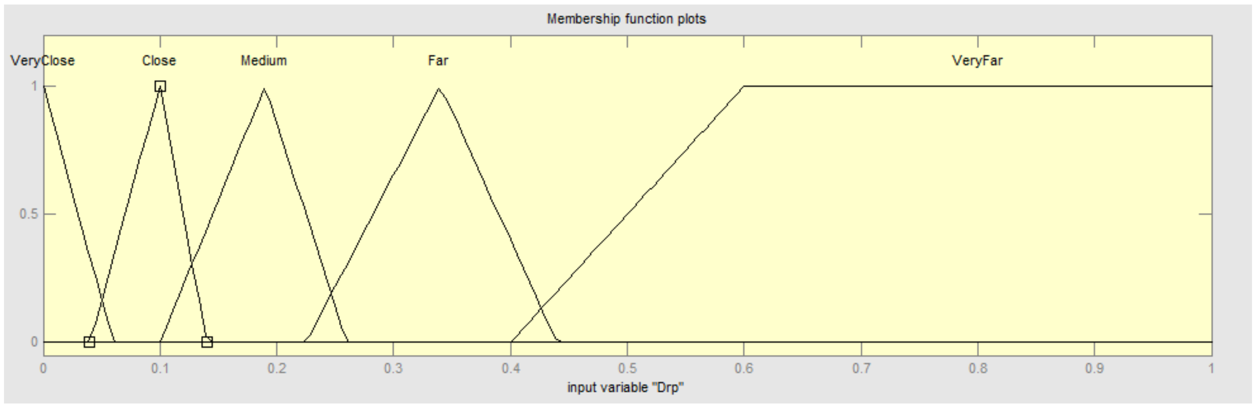


Figure 14 Membership Function of $D_{rp(i)}$

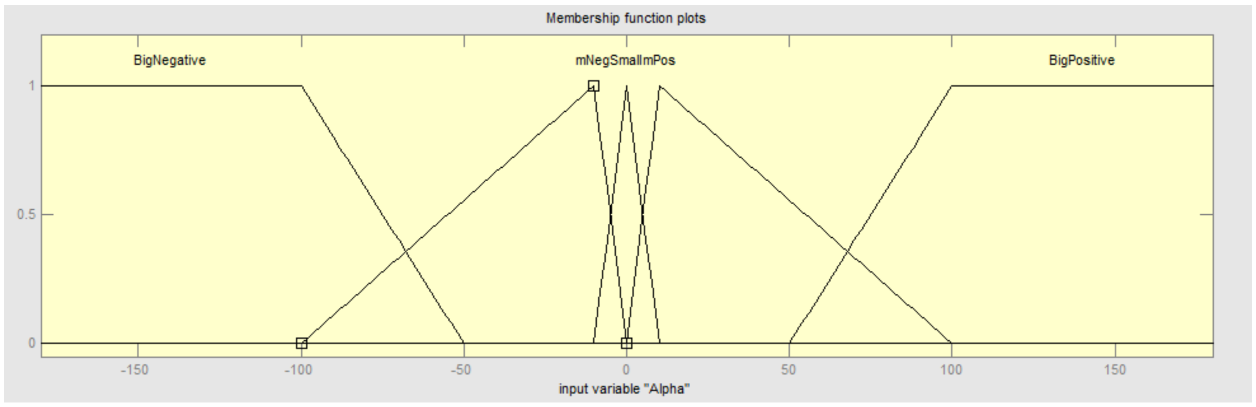


Figure 15 Membership Function of α_i

2.6.2.3 Linear and Angular Velocity obtained from $D_{rp(i)}$ and α_i

The linear and angular velocities obtained from the fuzzy controller related to $D_{rp(i)}$ and α_i are shown in Figure 16 and Figure 17 respectively. The linear velocity is measured in m/s (meter per second) while angular velocity is measured in degrees/sec. From Figure 16 it can be seen that the linear velocity increase as the distance between the robot and the next sampling point $D_{rp(i)}$ increases and vice versa. This way the robot moves faster when the sampling point is far and moves slower as it comes closer. Notice that the error in orientation α_i has minimal effect on the linear velocity and rightly so. However form Figure 17 it can be seen that the angular velocity profile is dominated by the α_i input, as the error in orientation increases the robot angular velocity increases and vice versa, this helps the robot make turns quickly and slowly as the error in orientation increases and decreases respectively.

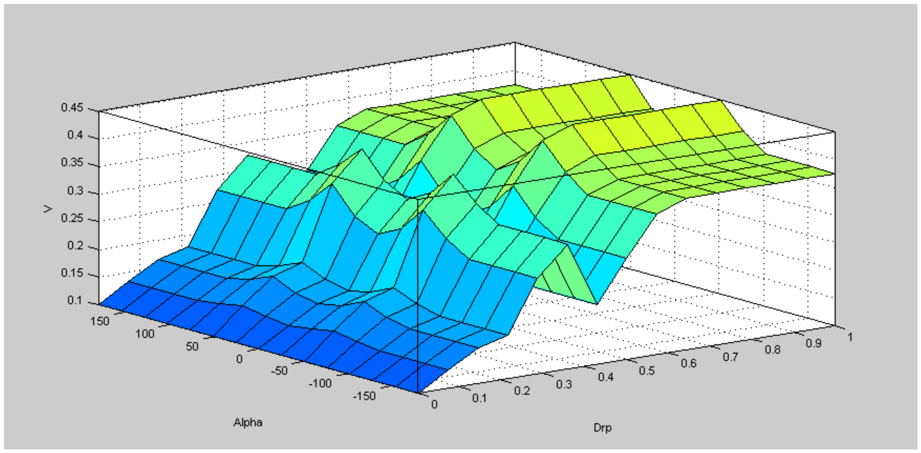


Figure 16 Linear Velocity obtained by Fuzzy Controller for inputs $\boldsymbol{D_{rp(i)}}, \boldsymbol{\alpha_i}$

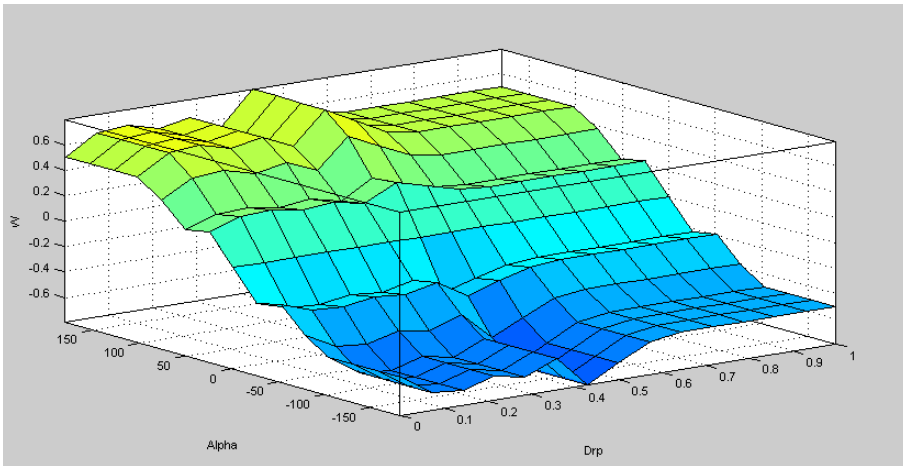
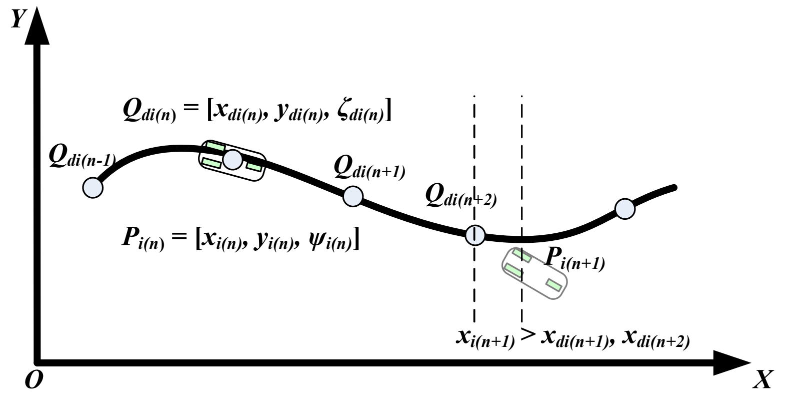


Figure 17 Angular Velocity obtained by Fuzzy Controller for inputs $\boldsymbol{D_{rp(i)}}, \boldsymbol{\alpha_i}$

2.6.2.4 Robot ahead of its Path Problem

One of the probems of cooperative path following is that all robots should reach their final goal at the same time. Let us consider a case where the robot is at position $P_i = [x_i, y_i, \varphi_i]^T$ has to move from point $Q_{di(n)} = [x_{di(n)}, y_{di(n)}, \xi_{di(n)}]^T$ to $Q_{di(n+1)}$ this means that the robot next desired sampling point is $Q_{di(n+1)}$. If the robot passes the point $Q_{di(n+1)}$ and $Q_{di(n+2)}$ thenwhen moving to next step the $Q_{di(n+1)}$ is left behind from the actual position of robot or mathematically ($x_i > x_{di} \Rightarrow x_{err(i)} < 0$). This probelm is explained in Figure 18.To avoid the condition of robot moving backwards we use the input $x_{err(i)}$. The fuzzy rule base contains the following rule to avoid the the above mentioned condtion.

- **Rule:** $if\ x_i > x_{di}\ then\ Robot\ will\ stop, untill\ the\ condition\ x_i < x_{di}\ is\ met.$



Y
$Q_{di(n)} = [x_{di(n)}, y_{di(n)}, \zeta_{di(n)}]$
$Q_{di(n-1)}$
$Q_{di(n+1)}$
$Q_{di(n+2)}$
$P_{i(n)} = [x_{i(n)}, y_{i(n)}, \psi_{i(n)}]$
$P_{i(n+1)}$
$x_{i(n+1)} > x_{di(n+1)}, x_{di(n+2)}$
O
X

Figure 18 Robot ahead of the Target Point(Mehrjerdi et al., 2011)

2.6.2.5 Robot moving Parallel to its Path Problem

Another problem related to path following that needed to be addressed,caused the addition of the input $y_{err(i)}$.If there is an error in the vertical position of the robot $y_{err(i)}$but not in the orientation then the robot will traval parallel to the desired path and will never reach it. This can be seen from Figure 19. The input $D_{rp(i)}$ has minimal effect on the angular velocity therfore it cannot turn the robot towards the target point there need to be an error in orientation $\alpha_i$to turn the robot towards the target point.

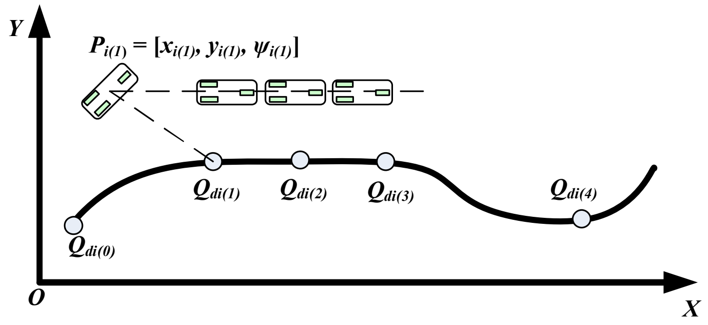


Figure 19 Robot travelling Parallel to the path(Mehrjerdi et al., 2011)

In order to address the above mentioned problem we introduced a new variable λ_i which is angle measured in degrees. λ_i is added to α_i if there is an error in y coordination. it will help the robot to turn towards the target point even if the $\alpha_i = 0$, it will tell the robot to turn towards the target point and will decrease as the $y_{err(i)}$decreases and will be zero when robot catch the desired path.

$$if\left|y_{err(i)}\right| > 0 \Rightarrow \alpha_i(new) = \lambda_i + \alpha_i$$

The membership function of $y_{err(i)}$ is shown in Figure 20. It is divided into seven membership functions namely **nVeryFar**, **nFar**, **nClose**, **VeryClose**, **Close**, **Far**, **VeryFar** distributed symetrically from -3 to 3, it is measured in meters. The relationship between $y_{err(i)}$ and λ_i is shown in Figure 20, it can be seen that λ_i increases as $y_{err(i)}$ increases and vice versa. Hence it makes sure that robot turn towards the target point if there is an error in the ycoordination then the robot.

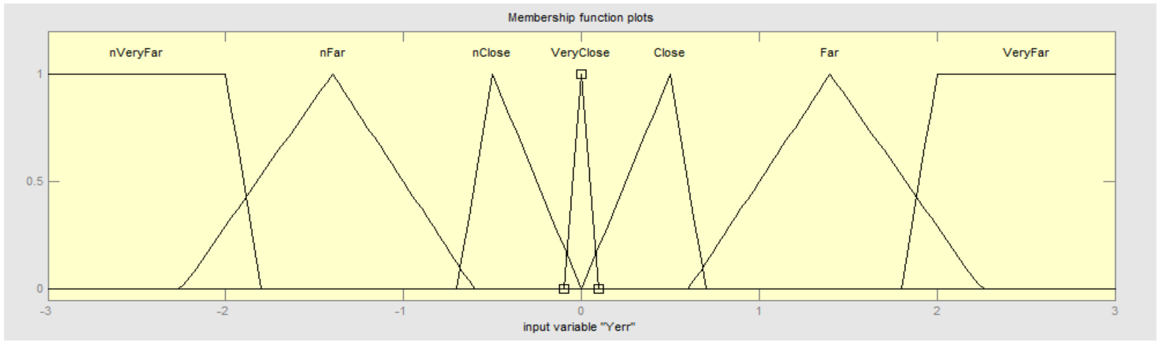


Figure 20 Membership function of $y_{err(i)}$

2.6.2.6 Cooperation Using X-Coordination

The path of each robot is divided into equal number of small segment and the parameter ς_i will keep track of the current segment the robot is executing. The robots are required to maintain $\varsigma_i = \varsigma_j$for all i, j. For cooperation it is necessary that all the paths are divided into equal number of segments regardless of their shape and length. If the path is longer then the sampling points will be far from each other or the distance between sampling point will increase and vice versa. Hence the robot will move faster or slower if the sampling point are close or far from each other respectively. Therefore if each robot track their

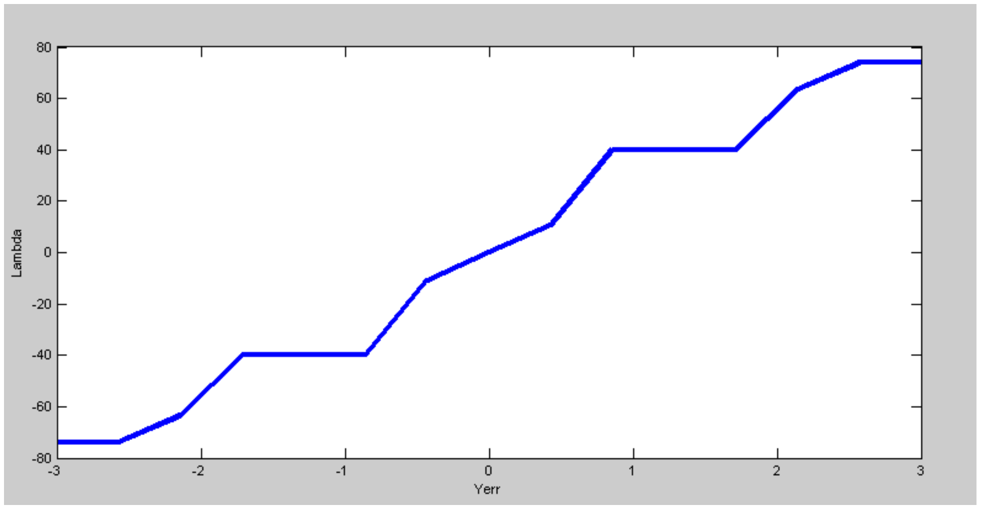


Figure 21 Lambda λ_i obtained form$y_{err(i)}$

respective sampling points then they will be moving in a cooperative behavior and will reach the final point at the same time. if a robot is ahead of its path then it will wait for other robots to catch up and then start following its path. This is achieved by x coordination error input i.e if $x_{err(i)} < 0$ then robot will stop. The effect of $x_{err(i)}$ and $D_{rp(i)}$on linear velocity $v_i$is shown in Figure 22and the obtained angular velocity$\omega_i$from the input$x_{err(i)}$ and $\alpha_i$is shown in Figure 23.

It can be seen from Figure 22 and Figure 23that if $x_{err(i)}$ is negative than both linear and angular velocities are zero.Hence if the robot is ahead of its path then it will not move and wait until the condition $x_{err(i)} < 0$ is present. However if the$x_{err(i)}$ is positive then the robot will move normally and track the desired path. However if the robot has fallen behind its desired path then it will move faster to catch the other robots once it catches the path then it will normally with the desired path.

2.6.2.7 Fuzzy Controller Rule Base

The rule base of the fuzzy controller is shown in Table 1. On the basis of these rules the fuzzy controller evaluates the inputs and generates the output. All the rules are very much self-explanatory like if the distance between the next sampling point and robot $D_{rp(i)}$ increase the

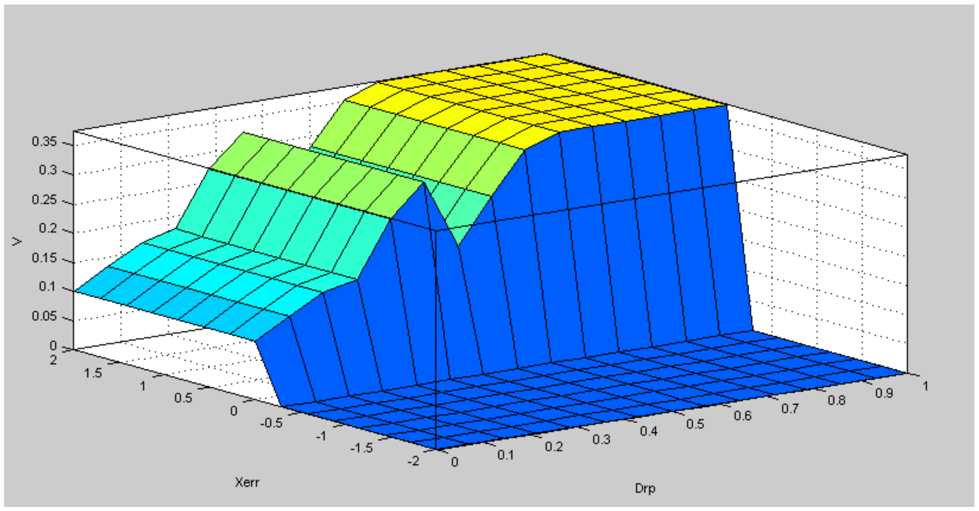


Figure 22 Linear Velocity obtained from $x_{err(i)}$ and $D_{rp(i)}$

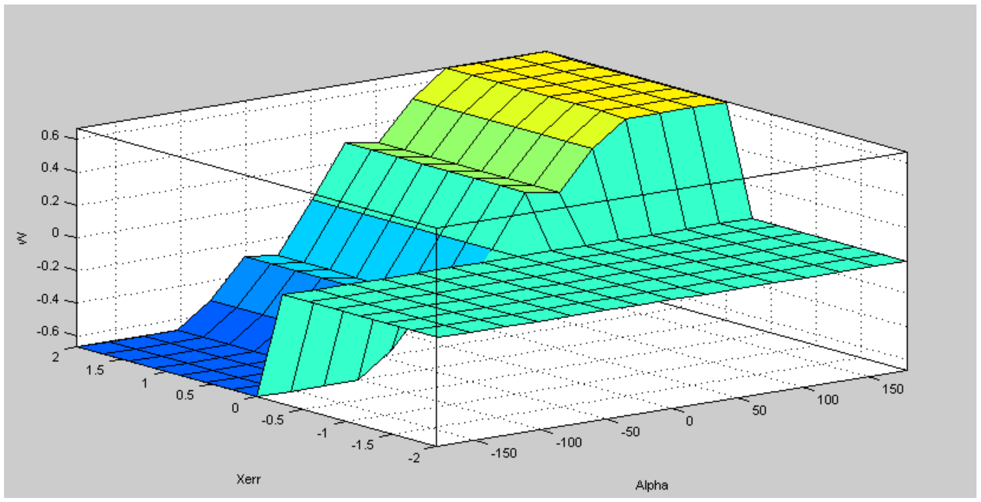


Figure 23 Angular Velocity obtained from $x_{err(i)}$ and α_i

fuzzy controller increases the linear velocity output so that the robot can catch its target. Similarly if the orientation error α_i increases then the fuzzy controller increases the angular velocity output to overcome the orientation error and correct the robot heading direction. Also if the robot is ahead of its path the fuzzy controller will make both linear and angular velocity outputs zero to stop the robot.

Table 1 Rule base of the fuzzy logic controller

Rule No	Fuzzy Rule Base
1	*If* $D_{rp(i)}$ *is* ***VeryClose*** *and* $X_{err(i)}$ *is* ***Positive*** *then* v_i *is* ***VeryVerySlow****,* ω_i *is* ***Zero***
2	*If* $D_{rp(i)}$ *is* ***Close*** *and* $X_{err(i)}$ *is* ***Positive*** *then* v_i *is* ***VerySlow,***ω_i *is* ***Zero***
3	*If* $D_{rp(i)}$ *is* ***Medium*** *and* $X_{err(i)}$ *is* ***Positive*** *then* v_i *is* ***Slow,***ω_i *is* ***Zero***
4	*If* $D_{rp(i)}$ *is* ***Far*** *and* $X_{err(i)}$ *is* ***Positive*** *then* v_i *is* ***Fast,***ω_i *is* ***Zero***
5	*if* $D_{rp(i)}$ *is* ***VeryFar*** *and* $X_{err(i)}$ *is* ***Positive*** *then* v_i *is* ***VeryFast,***ω_i *is* ***Zero***
6	*if* $X_{err(i)}$ *is* ***Positive*** *and* α_i *is* ***BigNegative****then* v_i *is* ***VeryVerySlow,***ω_i *is* ***BigNegative***
7	*if* $X_{err(i)}$ *is* ***Positive*** *and* α_i *is* ***MediumNegative****then* v_i *is* ***VeryVerySlow,***ω_i *is* ***Negative***
8	*if* $X_{err(i)}$ *is* ***Positive*** *and* α_i *is* ***Small*** *then* v_i *is* ***VeryVerySlow,***ω_i *is* ***Zero***
9	*if* $X_{err(i)}$ *is* ***Positive*** *and* α_i *is* ***MediumPositive****then* v_i *is* ***VeryVerySlow,***ω_i *is* ***Positive***
10	*if* $X_{err(i)}$ *is* ***Positive*** *and* α_i *is* ***BigPositive****then* v_i *is* ***VeryVerySlow,***ω_i *is* ***BigPositive***
11	*if* $X_{err(i)}$ *is* ***Negative*** *then* v_i *is* ***Zero,***ω_i *is* ***Zero***

2.6.3 Trajectory Planner

The idea of robot following a continuous path is to realize the path in set of discrete points. Each of the pointsare used as a destination for the robot one by one. Placing the points sufficiently close to each other makes the robot looks like it is following a continuous smooth path, this is the main task of the trajectory planner.

The trajectory planner has pre-defined paths, it divides each path into same number of pieces regardless of their length and shape. Each of these pieces serves as the intermediate destinations for the robots. The points are sent to the robot periodically and since the number of points are always the same hence the robots will always reach their destinations at the same time.

2.6.4 Simulation Results

In this section the results of the proposed high level Fuzzy Cooperative Controller and Low Level PID Controllers are presented and discussed. The differential drive robot model described earlier is used for simulations. The specification of the robot and DC Motor model are given below.

Table 2 Robot model and DC motor model parameters

S.No	Parameter	Value	Unit
Robot Model Parameters			
1	L_i	0.2	m
2	r	0.045	m
DC Motor Model Parameters			
1	R	3.07	$ohms$
2	L	0.04	mh
3	J	0.0294	$Kg.m^2$
4	B	0.0141	$N.m.s/rad$
5	K_e	0.65	$N.m/Amp$
6	K_t	0.65	$V.s/rad$

The simulation is carried out in MATLAB Simulink with a fixed step size of 1x10^{-2}secs. The Low Level PID Controller is tested in a separate simulation with the motor model and quantizer to simulate the effects of a low resolution digital encoder. The Fuzzy Path Following Controller is tested using a single robot model and trajectory generator, the controller is tested under different conditions. Finally the Fuzzy Cooperative controller is tested using three identical robot models and trajectory generator, two different experiments are carried out to test the cooperative behavior of mobile robots .

2.6.4.1 Speed PID Control of DC Motor

The DC motor speed control is essential part of the system. The fuzzy controller is working under the assumption that the output it is generating (v_i, ω_i) is tracked by the low level PID controller. The output of the fuzzy controller is the linear and angular velocities v_i, ω_i of the robot, these velocities are converted to Left and Right Linear velocities v_r, v_l for the left and right wheel motors and finally these Left and Right velocities are converted to Angular Left and Right Velocities which is used as a reference command for the PID Controller.

In this section we first present the open loop response of the PMDC motor, then the PID Speed controller for the DC motor result are presented and discussed.

Figure 24 shows the open loop response of the DC motor. There is no load torque applied to motor. The first graph shows reference vs. actual speed, the motor passes 1 rad/sec reference and settle to around 1.4 rad/sec value and then maintains that speed because of no external disturbances. The second graph shows the current as it can be seen that there is an initial peak, this is due to the fact that motor has to break the inertia to start moving after that the current remains steady as there are no further disturbances applied.

Figure 25 shows the speed PID controller response. The first graph compares the reference and actual speed. As can be seen that the PID controller quickly achieves the desired reference, the response time of the PID controller is less than 200 ms while the settling time is less than 500 ms. To check the controllers robustness against external disturbances we give step change in load torque at t = 5 sec. from figure it can be seen that the PID controller very quickly overcomes the external disturbance and converges to the reference value. In the current graph the initial peak as

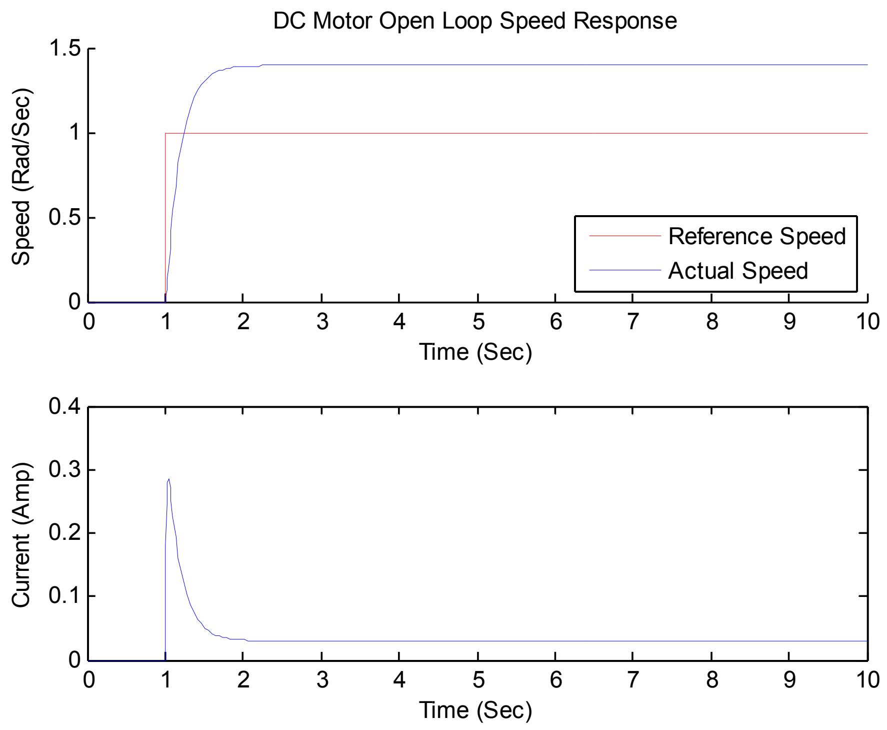


Figure 24 DC Motor Open Loop Response

discussed above is due to the inertia, however the step at t = 5 sec is because of the load torque step. The controller increase the effort i.e. Current to maintain the reference speed. The current then settles at a higher value, the same speed is regulated with a higher current value because of the external load torque.

2.6.4.2 Fuzzy Path Following

One of the objective of the robot is to follow its own path. For cooperative path following a necessary condition is that, the robot should be able to follow its own path therefore first we need to test the fuzzy controller for individual path following. We have tested the fuzzy controller with different scenarios in individual path following. First the robot is given a straight line path and then a curved path. The results of interest are the robots trajectory, linear and angular velocities and the y coordination error.

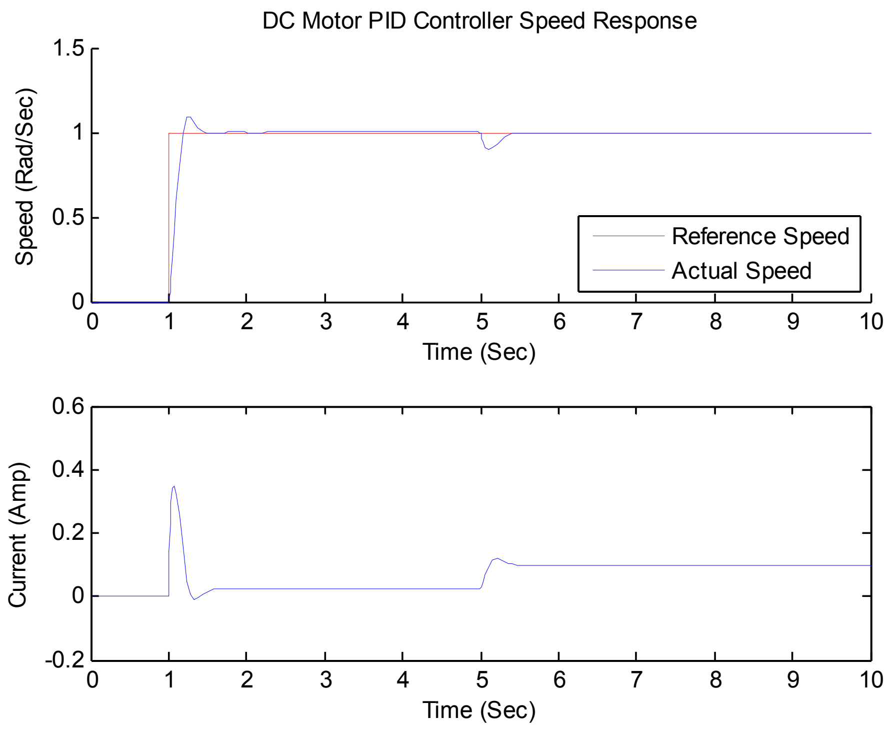


Figure 25 DC Motor Speed PID Controller Response

2.6.4.2.1 Straight Line path

In this test the robot is required to follow x-axis (straight line) the robot is initially placed away from its desired path $P_i = [x_i, y_i, \varphi_i]^T = [1, -1.5, 0]$. First the robot needs to catch the desired path and then follow it. Figure 26 shows the trajectory of the robot, it can be seen that the robot very quickly converges to the desired path and it follows a continuous smooth trajectory, after reaching the desired path it follows the path with minimal error.

Figure 27 shows the linear velocity of the robot. It can be observed that initially the robot didn't move because the robot was placed ahead of its desired path. The point at which the desired path crosses the robot's x position the robot now starts moving at a higher speed to overcome the distance it is lagging as soon as the robot catches the desired path it starts to move slowly to track its path.

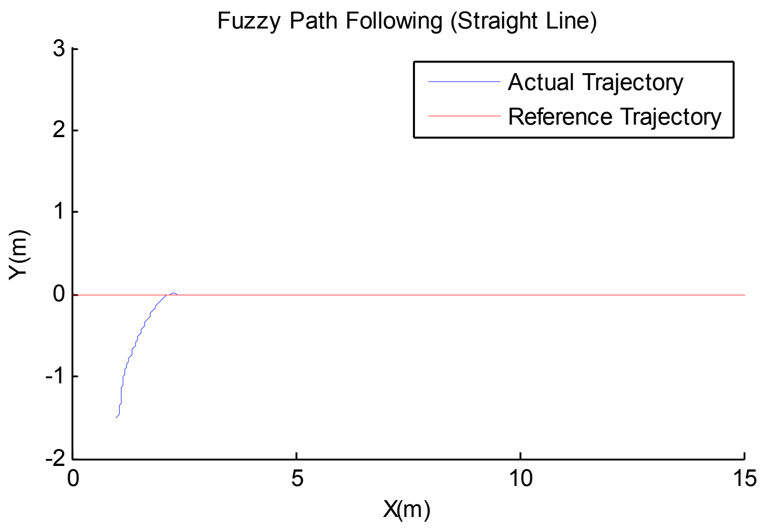


Figure 26 Fuzzy Path Following Straight Line Robot's Trajectory

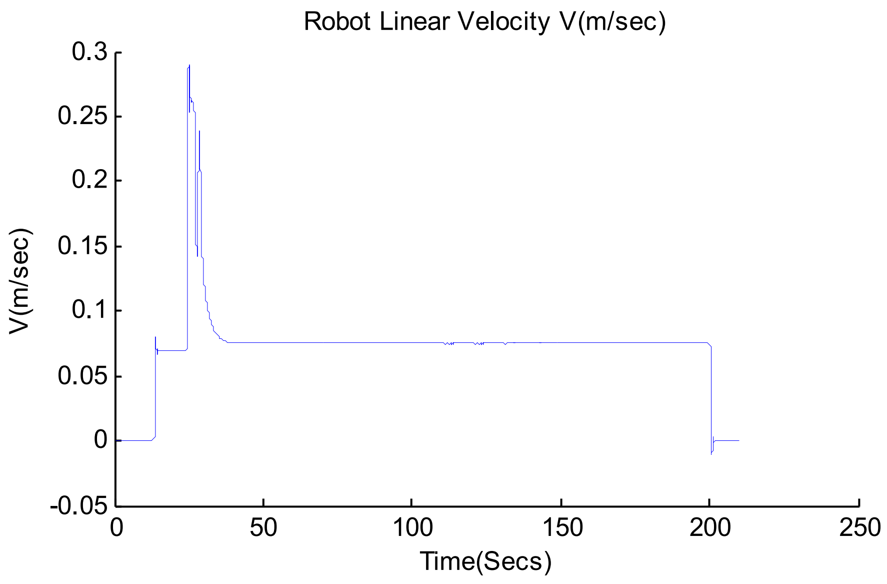


Figure 27 Fuzzy Path Following Straight Line Linear Velocity

Figure 27 shows the angular velocity of the robot following a straight line path. It can be seen that the robot takes a quick left turn then slowly turns right towards the path. This is due to the fact that robot's initial orientation is parallel to x- axis, therefore it must turn towards x-axis to reach it. After reaching the path, angular velocity becomes zero because there is no error in orientation.

Figure 29 shows the error in y-coordination y_{err} initially it remains constant as the robot was stationary, once the robot starts moving it quickly overcomes the y-coordination error and after it reaches the desired path y_{err} becomes almost zero as it moves forward along the path.

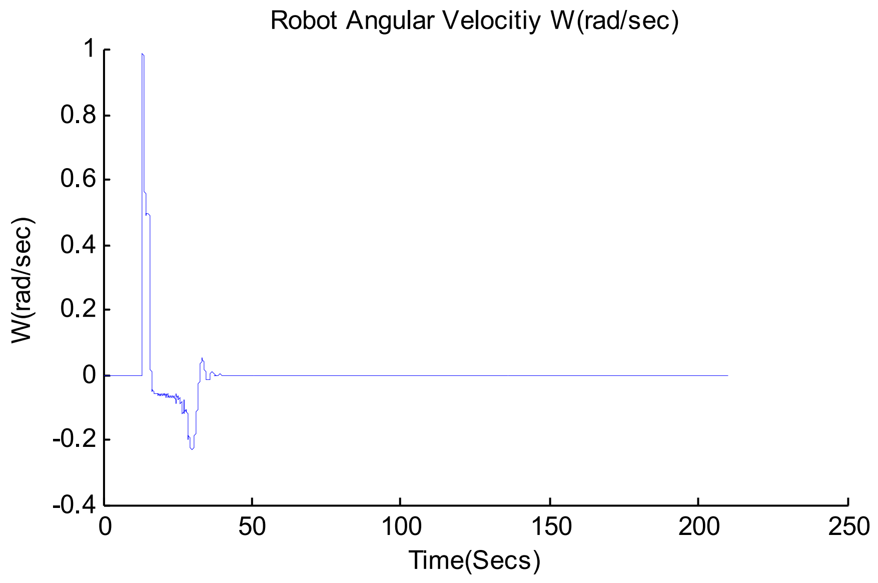


Figure 28 Fuzzy Path Following Straight Line Angular Velocity

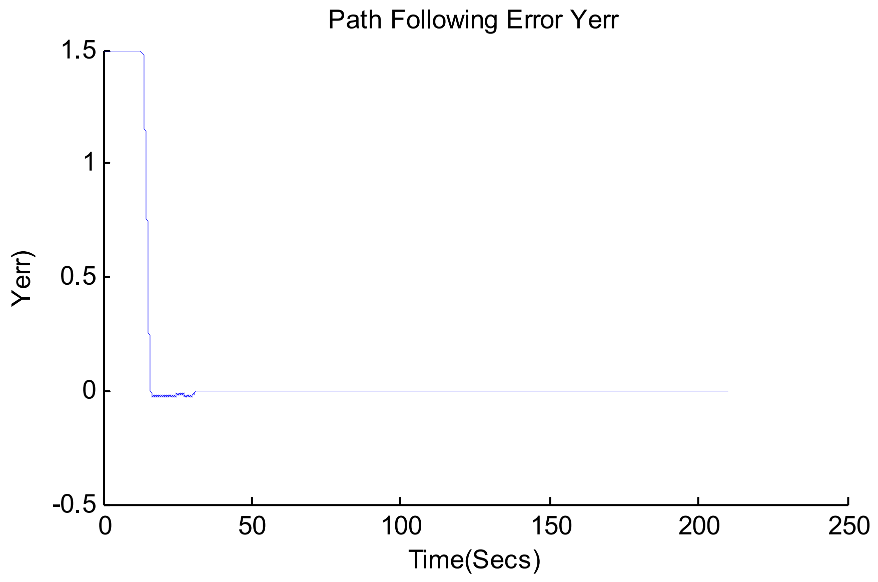


Figure 29 Fuzzy Path Following Straight Line y_{err}

2.6.4.2.2 Curved path

In the second test the robot is placed on a curved path. The curved path is generated using a fifth order polynomial to generate reasonably difficult curved path. This test is performed with two different initial conditions but on the same path, at first the robot is placed on the path secondly the robot is placed away from the path. Results are collected for the two cases and are discussed below.

2.6.4.2.3 Robot on path

In this experiment the robot is initially placed on the path. The position of the robot is given as$P_i = [x_i, y_i, \varphi_i]^T = [0.01, 1.5837, 0]^T$. Figure 30 shows the reference and actual trajectory of the robot , since the robot is already placed on the path therefore it doesn't have to travel long to catch the reference trajectory except for the small initial orientation error for which it compensate for very quickly, after that the robot tracks the trajectory with minimal error.

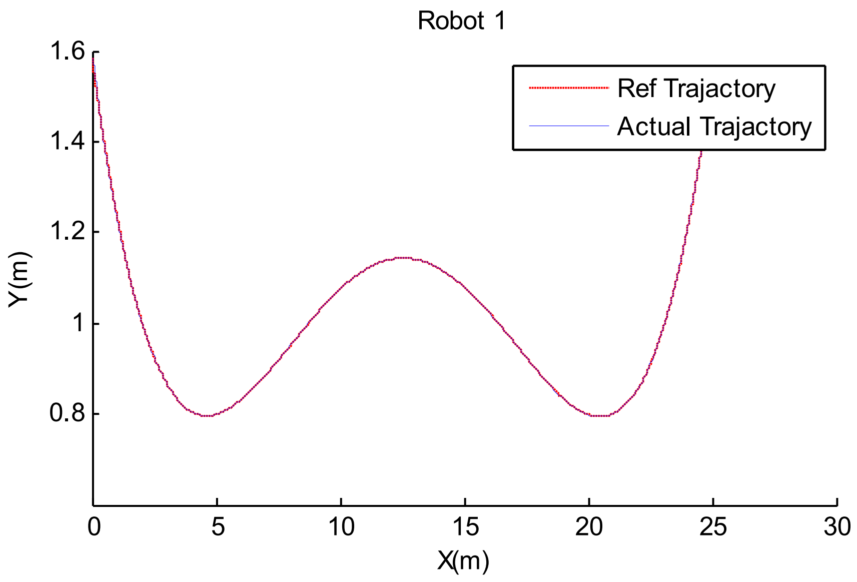


Figure 30 Fuzzy Path Following Curved Path Robot's Trajectory

2.6.4.2.4 Robot not on path

In this experiment the robot is initially placed away from the desired path. The initial position of the robot is given as$P_i = [x_i, y_i, \varphi_i]^T = [2.0, 0.9, 0]^T$. From Figure 31 it can be seen that the robot starts with a delay waiting for the desired trajectory to move ahead of its position. At start the robot turns towards the path and catches it, once it catches the path it tracks the path with minimal error as can be seen form figure.

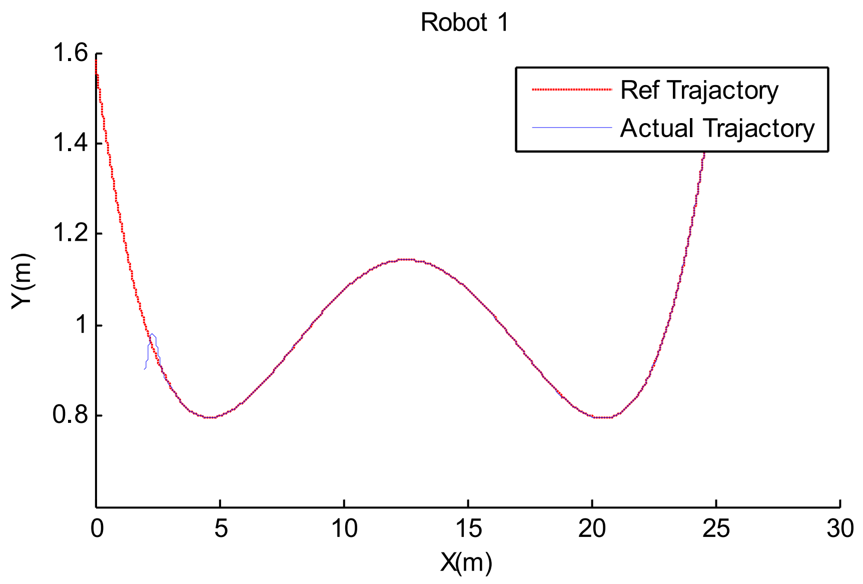


Figure 31 Fuzzy Path Following Curved Path Robot's Trajectory

2.6.5 Cooperative Path Following

Among the major objectives one of the main taskis cooperative path following of a team of mobile robots. This section discusses the results of cooperative path following experiments. To test the cooperation problem we used a team of three identical mobile robot models, two different experiments were designed to test the cooperative algorithm. First the robots were tested with paths of different lengths but placed on the path initially, then the robots are given same path but are placed away from the paths, simulation results are discussed below.

2.6.5.1 Experiment # 1

In this experiment three mobile robots are used each with a different path (different lengths). Robot 1 has the smallest path and Robot 3 has the longest path. The robots are initially placed on the path the initial positions of the robots and path lengths are given below.

$$P_1 = [x_1 \quad y_1 \quad \varphi_1]^T = \begin{bmatrix} 0.01 & 11.4303 & -\frac{\pi}{4} \end{bmatrix}^T, L_{path(1)} = 18.7372m$$

$$P_2 = [x_2 \quad y_2 \quad \varphi_2]^T = \begin{bmatrix} 0.01 & 6.7207 & -\frac{\pi}{4} \end{bmatrix}^T, L_{path(2)} = 21.2581m$$

$$P_3 = [x_3 \quad y_3 \quad \varphi_3]^T = \left[0.01 \quad 2.0489 \quad -\frac{\pi}{4}\right]^T, L_{path(3)} = 25.3926m$$

The robots are required to follow their respective paths as well as maintain a formation along the journey also they must reach the final destination at the same time regardless of the path lengths.

Figure 32 shows the cooperation of the mobile robots, as can be seen that robot 1 has the shortest path whereas Robot 3 has the longest path. The round markers are used to highlight the position of each robot at a given time. The first markers on each path will indicate the initial position of the robots att_0, the second marker will indicate the position of each robot at t_1 and so on. These markers are distributed equally with respect to time. As can be seen that the robot are placed vertically above each other initially, the successive markers shows that the robots are moving along their path but are also maintaining an inter robot formation throughout the journey, and reach their final destination at the same time.

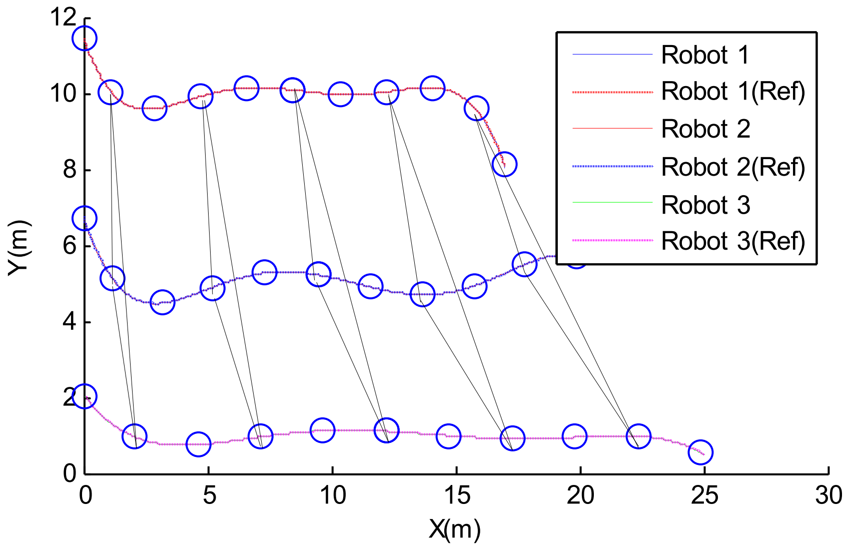


Figure 32 Cooperative Path Following Robot Formation (Different Paths)

Figure 33 shows the linear velocities of the robot. Since robot 3 has the longest path therefore it travel at a higher speed as can be seen from the figure v3 is greater than v1 and v2. The trajectory of robot 3 is not very curvaceous with respect to the other two robots therefore the linear velocity of robot 3 dominates angular velocity and it moves faster. However for robot 1 and 2 the trajectory is more curved and less flat therefore these two robots tends to move slowly.

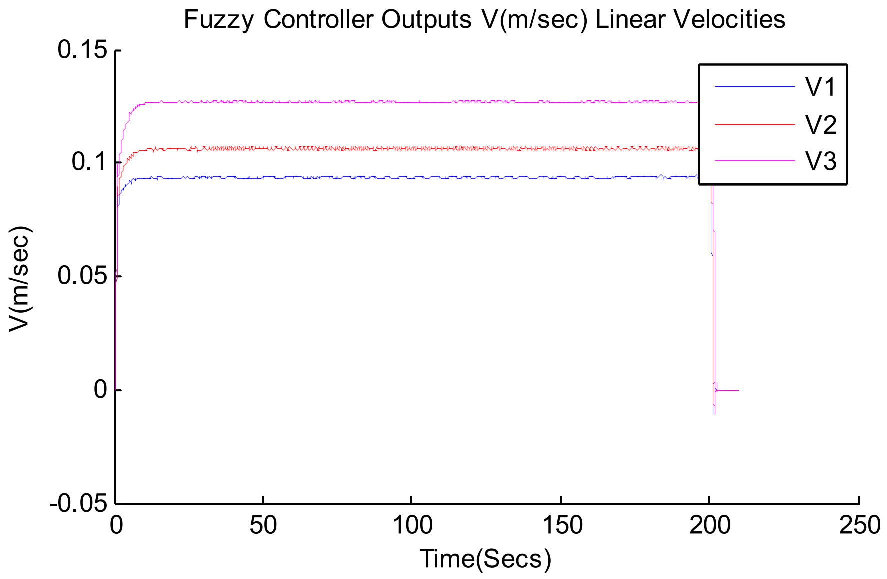


Figure 33 Cooperative Path Following Linear Velocities (Different Paths)

Figure 34 shows the angular velocities of the robots. The initial peaks represent the controller effort to compensate for the initial orientation error. It can be seen that robot 1 and 2 have higher angular velocity peaks this is because their paths are more curved, hence the fuzzy controller increases the angular velocity output to accurately track the curves. However for robot 3 it is the opposite case.

Figure 35 shows the y-coordination error$y_{err(i)}$. The initial peaks as discussed above are due to initial orientation error. It can be seen that once the robot track their respective path $y_{err(i)}$ becomes minimal. Robot 3 has the largest $y_{err(i)}$ because of the fact that it has the longest path it has to move fast, hence with lesser precision.

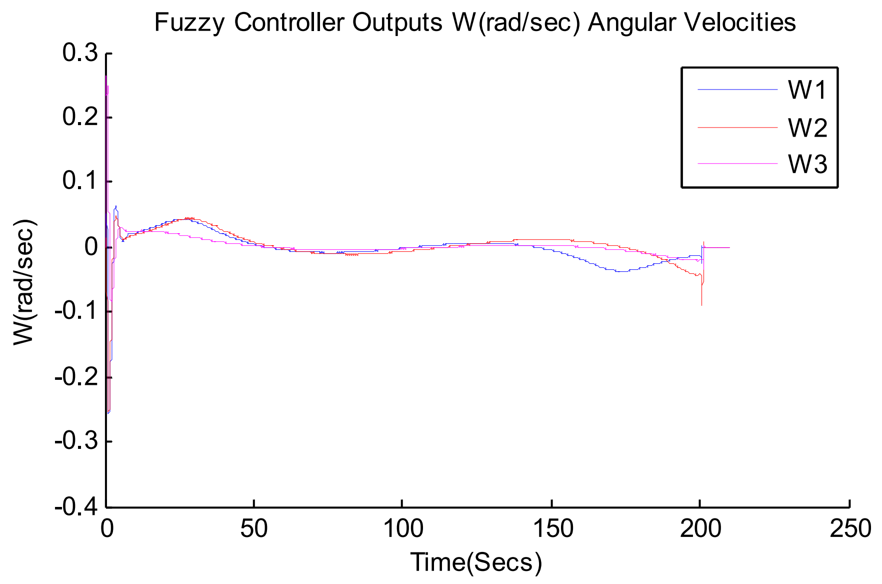


Figure 34 Cooperative Path Following Angular Velocities (Different Paths)

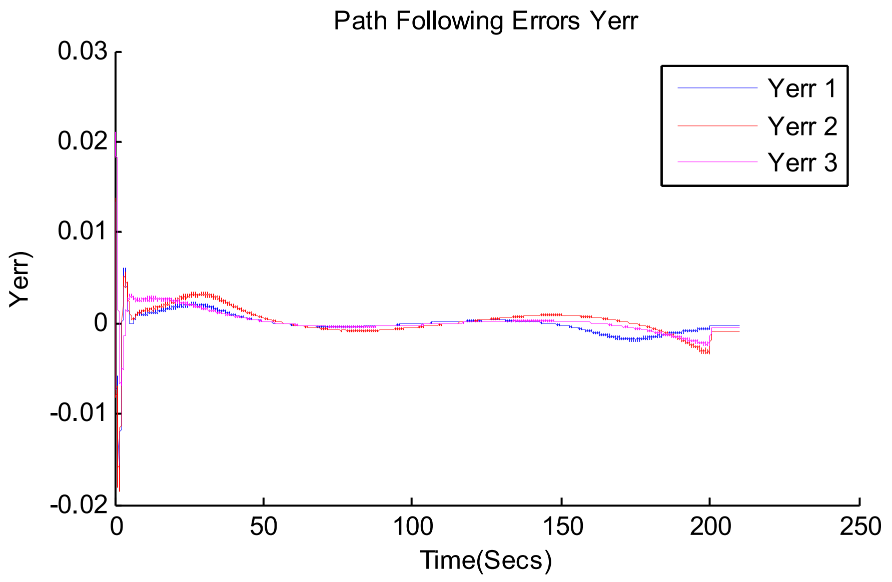


Figure 35 Cooperative Path Following y_{err} (Different Paths)

2.6.5.2 Experiment # 2

In this experiment three robots are used, each of which is provided with the same path, but they are placed away from their respective path initially. Robot 1 and robot 3 are placed ahead of their trajectory while Robot 1 is placed behind its trajectory. The robots initial positions are given below.

$$P_1 = \begin{bmatrix} x_1 & y_1 & \varphi_1 \end{bmatrix}^T = \begin{bmatrix} 1.0 & 11.0 & -\frac{\pi}{2} \end{bmatrix}^T, L_{path(1)} = 19.5288m$$

$$P_2 = \begin{bmatrix} x_2 & y_2 & \varphi_2 \end{bmatrix}^T = \begin{bmatrix} -0.5 & 8.0 & 0 \end{bmatrix}^T, L_{path(2)} = 19.5288m$$

$$P_3 = \begin{bmatrix} x_3 & y_3 & \varphi_3 \end{bmatrix}^T = \begin{bmatrix} 1.5 & -1.0 & 0 \end{bmatrix}^T, L_{path(3)} = 19.5288m$$

The robots are required to follow their respective as well as maintain an inter robot formation. Since the robots are not placed on their respective paths, therefore first they need to catch the trajectory, and they must reach their respective end points at the same time.

Figure 36 shows the cooperation of robots in this experiment. it can be seen that Robot 1 and Robot 3 are placed ahead of their respective paths, while Robot 2 is placed behind the trajectory. Note that Robot 2 starts immediately as soon as the simulation starts while Robot 1 and 3 remains idle waiting for the trajectory to move forward. This is because of the assumption that if a robot is ahead of its path then it will wait for the other robots to catch up. However if the robot has fallen behind then it will move faster and catch the trajectory.

The markers on the paths shows that around the 3 meter mark, the robots achieve synchronization (formation), then maintains the formation as they move forward along their respective paths and reach their final destination at the same time.

Figure 37 shows the y-coordination error of the robots. Initially y_{err} of Robot 1 and 3 is a function of the trajectory as the robots are standing still. However Robot 2 quickly overcomes its y-coordination error. It can be seen that as soon as the robot starts moving they overcomes the y_{err} very quickly. Once the robots achieve formation they follow their respective paths with minimal y-coordination error.

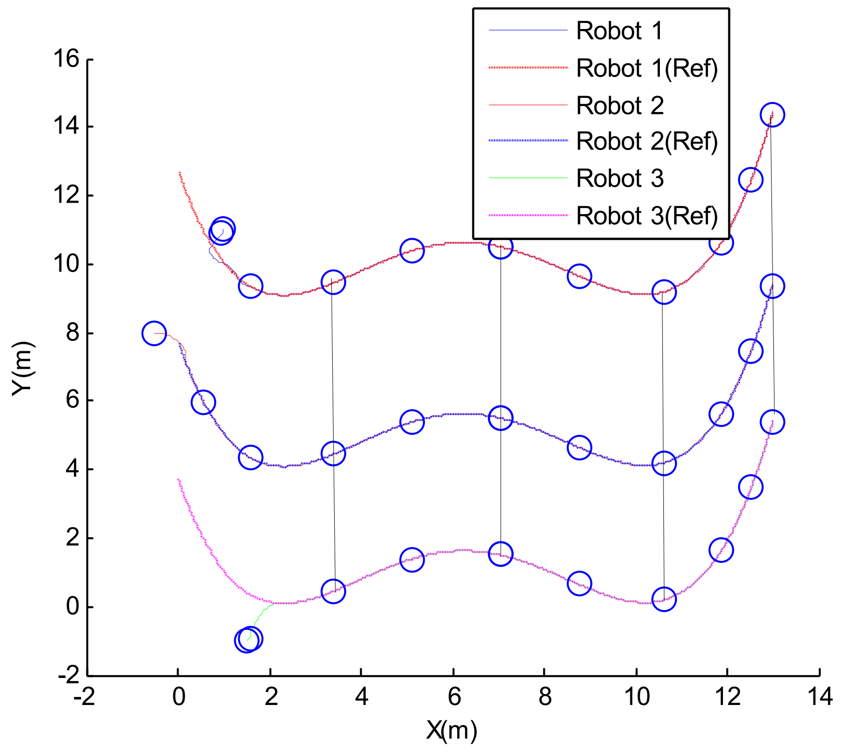


Figure 36 Cooperative Path Following Robot Formation

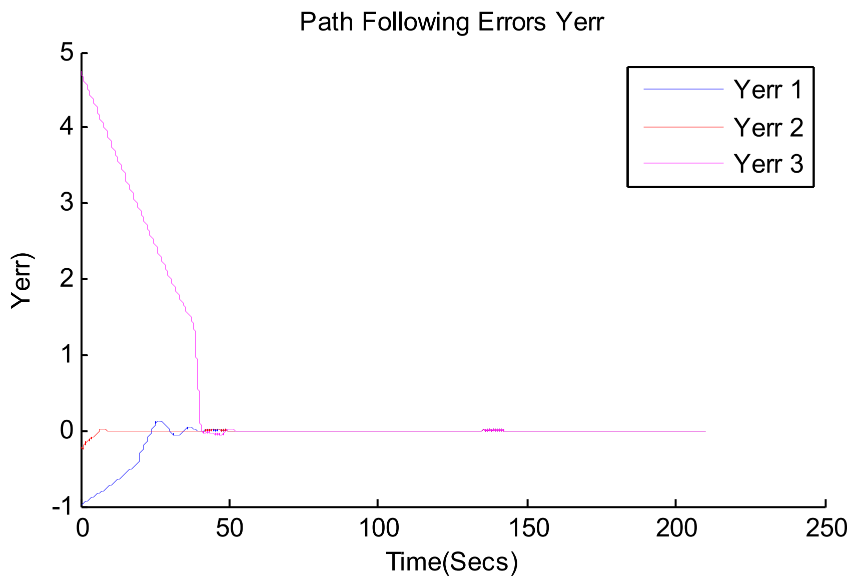


Figure 37 Cooperative Path Following y_{err}

2.7 Image Processing Module and SLAM

2.7.1 Thermal Imaging

2.7.1.1 Introduction

Thermal imaging is one of the most robust technologies ever developed to improve human vision. Normally, human vision is limited to a very narrow "visible" band of the electromagnetic spectrum. Thermal cameras usually detect radiation in the long-infrared range of the electromagnetic spectrum (8–14 µm) and convert the heat, emitted by objects on earth, into color images. These color images allow users to not only see in the dark, but to also observe differences in temperature to fractions of a degree. Since thermal imaging cameras allow viewing through darkness or smoke, they make it easy for the firefighters to quickly find the seat of a structure fire, or see the heat signature of visually obscured victims.

2.7.1.2 Hardware

A'FLiRDevKit'is used for performing thermal imaging, which includes a breakout as well as a Lepton® longwave infrared (LWIR) imager. The Lepton® LWIR module included in the kit offers a resolution of 80 × 60 pixels and a 51-deg Horizontal Field of View (HFOV) into the camera body. The camera has a thermal sensitivity less than 50 mK.

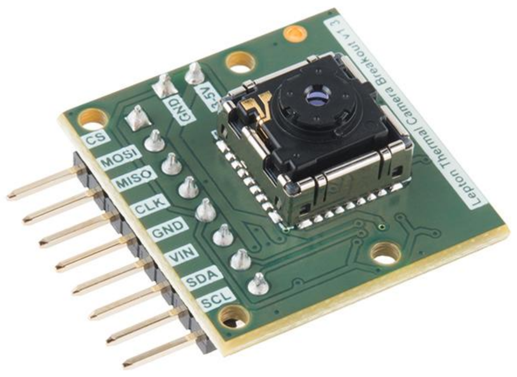


Figure 38 FLiRDev Kit

With a two-wire I2C-like serial-control interface, this kit can be easily connected with the GPIO pins of a Raspberry PI controller for processing the acquired thermal images easily.

2.7.1.3 Thermal imaging results

Fig. 40 below shows the images acquired from the thermal camera:

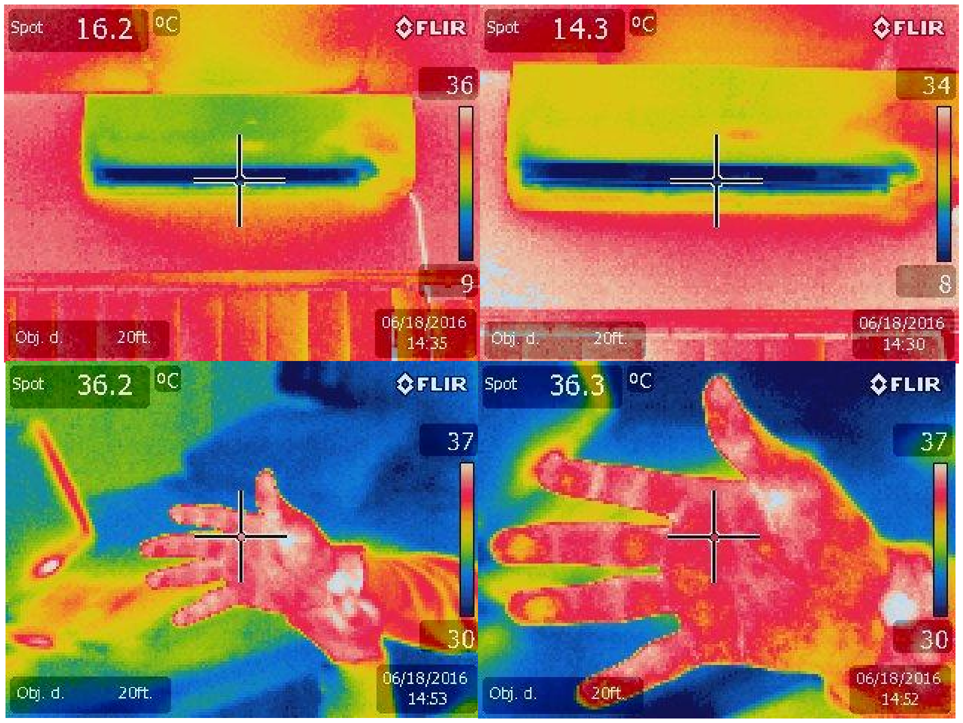


Figure 39 Multiple images taken at near and far distance from objects

Development of algorithms is underway to do the ‘ranging’ using image processing of the thermal images. Such algorithm will assist in measuring the objects’ distance from the camera(Yaofeng Sun, J. H. L. Pang, Xunqing Shi, & J. W. R. Tew, 2006).

2.7.2 Stereo Vision

2.7.2.1 Introduction

SLAM (Simultaneous Localization and Mapping) in reduced visibility requires extraordinary consideration to perform depth calculations for collision avoidance. Keeping in view the ruggedness of environment;this operation has been performed using Stereo Vision Process.

Stereo vision is the process of recovering depth from camera images by comparing two or more views of the same scene. The output of this computation is a 3-D point cloud, where each 3-D point corresponds to a pixel in one of the images. Binocular stereo uses only two images, taken with cameras that were separated by a horizontal distance known as the "baseline". Calibrating the stereo camera system allows us to compute the 3-D world points in actual units, such as millimeters, relative to the cameras [3,4]. The data gathered using Stereo Vision for IMR (Intelligent Mobile Robot) to perform SLAM can be very rich for depth calculations.

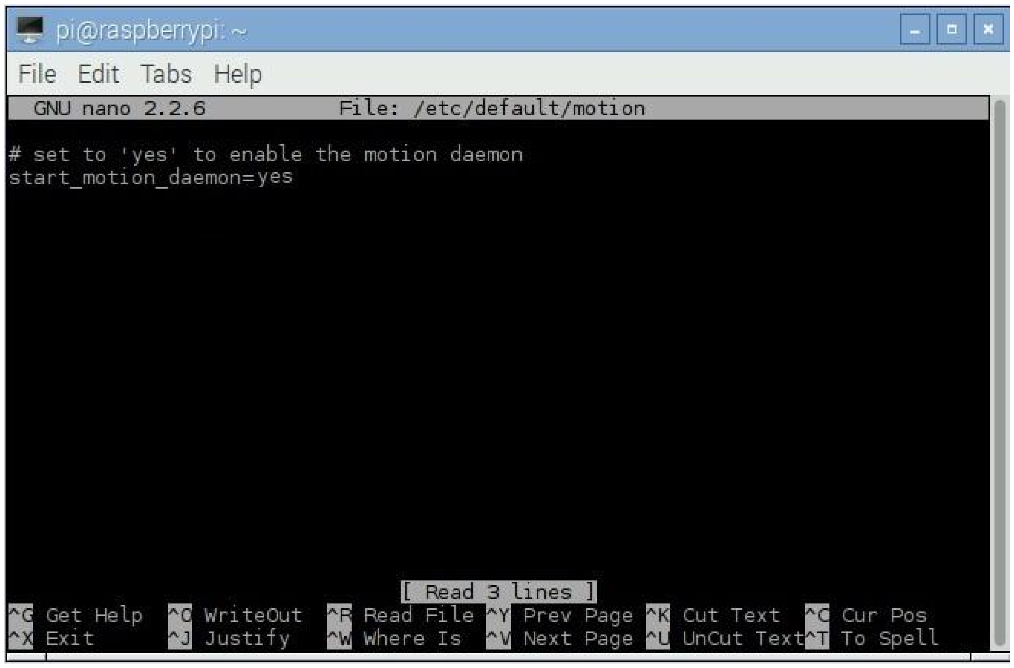
2.7.2.2 Hardware

To perform the Stereo Vision operation; following hardware has been used to capture and store the image:

- Raspberry Pi 2
- Logitech HD Webcam x 2

Figure 40 Stereo Camera and Raspberry Pi

The Raspberry Pi operates using OS Jessie image on SD card. The OS was then updated and upgraded. The “motion” package has also been installed to interface the webcams with the raspberry pi board. Once installed, the motion configuration file must be edited. The code for changing the motion file is shown in Annexure A.



```
  GNU nano 2.2.6            File: /etc/default/motion

# set to 'yes' to enable the motion daemon
start_motion_daemon=yes

                          [ Read 3 lines ]
^G Get Help   ^O WriteOut   ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Figure 41 Terminal of Raspberry Pi

The stereo images capture from Webcam is shown in Figure 43. Figure 44 shows the actual distance between 2 monitors and stereo cameras

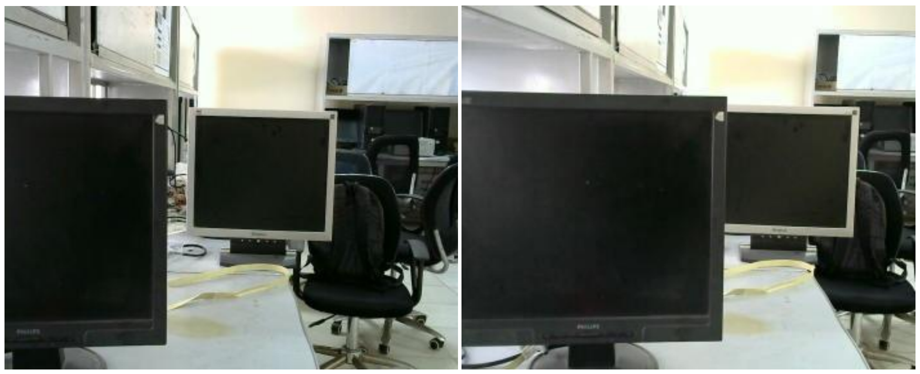


Figure 42 Stereo Images from Left and Right Camera

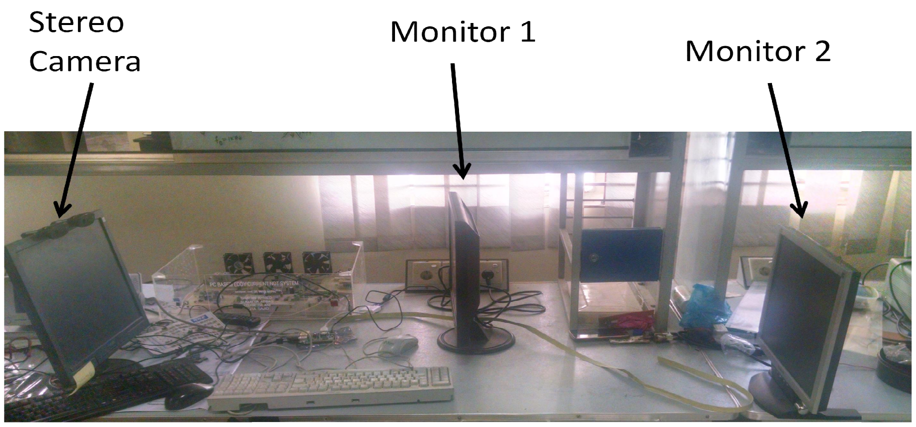


Figure 43 Side View of Stereo Image

2.7.2.3 Software Processing

MATLAB has been used to process the stereo images. The process consists of the following steps:

- Read Stereo Image Pair
- Collect Interest Points from Each Image
- Rectify Images

2.7.2.4 Read Stereo Image Pair

In the first step; MATLAB reads the two color images which were taken from different positions. Then, converts them to grayscale. Colors are not required for the matching process. Then, display a color composite demonstrating the pixel-wise differences between the images as shown in Fig.7

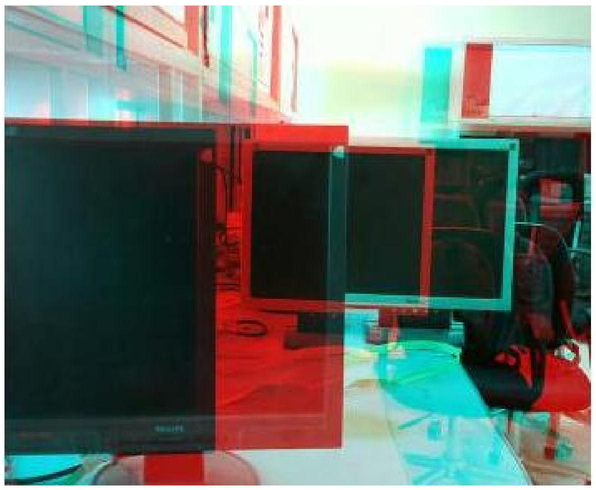
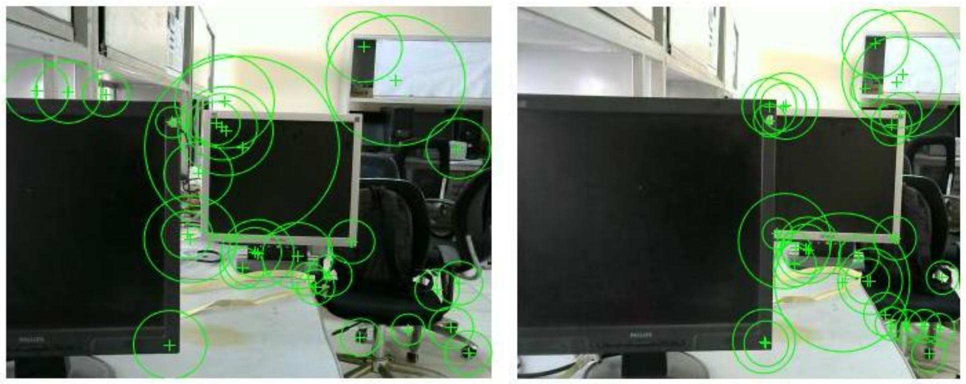


Figure 44 Composite Image

There is an obvious offset between the images in orientation and position. The goal of rectification is to transform the images, aligning them such that corresponding points will appear on the same rows in both images.

2.7.2.5 Collect Interest Points from Each Image

The rectification process requires a set of point correspondences between the two images. To generate these correspondences, we collected points of interest from both images, and then choose potential matches between them. All of the detected features can be matched because they were either not detected in both images or because some of them were not present in one of the images due to camera motion.



Surf Features in both images

2.7.2.6 Rectify Images

We used the function to compute the rectification transformations[5]. The function computes a projective transformation P1 for image1 and another projective transformation P2 for image2 in such a way that when using these two transformations (P1 and P2) on the corresponding images then the resulting images (R1 and R2) will be rectified in the sense that the corresponding epipolar lines will be in the same rows. The corresponding points will appear on the same rows as shown in Fig. 46

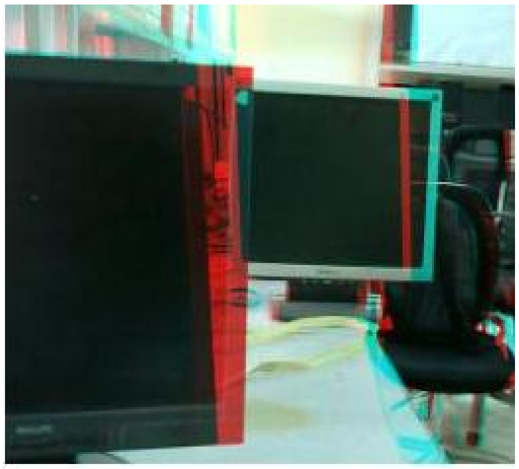


Figure 45 Rectified Image

2.7.2.7 Future Work

After the completion of depth calculation from stereo vision, the next step is to integrate PT100 and UT (ultrasonic sensor) with IMR. PT100 is used to measure the outer temperature while UT is attached for range finding.

2.7.3 SLAM

Making the mobile robot autonomous is realized through using the Simultaneous Localization and Mapping (SLAM) algorithms. The solution of SLAM is an active area of research from last many years in the field of robotics. Many solutions were proposed to solve the problems related to SLAM and it has been implemented in different scenarios from indoor robots to outdoor robots, underwater and airborne systems etc.

Determining the location of the Robot and its future movement is critical in the deployment of intelligent mobile robots (IMR's). The location of the robot in any environment can be determined by using the Global Positioning System (GPS) or by using the Inertial Measurement Unit (IMU). Second important thing for implementing SLAM is path planning. The task of finalizing and executing a sequence of motions which guides the robot to move from one place to another is called path planning. Different algorithms and approaches were used for path planning such as Graph Search Algorithms and the algorithms using the Dynamic Window Approach (DWA).

In our case, we will use a Particle Filter (PF) based SLAM. Particle filter is a more generalized version of Kalman Filter which is a non-linear filtering technique. The Posterior Density

Function (PDF) can be estimated by PF by extrapolating the prior PDF. However, PF uses Sequential Importance Sampling (SIS) to determine the next state.

Let x_t be the location of the robot at time instant *t* and o_t be the observations from the sensors such as thermal imaging sensor, stereo vision sensor, ultrasonic Sensor etc. Then the posterior PDF can be approximated through following equation:

$$p(x_t \mid o_t) \propto p(o_t \mid x_t) p(x_t \mid x_{t-1})$$

where $p(x_t \mid x_{t-1})$ is the prior PDF, $p(x_t \mid o_t)$ is the posterior PDF and $p(o_t \mid x_t)$ is the likelihood. Further details related to SLAM will be covered in the next report after finalizing the framework using the Particle Filter.

2.8 Acknowledgement

The fuzzy cooperative control strategy has been implemented by Mr. HussainAftab while working on his MS Thesis titled ***"Fuzzy Cooperative Control for a Team of Mobile Robots"*** under the supervision of Dr. Muhammad BilalKadri.

2.9 References

[1] Mehrjerdi, H., Saad, M., & Ghommam, J. (2011). Hierarchical Fuzzy Cooperative Control and Path Following for a Team of Mobile Robots. *IEEE/ASME Transactions on Mechatronics, 16*(5).

[2] Rezaee, H., & Abdollahi, F. (2014). A Decentralized Cooperative Control Scheme With Obstacle Avoidance for a Team of Mobile Robots. *IEEE Transactions on Industrial Electronics, 61*(1).

[3] Trucco, E; Verri, A. "Introductory Techniques for 3-D Computer Vision." Prentice Hall, 1998.

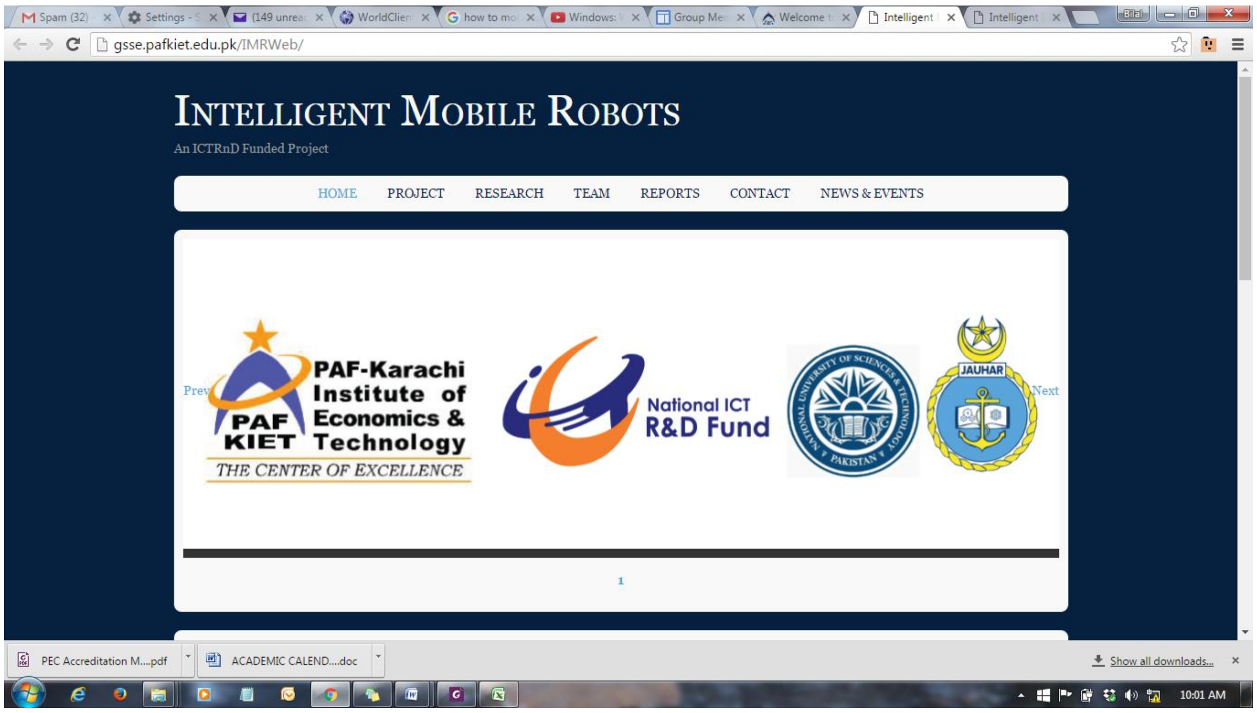
[4] Hartley, R; Zisserman, A. "Multiple View Geometry in Computer Vision." Cambridge University Press, 2003..

Chapter 3. Website

3.1 URL of the website

http://gsse.pafkiet.edu.pk/IMRWeb/

3.2 Screenshots of the website



Chapter 4. PAF-KIET's Hiring Policy

4.1 SOPS of Faculty Selection

1.1. Requirement for new faculty members have to be disseminated through emails and advertisements placed on the notice boards.

1.2. Evaluation of CVs for short listing received at different times.

1.3. Preliminary presentation: Presentation in front of class on a pre-given topic in presence of an evaluation committee. Candidates are evaluated on the basis of their knowledge and lecture delivery.

1.4. Final presentation: Candidates who pass the preliminary presentation present a topic of their interest in front of the Faculty of College of Engineering followed by a question answer session to analyze the depth of knowledge of candidate. This also serves the purpose of familiarizing session with the faculty.

1.5. Interview by Hiring and Selection Committee.

Chapter 5. Team Members

Principal Investigator

Dr. Muhammad Bilal Kadri (KIET)

Co-Principal Investigator

Dr. Tariq MairajRasool Khan (NUST-PNEC)

Research Assistants

1. ZaidPiwani (KIET)
2. NasirJumani (KIET)
3. SaadWaraich (NUST-PNEC)

PhD Students

1. Position vacant at KIET
2. MoezulHasan (NUST-PNEC)

MS Students

1. Sofia Yousuf (KIET)
2. UmairPathan (KIET)
3. Waleed bin Yousuf (NUST-PNEC)

Secretary

Adnan Ahmed (KIET)

Technician

Wajhatullah

Chapter 6. Software Development Platforms

All the following software development platforms have been acquired:

1. Matlab 2015a
2. Webots Robot Simulator Version 8.4.0
3. Robot Simulator Simulation V-REP Virtual Experimentation Platform
4. Arduino IDE
5. Raspbian for Raspberry PI

Chapter 7. ANNEXURE A

sudonano /etc/motion/motion.conf

webcam_localhost off:

thread /home/pi/webcam/cam1.conf

thread /home/pi/webcam/cam2.conf

where cam1.conf is the configuration file for the first webcam. In Raspberry PI directory /dev/video the motion configuration file must be coded in the following way:

(default /dev/video^*)

and by adding the following:

videodevice /dev/video0

videodevice /dev/video1

Once the configuration files have been created the following file shall be edited in order to enable motion to run:

sudonano /etc/default/motion

Now the live stream can be initiated using the following code:

sudo service motion start

Similarly to stop motion use:

sudo service motion stop

To capture the pictures from 2 cameras, as shown in Fig. 5, following code has been added:

fswebcam -r 1280×1024 -d /dev/video0 -l 10 test1.jpg

fswebcam -r 1280×1024 -d /dev/video1 -l 10 test2.jpg