Chapter 5

Indoor Self-Localization of Robots

5.1 Introduction

Autonomous mobile robots are promising candidates in several applications such as greenhouse monitoring, as tour guides, personal assistant for elderly people in an Ambient Assisted Living System (AALS), etc. In addition to the several technological issues which are yet to be resolved before the robots can be reliably employed in human-friendly applications, the cost associated with the design, construction and maintenance of these robots pose a serious threat to their widespread usage. For robots in SRS and MRS, operating in indoor environments where GPS is not accessible, relative localization is an essential capability required for cooperative task completion. Beacon based localization as presented in Chapter. 4 can be employed for localization in indoor environments. However, a large number of beacons may be required to ensure complete coverage in an indoor environment. In a large scale or medium scale multi-robot or swarm robotic system, if all the robots rely only on beacons for localization, as the number of robots in the system increases, the wireless channel will be extensively utilized for only localization and the channel will not be available for other communications required for co-operative task completion. Hence, to reduce the overall cost of implementation of the system and also for the effective utilization of wireless channel, it is essential that the robots have the capability to localize themselves for at least smaller distances.

5.2 Problem Formulation

As discussed in the previous section, to reduce the overall cost of localization hardware and to ensure that the wireless channel is free for other robotic functions, the mobile robots should be able to localize themselves in the indoor environments for at least smaller distances without the assistance of reference or beacon nodes. For short-term relative localization, robots may utilize odometry based dead reckoning. Odometry based localization techniques are prone to the systematic and non-systematic errors as mentioned in Chapter. 2. Once a precise error model of the system is developed and its parameters are determined, the accuracy of odometry can be remarkably improved by suitable control techniques based on the kinematics of the robot [72–74]. However, this requires a detailed understanding of the kinematics of the robot. In addition to this, if odometry based localization is utilized, any error in localization might lead to error in the navigation of robot and vice versa. It is desired that a localization system is developed, that does not utilize explicit inputs from the navigation system so that the localization module can independently provide inputs to the navigation system to achieve the desired movement of the robot.

For cooperative task completion in SRS, it is required that the robots move towards other robot(s) for appropriate pattern formations for performing tasks beyond their individual capabiities. In such scenarios, the robots may inform each other about their current position obtained via beacon based localization. After this, in most cases, the robots estimate their shortest path to the destination, which is often a straight line and then navigate towards the destination. Even though the robots may localize using image processing techniques or using Lidars as mentioned in Chapter 2, these techniques are not cost-effective. Also, the image processing techniques can be effectively utilized only in known environments. In SRS systems, aggregation or pattern formation of the robots can be performed using low cost cameras and suitable image processing techniques, if the robots are within a range of few centimeters. However, the accuracy of image processing based localization will deteriorate significantly if the robots are seperated by a distance in the range of a few meters or utilized for applications which require continous tracking or continous localization of the robot. A cost-effective scheme for self-localization has to be designed such that a wheeled robot can estimate its two dimensional position (x_1, y_1) relative to the last known position (x_r, y_r). It is desired that localization scheme can be reused for two-dimensional localization of any kind of wheeled robot without the need of calibration to suit the specific robotic structure. With the

advancements in MEMS technology, the inertial measurements units (IMU) can serve as a suitable substitute for motion sensor in the place of position or wheel encoders. In this chapter, the work carried out to evaluate the feasibility of utilizing IMU for self-localization of robots for short distances in terms of a few meters is presented. The scope of the current work is to design an IMU based self-localization scheme and evaluate the feasibility of IMU based localization system to localize the robot between the start and stop activities of the robot. Following assumptions are made for the design of the localization scheme.

- The robot is programmed to move towards the destination along straight line paths.
- Any change in direction of movement of robot is allowed through right angled movements only.

5.3 Introduction to Inertial Measurement Unit

An inertial measurement unit typically consists of the following.

- Two or three axis gyroscope and accelerometer to sense the angular velocity and linear acceleration respectively along the six degrees of freedom. IMU units may also incorporate two or three axis magnetometer which measures the strength and direction of the Earth's magnetic field to provide information about orientation
- Core logic, which mostly includes a digital signal processor which performs digital signal filtering and sensor data enhancement through calibration, thermal compensation, and data formatting
- An analog front end that filters and digitizes the accelerometer, gyroscope and magnetometer readings for further processing by the core logic
- A precision temperature sensor for internal temperature compensation
- Serial Peripheral Interface (SPI), Universal Asynchronous Receiver Transmitter (UART) or Inter-Integrated circuit (I^2C) interface for connectivity to host microcontroller/processor for the configuration of IMU and also for the access of data measured by the IMU

5.3.1 Sources of Error in IMU Measurements

Accelerometer, gyroscope and magnetometer are designed to measure linear acceleration, angular velocity and orientation of the object on which the sensor is mounted [110]. Even while the object (and hence IMU) is stationary, accelerometer and gyroscope still measures forces due to gravity and rotation of the earth respectively. The Earth moves through the inertial frame and can be seen as experiencing motion compared to the inertial frame. The rotation of the earth through the inertial frame may be observed also in the gyroscope measurements. As the IMU moves through the Earth's coordinate system, these forces must be removed from the measurements, leaving only the forces due to motion over the Earth's surface. Most common sources of error in IMU are mentioned in this section [111]. These errors should be carefully removed from the IMU measurements before the data provided by IMU can be utilized for localization. All the errors may not be relevant for any given IMU, as some of the error components may be too small to create a significant difference in the result.

Bias: For a given physical input, a sensor output may be offset by the bias. For example, if the IMU is stationary and horizontal to ground, the vertical axis measures the effect of gravitational acceleration. Gravity has a nominal acceleration of 9.81 m/s^2, however with bias, the IMU may report $9.81 \pm \delta\ m/s^2$. The difference between the measurement corresponding to the input applied to the sensor and the output provided by the sensor is referred to as bias. In an IMU, the accelerometer, gyroscope and the magnetometer readings may be offset by the bias. Also, the initial bias may be different after each power up of the device and is generally referred as "Turn-on to Turn-on Bias". This is mainly due to change in the physical properties of the IMU and initial conditions of signal processing within the core logic. The initial bias may change over time due to the changes in temperature, and/or mechanical stress on the system and is referred as "In-run Bias". IMUs are often manufactured with temperature compensation mechanisms to compensate for the In-run bias.

Scale Factor or Linearity : Scale-factor error is the ratio of the output error (deviation from the fitted straight line slope) over the input and is typically expressed as a percentage or ppm (parts per million). For eg, if the input is $10m/s^2$, and if there is a 6% scale factor error, the output measurement reported by the sensor will be 10.6 m/s^2. Scale factor effects are most significant during times of high acceleration and rotation. This deterministic error can be removed from the measurements at run time suitable programming.

Random Walk (Sensor Noise): If a sensor measures a constant signal for a longer period of time,

a random noise (error) may be present in the measurement. The integration of the random walk errors in the measurements can lead to significant errors in measurements.

Sensor Non-orthogonality (Misalignment): A 9-axis IMU consists of, three axis gyroscope, accelerometer and magnetometer mounted orthogonal to each other. Errors in mounting of the sensors may lead to error in sensor measurements. For example, for a 3-axis accelerometer, if one axis is pointed perfectly up and the IMU is kept horizontal to ground, only the vertical axis measurement must measure gravity. However, if there is nonorthogonality among other two axes, then those axes also measure gravity, leading to a correlation in the measurements. The non-orthogonality may also be observed between the accelerometer and gyroscope readings as well. Although misalignment is a manufacturing error, factory and runtime calibration can be performed to minimize this error.

It is therefore important to select an IMU module which allows removal of different errors and run-time calibration as mentioned in this section. The same algorithm can deliver better results using calibrated sensors when compared to the ones without calibration.

5.4 Overall Architecture of the two-dimensional Self-localization System

An Inertial Measurement Unit (IMU) based scheme for the continous localization of the robot for short distances is presented in this section. The localization system is developed as a stand-alone system and does not receive any input from the robot on which it is mounted (Appendix C). As the robot starts moving, the localization system estimates the position of the robot until it stops. If the robot stops, then a new localization cycle is started. The localization module consists of an Arduino Mega 2560 microcontroller board, MPU-9255 IMU unit, HC-06 Bluetooth module and the associated powering circuitry. The Arduino Mega2560 is interfaced with MPU-9255 and HC-06 module via the I^2C and UART interface respectively.

MPU-9255 is a MEMS-based, 9-axis inertial measurement unit manufactured by Invensense that includes a 3-axis gyroscope, 3-axis accelerometer and 3-axis magnetometer [112]. MPU-9255 features three 16-bit Analog-to-Digital Converters (ADCs) for digitizing the gyroscope outputs, three 16-bit ADCs for digitizing the accelerometer outputs, and three 16-bit ADCs for digitizing

the magnetometer outputs, thus providing high resolution measurements. The gyroscope within MPU-9255 can be user programmed to a full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps), the accelerometer can be programmed to a full-scale range of ±2g, ±4g, ±8g, and ±16g, and the magnetometer can be programmed to a full-scale range of ±4800μT. MPU 9255 is selected for this work as the range of accelerometer and gyroscope is sufficient to track both fast and slow motions as is indicated by their full-scale range as mentioned above. MPU-9255 provides programmable digital filters, a precision clock with a maximum clock drift of only 1% for a operating temperature range of 40°C to 85°C, and an embedded temperature sensor for compensating the "In-run bias" components as mentioned in Section 5.3.

Another significant feature of MPU-9255 is that it incorporates a Digital Motion Processor (DMP), which offloads the task of maintaining the timing requirements from the host processor, while capturing the real-time motion data. The DMP acquires data from accelerometers, gyroscopes, magnetometers, and stores the data in a 512- byte FIFO. The data can be read from the DMP's registers by the host processor/controller. The DMP will raise an interrupt to the host processor/-controller, when the FIFO gets filled beyond the required threshold. MPU 9255 device provides both Inter-Integrated circuit (I^2C) and SPI serial interfaces for capturing data and for the device configuration. Communication with all configuration and data registers of the device can be performed using either I^2C at 400kHz or SPI at 1MHz. For applications requiring faster communications, the sensor and interrupt registers may be read using SPI at 20MHz. The MPU 9255 is a low power consuming device which operates at a typical voltage range of 2.4V to 3.6V.

To develop a localization technique and to analyze the feasibility of IMU-only localization, it was necessary to analyze the information provided by the IMU, independent of the restrictions imposed by the processing capabilities of the microcontroller or microprocessor on which the algorithm is implemented. Hence, in this work, the information provided by the IMU is processed in MATLAB to develop a suitable localization technique. Arduino Mega 2560 based on Atmega 2560, a simple 8-bit microcontroller, is utilized to read the accelerometer, gyroscope and magnetometer values and transmit the data to a laptop via the Bluetooth interface. The Arduino module also supports a real-time visualizer software to assist visualization of data from Arduino thereby facilitating support for quick debugging of errors. Arduino Mega 2560 is henceforth mentioned in the thesis as Arduino module.

5.4.1 Software Architecture of Proposed 2D self-localization Scheme.

Figure. 5.1, provides an overview of the self-localization scheme. The self-localization is achieved via the following steps.

- Raw Signal Conditioning of the IMU data
- Instantaneous Activity Detection
- Activity Correlation
- Velocity and Heading Estimation
- Position Updation

The major contribution of this chapter is the design of the instantaneous activity detection unit and the activity correlation. The steps involved in localization are explained in detail in this section.

5.4.1.1 Input Conditioning

Necessary steps for intializing the IMU and conditioning of the accelerometer, gyroscope and the magnetometer outputs as mentioned in the data sheet of MPU-9255 module were performed using Arduino module [112]. The gyroscope's full range is set to 250°/sec (dps) and accelerometer's full range is set to ±2g. The module contains bias registers for accelerometer, gyroscope and magnetometer which are reset to zero during power up. The 'bias' values are measured based on the proceedure mentioned in datasheet and the measured bias values are written into the corresponding bias registers. The MPU-9255 will automatically subtract the bias from the further readings before outputting the data. MPU 9255 implements internal temperature compensation mechanism to minimize the 'In-run' bias in measurements. The module is factory calibrated for scale factor compensation. The 3-axis accelerometer, gyroscope and magnetometer readings are read from the corresponding data registers by reading the 16-bit data registers of MPU-9255 via I^2C interface of the Arduino at an interval of 0.2 seconds. The 16-bit data registers of MPU-9255 are read as two 8-bits read operations via I^2C. The two 8-bit data values are then concatenated together to form 16-bits of information. This process is repeated on the 3 sets of data from accelerometer,

gyroscope and magnetometer along the x,y and z axis. The calibration of the magnetometer is performed to compensate for magnetic declination. The effect of gravity component has to be removed from the z-axis component of accelerometer. To compensate for the effect of gravity, the bias value is measured after the reset proceedure of the IMU and is programmed into the bias register meant for the z-axis accelerometer readings to ensure that the gravity component is automatically removed from the data provided by the IMU.

The raw accelerometer readings along the 3-axes under static conditions of the robot, before input conditioning and the corresponding readings after input conditioning are shown in Figure 5.2 and Figure 5.3 respectively. Figure 5.3, shows that the input conditioning is accurately performed. Similarly, the input conditioning of gyroscope and magnetometer is also performed. The heading

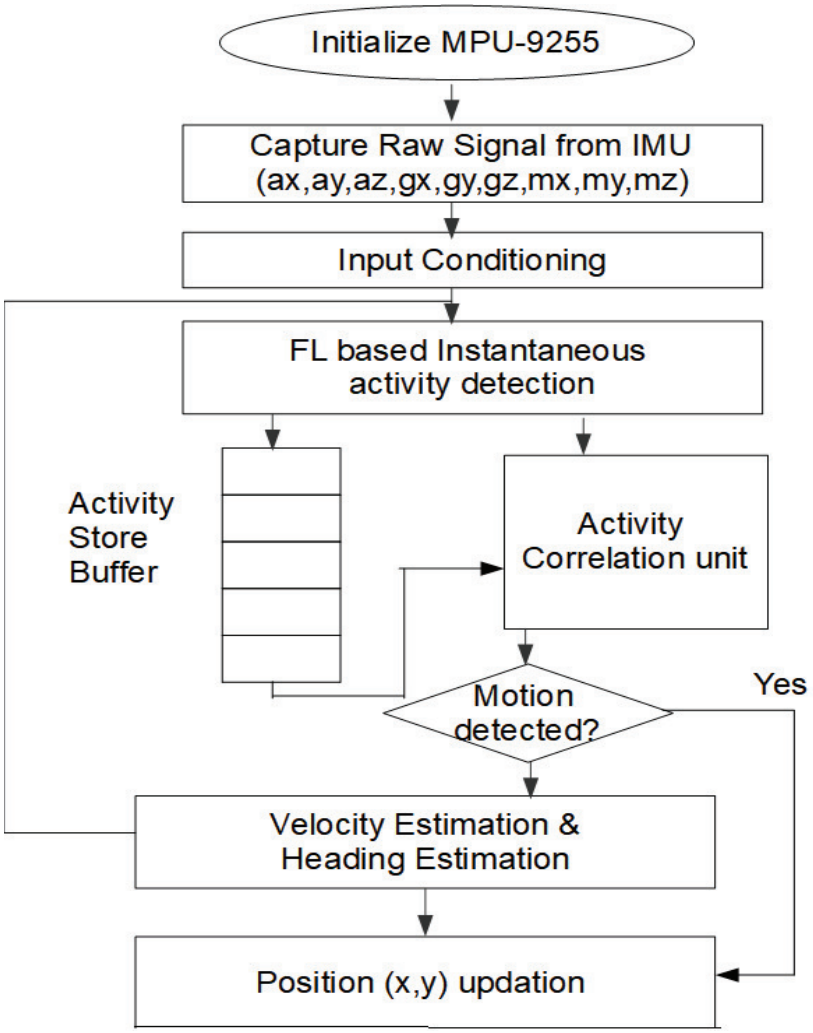

Figure 5.1. Flow chart depicting the steps involved in two-dimensional localization

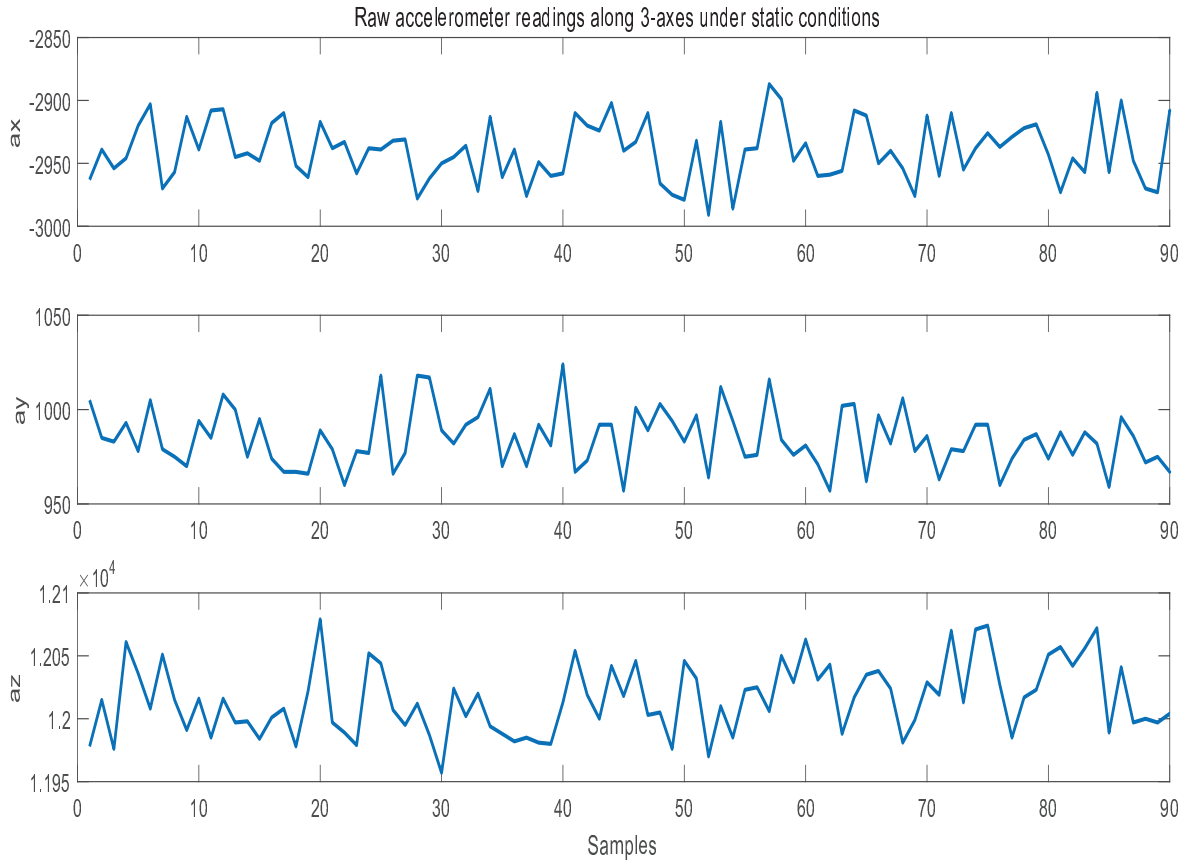

Figure 5.2. Raw accelerometer readings along 3-axes under static conditions of robot

(θ_m) of the robot is measured using the magnetometer reading, along the y and x axis as follows.

$$\theta_m = atan2(-m_y, m_x) * 180/3.14. \tag{5.1}$$

$$if(\theta_m < 0) then\ \theta_m = 360 + \theta_m; \tag{5.2}$$

The heading of robot can also be calculated from accelerometer readings as follows

$$\theta_a = atan2(a_x, a_y); \tag{5.3}$$

where a_x and a_y are the accelerometer readings in the x and y directions respectively. The accelerometer readings are prone to random noise in short-term. The gyroscope values are less noisy in short-term, however, will slowly drift away from the true value in the long-term. Hence, a complementary filter may be utilized to stabilize the gyroscope readings with the help of accelerometer readings. The stabilized gyroscope (angular turn) reading along the z-axis ($angle$)

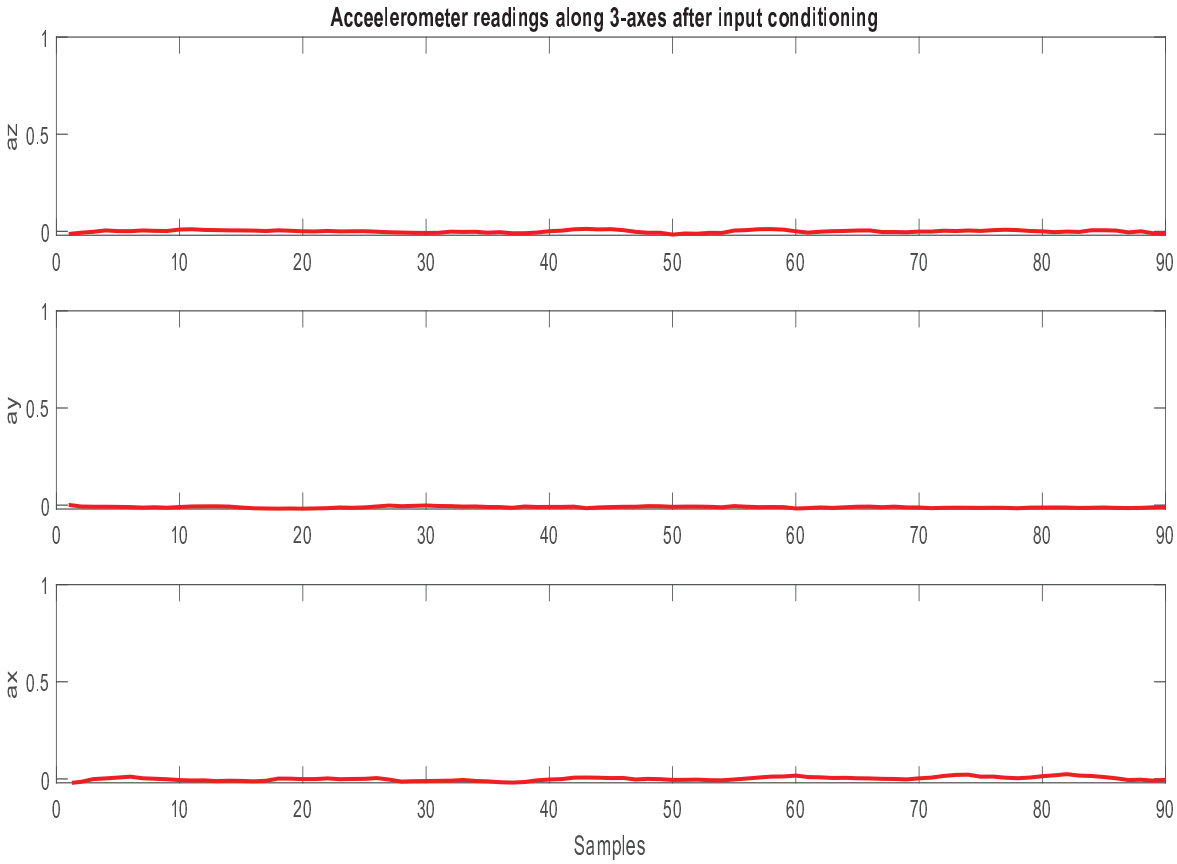

Figure 5.3. Accelerometer readings along 3-axes after input conditioning

can be calculated as follows.

$$angle = 0.98 * (angle + \theta_g) * \Delta T + .02 * \theta_a \tag{5.4}$$

where 'θ_g' is the raw gyroscope reading along the z-axis and ΔT is the sampling interval. However, in the present work, as only short term localization is attempted, 'θ_g' is directly utilized without utilizing the complementary filter. The conditioned gyroscope and accelerometer measurements along the three axis, the heading information (θ_m) and the time at which the data values are read by the Arduino are transmitted to the basestation laptop via a Bluetooth module.

5.4.1.2 Activity Detection

IMU is widely used in human activity recognition [113, 114]. Inspired by the effectiveness of IMU for activity recognition and based on the analysis of IMU data, it was concluded that if

various activities of the robot can be detected accurately, the location estimation of the robot can be achieved. Hence, in this thesis, activity recognition based on the accelerometer, gyroscope and magnetometer readings from MPU-9255 is implemented as a crucial step towards localization of robot.

Activity recognition of the robot may be implemented through several techniques. Activity recognition of a robot may be attempted by representing the various motions of a robot and the possible range of values represented by the accelerometer, gyroscope and magnetometer readings of IMU mounted on the robot as a look-up table. During the run-time, the robot can then continuously compare the measured readings against the table entries to identify the activity. The activity detection algorithm can scan the various entries of the look-up table, which has the record of the activity for every input or combination of inputs. However, as the number of activities to be detected increases, the size of the table also increases, leading to increased memory requirements for the implementation of the system. In addition to this, the tabular approach may lead to incoherent results especially in situations in which there can be several combinations of input readings leading to the same activity. Tabular mechanisms may lead to incoherent activity detection due to the discrete nature of input/output combinations that can be represented as a table. The alternative to look-up tables, is to model the activity using mathematical formulae, i.e to have a controller execute a set of equations that express the output as a function of the input. The formulas can be very complex, and executing them in real-time may not be affordable on a low-cost microcontroller or microprocessor.

Fuzzy Logic (FL) enables low-cost microcontrollers to perform advanced functionalities traditionally performed by more powerful processors. FL, like the human brain, can make precise decisions from imprecise information. One of the reasons for significant interest on fuzzy inference systems is due to its ability to describe the behaviour of the system inorder to incorporate semantic knowledge and intuition using FL logic. Human beings utilize this semantic knowledge to recognize an activity accurately. For example, if walking and sleeping activities of a person are recognized at the same time, it is implied that at least one of the recognition results is incorrect because people generally do not walk while sleeping. If an activity recognition system incorporates semantic knowledge about the activity, in addition to the information provided by the activity model (sensors), the accuracy of activity detection can be improved. FL overcomes the disadvantages of both table-based and formula-based systems. FL has limited memory requirements, when compared

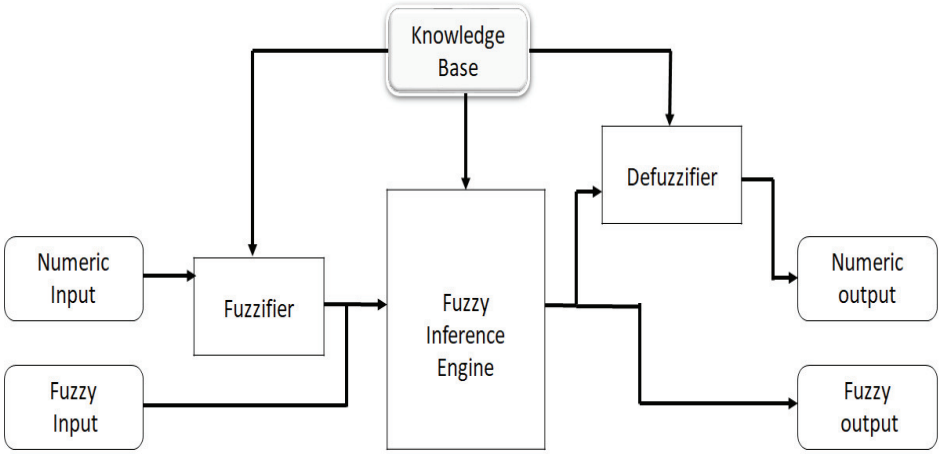

Figure 5.4. Structure of a Fuzzy inference system

to that of look-up table based activity recognition solutions, and have lesser processing capability demands than formula-based solutions. FL does not require intricate mathematical modeling and only requires a practical understanding of the overall system behaviour. FL can thus make control and analysis of complex systems much simpler. FL can result in higher accuracy and smoother control of systems as well.

A Fuzzy Inference System (FIS) is a system that uses fuzzy set theory to map inputs to outputs. A Fuzzy set is a class of objects with a continuum of grades of membership. Such a set is characterized by a membership (characteristic) function which assigns to each object a grade of membership ranging between '0' and '1' [115]. FIS have been successfully applied in several fields such as decision analysis, system modelling, system control, etc. In this work, a fuzzy inference system (FIS) is utilized for recognizing the different patterns in movement of the robot for activity detection.

5.4.1.3 Fuzzy Logic- A Brief Introduction

The typical structure of a fuzzy system consists of four major functional blocks (Figure 5.4) that are as follows [116] :

- **Fuzzifier:** Both linguistic values (defined by fuzzy sets) and crisp (numerical) data can be applied as inputs to a fuzzy inference system. If crisp data are applied as inputs, then the inference process is preceded by fuzzification, which assigns the appropriate fuzzy set to the

non fuzzy input. Fuzzifier converts the crisp input to a linguistic value using the membership functions stored in the fuzzy knowledge base.

- **Knowledge base:** The rule base and the database are jointly referred to as the knowledge base. Rule base contains the fuzzy IF–THEN rules. The membership functions of the fuzzy sets used in the fuzzy rules form the database.

- **Inference Engine:** Using IF-THEN type fuzzy rules, fuzzy inference engine converts the fuzzy input to the fuzzy output. Fuzzy inference engine combines the membership values on the antecedent (input) part to calculate firing strength (degree of fulfillment) of each rule. Inference engine generates the qualified consequents (outputs) which are either fuzzy or crisp, for each rule depending on the firing strength.

- **Defuzzifier:** In addition to the linguistic values, the numerical data may be required as the system output. In such cases defuzzifier assigns the representative crisp data to the resultant output fuzzy set. Defuzzifier converts the fuzzy output of the inference engine to crisp output.

The two main types of fuzzy inference methods/models are the Mamdani model and Sugeno or Takagi-Sugeno model [117]. In the Mamdani model, a fuzzy system with two inputs 'x_1' and 'x_2' (antecedents) and a single output 'y' (consequent) are described by the following fuzzy IF-THEN rule:

- IF x_1 is A_1 and x_2 is A_2, THEN y is B

where A_1 and A_2 are the fuzzy sets representing the antecedent pairs and B is the fuzzy set representing the consequent. To compute the output of Mamdani FIS, given the numeric inputs, the following steps are required.

1. Determination of the set of fuzzy rules

2. Fuzzifying the inputs using the input membership functions,

3. Combining the fuzzified inputs according to the fuzzy rules to establish a rule strength,

4. Finding the consequence of the rule by combining the rule strength and the output membership function,

5. Combining the consequences to obtain an output distribution, and

6. Defuzzifying the output distribution (if crisp output is required).

Sugeno, or Takagi-Sugeno-Kang method is similar to the Mamdani method in many respects. The first three steps of fuzzy inference process are the same for both models. The main difference between Mamdani and Sugeno model is that the Sugeno output membership functions are either linear or constant. In Sugeno model, a fuzzy system with two inputs 'x_1' and 'x_2' (antecedents) and a single output 'y' (consequent), then the fuzzy IF-THEN rule is as follows:

- IF x_1 is A_1 and x_2 is A_2, THEN y is $f(x_1, x_2)$

where $y = f(x_1, x_2)$ is a crisp function which is usually a polynomial function, and A_1 and A_2 are the fuzzy sets representing the antecent pairs.
A typical rule in a Sugeno fuzzy model has the form: If input 1 is 'x' and input 2 is 'y', then output is z = ax + by + c. For a zero-order Sugeno model, the output level 'z' is a constant (a = b = 0). Each rule weights its output level, z_i, by the firing strength of the rule, W_i. For example, for an AND rule with input 1 = x and input 2 = y, the firing strength is

$$W_i = AndMethod(F1(x), F2(y)) \tag{5.5}$$

where F1(x), F2(y) are the membership functions for inputs 1 and input 2 respectively.

The final output of the system is the weighted average of all rule outputs, computed as

$$Output = \frac{\sum_{i=1}^{N} W_i z_i}{\sum_{i=1}^{N} W_i} \tag{5.6}$$

where 'N' is the number of rules.

The main difference between Mamdani type FIS and Sugeno type FIS is in the way how the crisp output is generated from the fuzzy inputs. While Mamdani type FIS uses the technique

of defuzzification of a fuzzy output, Sugeno type FIS uses the weighted average of rule outputs to compute the crisp output. Hence, the Sugeno's output membership functions are either linear or constants. Sugeno fuzzy FIS is a more compact and computationally efficient system than a Mamdani system. Sugeno method incurs lesser processing time, since the weighted average replace the time consuming defuzzification process. The choice of membership functions, the range and the parameters associated with membership functions and the rules are a few major design decisions to be taken while designing fuzzy inference systems. The membership functions, parameters and the range associated with membership functions are to be selected based on the user's interpretation of the characteristics of the input signal. The parameters associated with a given membership function should be chosen so as to tailor the membership functions to account for the variations in the input data for a desired output. Also, the Sugeno system lends itself to the use of adaptive techniques such as Adaptive Neuro Fuzzy Inference Systems (ANFIS) which can be utilized to customize the membership functions of FIS so that the fuzzy system best models the data. Due to the above mentioned factors, a Sugeno fuzzy inference system is utilized for activity detection of the robot in this thesis.

5.4.1.4 Instantaneous Activity Detection

The conditioned gyroscope and accelerometer measurements along three axis, the heading information (θ_m) and the time at which the data values are read by the Arduino are transmitted to the base station laptop via a Bluetooth module. A Sugeno Fuzzy Inference System based activity recognition unit is designed for the instantaneous activity detection of the robot. As mentioned in Section 5.4.1.3, to design a Fuzzy Inference system, the set of input/outputs should be selected, the fuzzy rules should be defined and the inputs to FL system must be fuzzified using input membership functions. The activity recognition is based on the instantaneous values of magnetometer heading and gyroscope readings from the IMU mounted on the robot. The angular turn of the robot (θ_g) obtained from the z-axis reading of the gyroscope and (θ_m), the heading of the robot measured from the magnetometer (equation (5.1)) are selected as the inputs to the FL based activity detection unit. The heading of the robot ranges from 0-360°. The Fuzzy Inference System is simulated using MATLAB Fuzzy Logic tool box. The scope of the current work is to accurately localize the robot between the start and stop activities of the robot. For a such a system, triangular membership

would suffice to provide the reasonable accuracy. Triangular membership functions are also easier to implement on low-cost microcontroller boards.

The Sugeno Fuzzy FIS was designed based on the following principle. The FIS system is utilized to detect the instantaneous activity of the robot based on the receieved inputs. The FIS is used to detect if the robot is moving straight, whether the robot is turning right, turning left or whether robot has turned right or left. As the robot travels along the floor, the robot may slightly move to the right or left (deviate from straight line path) due to the factors such as the friction on the floor, slight imperfections in the odometry, etc. which can be observed from the gyroscope readings. The magnetometer readings are utilized to distinguish the unintended turns of the robot from the intended turns of the robot. The magnetometer (θ_m) reading is sampled when the robot starts turning left/right ('θ_g' increases in positive or negative direction). Also the magnetometer reading (θ_m) is sampled when the gyroscope readings returns back to zero. If there is a significant difference in the heading measurement at these two instances, then a valid turn is detected. Else, the turn observed by gyroscope is discarded as unintended turns. To implement this logic, five triangular membership functions are selected for the magnetometer input which are referred to as Low1 (L1), Low2 (L2), LowMotion1(LM1), LowMotion2 (LM2) and Turn as shown in Figure 5.5. The range of the magnetometer input varies from [0 360]°. Similarly, five triangular membership functions are selected for the gyroscope input. The membership functions are Very Low (VL), Low Left (LL), Low Right (LR), Right (R) and Left (L) as shown in Figure 5.6. The range of the gyroscope input is restricted to [-90 90]° as the robot is allowed to change the direction only through right angled movements. The output of the Sugeno fuzzy inference logic is configured to generate crisp constant outputs of the range [0 1] corresponding to the activities detected. The crisp values generated at the output of Sugeno Fuzzy logic on the detection of the different activities are as follows- Straight, Start Right, Start Left, Turn Right, Turn Left. The input and output membership functions and the parameter values of the membership functions are as shown in Table 5.1.

The rules for the fuzzy inference logic are defined using AND rules as shown in Table 5.2.

5.4.1.5 Activity Correlation Unit

The Sugeno Fuzzy Inference logic produces a constant output which correspond to the recognized activity, i.e, Start Right (output value=.2), Start Left (output value=.4), Turn Right (output value=.6),

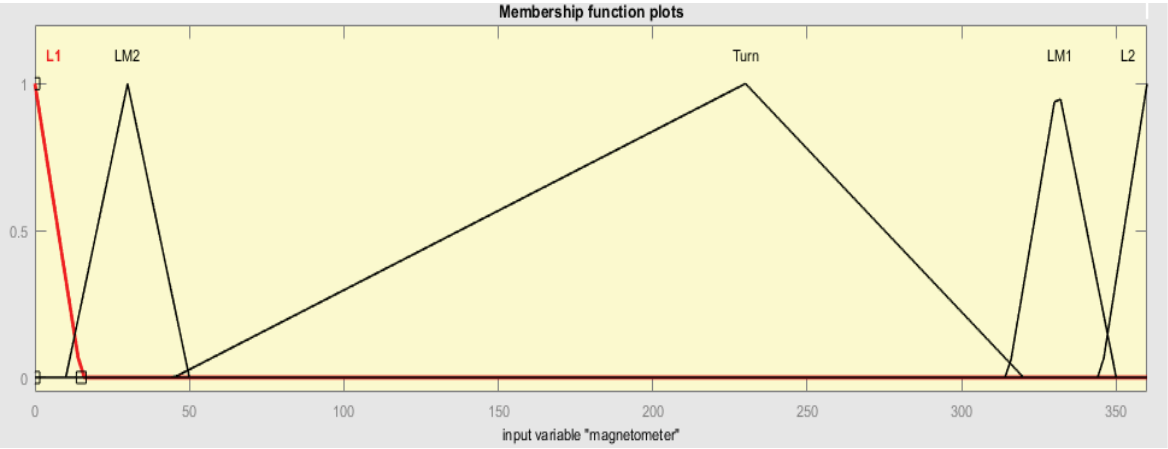

Figure 5.5. Membership function for the magnetometer input

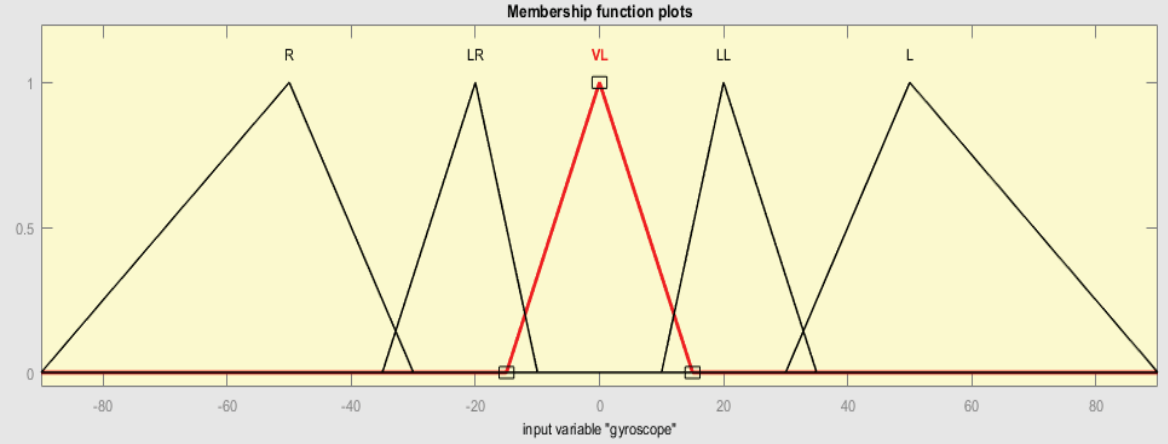

Figure 5.6. Membership function for the Gyroscope input

Turn Left (output value=1) and Straight (output value=0). Crisp outputs as mentioned above are generated when the inputs satisfy only one of the membership functions. For the inputs which satisfy two membership functions, the output is generated according to equation (5.6). However these output values are approximated to the closest value belonging to the 5 crisp outputs mentioned above. As the name of the constant membership functions from the FIS implies, the FIS logic will provide an output indicating whether the robot is moving straight, starts to turn left/right or is moving left/right. However, it is not possible to ascertain whether the robot has actually turned left/right or is moving straight, only from the instantaneous activity detection. The sequence of outputs from the FIS logic is stored in the activity store buffer as indicated in Figure 5.1. Whether the robot is in straight line motion, turning right or left or is in stop condition is ascertained only after correlation of the various outputs from fuzzy logic and the accelerometer readings as indicated

Table 5.1. Parameters for input and output membership functions

Gyroscope Membership Function	Gyroscope Parameter Range	Magnetometer Membership Function	Magnetometer Parameter	Output Constant Value	Parameter Value
VL	[-15 0 15]	L1	[0 0 15]	Straight	0
LL	[10 20 35]	L2	[345 360 360]	Start Left	0.4
LR	[-35 -20 -10]	LM1	[315 331 360]	Start Right	0.2
L	[30 50 90]	LM2	[10 30 50]	Left	1
R	[-90 -50 -30]	Turn	[45 230 320]	Right	0.6

Table 5.2. Rules for Sugeno Fuzzy Inference System

Gyroscope Input	Magnetometer Input	FIS Output
VL	L1	Straight
VL	L2	Straight
LL	LM1	Start Left
LL	LM2	Start Left
LL	Turn	Start Left
LR	LM1	Start Right
LR	LM2	Start Right
LR	Turn	Start Right
L	LM1	Left
L	LM2	Left
L	Turn	Left
R	LM1	Right
R	LM2	Right
R	Turn	Right

in Table 5.3. Similarly, the state 'Stop' is assigned only if the net acceleration along the 3-axis is lesser than 0.15 m/s^2 for at least 5 sampling intervals.

5.4.1.6 Velocity Estimation

The next steps in the localization are the velocity estimation and the position updation of the robot. The displacement of the robot is calculated by double integration of the acceleration. Hence, the

Table 5.3. Output state determined by the Activity Correlation unit

Accelerometer reading	**Sequence of outputs from FIS**	**Activity unit (output)**
<0.15	-	Stop (0)
-	Staight->Straight	Motion (1)
-	Straight->Start Left->Left->Start Left-> Straight	Left (3)
-	Straight->Start Right->Right->Start Right->Straight	Right (2)
-	Straight->Start Left->Start Left-> Straight	Motion (1)
-	Straight->Start Right->Start Right->Straight	Motion (1)

time at which the measurement of acceleration is made is very critical. It cannot be assumed that the sampling interval will remain same. Hence, along with the IMU readings, the Arduino module also transmits the timestamp at which the measurement was made. If any activity other than the 'stop' condition is detected, then the displacement of the robot relative to its starting point can be calculated if the velocity of robot is determined. The velocity is determined from the accelerometer readings of the IMU as follows. The net instantaneous acceleration 'a' of the robot is obtained from the accelerometer readings along x, y, z axis, i.e a_x, a_y and a_z as

$$a = \sqrt{a_x * a_x + a_y * a_y + a_z * a_z} \tag{5.7}$$

If Δ T is the interval at which the raw data from IMU is sampled, then the velocity $V_{(i)}$ is updated at every sampling interval based on the net acceleration 'a' is as follows.

$$V_{(i)} = (a_i - a_{i-1}) * \Delta T + V_{(i-1)} \tag{5.8}$$

In this work the Arduino module is programmed to sample IMU data at an interval of 0.2 seconds.

5.4.1.7 Position Estimation

Based on the four activities generated by the activity correlation unit (as indicated in Table 5.3), the position (x,y) and the velocity 'V' of the robot is updated as shown below.

Activity Detected by Activity Correlation unit = Stop

$$x_{(i)} = x_{(i-1)}; y_{(i)} = y_{(i-1)}; \tag{5.9}$$

$$V_{(i)} = 0; \tag{5.10}$$

Activity Detected by Activity Correlation unit = Straight

$$y_{(i)} = y_{(i-1)}; x_{(i)} = x_{(i-1)} + V * \Delta T; \tag{5.11}$$

or

$$x_{(i)} = x_{(i-1)}; y_{(i)} = y_{(i-1)} + V * \Delta T; \tag{5.12}$$

Activity Detected by Activity Correlation unit = Right or Left

$$y_{(i)} = y_{(i-1)}; x_{(i)} = x_{(i-1)} \pm (V * \Delta T * sin(\theta)); \tag{5.13}$$

or

$$x_{(i)} = x_{(i-1)}; y_{(i)} = y_{(i-1)} \pm (V * \Delta T * sin(\theta)); \tag{5.14}$$

where 'θ' is the angle turned by robot. In this work if the 'Right' or 'Left' activity is detected by the activity correlation unit, then 'θ' is assumed to be 90°.

5.5 Experimental Validation of the IMU based Self-Localization

The accuracy of the proposed system was tested in a indoor environment. Two specific test cases are described in this section to illustrate the accuracy of the proposed scheme.

5.5.1 Validation of Self-Localization- Test Scenario-1

The robot was programmed as a black line follower robot, i.e to follow a black-line drawn along the path to be travelled by the robot. The robot was programmed to travel from one room to

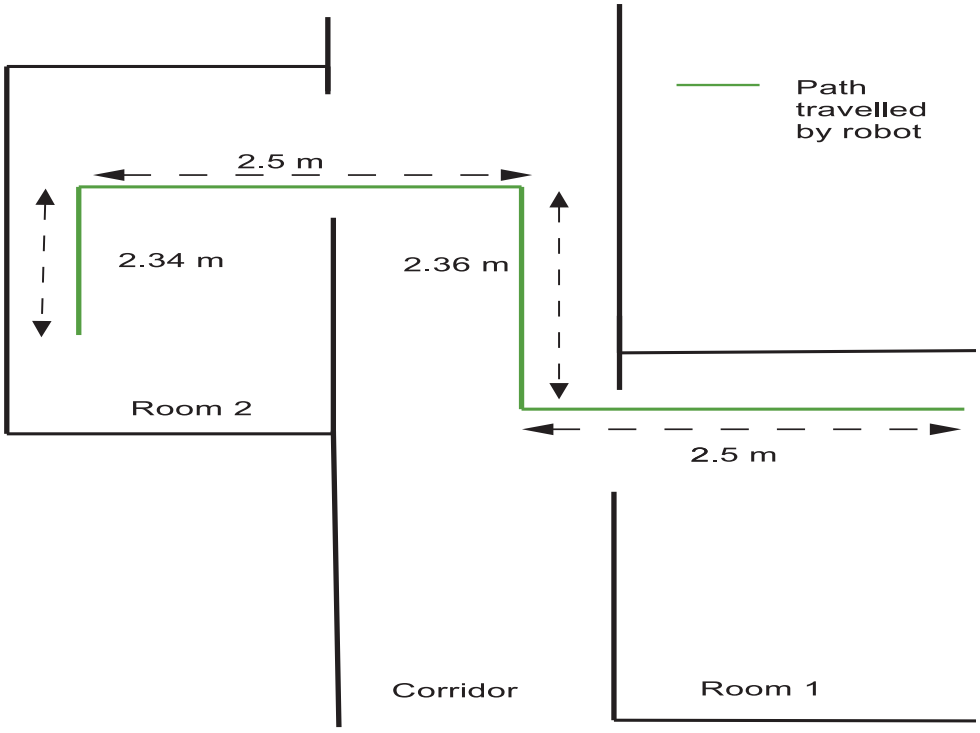

Figure 5.7. Test scenario-1 for validation of Self-localization

another in an indoor environment. The desired path to be followed by robot is shown in Figure 5.7. The conditioned accelerometer, gyroscope, magnetometer readings from the IMU along with the time at which the mesaurement were made is transmitted by the Arduino via Bluetooth module and the path traveled by the robot is traced using MATLAB. The path travelled by the robot and the estimated path is shown in Figure 5.8. Plot D and Plot E of of Figure 5.9 indicates the various activities detected by the Instantaneous activity detection logic -Straight (output value=0), Start Right (output value=0.2), Start Left (output value=0.4), Turn Right (output value=0.6), Turn Left (output value=1). and the activity correlation logic -Stop (output value=0), Straight (output value=1), Left (output value=3), Right (output value=2) respectively. Plot A of Figure 5.10, depicts the x-axis coordinates and y-axis coordinates determined by the robot. Plot B of Figure 5.10 depicts the angular turns traced by the robots to cover the path. Plot A of Figure 5.11 presents the measured velocity of the robot. It can be observed from Plot A and B of Figure 5.10, that whenever the robot turns by 90 degrees, the corresponding x-axis and y-axis distances are updated by the robot.

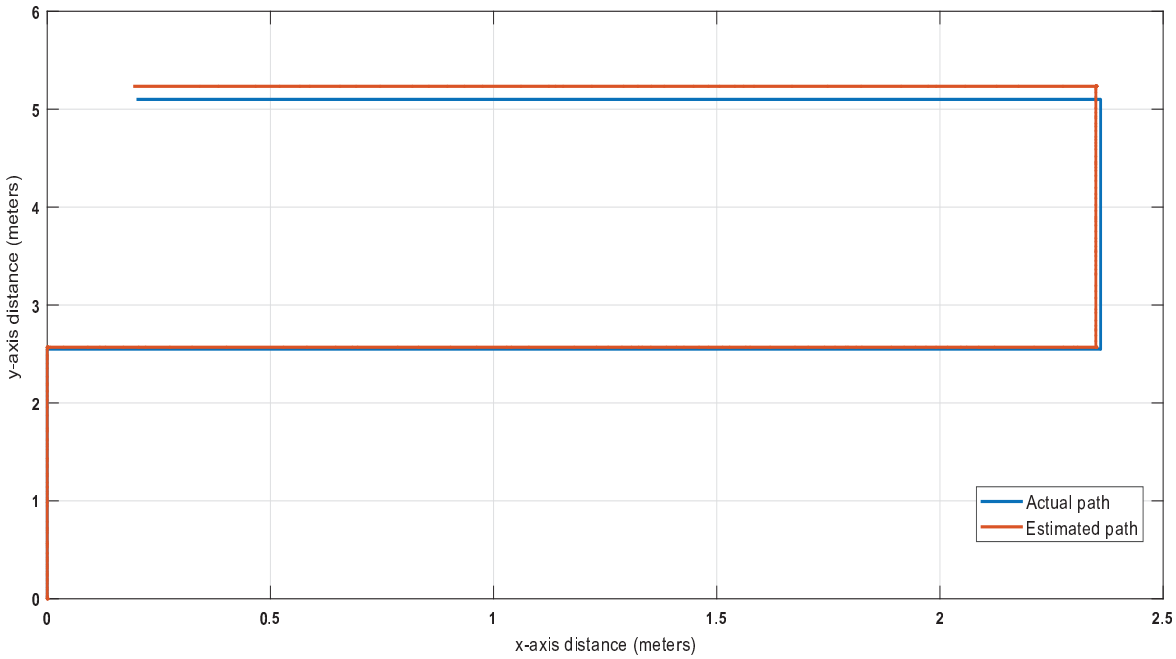

Figure 5.8. Path estimated by the self-localization scheme and the Path traced by robot for Test scenario-1

The experiment was repeated 10 times and the maximum error in position along the x and y-axis reported for Scenario -1 less than to 8cm.

5.5.2 Validation of Self-Localization- Test Scenario-2

The self-localization scheme was also tested on a robot deployed in a 3m x 3m room. The robot was programmed to move along a preplanned path as shown in the Figure 5.12. The robot was programmed to operate as a black-line following robot in a rectangular section of area 2x1.5 m. As in Test Scenario-1, the conditioned accelerometer, gyroscope, magnetometer readings from the IMU along with the time at which the mesaurement was made is transmitted by the Arduino via Bluetooth module and the path traveled by the robot is traced using MATLAB. The path traversed by the robot and the estimated path is shown in Figure 5.12. Plot A of Figure 5.13 depicts the x-axis coordinates and y-axis coordinates determined by the robot. Figure 5.13 Plot B depicts the angular turns traced by the robots to cover the path. Plot A of Figure 5.14 depicts the estimated velocity of the robot. It can be observed from Plot A and B of Figure 5.13 that whenever the robot turns by 90 degrees, the corresponding x-axis and y-axis distances are updated by the robot. The

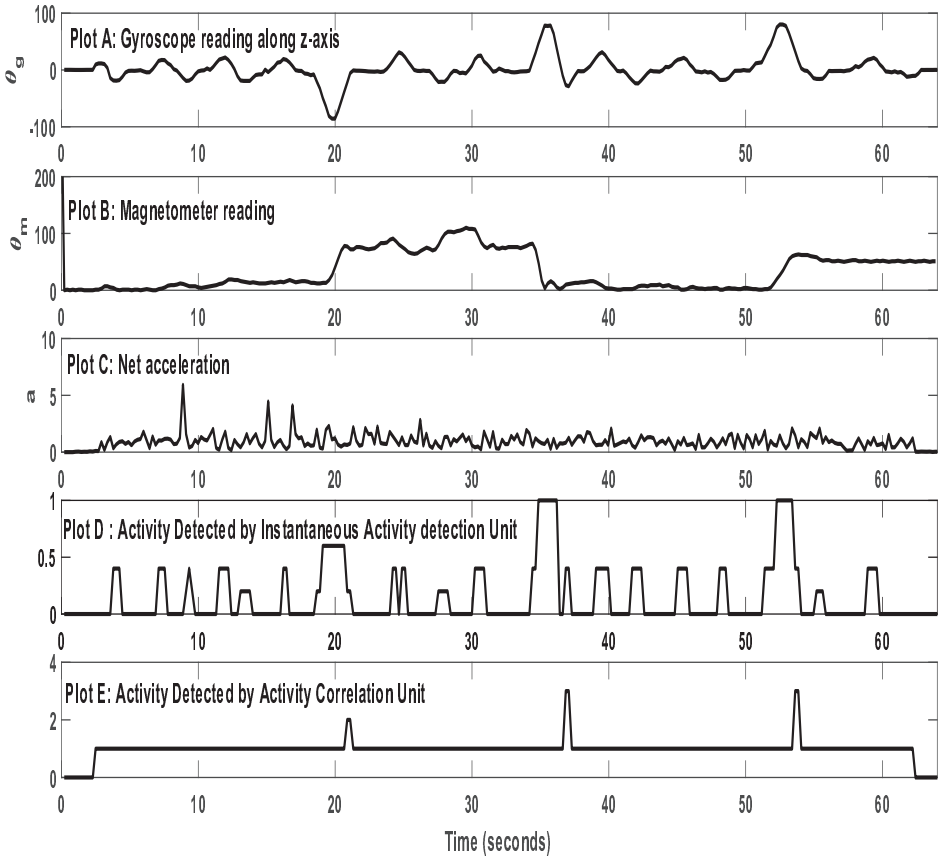

Figure 5.9. Activity detection for Test Scenario-1

experiment was repeated 10 times and the maximum error in position reported in x-axis and y-axis was less than 10cm for all trials. Since the localization was performed using timestamps obtained at an interval of 0.2 seconds, it also indicates the hardware realization of the proposed logic is feasible.

5.6 Conclusions

In this chapter, the design of an IMU based short-term, self-localization scheme for robots is presented. The major conclusions based on the work presented in this chapter can be summarized as follows.

- IMU based localization is a feasible solution for short-term indoor localization.

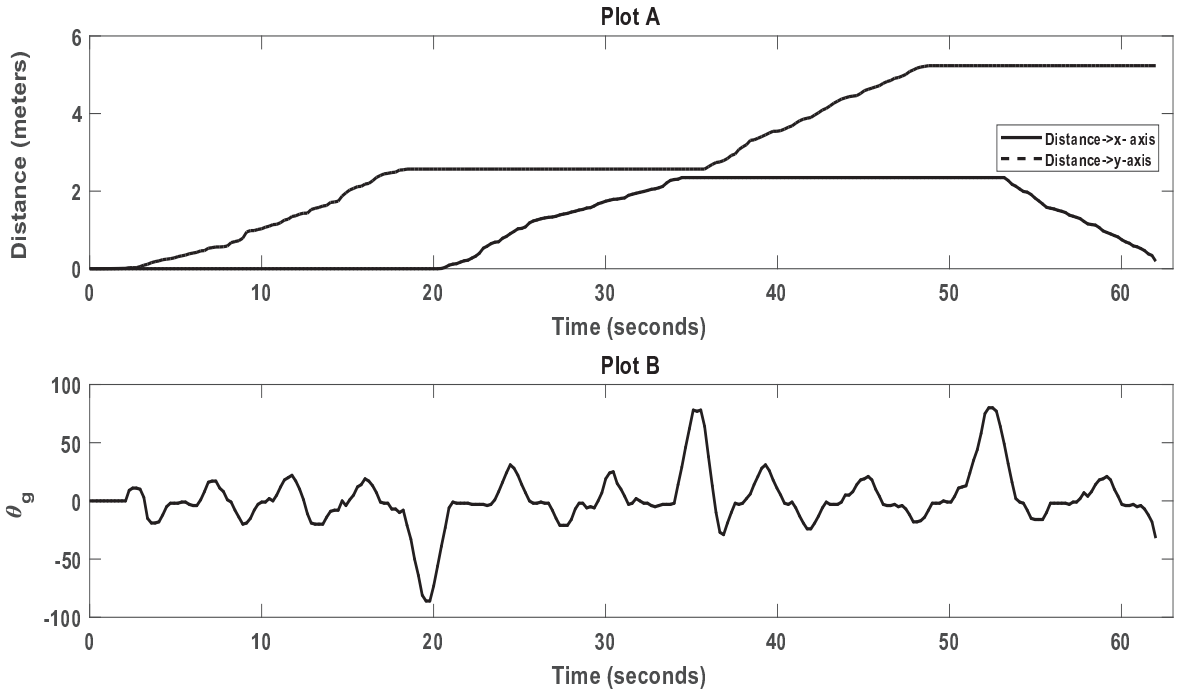

Figure 5.10. Measured x-axis and y-axis displacement of robot- Test scenario 1

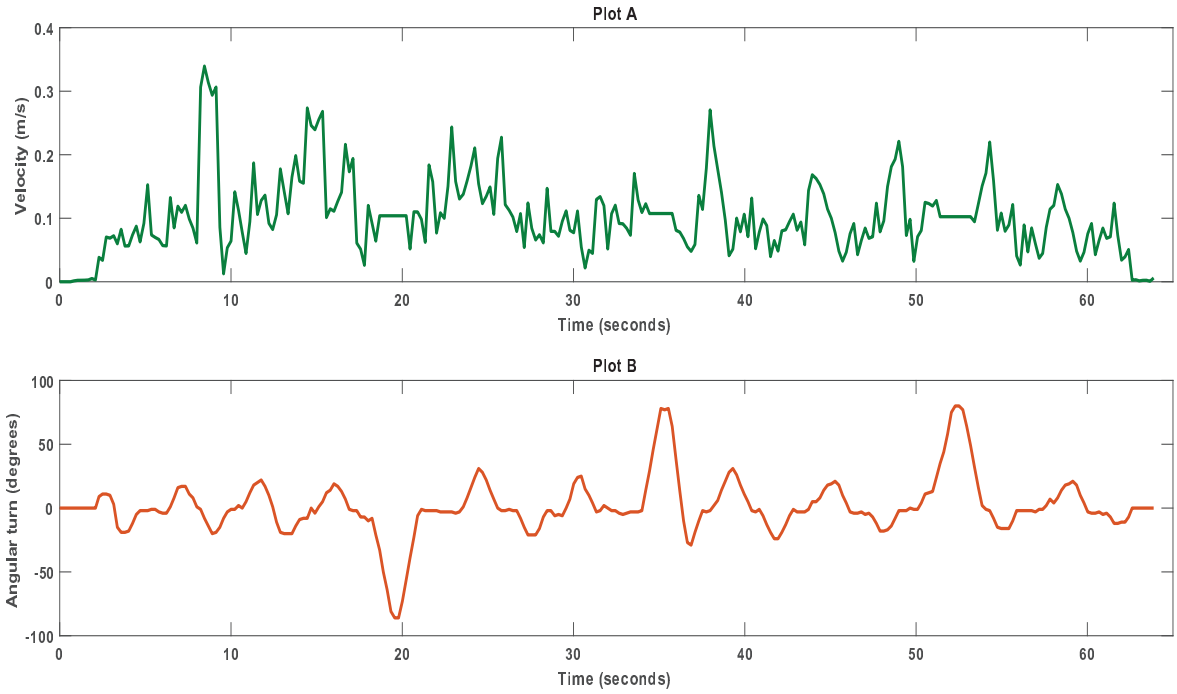

Figure 5.11. Measured velocity and angular turns of the robot for Test Scenario-1

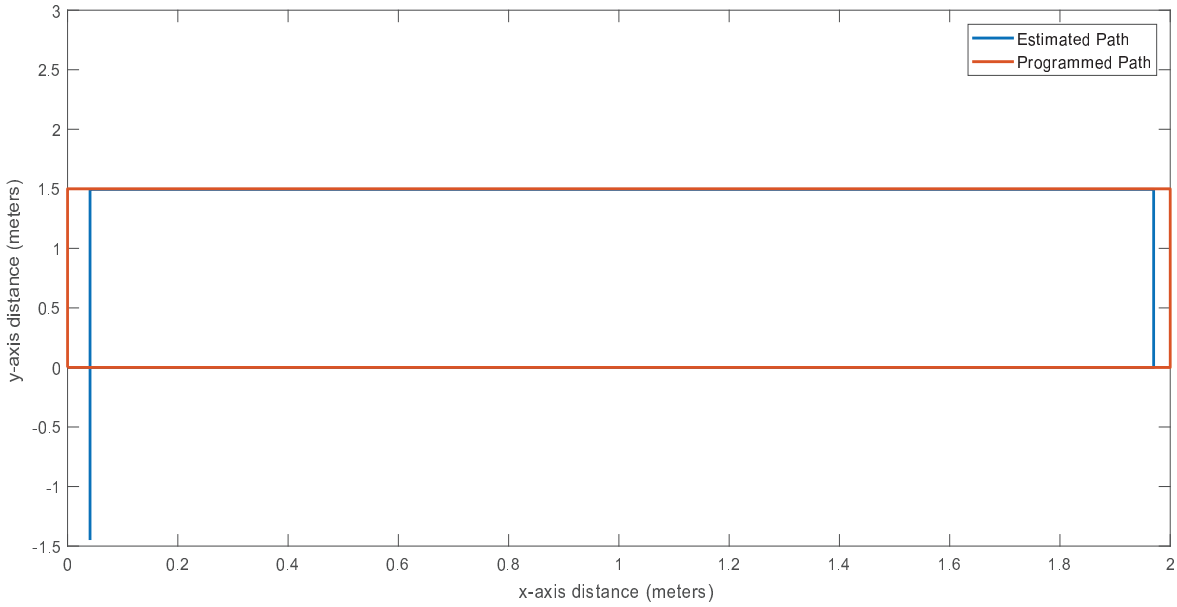

Figure 5.12. Actual and Estimated path of robot for Test Scenario 2

- To perform the crucial step in self-localization, i.e the activity detection, a Sugeno-Fuzzy Logic based activity detection unit was designed.
- It was observed during experimental validation that the maximum error in position estimation during these observations were restricted to 10 cm along x or y-direction for distances upto 10.7 meters travelled by the robot. Hence the proposed scheme can be utilized for reliable self-localization in indoor environments for short distance localization.
- The results obtained during the experiments are promising enough to extend the IMU-only localization for applications which involves more complex type of robot movements.
- The beacon-based localization scheme proposed in Chapter 4, along with the self-localization proposed in Chapter 5 together can be utilized as a cost-effective solution for accurate localization of robots in indoor environments. Since the robots can self-localize for short distances, the wireless channel can be optimally utilized for other robotic tasks.

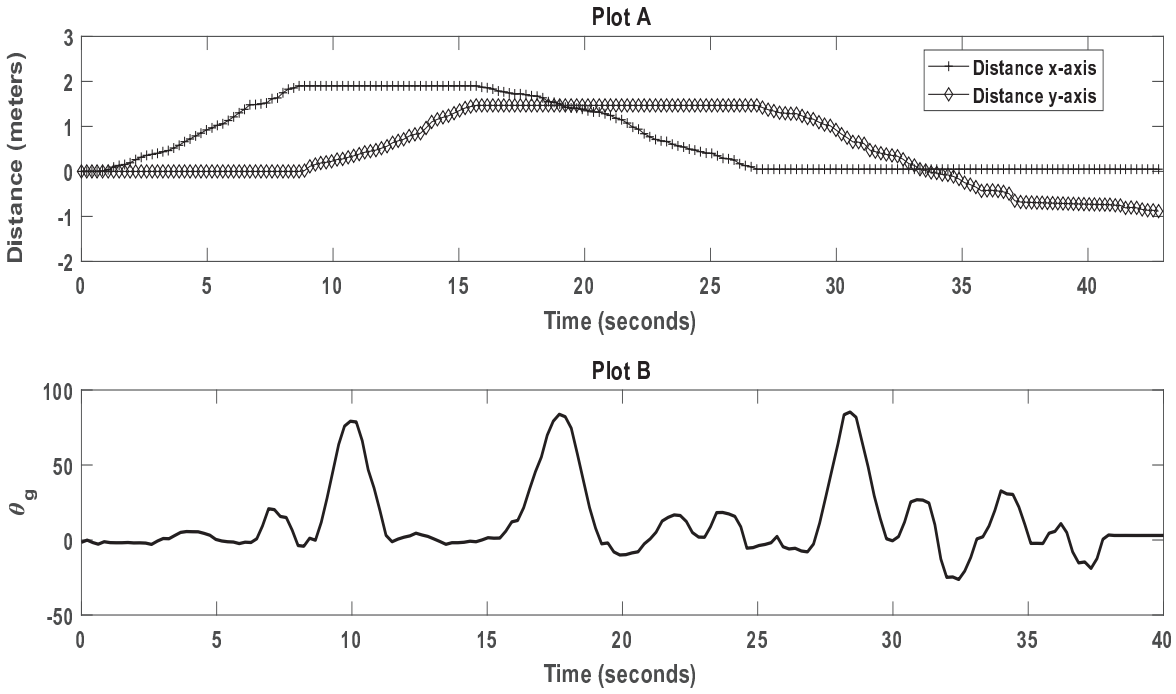

Figure 5.13. Measured x-axis and y-axis displacement of robot for Test scenario 2

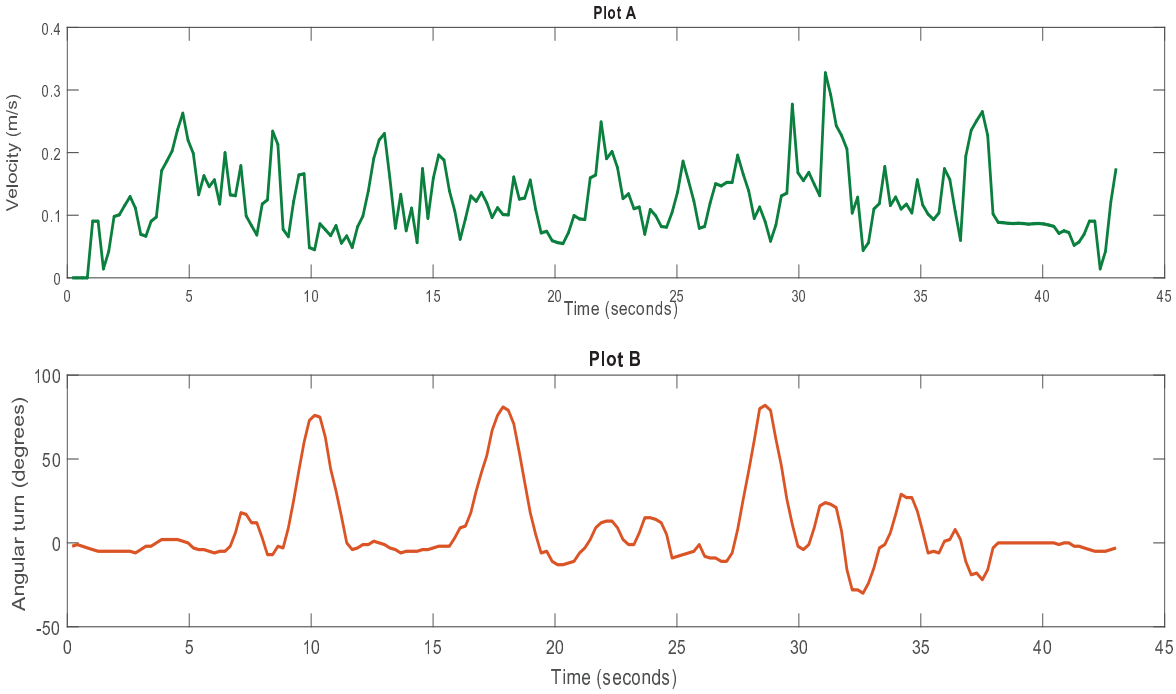

Figure 5.14. Velocity and angular turns of the robot measured for Test Scenario-2