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

Third Deliverable

Report submitted on 25th January 2017

PI: Dr. Muhammad Bilal Kadri

Co-PI: Dr. Tariq Mairaj Rasool Khan

Table of Contents

Chapter 1. Introduction

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

The deliverables to be submitted are listed below:

a) Image processing algorithm and its results in simulation.
b) Implementation issues in converting the image processing algorithm from Matlab script to Verilog/ VHDL code.
c) Details of the control algorithm and its results in simulation.
d) Simulation results of the Hardware in the loop (HIL) simulation.
e) A prototype of the IMR connected to the PC for simulation purpose.
f) Simulation results of the image processing algorithm specifically for fire and smoke
g) Demonstration of the prototype of IMR connected to PC will be given including simulation results of HIL, image processing algorithm for fire, smoke and integration of the FPGA & ARM processor testing results.
h) First Research paper to be submitted in International conference.
i) Bottlenecks faced in the development of the algorithms.

Due to the efficiency, low cost and easy availability of the PI boards we have not utilized FPGA boards. Hence, Verilog/VHDL code was not generated. All the milestones were successfully achieved.

Chapter 2. Control architecture for IMR and simulation results

This chapter will discuss the two-layered control architecture for the IMR. The low level control strategy i.e. the PID controller as well as the high level control scheme will be discussed. The IMRs are robots with several sub systems and controlling such a robot from a single perspective is not feasible. Hence different abstractions are created and utilized which help in figuring out solutions to the problems at hand while also providing a modular approach to focus on one part of the solution while assuming that the other parts will work.

2.1 Higher Level Abstraction

At the top most abstraction, the robots are assumed to be intelligent mobile units capable of moving around in a known or an unknown environment without the need of an operator to provide them with very specific movement instructions and even without the need of an operator to keep the robot from colliding into walls, another robot, humans or furniture nearby.

This abstraction is most useful as this assumed that the robot can move around in an autonomous manner without heeding any details to the inner workings of the robot. This abstraction level allows the development of algorithms which track the robot and plan its path while making sure that the costs (time to move, distance to move, battery usage, etc) are minimized and the goals (mapping of the location, determining fire locations, dousing of fire, etc) are achieved.

2.2 Robots as Potential Points

For our high level abstraction we consider the robots as charged points in a potential field. This technique is very robust and has been worked on by several different roboticists over the course of years however implementing this in hardware is very rarely reported and the implementations need much work as well. We studied this method to great extent and worked on to implement a journal paper “A Decentralized Cooperative Control Scheme with Obstacle Avoidance for a Team of Mobile Robots”, IEEE Transactions on industrial Electronics 2014. We have even went a step further and implemented the same technique in simulation of actual robot hardware and are also on track to implement the same in actual hardware created by us for the purpose of testing the algorithm and refining it so that this can be sue don’t he final IMR Robots.

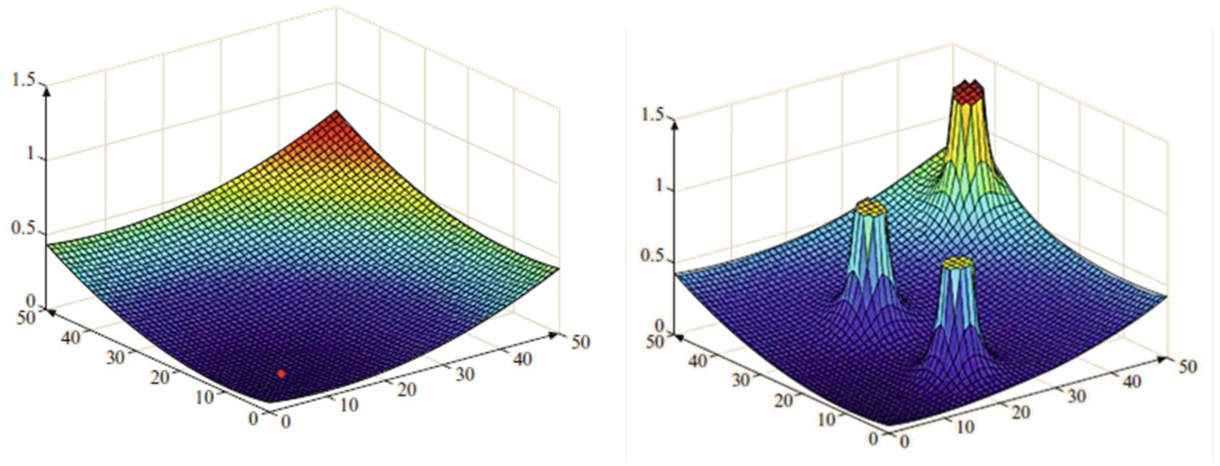


Figure 2.1: Charged Particle (robot) in a Potential Field with obstacles

When a robot is considered as charged particle in a potential field, one of the main concerns is for the robot to be stuck in a local minima.

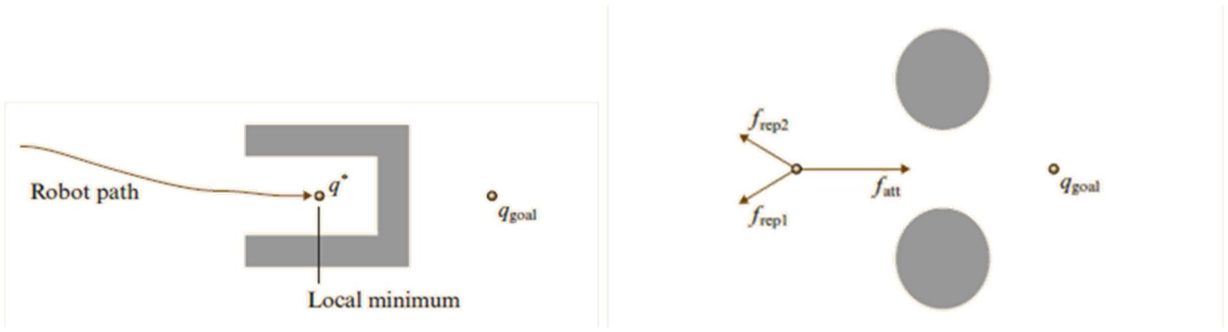
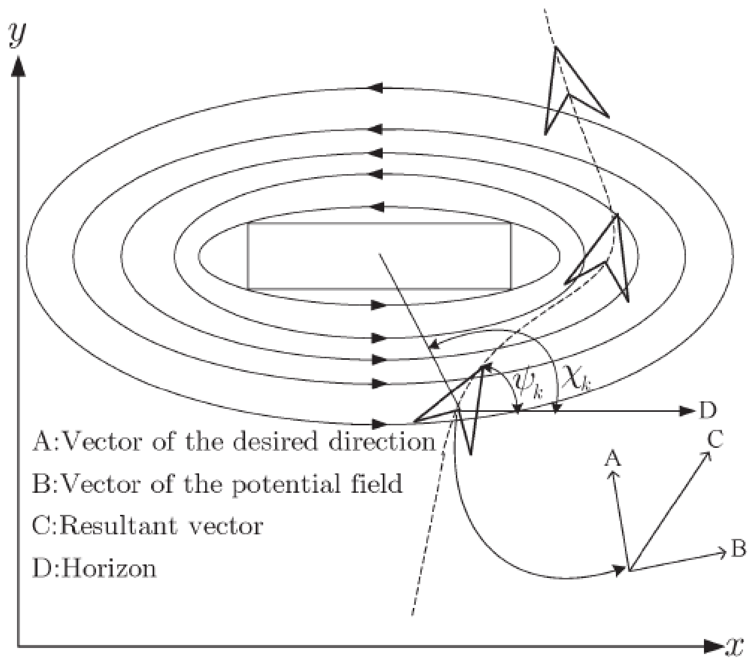


Figure 2.2: Local Minima in Potential fields

The solution proposed in the aforementioned journal paper to getting stuck in local minima is to utilize rotating vector fields which simply allow the robot (charged particle) to flow around any obstacle without getting stuck in one exact position as shown in figure 2.3.



y
x
ψ_k
χ_k
D
A
C
B
A:Vector of the desired direction
B:Vector of the potential field
C:Resultant vector
D:Horizon

Figure 2.3: Potential Field around an obstacle

A side benefit of considering a robot as a charged particle comes in the form when there are multiple robots and they are all considered as particles of same charge and then their movements are automatically formed into regular formations as the robots dynamically repel each other while the goal position or a virtual (central) leader attracts the robots. These movement patterns, formation and reformation is shown in the simulations results which were first done on point robots with dynamical equations as models of the robots and then in 3d physics based simulation environments which provide a much realistic view of the movement of robots.

2.3 Simulation Results

The following figures, 2.4 and 25 show how the robots behave when they first do a formation around a virtual leader and then follow along the virtual robot while avoiding obstacle son the way. The figure 2.5 also shows the interesting phenomenon when one robot leaves the formation for some reason (dead battery, collision, assignment of another task) the other robots in the formation reform themselves to form the polygon of the lower degree.

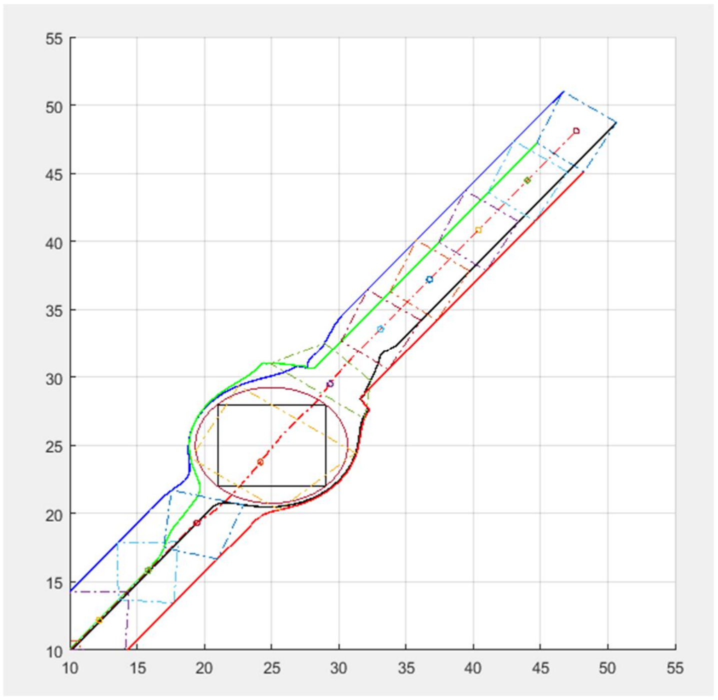


Figure 2.4: Collaborative obstacle avoidance and re-formation of robots

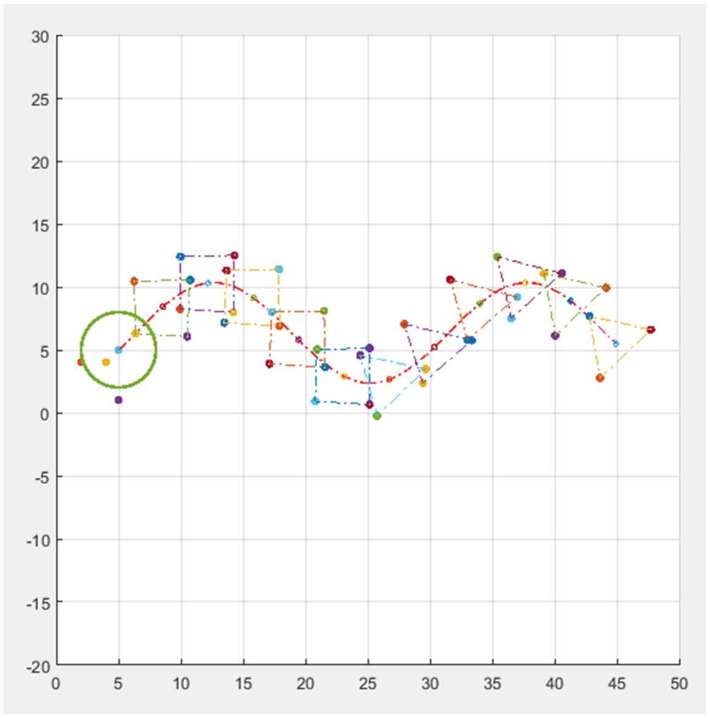


Figure 2.5: Re-formation of Robots when a robot exits formation

The virtual dynamic robots were then replaced by models of actual real world robots in a 3D physics based simulation environment which implements gravity, friction, wheel slippage, motor losses and other parameters usually ignored or considered zero. These simulations are done in a software environment called V-Rep which provides a free educational version or use in academia and also provides several accurate robot models and their representations.

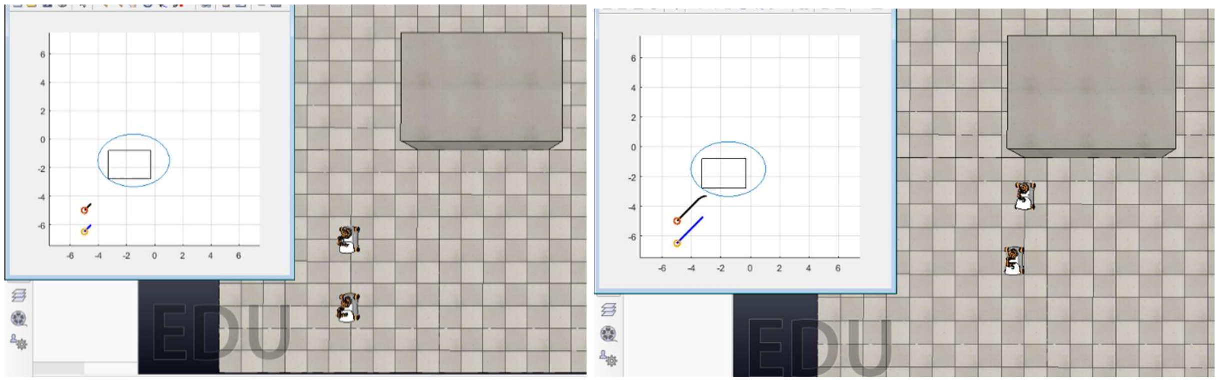
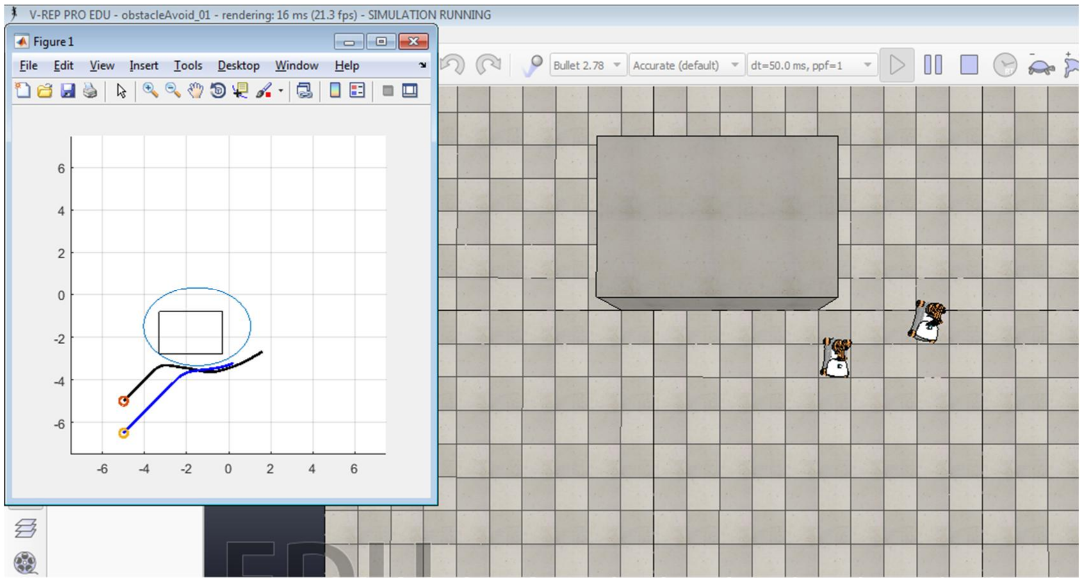


Figure 2.6: Robots moving towards an Obstacle


V-REP PRO EDU - obstacleAvoid_01 - rendering: 16 ms (21.3 fps) - SIMULATION RUNNING
Figure 1
File Edit View Insert Tools Desktop Window Help
Bullet 2.78
Accurate (default)
dt=50.0 ms, ppf=1
Figure 2.7: Crossing the obstacle while avoiding it

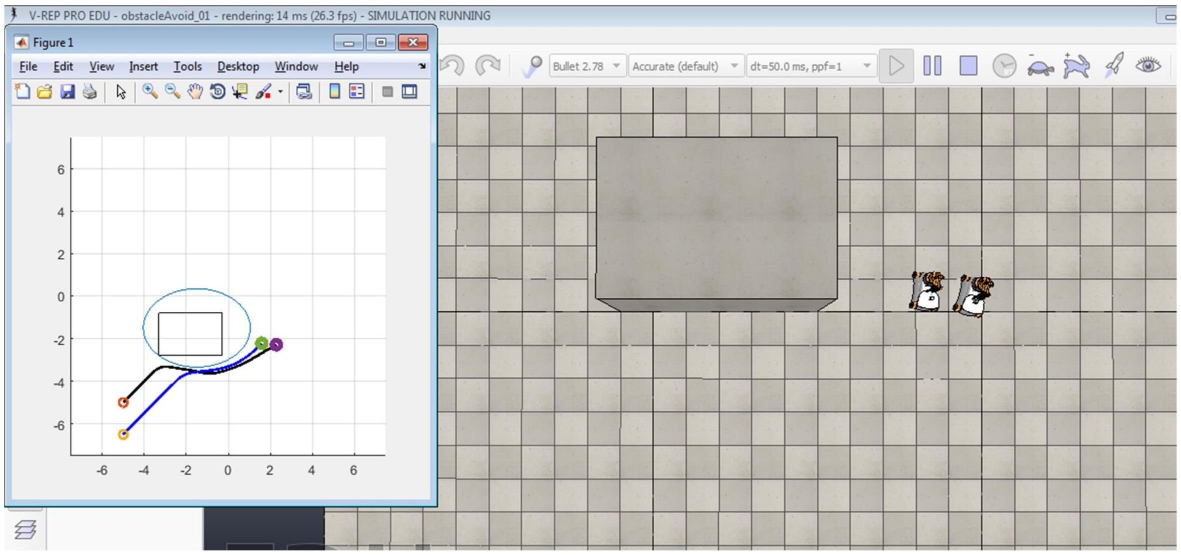

V-REP PRO EDU - obstacleAvoid_01 - rendering: 14 ms (26.3 fps) - SIMULATION RUNNING
Figure 1
File Edit View Insert Tools Desktop Window Help
Bullet 2.78
Accurate (default)
dt=50.0 ms, ppf=1

Figure 2.8: Obstacle Avoided and the robots stop – End of Simulation

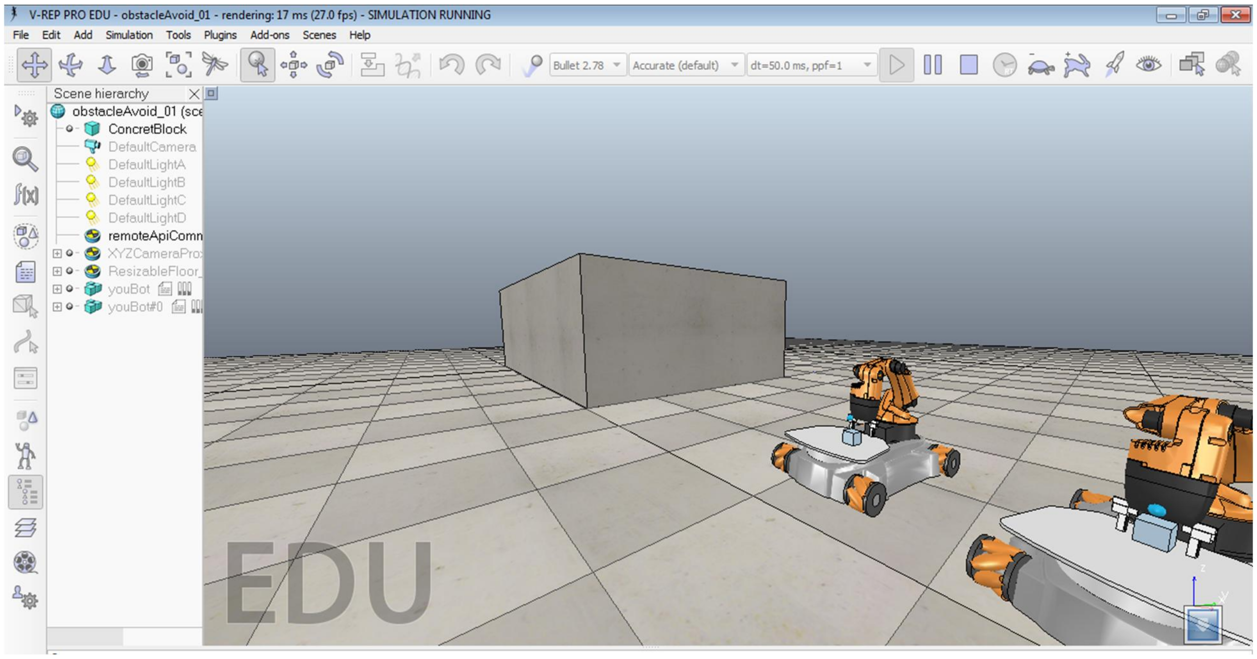


Figure 2.9: Close-up of the Simulation environment

2.4 Low Level Control

Once the High Level control gives commands to the robots to move, the low level control systems come into play and are then responsible for the actual movements of the robot while avoiding any sudden obstacles not foreseen by the high level layer.

The High Level abstraction may send the robot a point based data to move to a certain point, or send the robot a pair of linear and angular velocities or it may send the robot direct wheel velocities as determined by the high level control and as required by the algorithm. The flow of the data and the transformations applied are shown below in several block diagrams detailing the low level control methodologies.

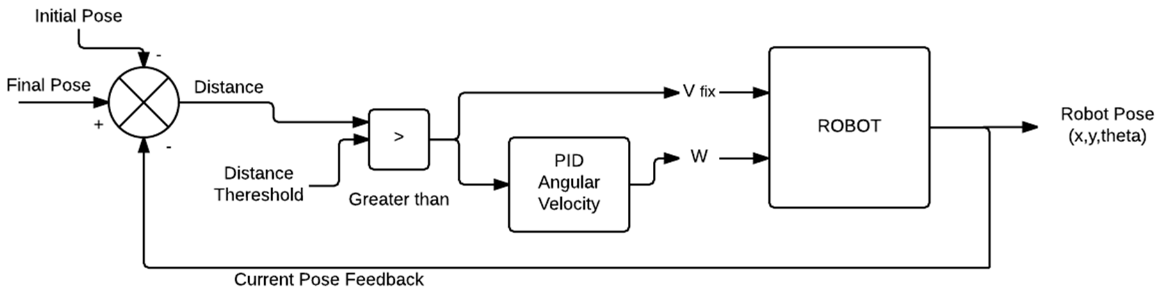


Figure 2.10: Initial Pose to Final Pose – Control Loop

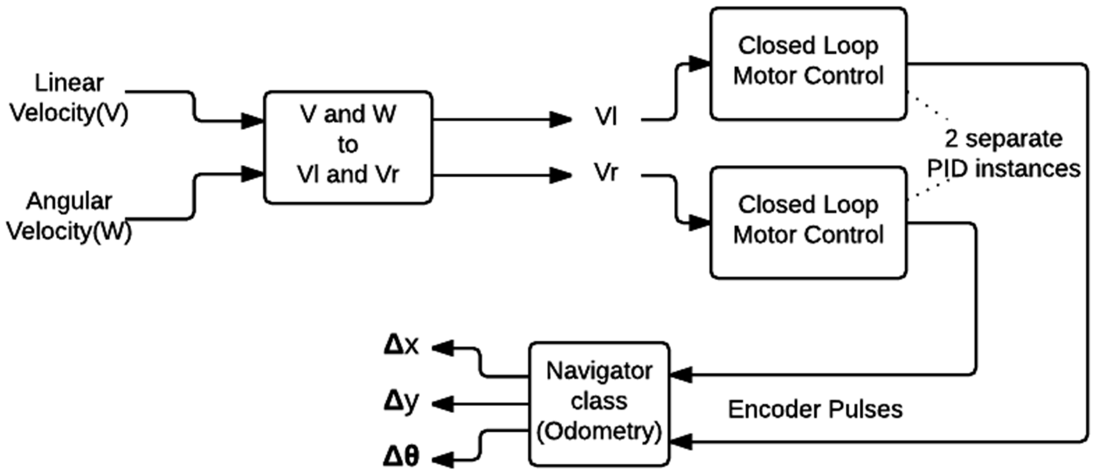


Figure 2.11: Linear/Angular Velocity pair to controlled Wheel velocities

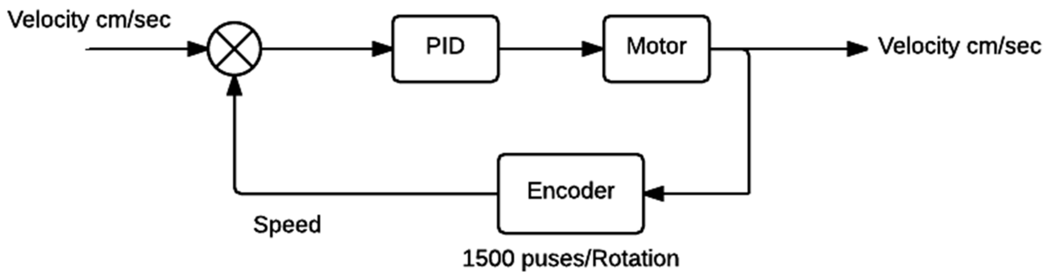


Figure 2.12: Motor Level Velocity PID Control loop

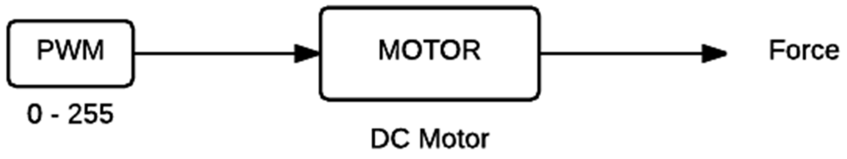


Figure 2.13: Open Loop motor Control – NOT USED, but implemented

The low level controllers are run at 16MHz 8-bit microcontrollers with emphasis to high speed operation and no delays, this results in smooth control of motor velocities and accurate estimation of robot pose, even though we are determining the pose of the robot only by means of tracking the encoder but we have achieved cm accuracy by eliminating and minimizing the sources of errors.

The different mechanisms of movement at the low level we have employed are provided below as mentioned, each method has its own benefits and loss.

2.4.1 Rotate Then Go

In rotate then go, the approach would be to first rotate the face towards the goal and then move towards the goal. That was initially implemented on a MUSAFIR. Since it's an inefficient algorithm due to the fact that even a very minute error in the heading of the robot will result in large displacement error between the robot and the final goal.

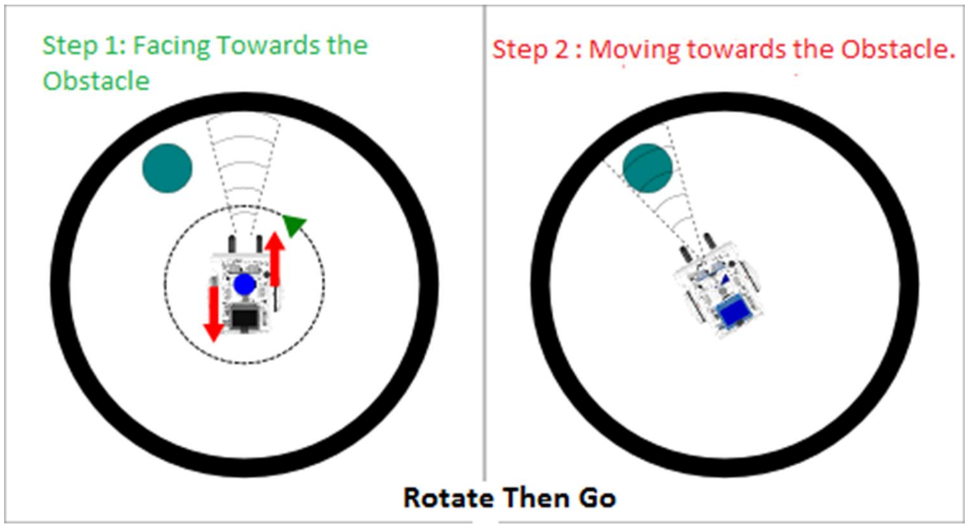

Step 1: Facing Towards the Obstacle
Step 2 : Moving towards the Obstacle.
Rotate Then Go

Figure 2.14: Rotate Then Go Method Procedure

2.4.2 Rotate while Movement

This is the better and a realistic approach to drive robot because it compensates in itself the hardware limitations. The angular linear velocities are directly converted into the left and right wheel velocities.

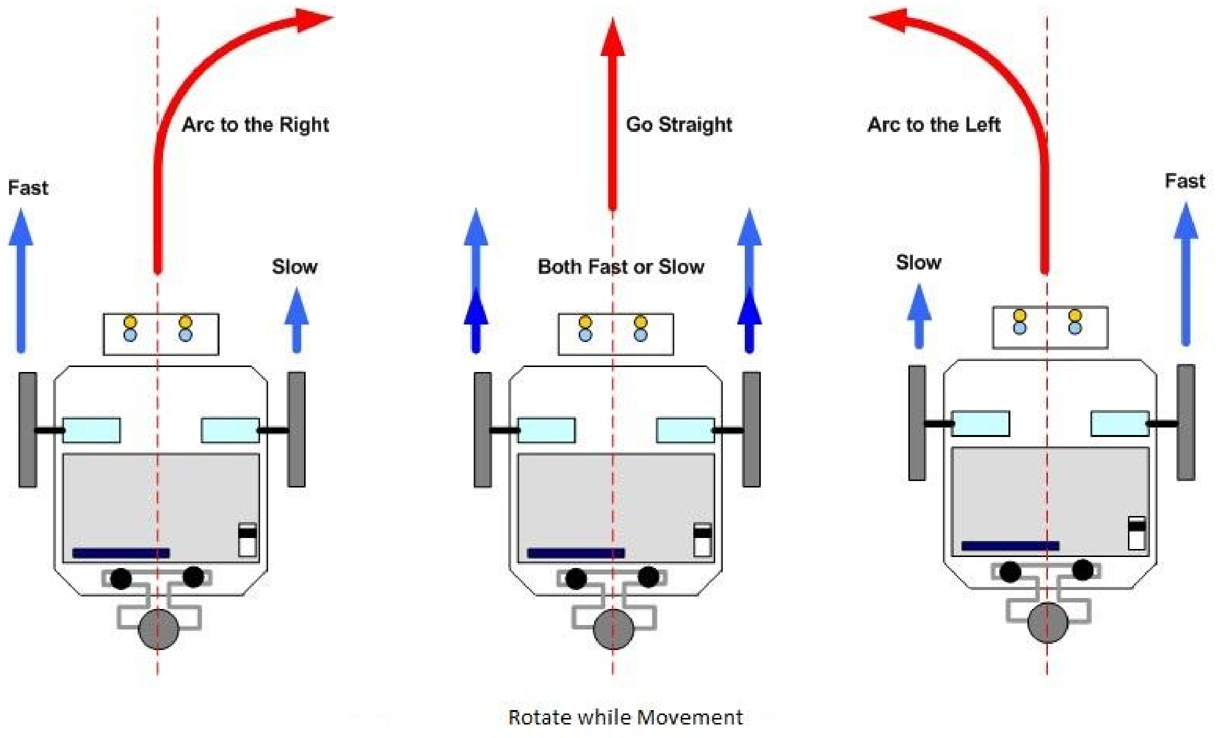


Figure 2.15: Rotate While Movement Method Procedure

2.5 Communication Link

The High Level control runs on computer systems running in the backend while the low level control runs on the robots themselves, these is need of a very reliable and fast communication medium between the two, this is done by means of 2.4GHz Digital Transceiver modules which allow for communication at 2Mbps with ACKs, CRC Error correction and auto retry. The robots have the modules attached to them however for the computers to be able to talk to the robots a TRASNPARENT Wireless Serial bridge is implemented using a USB to Serial Converter, a microcontroller and a 2.4GHz transceiver module. More about this communication link is discussed in the Hardware in the Loop chapter.

2.6 MATLAB as Controller

Complex dynamic systems for robots present challenges to control systems design. MATLAB is our primary control for the mobile robots. All the high level control algorithms are implemented in MATLAB and then the code is ported to either the onboard computers on the robot or on the onboard 8bit microcontrollers.

Chapter 3. Hardware in Loop testing of the IMR

3.1 Introduction and need for HIL

Upon completion of basic prototype of IMR Robot and also of multiple MUSAFIR Robots in working condition, we needed ways to adjust parameters of the robot during live working conditions, we also needed methods to keep track of the internal states of the robot while working and also be able to visualize and graph the relevant parameters. Hardware-In-the-Loop or HIL allows us to test different algorithms, parameters, techniques without modifying the hardware every time.

As our tests of the prototype algorithms and techniques are being tested on the lab based mobile platforms, MUSAFIR, we implemented a complete reporting and messaging system between the robots and the MATLAB server running the control algorithms.

3.2 HIL Block Diagram

The MUSAFIR Robots we developed have RF communication capability and integrating that with a computer which has MATLAB allows us to do HIL simulations and test parameters in real time.
The robots provide the MATLAB server their coordinates and current parameters and the MATLAB server then determines the velocities (linear and angular) of the robot or the velocities as according to the two wheels of the robot.

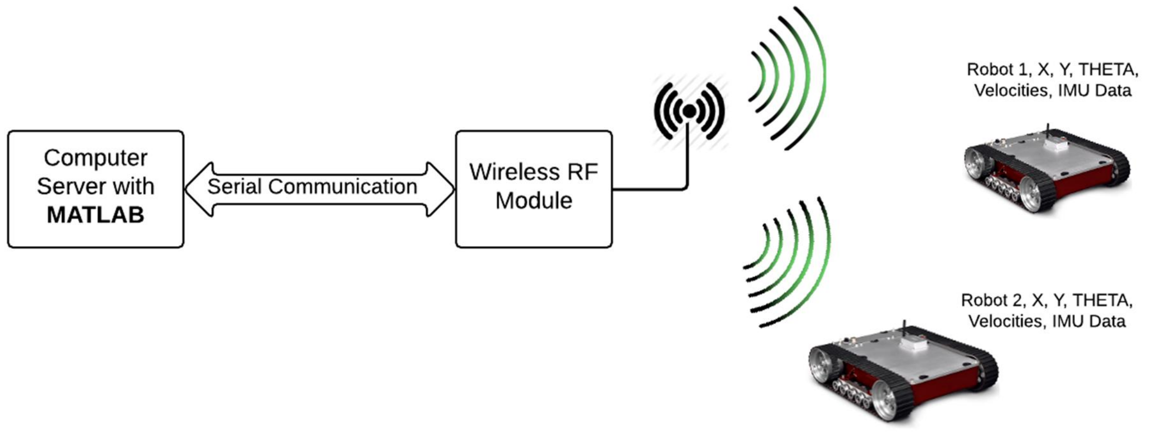

Computer Server with MATLAB
Serial Communication
Wireless RF Module
Robot 1, X, Y, THETA, Velocities, IMU Data
Robot 2, X, Y, THETA, Velocities, IMU Data

Fig 3.1: HIL Overall Block Diagram

3.3 MATLAB API for MUSAFIR

In order to have MATLAB work as robot controller in real time we created a complete monitor, command and control API of MUSAFIR Robots for MATLAB. This API allows for connecting to the MUSAFIR robots, controlling their parameters and monitoring/logging of the parameters for later visualizations and graphing.

The MATLAB API is composed of several commands and options which are provided below, as there may be multiple MUSAFIR Robots in operations at any given time, the commands have a unique identifier and all the responses also contain the same identifier and the command symbol.

MATLAB API Commands
SetVelocity(Serial object, RobotID, vl, vr)
SetPwm(Serial object, RobotID, PWM-Left, PWM-Right)
SetPose(Serial object, id , x , y , theta)
SetPid(Serial object,RobotID,P,I,D,1/2\n)
SetRobotData(data, RobotID)
GetPose(Serial object, RobotID)
GetPosition(Serial object, RobotID)
GetOrientation(Serial object, RobotID)
GetPid(Serial object, RobotID)
GetRobotData(RobotID)
EnablePose(Serial object, RobotID, Enable/Disable, SampleTime)
MusafirSerialInitialize(Com-port)
closePort(Serial object)
SetLinearVelocity(Serial object, RobotID, LinearVelocity)
SetAngularVelocity(Serial object, RobotID, AngularVelocity)

Table 3.1: MUSAFIR MATLAB API Commands List

3.3.1 SetVelocity

This sets the velocities of the Left and Right wheels of the robot.
Parameters: Serial object, Robot ID, Left Wheel velocity (in cm/sec), Right Wheel Velocity (in cm/sec)
Return: ‘1’ if job done, ‘-1’ in case of any error.

3.3.2 SetPwm

This Sets the PWM of the Left and Right wheels of the robot.

Parameters: Serial object, Robot ID, Left Wheel PWM, Right Wheel PWM
Return: ‘1’ if job done, ‘-1’ in case of any error.

3.3.3 SetPose

This Reset the pose of the Robot as desired by higher level.
Parameter: Serial object, Robot ID, x-coordinate, y-coordinate, Pose
Return: ‘1’ if job done, ‘-1’ in case of any error.

3.3.4 SetPid

This command set the PID parameter of the Right or left wheel of the robot.
Parameter: Serial object, Robot ID, kp, ki, kd, Left\Right Motor
Return: ‘1’ if job done, ‘-1’ in case of any error.

3.3.5 SetRobotData

This command will call back by Matlab when the broadcasted pose is received.
Parameter: Data, RobotID

3.3.6 GetPose

This command is a superset of GetPosition and GetOrientation command. It will call out both of them sequentially.
Parameter: Serial object, RobotID
Return: x-coordinate, y-coordinate, Robot Orientation.

3.3.7 GetPosition

Command used to get the Position of robot in Cartesian coordinates.
Parameter: Serial object, RobotID
Return: x-coordinate, y-coordinate.

3.3.8 GetOrientation

Command used to get the Position of robot in Cartesian coordinates.
Parameter: Serial object, RobotID
Return: Robot Orientation.

3.3.9 GetRobotData

This command is used to get the stored broadcasted data packet from in MATLAB from MUSAFIR.
Parameter: RobotID
Return: DataPacket (packet time, x-coordinate, y-coordinate, Robot Orientation.)

3.3.10 EnablePose

To enable the MUSAFIR to broadcast its data to MATLAB with a certain frequency.
Parameter: Serial object, RobotID, Enable/Disable, SampleTime(Duration)
Return: ‘1’ if job done, ‘-1’ in case of any error.

3.3.11 MusafirSerialInitialize

Initialize the Com-port for serial communication.
Parameter: Com-port.
Return: Serial object.

3.3.12 closePort

To close the port of Serial communication.
Parameter: Serial object
Return: '1' if job done, '-1' in case of any error.

3.3.13 SetLinearVelocity

This command set the linear velocity of the MUSAFIR
Parameter: Serial object, RobotID, LinearVelocity
Return: '1' if job done, '-1' in case of any error.

3.3.14 SetAngularVelocity

This command set the angular velocity of the MUSAFIR
Parameter: Serial object, RobotID, AngularVelocity
Return: Return: Return: Return: '1' if job done, '-1' in case of any error.

3.4 MUSAFIR MATLAB API Limitations

As the MUSAFIR MATLAB API was completely written from scratch, it was soon realized that there are several places that can be made better and several complications arose with respect to time stamping of the data, race conditions among incoming messages and acknowledgement of the transmitted and received data among other things. This led the team to look for and research into other avenues and work done by other people in similar regards. Multiple sources and Labs working on similar work outside the country have proposed using ROS as the framework of choice for any such robotics work and hence the team is now looking to implement ROS in its day to day workings.

3.5 ROS Introduction

ROS or Robot Operating System is in contrast to its name, NOT an operating system per say but a collection of tools, libraries, parsers, loggers, data visualizers, algorithms and hardware which allow research teams across the world to collaborate on Robotics and to benefit from the work done by others and to share their work as well for further inspection, testing and analysis by the international community.

Fig 3.2: The ROS Equation

ROS is a framework for writing software for Robotic Systems, involving mapping, navigation, imaging sensors, collaborative control and manual teleoperation modules to name a few. A basic block diagram of ROS Architecture for a distributed system of robots is given in figure 3.3.

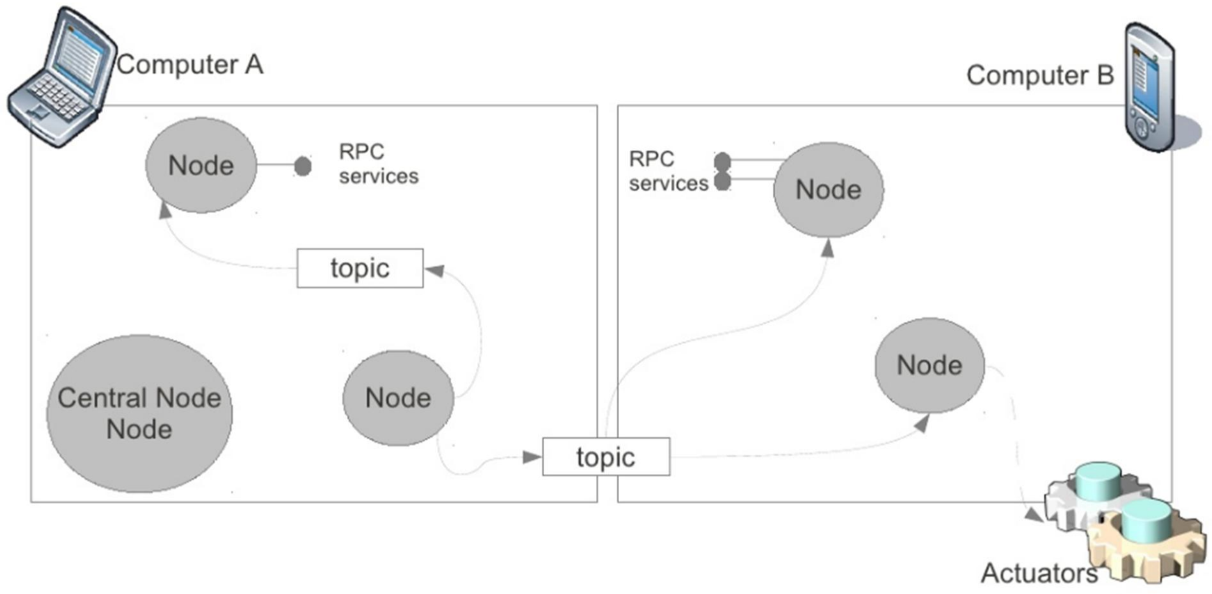


Fig 3.3: Distributed Robotics Architecture with ROS

3.6 Conversion of MUSAFIR APIs to ROS

We have already started the work on conversion of our MUSAFIR APIs (C/C++ and MATLAB) to ROS, and have also started the work on learning how ROS works so that we integrate it into our work. ROS Mapping packages and Navigation Algorithms are of particular interest and if we can completely port all our code and hardware to be ROS compatible, we will be able to utilize the vast array of contributed code resulting in faster prototyping and development at least in the software side, giving us time to work on the hardware more.

3.7 Implementation of ROS Driver Layer

ROS supports specific hardware and to be able to implement ROS packages and nodes on a robot designed and built from scratch, ROS requires drivers and a communication layer which will transform the data coming from the robot to structure and types supported by ROS and vice versa. To implement this driver layer, we first need to get a basic understanding of ROS and also to get understanding of some Linux related systems and software which we are already working on.

As our robots have an onboard computer, Raspberry Pi 3, which is able to run Linux OS and is quite capable in terms of power, performance and storage, given its size and cost, our goal is to port and run ROS on Raspberry Pi, as ROS is by default supported only with Ubuntu Linux machines, we plan to either re-compile ROS for the Raspbian operating system which runs on the Raspberry Pi or utilize one of the un-official Ubuntu releases which work on the ARM7 processor which is used in the Raspberry Pi.

Chapter 4. Image processing algorithms implemented for IMR

Image processing results for stereo vision, UT sensor, LIDAR and FLIR will be discussed. The main objective is to help the IMR navigate in an unknown environment.

4.1 Stereo Vision

4.1.1 Experimental Setup

In this work, the actual stereo cameras were not used. We use two different cameras of same specifications separated by a baseline distance B as shown in fig. 1.

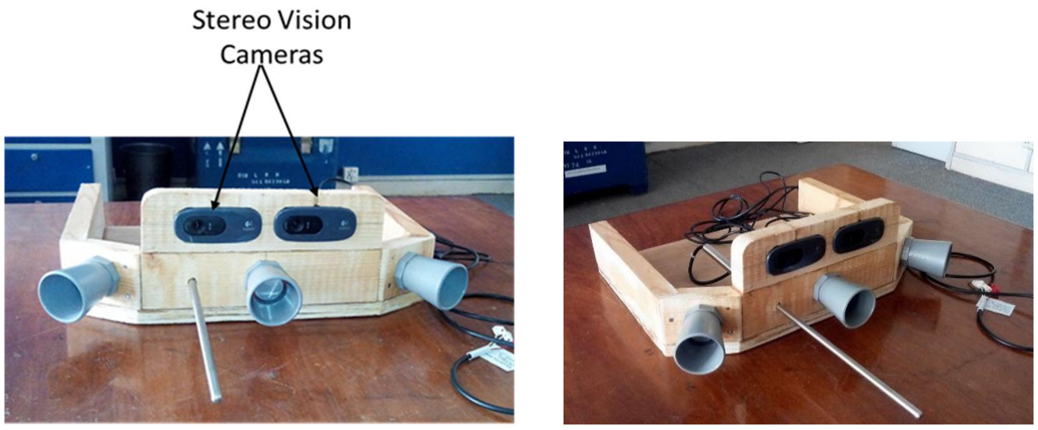


Fig. 1. Cameras separated by a baseline distance B

The images are captured from the camera. They are then converted into grayscale for proper matching. Left and right images from left and right camera respectively are used to determine the disparity between the images. This disparity is determined by using the correlation and block matching mechanism as shown in fig. 2

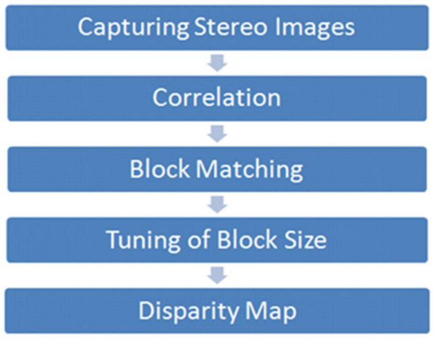


Fig. 2. Process flow

The images taken by the left and right camera are shown in fig. 3. It is clearly visible that these are images are shifted horizontally because of the baseline distance (B) between the cameras. The disparity between these images can be calculated using this horizontal shift. The next step is to perform the basic block matching. For every pixel in the right image, we extract the 7-by-7-pixel block around it and search along the same row in the left image for the block that best matches it. The basic block matching does well, as the correct shape of the stereo scene is recovered. However, there are noisy patches and bad depth estimates everywhere, especially on the ceiling. These are caused when no strong image features appear inside of the 7-by-7-pixel windows being compared. Then the matching process is subject to noise since each pixel chooses its disparity independently of all the other pixels. A noisy disparity image can be improved by introducing a smoothness constraint using dynamic programming.

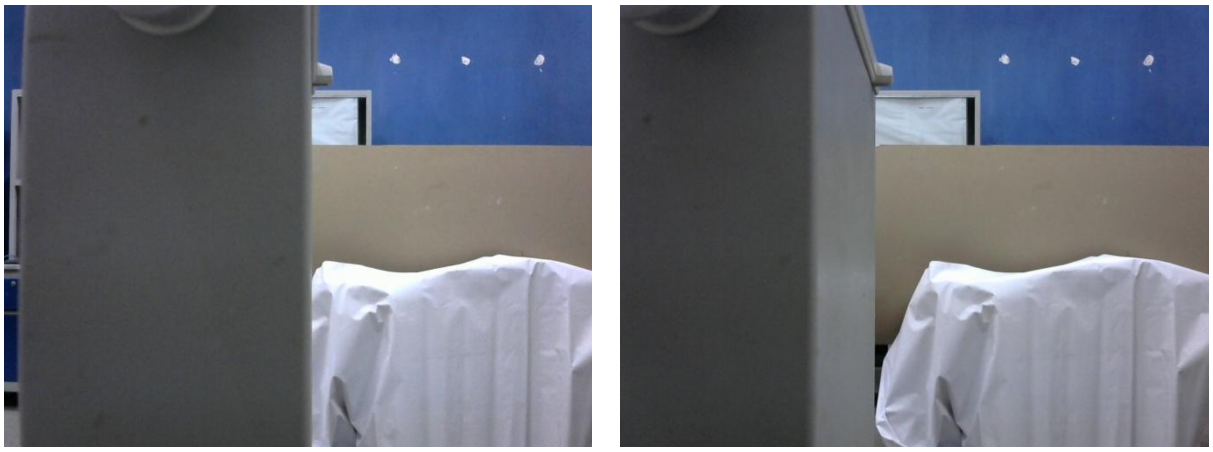


Fig. 5. Left and Right images for Depth Estimation

The depth (Z) from the image pair can be estimated using (1)

$$Z = \frac{fB}{x1 + x2} \qquad (1)$$

where f is the focal length of the lens used, B is the baseline distance between the two cameras, (x1+x2) is the disparity as shown in fig. 4.

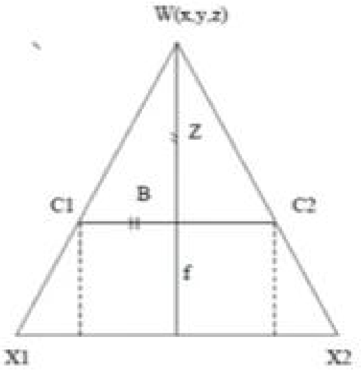


Figure 4 Triangulation

The left and right images are compared using the Sum of Square Difference (SSD) between each pixel of the window as in (2).

$$SSD = \sum (I_{left} - I_{right})^2 \qquad (2)$$

Where Ileft and Iright are the left and right images respectively

4.1.2 Results

The images available in the database of Raspberry pi shown in fig. 5 and their corresponding depth image is shown in fig.6.

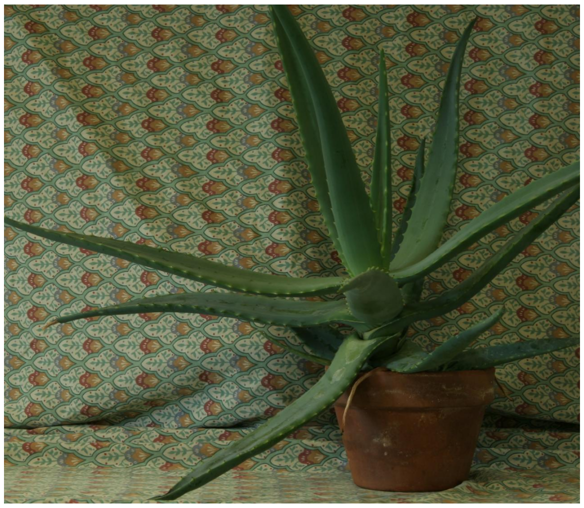


Fig. 5. Image of an Aloe Vera

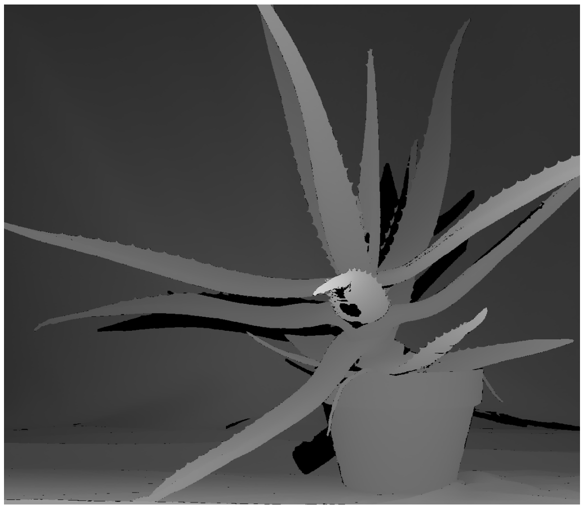


Fig. 6. The depth image of an Aloe Vera image

The color intensities in fig. 6 are showing the depth information. The white color is showing that the object is near the camera whereas the black color is showing that the object is away from the camera.

The test image taken in the lab is shown in fig. 7 and there corresponding depth image is shown in fig. 8.

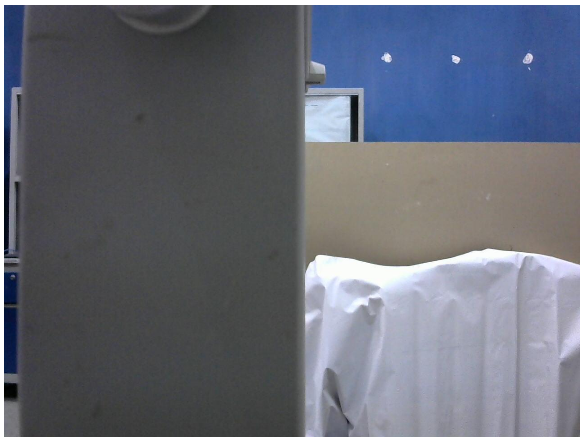


Fig. 9 Test image taken in the lab

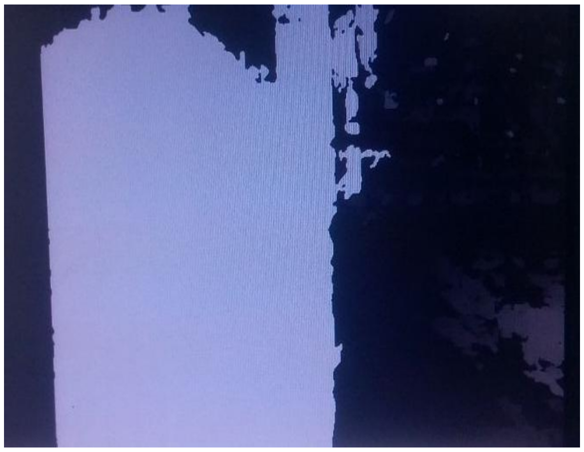


Fig. 10 Depth image of the test image of figure 9

4.1.3 Live Video Capturing

After successful image processing on the images which we captured from both VGA cameras serially, we then captured live video from both the VGA cameras separately so that we can reduce the processing time of the stereo depth perception.

The Python code for live video capturing from the VGA camera is attached in **Annexure-A.**

4.1.4 Two VGA Cameras on single Pi board

Two VGA cameras were then run on a single raspberry pi and the frames from both these cameras were then captured serially. The delay between the frames captured from both the cameras was 3mSec.

The Python code for running two VGA cameras on single raspberry pi board and capturing frames from those VGA cameras is attached in **Annexure-B.**

4.1.5 Real time Stereo Depth Perception

Instead of saving those captured frames into the memory of raspberry pi, we used those frames directly into depth perception code. So that we can increase the processing speed and can obtain the maximum frames possible in least possible time.

The Python code for Stereo Depth perception is attached in **Annexure-C.**

4.1.6 Results of Real Time Stereo Vision

The results of stereo depth perception are as under,

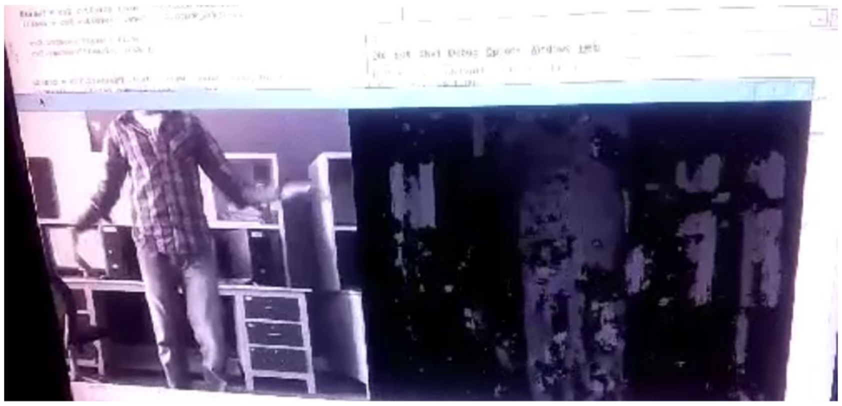


Fig 11(a): Real time Stereo Results

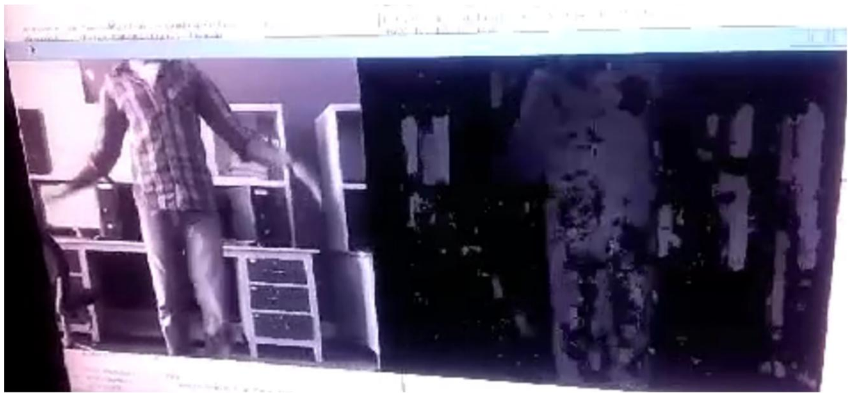


Fig 11(b): Real time Stereo Results

4.1.7 Future Goals

- Calibration of Stereo Results
- Try to achieve fastest possible stereo processing

4.2 UT Sensor (MaxSonar Sensor)

The HRXL-MaxSonar-WR ultrasonic sensors are in-air, non-contact object detection and ranging sensors that detect object within an area. These sensors are not affected by the color or other visual characteristics of the detect object. Ultrasonic sensors use high frequency sound to detect and localize objects in variety of environments. Ultrasonic sensors measure the time of flight for sound that has been transmitted to and reflected back from nearby objects. Based upon the time of flight, the sensor outputs a range reading.

It provides high accuracy and high resolution ultrasonic proximity detection and ranging in air. This sensor line features 1-mm resolution, target-size and operating voltage compensation for improved accuracy, superior rejection of outside noise sources and internal speed of sound temperature compensation and optional external speed of sound temperature compensation. This ultrasonic sensor detects object from 1-mm and ranges to objects from 30-cm to maximum range. Objects closer than

30-cm are typically reported as 3-cm. the interface output formats are pulse width, analog voltage and digital serial in either RS232 or TTL.

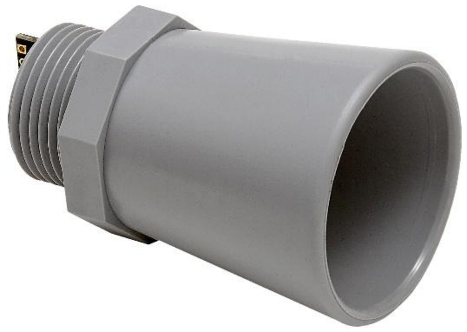


Fig 4.1: HRXL-MaxSonar sensor

4.2.1 Range Sensing Precision

- Determines range to the first detectable object
- Excellent clutter rejection
- Internal temperature compensation is standard and optional external temperature compensation
- Provides typical accuracy of 1% or better
- Typical reading-to-reading stability of 1mm at 1-meter

4.2.2 Low Power Requirement

- Fast first reading after power-up eases battery requirements
- Wide, low voltage supply requirements eases battery powered design
- Low current draw reduces current drawn for battery operation
- Very low power range finder, excellent for multiple sensor or battery based systems

4.2.3 HRXL-MaxSonar sensor pin out

Pin 1-Temperature Sensor Connection: Leave this pin unconnected if an external temperature sensor is not used. For best accuracy, this pin is optionally connected to the HR-MaxTemp temperature sensor.

Pin 2-Pulse Width Output: This pin outputs a pulse width representation of the distance with a scale factor of 1uS per mm. The pulse output is sent with a value within 0.5% of the serial output.

Pin 3-Analog Voltage Output: This pin outputs a single ended analog voltage scaled representation of the distance. This output is referenced to the sensor ground and Vcc. After the ~50mSpower up initialization, the voltage on this pin is set to a low voltage. Once sensor has completed a range reading the voltage on this pin is set to the voltage to the corresponding to the latest measured distance.

Pin 4-Ranging Start/Stop: When pin 4 is low and then brought high, the sensor will operate in real time and the first reading output will be the range measured from the first commanded range reading. When the sensor tracks the RX pin is low after each range reading. And then RX pin is brought high, unfiltered real time range information can be obtained.

Pin 5-Serial Output: Outputs Serial data

Pin 6-V+: Positive Power, Vcc: The sensor operates on voltages from 2.7V – 5.5V DC. For best operation, the sensor requires that the DC power be free from electrical noise.

Pin 7-GND: Sensor ground pin: DC return, and circuit common ground

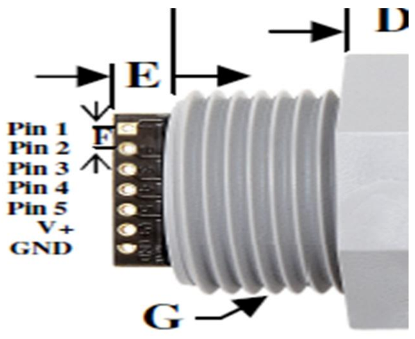


Fig 4.2: MaxSonar pin out

4.2.4 Auto Calibration

Each time a HRXL-MaxSonar-WR series sensor takes a range reading, it calibrates itself. The sensor then uses this data to range objects. If the temperature, humidity or applied voltage changes during sensor operation, the sensor will continue to function normally over the rated temperature range while applying compensation for changes caused by temperature and voltage.

4.2.5 Independent Sensor Operation

The HRXL-MaxSonar-WR sensors have the capability to operate independently when the user desires. When using the HRXL-MaxSonar-WR sensors in single or independent sensor operation, it is easiest to allow the sensor to free-run. Free-run is the default mode of operation for all of the MaxBotix Inc., sensors. The HRXL-MaxSonar-WR sensors have three separate outputs that update the range data simultaneously: Analog Voltage, Pulse Width and Serial Data. Below are diagrams on how to connect the sensor for each of the three outputs for single or independent sensor operation.

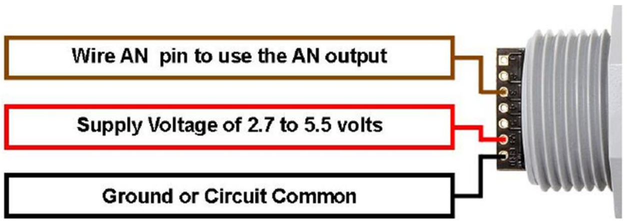

Wire AN pin to use the AN output
Supply Voltage of 2.7 to 5.5 volts
Ground or Circuit Common

Fig 4.3: Analog output sensor operation

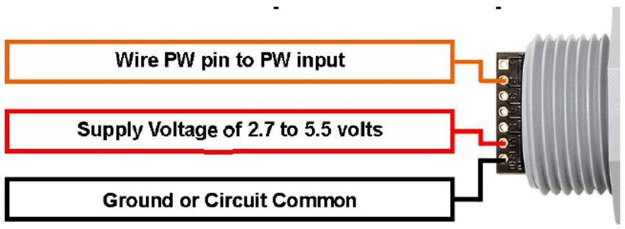


Fig 4.4: Pulse width Sensor Operation

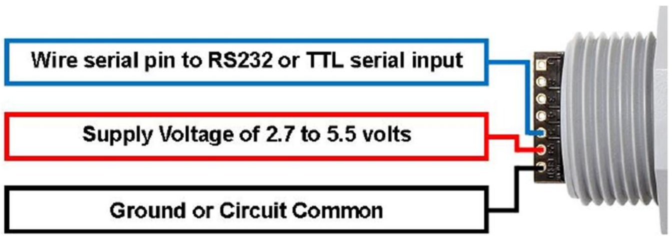

Wire serial pin to RS232 or TTL serial input
Supply Voltage of 2.7 to 5.5 volts
Ground or Circuit Common

Fig 4.5: Serial Output Sensor Operation

4.2.6 Interface with Raspberry pi

MaxSonar sensor is interfaced with raspberry pi using Pulse Width output (pin 2). Detail of pin connection is as under:

Pin-2 of MaxSonar sensor to **Pin-18** of raspberry pi

Pin-4 of MaxSonar sensor to **Pin-16** of raspberry pi

Pin-6 of MaxSonar sensor to **Pin-1** of raspberry pi

Pin-7 of MaxSonar sensor to **Pin-6** of raspberry pi

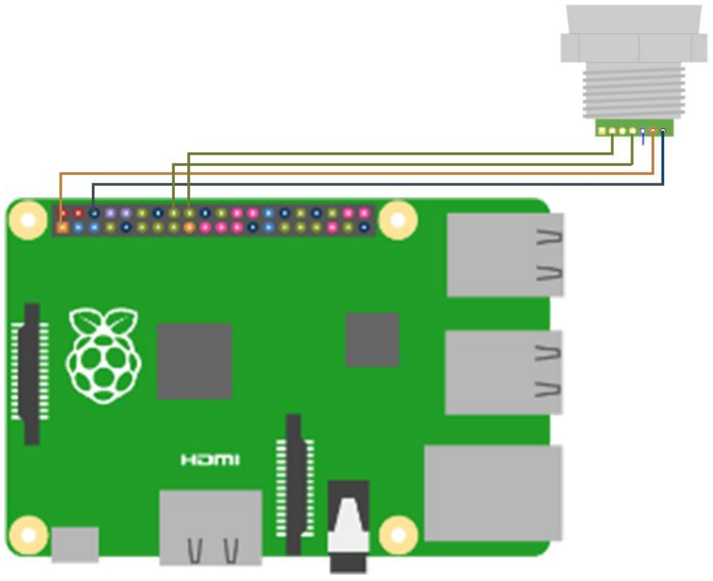
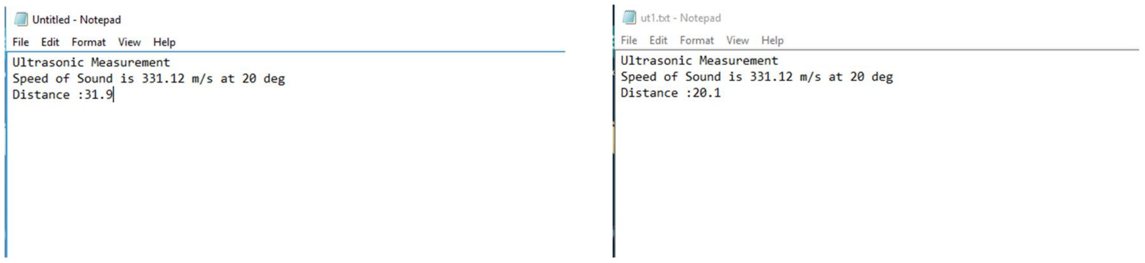


Fig 4.6: MaxSonar interface with raspberry pi

The Python code used for MaxSonar sensor is attached in **Annexure-D.**

4.2.7 Results from MaxSonar sensor

After successful interface of MaxSonar sensor with raspberry pi, the reults which we achieved were as in fig 4.7



```
Ultrasonic Measurement
Speed of Sound is 331.12 m/s at 20 deg
Distance :31.9
```

```
Ultrasonic Measurement
Speed of Sound is 331.12 m/s at 20 deg
Distance :20.1
```

Fig 4.7: Results from MaxSonar sensor

These results were quite accurate when compared to the actual distance from the obstacles.

4.2.8 Map Genration based on MaxSonar sensor

MaxSonar sensor mounted on robot face was then rotated over an angle of 180˚ from 0˚ to 180˚ with step size of 10˚ to generate initial maps of the environment.

The actual image of the environment is as shown

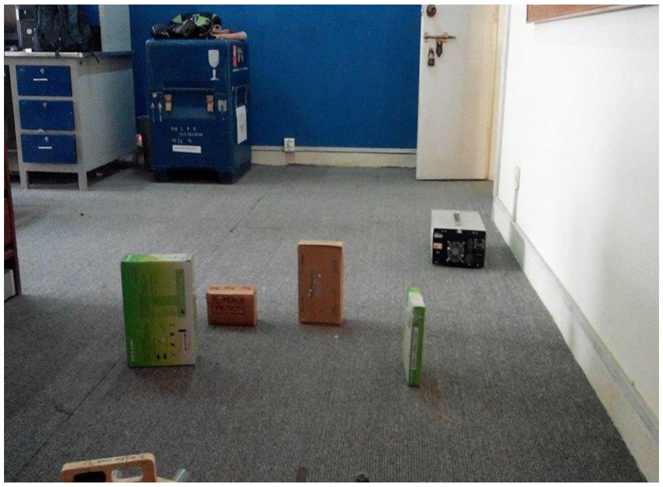


Fig 4.8: MaxSonar Testing Environment

The Map which we generated based on the readings from MaxSonar sensor at its initial position was as

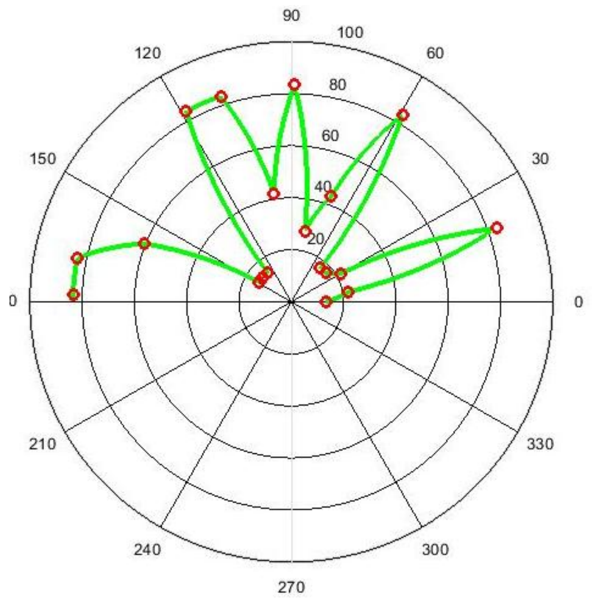


Fig 4.9: Initial Map based on UT sensor

The robot was then moved 6 inch further and a new set of readings were taken from 0˚ to 180˚ with step size of 10˚. The Map which we got this time was as shown under

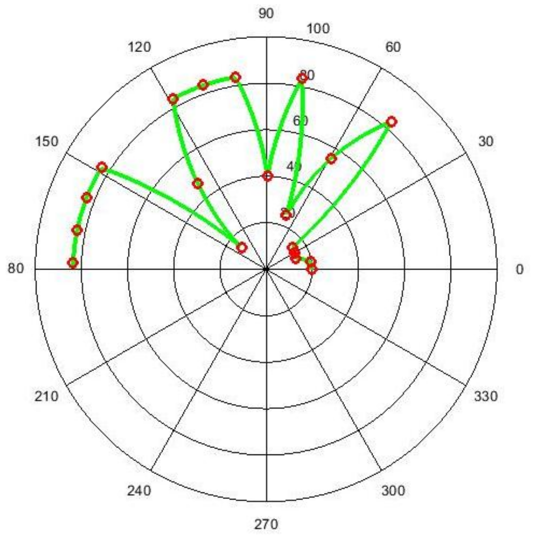


Fig 4.10: Map at Position 1

The same procedure was repeated at four unique positions of the robot and the results which we got are as shown

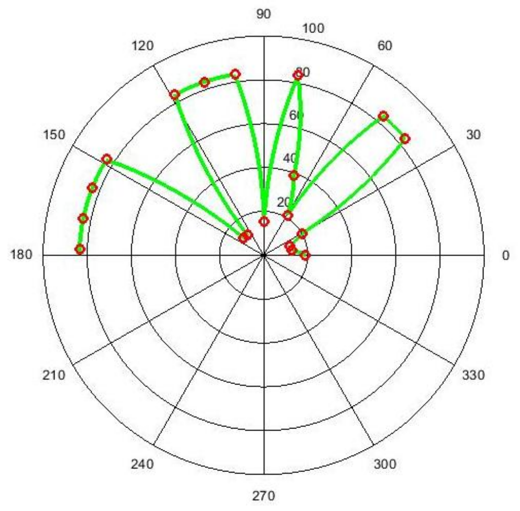


Fig 4.11: Map at Position 2

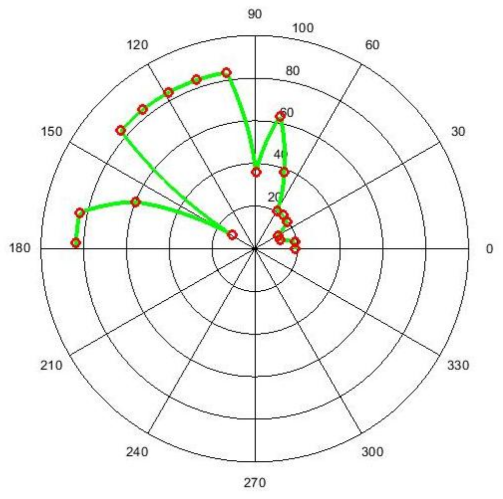


Fig 4.12: Map at Position 3

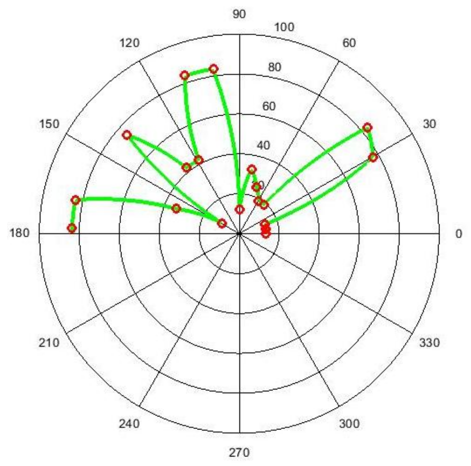


Fig 4.13: Map at Position 4

4.3 Flir Thermal Imaging Camera

The thermal imaging camera has the sensitivity of <50mk, which means it can detect the slightest change in outside temperature. This thermal imaging camera is integrated on the raspberry pi, which is mounted on the IMR. This will detect images of various scenarios in front of the camera. The images will be processed to take out the norm, taking two images at a time and comparing them. The one with the highest norm will enable the robot to move towards the hotter region from its current position. High norm mean that the image contains higher temperature region. This thermal imaging camera will thus tell the robot where the highest temperature region is and robot will move towards that.

4.3.1 Progress with the sensor

Our progress with this sensor is that we have changed the already given code of thermal imaging camera to work with our conditions. Now the sensor can take images as per the user desire or predefined delay in the code. Currently, we have used the delay of two seconds in our code of thermal imaging camera. So that our camera takes imaging after every two seconds in its directory where the images are exported to take out the norm. We have successfully written the code in python for taking out the norm from the images captured by Flir thermal imaging camera. Which is attached in **Annexure-E.**

Based on these norm values and the distance values which we got from MaxSonar sensor, a 3D map of the environment was generated which tells the temperature of different regions in the environment around the robot.

Chapter 5. Demonstration of prototype IMR

5.1 Introduction

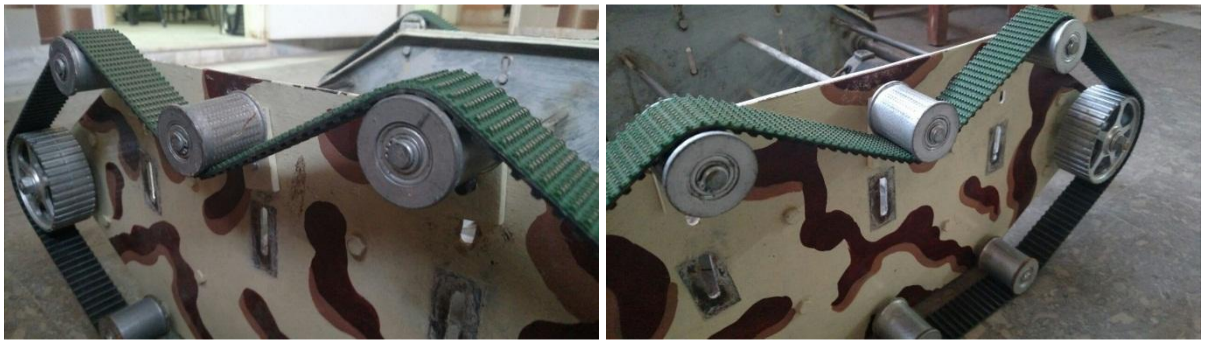
The resultant of this project is the design and development of Intelligent Mobile Robots IMRs for Firefighting and Disaster Mitigation work, for that purpose, a prototype IMR is being developed to test out the motor dynamics, mobility mechanism, fire suppression system, sensor placement and maneuverability requirement and capabilities of the final IMR.

5.2 IMR Robot Prototype

The IMR prototype discussed in this group has been developed completely indigenously by the team working at the IMR Lab, this robot which is designed to handle somewhat extreme situations and harsh temperature is build using Steel sheets and moves by means of treads in a differential drive manner.

The IMR prototype design is expected to guide us in finalizing a design and resolving any issues with respect to the hardware mechanisms of the final robots. This prototype will also serve as the main test bed for all sensors and other electronics, including high power motor drivers, wireless communication circuitry, battery capacity and management, thermal and optical imaging systems and the fire suppression system.

The below images show the state of the IMR prototype as work is being done to perfect the tread mechanism for movement. These images show the IMR Prototype without any electronics other than the motors as those were being developed and tested in parallel.



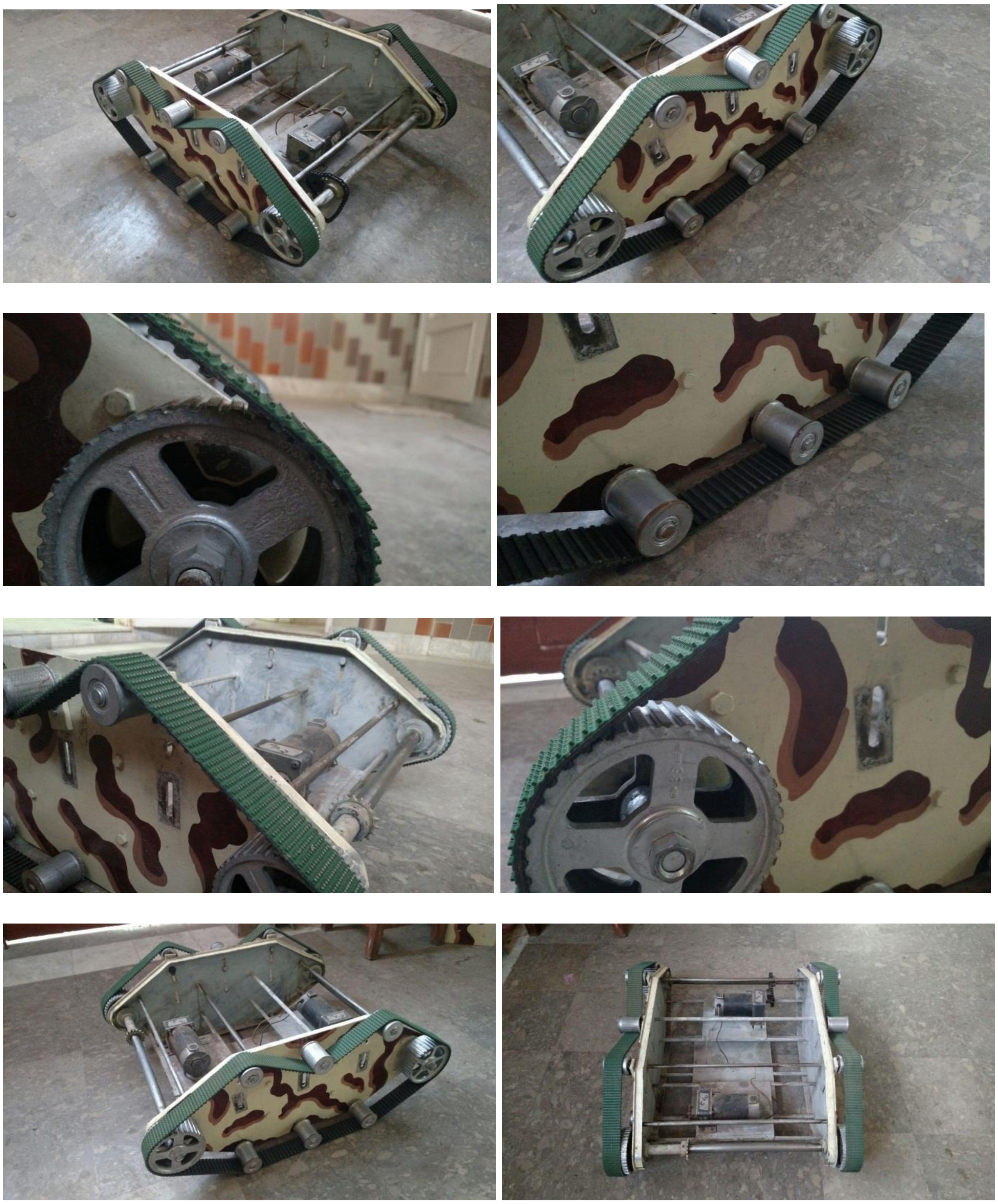


Fig 5.1: Some close-ups of the IMR Prototype

5.3 IMR Robot Specification

The IMR robot prototype as depicted in figure 5.1 has the following physical and electronic specifications:

Height: 14”

Width: 25”

Length: 30”

Weight: ~45 KG

Motor Voltage: 24V

Battery Voltage: 12V (2 batteries in series)

Battery Capacity: 12V 10Ah

Wireless Communication: 2.4GHz 6 Channel Flysky ia6 Module

Radio Audio/Video Link: 2.4GHz Audio/Video Transmitter with Camera and Receiver

Manual Control Module: Turnigy TGY-i6 6 Channel Remote

5.4 Manual Control of Prototype

As the main reason for the IMR prototype robot is to test the behavior of the hardware mechanisms and the sensor systems, the prototype will be mostly controlled manually and the run time data will be gathered for further offline processing and development and testing of the algorithms. The Manual control is achieved by integrating a 2.4GHz radio receiver in the robot and sending it commands via a 2.4 GHz Radio Transmitter.

The transmitter being used is a Turnigy TGY-i6 6 channel Transmitter shown in figure 5.2.

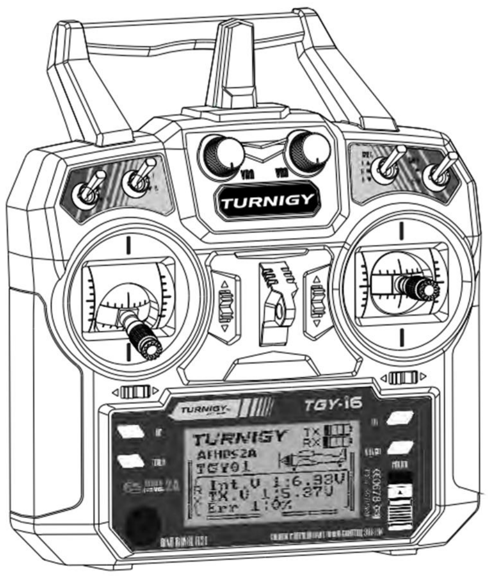


Fig 5.2: Turnigy TGY i6 2.4GHz 6 Channel Transmitter

5.5 Audio/Video Link from Prototype Robot

The Prototype IMR robot is also being fitted with a one-way audio/visual system which will provide the person controlling the robot real time view from the front of the robot. This system is analog and does not interfere with the digital broadcasts and other communication, even though it runs on the same frequency but the channels are different.

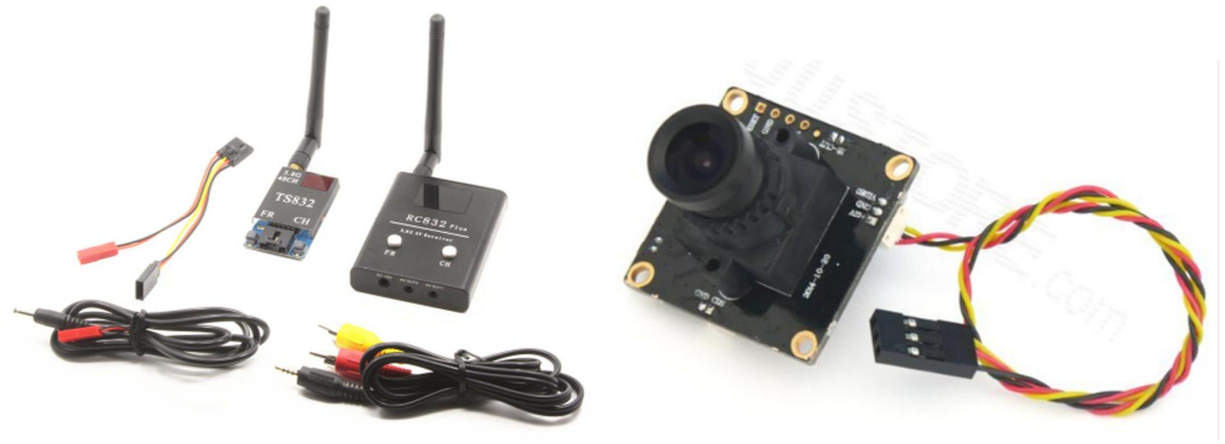


Fig 5.3: A/V Link Apparatus

5.6 Sensor Suit Testing

One of the main goals of the IMR prototype robot is to determine the ideal positions and orientations for placement of different sensors and their requirement. The sensors to be tested on the IMR prototype are the Stereographic cameras, thermal camera, ultrasonic range sensors, PT100 temperature sensors, LIDAR and even Kinect.

Some of these sensors were not initially included in the robot hardware, however as the work is progressing, the need to test different sensors and their utilization in the project needs to be ascertained and finalized. Specifically, for the Kinect Sensor which can provide RGB-D data and the LIDAR sensor which can provide highly accurate data about obstacles around the whole 360 view of the robot.

5.7 Automation Considerations

Another aspect of IMR Prototype is to determine the controllability of the robot by means of an autonomous controller which will be sending the commands to the robot to keep ascertain velocity and orientation. As this process has been heavily tested on the MUSAFIR Robots, however the final IMR robot is to be made with treads instead of wheels and the treads pose a significant issue of slippage and hence in the accurate localization and position tracking of the robot.

5.8 Hardware Improvements

As the IMR prototype robot has been in testing for some time, some improvements to the hardware are already in consideration are listed below.

5.8.1 Increasing Ground Clearance

The current ground clearance of the robot is very minimal and this leads to the robot gathering any obstacles and rubble in front of it, which hinders in proper movement of the robot.

5.8.2 Tread Slippage Issue

The Current method of fixing the treads leads to tread slippage after the robot has moved for some time, this needs to be rectified in later versions and the treads need to be more properly fixed and coupled with the robot body.

5.8.3 Access Hatch

The robot needs an access hatch for easy access to the internal circuitry and the mechanics. This will allow for easier replacement, reprogramming update of the robot electronics. However, the access hatch to be designed must also take into account water proofing of the main hulk of the robot.

5.8.4 Separate Compartment for Main Electronics

Currently the main robot body is the only compartment and houses the motors, the batteries and all of the electronics and the sensors are then placed outside on top of the robot. This needs to change and if we can get a separate compartment which houses all of the electronics including the sensors but excluding the motors and the batteries will allow for easier replacement of parts and diagnosing of any electrical or electronics related issues.

Chapter 6. Research paper submitted and presented in ICSPC 2016

Research paper titled "Sensor Fusion of INS, Odometer and GPS for Robot Localization", was presented by Dr. Muhammad Bilal Kadri at 2016 IEEE Conference on Systems, Process and Control (ICSPC 2016), that was held from 16-18 December 2016 in Melaka, Malaysia."

Chapter 7. Sensor fusion techniques for determining the exact geo-location of robot

7.1- Introduction:

Robot Localization techniques refers to the techniques that are employed for the determination of exact **'position'** estimate of a robot in an environment. The position can be represented either in the form of latitude and longitudes or in x-y position co-ordinates.

Robot Localization is necessary because for the sake of controlling a robot at a certain time, it is really important that we have the knowledge that where is the robot at that particular time. Also robot localization is useful for tracking the robot movement or path at all times. And not only this, many R&D activities are carried out on the mobile robots; therefore, the accurate travel to the desired location is of primary importance.

In this chapter, the techniques for finding the geo-location of robot are presented. Many of the localization techniques we have developed are based mainly on the sensor fusion concept. Therefore, first, the essence of the importance of this valuable technique is presented.

7.2-Sensor Fusion:

Sensor fusion is the combination of data from one or more sensors that produces a result that is more acceptable than the data obtained from the individual sensors.

The motivation behind this technique is that the data from individual or built-in sensors is subjected to noise, external disturbances, slip etc. A sensor fusion technique cancels that noise and produces a considerably noise-free result.

Kalman Filter is a well-known sensor fusion tool that is used for this purpose. And it is used in all the techniques that we have developed for robot geo-location so far. A detailed discussion of Kalman filter (KF) is presented in Section 7.3.2.

7.3-Techniques for Finding the Exact Geo-Location of Robot:

In this section, different techniques for determination of exact geo-location are discussed:

(a) Sensor Fusion of GPS/INS Only
(b) Sensor Fusion of GPS/INS and Odometer
(c) Combination of Sensor Fusion of GPS/INS and Artificial Neural Networks

7.3.1- Sensor Fusion of GPS/INS Only:

In this scheme of robot localization, only GPS (Global Positioning System) and INS (Inertial Navigation System) is used with Kalman Filter to produce the desired position of robot by sensor fusion as shown in the following block diagram:

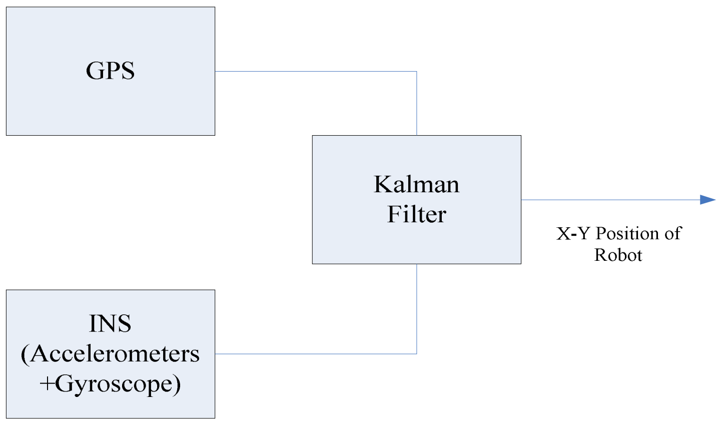


Fig7.1 Sensor Fusion of GPS/INS Only

Inertial navigation system is composed of two sensors: accelerometers and gyroscopes and depending upon the application, these sensors could be dual-axis or tri-axis.

7.3.1.1-Advantages of the Scheme:

Due to Kalman filter, the external disturbances and noise are considerably removed and an acceptable result is obtained.

7.3.1.2-Drawbacks of the scheme:

The two major draw-backs of the scheme that must be highlighted here are:

1. The scheme fails to work when there is no GPS signal as well as during GPS outages.
2. The scheme cannot be used for indoor applications as GPS works in outdoor environment only.

Therefore, due to the limitations of above scheme not suitable for indoor environments such as inside buildings etc. A system capable of working indoors is developed. This system is discussed in the next section.

7.3.2-Sensor Fusion of GPS/INS and Odometer:

A system capable of giving exact position of robot can be implemented by sensor fusion of data from three sensors: GPS, INS and odometer. The sensor fusion is carried out according to the following steps:

- **Sensor Fusion of Odometer and INS Using Kalman Filter**
- **Sensor Fusion of Kalman Filter Position Output and GPS Using Weighting**

The following block diagram shows how final position output is obtained using this scheme.

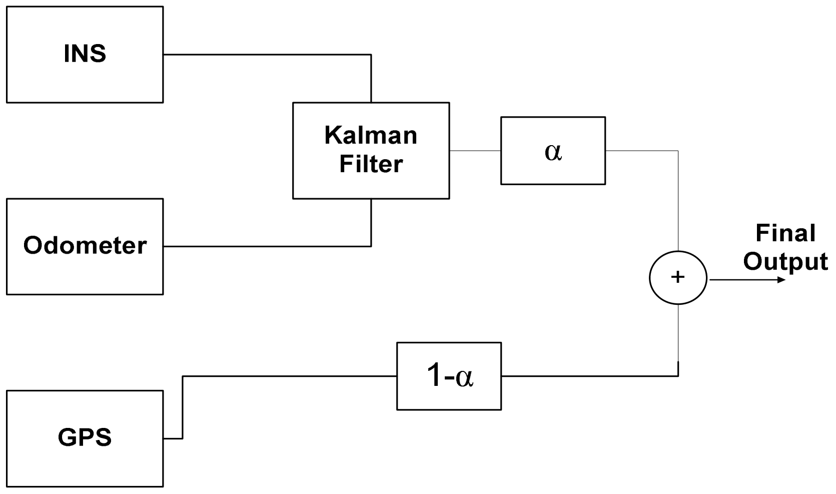


Fig7.2 Sensor Fusion of GPS/INS and odometer

In the above diagram it is shown that the weighting parameter 'α' is used to fuse the data from Kalman Filter and odometer in order to obtain the exact position result, i.e:

Final Position = (1-α) GPS + α (Kalman Filter Position Output)

From above equation for final output, it can be seen that the system still works when there are no GPS signals that is in indoor environment as well as when there are GPS signals present that is: in outdoor environment. The only parameter that needs to be adjusted is parameter, 'α'.

Rules for Adjusting Weighting Parameter, α:

1. The value of weighting parameter 'α' depends on the availability of GPS signals. If GPS signals are available, then:

$$\mathbf{0 \leq \alpha \leq 0.1}$$

2.If the values of GPS signals are not available, then the weighting parameter 'α' is assigned the following values:

$$\mathbf{0.9 \leq \alpha \leq 1}$$

7.3.3-Kalman Filter for Sensor Fusion:

The Kalman Filter is used to fuse the data from odometer and INS. Since the INS consists of accelerometers and gyroscopes. Accelerometers data contains many kinds of errors such as bias and random noise.

When the accelerometer data is double integrated to get position, there is error drift increasing with time and produces wrong position results. Therefore, odometer position data is combined with INS with Kalman filter to utilize the advantages of both the sensors and to get a correct estimate of position.

Kalman Filter algorithm:

The Kalman Filter algorithm consists of the following steps:

1. Prediction and
2. Correction.

The algorithm consists of the following steps:

1. **Set the initial values of states and error covariance**

   Matrix:

$$x\hat{o}, Po$$

2. Predicting state and error covariance matrix.

$$\hat{x}_k = A\hat{x}_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

Where,

A is the state transition matrix or the system model and

Q is the process noise covariance matrix

3. Compute the Kalman Gain

$$K_K = P_K^- H^T (H P_K^- H^T + R)^{-1}$$

Here, H is the observation matrix

R is the measurement noise covariance matrix

4. Calculate the state estimate:

$$\hat{x}_k = \hat{x}_k^- + K(z - \hat{H} x_k^-)$$

5. Calculate the error covariance:

$$P_k = P_k^- - KHP_k^-$$

And the above steps are repeated iteratively.

7.4-Implementation:

To test the above sensor fusion algorithm, simulations were carried out in MATLAB/ Simulink environment.

Mathematical models of, INS, GPS and Odometer were used to generate simulated data from sensors.

The following matrices were used in applying Kalman Filtering to INS and Odometer data:

Here, a six state Kalman Filter is implemented for the

The system state vector is:

$X = [ax \quad ay \quad vx \quad vy \quad x \quad y]'$

Where,

'ax' is the robot acceleration in x – direction.

'ay' is the robot acceleration in y-direction.

'vx' is the robot velocity in x – direction.

'vy' is the robot velocity in y – direction.

'x' is the robot position x- co-ordinate.

'y' is robot position y – co-ordinate.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ t & 0 & 1 & 0 & 0 & 0 \\ 0 & t & 0 & 1 & 0 & 0 \\ 0.5t^2 & 0 & t & 0 & 1 & 0 \\ 0 & 0.5t^2 & 0 & t & 0 & 1 \end{bmatrix}$$

The sensor measurements are:

$$Z = [ax\_acc \quad ay\_acc \quad x\_od \quad y\_od]$$

Where ax_acc and ay_acc are accelerations measured from accelerometers, and x_od and y_od are x and y positions determined from odometer readings.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The observation matrix is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Process Noise Covariance matrix:

$$Q = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}$$

The measurement noise covariance matrix is:

$$R = \begin{bmatrix} \sigma^2_{ax\_acc} & 0 & 0 & 0 \\ 0 & \sigma^2_{ay\_acc} & 0 & 0 \\ 0 & 0 & \sigma^2_{x\_od} & 0 \\ 0 & 0 & 0 & \sigma^2_{y\_od} \end{bmatrix}$$

The error covariance matrix P is initialized to:

$$P_0 = Q$$

The system states are initialized to:

$$X = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

The weighting parameter'α' is assigned a value '0.5' to test the effect of both KF filtered signals and GPS signals on the final output.

The simulation results are shown in Fig7.2 and Fig7.3.

7.6- Simulation Results:

The simulation results are presented in Fig7.2 and Fig7.3.The actual path of robot in both x and y direction are shown in red. The KF estimated output is shown in green. The weighted output from GPS and KF is shown in blue.

It can be seen from Fig7.2 and Fig7.3 that since the initial values of x and y were initialized to zero, that is we do not know initial values of x and y, the KF estimate therefore takes time to converge to acceptable values. The weighted data from GPS and KF produces a better estimate of the robot's location. The proposed scheme produces estimates of the actual x and y location with minimum error and takes less time to converge than the KF output which is evident from the two graphs in Fig7.2 and 7.3.

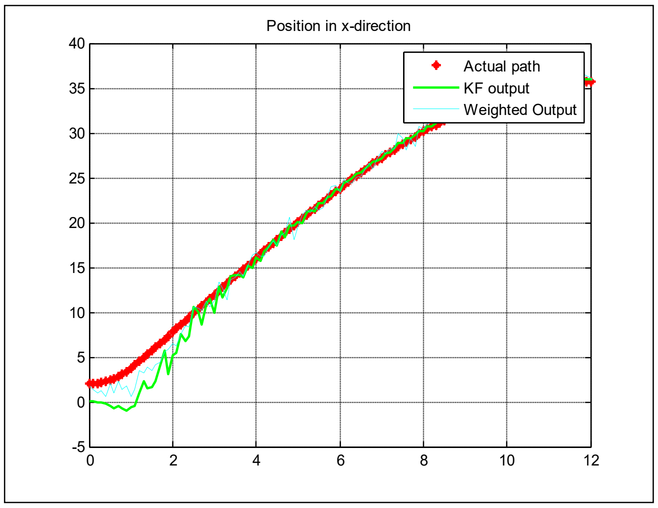


Fig. 7.2 – Position in x-direction (α=0.5)

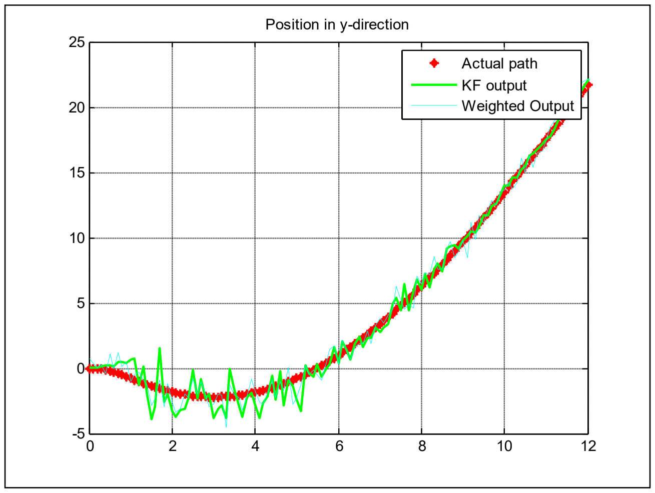


Fig. 7.2 – Position in x-direction (α=0.5)

7.7-Combination of Sensor Fusion of GPS/INS and Artificial Neural Networks

Using neural networks for the prediction of position in case GPS signal lost is presented by Jing Li et al. in their paper, **"Improving position accuracy of vehicular navigation system during GPS outages using ensemble algorithms".**

The Approach:

In the approach presented in this paper, when the GPS is on, Kalman Filter is used to provide position estimation, by fusing the data from GPS and INS.And the following data is saved in the database in the meanwhile. acceleration , attitude information and velocities from the

INS and position data obtained from the Kalman Filter. This data is then used to train and test the Neural Network when the GPS is off.

Shortcomings of the scheme:

This approach can be used for predicting robot position. However in this scheme, the authors have assumed very large GPS time outages (i.e: 600s GPS on and off time).And therefore, in this way they are able to train the neural networks for large time interval to achieve accuracy in predicted position. This approach is not practically useful for our case where the GPS outages are 1 or 2 Hz. In other words, the author's assumption of large GPS outages are strange and not practically be adopted.

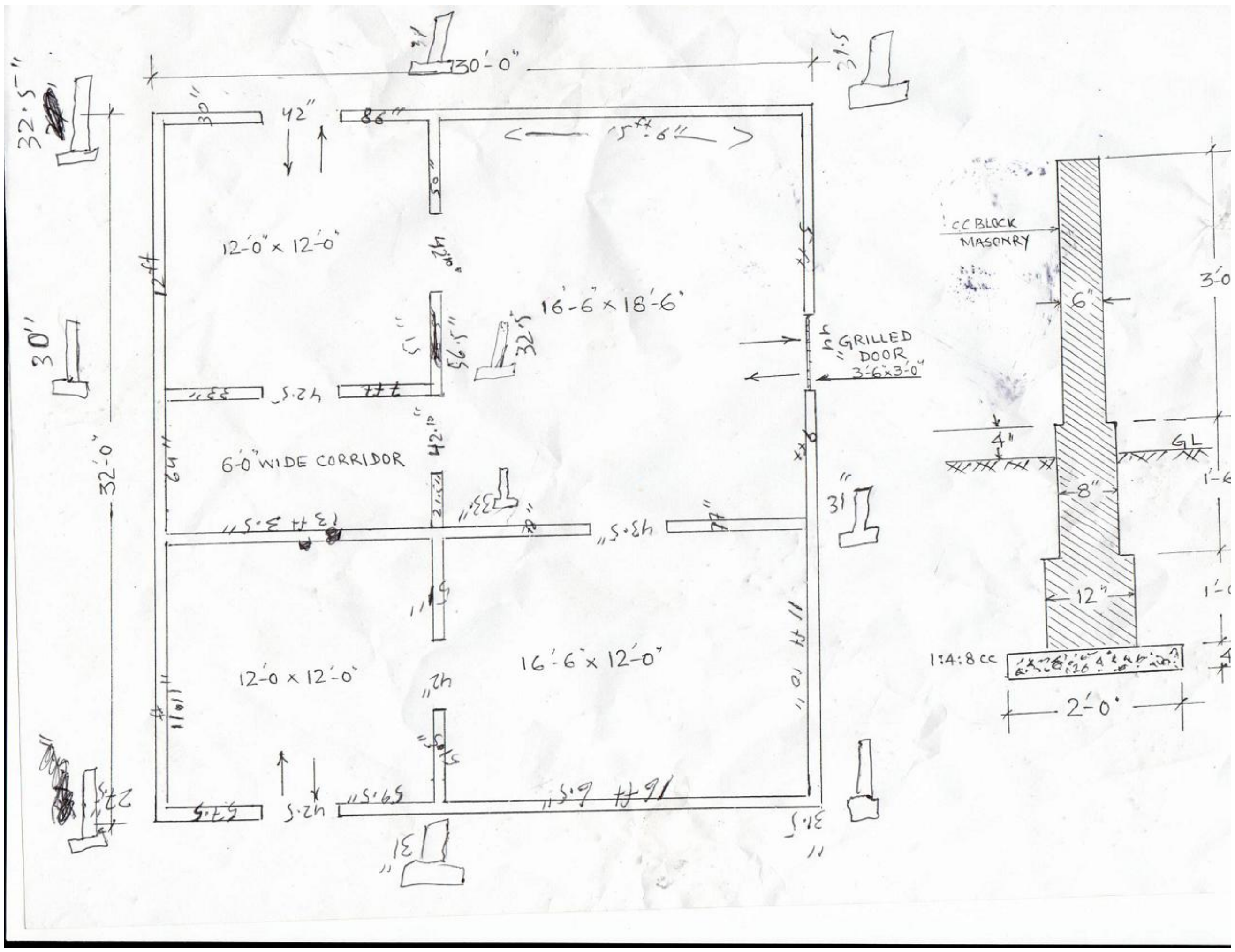
Currently we are working for the modification of this scheme so as to remove its drawbacks and make it perfectly applicable to our scheme for robot localization where we have small GPS time outages practically

Chapter 8. Development of the fire room for realistic testing of IMR

8.1 Requirement of Fire Room

We started work on construction of a Fire Room for the purpose of testing the IMR robots in simulations of realistic scenarios. This room was build keeping in view that the normal household or cottage industry or even a normal industrial plant in our country is usually built as needed and most times are NOT up to standard of safety and other regulations.

8.2 Fire Room Map



8.3 Fire Room Construction

The Fire Room is constructed at the main campus of PAF KIET as the university has a relatively large uncovered ground area, we selected and got approval to use a far off area for such a test Fire Room near the border of the university premises.

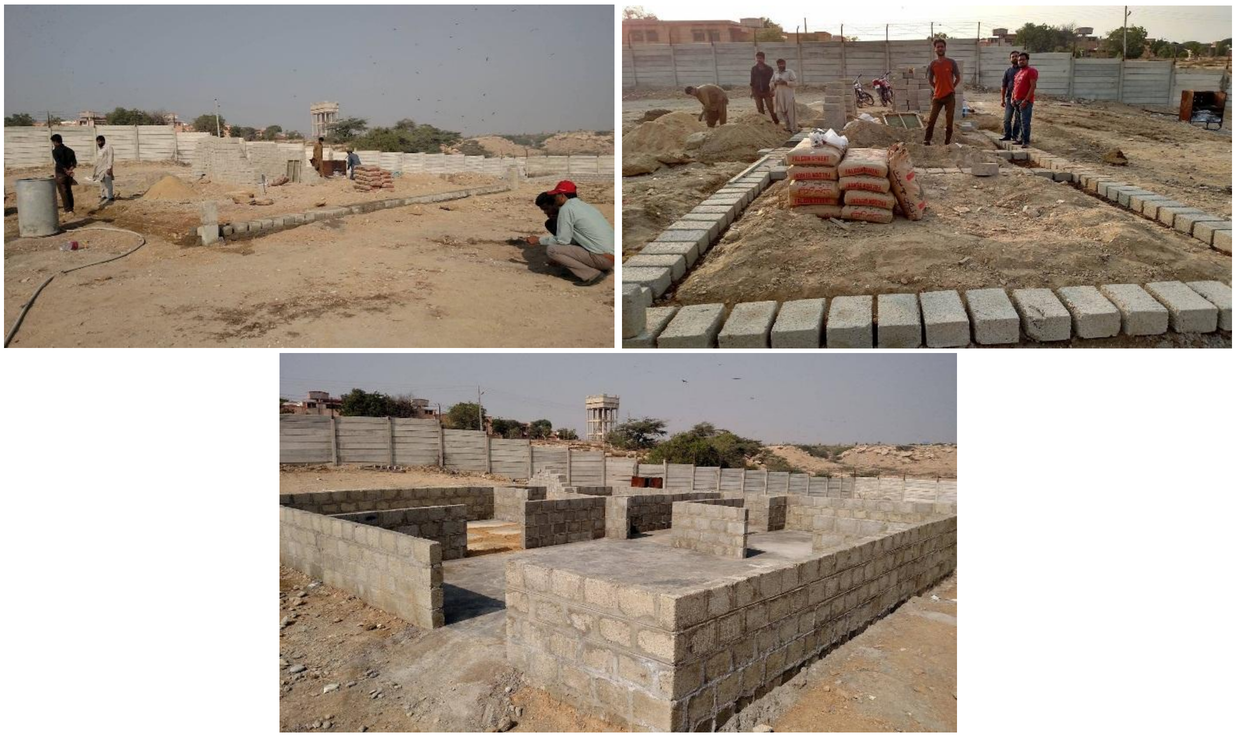


Figure 8.1: Setting Perimeter and Laying out foundation of the room

8.4 Water Reservoir:

Almost 50 feet away from the fire site, there is a water reservoir on which a motor is placed to pump water from the reservoir to the fire site. The capacity of water reservoir is around 100 cubic meter of water, which can provide water at full pressure from 1.5” hose pipe for about 30 minutes.

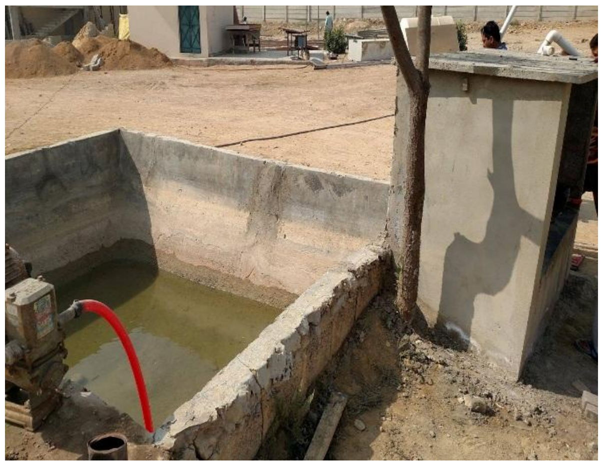


Figure 8.2: Water Tank at IMR Test Site

8.5 Completed Construction

The completed Fire Room site was painted with appropriate caution color theme and also with sliding metal doors which will allow for the map of the internal area to be changed as needed, this is essential as this will allow for this one site to be used for several different scenarios and also for several different algorithms and possibilities to be tested out.

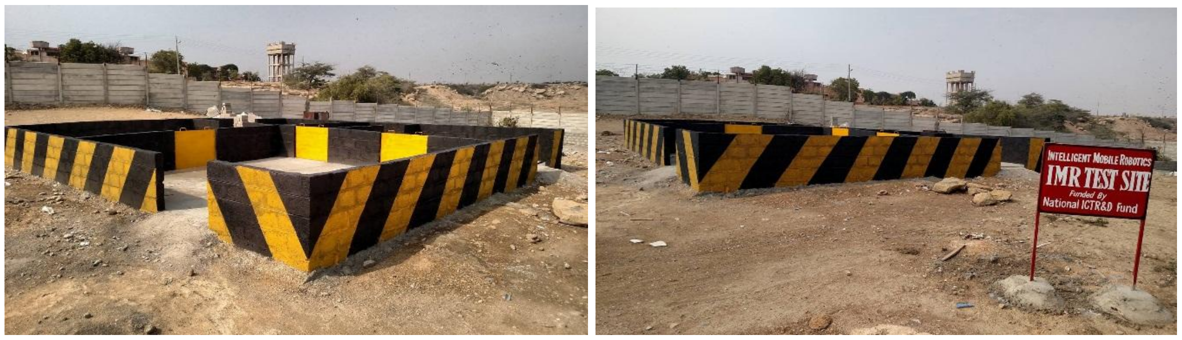


Figure 8.3: Fire Room at IMR Test Site

8.6 Water Pump Chamber:

The main Water Pump is installed near the water reservoir which also houses the electronics outlets and other necessary fire safety equipment for use when testing. These items include buckets for sand/water, fire extinguishers, fire ball for rapid fire suppression, hose for pipes and other such safety equipment.

Figure 8.4: Pump Chamber at IMR Test Site

8.7 Sensor Test Procedures

As the Fire Room has been constructed for the test of the IMR robots, the sensors all utilized will also be verified and tested based on their working in the room, the sensors will need to be tested and their results verified so that any and all optimizations, calibration, changing of sensors or other such issues are readily addressed within due time.

8.7.1 Stereo Vision

The stereo cameras used in the robot will be individually tested to determine how effective they are in a scenario such as the fire test room. The stereography should result in disparity data which will then be checked to determine the accuracy of the data by comparing it to the real values. Another aspect of the Stereo Vision test is its effectiveness or lack thereof in case of smoke and fire and how the smoke affects the disparity data.

The stereo test will need to be done at several different times of the day to determine the effectiveness of the method in daylight and at night time as well when the scene will be lighted with the lights on board the robot. The stereo algorithm will be most utilized to avoid obstacles and other such blocks in the way and to determine where the robot can and should move safely, hence the rate at which the stereovision algorithms returns the data will then determine the max speed of the robot as if the robot moves very rapidly the stereovision will not have enough time nor will it be able to give enough data for proper mapping, path planning and obstacle avoidance.

8.7.2 Ultrasonic Sensor

There are multiple ultrasonic sensors fixed to the frame of the robot which are attached at different angles. These sensors provide a point or more accurately a curve approximation of any obstacles in front of the sensor. By utilizing multiple sensors the distance data can then be used to augment the stereovision system running on the robot. The Ultra Sonic sensors have been selected as the ultrasonic waves can penetrate through smoke and provide a clearer view where image based systems will fail due to low visibility. However this needs to be determined how temperature variations will affect the output form the sensors and also the max rate of the ultrasonic sensor measurement.

8.7.3 Thermal Sensors

The Thermal sensors as used on the robot will help in determining the temperature gradients around the robot and will also allow more precise measurement of the ultrasonic signals as the speed of the ultrasonic waves is affected by the ambient temperature and the system needs to be able to determine the offsets required due to temperature variations. We will need to determine the rate at which the temperature sensors can provide data and how fast they can detect changes in the ambient temperature.

8.7.4 Thermal Camera

Thermal Camera is the final sensor to be tested at the fire room site, the thermal camera will provide a final image of what's in front of the robot in terms of temperature values and once this data is put on top of the disparity map the system should be able to determine how far away flame / fire / smoke is and if there are any persons in the view (depending on the temperature gradients and blobs of same temperature as human body).

As with the Ultrasonic sensors and the stereovision system the thermal cameras need to be fast enough as well and all these sensors combined should provide enough data for the system to be practically implemented in real life scenarios as having data coming in at speeds slower will simply slow down the whole operation of the robot.

8.8 Safety Precautions

As with any test site, emphasis has been done on safety procedures and precautions, we have equipped the test site and our lab with several different equipment which will be helpful in case of any undesirable events.

8.8.1 Safety Goggles & Reflective Jacket:

Safety goggles made of Polycarbonate material are obtained because of its certain properties such as being lightweight and have impact strong resistance as well as built-in radiation protection. Similarly, high visibility safety vest which complies with the ANSI/ISEA 107-2010 standard are procured to be used during the fire testing activity.

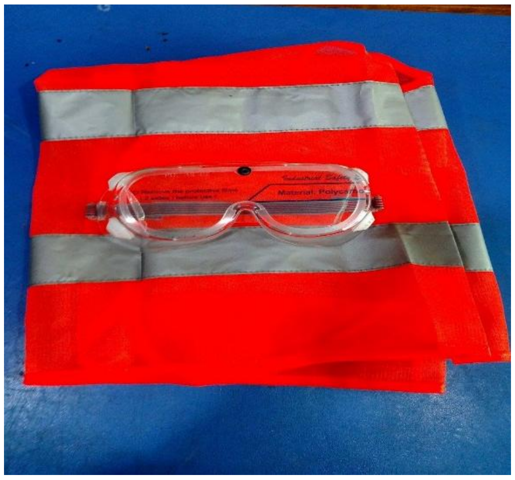


Figure 8.5: Safety Goggles and High Visibility Jacket

8.8.2 Smoke Alarm

These are roof sensors and although our test site does not have a ceiling these are being used to determine their behavior and how they can be utilized or their signals be integrated into the robot.

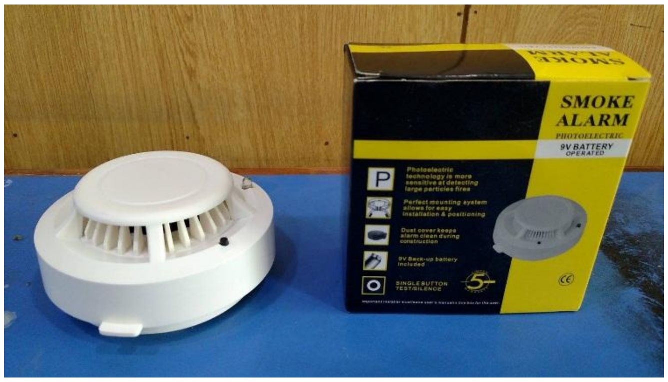


Figure 8.6: Photoelectric Smoke Alarm

8.8.3 Structural Firefighter & Normal Gloves

Structural firefighting gloves provide a delicate balance for adequate hand protection while allowing firefighters to effectively conduct essential firefighting operations. The National Fire Protection Association (NFPA, USA) stipulates that the gloves meet performance requirements as an outer shell, moisture barrier, and thermal barrier. These gloves provide firefighters with protection from sharp objects, fluids, flame, and heat. For IMR Test Site we have pairs of both structural firefighting and normal gloves.

Figure 8.7: Firefighter Gloves

8.8.4 Gas Mask

A respirator is a device to protect you from inhaling dangerous substances, such as chemicals and infectious particles. There are several different types of respirators from which we will be using chemical respirators because our objective is to protect the person who is entering the IMR Test Site from toxic and chemical fumes arising in result of the planted burnout of raw material. The cartridge or filter being used is RC 202 which is for organic vapors, mists and fumes of low toxicity.

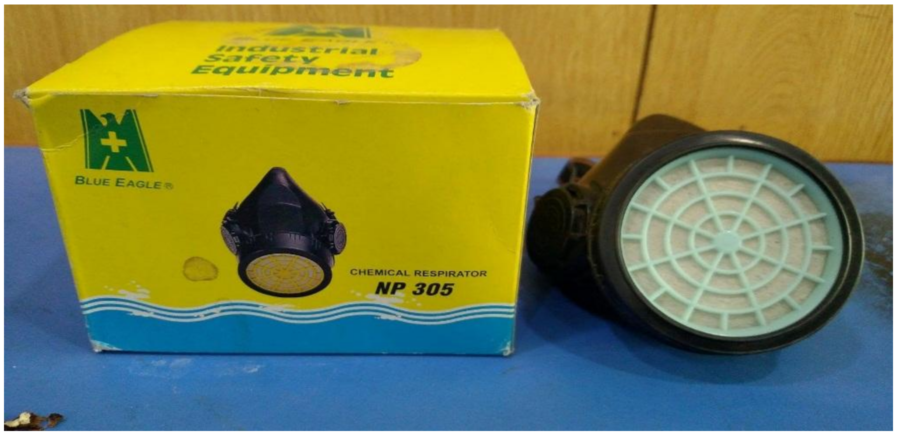


Figure 8.8: Chemical Respirator

8.8.5 Fire Extinguisher

For fire suppression purposes, two fire extinguisher units have been placed near the IMR Test Site. These are dry powder filled cylinders with 14KG total weight and can suppress Class D or combustible metal fires. The discharge time is up to 10-12 seconds applying full pressure.

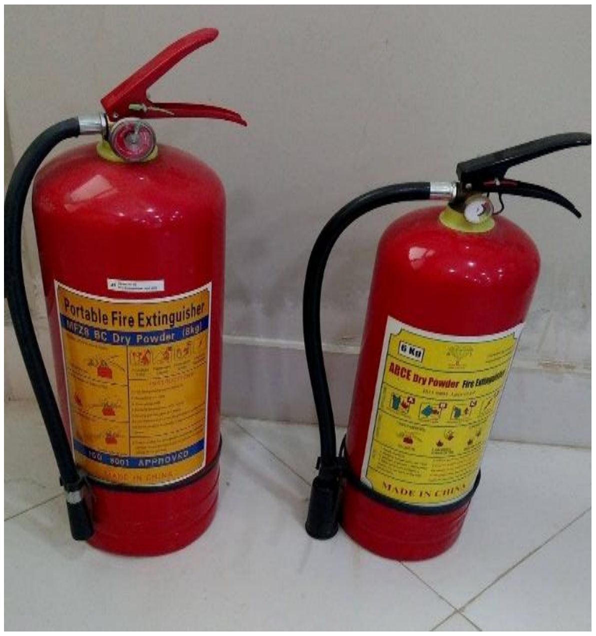


Figure 8.9: Fire Extinguisher for IMR Test Site

As stated on the FEMA site (Fire Equipment Manufacturers Association)

> "Carbon Dioxide fire extinguishers extinguish fire by taking away the oxygen element of the fire triangle and also be removing the heat with a very cold discharge. Carbon dioxide can be used on Class B & C fires. They are usually ineffective on Class A fires."

Source: http://www.femalifesafety.org/types-of-extinguishers.html

This means the fire extinguisher can be classified to use for flammable liquid and gas.

8.8.6 Fire Ball

We also have placed a Fire ball near the test site, in case of extreme emergencies this fire ball can be thrown into a fire and the fire suppressant material will explode out of the ball to cover the nearby area.

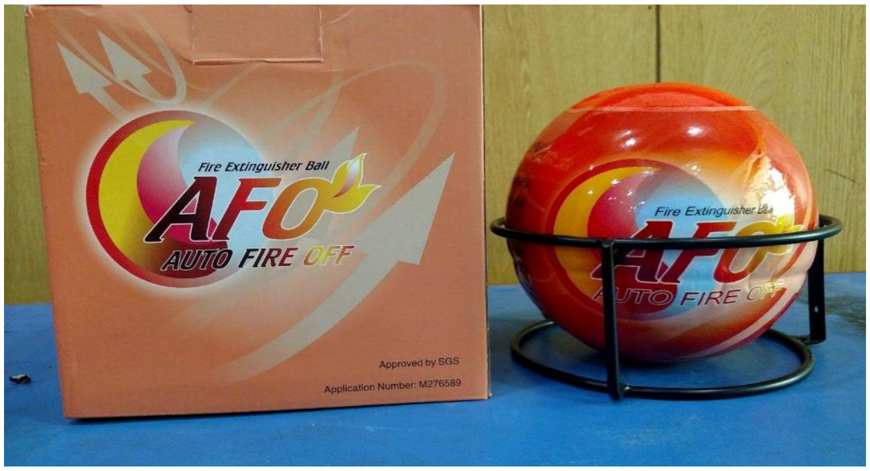


Figure 8.10: Fire Extinguisher Ball for IMR Test Site

8.8.7 Water Pipe

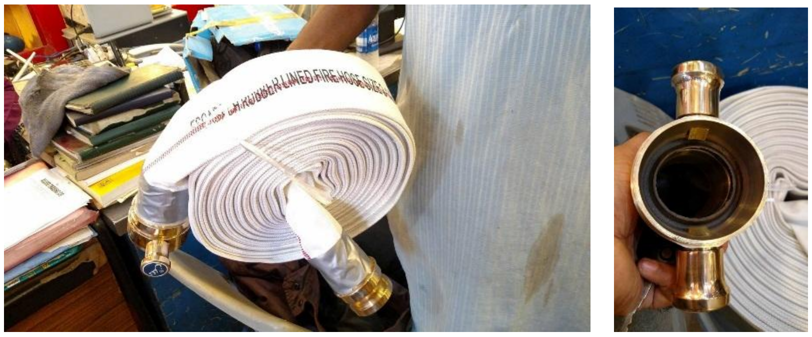


Figure 8.11: Fire Hose and Pipe

8.8.8 Pipelining for Water

PVC PIPE FOR DRAIN	3.5 INCH
LENGTH	50 F'T
FIRE HOSE	100 F'T (16) BAR 1.5 INCH

Chapter 9. Bottlenecks, Challenges and their Solution

The idea of navigating the robot independently in the field has some challenges due to the unique scenario i.e. a fire environment. Just like firefighters eyes, we can use stereo vision sensors to visualize the hazard area in the indoor environment. Based on the intelligent processing just like human brain, the Robot can determine its future direction. But the main problem in using the stereo vision sensors is the smoky environment. Because of the smoke, it is difficult to visualize the scene as the information is not clear that what is behind this smoke. An example of the smoky environment is shown in figure 9.1

Figure 9-1: Indoor Smoky Environment

To mitigate this problem, we use Ultrasonic (UT) Sensors to determine the obstacle in front of the IMR that will help the algorithm in computing the future direction. The problem with UT sensors is; they are highly directive. It works on the reflection principle i.e. they receive the echo only when the object is aligned with the sensor as shown in figure 9.2. But when the object is not aligned with the sensor, it won't receive any signal as shown in figure 9.3

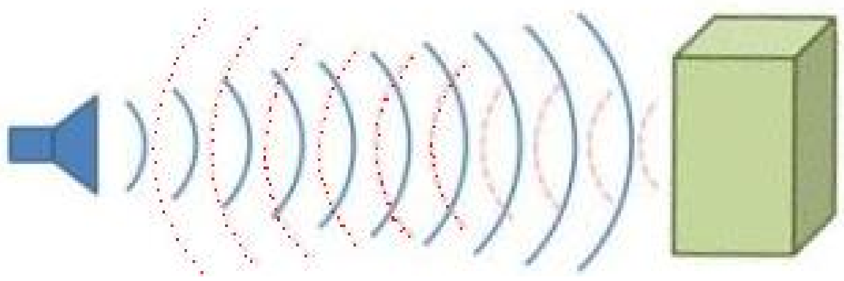


Figure 9-2 Object aligned with the sensor

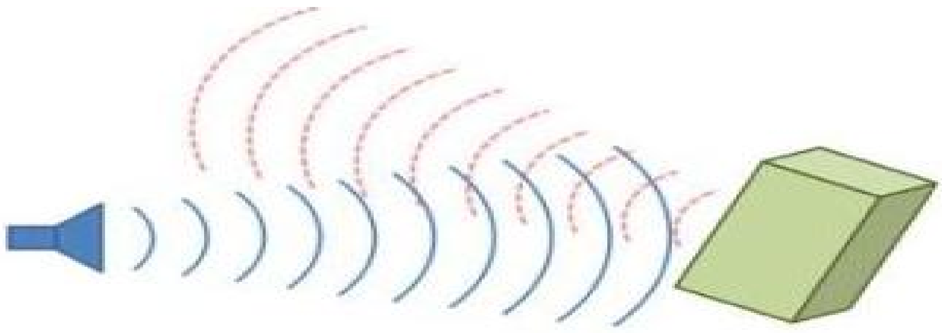


Figure 9-3 Object not aligned with the sensor

To mitigate this problem, there are two options. First is; Increase the number of UT sensors mounted on IMR. Second is; rotate the UT sensor after particular angles to scan the environment. We are currently working on these problems.