A ROBUST APPROACH FOR IMPROVING THE ACCURACY OF IMU BASED LOCALIZATION

A PROJECT REPORT

submitted by

CB.EN.U4CSE12123 KIRAN KASSYAP S

CB.EN.U4CSE12144 SRIVENKATA KRISHNAN S

CB.EN.U4CSE12147 SUNDARRAJAN G

in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE 641112

May, 2016

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ENGINEERING, COIMBATORE-641112

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "**A ROBUST APPROACH FOR IMPROVING THE ACCURACY OF IMU BASED LOCALIZATION"** submitted by **KIRAN KASSYAP S (CB.EN.U4CSE12123), SRIVENKATA KRISHNAN S (CB.EN.U4CSE12144), SUNDARRAJAN G (CB.EN.U4CSE12147)** in partial fulfilment of the requirements for the award of the **Degree of BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE ENGINEERING** is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Engineering.

Guide

Dr. Vidhya Balasubramanian
Associate Professor,
Department of CSE

Dr. Latha Parameswaran
Chairman, Department of CSE

This project report was evaluated by us on …………

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our gratitude to all who have helped us directly or indirectly in our project. To start with, we thank the Almighty for all his blessings showered on us during the tenure of our project.

We express our sincere gratitude to Brahmachari Abhayamrita Chaitanya, Pro Chancellor, Dr.P.Venkat Rangan, Vice Chancellor and Dr Sasangan Ramathan Dean Engineering of Amrita Vishwa Vidyapeetham for providing us the opportunity to undergo this programme.

We express our sincere thanks to Prof. Dr. Latha Parameswaran, Chairperson, Department of Computer Science and Engineering, for her support and encouragement.

We extend our heartiest gratitude to my guide Dr. Vidhya Balasubramanian for her valuable guidance. We also thank our project coordinators, for their co-operation. We are also grateful to all other members of faculty for their valuable guidance.

Finally, we wish to express our sincere thanks to our parents and our friends who have contributed a lot towards our project work and our mental well-being during this period.

Abstract

Indoor localization is a vital part of autonomous robots. Obtaining accurate indoor localization is difficult in challenging indoor environments where external infrastructures are unreliable and maps keep changing. In such cases the robot should be able to localize using their on board sensors. IMU sensors are most suitable due to their cost effectiveness. We propose a novel approach that aims to improve the accuracy of IMU based robotic localization by analysing the performance of gyroscope and encoders under different scenarios, and integrating them by exploiting their advantages. In addition, the angle computed by robots to avoid obstacles is used as an additional parameter and appropriately integrated using a complementary filter. Our experiments that evaluated the robot over different trajectories demonstrated that our approach improves the accuracy of localization over applicable existing techniques for above scenarios.

Table of Contents

List of Figures and Tables:

List of Abbreviations:

ABBREVIATIONS	EXPANDED FORM
IMU	Inertial Measurement Unit
Lipo	Lithium polymer
RSSI	Received Signal Strength Indication
RFID	Radio-Frequency Identification
CDA-nO	Curvature based Decision Approach(without Obstacle)
CDA-O	Curvature based Decision Approach(with Obstacle)

CHAPTER 1

INTRODUCTION

1. Introduction

Robotic indoor localization is a method in which the position and orientation of the mobile robot is determined with respect to the indoor environment and is an important part of any autonomous mobile robot. Autonomous robot systems are commonly being used in disaster response and rescue, in industries, as assistive robots etc. In order to support the effective functioning of the robots in such scenarios there is a need for accurate and efficient indoor localization, navigation and mapping methods. One of the primitive challenges in indoor environments is localization and this supports the other two functionalities. In this paper we primarily focus on indoor localization of mobile robots.

Over the past decade, there have been significant developments in the mobile robot localization. Several approaches have been used, which include infrastructure and non-infrastructure based approaches. Infrastructure based approaches use existing WiFi or RFID [4] installations to help localize the robot. These approaches become infeasible in challenging environments such as disasters, chemical spills etc. where there is no available infrastructure. Approaches that do not rely on infrastructure use laser range finders, camera and inertial sensors [5] for localization of the robot. Laser range finder based approaches provide accuracy, but are very expensive. On the other hand, camera based approaches are computationally complex and performs poorly in low light conditions and presence of occlusions.

The use of inertial sensors and digital encoders [10] for localization have gained more popularity because of its low cost, small dimensions and low power consumption. These sensors are commonly used in dead reckoning based localization methods. The dead reckoning method computes new state estimates with the help of previous state estimates. Accelerometers and encoders have been used for distance estimates while gyroscopes and magnetometer sensors have been used for orientation estimates. The accuracy of these methods are affected due to accumulation of noise and drift errors from accelerometers and gyroscopes respectively. Digital encoders tend to produce errors because of slippage, mechanism symmetry and sensor accuracy limitations. To overcome these errors, sensor fusion [7] techniques have been proposed.

Sensor fusion is the process in which data from multiple sensors are fused to a combined estimate, resulting in a better position and orientation values. Over the years there have been several sensor fusion approaches that varies from simple aggregation to complex systems which uses either of Kalman filter, extended Kalman filter, complimentary filter etc. [13]. The performance of filters deteriorates when we solely make use of inertial sensors as these depend on the parameters such as sensor bias, drift which are context-specific (dependent on environment, device). Therefore, it poses a great challenge to the widespread applications that uses these sensors and localization methods.

We propose an approach that effectively combines the advantages of the encoders and gyroscopes to improve the accuracy of localization of the robot. While existing techniques like gyrodometry [11] have been developed, which use Kalman Filters for sensor fusion, their performance is affected when accelerometers are ineffective (as in the case of wheeled robots). Our approach exploits the curvature of the robot's trajectory to determine when the gyroscopes and encoders are used. This helps limit the continuous use of the gyroscope thereby reducing drift errors. In addition, we exploit the obstacle avoidance capabilities of autonomous robots to provide another source of orientation estimate. This is combined with the gyroscope's orientation estimate using complementary filter to improve the localization accuracy.

CHAPTER 2

EXISTING METHODOLOGIES

2. Existing Methodologies

The fundamental requirement for indoor mobile robots is localization. Many localization techniques have been developed and implemented in robots over a period of time. Common indoor localization methods for robots rely on external infrastructures and sensors which are part of the robot. Localization techniques use RF technologies such as RFID, Wi-Fi [4] and Bluetooth, while other prevalent techniques that operates with infrared and ultrasonic sensors rely on additional external infrastructure like RF antennas and ultrasonic transceivers placed in the environment. On the other hand, techniques that use cameras, laser range finders or inertial sensors [5] do not rely on external infrastructure.

In the infrastructure based approaches, RF based localization is one of the most prevalent techniques and is easy to implement. In RF based approaches, the robot requires external infrastructure like antennas/reference tags placed in the environment for communication with the tags or receiver which is carried by the robot. RF based localization techniques can be either finger printing or non-finger printing approaches. The non-finger printing approaches include the trilateration method. The trilateration method estimates the position of the target with respect to reference points using the Received Signal Strength Indication (RSSI) [6] values. This method suffers from accuracy.

The Wi-Fi fingerprinting technique compares the RSSI [6] observations made by the mobile node with a trained database to determine the location of the moving object. The K-NN (K Nearest Neighbours) [14], Bayesian and decision tree methods are representative techniques used in the fingerprinting approach. In RFID based localization, a large number of RFID tags in the environment act as reference points. So, when a new RFID tag enters the space, the signal strength is compared with the reference points signal strength and location of the robot is determined. In both RFID and Wi-Fi finger printing approaches, the collection of the data set (offline phase) is extremely tedious and time consuming.

The techniques that do not rely on infrastructure involve the use of laser range finders, cameras and inertial sensors [5]. Camera based approaches use the entire visual information as input or interest points or local features or combination of all these. These methods are usually prone to occlusions, changes in scale, rotation and illumination. In laser based approaches, lasers have been used to emit pulses that are reflected off a rotating mirror from which time of flight is determined and used to calculate distances. The main issue here is that the laser range finders are expensive in this method.

The low-cost sensors such as accelerometer, gyroscope, encoders have been used for localization. These IMU sensors are commonly used in dead reckoning method. The dead reckoning method is the method in which new state estimates are calculated with the help of old state estimates. The accelerometer and encoders have been used to find the distance while gyroscope and magnetometer sensors have been used to determine the orientation. These methods do not give accurate results because accelerometer suffers from noise error, gyroscope suffers from static drift and magnetometers are error-prone in places where magnetic interferences are high. Using only one of these sensors is insufficient to provide desired results, since each of them have high error rates. To overcome these errors, sensor fusion based approaches have been proposed.

The sensor fusion in this context is the translation of different sensory inputs into reliable estimates. The types of sensor fusion are: complimentary, competitive and cooperative. In complimentary fusion, sensors which are not dependent on each other are fused to give an accurate estimate. For example, positioning of cameras at distinct points of the room and combining them to give us an estimate. The competitive approach fuses sensors which gives similar information. Example of this approach is fusing the orientation estimate obtained from both encoders and gyroscope. The cooperative approach combines sensors to get a new estimate which cannot be obtained by using a single sensor alone. Example is using the output from the cameras placed at distinct viewpoints to obtain a three dimensional view. The complexity of the environment and the errors from the sensors make the robot localization problem ill posed.

Therefore, we need a better method that is able to effectively localize the robot with minimum overhead and no external infrastructure. In addition, it must be able to exploit the advantage of different sensors to improve accuracy and without relying on maps. We aim to achieve this using a novel decision based approach that uses the scenarios where different sensors work accurately. The next sections will describe our approach for improving the localization accuracy of indoor robots using inertial measurement units[2].

CHAPTER 3

DESIGN AND ARCHITECHURE

3. Design and Architecture

3.1 Design of the robot:

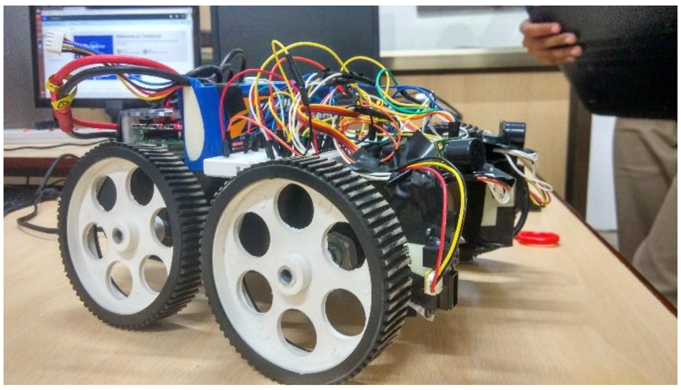


Figure 1:Design of Robot

An image of our robot is shown above in Fig 10. Our robot is a four wheeled autonomous light weight vehicle and has a dimension of 105mm*55mm*57mm. Each of the 4 wheels has a diameter of 110mm. A metallic alloy chassis is used to assemble the robot. It is equipped with bStem single chip computer, an Arduino micro controller, sensors and motor drivers. The environment in which the robot navigates is a partially known environment, in which walls and corridors are known but any furniture or moving obstacles are unknown. However, for the purpose of our localization we do not use any prior map information including walls.

The ultrasonic sensors are placed on top and Infrared sensors are place just below ultrasonic sensors. Ultrasonic sensors are for detecting long range obstacles. It can sense up to 700cm distance. Infrared sensors are for detecting short range obstacles less than 20cm. Hence with the sensor fusion we implement obstacle avoidance in the robot, where bStem acts as the primary processor for computing and controlling motors accordingly.

3.2 Hardware Specifications:

3.2.1 bStem and AdCord:

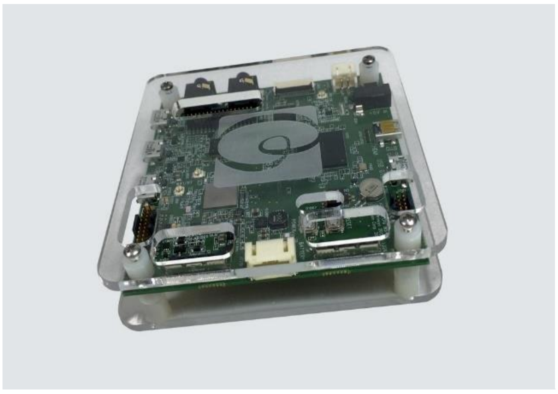


Figure 2: bStem single chip computer

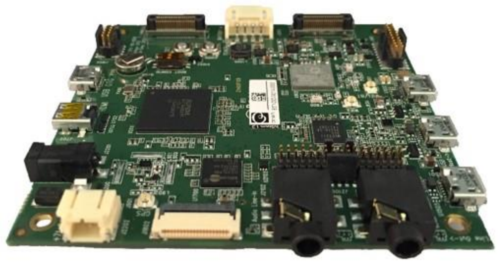


Figure 3 : Adcord extension board(motors)

bStem [1] is a single chip computer with a Qualcomm Snapdragon S4-Pro processor designed to be an advanced brain for robots. bStem has a wide array of sensors and programmable logic with a full desktop Ubuntu Linux distribution. Sensors and motor control functionality have been tuned for accurate timing, and the on-board FPGA (field programmable gate array) provides low-level control and connectivity for many existing robotic standards. We can program bStem on and off the board using a Python and C++ API, which provides an easy interface to all sensors and controls.

AdCord (fig 3.) is designed for wheeled platforms where precise DC motor control with quadrature feedback is required. It also provides 8 RC servo outputs. It acts as a bridge between bStem and DC motors.

bStem runs on BRAIN OS which controls the hardware present in the chip. It is interfaced to an Arduino board with an USB cable for sensor data transmission.

3.2.2 Arduino Uno Microcontroller:

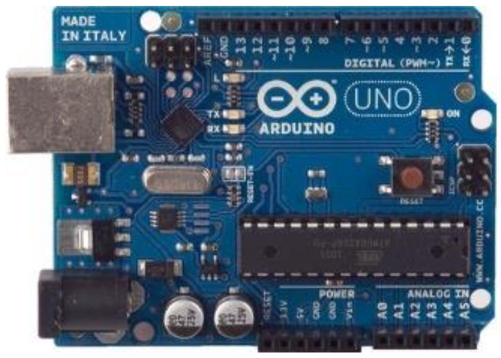


Figure 4:Arduino Microcontroller

Arduino Uno[7] is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It acts as an interface between bStem board and range finder sensors which will be explained later.

3.2.3 Stepper motors and encoders:

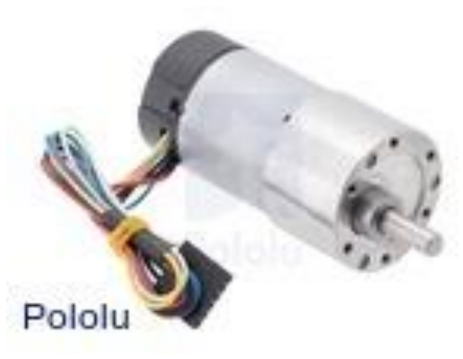


Figure 5:Stepper motors with integrated encoders

This gear motor is a powerful brushed DC motor with 30:1 metal gearbox intended for operation at 12 V. These units have a 0.61"-long, 6 mm-diameter D-shaped output shaft. It runs at a velocity of 350rpm and also has integrated digital encoders.

3.2.4 Lithium polymer battery:

Figure 6: Lipo battery

The robot is powered by an on-board 12V battery with 5000mah capacity and a discharge rate of 30C, which could power the robot and its components up to 4 hours.

3.2.5 Range finder sensors:

- Ultrasonic sensors:

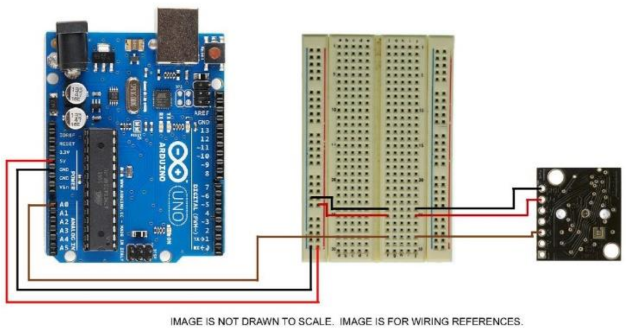


IMAGE IS NOT DRAWN TO SCALE. IMAGE IS FOR WIRING REFERENCES.

Figure 7 : Ultrasonic sensor *Figure 8: Circuit connection for integrating Arduino and ultrasonic sensor*

The ultra-sonic sensor(fig.7) detects objects from 20-cm to 765cm (25.1 feet) and provide sonar range information from 20-cm out to 765-cm with 1-cm resolution.

Hence we could identify objects within 25 feet range. Figure 8 shows the circuit diagram used to connect Arduino and the sensor.

- Infrared sensors:

Figure 9 : Infrared sensor

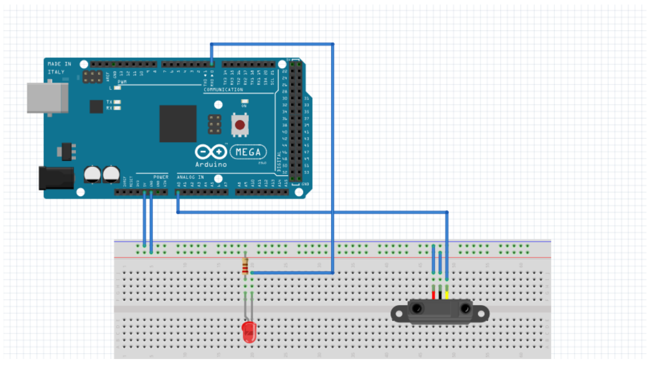


Figure 10 : Circuit connection for Infrared sensor and Arduino

Infrared distance measuring sensor unit capable of measuring distance from 4 to 30cm and is primarily used for detecting nearby obstacles. It is composed of an integrated combination of IRED (infrared emitting diode) and signal processing circuit to determine distance accurately operating at 5V DC. Figure 10 shows the circuit connection for Arduino and IR sensor.

CHAPTER 4

CALCULATION OF POSITION AND ORIENTATION ESTIMATES

4. Calculation of position and orientation estimates

The robot is defined by its pose p which is given at time t as,

$$p = [x_{t,} \, y_t, \Theta_t]$$

where $x_{t,} \, y_t$ represents the position estimate and Θ_t represents the orientation estimate.

To measure the distance travelled by the robot, two quadrature encoders associated with the gear motors are added to the rear wheels of the robot. These encoders output the left and right ticks (l, r), using which the distance travelled by the robot is calculated. It also provides us with an estimate of the robot's orientation Θ_e. The tri-axial gyroscope present in bStem provides the angular rates with respect to each axis. Using the z-axis angular rate we calculate the orientation angle Θ_g. Two ultrasonic and two infrared sensors are embedded on either sides in-front of the robot. Both the sensors provide distance between obstacle and robot. This data serves to provide us with an additional orientation estimate. We use the encoders to determine the distance estimate and a combination of gyroscope, encoders and obstacle avoidance sensors for orientation estimate to effectively determine pose.

The mathematical determination of position and orientation estimate from encoders, gyroscope and obstacle avoidance sensors are explained in this chapter.

4.1. Position and Orientation estimate using Encoders:

The encoders provide ticks l and r from left and right motors respectively. The width of the robot is defined as W. Using these, the position and orientation is calculated by applying geometric techniques. In this technique the pose is estimated for the following two cases:

- Robot is taking turns.
- Robot is moving straight.

4.1.1 Determination of position when the robot is taking turns

Figure 11 represents the situation where the robot is taking a left turn. Let *p'* be the previous position of robot and p be the current position of the bot, we need to calculate the orientation and position. The angle α represents the degree of turn taken by the robot to traverse from position *p* to *p'* . α is calculated as follows,

In figure 11, when the arc is extended, it results in a circle with centre at C and radius R units. Practically, *l* is ticks given by left encoder to traverse from A to A' and *r* is the ticks given by right encoder to traverse from B to B'.

Mathematically, *l* is determined by considering *A'AC*,

$$l = \alpha R$$

Similarly, *r* is determined by considering *B'BC,*

$$r = \alpha(R + W)$$

Using the mathematically obtained *l* and *r, α* is given as,

$$\alpha = \frac{(r - l)}{W}$$

The radius of the bot is proportional to *l* and inversely proportional to α,

$$R = \frac{l}{\alpha}$$

Now in figure 12, at point *p*, the direction vector representation of Θ_t is determined. The direction changes when the bot moves straight or turns right.

The new orientation estimate is summation of Θ at time *t-1* and calculated α value and is given as,

$$\Theta_t = \Theta_{t-1} + \alpha$$

In order to obtain position estimates x_t, y_t, the centre C(C_x, C_y) must be determined. The centre is calculated with respect to the direction vector at point *p'*.

$$C_x = x_{t-1} - \left(R + \left(\frac{W}{2}\right)\right) \sin \Theta_{t-1}$$

$$C_y = y_{t-1} - \left(R + \left(\frac{W}{2}\right)\right) (-\cos \Theta_{t-1})$$

Now, the new position estimates is determined using the centre calculated previously,

$$x_t = C_x + \left(R + \left(\frac{W}{2}\right)\right) \sin \Theta_t$$

$$y_t = C_y + \left(R + \left(\frac{W}{2}\right)\right) (-\cos \Theta_t)$$

The above calculated orientation and position estimates are used to determine pose at time t.

$$p = [x_{t,} y_t, \Theta_t]$$

The above pose is calculated considering the bot turns left. Similarly, this can be extended to calculating the new pose when robot turns right.

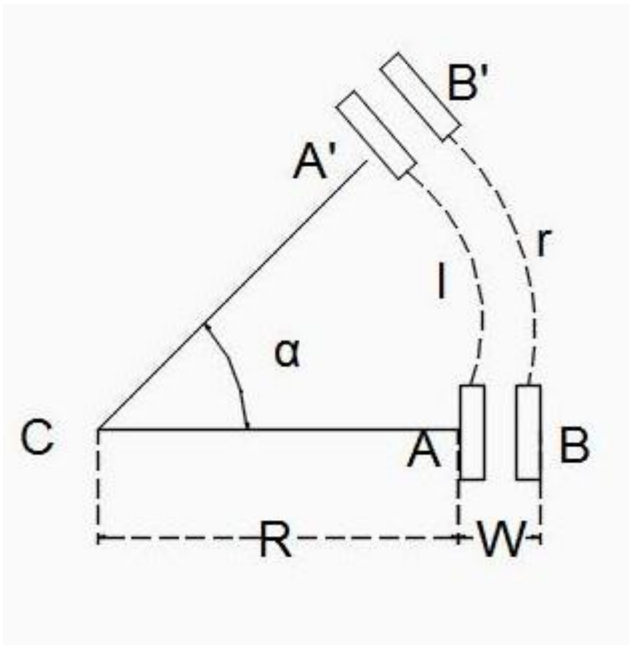


Figure 11 : When the robot is taking a left turn

4.1.2 Determination of pose when the robot is moving straight:

When the robot is moving straight, there is no change in orientation of the robot. So, new orientation is same as orientation at time *t-1*.

$$\Theta_t = \Theta_{t-1}$$

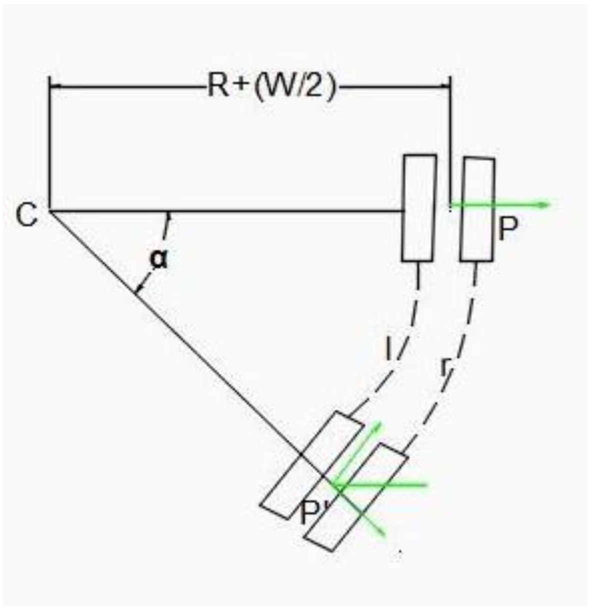


Figure 12 : When the robot is taking a left turn

Now, the corresponding $x_{t,}\ y_t$ is calculate with previous position estimates ,l , r is given as

$$x_t = x_{t-1} + l\cos\Theta_t$$

$$y_t = y_{t-1} + l\sin\Theta_t$$

Here, both left and right tick values are equal

4.2 Position and Orientation estimate using Gyroscope:

Gyroscope is a sensor used to measure angular velocity. In general, angular velocity is the change in rotational angle per unit time, which is expressed in radians per second(rad/sec). In our robot, the L3GD20 MEMS motion sensor 3-axis digital gyroscope which is part of the bStem, provides angular velocities along three axes x, y, z respectively. The sensitivity of the gyroscope is set at 250dps, ideal for a robotic system that undergoes heavy vibrations due to motor and chassis movement.

The goal is to calculate the orientation angle of the robot over a period of time, which is done by integration of angular velocity about the z-axis over a certain time interval. In order to get reliable angle estimates we first calculate and eliminate the *offset* and noise of our gyroscope. The *offset* is calculated as an average of angular velocities when the robot is at rest.

$$offset = \frac{1}{N}\sum_{i=0}^{N}\Omega$$

Where Ω is angular velocity at that instant of time and N represents the number of angular velocity values taken. The offset is calculated separately for the three axes.

The noise is characterized by the maximum difference between angular velocity Ω and *offset* for values in which robot is at rest.

$$noise = max(\Omega_i - offset) \quad where\ 0 \leq i \leq N$$

The angular velocities which fall in the range of [-noise, noise] are ignored and the remaining values are taken for the calculation of orientation. The orientation is the integration of angular rates over an interval of time. The orientation angle $\Theta_g\,(t)$ at time t is given by,

$$\Theta_g\,(t) = \Theta_g\,(t-1) + \Omega\,dt$$

where *dt* represents the time period over which Ω can be periodically integrated to determine the angle, in our case this value is equal to 20ms. Using the above $\Theta_g\,(t)$ value we estimate the orientation and position.

The position estimate is calculated using previous position values, the encoder values and the newly calculated orientation value and is determined as follows.

$$dist = \frac{(l+r)}{2}$$

$$x(t) = x(t-1) + dist(\cos(\Theta_g\,(t)))$$

$$y(t) = y(t-1) + dist(\sin(\Theta_g\,(t)))$$

Where x(t-1) and y(t-1) represents previous estimated values

4.3 Orientation estimate using obstacle avoidance sensors:

The distance between obstacle and robot is estimated by the ultrasonic and infrared sensors. The sensor specifications are given below,

- Ultrasonic Range Finder XL-EZ3 (MB1230) which has a range of 20cm to 760cm and a resolution of 1cm is used to detect any obstacles present in long range.
- Sharp GP2Y0A41SK0F IR Distance Sensor which has a range of 4cm to 30cm is used to determine obstacles in close range. These infrared sensors have faster response time than ultrasonic sensors and are used in addition.

A robot that is moving autonomously takes navigational decisions based on the output from the above sensors. An obstacle avoidance method [19], is adapted to calculate the orientation of the robot as explained below.

When an obstacle is detected, the two ultrasonic sensors return the distance between the obstacle and robot, which is denoted by d_{lu} and d_{ru}. d_{lu} is the distance from obstacle and left ultrasonic sensor. Similarly, d_{ru} is the distance from obstacle and right ultrasonic sensor. d_{safe} is the minimum distance at which robot can start to maneuver to avoid obstacle. $d_{safe}$is called the flank safety distance. This value is calculated experimentally based on the range of the ultrasonic sensors and width of the robot.

To determine the angle from the distances given by ultrasonic and infrared sensors, the following cases must be considered.

4.3.1 : d_{lu} >$d_{safe}$and $d_{ru} > d_{safe}$

This situation occurs when there is no obstacle in front of the robot and the robot can continue to move straight, the orientation angle does not change. So the angle at time *t*, is given by,

$$\Theta_{oa}(t) = \Theta_{oa}(t-1)$$

4.3.2 : $d_{lu} < d_{safe}$ and $d_{ru} > d_{safe}$

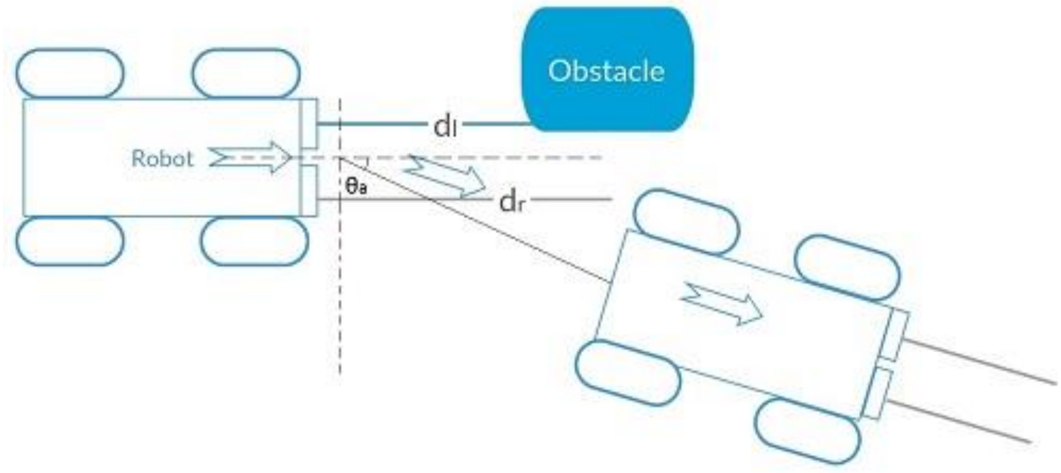


Figure 13 : Right turn after detecting an obstacle on left side

.

The situation is shown in Figure 12. In order to avoid the obstacle, the robot has to turn right. The ideal angle for robot to avoid the obstacle is calculated using distance from left ultrasonic sensor. The angle Θ_{oa} (t) is inversely proportional to distance d_{lu} i.e. smaller the distance higher the angle and vice versa.

$$\Theta_{oa}(t) = \quad \Theta_{oa}(t-1) - \left(\frac{\pi}{2}\right)\left(\frac{k_{\Theta u}}{d_{lu}}\right)$$

where $k_{\Theta u}$ represents the collision angle constant for the ultrasonic sensor, which is calculated based on d_{safe} and d_{lu} .

4.3.3 : $d_{lu} > d_{safe}$ and $d_{ru} < d_{safe}$

When the bot has to turn left in order to avoid obstacle, the ideal angle is calculated using the distance from the right ultrasonic sensor and the angle is calculated similarly,

$$\Theta_{oa}(t) = \Theta_{oa}(t-1) + \left(\frac{\pi}{2}\right)\left(\frac{k_{\Theta u}}{d_{ru}}\right)$$

4.3.4 : $d_{lu} < d_{safe}$ and $d_{ru} < d_{safe}$

When there is a nearby obstacle, this case comes into picture. Here, we use the output from infrared sensors i.e., distance between the obstacle and robot to determine the angle.

Similar to ultrasonic sensors, we have cases in which the robot must turn left, right and in cases when the robot is too close to the obstacle, move backwards.

The angle calculation for right turn is given by,

$$\Theta_{oa}(t) = \Theta_{oa}(t-1) - n\left(\frac{\pi}{2}\right)\left(\frac{k_{\Theta i}}{d_{ri}}\right)$$

Similarly, the angle calculation for left turn is given by,

$$\Theta_{oa}(t) = \Theta_{oa}(t-1) + n\left(\frac{\pi}{2}\right)\left(\frac{k_{\Theta i}}{d_{li}}\right)$$

where n is experimentally calculated to increase or decrease the speed appropriately to avoid close obstacles. $k_{\Theta i}$ is the collision angle constant for infrared sensors and d_{li} is the distance between left infrared sensor and obstacle. Similarly $d_{ri}$is the distance between right infrared sensor and obstacle.

Finally, if the robot is too close to obstacle and is impossible for it to maneuver an obstacle smoothly, i.e. if $d_{li} < d_n$, then the robot moves backwards by a certain distance and checks for all the above possibilities and estimates the orientation angle. d_n is the maximum distance before which the robot should start maneuvering to avoid the obstacle. This value is calculated experimentally based on the range of the infrared sensors and width of the robot. The region $(d_{safe} - d_n)$, is the buffer region for smooth maneuvering.

Using the above equations, we estimate the orientation angle based on the data from the obstacle avoidance sensors. We assume that the robot follows this orientation angle to avoid the obstacle during movement. Based on this, the pose of the robot is estimated as follows.

$$dist = \frac{(l + r)}{2}$$

$$x(t) = x(t-1) + dist(\cos(\Theta_{oa}(t)))$$

$$y(t) = y(t-1) + dist(\sin(\Theta_{oa}(t)))$$

Where x(t-1) and y(t-1) represents previous estimated values

CHAPTER 5

Hybrid Approach for pose Estimation

5. Hybrid Approach for pose Estimation

This chapter describes our novel approach to improve the location accuracy by fusing information from the different sensors appropriately. In robotic localization inaccuracies in orientation measure affects the position more when compared to inaccuracies in distance measure. For example, a robot with 8-inch wheel base and slip of *1/2* inch in one of the wheels results in an error of approximately *3.5* degrees in orientation. So it is important to have a highly accurate value of the orientation measure when compared to that of a distance measure. Our method assumes that the distance *(l, r)* measured using the encoders is reliable, since the systematic errors can be easily removed using UMBenchmark [3] and non-systematic errors do not affect the distance estimate much, as explained in the above example. The orientation estimate from encoders however are not reliable since they undergo huge non-systematic errors due to sudden turns and heavy slippage. Slight deviations in distance due to non-systematic errors results in huge deviations in orientations. The gyroscopes are generally accurate in estimating orientation, and is preferred over encoders. However, over longer durations the performance degrades due to the accumulation of drift errors. Our approach also enhances the accuracy of orientation estimation in gyroscope by

- reducing the period of time, the gyroscope is used, so that drift errors are minimized
- improve orientation estimation accuracy by fusing the estimates from gyroscope with those provided by obstacle avoidance system as discussed in chapter 4.3.

To achieve above approach, the trajectory in which the robot travels is analysed. We identify the part of the trajectory that have low non-systematic errors, so that orientation measure from encoders can be used reliably. In the other parts of the trajectory the gyroscope is employed. Since environments contain obstacles, orientation estimations can be obtained from the robot's obstacle avoidance system, and this is fused with the gyroscope data to get a better accuracy of localization. The main steps in our approach are as follows

- Estimation of curvature of the robot's trajectory
- Using the trajectory to determine the appropriate sensor to use
- Enhance orientation estimation accuracy using a complementary filter to fuse orientation from above step and obstacle avoidance system

Figure 13 outlines the flow of the above steps in the form of an architecture diagram.

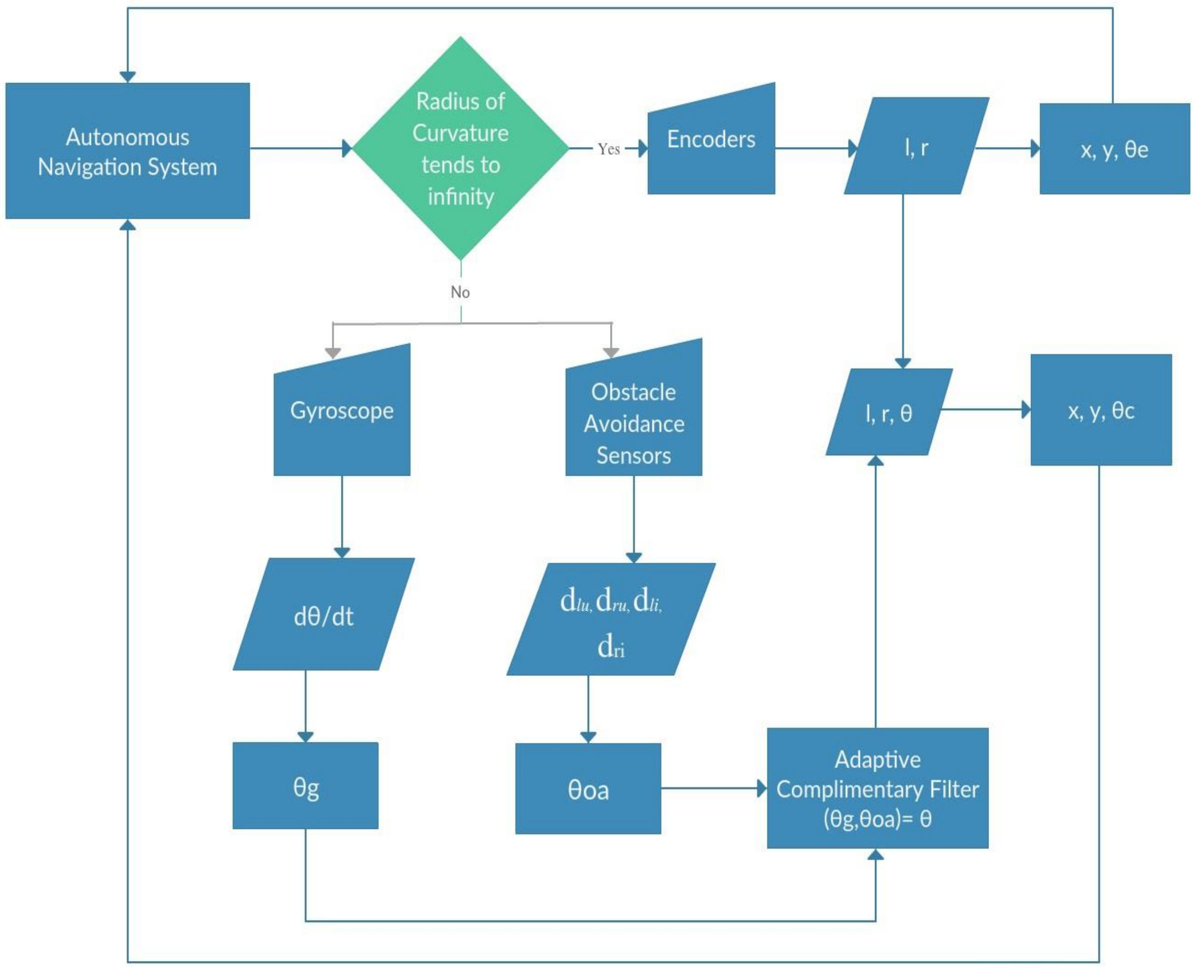

Autonomous Navigation System
Radius of Curvature tends to infinity
Yes
Encoders
l, r
x, y, θe
No
Gyroscope
Obstacle Avoidance Sensors
l, r, θ
x, y, θc
dθ/dt
$d_{lu}, d_{ru}, d_{li}, d_{ri}$
θg
θoa
Adaptive Complimentary Filter (θg,θoa)= θ

Figure 14 : Architecture diagram

5.1. Curvature Estimation

The robot's trajectory is a 2D smooth plane curve. The curvature of the past trajectory of the robot determines the sensor(s) to be used for the orientation estimation at the current point. In order to find the curvature, we find the angle between the slopes of tangents drawn to the previous points P1(x, y) and P2(x, y) on the trajectory. To draw the tangents we find the centre of curvature C (C_x, C_y) (as explained in previous chapter 4.1) of the curve. We then calculate the angle between the slopes of the tangents, which gives the curvature at the point. This method is efficient in curvature estimation since we only store the previous two points of the trajectory and not the complete trajectory to decide on the appropriate sensors.

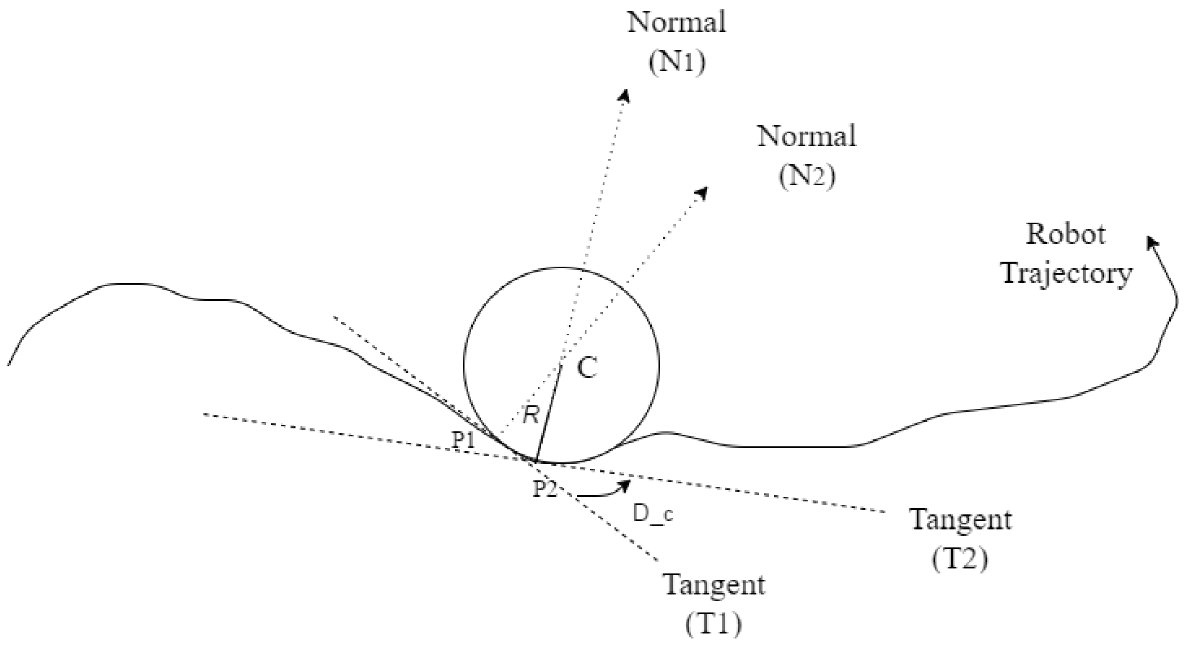
To find the slope $m1$ of the tangent vector T at point P1, we first find the slope *n1* of the normal vector N at P1 passing through centre of curvature C. The slope of *n1* is calculated using,

$$n1 = (C_y - P2(y))/(C_x \ - P1(x))$$

The slopes *m1* of the tangent vector T, which is perpendicular to N is then calculated as the negative of the inverse of *n1; m1 = -1/n1*. Similarly, we calculate *m2*, which is the slope of the tangent vector at point P2.

$$m1 = -1/n1$$

Once these two slopes are calculated the degree of curvature D_C can be determined as follows



$$D_C = \tan^{-1}\frac{m1 - m2}{1 + m1 * m2}$$

Using this D_C value we calculate the orientation angle as explained next.

5.2. Gyroscope Controller

The key to determining an accurate orientation angle is to calculate the precise start and stop threshold values. These threshold values are calculated experimentally, which depend on the width, speed and turning radius of the robot. The start threshold value µ1 is calculated by computing the average of degrees of curvature when the robot begins to make a turn. In scenarios where the robot completes a maneuver by avoiding an obstacle and starts to follow a linear trajectory, the encoders estimate cannot be instantaneously considered for computation. This is due to the fact that there occurs abrupt difference in the left and right encoder ticks immediately after a sharp turn and hence the average of degree of curvatures of multiple linear paths is computed and used as the stop threshold value µ2.

When the D_C keeps increasing and goes over start threshold value µ1, we initiate the gyroscope and use it for orientation measure and allow it to summate as the robot traverses the curved path. The use of gyroscope is ceased when the degree of curvature becomes closer to the stop threshold value µ2. By doing this we ensure that the gyroscope is not made to run continuously for a long time, thereby ensuring that the drift errors from the gyroscopes are not accumulated. When D_C keeps decreasing and becomes equal to or less than µ2, the encoders take over the orientation estimation of the robot. Using the encoders when the D_C is less than or equal to the µ2, reduces the effect of the non-systematic errors resulting in an accurate position calculation.

In order to achieve the above the following conditions have to be met,

- µ2 < < µ1, ensuring the left and right ticks are equal in straight paths.
- µ1 is given higher precedence than µ2, ensuring gyroscope is started as soon as the robot begins to make a turn.

In order to further increase the orientation estimate obtained from gyroscope, it is fused with the angle obtained from the obstacle avoidance sensors explained below.

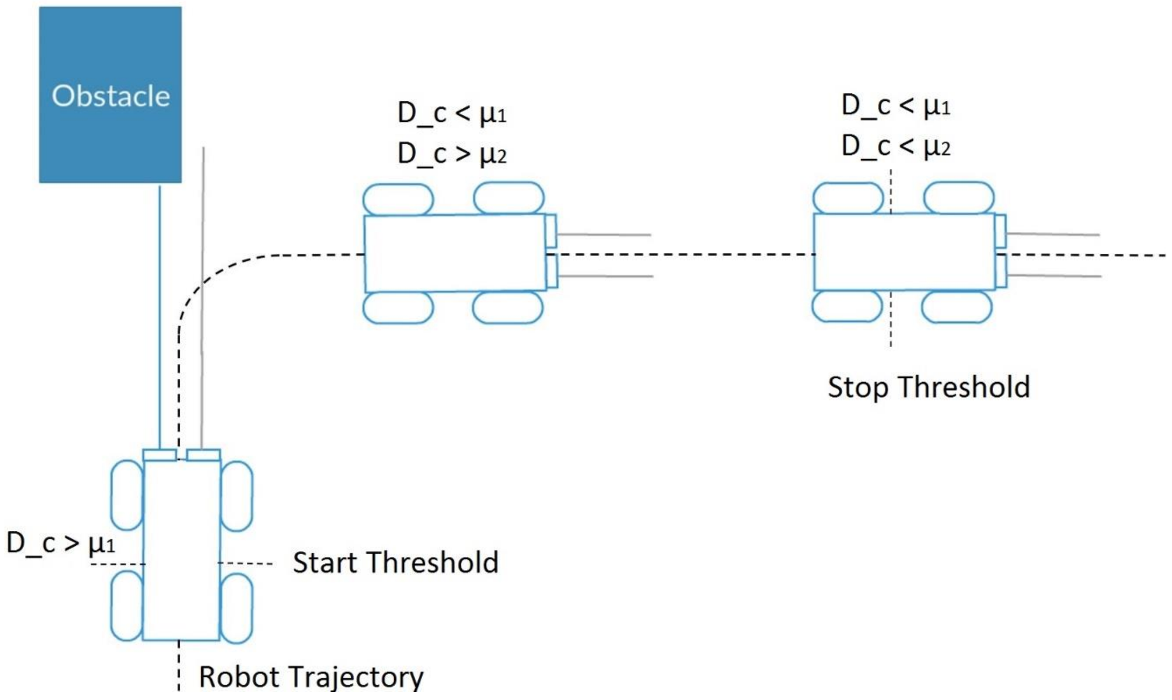


Figure 15 : Gyroscope controller and threshold conditions

5.3. Gyro-Obstacle Fusion

To make the orientation estimate even more accurate, we get an angle from the obstacle avoidance sensors as explained in chapter 4.3. This angle is obtained only when the autonomous robot encounters an obstacle. This angle is then used along with the gyroscope as a parameter to the complementary filter to obtain an accurate orientation. We use a complementary filter since it is very easy and light to implement making it perfect for embedded systems application as in our case. The complementary filter is given by the equation,

$$\Theta = \alpha\Theta_g + (1-\alpha)\Theta_{oa}$$

where Θ_g is the angle obtained from gyroscope, Θ_{oa} is angle obtained from obstacle avoidance sensors and α is the factor which is determined experimentally.

Therefore, we get an accurate orientation estimate from encoders, gyroscope and obstacle avoidance system. This orientation is then combined with the distance measure *(l, r)* from encoders to get the pose estimate and are given by equations,

$$dist = \frac{(l+r)}{2}$$

$$x(t) = x(t-1) + dist(\cos(\Theta\ ))$$

$$y(t) = y(t-1) + dist(\sin(\Theta\ ))$$

On fusing the data by considering the advantages of individual sensors, the position is obtained.

CHAPTER 6

EXPERIMENTAL RESULTS

6. Experimental results

The previous section discussed our approach for improving the accuracy of localization by combining data from different sensors used in our robot. In this section we analyse the performance of this approach. First we discuss the arena and experimental setup followed by the metrics used for analysing our algorithm. We also discuss the methodology for obtaining the ground truth.

6.1. Environment setup

The experiments are carried out in an arena of dimensions 1710 * 480 cm using a four wheeled autonomous robot as shown in figure 1. The arena is then transformed into a grid of squares, each of area 900 cm. Obstacles are placed in different parts of the arena. Different trajectories are considered for evaluating our algorithm. In order to generate the ground truth, the grid is used to identify the points the robot passed through. These points are plotted on a raster map. The following algorithms are evaluated

- Localization using encoders as described in Section 3.1
- Localization using gyroscope as described in Section 3.2
- Localization using Curvature based Decision Approach without integrating obstacle avoidance information (CDA-nO) as described in Section 4.1 and 4.2
- Localization using Curvature based Decision Approach that uses all three sensors (CDA-O).

6.2. Metrics of Evaluation:

The most important indicator in evaluating the performance of a localization algorithm is accuracy. Accuracy refers to how close the estimated trajectory as obtained from the localization method is to the ground truth. In order to determine the closeness, mean and standard deviation of the Euclidean distance between the ground truth and estimated point is calculated. The mean Euclidean distance is the sum of Euclidean distance at each corresponding point divided by the total number of such corresponding points which is given by N.

$$\mu = \frac{\sum_{i=0}^{N}(\sqrt{(x_{gt}-x_{est})^2 + (y_{gt}-y_{est})^2}}{N}$$

(x_{gt}, y_{gt}) refers to the ground truth and (x_{est}, y_{est}) is the estimated position from the localization method used. While the mean provides the error estimates, the standard deviation gives an idea of how consistently the algorithm performs.

Table 1. Average Euclidean distance error(µ)

Methods implemented	Trajectory 1 (Square) (cm)	Trajectory 2 (Eight) (cm)	Trajectory 3 (Random) (cm)
Encoders	121.96	89.27	333.65
Gyroscope	40.50	44.22	75.33
CDA_nO	28.64	39.50	65.38
CDA_O	20.25	34.68	20.49

6.3. Performance analysis:

We evaluate the above mentioned approaches over the arena specified. The evaluation is done by making the robot navigate along different trajectories in the arena. We have considered three trajectories and their corresponding ground truth is plotted and used for comparison. Table 1 and 2 shows the mean and standard deviation of the Euclidean distance error in the corresponding trajectories and methods used for comparison respectively.

6.3.1. Case1: Square trajectory

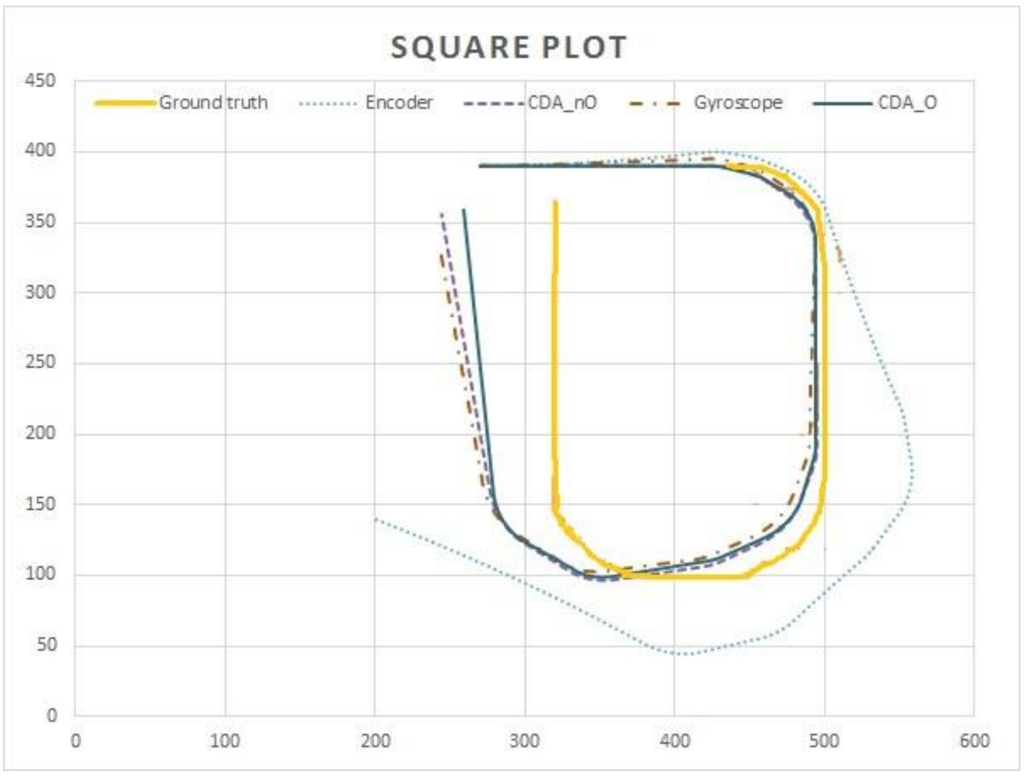


Figure 16 : Square trajectory

The first case is when the robot traversed along the periphery of the arena resulting in a square trajectory. Figure 15 shows the plot of the estimated trajectories for different techniques along with the ground truth trajectory. From the figure and the table, we can see that the average error of encoder based localization is comparatively high. This is because, in the trajectory, the robot has taken sharp turns, which leads to slips and it affects the performance of encoder based method. The performance of gyroscope based localization is better than encoder because its orientation estimates are more accurate during turns, and that dictates the accuracy of the trajectory. The curvature based approach improves the accuracy of the localization over the gyroscope on average. However, using the orientation estimates from the obstacle avoidance system further improves the accuracy, which leads us to the conclusion that orientation estimates during obstacle avoidance has high accuracies and can be reliably used.

6.3.2. Case 2: Eight shaped Trajectory

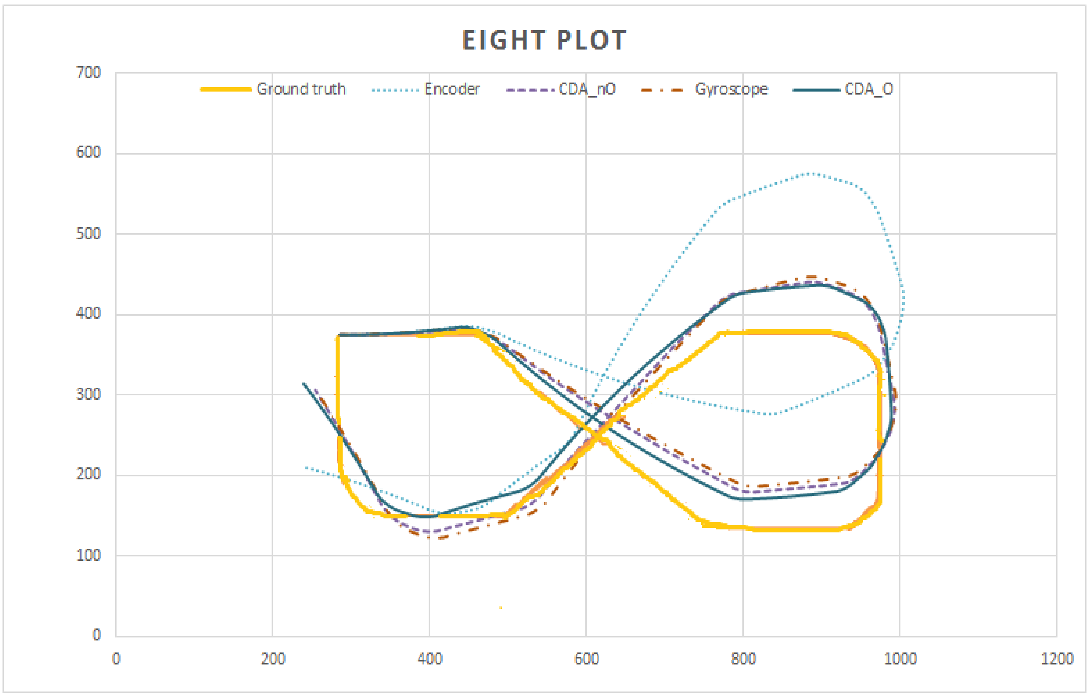


Figure 17 : . Eight shaped trajectory

Figure 17 shows the output of the different localization methods when the robot is allowed to navigate in an eight shaped trajectory. Here we can notice that since there are not many sharp turns, the encoder performance improves. Here too our curvature based approaches (both with and without input from obstacle avoidance sensors) perform the best. While the gyroscope plays a major role in the improvement of accuracy, the nature of the trajectory limits the curvature based decision approach. However, we can see that employing the input from obstacle avoidance improves the accuracy to a certain extent.

6.3.3. Case 3: Random trajectory

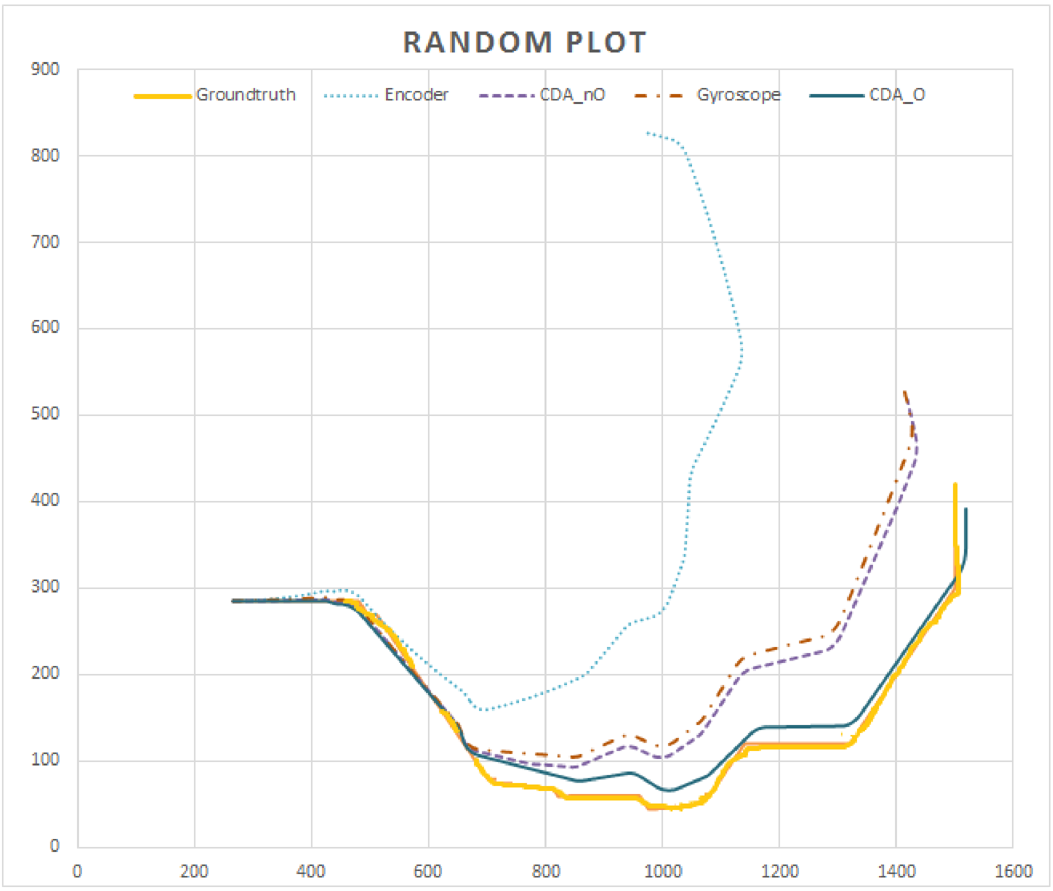


Figure 18 : Random trajectory

To obtain a better idea of how well our approach for localization works, we allowed the robot to take a random path filled with obstacles, and this path was much longer than the previous ones. When the path is longer the drift errors in the gyroscope play a role. Figure 17 shows the plots of the estimated trajectories for this path. We can see that here that our curvature based approach that utilizes the obstacle avoidance system performs exceedingly well in comparison to all other strategies. This is because the path has lot of obstacles and that is well exploited here. The gyroscope performs poorest here, since it is more prone to drift errors here. Therefore, using the curvature based approach helps reduce the drift error, and improve accuracy to a small extent.

CHAPTER 7

CONCLUSION

7. Conclusion

Overall we can see that our approach performs consistently well in all scenarios and gives an accuracy of around 20 to 30 cm. We also observe that considering the curvature and fusing the estimate of the orientation from the obstacle avoidance systems with gyroscopes significantly improves accuracy. The trajectories generated by our approach are very close to the trajectory both in terms of distance and the overall pattern, as can be seen in the plots. The standard deviation values, especially for the rectangular and random plots show that our approach not only provides good accuracies, but does it consistently along the path. Additionally, the complexity of this approach is low and hence is easily implementable in any autonomous robot.

In this project, we have described out approach for improving the localization accuracy of indoor mobile robots, which use inertial sensors for localization. The curvature of the robot's trajectory is analysed to determine when the gyroscope and encoder data are used. In addition to improve the accuracy, orientation estimates from obstacle avoidance systems are fused with the gyroscope's orientation estimates. Our experiments have shown that by using only gyroscopes as and when needed, the drift errors are reduced considerably. By employing the additional data from obstacle avoidance sensors, we see that the accuracy is improved multifold. It is seen that since environments usually have obstacles, and robots have to navigate around them, using that as a parameter is an effective approach for localization. Our approach provides high accuracies at a low overhead, and this approach is computationally efficient compared to many probabilistic approaches. However, when the turns are very sharp, and acute, our approach suffers a bit. Experiments are ongoing to improve the accuracy of complex trajectories that include acute angles.

References:

1. (2015), bstem developer kit integrated robotics platform: http://www.braincorporation.com.
2. Ahn, H.-S. and Yu, W. (2007). ”Indoor mobile robot and pedestrian localization techniques”. In *Control, Automation and Systems, 2007. ICCAS’07. International Conference on*, pages 2350–2354. IEEE.
3. Borenstein, J. and Feng, L. (1994). Umbmark: “a method for measuring, comparing, and correcting dead-reckoning errors in mobile robots”. WP6, March 27, 1995. The University of Michigan.
4. Choi, B.-S., Lee, J.-W., Lee, J.-J., and Park, K.-T. (2011). “A hierarchical algorithm for indoor mobile robot localization using rfid sensor fusion”. *Industrial Electronics, IEEE Transactions on*, 58(6):2226–2235.
5. DeSouza, G. N. and Kak, A. C. (2002). “Vision for mobile robot navigation: A survey. *Pattern Analysis and Machine Intelligence”, IEEE Transactions on,* 24(2):237–267.
6. Easton, A. and Cameron, S. (2006). “A gaussian error model for triangulation-based pose estimation using noisy landmarks”. In *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*, pages 1–6. IEEE.
7. Arduino uno,https://www.arduino.cc/en/main/arduinoBoardUno
8. Appearance-based navigation, localization, mapping, and map merging for heterogeneous teams of robots.”, UC Merced Electronic Theses and Dissertations ,2013
9. Granados-Cruz, M., Pomarico-Franquiz, J., Shmaliy, Y. S., and Morales-Mendoza, L. J. (2014).” Triangulationbased indoor robot localization using extended kalman filtering”. In *Electrical Engineering, Computing Science and Automatic Control (CCE), 2014 11th International Conference on*, pages 1–5. IEEE.
10. Guran, M., Fico, T., Chovancova, A., Duchon, F., Hubinsky, P., and Dubravsky, J. (2014). Localization of irobot create using inertial measuring unit. In *Robotics in Alpe-Adria-Danube Region (RAAD), 2014 23rd International Conference on*, pages 1–7. IEEE.

11. Ibrahim Zunaidi, Norihiko Kato, Y. N. and Matsui, H. (2006). Positioning system for 4wheel mobile robot: Encoder, gyro and accelerometer data fusion with error model method. In *CMU Journal*, volume 5.
12. Jensfelt, P. (2001). “*Approaches to mobile robot localization in indoor environments”*. Citeseer. Master’s thesis, School of Informatics, University of Edinburgh
13. Kam, M., Zhu, X., and Kalata, P. (1997). “Sensor fusion for mobile robot navigation. *Proceedings of the IEEE,* 85(1):108–119.
14. Kelley, K. J. (2015). “Wi-fi location determination for semantic locations. *The Hilltop Review”*, 7(1):9.
15. Krishnan, P., Krishnakumar, S., Seshadri, R., and Balasubramanian, V. (2014).” A robust environment adaptive fingerprint based indoor localization system. In Proceedings of the 13th International Conference on Ad Hoc Networks and Wireless “(ADHOC-NOW-2014), volume 8487, pages 360–373.
16. Li, J. (2012). Characterization of wlan location fingerprinting systems. Master’s thesis, School of Informatics, University of Edinburgh. Liu, H.,
17. Darabi, H., Banerjee, P., and Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080.
18. Surmann, H., Nuchter, A., and Hertzberg, J. (2003). An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45(3):181–198.
19. Widodo Budiharto, D. P. and Jazidie, A. (2011). A robust obstacle avoidance for service robot using Bayesian approach. *International Journal of Advanced Robotic Systems*, 8:52–60.
20. Xiao, Z., Wen, H., Markham, A., and Trigoni, N. (2014). Lightweight map matching for indoor localisation using conditional random fields. In *Information Processing in Sensor Networks, IPSN-14 Proceedings of the 13th International Symposium on*, pages 131–142.
21. Zhang, H., Chen, J. C., and Zhang, K. (2014). Rfid-based localization system for mobile robot with markov chain monte carlo. In *American Society for Engineering Education (ASEE Zone 1), 2014 Zone 1 Conference of the,* pages 1–6. IEEE