2005

Indoor Geo-location And Tracking Of Mobile Autonomous Robot

Mahesh Ramamurthy
University of Central Florida

INDOOR GEO-LOCATION AND TRACKING OF MOBILE AUTONOMOUS ROBOT

by

MAHESH RAMAMURTHY
B.E. University B.D.T. College of Engineering, 2001

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Computer Engineering
in the College of Electrical and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2005

ABSTRACT

The field of robotics has always been one of fascination right from the day of Terminator. Even though we still do not have robots that can actually replicate human action and intelligence, progress is being made in the right direction. Robotic applications range from defense to civilian, in public safety and fire fighting. With the increase in urban-warfare robot tracking inside buildings and in cities form a very important application. The numerous applications range from munitions tracking to replacing soldiers for reconnaissance information. Fire fighters use robots for survey of the affected area.

Tracking robots has been limited to the local area under consideration. Decision making is inhibited due to limited local knowledge and approximations have to be made. An effective decision making would involve tracking the robot in earth co-ordinates such as latitude and longitude. GPS signal provides us sufficient and reliable data for such decision making. The main drawback of using GPS is that it is unavailable indoors and also there is signal attenuation outdoors.

Indoor geolocation forms the basis of tracking robots inside buildings and other places where GPS signals are unavailable. Indoor geolocation has traditionally been the field of wireless networks using techniques such as low frequency RF signals and ultra-wideband antennas. In this thesis we propose a novel method for achieving geolocation and enable tracking.

Geolocation and tracking are achieved by a combination of Gyroscope and encoders together referred to as the Inertial Navigation System (INS). Gyroscopes have been widely used in aerospace applications for stabilizing aircrafts. In our case we use gyroscope as means of

determining the heading of the robot. Further, commands can be sent to the robot when it is off balance or off-track.

Sensors are inherently error prone; hence the process of geolocation is complicated and limited by the imperfect mathematical modeling of input noise. We make use of Kalman Filter for processing erroneous sensor data, as it provides us a robust and stable algorithm. The error characteristics of the sensors are input to the Kalman Filter and filtered data is obtained.

We have performed a large set of experiments, both indoors and outdoors to test the reliability of the system. In outdoors we have used the GPS signal to aid the INS measurements. When indoors we utilize the last known position and extrapolate to obtain the GPS co-ordinates.

Dedicated to my parents

ACKNOWLEDGMENTS

What an experience graduate school has been. It is hard to believe that my graduate study at UCF is drawing to an end. This would be a right opportunity to show my gratitude to a number of people who have made this achievement possible.

I would like to thank my advisor Dr.Guy Schiavone for giving me an opportunity to work with him and also for the help and support he has provided me throughout my thesis work. I would like to thank Dr.Bauer and Dr.Turgut who were on my thesis review committee and provided me with constant encouragement and support and also giving me valuable insights in my work.

I would like to thank all my friends at UCF and IST who have made my stay at UCF so fruitful. They were my bouncing boards for discussing ideas, while also cheering me when the going was tough. Most of all I would like to thank my parents who have made all this possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE: INTRODUCTION

Sensing, perception and effective decision making are the primary tasks involved in moving from one location to another; without any of these components the task of movement would be complicated. For us humans the task of moving around is easy and natural. We do not comprehend the subtasks that are necessary to achieve the end result of crossing a street or navigating a crowded sidewalk. In order to achieve such movements, we combine the knowledge of our immediate environment observed through our eyes, and the sound heard through our ears, and use our reasoning to safely navigate/move without colliding into other's or a stationary object.

It would be a Herculean task for us if we were to walk with our eyes closed; even on a known path. A robot faces such similar problems. If a robot moves in an unknown environment with no sensing powers to know about its surroundings, it is inevitable that it will collide with a stationary or moving obstacle. Furthermore the robot would be lost in the map, as it doesn't know its relative location with respect to initial point. Sensors are mounted on a robot to provide the means of perception and sensing.

In case of a robot, sensing is limited by the capability and the number of sensors. In our case the *Inertial Navigation System, Global Positioning System and an Optical Encoder* serves as the primary navigational sensors. The GPS output is affected by availability of satellite signals. The INS and encoder output are independent of environmental changes. But the navigational performance of an INS degrades with time due to dead reckoning errors. Research has established that the INS errors can be limited by supplementing it with external measurements [1]. *Global Positioning System* has been used in a wide number of applications as

a complementary sensor to the INS. GPS provides external measurements independent of time, weather and location [2]. But GPS measurement is limited by signal blockage and attenuation inside high-rise buildings, mountainous areas and dense forests. Thus combinations of output from GPS-INS fusion provide more accurate and reliable positioning data.

The advantages of GPS/INS integration are more than an improvement of accuracy. For example, the INS solutions can be used to identify and correct GPS carrier phase cycle slips [3]. Improved receiver reacquisition time by using INS to bridge GPS gaps in a tightly coupled GPS/INS Real Time Kinematic (RTK) robust positioning system has been achieved [4]. Significant improvement in anti-jam capability of GPS receiver has been achieved by using a deeply integrated signal processing technique [1]. Finally, positioning availability can be increased greatly.

Mobile Robots

The word robot was introduced by Karel Capek in a play in 1920. A more formal definition is found in the Webster dictionary as "*An automatic device that performs functions ordinarily ascribed to human beings"*. We shall now briefly look into the two of the most important attributes of a robot: Mobility, Autonomy.

Mobility

Mobility can be defined as the degree to which a robot can move freely in its environment. In the early stage of robot research and application, robots were primarily used in assembly lines. They were programmed to repeat the same task faster than a human being. Typically such robots were fixed to the ground with a movable arm, more commonly known as the *robot-manipulators*.

A mobile autonomous robot is one which has complete freedom to move in its environment. The actions are not a set of programmed tasks, but depend on the state of robot and the environment and involve intelligent decision making.

Autonomy

Autonomy of a robot depends on the extent to which a robot depends on prior knowledge. Robot autonomy can be broadly classified in to three classes: Non-Autonomous, Semi-Autonomous, Fully-Autonomous [5].

- *Non-Autonomous robot*s are completely controlled and navigated by humans. They have no intelligence or decision making capabilities. Such robots interpret the commands issued by humans and perform the task.
- *Semi-Autonomous robots* can either navigate by themselves or can be controlled by humans. Krotkov and colleagues [6] do research in how delayed commands from humans can be combined with autonomously made decisions for controlling robots that have to navigate on the Moon and on Mars. In dangerous situations the robot takes full control; in less dangerous situations humans can control the robot. Another form of semi-autonomy is achieved by adjusting the robot environment to avoid the robot from colliding with obstacles or providing input maps to the robot [7].
- *Fully-Autonomous robots* require no human intervention to navigate or perform any specified tasks. Fully autonomous robots are capable of intelligent decision making and navigation. In this case have the problem of navigation-the process of decision making to go from the current position to a destination.

The choice of degree of mobility and autonomy depends on the type of application and the subsequent tasks to be carried out by a robot. In case of industrial robots it is desirable that they have no mobility and autonomy. No artificial intelligence is required as they carry out the same task repeatedly.

In real time applications such as the exploration building applications it is desirable that the robots have freedom of movement and be either semi or fully autonomous. The environment in such cases would be completely unknown and the robot would have to make decisions based on the surrounding conditions.

We will now address the issue of decision making capabilities of a robot. More formally this process refers to the problem of Robot Navigation.

Robot Navigation

Robot Navigation involves the process of going from the current position to a specified destination. To achieve the goal the robot needs to address four points.

1. ***Where am I-Problem Of Localization.*** The robot needs to be aware of its current position in the overall map of the environment.
2. ***Where do I go-Goal Recognition***. The robot needs to find its destination in the global map and move to the position.
3. ***How do I get to the goal-Path Planning***. The robot needs to find an effective and the shortest path to move from its current location to the destination.
4. ***Is the path safe-Obstacle Avoidance***. In this phase the robot needs to evaluate if the chosen path is obstacle free.

Difficulties

The above mentioned four tasks form the basis of an autonomous robot. To achieve these goals involves solving complex subtasks. Some of the foremost difficulties faced in making a mobile robot autonomous are *computational power* for real time data processing, the inherent *uncertainty of sensors* and their susceptibility to external noise and drift, difficulties in *recognizing obstacles* and avoiding them and the difficulty of using the available information in an effective manner. We shall look briefly at each of them.

Computation Power Limitation

The computational power of processing units available is on the rise. But to perform a combination of linear filtering of sensor data, implementing vision algorithms for camera images, building and maintaining a map of the environment in real time necessitates considerable amount of processing power and storage capabilities. An effective alternate would be to build a small to medium sized cluster. But this method incurs higher costs than conventional systems while providing higher computational capabilities.

Uncertainty in sensors

The sensors that are generally deployed on a robot are the *Inertial Navigation System, Global Positioning System, Sonar, Laser, Camera and Optical Encoders*. Each of the sensors is affected by external noise of the environment and mechanical drift. We shall detail the different errors in the sensors more fully in the Chapter 3.

Obstacle Recognition

Robots need to recognize the obstacle to effectively navigate. This is facilitated by sensors such as sonar, laser and the camera. As mentioned before the sensor uncertainty restricts the effective recognition of the obstacles. Furthermore if no input map is provided detecting obstacles and obstacle recognition becomes a more complex task.

Obstacle Avoidance

This is the most important phase in path planning and navigation. A robot needs to effectively avoid all obstacles. This is a tough task when we have dynamic objects in the robot path. The task is comparatively simpler in the case of static obstacles. The problem of path planning for obstacle avoidance is a classical NP-hard problem (Non-Deterministic Polynomial time-hard) [8].

Multi-Sensor Fusion

Multi-Sensor fusion is phase where information from different sensors is fused to get an overall picture about the environment and the robot state. Sensors can be broadly divided into three groups.

1. ***Proximity Sensors***. The SONAR, Laser and Camera come under this category. These sensors aid in detecting obstacles in the proximity of the robot.
2. ***Inertial Sensors***. These sensors aid to detect the state of the robot body, inclination, acceleration. The Inertial Navigation System (INS) including gyroscope, accelerometer, encoders constitute such sensors. The INS is not affected by the change in the environmental conditions.

3. ***Positioning Systems.*** These are positioning systems independent of the mechanical considerations of a robot. The most widely used such system is the Global Positioning System (GPS). They provide the relative co-ordinates of the location of the robot through satellites.

The inertial system and the positioning system are complementary; they provide data relating to the location of the robot. Sonar and laser are complementary that they provide data to determine the obstacles in the robot environment. Thus sensor fusion process is complicated due to the different references used by the sensor data. For such a purpose we use a linear filter to fuse the differing data; *Kalman Filter, Dempster-Shafer and Bayesian Networks* are some of the important ones.

Problem Outline

The focus of this thesis is Indoor Geolocation and Tracking of Mobile Robots. In the area of robotics it is also referred to as Localization. For this purpose we use the Kalman Filter as a linear estimator and data processing is done in real time. From the discussion it can be inferred that the most important problem in geolocation and tracking is the use of sensor data to make the robot autonomous. Current research in the field of robotics particularly focuses on this issue. Some of the main researcher's in the area have proposed new ideas and methods [9] [10] [11] [12].

The problem of geolocation and tracking is complex and is made more difficult due to the existence of uncertainties in sensor readings. Furthermore the available Optimal Estimators are not perfectly suited to the problem at hand. In this thesis we shall explore the application of Kalman Filter as a linear estimator to solve the problem of unreliable sensor data.

In case of geolocation the robot estimates its position continuously and extrapolates the position in to global co-ordinates. This enables tracking the robot in a global map independent of the local environment.

Thesis Outline

In the next chapter we will look at the traditional problem of robot localization and some of the more important algorithms and associated mapping methods. This will provide us with a sound basis for the subsequent discussions of geolocation. In chapter-3 we shall take a look at INS and GPS sensors and the associated errors. In chapter-4 we shall deal with linear estimators to filter the noise in sensor data. We look at Bayesian Networks, Dempster-Shafer theory and Kalman Filter. In chapter-5 we shall provide the detail of the robot architecture that we are using and the method of implementation. Chapter-6 provides the results and conclusions.

CHAPTER TWO: BACKGROUND RESEARCH

Background

Robot localization and making robots truly autonomous has been one of the main research focuses in the field of robotics. The earliest know successful implementation was by Alberto Elfes [7] [13] at the Stanford University in 1988.

Formally localization is defined as: *Given a world map, localization is the issue of the robot finding its relative position in the global map.* Indoor geolocation provides a mechanism for the robot to localize when it is indoors and has no access to GPS signals. Thus the process of the localization can be performed irrespective of whether GPS signals are available or otherwise. Furthermore in the method that we present, localization is not dependent on any preexisting landmarks. We shall now look at some of the previous methods for localization.

Localization methods can be classified into two groups namely: *Landmark Localization* and *Probabilistic Localization.*

We shall first look at landmark based localization, which has been the method of choice for a long period of time. Subsequently we shall look at the more important probabilistic methods that are being widely used.

Landmark Localization

Landmark localization is one of the earliest and most widely used methods for localization. Sutherland and Thompson [14] proposed one of the earliest methods of localization using landmarks. They applied heuristic functions to select a landmark triple, from the set of such triples that is likely to yield a good localization result. Greiner and Isukapalli proposed a learning function to select landmarks that minimize the expected localization error [15]. A

related technique was given by Thrun [16], who trains a neural network to learn landmarks that optimize the localization uncertainty. Other research includes methods by Yuen [17] and Olson [18].

A general method is described below.

- Landmarks are given as input to the robot.
- Compared with the landmark on the world map.
- Possible misidentification of a landmark is one of the main drawbacks.

Landmark Localization using Complex Number Representation

A variant of the landmark based localization is the representation of landmarks in terms of *Complex Numbers* (of the form x+i.j). This method was first used by Betke [19]. Here X_e and Y_e represent the external co-ordinate form and X_r and Y_r represent the robot centered co-ordinate system. The robot is described in terms of X_e and Y_e. ***T*** represents the orientation.

Example:

If F is the visual angle 2 landmarks Z_0 and Z_1 measured at unknown position *P* and if the distance between the two landmarks is known then *P* lies on an arc of the circle spanned by Z_0 and Z_1. We would need three landmarks to pinpoint the co-ordinates of *P*. The example described is an ideal one and the algorithm can be implemented in O (n). Here n is the number of landmarks.

Linear 2D Localization

This is landmark based localization. The method supplies a good estimate of the geometry even without odometry inputs [20] [21]. The method requires atleast seven landmarks

from three vantage points. An initial estimate of the map is provided by using Computer Vision algorithms.

To this end, the bearings are converted to projective coordinates in a virtual 1-D camera, after which a linear 2D method for projective structure from motion (SFM) is used to recover the position of the landmarks and the robot poses. As the method starts from bearing measurements, no calibration is needed and a metric reconstruction is obtained up to a 2D similarity transform.

Probabilistic Methods

A majority of the localization methods now being used are based on concepts of *probability* and the principle of *maximum likelihood estimation*. The method allows any (vision, sonar, laser-range finder) of the sensors to be use for map generation.

The map generated at the current time is compared with a previously generated map to probabilistically maximize the co-relation between the two. The best relative position between the maps is compared using branch-bound techniques.

The major advantages of probabilistic methods are:

1. Can accommodate inaccurate models.
2. Can accommodate imperfect sensors.
3. Robust in real-world applications.

Some of its pitfalls are:

1. Computationally demanding.
2. Inaccurate assumptions.
3. Approximations for reaching solutions.

Considering the above factors, probabilistic methods are one of the important solution types for the SLAM problem. We shall now briefly discuss some of the important probabilistic approaches.

Monte-Carlo Localization

Monte Carlo based localization was one of the earliest localization methods. It is a probabilistic approach based on probability density distribution of the samples. Monte Carlo was explored by Dellaert and Fox [22] [23].

It is a sample based localization method. A set of samples of the robot state is maintained, from which a random sample is drawn and the Probability Density Function is formed. The set of samples denote the probability density function at fixed intervals of time. Here the initial position of the robot is assumed to be known and we recursively calculate the probability $P(X_k | Z_{k)}$. Here Z_k represents the robot state at k given the initial and all the previous state conditions. X represents the 3-D vector of position and orientation of the robot $X=[x, y, T]^T$.

Monte Carlo localization is a 2-phase process.

1. ***Prediction Phase***. It is assumed that the current state X_k is dependent only on the previous state and a known control input U_k.

$$P(X_k | Z^{k-1}) = [P(X_k | U_{k-1}, X_{k-1}) * P(X_{k-1} | Z^{k-1}, X_{k-1}) dX^{k-1}] \quad \text{-----1}$$

2. ***Update Phase***. This step incorporates the information from the sensors to obtain

$$P(X_k | Z^k) = P(Z_k | X^k) * P(X_k | Z^{k-1}) / P(Z_k | Z^{k-1)} \quad \text{------ 2}$$

Here it is assumed that Z^k is independent of the previous state X_k/Z^k-1. In this case the initial position is given as probability distribution function $P(X_0)$.

The main drawback of the Monte Carlo Localization is that samples are taken only at discrete time intervals. There is no continuous localization. Furthermore the method considers only the previous position.

Markov localization

Markov Localization is another probabilistic based localization method. One of the earliest probabilistic Markov methods was proposed by Simmons and Koening [24] in 1995. This method was further explored by Thrun, Burgard and Fox [25] [26]. The key idea of this method is to maintain a probability density function over the entire state space of the robot within the environment. The method corresponds to global localization. The method starts with a uniform distribution over the entire 3-d space. It incorporates sensory data and refines the probability until a uni-modal distribution is obtained. It also incorporates methods for filtering the sensor inputs. For every position represented by [x, y] and the orientation T the algorithm assigns a probability. L_t represents the state at time t.

For any sensory input A_t, at time t we have:

$$P(L_t = 1 | S_t) = \alpha_t * P(S_t | 1) * P(L_t = 1). \quad \text{------3}$$

α_t : Normalizer.

An action A_t is taken based on this probability. Upon executing the action the state of the robot is updated with a new probability given by

$$P(P_{t+1} = 1) = \sigma_1 * P(L_{t+1} = 1 | L_t = (1^{\hat{c}}, A_t)) * P(L_t = 1^{\hat{c}} | S_t). \quad \text{--------4}$$

This is referred to as the action model and it represents the robot position when action $1^{\dot{c}}$, A_t is executed at position $1^{\dot{c}}$. An entire state space is maintained representing all the possible robot positions and they are successively updated with the robot movement.

The main advantage of using such a method is the entire state space model of the robot is maintained and hence continuous localization is possible. But the main drawback is the storage issue and the search process. An entire state space of a robot represents a large number of states and requires a significant amount of memory resources. Furthermore a search in the state space model will have to be optimized to produce real time results.

Frontier Based Localization

Frontiers are regions between open and unexplored space. Open space is defined as the region where there are no obstacles and the robot has not previously visited the area. When a robot moves into an open space it sees new area and this is added to the map, increasing the knowledge about the environment it is working in. A point that a robot can navigate to is an accessible point, moving in a contiguous path. This path consists partially of known territory and leading into the unknown. Evidence grids are used to represent the environment with probabilities assigned to each of the grid. An initial probability of 0.5 is assigned to each of the cells. We can classify each of the cells into 3 categories based on the following rule:

1. Open: Occupancy probability < prior probability.
2. Unknown: Occupancy probability = prior probability.
3. Occupied: occupancy probability > prior probability.

As a final note, most of the algorithms that are applicable for single autonomous robot can be extended for multiple robots working as a team. Such a Markov-localization has been

implemented for robot soccer competition and there are certain extensions of the Monte-Carlo algorithm, which can also be used.

Map Representation

Representation of the environment in map form is one of the important issues to be considered. The map representation should be optimal; the amount of space that is occupied should be kept to a minimum. Such a representation would aid faster search and update algorithm on the map. Various methods have been used to make map-building an optimal process. Some authors have a main map of the environment in a larger storage drive and a copy of a part of it as the active map. The main problem in such a method is that there should be periodic update of the main map to reflect the changes in the environment. Further if there are any sudden changes in the state of the robot and information about the environment is needed, that is not a part of the partial map, the latency involved in obtaining the map reduces effective real time solution. Thus choosing the optimal map representation plays a very important role in real time localization. In the following sections we shall outline a number of methods with their advantages and disadvantages.

Topological Maps

Topological maps represent an important class of map building and representation method. The method was pioneered by Kuipers [27] [28] and others [29] [30] [31] [32].
In this method the robot environment is represented as a *graph* (A set of node points connected by directed/undirected edges). Nodes in such a graph correspond to distinct situations, places and landmarks with unique features or easily recognizable sensor signatures. Often, these places occur at junctions of hallways.

Edges in topological maps may represent:

- Specific paths between places.
- Classes of paths between places, e.g. all different paths that travel down a corridor.
- A behavior or sequence of behaviors that take the robot from one place to another and the initial conditions for these behaviors.

One of the fundamental problems in topological mapping is *closing the loop*- recognizing when the robot has returned to a place it has previously visited. A simple but impractical approach is to *drop a pebble*— upon returning to the pebble, the loop has been closed.

Kuipers takes a different approach: he uses supervised learning to recognize the distinctive places" that are nodes in their topological maps. With sufficiently rich sensing, distinctive places are easily differentiated by recording a unique s*ensing signature* for each place. Without sensing rich enough to recognize distinctive places, the robot must continue to traverse the environment to deduce whether a loop has been closed.

Kuipers rehearsal *procedure* encapsulates the general idea of using the map topology to make this decision.

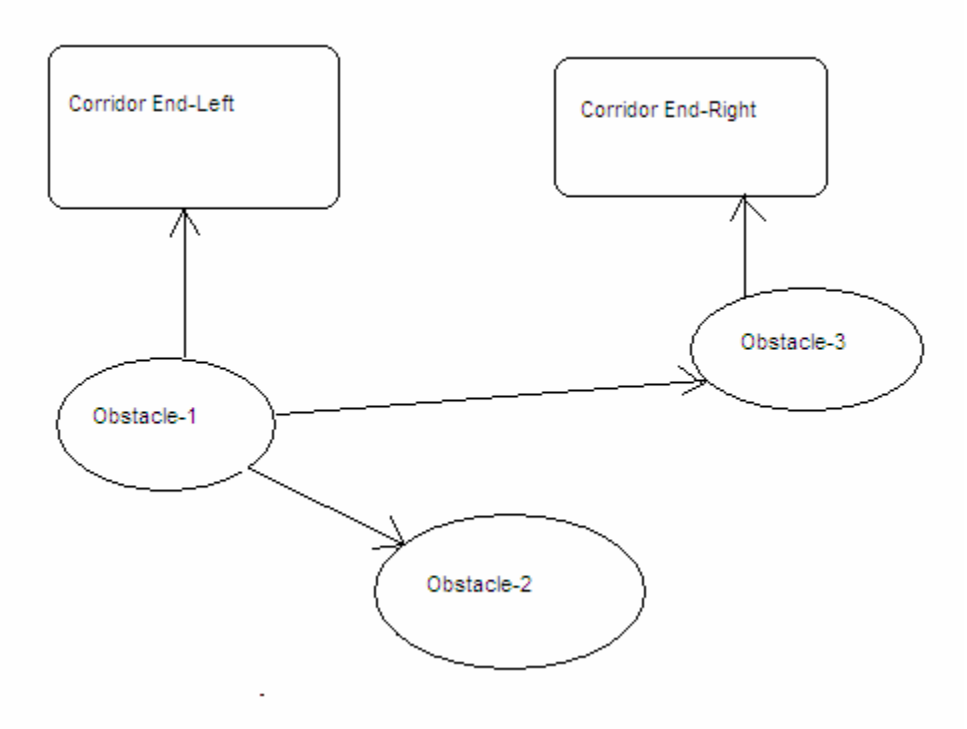

Figure 1: Topological Map

Choset and Nagatani [33], whose topological maps are based upon the Generalized Voronoi Diagram of the environment, describe an approach where structural characteristics of the map (e.g., the degree of vertices and the order of incident edges) are the primary criteria for verification. Tomatis [34] embed this comparison in a POMDP that should show a single peak when the loop has been closed.

A typical topological map consists of nodes representing locations such as T-junctions, corners, dead-ends and closed doors. Each node is connected to its neighbor by an arc representing distance, bearing, and compass information and attributes describing its nature.

Geometric Maps

The geometric representation is a 2-dimensional modeling system using triangulated polygons. The system is hierarchical, allowing the representation of articulated objects. The geometric hierarchy is shown in Figure 1. The objects in the environment and the robot are described in configuration space. Objects (including sub-objects) have transformation matrices associated with them, so they can be translated or rotated by manipulating the matrices. As an object's hierarchy is traversed, the transformation matrices are multiplied so sub-objects are located relative to their parents. This supports articulated objects, in which the sub-objects are able to move independently of one another. Objects also have sets of colors associated with them (so the object can be regarded as being of more than one color), for use in enabling rules for firing.

Typically, we will have two top-level objects: the robot and its environment. Figure 2 [35] is an example of a possible environment. The various objects in the environment are given different colors, to sensitize them for different rules.

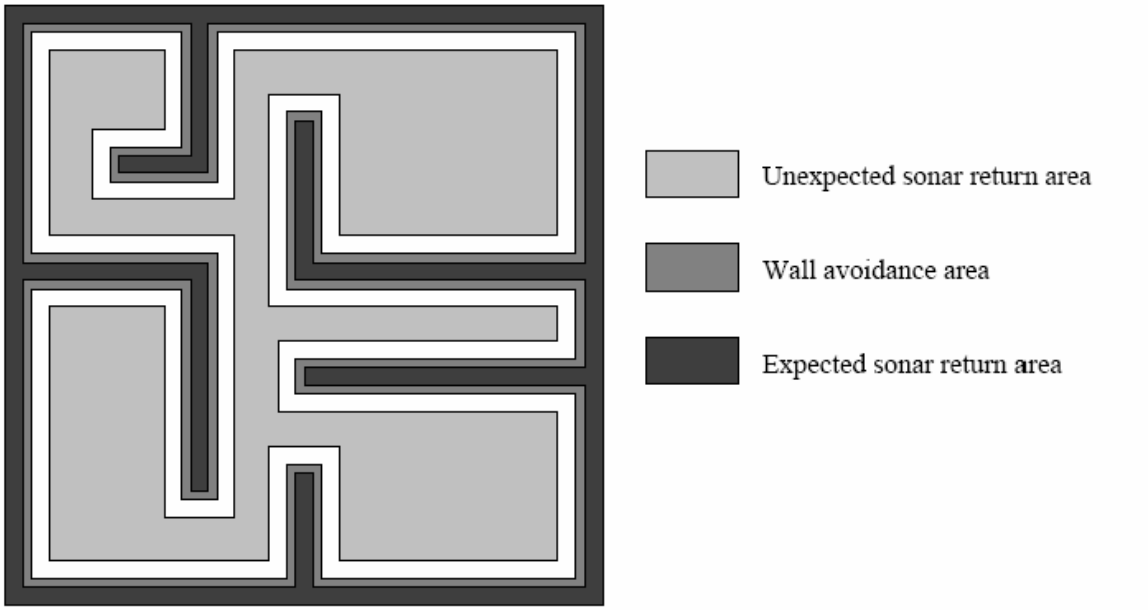

Figure 2: Geometric Map

Voronoi Diagrams

A Voronoi diagram of a set of "sites" (points) is a collection of regions that divide up the plane. Each region corresponds to one of the sites, and all the points in one region are closer to the corresponding site than to any other site. Aurenhammer [36] provides a comprehensive treatment of Voronoi Diagrams.

All of the Voronoi regions are convex polygons. Some of them are infinite; these correspond to the sites on the convex hull. The boundary between two adjacent regions is a line segment, and the line that contains it is the perpendicular bisector of the segment joining the two sites. Usually, Voronoi regions meet three at a time at Voronoi points. If three sites meet at a Voronoi point, the circle through those three sites is centered at that Voronoi point, and there are no other sites in the circle.

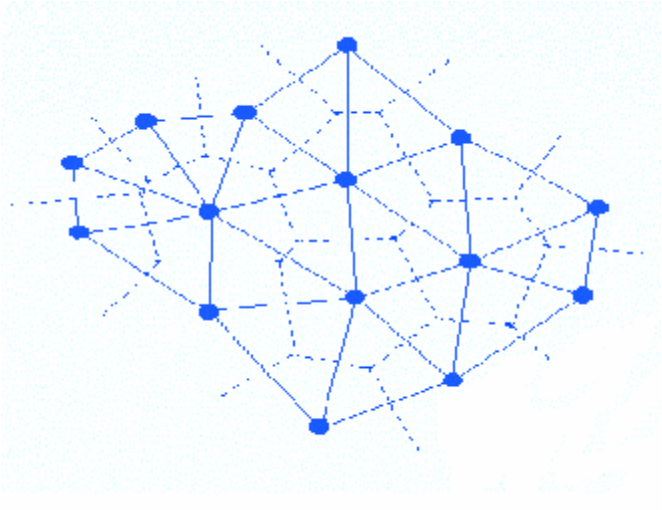

Figure 3: Voronoi Diagram

Choset [37] [38] and others [39] [40] have done considerable research in the use of voronoi diagrams. In graph theoretical operations, it is sometimes easier to use the dual of a graph [41]. The *Delaunay Triangulation* is the dual of a Voronoi diagram. Some of the important properties of a Voronoi diagram are listed below.

- It's dual to the Voronoi diagram, so computing one automatically gives you the other.
- The Empty Circle Property -- If you draw a circle through the vertices of any Delaunay triangle, no other sites will be inside that circle.
- It's a planar graph. By Euler's formula, it has at most 3n-6 edges and at most 2n-5 triangles. This property can be used to reduce many problems with quadratic size (like closest pair) down to linear size in the time it takes to construct the triangulation.
- It contains *fat* triangles, in the sense that the minimum angle of any Delaunay triangle is as large as possible. In fact, if you write down the list of all angles in the Delaunay triangulation, in increasing order, then do the same thing for any other triangulation of the same set of points, the Delaunay list is guaranteed to be lexicographically smaller.

Occupancy Grid Maps.

An Occupancy Grid is a multi-dimensional random field that maintains stochastic estimates of the occupancy state of the cells in a spatial lattice [7]. The method of occupancy grid mapping was pioneered by Moravec and Elfes [7] [42] [43]. To construct a sensor-derived map of the robot's world, the cell state estimates are obtained by interpreting the incoming range readings using probabilistic sensor models. Bayesian estimation procedure allows the incremental updating of the occupancy grid using readings taken from several sensors over multiple view points.

Sensor readings supply uncertainty regions within which an obstacle is expected to be. The grid locations that fall within these regions of uncertainty have their values increased while locations in the sensing path between the robot and the obstacle have their probabilities decreased.

The technique divides the environment into a discrete grid and assigns each grid location a value representing the probabilistic estimate of its state. The state variable associated with each grid is a discrete random variable with states *occupied* (OCC) and *empty*. Initially, all grid values are set to a probability value .5 (i.e., equal probability for occupied and unoccupied) representing unknown state. To update the map using incremental composition of sensor data, the grid cells in the occupancy grid are updated using Bayes Theorem. Given a current estimate of the cell C_i $P[S(C_i) = OCC \mid r_i]$ based on observations $r_i = \{ r_1, r_2, r_3., ..., r_i\}$ and given a new observation r_{i+1}, the improved estimate is given by

$$P[S(C_i) = OCC \mid r_{i+1}] = \frac{P[r_{i+1} \mid S(C_i) = OCC] * P[r_{i+1} S(C_i) = OCC \mid r_i]}{P[r_{i+1} \mid S(C_i)] * P[S(C_i) \mid r_i]} \quad \text{------5}$$

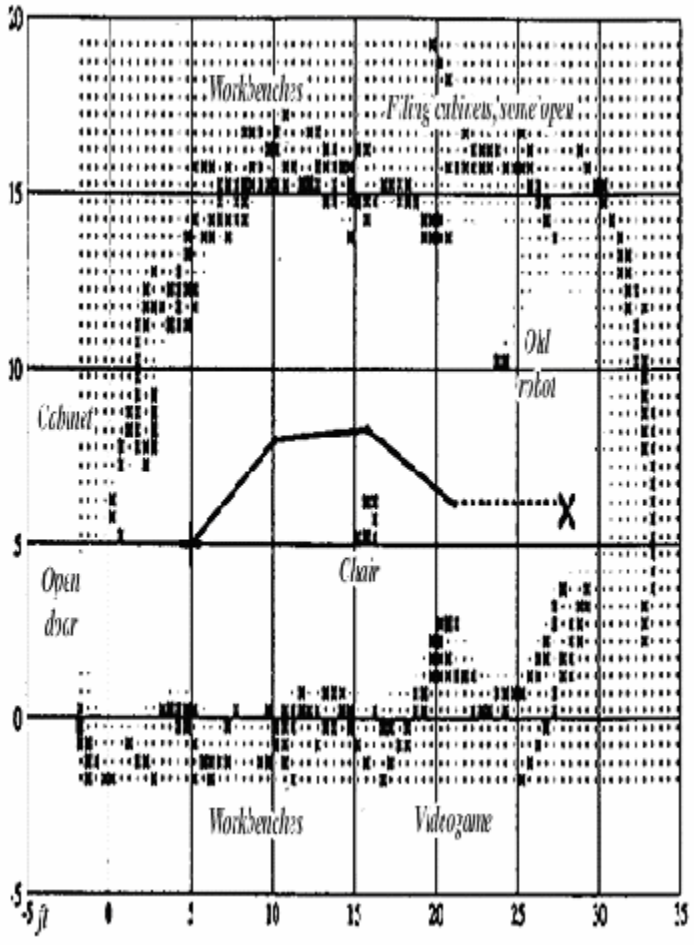

Figure 4: Occupancy Map Generated by SONAR- Adapted from Elfes.

Once the new probability is available representing the *Occupancy Probability*, the cells are update according to relation:

- *Open*: Occupancy Probability < Prior Probability.
- *Unknown*: Occupancy Probability = Prior Probability.
- *Occupied*: Occupancy Probability > Prior Probability.

The occupancy grid representation is the most widely used robot mapping technique due to its simplicity and robustness and also because it is flexible enough to accommodate many kinds of spatial sensors. Furthermore, it also adapts well to dynamic environments.

CHAPTER THREE: SENSORS AND ERROR CHARACTERISTICS

Sensing is the most integral part for the successful operation of a robot. The various sensors on the robot provide it with detailed knowledge of the surrounding environments and also its own state (alignment, inclination, heading). In this chapter we shall look at the sensors that have been deployed on our robot platform and the specific functions that they perform. Further we shall discuss about the various errors and form factors associated with a sensor. This chapter also provides a basis and input for the subsequent two chapters where we take a detailed look at some of the main Linear Estimators for sensor error correction.

The sensors that are typically deployed in a robot can be classified into three groups namely *Proximity Sensors, Inertial Sensors and Positioning Sensors.* We are mainly concerned with the INS and GPS and hence will be discussing them in detail. We shall outline the general characteristics and also the particular sensors that have been used.

Inertial Navigation System

Inertial sensors aid in the detection of robot body state, inclination, acceleration. The most commonly used inertial sensors are accelerometers, magnetometers, encoders, and gyroscope. The recent advancement in MEMS technology has enabled these individual sensors to be bundled in to a single integrated package namely *Inertial Navigation System.* The mounting of the gyroscope is an important aspect for effective data collection. Based on mounting methods three are 3 types of INS: *Space-Stabilized Systems (SSINS), Local-Level systems (LLINS), and Strapdown systems (SINS).*

Space-Stabilized Systems

The space-stabilized system keeps its sensor axes coinciding with an inertial frame. It requires the system to establish its orientation with respect to the inertial frame and to torque the platform back by the amount of rotation it senses, as shown in Figure 2-4.

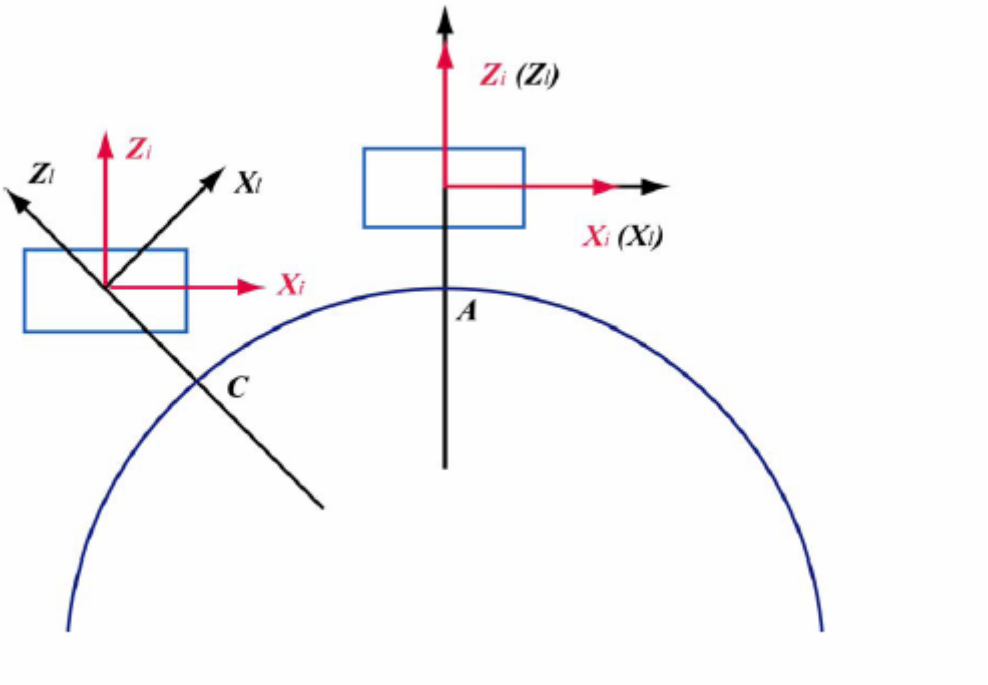

Figure 5: Space Stabilized INS

The rectangular box represents the platform; the frame axis with a subscript *l* refers to the local level frame, and the inertial frame axis is subscripted with an *i*. At the starting point A, it is assumed that the local level frame and the inertial frame coincide. When the SSINS moves from point A to point C, the local level frame rotates an angle relative to inertial space; however, the platform frame tracks the inertial frame so, it still coincides with inertial frame. The integration of the raw data is performed in the inertial frame as well. The result can then be transformed to the local level frame. The main disadvantage of this system is that the gyros and accelerometers are put into a varying gravity field.

Local-Level System

A local-level system aligns its sensor axis with the local level frame. The platform is constantly torqued in order to coincide with the local level frame; the navigation solutions will be obtained in this frame as well. The advantage of a local level system is that no coordinate transformation is needed so the navigation calculation is relatively simple. The problem is that when the system works in the polar region, the control torque becomes very large, so the local level system usually transfers to a wandering mode when it works in high latitude areas. Figure 2-5 shows how the LLINS works.

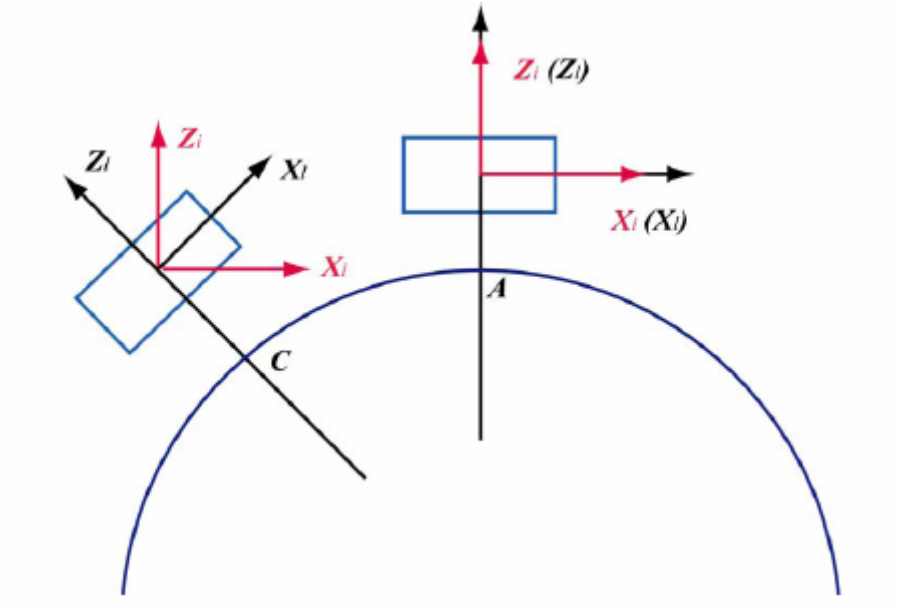

Figure 6: Local Level INS

When the LLINS t moves from point A to point C, the local level frame rotates an angle with respect to the inertial space, and the platform tracks the rotation of the local level frame. The platform frame axis still coincides with the local level frame.

Strapdown Inertial Navigation System

Both above mentioned systems have a mechanical platform, which is torqued to track the navigation frame. Therefore, they are called gimbaled systems. The third system is the strapdown inertial navigation system (SINS). In SINS, the gyroscope triads are directly mounted on the moving vehicle. The sensors measure the rotation rates and the specific forces along the axes of the body frame.
An IMU and the navigation mechanization algorithms form an INS. The IMU is a single unit, which collects angular velocity and linear acceleration data and then sends it to the onboard microprocessor; the signals it outputs describe the vehicle angular rate about each of the sensor axes. Even though the IMU is not located at the vehicle centre of mass, the angular rate measurements are not affected by linear or angular accelerations.

3D-MG Gyroscope

We have used the gyroscope model number 3D-MG developed by Microstrain and have utilized a Strapdown System. It provides us the 3-degrees of orientation and the robot state.

Figure 7: 3D-MG Gyroscope

The gyroscope utilizes:

- 3 accelerometers to measure earth's gravity.
- 3 magnetometers to measure magnetic fields.

- 3 rate gyroscopes to measure the rate of rotation.

The coordinate system of the gyroscope is fixed to the earth with the Z-axis pointing down through the center, X-axis pointing north and Y-axis pointing east as illustrated in the figure.

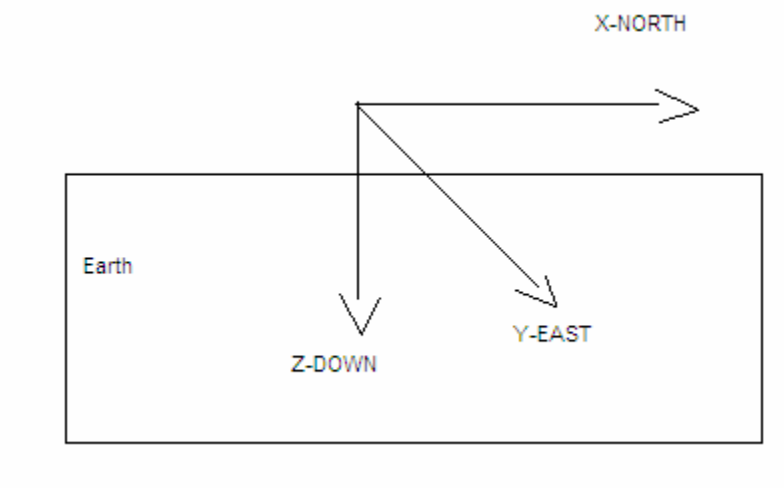

Figure 8: Earth Co-ordinate System

We use the gyroscope to obtain the Roll (-180^{o} to $+180^{o}$), Pitch (-90^{o} to $+90^{o}$) and Yaw (-180^{o} to $+180^{o}$) values of the robot. We are particularly interested in the Yaw (heading). The yaw provides us with the direction of the travel and hence enables us to track the robot effectively and also for the purpose of integrating the position obtained with the data given out by the GPS.

The gyroscope is not affected by the external environmental conditions and hence provides us with a stable measuring platform. The GPS signal is not available indoor, under a canopy of a tree or due to cloud cover. The gyroscope reading is affected by any magnetic field around it. An example would be to wave a screwdriver in front of a gyroscope. The magnetic thus generated would make the gyroscope provide a change in the values even when the body of

the robot is stationary. Later in the chapter we shall look at the possible error calibration of the gyroscope.

Optical Encoder

Encoder forms a part of the Inertial Measurement Unit. We use US Digital made E3 incremental shaft encoder illustrated in Figure-9. An encoder provides us with the distance traveled and the acceleration of the body in motion. The encoder is mounted on either side shaft of the robot. Thus we can measure the individual rates of rotation of each wheel. This is particularly important when the robot is turning left or right as there is motion on one side of the wheels. The encoder delivers a specific number of signals per shaft rotation. The measurement of the cycle duration or counting the pulses per unit time indicates the speed of motion. When the pulses are integrated over time with reference to a known point the distance traveled and the speed of motion is obtained. The angular distance is converted into the linear distance.

Figure 9: Optical Encoder

Global Positioning System

Positioning systems are independent of the mechanical considerations of a robot. The most widely used such system is the Global Positioning System (GPS). GPS is a satellite based navigation maintained by the U.S Department Of Defense. The current constellation consists of 28 geo-stationary satellites with a period of 12 hours. The satellites transmit signals on two frequencies; *L1 at 1575.42 MHz and L2 at 1227.6 MHz.*

These signals are bi-phase modulated by one or two PRN codes; the Coarse/Acquisition, C/A-code, and the Precise, P-code. The L1 carrier is modulated by both the C/A- and the P-codes while the L2 carrier is only modulated by the P-code. The C/A-code is transmitted at 1/10 of the fundamental GPS frequency (10.23 MHz) and is repeated every one millisecond. In contrast, the P-code is transmitted at the fundamental frequency and is only repeated every 267 days. The C/A-code is unrestricted and is used for the Standard Positioning Service (SPS) for commercial use, while the P-code is restricted for use by U.S military only. The GPS is affected by 3 types of error: *Ionospheric error, Troposheric Delay, Satellite clock error, Receiver clock error, and Multi-path and noise errors.*

Ionospheric Error

The ionosphere is a region of the atmosphere extending roughly from 50 to 1,500 km and it is characterized by a significant number of free electrons (with negative charge) and positively charged ions. Free electrons affect the propagation of radio waves, so they are of interest to GPS users. The ionosphere activity is affected by the by the number of sun spots. Figure 2-1 shows the sunspot numbers in the last solar cycle

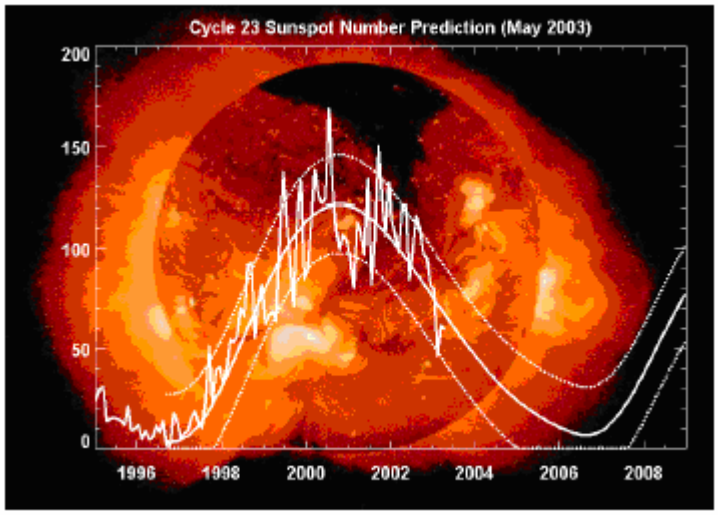

Figure 10: Sunspot and Periodic Behavior (Adapted from NASA).

Tropospheric Delay

The troposphere is the lower part of the Earth's atmosphere where temperature decreases with an increase in altitude. The thickness of the troposphere is not the same everywhere. It extends to a height of less than 9 km over the poles and in excess of 16 km over the equator. The troposphere is electrically neutral and non-dispersive for frequencies as high as about 15 GHz. Within this medium, group and phase velocities of the GPS signal on both the L1 and L2 frequencies are equally reduced. The influences of the troposphere on the GPS measurement can be expressed by wet and dry components. The wet component depends on the distribution of the water vapor in the atmosphere and is harder to model. However, it is responsible for only 10% to 20% of the total troposphere refraction. The dry component has been precisely described by models [44]. The resulting delay is a function of atmospheric temperature, pressure, and moisture content. Without appropriate compensation, tropospheric delay will induce pseudorange and

carrier-phase errors from about 2 meters for a satellite at the zenith to more than 20 meters for a low-elevation satellite [45].

Satellite Clock Error

GPS satellites use two types of atomic clocks: rubidium and caesium. Satellite clock error is referred to as the difference between the satellite clock and the GPS system time (reference clock). This is monitored by the Master Control Station (MCS) and the errors are transmitted as coefficients of a polynomial as a part of the navigation message. Satellite clock errors can be effectively eliminated through DGPS.

Receiver Clock Error

Receiver clock error is the offset between the receiver clock and the GPS system time. The error magnitude is a function of the receiver's internal firmware. It can vary between 200 μs to a few milliseconds. Receiver clock error changes with time due to the clock-drift.

Multipath and Noise Errors

The above discussed errors can be minimized or removed by DGPS corrections; however, multipath and receiver noise cannot be compensated by using DGPS. Multipath is the corruption of the direct GPS signal by one or more signals reflected from the local surroundings. These reflections affect both pseudorange and carrier-based measurements in a GPS receiver. As shown in Figure 2-3, the reflector of electromagnetic signals could be buildings, metal surfaces, water bodies, and the ground.

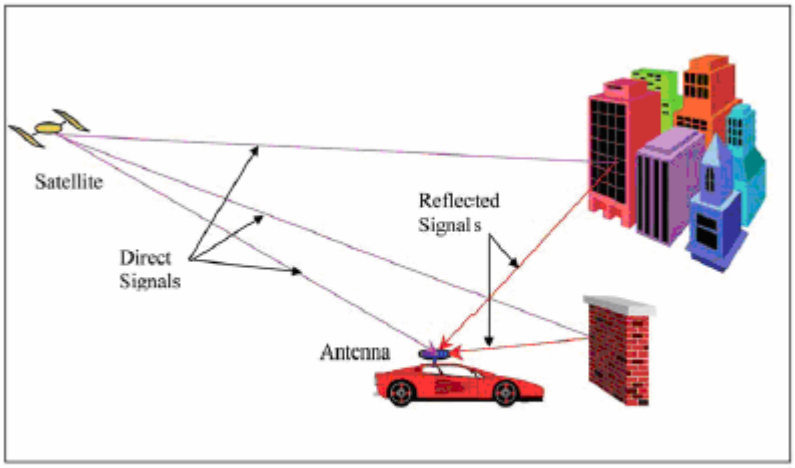

Figure 11: Multipath Errors

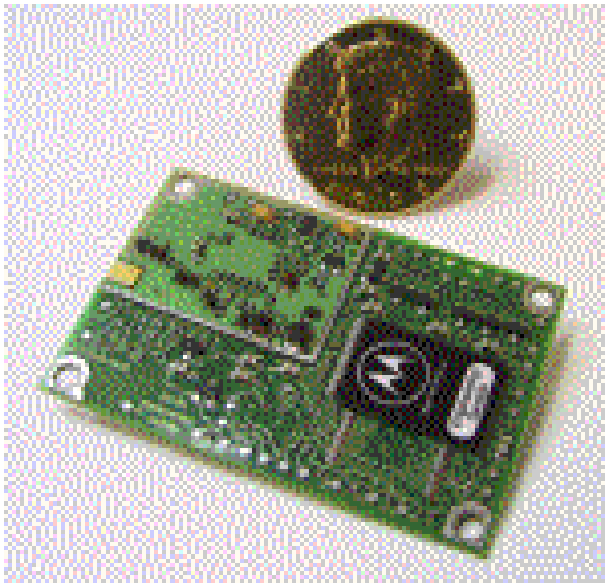

Figure 12: Motorola M12+ GPS Receiver

In our system implementation we use a Motorola 12+ GPS receiver. The receiver is shown in the Figure 11**.** The refresh rate for new data is set at 1second. The data is output in the National Marine Electronics Association (NMEA.org) format.

CHAPTER FOUR: LINEAR ESTIMATORS

The last chapter discussed sensor types and the inherent uncertainties associated with sensors. As mentioned before sensors readings are inherently uncertain due to noise and bias. Hence to implement any localization method it is necessary to understand and implement the error correction in sensors. Hence it is important to choose an optimal filter to obtain the true values.

The filter that are implemented, filter out the noise in the sensor readings. Such filters are referred to as Linear Estimators. In this chapter we look at some of the most important and commonly used linear filters. Presented here is the description and application of ***Bayesian Networks, Kalman Filter, Dempster-Shafer belief theory,*** three most important mathematical model that are used for filtering and sensor data fusion.

Optimal Estimators are used to filter out noise and bias in the sensor readings. Bayes Rule from probability theory forms the basis of a majority of the probabilistic localization methods.

Bayes Rule

If A and B are two independent events, Bayes rule is given by:

$$P(A|B) = P(A)*P(B/A) / P(B) \quad \text{-----6}$$

Example: Let D denote Disease and T denote Test. Let P (Test=true| Disease=true) be 0.95 and P(Test=true| Disease=false) is 0.05. Suppose the disease is rare: P (Disease=true) is 0.01 Then from bayes rule we have

$$P(D|T) = \frac{P(T=true \mid D=true)*P(D=true)}{P(T=true \mid D=true)*P(D=true)+P(T=true \mid D=false)*P(D=false)} \quad \text{-------7}$$

$$P(D|T) = 0.161$$

The linear estimators that we shall discuss are Bayesian Network, Dempster-Shafer theory and Kalman Filter. The first two are based on Bayes rule, while Kalman Filter is a parameterized implementation of the Bayesian network with various extensions. This chapter primarily deals with Bayesian Network and Dempster-Shafer theory, with a brief introduction to Kalman Filter. The next chapter more fully discusses Kalman Filter and its extensions as applied to out problem.

Bayesian Networks

Bayesian network is a graphical model of the possible states of the systems under consideration [46] [47]. In a graph model the nodes represent the random variables and the lack of edges represent conditional independence. A Bayesian Network is depicted as *acyclic directed graph* and has a more complicated notion of independence and takes into account the direction of the edge.

We calculate the probability that an object is in state S after n+ 1 trial using a combination of the values of the probability of the states till time n and further the observed value at time n+1. Thus here we take into account (all) the prior states of the object. The state S is represented as a node in the graph. The probability that we assign to a particular state as an aprior probability does not adhere to the classical rules of probability theory. Based on observed states and the condition at the hand we could assign an apriori probability. This is referred to as the belief. A variable THETA is considered whose values theta corresponds to the probability obtained by the classical method. It is then averaged over the possible values of THETA to make the prediction.

A simple example is illustrated below.

Given a situation where it might rain today, and might rain tomorrow, what is the probability that it will rain on both days? Rain on two consecutive days is not independent events with isolated probabilities. If it rains on one day, it is more likely to rain the next. Solving such a problem involves determining the chances that it will rain today, and then determining the chance that it will rain tomorrow conditional on the probability that it will rain today. These are known as "joint probabilities." Suppose that P (rain today) = 0.20 and P (rain tomorrow given that it rains today) = 0.70. The probability of such joint events is determined by:

$P(E_2, E_1) = P(E_1, E_2) / P(E_1)$ --------8

Determining the joint probability of all combination of events we have.

Table 1: Joint Probability Distribution

Event	Rain Tomorrow	No Rain Tomorrow	Marginal Probability of Rain Today
Rain Today	.14	.06	.20
NO Rain Today	.16	.64	.80
Marginal Probability Of Rain Tomorrow	.3	.7	-

Probabilistic Inference.

In the Bayesian network we need to compute the individual probabilities and probabilities of interest. This is referred to as probabilistic inference. Bayes rule as described above can be used. The formula depends on the number of sensors we are considering and the amount of data that would be used and could be tailored to suit the application requirements.

Inference from Bayesian Network model.

As defined above, a Bayesian network graph has vertices, which are discrete valued variables and are called chance nodes. The nodes encode the states of knowledge about the world and the edges encode the dependencies. Given the sense data we fix probability distributions over the states of the corresponding nodes. Evaluation of the network yields the likely value (posterior probability) that a node can have. For each alternative an expected value is calculated and the best estimate is considered and the particular action is executed [46]. Further the model can be extended for any number of time step history we need to consider. This would amount to adding one node for every time step.

Dempster-Shafer Theory

The Dempster-Shafer theory also known as *theory of belief functions* is a mathematical theory of evidence that was introduced in the late seventies by Glenn Shafer [48]. The DS theory is a generalization of the Bayesian theory of subjective probability. Bayesian theory requires probabilities for each question of interest, but belief functions allow us to base degrees of belief for one question on probabilities for a related question. These degrees of belief may or may not have the mathematical properties of probabilities (sum of individual properties equal to one).

AI researchers started using DS theory due to a characteristic feature. Degrees of belief in Dempster-Shafer suggest that we can combine the rigor of probability theory with the flexibility of rule-based systems. The Dempster-Shafer theory works in two phases: *the idea of obtaining degrees of belief for one question from subjective probabilities for a related question, and Dempster's rule for combining such degrees of belief when they are based on independent items of evidence.*

Example: Consider a seminar where a presentation is followed by a coffee break. The probabilities are as presented below using Dempster's theory.

Belief is distributed across all possible combinations of events: P= {presentation, coffee break, {presentation, coffee break}, δ}.

Universe Of Discourse(P)	Probability Mass Function
P_1: {Presentation}	0.4
P_2: {Coffee Break}	0.3
P_3: {Presentation, Coffee Break}	0.3
P_4: {δ}	0

Table 2: Mass Function for Standing Ratio

Universe Of Discourse(P)	Probability Mass Function
P_1: {Presentation}	0.6
P_2: {Coffee Break}	0.6
P_3: {Presentation, Coffee Break}	0.2
P_4: {δ}	0

Table 3: Mass Function for volume level

Universe Of Discourse(P)	Probability Mass Function
P_1	0.4
P_2	0.3
P_3	0.3
P_4	0

Table 4: Confidence Interval

Dempster's rule begins with the assumption that the questions for which we have probabilities are independent with respect to our subjective probability judgments, but this independence is only a priori; it disappears when conflict is discerned between the different items of evidence.

Implementing the Dempster-Shafer theory in a specific problem generally involves solving two related problems. First, we must sort the uncertainties in the problem into a priori independent items of evidence. Second, we must carry out Dempster's rule computationally. These two problems and their solutions are closely related. Sorting the uncertainties into independent items leads to a structure involving items of evidence that bear on different but related questions, and this structure can be used to make computations feasible.

Kalman Filter

R.Kalman proposed a recursive solution to the discrete linear filtering problem in 1960 [49]. Since that time, Kalman Filter has gained in popularity and importance and has been widely applied in a number of areas ranging from military applications, space programs, aircraft navigation to civilian applications. A Kalman Filter is a parameterized Bayesian estimation and is based on the *minimum mean square error* estimation. Kalman Filter process proceeds in two

stages: *Prediction Phase* and *Update Phase*. We shall delve in to the algorithm without going into the nuances of the derivation of the associated equations. For a detailed study the reader is referred to [50] [51]. The Kalman filter algorithm has been widely used to process the data since it has many advantages over other estimators such as Bayesian Networks, Dempster-Shafer. This is due to the reason that a Kalman Filter is easy to implement and convenient for real time processing. Further, Kalman filtering offers flexibility such that it can be used in either in a real-time or a post-mission environment. It can also accommodate measurement updates from a wide variety of sensors including GPS, INS and encoder.

State Space Model

According to linear system theory, the dynamics of a linear system can be represented by a state space model, where a set of first order differential equations express the deviation from a reference trajectory (Liu, 1994).

$x = F*x + w$ -----9

$z = H*x + v$ -------10

Where,

X: x is an n×1 state vector

F: n × n system dynamic matrix

z: is an m × 1 observation vector

v: m × 1 measurement noise

H: H is an n × m design matrix

m: is the number of measurement

n: Number of the states.

Equation 9 is the dynamic equation and equation 10 is the observation equation. Since the implementation of the estimation process is done on a computer, the discrete form is generally more convenient to use.

Corresponding to equations 11 and 12, the discrete system equations are derived as follows:

$x_{k+1} = \Phi_{k+1,k} * x_k + w_k$ -------11

$z_k = H * x_k + v_k$ ----------12

Where,

k: epoch t_k

Φ: N x n State Transition Matrix

$x_{k:}$ State vector at a discrete epoch k

$z_{k:}$ Observation vector at a discrete epoch k

$w_{k:}$ System noise at epoch k

$v_{k:}$ Observation noise at epoch k.

In a Kalman filter, it is assumed that w_k and v_k have white noise characteristics with the following properties:

$E[w_k = 0$ and $E[v_k] = 0$

$E[w_i, w_j] = Q * \delta_{ij}$, $E[v_i\ v_j] = R * \delta_{ij}$, $E[w_i\ v_i] = 0$

$\delta_g = 1$ when i = j and 0 $i \neq j$, Where E () is the mathematical expectation.

Kalman Filter Algorithm

A Kalman Filter operates in 2-phases. A time update followed by a corresponding measurement update, using the state space approach. The system is described by a set of variable known as the state variables. A state contains all the information about the system at the current point of time. Further this set of variables should be the least amount of data required to represent the past behavior of the system in order to predict its future behavior.

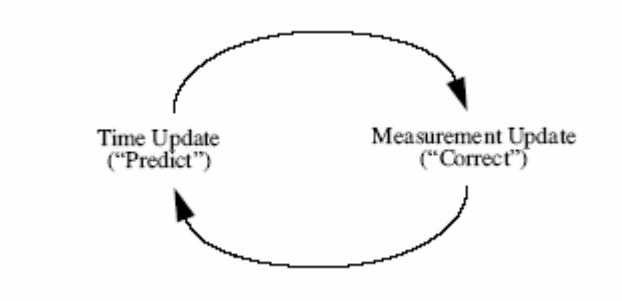

Figure 13 -Kalman Filter Cycle

Prediction Phase

In the time update step the Kalman filter predicts the value of the state variables based on the previously known initial conditions. This step consists of two equations.

$x_k(-) = \varphi_{k,k-1} * x_{k-1}(+)$ ---------13

$P_k(-) = \varphi_{k,k-1} * P_{k-1}(+) * \varphi_{k,k-1}^T + Q_{k-1}$ ----------14

$x_k(-)$ denotes the predicted value of the state vector and $P_k(-)$ is the error covariance estimate. These two vectors serve as the input to the 2nd stage of the Kalman Filter.

Update Phase

In the update phase the linear process is measured for input value. The steps corresponding to this stage are given below.

$K_k = [P_k(-) * H_k^T * H_k * P_k(-) * H_k^T + R_k]^{-1}$ ----------15

$P_k(+) = [I - K_k * H_k] * P_k(-)$ ----------16

$X_k = X_k(-) + K_k[z_k - H * x_k(-)]$. ---------17

The most important task here is to calculate the Kalman Gain K_k. The actual measurement of the process is input as Z_k. This is used in combination with the predicted state vector to obtain the updated state vector X_k. A flowchart of the Kalman Filter operation is illustrated below:

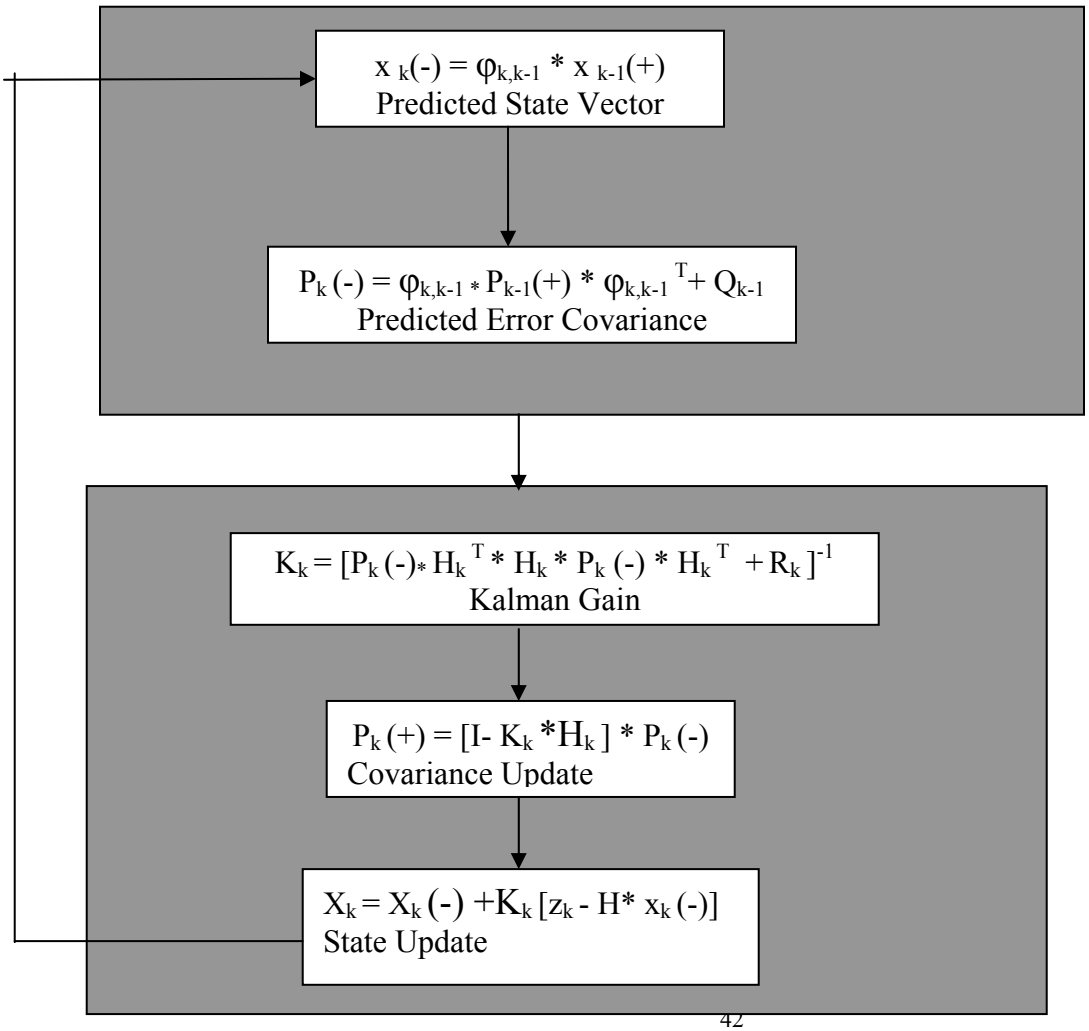

Figure 14: Flowchart of Kalman Filter

Sensor Fusion Architecture

The choice of integration architecture is largely dependent on the performance needed by the filter and also the available inputs. The strategies range from the complex to the simple. The 3 main schemes are: Uncoupled Mode, Loosely-coupled Mode, Tightly-coupled Mode. We shall in turn look at each of them.

Uncoupled Mode

In this method the GPS and INS produce independent navigation solutions with no influence of one on the other. (PV)*GPS* are the GPS-derived position, velocity while (PV)*INS PV A* are INS-derived position, velocity and attitude. (PV)*est PV A* are the estimated positions, velocities and attitudes parameters. The integrated navigation solution is mechanized by an external integration processor which in our case is the Kalman Filter. Figure- illustrates the operation.

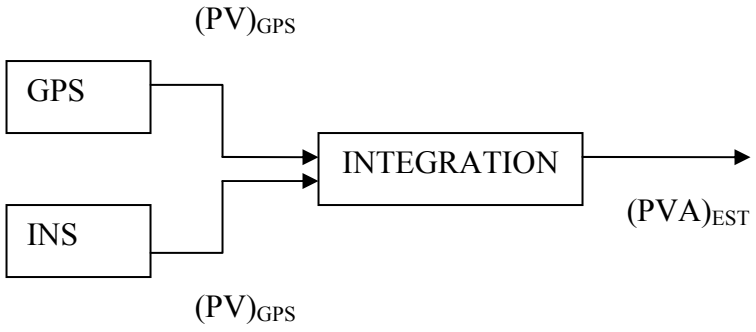

Figure 15: Uncoupled Mode

Loosely Coupled Mode

Inertial navigation systems in principle permit autonomous operation. However, due to their error propagation properties, most applications require high-terminal accuracy, and external aiding is usually utilized to bound the INS errors. Figure 15 shows a loosely coupled integrated configuration with a feedback loop.

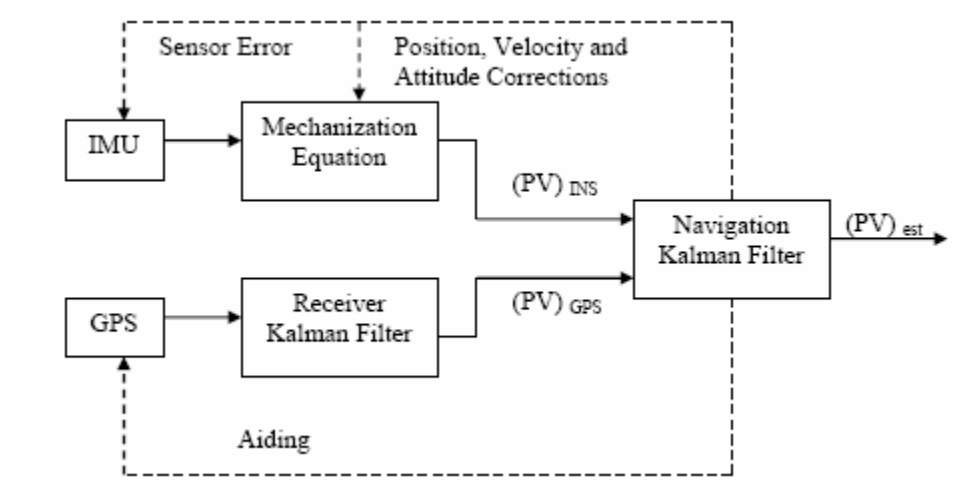

Figure 16: Loosely-coupled Mode

In a loosely coupled system, the GPS receiver has its own Kalman filter to process which are used to calculate positions and velocities. GPS-derived positions and velocities are combined with INS positions and velocities to form the error residuals which are sent to the navigation Kalman filter. This filter corrects the INS in a feedback manner, and the effects of biases and drifts, as well as misalignment errors, will be significantly decreased. All measurements are considered without pre-correction.

Tightly Coupled Mode

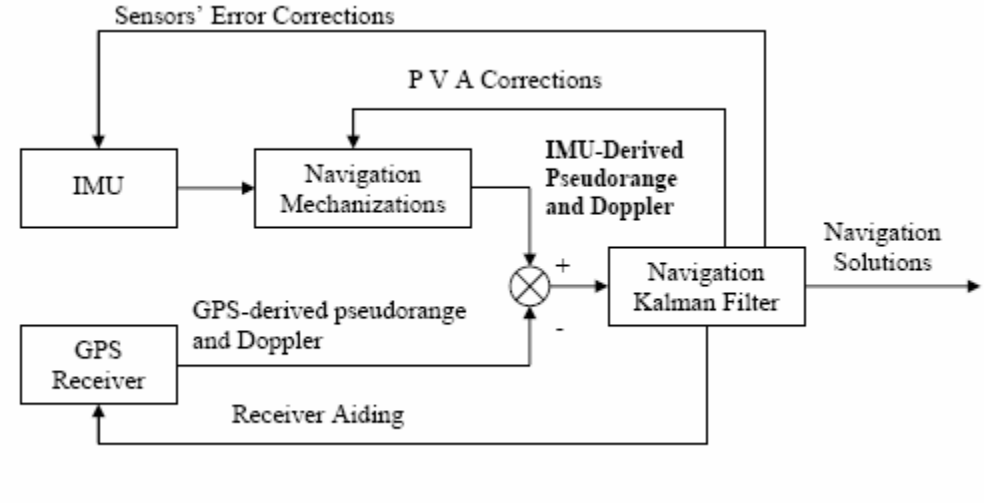

Figure 17:Tightly Coupled Mode

In tightly coupled system architecture, separate Kalman filters for the GPS receiver and the navigation process are combined into a single integrated filter. The operation is as shown in Figure 16. This filter accepts GPS pseudorange and Doppler measurement residuals directly. The filter error states now include the INS error states (position, velocity, attitude, gyro drift, accelerometer bias) as well as new states representing the GPS receiver clock bias and drift. The components of the filter state vector that represent the INS errors are used to calibrate the INS and correct its estimates of position and velocity and the direction cosine matrices (DCM) describing vehicle attitude. The filter estimates of clock bias and drift are used to correct the GPS measurements.

The tightly coupled architecture more effectively utilizes the available measurements and *a priori* information to determine and correct for system errors in a highly integrated fashion. It can thus yield better performance than the loosely coupled system, providing accurate navigation estimates during periods of high vehicle dynamics or jamming.

CHAPTER FIVE: INDOOR GEOLOCATION

In this chapter we shall look into detail about indoor geolocation which forms the crux of the thesis. The problem of Indoor Geolocation has traditionally been the domain of wireless networks and robot applications have been limited to exploring. As mobile robots have found new applications it has become capital to track the robot in the environment. The solution to the problem till now has been the use of Global Positioning System (GPS). GPS signal and data is available round the clock. But, despite the availability of GPS positioning system, indoor locations are still out of reach of the satellites due to the fact that the signals obtained from the satellite cannot penetrate into closed structures. With robots finding widespread use in application ranging from military to commercial, accurate indoor geolocation has gained a lot of importance. Some of the most important of such a system would include commercial, military applications and public safety. In nursing homes and hospitals there is a need to track the elderly and the disabled at all times. Navigation for the blind forms another very important application. Military personnel need the means of tracking soldiers and supplies in Urban Warfare to ensure successful completion of the mission. Fire fighters and policeman use this to effectively navigate and provide efficient public safety. These are just some of the few applications of an indoor geolocation system.

A typical geolocation system is illustrated in Figure 14. The main components of such a system consist of location sensing elements (Gyroscope, Encoder) that measure metrics related to the relative position of the robot with respect to a known reference point. In our case the reference point would be the last known GPS position. The heading angle and the distance traveled provide the relative measurements. These serve as the input to the positioning algorithm that calculates

the relative GPS co-ordinates. The updated position is transmitted to a central server/display unit. The accuracy of such a positioning system is based on the reliability of the data obtained from the location sensors.

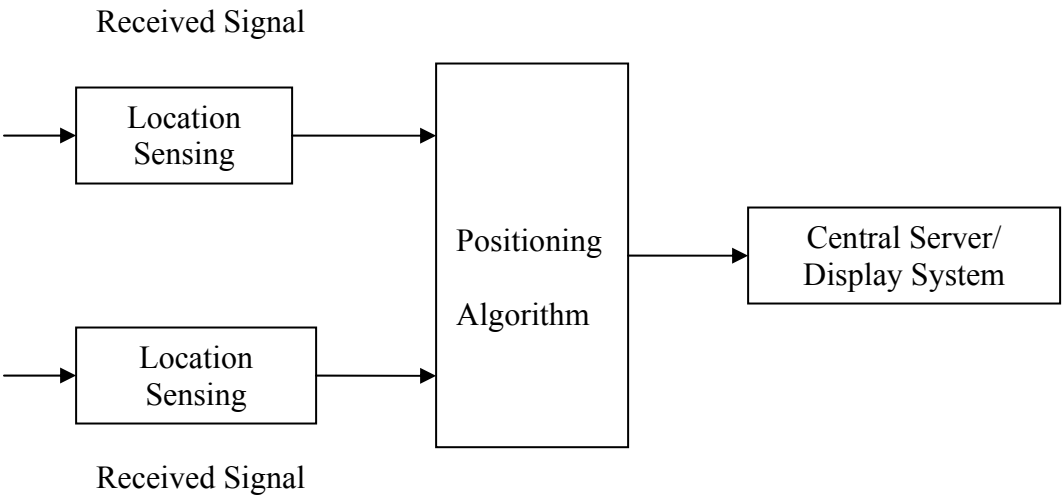

Figure 18: Functional Block Diagram of Indoor Geolocation System

Before we outline our approach we shall look into some of the traditional approaches to geolocation. They are based on the use of *Radio Signal Strength (RSS)* and *Direct Line of Sight (DLOS)*. Typically the indoor location system follows two approaches. In the first, a separate infrastructure is deployed primarily for geolocation purposes. The second approach uses an existing wireless communication network to locate a mobile terminal. Our proposed approach overcomes the problem of an existing network and can be deployed independently of the existing network.

System architecture

In this section we shall present a detailed description of the robot that we use as the platform of geolocation.

Figure 19: Robot platform for indoor geolocation

The tank robot illustrated in the figure is the primary platform that we have developed for indoor geolocation. The platform is a 1:8 scaled version of WW II Panther tank. We have custom built the interior of the tank to suit our purpose and also to provide ample space for hardware deployment. Figure16 shows the interior view of the tank with all the components.

Figure 20: Components inside the Platform

The flowchart illustrates the control flow of our platform. The JStik is the main processing unit and the various sensors are interfaced to it via RS-232 serial communication. We shall now look at the main components of the system.

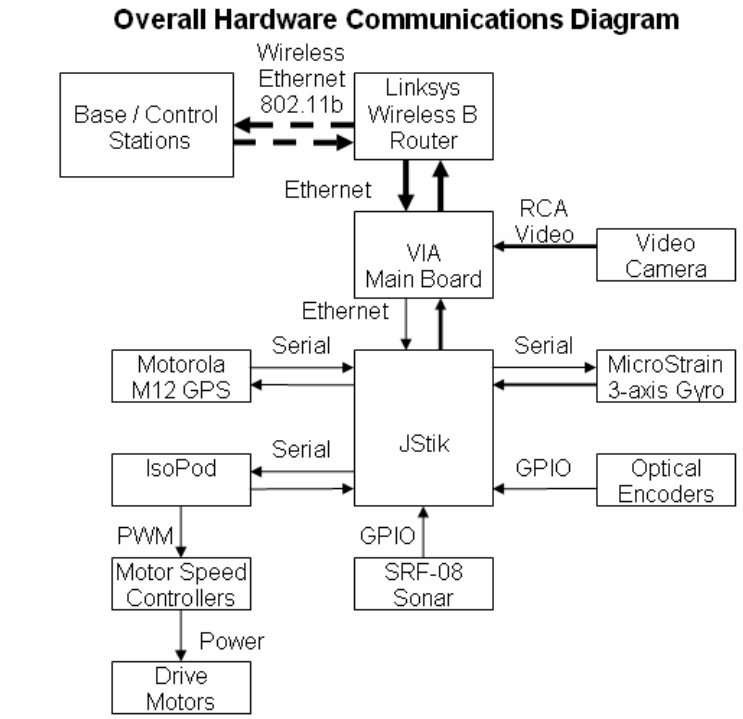

Figure 21: Hardware Communication

JStik

The JStik is a native Java execution based board-level microprocessor developed by Systronix. The JStik has a Simm30 form factor with an on-board Ethernet. The JStik has no firmware or Java Virtual Machine (JVM) or operating system. The java files are converted into assembly and downloaded into the JStik RAM/Flash. This provides us with a high speed execution rate, typically fifteen to 20million instruction per second at 103MHz. The JStik provides multiple RS-232 serial ports for external sensor connection. Currently, the JStik is interfaced to GPS, Gyroscope and the encoder.

VIA EPIA-SP

The VIA is a main board CPU based on x86 includes a embedded VIA Eden processor. The VIA provides high performance at low power consumption and cost. The VIA has a 512MB DDR SDRAM. It provides up to 1GB memory extension.

Isopod

The Isopod is a real-time micro-controller in small size (1.2" X 3.3") with dense features and accessible connections. It has a DSP based core for speeds up to 40 Million Instructions per Second. The board is equipped with 16 General Purpose Input Output (GPIO) pins. They can be used for RS-232 and RS-422 based communication and serial interfacing at various baud rates to other devices. For networking purposes it has a CAN 2.0 A/B serial bus system. The figure illustrates the Isopod.

The most important feature about the Isopod is the Virtual Parallel Machine Architecture (VPMA). The VPMA allows small independent virtual machines to be constructed and seamlessly added to the main system. This allows the various virtual machines to run in parallel. Furthermore, Isopod comes with IsoMax a language designed for programming the Isopod. The IsoMax can be for programming multi-tasking applications easily as opposed to a conventional programming language. In our system, the Isopod is used for controlling the motors on-board the robot. The commands are issued by the user, which are then translated into machine understandable format. Thus the Isopod provides an easy to use method to control the movement of the robot.

Networking architecture

For interprocess communication we follow standard 2-tier Client-Server architecture. A base/control station used to control the robot movement and also to display data to the end user serves as the client. The JStik acts as the server serving data from an array of sensors and running geolocation algorithms. Figure- illustrates the 2-tier architecture.

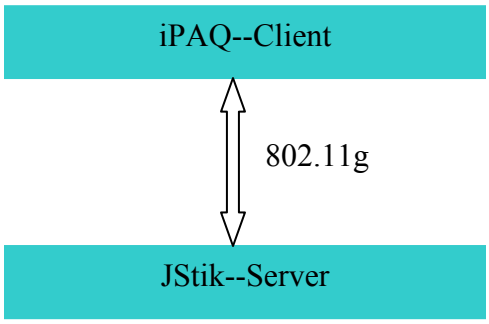

Figure 22: 2-Tier Client-Server Architecture

We have implemented wireless communication using 802.11g standards. User Datagram Protocol (UDP) protocol has been implemented for wireless broadcast communication. This is superior to TCP/IP performance in our case. This is due to the fact that there are multiple clients (iPAQ's) receiving the same data. A point to point TCP communication would involve a lot of overhead (Checksum, ACK/NAK packets) for every transmission between the base station and the robot. Further this process would have to be repeated for each of the communication link.

We use a cluster of HP iPAQ 5550 palm tops as our first level of control stations. The use of iPAQ does provide dynamic mobility of the network and also the capability to add more control stations as necessitated. The combination of the iPAQ and the tank platform forms a mobile ad-hoc network as illustrated in Figure-17.

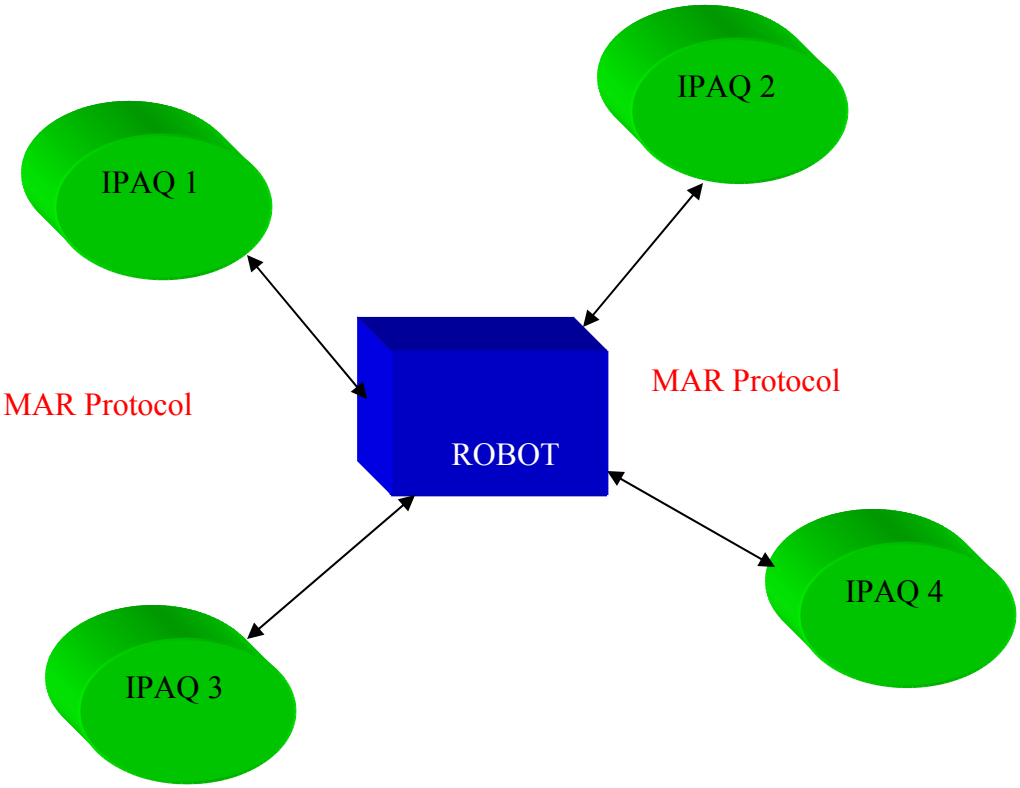

Figure 23: Mobile Ad-hoc Network

Thus we can use the principles and protocols associated with such networks. Communication between the iPAQ and the tank is achieved via on-board wireless router. Assigning static IP address to both the iPAQ and the on-board router, we have achieved a closed network security with no unauthorized user able to hack into the network. Further we use encrypted data communication by utilizing channels 12-14 in the wireless configuration. Figure-18 illustrates the low level communication architecture.

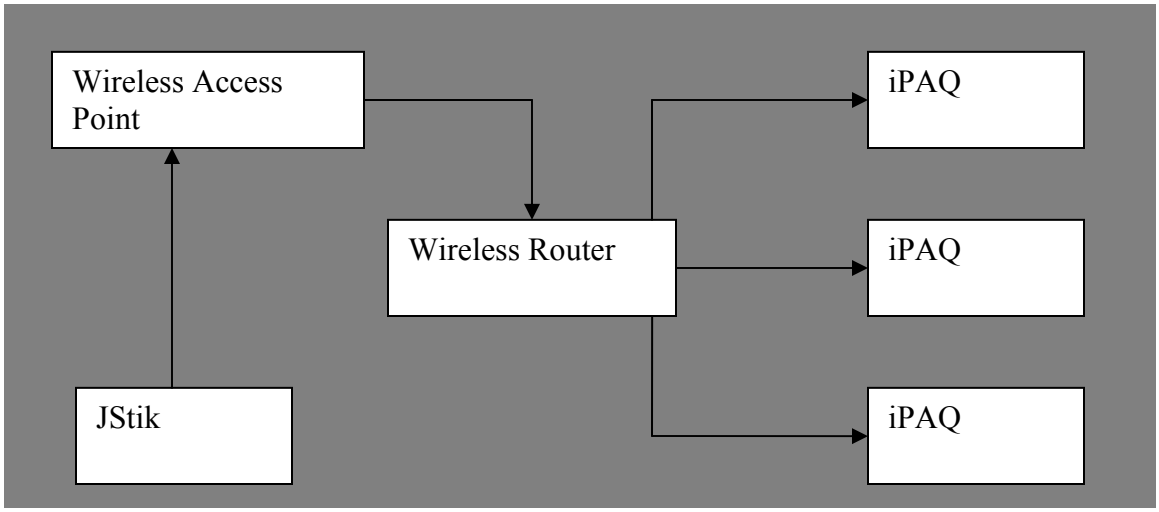

Figure 24: Communication Architecture

We have implemented the MAR wireless ad-hoc protocol developed by Intelligent Automation Inc. MAR provides a stable and secure network protocol for our purpose. MAR treats the iPAQ's and the tank robot as nodes in a wireless network. Routing is based on battery life and shortest path. The next section provides an overview of the workings of the Mobile Agent-based Routing (MAR) protocol.

Mobile Agent-based Routing

The MAR protocol is a ad-hoc network based implementation providing automatic rerouting of connections in a mobile ad-hoc network and also Quality of Service (QoS). Some of the main features of MAR are:

- Ad-hoc routing with Quality of Service (QoS) guarantee
- Hybrid approach
- Priority based link preemption

- Negotiation-based topology exchange for overhead reduction
- Supports reconnection, preventive, opportunistic and priority rerouting
- Automatic multi-channel routes
- Hidden terminal interference considered during routing
- Seamless integration with IP networks (no bridges required

MAR protocol is accompanied with the NetSim, a distributed agent simulation tool. NetSim provides a easy to use visualization of the network nodes and the associated traffic flow. Performance metrics can be easily evaluated using NetSim. Some of the important features of NetSim are:

- Independent OSI layer simulation
- Modeling of node mobility and connectivity
- Multiple levels of fidelity
- Evaluation of multiple routing and transport protocols
- Ability to run domain applications (e.g., data mining) on simulated nodes
- Node logging and connection tracing
- Node implementation portable to Linux

NetSim can be adapted to our use to visualize the various nodes such as the iPAQ and the robot on a world map using the latitude-longitude co-ordinates. This facilitates easy tracking of node movement and changes in the network information flow.

The flowchart below illustrates the implementation of the MAR protocol.

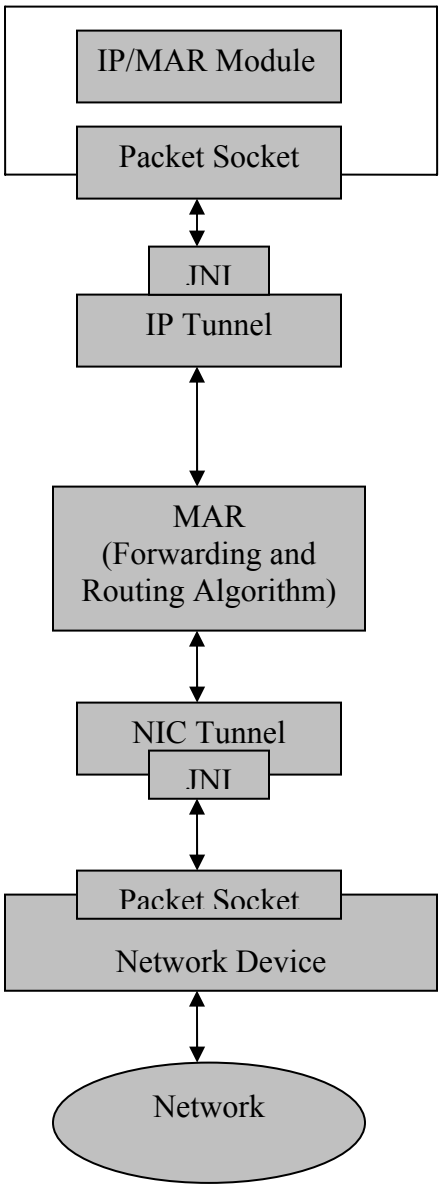

Figure 25: MAR Implementation

GUI/Main Server

As mentioned, the robot is controlled via a HP iPAQ 5550. To ensure easy user control of the robot we have provided a custom built Graphical User Interface (GUI). The GUI has a

combination of control area from which robot movement commands (stop, start, turn) can be issued and also a live feed video from the camera mounted on the hood of the tank. Thus the user is continuously aware of the current environment and the robot state and position. Further the robot position is being continuously tracked in GUI area which has the current environment input as a map. The GUI has provision to receive and display sensor data. The amount of power supplied to motors can also be controlled from the GUI. This adds the ability of controlling the speed of movement of the robot to maneuver rough terrains and left-right turns.

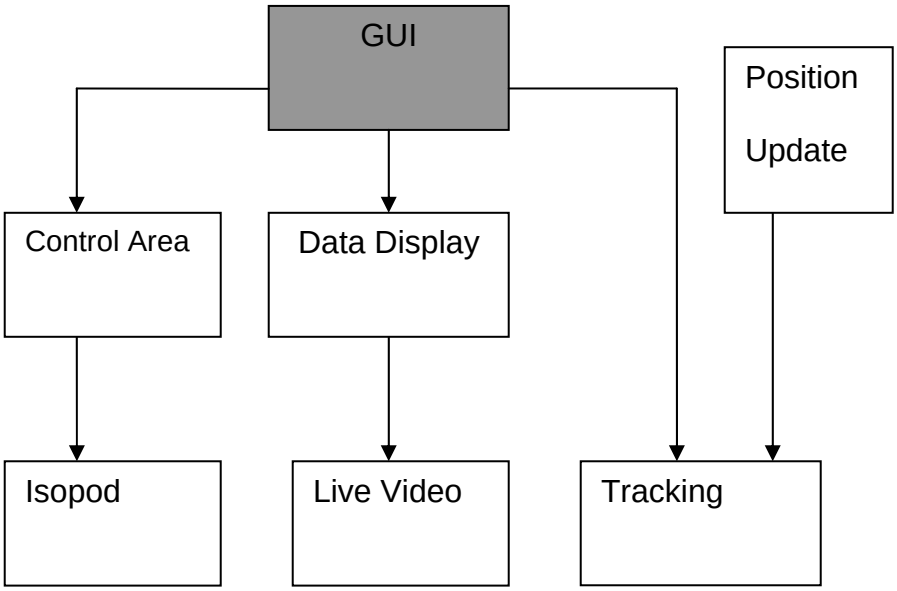

Figure 26: High Level Design of GUI

Figure-25 illustrates the high level design of the GUI. The GUI area is mainly divided in to 3 components.

- Control Area: The control area provides the user a variety of commands that can be issued to the robot to control its movements. Furthermore it also provides the ability to change the speed the motion of the robot.
- Tracking Area: This area provides the relative location of the robot in the current environment. Knowledge of the current position enables the user to make effective decisions and issue commands.
- Display: The display area is a combination of text and video. The onboard CCD camera provides a continuous live feed of the robot environment. The text area provides the current speed, geo-coordinates, inclination with respect to earth and the current heading angle. The data is obtained from the IMU units and the GPS.

Geolocation

Now that we have looked at the components and data necessary for geolocation, we shall delve into method of geolocation implemented. The main issue in geolocation is to determine the velocity along the magnetic North and East components given the linear distance traveled by the robot. The linear distance is obtained from the shaft encoder. The angular heading Φ is obtained from the gyroscope. The velocity along the North and East components are given by:

$V_{North}(t)$ = Linear Velocity * Cosine (Φ). -----18

$V_{East}(t)$ = Linear Velocity * Sine (Φ). -------19

The velocity along the components are integrated over a time domain to obtained the distance traveled in the North-East direction inducing change in the latitude-longitude co-ordinates.

$$Distance_{North} = \int_{to}^{t1} V_{North}(t)\, dt \quad \text{-------20}$$

$$Distance_{East} = \int_{to}^{t1} V_{East}(t)\, dt. \quad \text{--------21}$$

Thus we have obtained the amount of distance traveled along the North and East directions. The distance represents the linear movement along the axes. A further conversion to angular distance in terms of number of degrees, minutes, seconds is performed to obtain the actual change in the latitude-longitude. The table below lists the various conversion constants.

	Meters	Miles	Nautical Miles
1 Degree	93777	58.267	60
1 Minute	1562.95	0.971	1
1 Second	26.04	0.016	0.167

Table 5: Latitude-Longitude Conversion

The linear distance obtained is divided by the appropriate value from the table. Thus we obtain the change in the latitude-longitude. This change is added to the previously known position to obtain the new set of co-ordinates.

The flowchart illustrates the flow of operation.

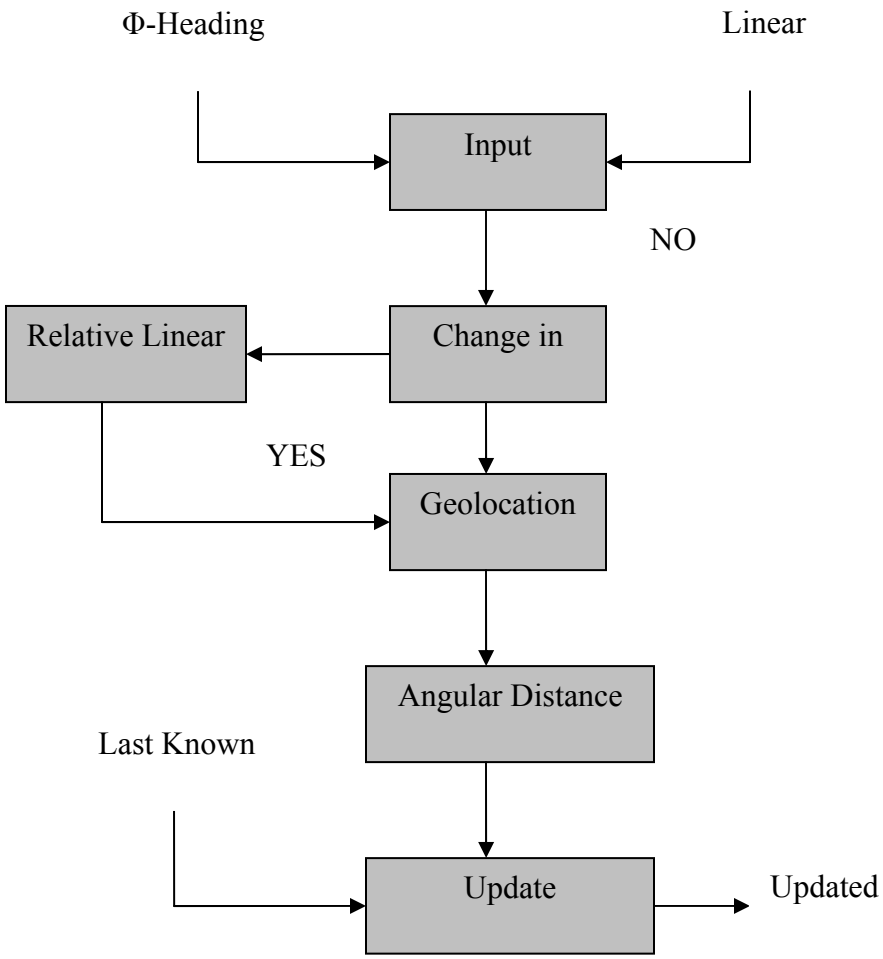

Figure 27: Geolocation

CHAPTER SIX: RESULTS AND CONCLUSIONS

In this chapter we shall present the various results and analysis that have been performed. We have established the mathematical model for Kalman Filter. The Kalman Filter gets its input from the Gyroscope and the data is filtered. The filtered data is used in the process of geolocation. We shall first present the analysis related to Kalman Filter and subsequently look at geolocation.

We have used the robot that we have developed as a platform for all our experiments. The robot specifications are as described in Chapter-4.

Kalman Filter Analysis

Kalman Filter implementation forms a very important part of our overall implementation. The important part of Kalman Filter is establishing a mathematical model for the sensor error characteristics. An accurate modeling of the error characteristics is very important for the effective implementation of the Kalman Filter. We have performed offline tuning to obtain accurate sensor error characteristics.

Sensor Error

We consider the roll, pitch, yaw outputs obtained from the gyroscope. We need to obtain the individual error co-variances of each of the quantities. This is performed by offline tuning of the gyroscope. We have obtained a large set of readings, typically 100 iterations for each of roll, pitch and yaw. We have obtained the standard deviation for a set of values. This test was performed for over 20 data sets and the standard deviation averaged. This provides us the

average standard deviation of error for roll, pitch and yaw. The experiment was conducted over a period of time to record any external conditions that influence the functioning of the gyroscope. Figure-26,27,28 illustrates the variation of standard deviation of roll, pitch and yaw over time.

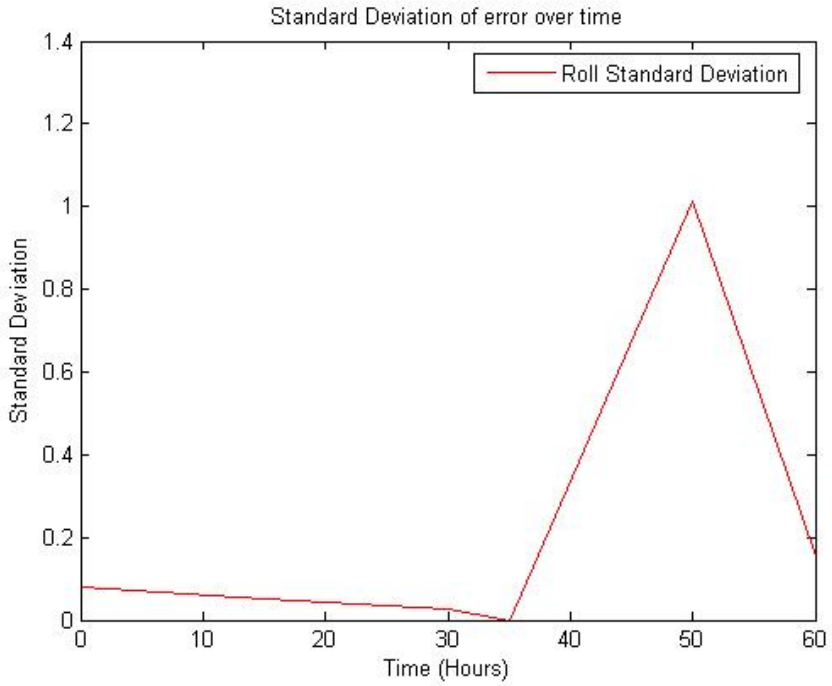

Figure 28: Standard Deviation of Roll Error

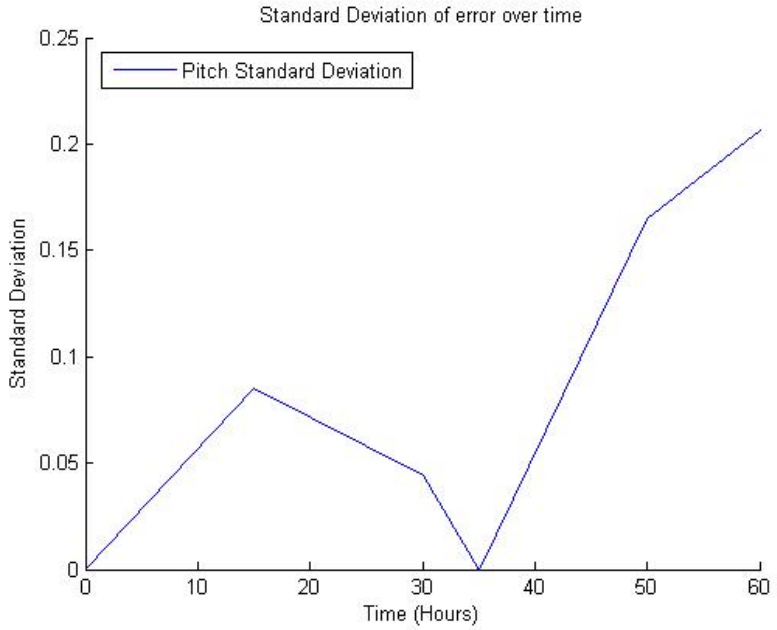

Figure 29: Standard Deviation of Pitch over time

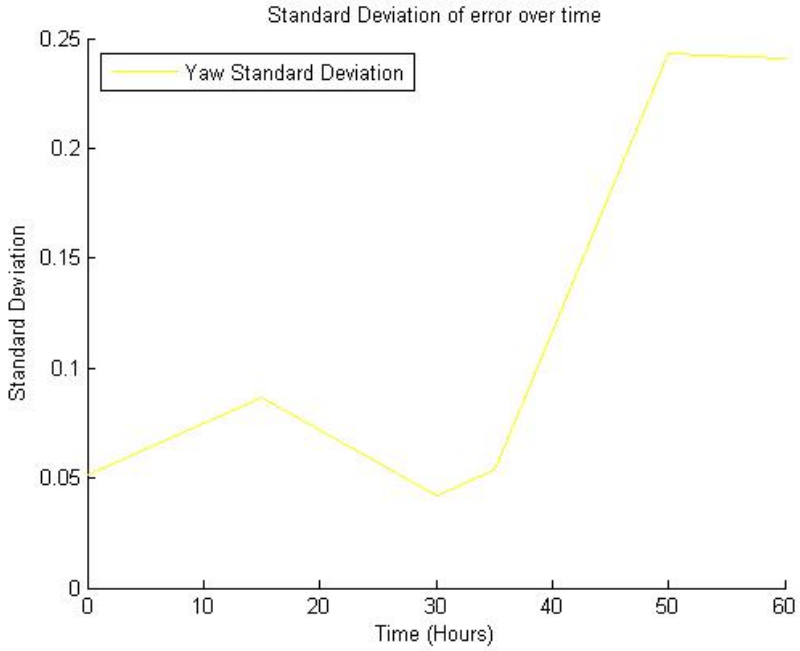

Figure 30: Standard Deviation of Yaw over time

Kalman Filter tuning forms a very important part of designing a Kalman Filter. The standard deviation that we have obtained cannot be directly used as the error covariance in a Kalman Filter. We have performed a number of simulation of the Kalman Filter to obtain the accurate co-efficient of the error covariance matrices in the Kalman Filter.

Pk - n x n covariance matrix of state vector, Qk – n x n Covariance matrix and Rk – m x m covariance matrix of measurement noise; are the matrices that model error in a Kalman Filter.

Kalman Filter

As we mentioned before, Kalman Filter forms a central part of the thesis. The geolocation algorithm is inherently dependent on the Kalman Filter to obtain the corrected roll, pitch and yaw values. In our case we just need to consider the pitch and yaw as we consider only 2-D motion on land surface.

The Kalman Filter is a Linear Estimator and with time the input data and the corrected data should converge. This shows that the Kalman Filter is being adaptive to the changes and errors in the readings. Further this also verifies that the robot is being held in a known trajectory and thus is track able.

We have performed an average of 100 runs for all the data plots of Kalman Filter. The convergence factor is easy to observe even with a medium sized sample population. This also further vindicates the fact that accurate modeling of sensor error helps in the relatively fast convergence of the Kalman Filter. The roll and pitch have value ranging from -180° to +180°, while pitch has a range -90° to +90°. The plots compare the variation of sensor input data (with noise) and filtered data over time. From Figure-28 we see that the convergence of pitch takes a

long time and also faithfulness of measured and filtered data are not totally accurate. This is due to fact that when the robot goes up/down a slope a change in pitch is induced. In our test environments we didn't have smooth slopes and also there were sudden drops in the angle of the slope. These are some of reasons for the variance in the pitch values.

The best estimate was obtained for the yaw measurements. From Figure-30 we see that the measured value of pitch and the filtered value are converging quickly and also are faithful and follow each other. This was due to the fact that we were able to obtain a better mathematical model of sensor noise relating to yaw. Figure-29 shows the variation of roll values. We do not consider roll values for further calculations and we have shown it here for completeness.

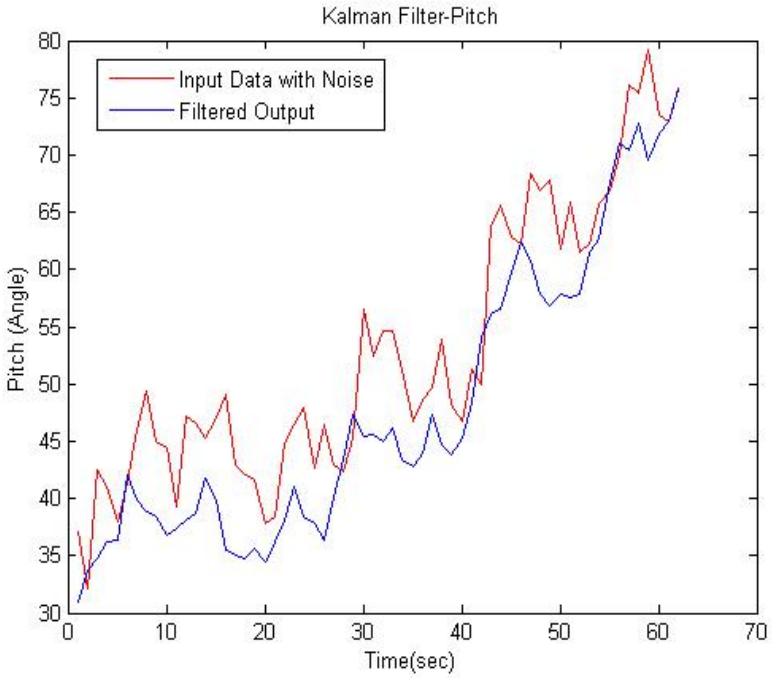

Figure 31: Kalman Filter-Pitch

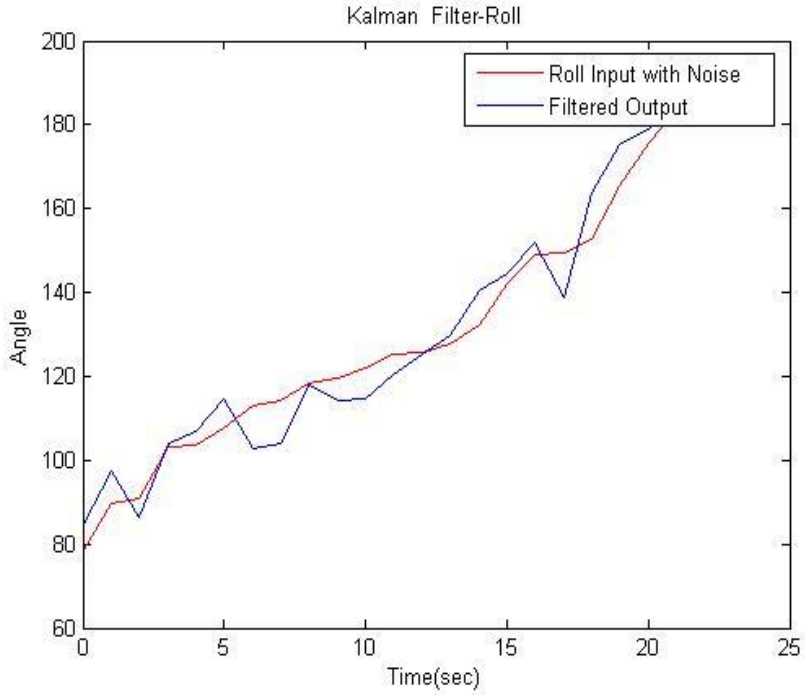

Figure 32: Kalman Filter-Roll

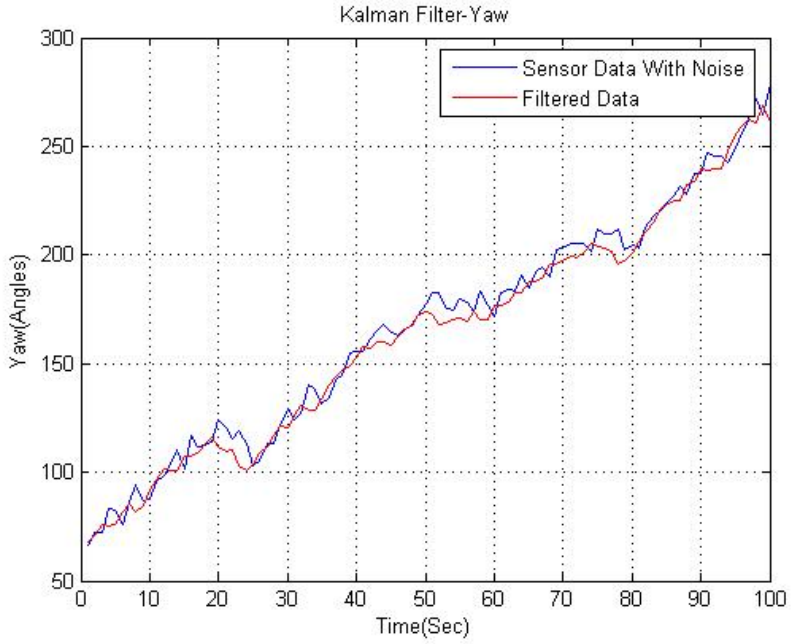

Figure 33: Kalman Filter-Yaw

To better understand the point that the Kalman Filter eliminates noise in the data, we have converted the readings in the time domain in to frequency domain. In the frequency domain, if any noise is eliminated from the input data we get a smoother curve. To obtain plots in the frequency domain we have obtained the Fast Fourier Transform (FFT). We obtain the phase and magnitude values of the input data.

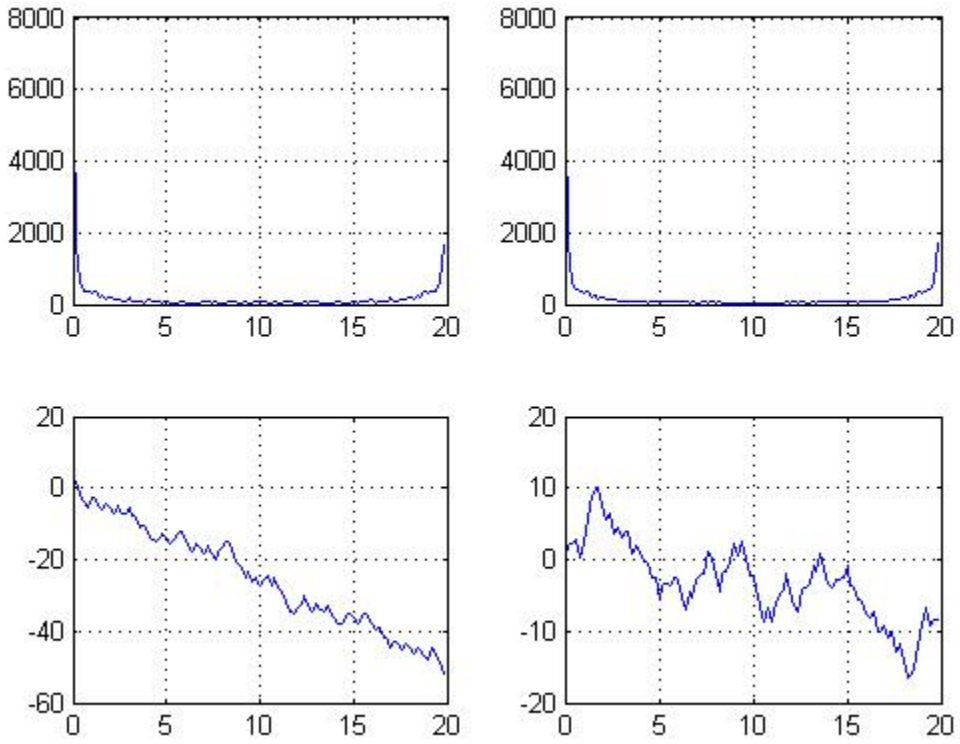

Figure 34: Yaw noise in Frequency Domain

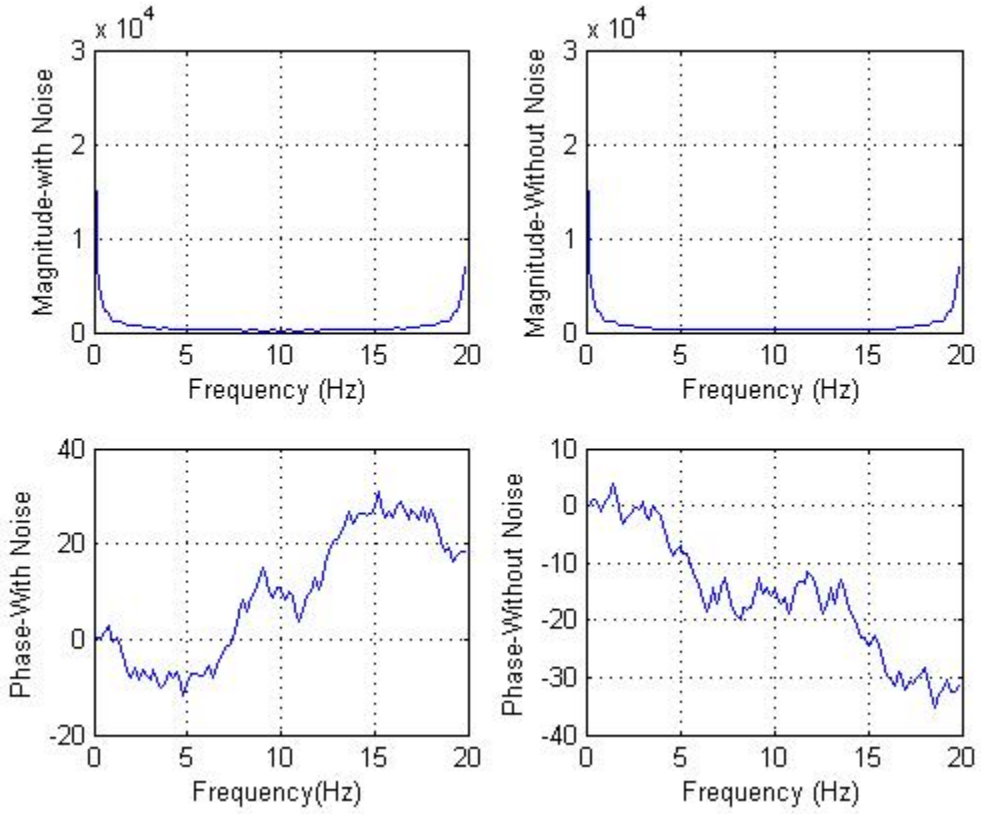

Figure 35: Yaw noise in Frequency Domain

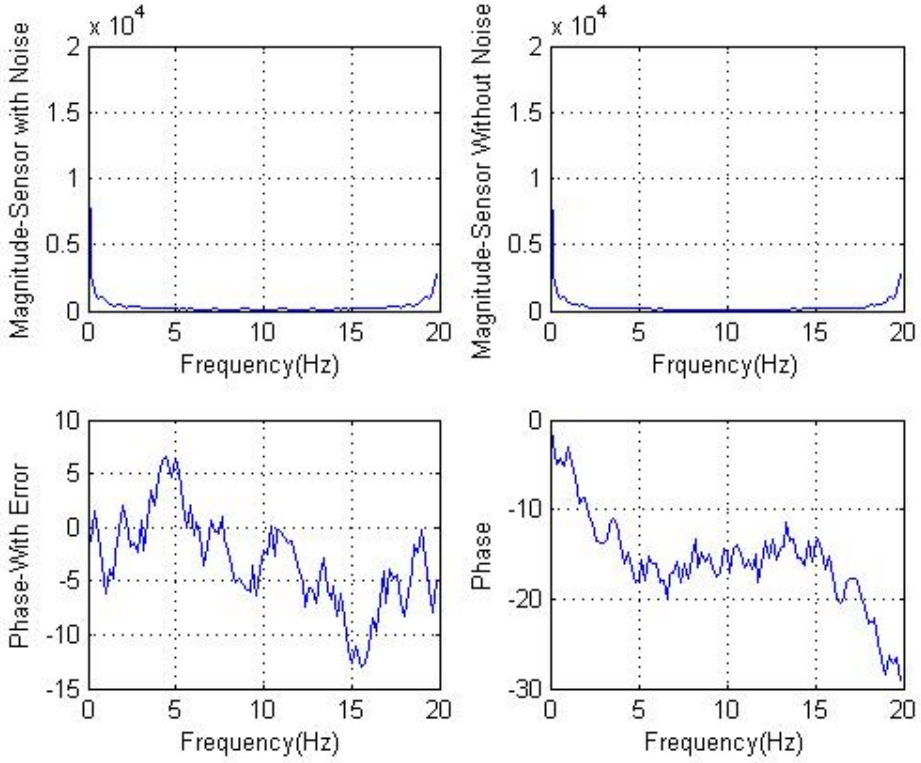

Figure 36: Yaw noise in Frequency Domain

In Figure-34 we have shown the yaw noise in the frequency domain and we shall consider this for discussion. In the plot of frequency versus magnitude we have shown the elimination of noise frequencies in the range of 5-15Hz. This is further visible in the phase plot, where the phase is modulated to eliminate the erroneous frequencies.

Error Characteristics of Kalman Filter

In this section we shall look into the error characteristics of the noise in the sensor data. This noise is obtained by a comparison between the measured data and the filtered data. The variation of noise with respect to pitch and yaw is shown in Figure-31. We see that the pitch data is

affected by a noise and also the noise is not consistent over a set of samples. This inconsistent noise delays the convergence of the Kalman Filter. As outlined in chapter four, the sensor noise input to a Kalman Filter is assumed to be Gaussian White Noise. This is a critical requirement of the Kalman Filter as it does not effectively process other types of noise. We have performed the Kolgomorov-Smirnov test determine the type of noise in a signal.

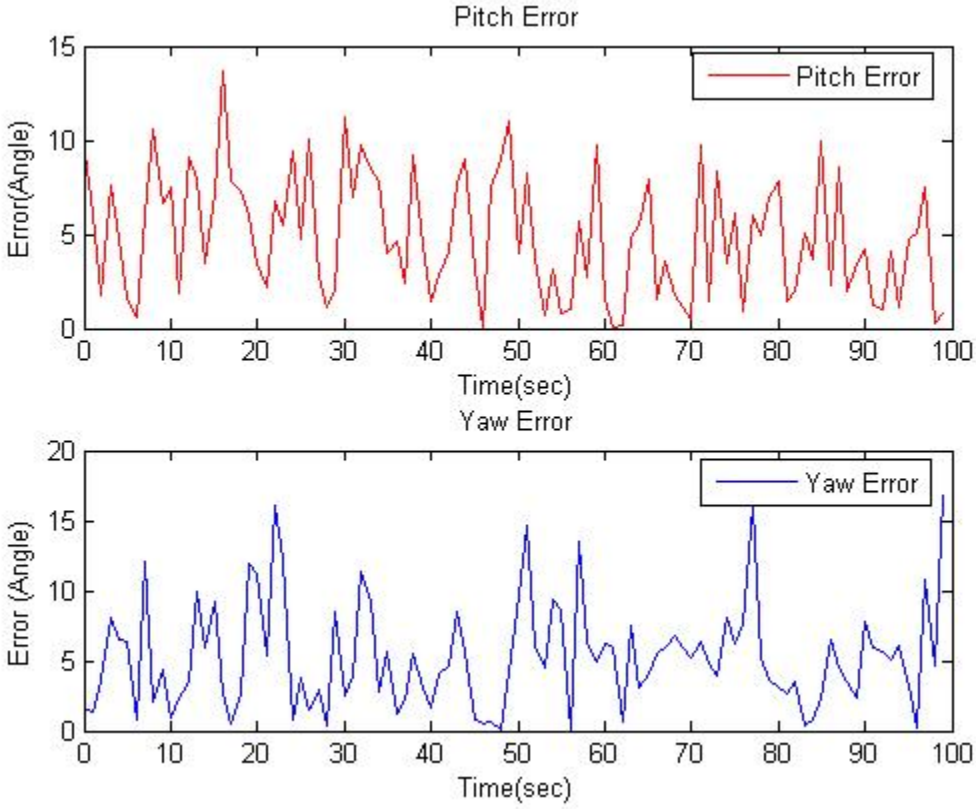

Figure 37: Pitch, Yaw Error

Geolocation

In this section we shall present the results associated with indoor geolocation. We have performed a number of tests outdoors. This is to validate our claims of geo-location as we have

access to GPS signal data. Hence we can compare the result obtained by our geolocation algorithm.

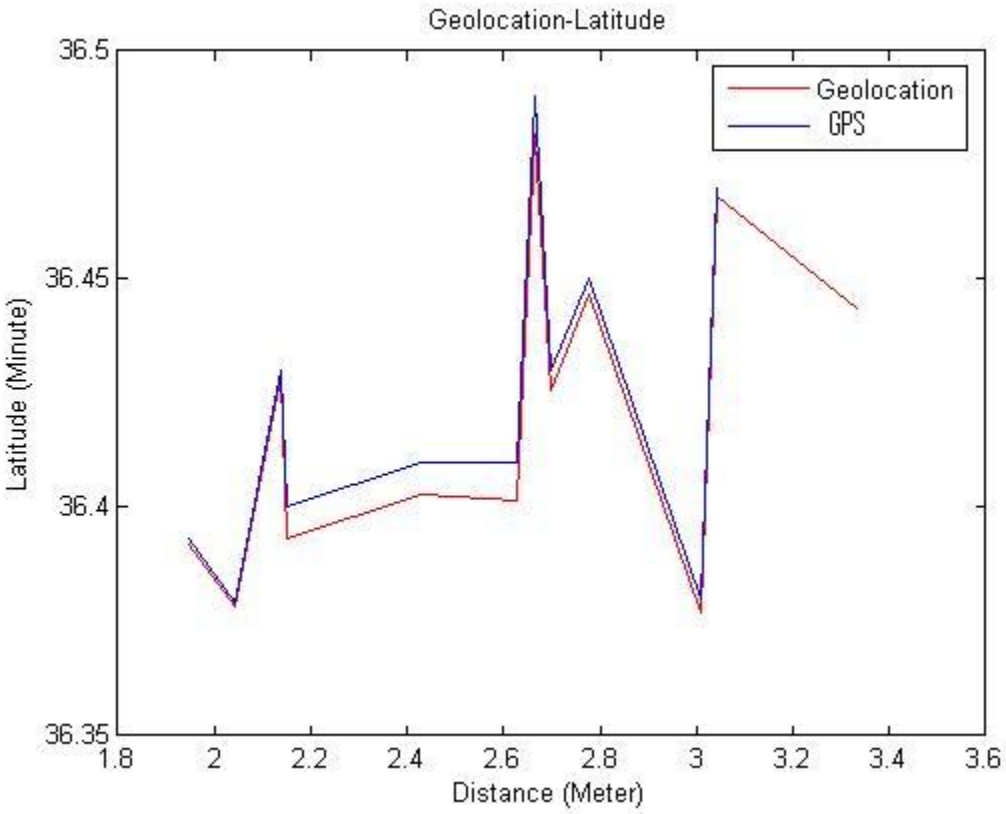

Figure 38: Indoor Geolocation-Latitude

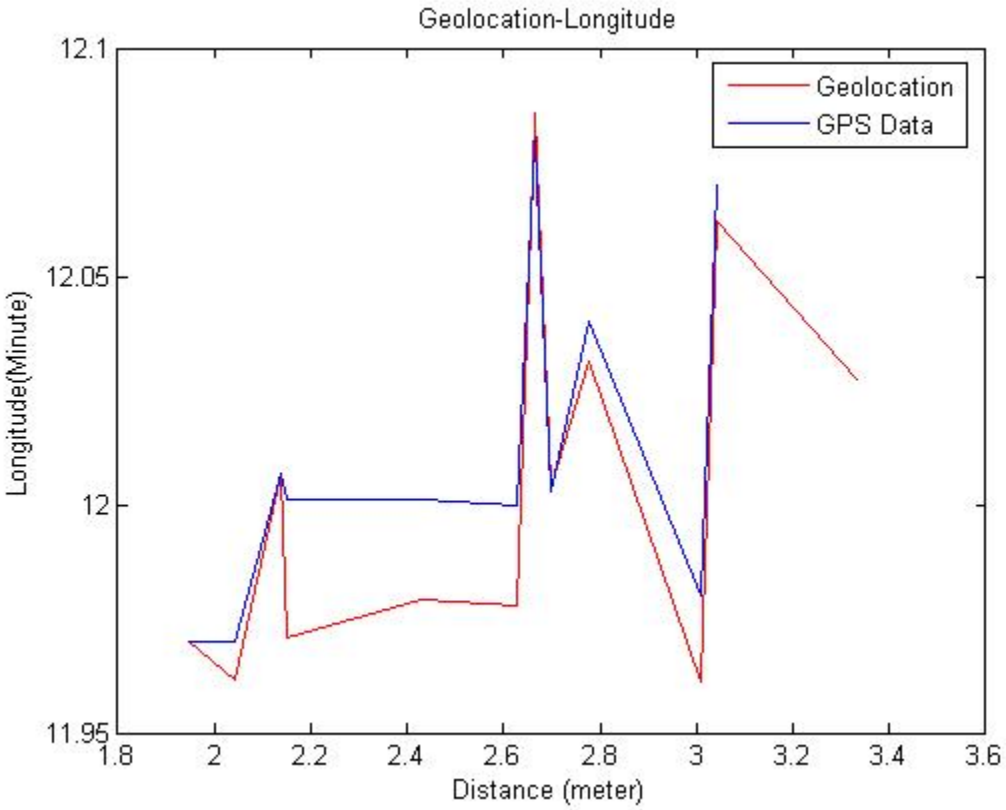

Figure 39: Indoor geolocation-Longitude

Figure-36, 37 indicate the comparison of data obtained from GPS and the results calculated by the geolocation algorithm. The X-axis denotes the distance traveled and the y-axis denotes the minutes for latitude and longitude. We do not indicate the degrees as it requires us to move 93774 meters. We see a large variation in the calculated co-ordinates and the GPS data in the range between 2.2-2.6m on the y-axis. This was due to a drastic shift in the pitch angle. If there is no smooth slope, the jerks induced into the gyroscope affects the pitch readings and hence the calculated values. A smoother down slope was considered in the range 2.65-3m. As the plot indicates there is a much better tolerance in this case. We shall now look at the effect of using a Kalman Filter and also the results obtained when not using a Kalman Filter.

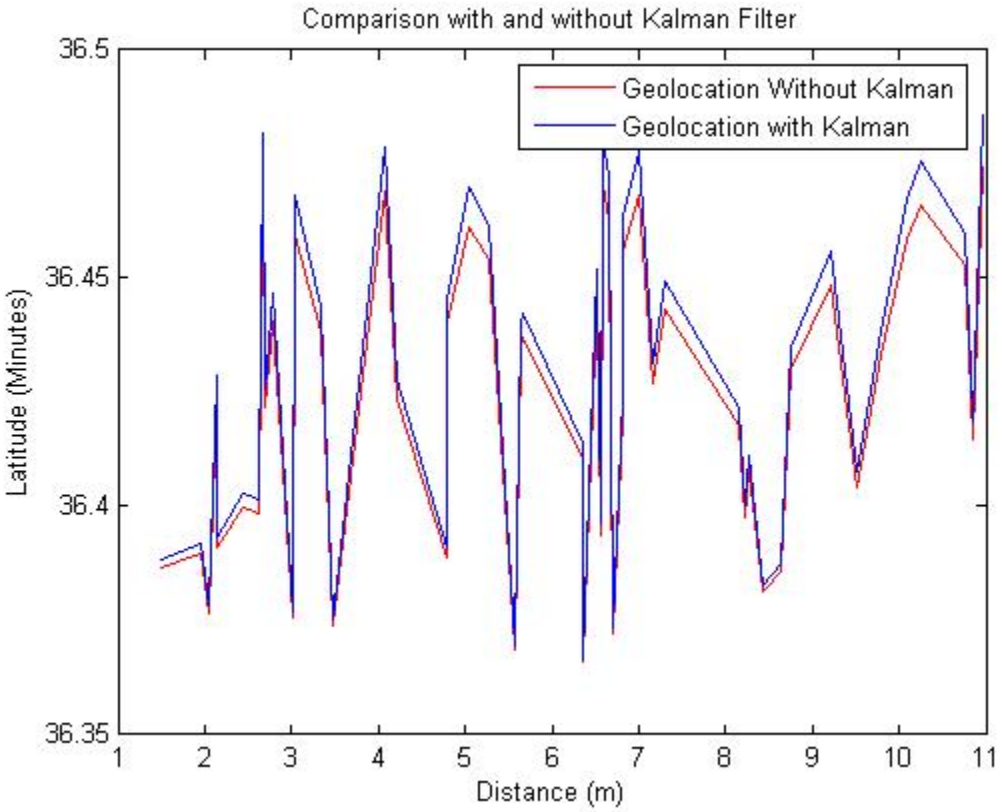

Figure 40: Latitude with and without Kalman

The above plot indicates the effect of using Kalman Filtered data for geolocation as compared to using the data directly from the gyroscope. This test was performed to indicate the gradual accumulation of error over time that affects the accuracy of the geo-location algorithm. At the beginning of the trial, the variation between the results obtained is negligible. But with time the errors keep getting added and the drift from the actual value is significant.

Figure-39 illustrates the error accumulated when for longitudinal co-ordinates.

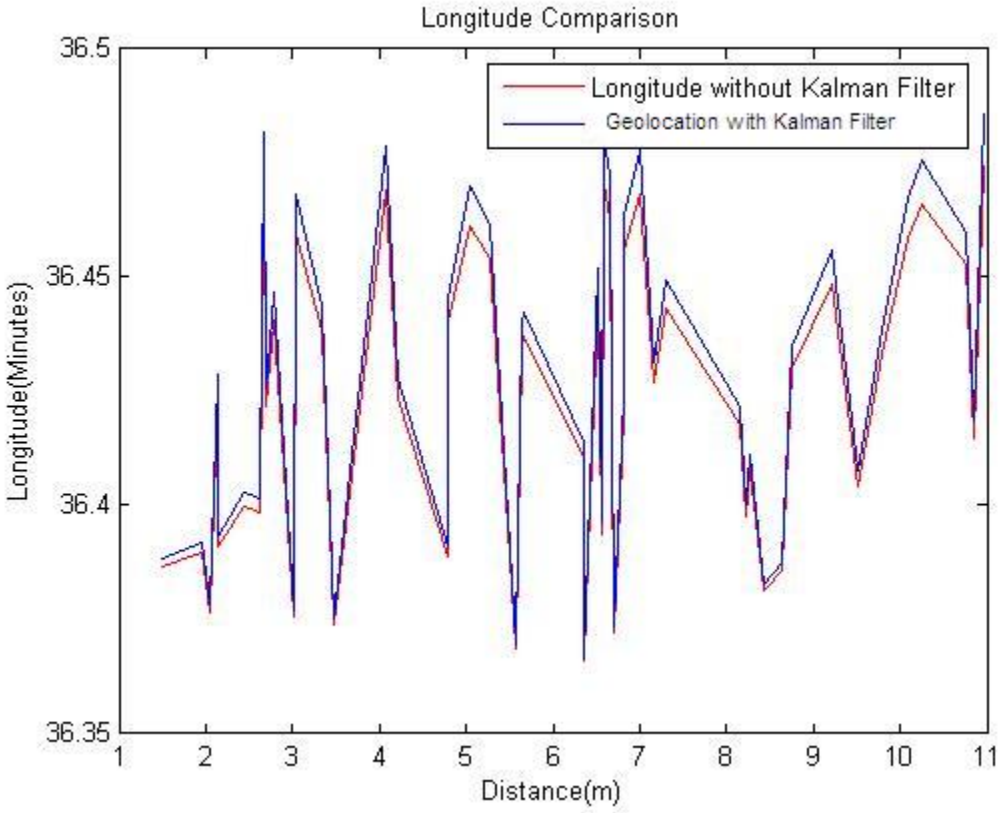

Figure 41: Longitude with and without Kalman

In Figure-40 we illustrate the average error accumulated over time. At the start of the trial the error induced is .002 minutes or 3.125m. This error keeps accumulating as the trial progresses. We have an average error of .006minutes or 9.375m which is quite significant.

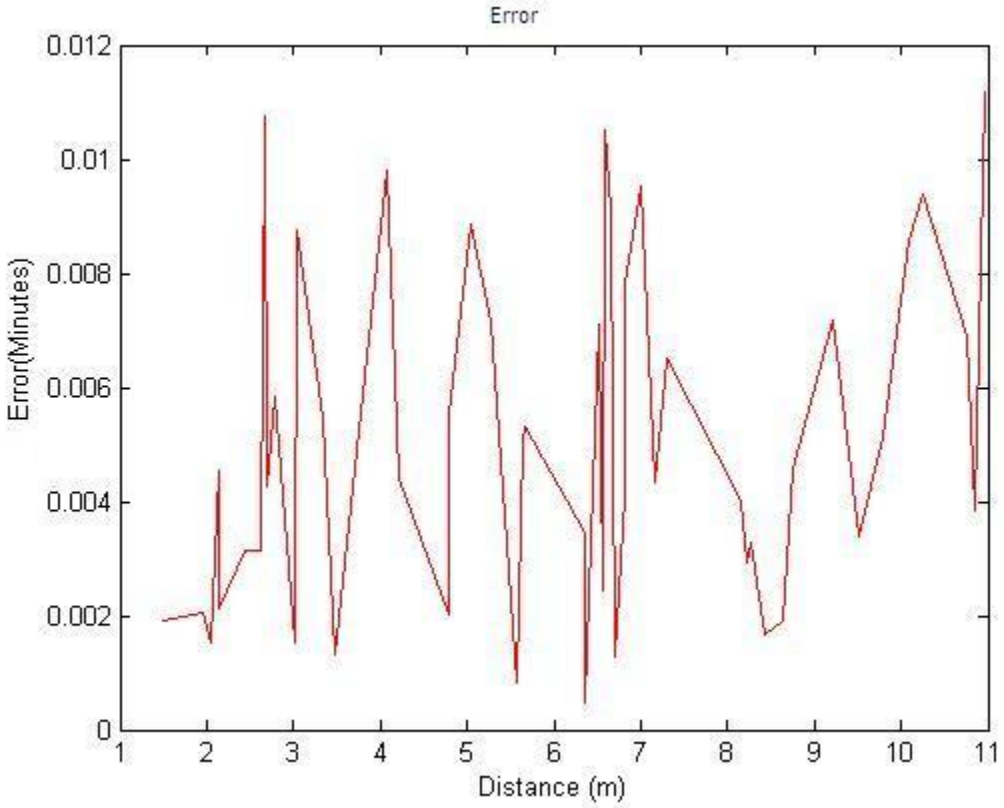

Figure 42: Geolocation Error

Conclusions

In this research, we proposed a novel method of integrating the GPS and IMU measurements for effective geolocation and tracking of mobile robots. INS mechanization model was developed and navigation solutions obtained by integration with GPS signals. The GPS-INS was combined in a tightly-coupled Kalman Filter. Kalman Filter provides an effective, stable and easy to use estimator to eliminate sensor error drifts. The main advantage of the system is the ability to

predict the geographical co-ordinates using the INS data when there are GPS outages. The performance of the system was tested both indoors and outdoors. Some of the main achievements of the research are:

- The achievable accuracy of GPS/INS integration using a medium accuracy IMU is at the meter-level using both simulated data and land vehicle field test data when consistent GPS updates are available. The GPS updates were at a 1 Hz data rate in this case.
- The geo-location algorithm is run continuously increasing the positioning accuracy and continuous update when GPS signal is available.
- The Kalman Filter has a sequential convergence property and provides an effective mechanism to combine data from a variety of sensors.
- An easy to use GUI provides a constant position update to the user to help make tactical decisions.
- Constant feedback from the IMU unit helps keep the robot on target and also detects any variation in the course.
- We have created a secure encrypted network to keep away any defaulters. Hence the application is suited for high-security applications. Further extension to the network is done easily.
- Enabled multiple users to issue control commands to the robot. Commands are queued and are executed on a FCFS basis, with the exception of highest priority commands such as stop which is executed immediately and the queue cleared.

Future Work

The research conducted here provides a solid platform for future work and improvements. Based on the obtained results we put forth certain recommendations.

- In order to improve the IMU prediction accuracy during GPS outages, an initial calibration and gyro drift testing algorithm should be introduced and applied before the system starts to work.
- The current robot is semi-autonomous and assumes the knowledge of current environment. The robot can be made autonomous by the use of combination of SONAR and Laser. The robot can autonomously detect obstacles in the map and navigate.
- The current system can be combined with an Indoor Mapping Robot to achieve continuous Map building and Geolocation. This application would be particularly useful in completely unknown territories.
- A system can be extended to include multiple/team of robots performing a variety of tasks under the guidance of the robot performing geolocation.
- Image processing algorithms can be used for obtaining high-resolution images obtained from the CCD camera mounted aboard the robot.

REFERENCES

1. Cannon, M.E., *Airborne GPS/INS with an Application to Aerotriangulation*. UCGO Report 20040, 1991. Department of Geomatics Engineering, University of Calgary.

2. Scherzinger, B., *Robust Inertially-Aided RTK Position Measurement*, 2001. Proceedings. of the International Symposium on Kinematic Systems in Geomatics and Navigation. Banff, Canada, June 5-8, 2001. pp. 265-272.

3. Gustafson, D., Dowdle, J., Flueckiger, K.A *Deeply Integrated Adaptive GPSBased Navigators with Extended Range Code Tracking*. Proceedings of PLAN IEEE 2000. San Diego, CA. March 13-16, 2000, pp 118-124.

4. Greenspan, R.L., *GPS and Inertial Integration, In: Global Positioning System: Theory and Applications*, 1996, Chapter 7, Vol. II.

5. I.J. Cox and G.T. Wilfong. *Autonomous Robot Vehicles*. Springer Verlag, 1990.

6. R. Simmons, E. Krotkov, L. Chrisman, F. Cozman, R. Goodwin, M. Hebert, G. Heredia, S. Koenig, P. Muir, Y. Shinoda, and W. L. Whittaker. *Mixed-Mode Control of Navigation for a Lunar Rover*. In Proceedings of the SSI/Princeton Space Manufacturing Conference**,** 1995.

7. Alberto Elfes. *Using occupancy grids for mobile robot perception and navigation*. Computer, 1989, 22(6):46-57.

8. J.R. Burch, V. Singhal. *Robust latch mapping for combinational equivalence checking*. In Proc. International Conference on Computer Aided Design, San Jose, 1998, pp.563--569.

9. Wolfram Burgard, Dieter Fox, Sebastian Thrun. *Active Mobile Robot Localization by Entropy Minimization.* Proceedings of the 2nd Euromicro Workshop on Advanced Mobile Robots, IEEE/CS, 1997.

10. Alan C. Schultz, William Adams. *Continuous localization using Evidence Grids.* Proceedings of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 1998, pp 2833-2839.

11. Alan C. Schultz,William Adams, Brian Yamauchi. *Integrating Exploration, Localization, Navigation and Planning with a common representation*. Proceedings of the 1998 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '98), Gaithersburg, MD, September 1998.

12. Alan C. Schultz, William Adams, Brian Yamauchi. *Mobile robot exploration and map building with Continuous Localization*. Proceedings of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 1998.

13. Alberto Elfes. *Using Occupancy Grids for Mobile Robot Perception and Navigation*. Computer, 22(6):46-57, 1989.

14. Sutherland, K.T. and Thompson, W.B. 1994. *Localizing in unstructured environments: Dealing with the errors*. IEEE Transactions on Robotics and Automation, 10(6):740–754.

15. Greiner, R. and Isukapalli, R. 1996. *Learning to select useful landmarks*. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 26(3):437–449.

16. Thrun, S. *Bayesian landmark learning for mobile robot localization.* Machine Learning, 1998, 33(1):41–76.

17. David C.K. Yuen, Bruce A. MacDonald. *Natural landmark based localisation system using panoramic images*. In Proceedings of Image and Vision Computing New Zealand (IVCNZ), Auckland, New Zealand, 2002, pages 335-340.

18. Clark F. Olson. *Selecting Landmarks for Localization in Natural Terrain*. Autonomous Robots, 12(2):201-210, 2002.

19. Margrit Betke and Leonid Gurvits. *Mobile Robot Localization Using Landmarks*. In Proceedings of the IEEE International Conference on Robotics and Automation,vol 2,1994, pp 135-142.

20. F. Dellaert, A. W. Stroupe. *Linear 2D localization and Mapping for Single and Multiple Robot Scenarios.* In Proceedings of the IEEE International Conference on Robotics and Automation, 2002, IEEE, 2002.

21. N. M. Kwok, G. Dissanayake. *Bearing-only SLAM in Indoor Environments Using a Modified Particle Filter,* IEEE Transactions of Robotics and Automation, 2001.

22. F. Dellaert, W. Burgard, D. Fox, S. Thrun. *Monte Carlo Localization for Mobile Robots*. In Proceddings of IEEE International Conference on Robotics and Automation (ICRA99), 1999.

23. D. Fox, W. Burgard, F. Dellaer, S. Thrun*. Monte carlo localization: Efficient position estimation for mobile robots.*. AAAI-99.

24. R. Simmons, S. Koenig. P*robabilistic Robot Navigation in Partially Observable Environments*, Proceedings of the International Joint Conference on Artificial Intelligence, 1995, 1080--1087.

25. W. Burgard, D. Fox, S. Thrun. *Markov Localization for Reliable Robot Navigation and People Detection.* In Proceedings of the Dagstuhl Seminar on Modelling and Planning for Sensor-Based Intelligent Robot Systems, 1999, pp 1-20.

26. W. Burgard, D. Fox, S. Thrun. *Markov Localization for Mobile Robots in Dynamic Environments*, Journal of Artificial Intelligence Research, vol 11, pp 391-427,1999.

27. D.Pierce, B.Kuipers. *Learning to Explore and Build maps*. Proceedings of the 12th National Conference on Artificial Intelligence, 1994, pp 1264-1271.

28. M.J.Mataric. *A Distributed Model for Mobile Robot Environment-Learning and Navigation.* Master's Thesis, MIT Cambridge, 1990.

29. B.Yamauchi, R.Beer. *Spatial Learning for Navigation in Dynamic Environments*. IEEE Transactions on Systems, Man and Cyberbnetics-Part B:Cybernetics, 1996.

30. U. R. Zummer. *Robust World-Modeling and Navigation in a real world*. Neurocomputing, 1996, Vol. 13, pp 2-4

31. D. Kortenkamp, T. Weymouth. *Topological Mapping for Mobile Robots Using a combination of sonar and vision sensing.* Proceedings of the 12th National Conference on Artificial Intelligence, 1994, pp 979-984.

32. S.Engelson, D.McDermott. *Error Correction in mobile robot map learning.* Proceedings of the 1992 IEEE International Conference on Robotic and Automation, 1992, pp 2555-2560.

33. H. Choset, K. Nagatani. *Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization without Explicit Localization*. IEEE Transactions on Robotics and Automation, 2003, Vol 19, No 3, pp 513-523.

34. N. Tomatis, I. Nourbakhsh, R.Siegwart. *Hybrid Simultaneous Localization and Map Building: Closing the Loop with Multi-Hypotheses Tracking*. In Proceedings of the IEEE International Conference on Robotics and Automation, 2002.

35. J. Pfeiffer. *A Language for Geometric Reasoning in Mobile Robots*, Visual Languages, 1999.

36. F. Aurenhammer. *Voronoi diagrams-survey of a fundamental geometric data structure,* ACM Comput. Surv., vol 23,No.3, 1991, pp 345-405.

37. H. Choset, I. Konukseven, J. Burdick. *Mobile Robot Navigation: Issues in Implementation the Generalized Voronoi Graph in the Plane*, IEEE/MFI, 1996.

38. H. Choset, I. Konukseven, A. RIzzi. *Sensor Based Planning: A Control Law for Generating the Generalized Voronoi Graph.* IEEE/ICAR, 1997.

39. H. Choset, K. Nagatani, A. Rizzi. *Sensor Based Planning: Using a Honing Strategy and Local Map Method to Implement the Generalized Voronoi Graph*. SPIE Mobile Robotics, 1997.

40. H. Choset, J. Burdick. Sensor-Based Exploration.*The Hierarchical Generalized Voronoi Graph*. The International Journal of Robotics Research , 2000, pp 96-125.

41. N. Deo. *An introduction to Graph Theory.*

42. H. P. Moravec. *Sensor Fusion in Certainty Grids for Mobile Robots*. AI Magazine, 1988, pp 61-74.

43. H. Moravec, A. Elfes. *High-resolution maps from wide-angle sonar*. In Proc. IEEE International Conference on Robotics and Automation, 1985.

44. G. Seeber. *Satellite Geodesy: Foundations, Methods & Applications*. Walter de Gruyter, Berlin New York, 1993, 531pp.

45. J.J. Spilker.*GPS signal structure and performance characteristics*. In proceedings of Navigation, reprinted by the U.S. Inst. of Navigation, 1980, vol.1, 29-54.

46. K. Basye, T.Dean, J. Kirkman, M. Lejter. *A decision theoretic approach to planning, perception and Control*.Expert, IEEE , Volume: 7 , Issue: 4 , Aug. 1992.

47. D. Heckerman. *A tutorial on learning with Bayesian networks*. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

48. G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

49. R. Kalman, *A new approach to linear filtering and prediction problems.* Transactions ASME Journal of Basic Engineering82, 1960, 35-44.

50. M. Grewal and A. Andrews, *Kalman filtering: theory and practice*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1993). H. G.

51. G. Welch and G. Bishop, *An Introduction to the Kalman Filter*. Chapel Hill (2001). SIGGRAPH 2001.

52. X. Liu and A. J. Goldsmith. *Kalman Filtering with Partial Observation Losses*, Proc. IEEE Conference on Decision and Control, 2004.

53. P.Krishnamurthy and K.Pahlavan. *Analysis of the Probability of Detecting the DLOS Path for Geolocation Applications in Indoor Areas.* 49th IEEE Vehicular Technology Conference, Houston May 1999.

54. K. Pahlavan, X. Li, J. Mäkelä. *Indoor Geolocation Science and Technology*. IEEE Communications Magazine, vol. 40, no.2, 2002, pp. 112 - 118.