AN IMPLEMENTATION OF MONO AND STEREO SLAM SYSTEM UTILIZING EFFICIENT MAP MANAGEMENT STRATEGY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ADNAN KALAY

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2008

Approval of the thesis:

AN IMPLEMENTATION OF MONO AND STEREO SLAM SYSTEM UTILIZING EFFICIENT MAP MANAGEMENT STRATEGY

submitted by **ADNAN KALAY** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Assist. Prof. Dr. İlkay Ulusoy _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. İlkay Ulusoy _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Erol Şahin _____
Computer Engineering Dept., METU

Assist. Prof. Dr. Afşar Saranlı _____
Electrical and Electronics Engineering Dept., METU

Sevda Erdoğdu (MSc.) _____
ASELSAN Inc.

Date: 05.09.2008

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Adnan KALAY

Signature :

ABSTRACT

AN IMPLEMENTATION OF MONO AND STEREO SLAM SYSTEM UTILIZING EFFICIENT MAP MANAGEMENT STRATEGY

Kalay, Adnan

M. S., Department of Electrical and Electronics Engineering
Supervisor : Assist. Prof. Dr. İlkay Ulusoy

September 2008, 181 pages

For an autonomous mobile robot, localization and map building are vital capabilities. The localization ability provides the robot location information, so the robot can navigate in the environment. On the other hand, the robot can interact with its environment using a model of the environment (map information) which is provided by map building mechanism. These two capabilities depends on each other and simultaneous operation of them is called SLAM (Simultaneous Localization and Map Building). While various sensors are used for this algorithm, vision-based approaches are relatively new and have attracted more interest in recent years.

In this thesis work, a versatile Visual SLAM system is constructed and presented. In the core of this work is a vision-based simultaneous localization and map building algorithm which uses point features in the environment as visual landmarks and Extended Kalman Filter for state estimation. A detailed analysis of this algorithm is made including state estimation, feature extraction and data association steps. The algorithm is extended to be used for both stereo and single camera systems. The

core of both algorithms is same and we mention the differences of both algorithms originated from the measurement dissimilarity. The algorithm is run also in different motion modes, namely predefined, manual and autonomous. Secondly, a map management strategy is developed especially for extended environments. When the robot runs the SLAM algorithm in large environments, the constructed map contains a great number of landmarks obviously. The efficiency algorithm takes part, when the total number of features exceeds a critical value for the system. In this case, the current map is rarefied without losing the geometrical distribution of the landmarks. Furthermore, a well-organized graphical user interface is implemented which enables the operator to select operational modes, change various parameters of the main SLAM algorithm and see the results of the SLAM operation both textually and graphically. Finally, a basic mission concept is defined in our system, in order to illustrate what robot can do using the outputs of the SLAM algorithm. All of these ideas mentioned are implemented in this thesis, experiments are conducted using a real robot and the analysis results are discussed by comparing the algorithm outputs with ground-truth measurements.

Keywords: Robot Localization, Map Building, Visual SLAM, Extended Kalman Filter, Landmark Detection, Map Management

ÖZ

ETKİN HARİTA YÖNETİM STRATEJİSİ KULLANAN MONO VE STEREO SLAM SİSTEMİ UYGULAMASI

Kalay, Adnan

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Yard. Doç. Dr. İlkay Ulusoy

Eylül 2008, 181 sayfa

Otonom bir robot için lokalizasyon ve harita çıkarma yetenekleri hayati önem taşımaktadır. Lokalizasyon kabiliyeti robota konum bilgisi sağlayarak robotun navigasyonuna katkı sağlar. Harita çıkarma mekanizması ise robotun çevresiyle etkileşimde bulunması için ihtiyaç duyabileceği çevresel modeli, yani haritayı oluşturur. Bu iki yetenek birbirine bağımlıdır ve aynı anda işletilme durumuna SLAM (Eş Zamanlı Lokalizasyon ve Harita Çıkarma) denir. Birçok sensörün kullanıldığı bu algoritmada, görüntü tabanlı yaklaşımlar nispeten yenidir ve son yıllarda daha çok ilgi çekmektedir.

Bu tez çalışmasında, çok yönlü bir Görsel SLAM sistemi oluşturulmakta ve sunulmaktadır. Çalışmanın özünü çevredeki noktasal işaretleri görsel işaretler olarak algılayan, durum tahmini için de Genişletilmiş Kalman Filtresini kullanan görüntü tabanlı eş zamanlı lokalizasyon ve harita çıkarma algoritması oluşturmaktadır. Bu algoritmanın durum kestirimi, işaret çıkarımı ve veri eşlenmesi adımlarını da içerecek şekilde detaylı bir analizi yapılmaktadır. Algoritma, stereo ve

tek kameralı sistemlerde kullanılabilecek şekilde genişletilmektedir. İki algoritmanın da özü aynıdır ve ölçüm farklılıklarından doğan farklı sonuçlara değinilmektedir. Algoritma ayrıca ön tanımlı, manuel ve otonom olmak üzere farklı hareket modlarında çalıştırılmaktadır. İkinci olarak, özellikle geniş ortamlarda kullanılmak üzere bir harita yönetim stratejisi geliştirilmektedir. Robot SLAM algoritmasını geniş ortamlarda çalıştırdığında, oluşturulan harita doğal olarak fazla miktarda işaret içermektedir. Toplam işaret sayısı sistem için kritik bir seviyeyi aştığında, verimlilik algoritması devreye girmektedir. Bu durumda harita, yer işaretçilerinin geometrik dağılımını kaybetmeyecek şekilde seyreltilmektedir. Ayrıca geliştirilen grafiksel kullanıcı arayüzü ile operatöre operasyonel mod seçimi, ana SLAM algoritmasının çeşitli parametrelerini değiştirme ve SLAM operasyonunun sonuçlarını metinsel ve grafiksel olarak görme imkanı sunulmaktadır. Son olarak, SLAM algoritmasının çıktılarını kullanarak robotun neler yapabileceğini göstermek için, basit bir görev konsepti tanımlanmaktadır. Bu tezde tüm bu bahsedilen fikirler uygulanmakta, gerçek bir robot üzerinde deneyler yapılmakta ve analiz sonuçları algoritma çıktılarının gerçek konum ölçümleriyle karşılaştırılması suretiyle tartışılmaktadır.

Anahtar Kelimeler : Robot Lokalizasyonu, Harita Çıkarma, Görsel SLAM, Genişletilmiş Kalman Filtresi, Yer İşareti Tespiti, Harita Yönetimi

To My Mother

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Dr. İlkay Ulusoy for her guidance, advice, criticism, encouragement and insight throughout the completion of the thesis.

I am indebted to all of my friends and colleagues for their support and encouragements. I am also grateful to ASELSAN Inc. for the facilities that made my work easier.

Finally, I am grateful to my wife for her continuous support and encouragements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Definition and Motivation

With the continuous evolution of the technology, it becomes possible to use remote agents in many areas in order to attain several goals such as increasing the automation, removing the human operators from dangerous conditions, defense and mining. Development of new technologies provides better opportunities in terms of cost, speed and tractable algorithms which results in increased intelligence of these agents.

Robotic systems have taken a great progress for the last decades. Having the ability of implementing predefined instructions in many areas, robots are now attracting more attention due to their autonomy. Many independently acting systems have been constructed in different areas such as industry, space and underwater [7, 8, 9, 10 and 11]. The development of such systems requires reliability, safety and robustness properties in order to be acceptable in real world applications. For an autonomous robot, localization is a crucial capability that enables the robot to know where it is and navigate accordingly. In order to localize accurately, the map of the environment should be known. For some indoor applications such as industrial robots, the map is pre-known and localization is done according to this map. But in most cases, especially for outdoor applications the environment that the robot roams is not pre-known and the robot has to construct a map as it moves.

SLAM (Simultaneous Localization and Map Building) is the combination of two crucial tasks for autonomous robot localization, namely localization and map building. These tasks are done simultaneously in a way that the output of one task becomes the input of the other. The ability to simultaneously estimate the position of a robot and build the model of the environment is a challenging problem. Several methods making use of various sensors have been proposed to solve this problem. One of the most interesting methods employs cameras which are generally called as vision-based methods. The vision-based approaches for the SLAM problem are relatively new solution suggestions and need to be investigated in detail. There are two main solutions for the vision-based SLAM problem from the quantity of used sensors point of view, namely the methods utilizing single camera so called "Mono SLAM" [12, 13, 14] and utilizing stereo cameras so called "Stereo SLAM" [15, 16]. While the former suggests a solution with lower cost, the latter one seems to be still more robust. The advantages and the disadvantages between these two types of SLAM also need to be compared and present a considerable motivation for research [17].

1.2 Thesis Contribution

The major contributions of this thesis are related with constructing partially and fully autonomous simultaneous localization and map building capabilities utilizing visual sensors. The thesis develops a mathematical solution to the SLAM problem similar to the solutions based on Extended Kalman Filter in the literature, presents SLAM administration strategies and experimental results.

The principal contributions of this thesis are as follows:

- An online vision-based SLAM implementation employing Extended Kalman Filter is presented. Two main visual SLAM types, namely Mono SLAM and Stereo SLAM, are investigated, implemented and compared.

- A map management strategy to efficiently administer the SLAM operation in large environments and achieve adaptability to long-term runs is

presented. This strategy is also responsible for other important decisions such as adding new landmarks to the map, deleting obsolete features from the map etc. As it can be understood, the proposed map management technique provides necessary decision making mechanisms for necessary conditions.

- The thesis presents a mission concept, which means that the robot is given a mission to reach a specified destination. The constructed system provides the robot with necessary intelligence in order to achieve the given mission, although there may be obstacles on the potential robot path.

- A well-organized software tool, called SLAM Suite, which allows changing various parameters related to SLAM subtasks such as data association, landmark detection, feature deletion, feature remeasurement, robot system configuration, real-time efficiency algorithm parameters, camera parameters and predefined motion patterns is presented. One can interact with the robot system through this interface and make it run in various operational modes. Moreover, the running SLAM algorithms can be paused, continued or terminated at any time and the constructed maps can be viewed in graphical or textual formats.

- All of the approaches mentioned above are executed on a real and a simulator Pioneer robot, experiments are conducted, analyses are made and results are presented.

CHAPTER 2

THEORETICAL BACKGROUND AND RELATED WORKS

In this chapter, a theoretical background is provided and related works are given. The topics to be touched in this chapter are localization, map building, simultaneous localization and map building (SLAM), sensors used for SLAM and Visual SLAM.

2.1 Localization

Localization is the ability to answer the question "Where am I?" for a questioner's point of view, that is to determine the location of the questioner with respect to a defined reference frame. In order for a robot to operate autonomously, one of the most crucial capabilities it must have is the localization ability. An autonomous robot having the knowledge of its location can decide what to do in the next step; perform a given mission if it has communication with a master, explore its environment and avoid obstacles. The robot localization problem is so important that it has been mentioned to be the most fundamental problem to provide robots truly autonomous capabilities by some authors [2]. Accurate localization is especially crucial for navigation and map building tasks, since following a path and gathering relative positions of the landmarks are highly dependent on the location of the robot. In this thesis, the location of the robot is defined by its x, z and heading components in a world coordinate system. Although the degree of the accuracy depends on the specific operation, autonomy of the robot without a notion of

location is unthinkable. If the localization information does not exist, the robot is unable to plan further actions that are beyond its measurement range.

There are two main approaches for localization, namely incremental localization and global localization. The first one assumes that the initial position is known and the distributions are Gaussian. The robot position is then estimated using the measurements at each time step, while the robot is navigating in the environment. This type of localization is also called “position tracking”. The global positioning problem is a more challenging one because of the unknown initial pose. The localization has to be done from scratch. Although the latter approach does not require the assumption of a pre-known initial position and Gaussian distribution, it has high memory requirements. In both of these approaches the map of the environment needs to be known for the localization task.

One complication for the localization task is the disturbance of the robot position via a collision with an obstacle or even a harder situation is that the robot may be transferred to a new position unknowingly. Then the localization property plays an important role when the robot is relocated to an unknown position by an external force meaning that the robot is kidnapped. In this situation the robot has to realize that it is kidnapped and determine its new location. Another complication is the dynamics of the environment that the robot roams in. In general, the environments are assumed to be static, while the localization task is being performed. But this is not usually valid for the real world case. Localization in dynamic environments becomes a more complicated task, since the robot is not the only moving object. The moving objects may corrupt the localization information and cause the robot to get lost.

Localization can be classified into four main types each of which is described in the following subsections:

2.1.1 Dead Reckoning Localization

Dead reckoning is the most simple and cost-effective localization type that relies on estimating the position of the robot by integration of its motion estimates such as translation and rotation without making use of any external observations. In spite of its simplicity, dead reckoning is the most error prone localization type. Errors in the location accumulate proportional to the distance traveled and sensor inaccuracies, therefore the pose uncertainty increases at each time step. Because there is no chance to verify the new position, lower rate of uncertainty increase can be achieved by only with improvements on motion and sensor models. Still, at some time the uncertainty of robot location grows too much that it never provides useful information making this approach unsuitable for long-term localization [19, 20]. Odometry and INS (composed of accelerometers and gyroscopes) are the main sensors for the dead reckoning localization. This localization type is also referred as relative (local) localization, since the position and orientation evaluations are made by only on-board sensors.

2.1.2 Beacon (Artificial Landmark) Based Localization

Beacon based localization uses the specified beacons to determine the location of the robot. These landmarks are generally uniquely identifiable, therefore they can reliably be used for robot position determination and unambiguous global localization becomes possible [21]. Knowing the positions of these artificial landmarks, the robot is able to localize itself for long time periods accurately which makes this approach suitable for long-term localization. Since the localization relies on observing the positions of the known landmarks rather than observing the motion, the accumulation of errors does not exist and the accuracy does not deteriorate [22]. Some examples of these intentionally placed landmarks are GPS satellites, reflecting tapes, visual patterns, acoustic beacons and infrared beacons. These landmarks may be passive (reflecting) such as acoustic beacons and active (emitting) such as GPS satellites and infrared beacons. In spite of the high accuracy of the beacon based localization, there are some problems including installation

cost, situation of getting damaged and obscurity. The installation of some beacons may be too expensive or not suitable for some kind of environments. Moreover, the artificial landmarks may be damaged or occluded by other objects causing them not to provide localization information.

2.1.3 A Priori Map (with Natural Landmarks) Based Localization

This type of localization requires a previously known map which means that the environment related with the map has to be explored beforehand. Similar to the beacon based localization, a priori map based localization provides long term localization and the uncertainty of the robot location remains bounded. Since the a priori map consists of natural landmarks of the environment, usage of it clears the installation cost away which is present in the beacon based localization. These natural landmarks include corners, edges, walls, doors etc. present in the environment. Another advantage of this method is the robustness of these natural landmarks to damage. On the other hand, difficulties of the a priori map construction for each new environment, static nature of the map that cannot handle the dynamics of the environment and the fragility of the landmarks under variations of viewing direction and lighting conditions are regarded as the disadvantages.

2.1.4 Simultaneous Localization and Map Building (SLAM)

SLAM is the main topic of this thesis and thoroughly examined in the section 2.3.

There are several methods to solve the localization problem in the literature some of which are briefly presented:

2.1.5 Kalman Filter Localization

In this thesis this type of localization is used. The information about Kalman Filter is given in chapter 4 and Kalman Filter Localization is mentioned in chapter 6.

2.1.6 Markov Localization

The main idea of Markov localization is to compute a discrete approximation of a probability distribution over all possible poses in the environment and use Bayes rule to update the belief when the robot moves or senses [4, 6]. In Markov localization, each possible robot state is represented by a probability value resulting in many states. High amount of probability values require large memory and high processing time. Localization can be possible starting from any unknown position. But a discrete representation of the space using grids is required to update the probability of all positions in the state space. Memory and time requirements increase with the number of grids that will be used. This grid-based localization method is more robust than Kalman Filter localization, while the latter is more efficient and accurate [3]. The Markov localization is used for global positioning with high robustness on sensor noise and fast recovering from manual robot displacement [5]. In this type of localization "multi-modal probability densities are allowed and propagated through the motion model" [7].

2.1.7 Monte Carlo Localization (MCL)

This type of localization is a sample-based Bayesian method and can be viewed as a Markov method with random sampling. In this method particle filters are used which represent the probability distribution as a set of discrete particles which occupy the state space. Each particle stands for a possible robot location with a probability value. Initially, a large number of hypothetical configurations are randomly scattered. In a particle filter update cycle; [18]

- New particle distribution is generated given the motion model and controls applied,
- For each particle, an importance weight is assigned by comparing the particle's prediction of measurements with actual measurements,
- Particles are resampled based on their weights.

When the probability of a particle becomes very low, a new random particle takes place. At first, the robot does not know its position, so the particles are evenly distributed over the potential locations. In the next time steps, the samples (particles) next to the actual position become more likely. Only a subset of total states (poses), which is probabilistically chosen, is tracked and analyzed. A particle filter has an advantage of ability to represent multi-modal distributions. However, the number of particles needed to represent a posterior grows exponentially with the dimensionality of the state space, that is n particles for $1-d$, n^2 particles for $2-d$ and so on. Each particle indicates a robot pose and feature measurements are correlated with the robot poses. But, if the robot pose is known, features will be uncorrelated. Even though the Monte Carlo Localization is less robust than Markov Localization, its computational cost is considerably lower.

2.2 Map Building

Building and maintaining a map of the environment are other vital requirements for autonomous robotics in addition to localization. The map of the environment can be defined as a set of objects with defined positions and attributes needed to help navigation and localization of the robot. These objects have to be distinct, so that they can easily be recognized. Even though there are some other objects in the environment they don't contribute to the map, unless they have salient characteristics [25]. These map features are sensed via the sensors of the robot and navigation is done accordingly. The relationships between the map features are generally geometrical, but other characteristics such as shape and color can also be incorporated to the map by making use of suitable sensors [26]. A map can also be composed of equal or variable size of cells instead of features. In this case, the navigation task can be achieved by considering the probabilities of the occupancies. Robots use the maps to plan their actions and act accordingly in order to achieve the necessary tasks such as obstacle avoidance and reaching a destination. The maps are constructed by the measurements obtained from the sensors and the current location information of the robot, so that a world model is generated in this way. During the

map building process the changes in the environment, accessible and inaccessible regions are also identified. In order to attain a consistent map, reliability of the localization has to be high.

In map building task, there are some criteria that influence the quality of the map. First of all, the uncertainty needs to be modeled accurately in order to reflect the error between predicted and the actual system states. The map convergence can be achieved by well handling of the uncertainty, so that as new measurements are made the estimated map converges to the real map. Data association comes as another important factor for mapping. The correspondence between the map and the measurements obtained via sensors must be reliable. Here, real time efficiency of the mapping algorithm and robustness to variability of several aspects such as distance and viewing angle play an important role. For loop closing cases, the loop detection is the most significant feature of the mapping job that if the loop detection fails the map diverges and loses its functionality. One of the requirements for successful loop detection is the data association in a more global manner. Another important issue is the accumulated error handling at the loop detection time. The error must be handled so carefully that the map does not diverge in the next cycles. Map capacity and the computational complexity are the other criteria. The map must contain minimum amount of information sufficient to allow proper navigation tasks. In this way, the computational cost for the map management also stays minimum improving real time performance [20].

The maps can be classified with respect to their reference as relative maps and absolute maps.

In relative maps the relationships between landmarks are maintained. Considering geometric maps for instance, the relative relationship between two landmarks is the displacement between them. The relative map of an environment with n landmarks can be expressed with minimum $n-1$ and maximum $n(n-1)/2$ relations. Actually, the minimum number is sufficient for the map expression and the relations not specified explicitly can be derived [27].

In absolute maps the locations of the landmarks with respect to a defined reference frame are stored. While using absolute maps, an absolute state vector can be formed via combining the robot state and landmark locations expressed in the same coordinate frame [27].

The navigational maps can also be classified as geometric maps, topological maps and hybrid maps of former types:

2.2.1 Geometric Maps

These types of maps store the metric information about the relationships and positions of the objects existing in the environment [28]. When a fine grained map and precise motion control are desired the spatial information is required to be in geometric form, so that an exact navigation control can be achieved. The sensors measure the geometric attributes of the environment generally, so these maps are the natural outputs of the map building task. On the other hand, geometric maps are expensive to maintain, especially for large environments. The real time operation limits are exceeded, when the robot goes too much from the origin or the granularity of the map is asked to be higher. Moreover, a geometrical map of the environment with dynamic nature is difficult to maintain, since it probably will change with time [20].

There are two types of geometric maps, namely occupancy maps and feature (landmark) maps:

2.2.1.1 Occupancy (Evidence) Maps

Occupancy maps, also called dense maps, evidence maps or certainty maps, represent the environment as an array of rectangular grids associated with occupancy belief of corresponding cell. In general, this belief is probabilistic having a value in the interval [0, 1] and updated for each observation according to the Bayes Rule. These updates are tightly related with the sensor models being used. In graphical representation of occupancy maps, the intensity of each cell shows the

probability of being occupied. While the dark cells stand for the obstacles such as walls, light ones indicate the free area. The unobserved regions are specified with gray tone [28]. Using evidence grids can be an appropriate solution for relatively small environments. The occupancy and free space information facilitate the robot path planning. Furthermore, they can be used for representing the unstructured regions. Another advantage is that there is no requirement for a feature extractor [30]. As the size of the environment increases, the computational requirements rise too, but the cost can be kept reasonable by increasing the cell dimensions at the expense of accuracy. The navigational tasks also become computationally expensive in the case of high granularity. Some map building methods that apply variable granularity have been proposed in order to attain the optimal solution [31]. For SLAM applications, the occupancy grids devoid of an appropriate uncertainty model and diverge in long term localizations. Although the occupancy maps are suitable for either localization or map building, not for the SLAM task due to lack of integrated modeling of motion and sensor uncertainties and their correlations. The cells are considered to be independent and no relationship between them is maintained. Since the occupancy grids can not represent the uncertainty globally, loop closing task becomes challenging. There is also not any mechanism to recover from the errors at the end of the loop in occupancy mapping [20]. In order to achieve convergence for an occupancy map, many observations of each grid must be done. Occupancy grids are well suited to the range scan matching implementations, where high resolution sensors such as laser range finders are employed.

2.2.1.2 Feature (Landmark) Maps

These types of maps represent the environment as a set of distinct features with defined locations. Landmarks are simple parametric features such as points and lines. This kind of representation is more suitable for mapping the large regions than occupancy maps, since it conveys a sparse set of objects. Unlike the occupancy grids, the free areas are not represented and so no computation is required for them. The disadvantage of this situation is that some crucial tasks such as obstacle

avoidance and path-planning must be achieved via different mechanisms other than the map knowledge [20]. Although these maps are more useful for large environments, they are not well suited for small rooms as occupancy grids. Because these maps do not contain so much detail as the occupancy maps do. As a result, the estimation procedure used in map building may suffer. Another disadvantage is the requirement for a feature extraction method in order to observe the necessary landmarks. This type of mapping also requires a structured environment in some way, meaning that the environment contains appropriate objects to be sensed through the available sensors. The features of the map are sensed through the robot sensors and localization is accomplished by the association of the map features with the measurements. In this thesis, this representation of environment is used, employing the corner detection methods. With knowledge of these features and observations, the localization problem becomes an estimation problem. The incorrect data associations result in inconsistent maps and in order to mitigate this problem, batch associations where a number of measurements are used at the same time can be used.

In landmark mapping, the feature initialization can be a problem depending on the sensors used. If the available sensor provides sufficient information about the observed landmark, initialization process can be done easily. But for some sensors, especially infrared, sonar sensors and monocular cameras, single observation is not enough and several data must be gathered from several robot locations.

2.2.2 Topological Maps

Another way for map representation is encoding the topology of the environment instead of the geometry. The topology of the environment simply holds the structural information and the relationships of the places existing in the environment. Construction of these maps does not depend on metric information and is achieved by using the places and the paths connecting them. They have a graph structure, where nodes correspond to distinctive places and edges define distinctive paths between these places [20, 28, and 32]. The robot must have

capabilities such as localizing relative to the nodes and traveling between the nodes in order to navigate in the environment robustly. The most important advantage of topological mapping is the capability of path planning and navigating according to a topological map using the straight forward graph search methods. In these maps, the navigation from one node to another one is achieved by a sequence of node transitions. Since no metric information is employed, the uncertainty estimation of the robot pose becomes unnecessary. There are some disadvantages of using these maps. One of them is the reliability issue especially for the complex environments. Although this type of mapping is sufficient for static and simple environments, pure topological data without any metric information may fail in complex and dynamic environments. Actually, the most critical problem is the misrecognition of places. In this case, topological structure in the map is broken and the map can give no more useful localization information. This possibility may generally come true for environments with similar places.

2.2.3 Hybrid Maps

This type of mapping makes use of both geometric and topological maps in order to combine their advantages and eliminate their limitations. The complementary nature of these maps makes them appropriate to use together. For example, geometric maps have much power on representing local relationships more accurately with appropriate uncertainty estimation. They are also crucial for optimal path computations. On the other hand, the topological maps are composed of locally connected regions with reduced representations and can get global information without a global reference frame [20, 28]. Furthermore, they present more understandable information for humans and symbolic systems [33]. While metric maps are suitable for relatively small areas due to computational reasons, the topological maps only permit for coarse localization and so suboptimal path planning. Because of these reasons, the hybrid maps have the accuracy of metric maps and the scalability of the topological maps. Although geometric maps are used in this thesis, these maps can be incorporated to hybrid maps. The hybrid maps can

differ in terms of heterogeneity, hierarchy and separability, but the most common ones are geometrical maps on topological nodes and local submaps [28, 33].

2.3 Simultaneous Localization and Map Building (SLAM)

Simultaneous Localization and Map Building (SLAM), also called Concurrent Mapping and Localization (CML), is the concurrent execution of robot localization and map building tasks to achieve autonomous navigation goal. These tasks depend on each other, since map building without knowledge of position or localization without a known map is unthinkable. While sensor measurements are employed in map estimation, the robot motion also influences the map construction since measurements are relative to the robot location. Therefore there is a chicken and egg relationship between them. At one step of this cycle the robot observes its environment via its sensors and determines landmark locations making use of its current position estimate, whereas at the other step it improves its position estimate by reobserving the existing landmarks. As this cycle goes on and the landmarks are measured repeatedly, the uncertainties of landmark positions decrease and present more reliable localization information. Since the map of environment is constructed incrementally, a priori map is not needed eliminating the disadvantages of beacon based and a priori map based localization. Even so, the map can be built based on some initial information. Furthermore, the dynamic nature of the algorithm allows the robot to adapt to the environmental changes.

SLAM algorithm provides the robot with fully autonomous navigation capability allowing long term operations in unknown environments. By this capability the robot can be left to wander in an unmapped place and explore the area without any external aid, although some kind of intervention may still be required in the case of physical handicaps that robot cannot handle with its available sensors such as stair wells, hollows or transparent objects. Moreover having the necessary information about the environment and the self-location, the robot can also perform other high level tasks.

There are some technical difficulties related to the SLAM problem including uncertainty, complexity, linearization, feature extraction and data association [23, 24]. For SLAM solutions, stochastic methods are suggested generally in order to handle the uncertainty in the motion and measurement models. The uncertainties in the models need to be bounded in order for the robot to be able to localize itself accurately. Computational complexity is especially a crucial problem for real time systems. The cost increases with the size of the map making it limited to a maximum value to allow the real time operation. Linearization may be another problem for the systems in which this assumption is made. In Extended Kalman Filtering, the system is assumed to be linear ignoring the relatively small nonlinear components. As the map grows too much, this assumption may begin to fail and the map may become inconsistent. Feature extraction is a task to be handled for feature-based methods and has the meaning of obtaining the distinct entities that are easily recognizable in the environment. The feature representation is generally decided by considering the environment and the sensors used. In order the obtained features to be useful they must be salient and invariant to some extent in terms of viewing angle and scale. Another difficulty about SLAM is the data association problem. It is the correspondence between the features of the map and the measurements. The computed correspondence is used to improve the estimates. If the data association is done incorrectly, the estimation process will diverge.

There are generally two types of SLAM, namely Full SLAM and Online SLAM. Both types are probabilistic approaches to the SLAM problem. In Full SLAM [35] the entire path and the map is estimated. To say more obviously, this type requires that a state vector containing all the states in the robot model and all landmark states have to be maintained and updated after each measurement. In the following figures, X is the system state, U is the control input, Z is the sensor input and m stands for map of the environment with all subscripts denoting the time steps.

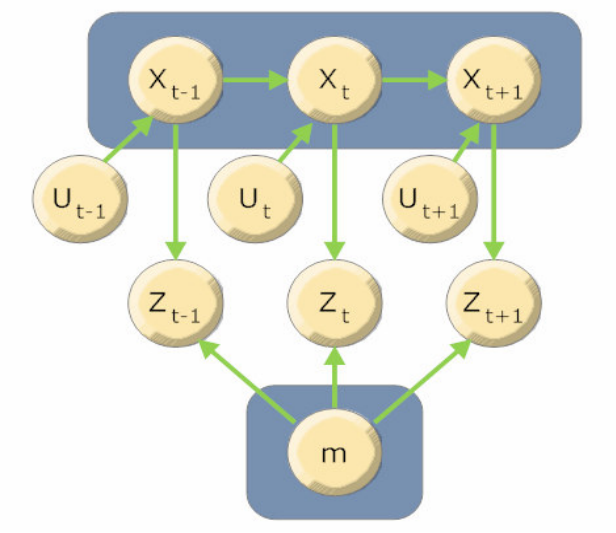

Figure 2-1 Full SLAM [43]

Fast SLAM, which uses a sampled particle filter distribution model, is an example solution for Full SLAM problem.

Online SLAM estimates the most recent robot state with map features and is expressed with the following equation [43]:

$$\mathrm{p}(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int \mathrm{K} \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1 dx_2 \ldots dx_{t-1} \tag{2-1}$$

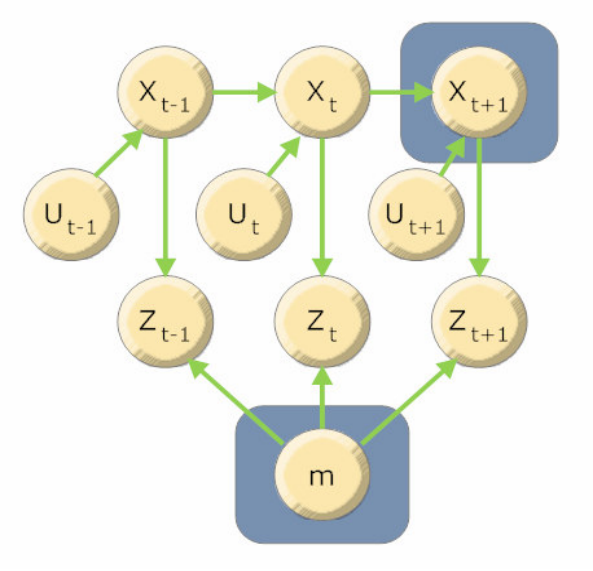

Figure 2-2 Online SLAM [43]

Extended Kalman Filter SLAM is an example solution for Online SLAM problem [34]. A linearized Gaussian posterior over the positions of environmental features and the robot position are calculated in this solution. This method is used in this thesis in order to solve the SLAM problem.

2.4 Sensors Used For SLAM

There are various sensors that can be used in SLAM applications including odometry, laser ranging and detection (LIDAR), acoustic sensors, radar, inertial navigation sensors (gyroscopes, accelerometers), GPS, visual sensors (monocular, omnidirectional, stereo cameras) and even sun sensors. These sensors should be chosen by considering the application characteristics. Noise, dimensionality of the output, range, the frame of reference, sample rate, robustness, cost, accuracy and operational conditions are some characteristics to consider during sensor selection.

A group of these sensors are called external sensors, since they provide localization information with respect to external environment [25]. Global Positioning System

(GPS), laser scanners, ultra-sonic scanners, acoustic sensors, radar, vision-based sensors, sun sensors and compass can be regarded as in this class of sensors. The sensors that do not give information referenced to the external world are internal sensors. Inertial navigation systems, gyroscopes, steering sensors and odometers are these types of sensors. Their measurements generally provide information about position variation rates. The information obtained from the internal sensors is utilized with the help of a vehicle model, so that the robot pose can be estimated. The measurement errors are then integrated in time, due to this incremental nature of information gathering. For instance, odometry-based sensing which relies on vehicle dynamics such as wheel diameters, wheel speeds and axle length is exposed to accumulated errors. These errors are originated from unequal wheel diameters, wheel misalignment, wheel slippage and surface irregularities [36]. As a result, accurate long-term localization cannot be achieved. Though odometry based localization is erroneous, it is still the most basic localization type due to its simple operation. In spite of these limitations, internal sensors have some advantages one of which is that they work independently from the environmental features, allowing them to be employed in various vehicles. Another advantage is their high frequency sampling. The fact that they do not have to emit waves in order to sense and their energy efficiency are also good characteristics. These properties make their usage advantageous especially for slip-free regions.

Other sensors also have several advantages and disadvantages. For instance, sonar sensors are cheap and work fast, but their measurement accuracies are low. Laser range finders have high accuracy; however they are relatively slow devices. With the advent of sun sensors, they are able to provide absolute heading updates in order to correct the heading information obtained from gyros [37].

Vision-based sensing has several advantages. Visual sensors provide high resolution data allowing high level tasks such as feature extraction, object recognition and 3D reconstruction. By extracting features with their depths, the robot can build a map of the environment and localize itself. In the meantime, it can process the visual data for other missions. Vision-based techniques can provide good state estimates

for SLAM problem even in slippery terrains. On the other hand; there are some drawbacks of these sensors. Variations of illumination, viewing angles and scales, as well as occlusions deteriorate the data association performances of the methods utilizing these sensors. Moreover, they have high power requirements and limitations on the speed of maneuvers [37]. Even so, the advantages of them outweigh the disadvantages and their limitations can be compensated to some extent with appropriate ways. Because of these reasons, visual sensors are used in this thesis in order to solve the SLAM problem.

For the sake of robustness, different techniques utilizing different sensors are tried to be fused in order to obtain more accurate systems and compensate for each other's limitations. Especially a long-term and long-distance navigation entails both a low rate of error growth and robustness. A very successful example for a system with combination of several sensors is NASA's MER mission rovers Opportunity and Spirit. Odometry and gyro data were fused and used with an Extended Kalman Filter for state estimation. A sun sensor was used to correct the heading information. In the case of slippery areas, a vision based approach was utilized for compensating odometry error [37]. In this thesis, this kind of approach is also used for autonomous SLAM. While the robot moves through the unknown environment and observe its surroundings via visual sensors, it also detects obstacles next to it by the help of it sonar sensors. In this way, an autonomous navigation capability for the robot system becomes possible.

2.5 Visual SLAM

Visual SLAM is the process of constructing a map of the environment and staying localized by mainly using vision-based information. The vision-based measurements are effectively utilized, so that necessary distinct features are extracted with their depth information and data association is done in order to correct the state estimations. The map of the environment is built incrementally by adding the obtained landmarks to the current map, while the robot determines its pose relative to this changing map. In Visual SLAM operation, different number of

cameras can be used. The two main approaches are those using a single camera, called Mono SLAM methods and those using two cameras, called Stereo SLAM methods. More than two cameras can also be used in order to make more certain observations, but in most cases this approach is redundant. In this thesis, Mono SLAM and Stereo SLAM approaches are investigated, implemented and presented by comparing their different properties.

CHAPTER 3

STEREO VISION

In this chapter, some stereo vision concepts will be presented which are used for the thesis. The first subject to touch is the camera calibration process. In order to effectively use the visual information obtained from the cameras, their internal and external parameters must be known and this can be achieved via a calibration procedure. Second, several camera models will be explained including the basic pinhole camera model and finite projective model. Finally, the techniques to determine the distances of the objects, especially the depth extraction for point features will be examined.

3.1 Camera Calibration

Camera calibration is the process of determining the internal and external parameters of the camera. Internal parameters are the quantities internal to the camera that play important role in imaging such as focal length, image center (principal point), skew factor, scaling factor and lens distortion. They represent the relationships between the pixel coordinates and the camera coordinates. External parameters, on the other hand, are the parameters that define the relative position of the camera, location and orientation of the camera, in the 3D world coordinate system. Camera calibration is necessary for the mapping of the camera pixel coordinates to rays in the scene, and also the points in the scene to camera pixel

coordinates. In this way, 3D quantitative measures of the objects existing in the observed scene can be recovered from the 2D images such as depth and height of an object.

In this thesis, Caltech Camera Calibration Toolbox for Matlab is used for camera calibration which is easy to use and works on various Matlab versions and different operating systems [38]. For the calibration procedure, a chessboard like pattern has to be photographed from a number of different viewing angles as shown in Figure 3-1.

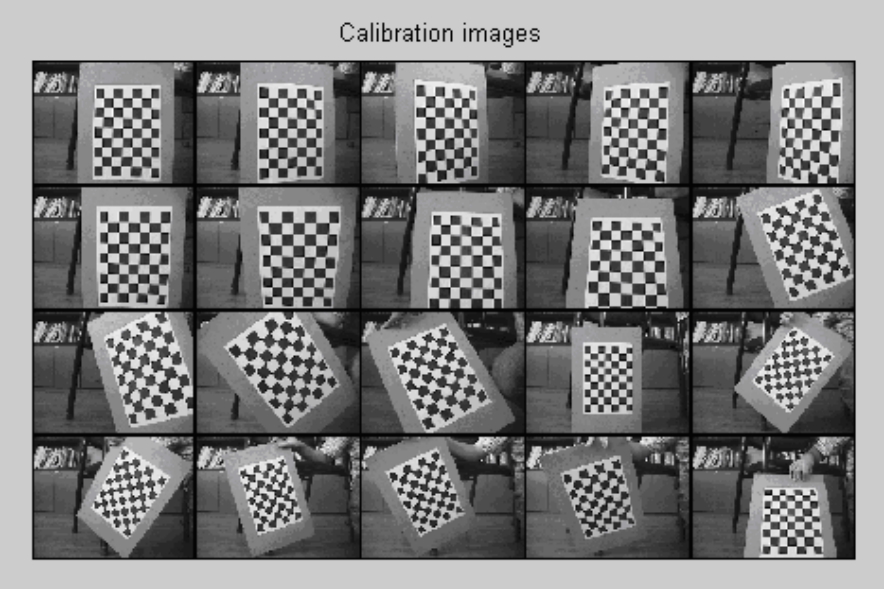

Figure 3-1 Calibration Images

After loading all photographs, the outermost corners of each pattern must be marked manually, the tool then automatically determines the remaining corners.

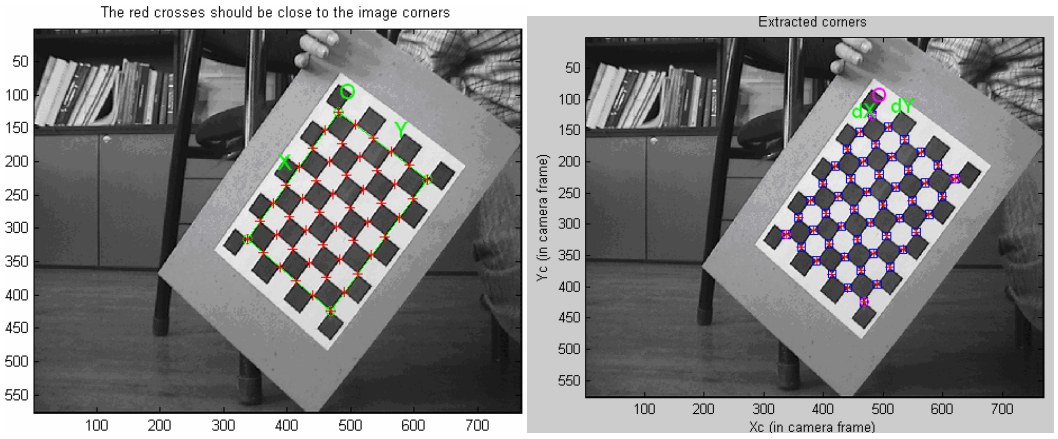

Figure 3-2 Automatically found corners

After corner extraction is done, the calibration process is started and the calibration parameters are stored in the specified variables with some uncertainties. The specified parameters are focal length, principal point, skew coefficient and the distortion coefficients. These parameters are determined by the toolbox via the true and the projected positions of the grid points. The focal length is modeled in pixels for both x and y directions independently. Similarly, the image center is determined in pixels. The skew coefficients which stand for the deviation of the coordinates of the camera pixels from perfect orthogonality are calculated. Their values are zero, when perfect orthogonality exists as in the calibration procedure made in this thesis. Lastly, the lens distortion coefficients are determined which represent the radial and tangential components of the distortion. After the calibration has been done, the reprojection errors of the grid points and the extrinsic parameters can be seen graphically, in addition to the textual representation of the intrinsic camera parameters.

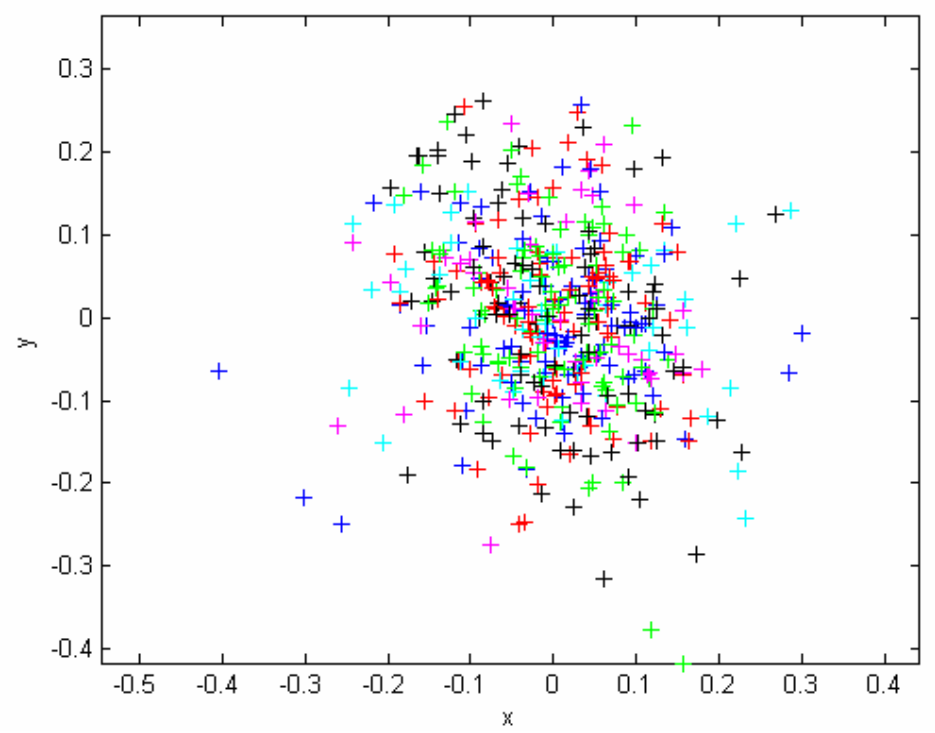

Figure 3-3 Reprojection Error (in pixel)

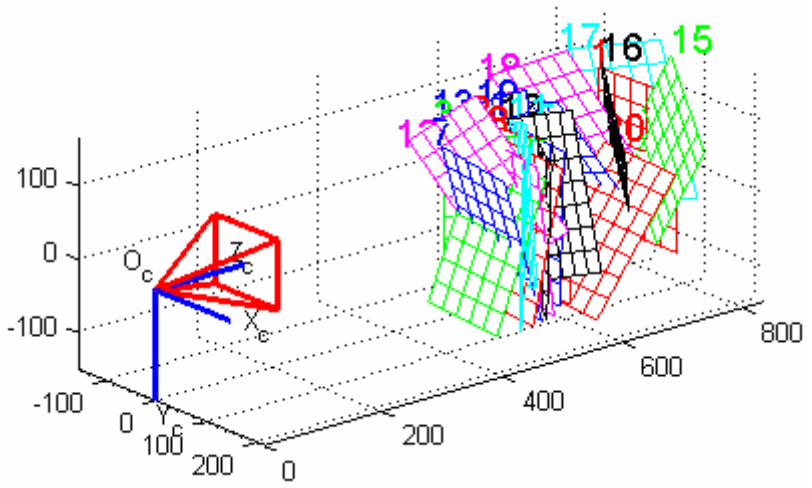

Figure 3-4 Extrinsic Parameters

After stereo calibration has been done, the extrinsic parameters of the stereo camera system are determined and can be viewed graphically as indicated in Figure 3-5.

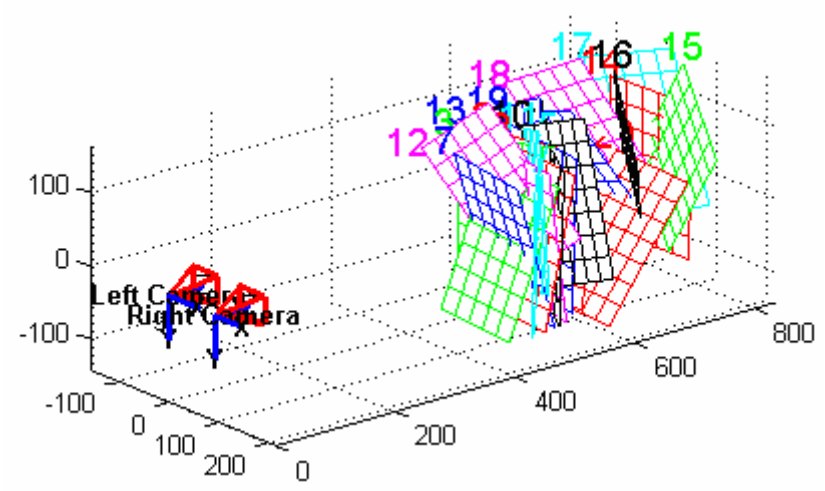

Figure 3-5 Extrinsic Parameters of the Stereo Camera System

Table 3-1 Intrinsic Camera Parameters

Intrinsic Parameters		
	Right Camera	**Left Camera**
$\mathbf{f_u}$ (pixels)	901.47	904.69
$\mathbf{f_v}$ (pixels)	907.71	911.04
$\mathbf{u_0}$ (pixels)	359.91	368.308
$\mathbf{v_0}$ (pixels)	302.26	307.804
s (degrees)	0	0
$\mathbf{d_0}$	-0.29042	-0.27717
$\mathbf{d_1}$	0.42845	0.36902
$\mathbf{d_2}$	-0.00054	0.00045
$\mathbf{d_3}$	-0.00143	-0.00121

After the calibration has been done, the parameters are found as in **Hata! Başvuru kaynağı bulunamadı.** where f_u and f_v are focal length parameters; u_0 and v_0 are image center parameters; s is skew parameter; d_0, d_1, d_2 and d_3 are the image distortion coefficients.

Table 3-2 Extrinsic Camera Parameters

Extrinsic Parameters		
	Right Camera	**Left Camera**
α (radians)	0	-0.00931
β (radians)	0	-0.02528
γ (radians)	0	-0.0037
$\mathbf{t_x}$ (mm)	0	-94.48839
$\mathbf{t_y}$ (mm)	0	0.20305
$\mathbf{t_z}$ (mm)	0	-0.28431

The extrinsic parameters of our stereo camera system are shown in After the calibration has been done, the parameters are found as in **Hata! Başvuru kaynağı bulunamadı.** where f_u and f_v are focal length parameters; u_0 and v_0 are image center parameters; s is skew parameter; d_0, d_1, d_2 and d_3 are the image distortion coefficients.

Table 3-2, where α, β and γ are the rotational parameters; t_x, t_y and t_z are the translational parameters. Since the right camera is chosen as reference, the corresponding parameters have zero values.

3.2 Camera Models

A camera model is a theoretical model which represents the transformation of scene points into an image, a mapping from point space R^3 to image space R^2. Various camera models exist which describe different characteristics of the cameras. While some of these models rely on physical camera parameters, called explicit camera models, some of them represent only a projection of the scene points into the image, called implicit camera models [39]. In this section, the pinhole camera model and the finite projective model will be examined.

3.2.1 The Pinhole Camera Model

The pinhole model indicates the mathematical relationship between the coordinates of a 3D point and its projection on the image plane of the camera. Only rotation and translation of the camera followed by a perspective projection is represented. Other cases such as geometric distortions and blurring of unfocussed objects are not taken into account.

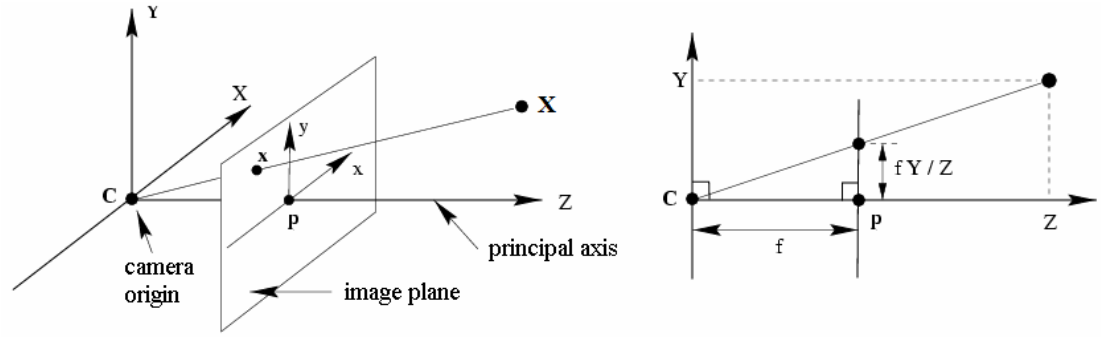

Figure 3-6 The Pinhole Camera Model

The model can be well understood by the help of Figure 3-6. If the camera center is taken as the origin of the Euclidean coordinate system, there is an image plane at

the focal length of the z-axis. Then the projection of a scene point X on the image plane is the point where the line between the camera origin and the point X intersects the image plane. The image point coordinates can be easily determined as (fX / Z , fY / Z) from similar triangles. The line originated from the camera center and perpendicular to the imaging plane is the principal axis. The point at which the principal axis intersects the image plane is the principal point.

The projection of a 3D point can be expressed as follows:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \alpha \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3-1}$$

3.2.2 Finite Projective Camera Model

In this model, the mapping between scene points and the image space is constructed by a projection matrix P as follows [36]:

$$x = PX \tag{3-2}$$

where x = $(x, y, 1)^T$ and X = $(X, Y, Z, 1)^T$. The point X is in a 3D homogeneous coordinate system and x is in a 2D homogeneous coordinate system in the image plane. P is a 3×4 matrix and expressed by:

$$P = K[R \mid t] \tag{3-3}$$

where K is a 3 x 4 *calibration matrix* (internal calibration of the camera), R is a 3×3 *rotation matrix*, t is a 3×1 *translation matrix* and they represent an inverse

motion of the camera in the world coordinate system. R and t matrices are called the external calibration of the camera, since they are dependent on the world coordinate system [40]. By concatenating the translation vector to the end of the rotation matrix, the transformation matrix $[R|t]$ is formed. The calibration matrix which conveys the information of internal characteristics of the camera is expressed as follows:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3-4}$$

where x_0 and y_0 are the coordinates of the image center, s is the skew parameter; α_x and α_y are focal lengths in pixels. Even though the principal point is assumed to be at the center in pinhole model, this does not hold generally and for this reason the image center parameters are incorporated into the calibration matrix. The image coordinate system and the camera coordinate frame does not coincide and the relationship between them is shown in Figure 3-7.

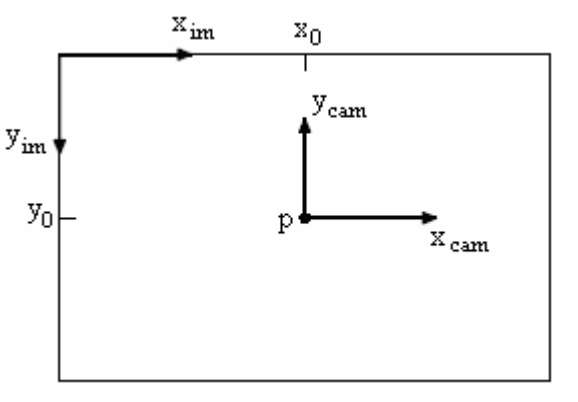

Figure 3-7 Relationship between image and camera coordinate systems [36]

The image coordinates are expressed in pixels and the principal point has the values that are half of the image width and height in the ideal case.

In some cases x and y-axis may not be perpendicular, then the skew parameter takes non-zero value in this cases in order to model the skewness and non-rectangular pixels. But most of the cameras are manufactured perfect enough today, so that the skew is generally assumed to be 0.

When we consider m_x and m_y as number of pixels per unit distance, the focal lengths can be expressed as $\alpha_x = fm_x$ and $\alpha_y = fm_y$. There are different focal lengths, because pixels can be rectangular. The ratio f_y / f_x is called the aspect ratio and it takes value of 1 in the case of square pixels.

There is a need for a relationship between the camera and world coordinate frames, since it is necessary to make conversions from one frame to another for various tasks. This relationship is expressed by the rotation matrix R and translation matrix t and illustrated in the Figure 3-8. These parameters depend on the world coordinate system, so they are called external parameters.

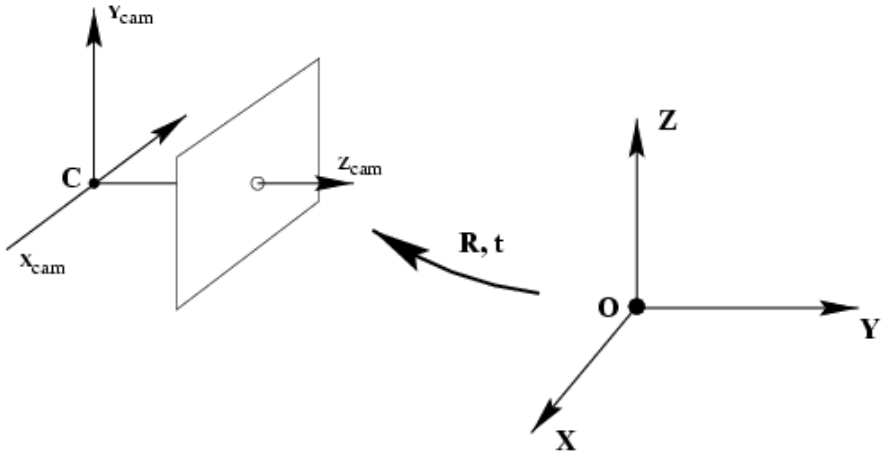

Figure 3-8 Transformation from world frame to camera frame

When the camera and world coordinate systems coincide, R becomes the identity matrix and t becomes the zero vector.

3.3 Depth Perception

Depth perception is the visual ability to perceive surroundings and get information about the 3D structure of the world. At least two images of an object is needed in order to perceive its depth. Therefore, multiple camera systems such as stereo camera pairs or multiple image shots of the same object taken by a single camera from different point of views are used in order to obtain the distance information. The multiple images provide the disparity information which is defined as the relative movement of an object in two or more views. In stereo systems, the cameras are separated by some amount and the disparity is related with the depth. The disparity is computed by corresponding points in the images and the depth information is extracted from the disparity. An alternative approach to using two cameras is to use a single camera and take images from different views. In both approaches when the lines of sight of the cameras are not parallel, perspective distortion must be considered while the point correspondence between images is being done. Since the images cannot be taken at the same time in the second approach, the depth sampling rates are lower.

3.3.1 Non-parallel Image Planes

In a non-planar stereo camera system, the cameras view the 3D objects from different positions and viewing angles and there are geometric relations between these 3D object points and their corresponding image points which put some constraints between the image pairs. The epipolar geometry is the geometry of stereo vision and explains the situation of intersecting image planes with the plane defined by image points and camera centers. It is actually the intrinsic projective geometry of the two views and do not rely on the scene but only intrinsic camera parameters and their relative poses. This geometry is usually used for image

correspondence search and matching for image pairs. In the Figure 3-9, the epipolar geometry is presented:

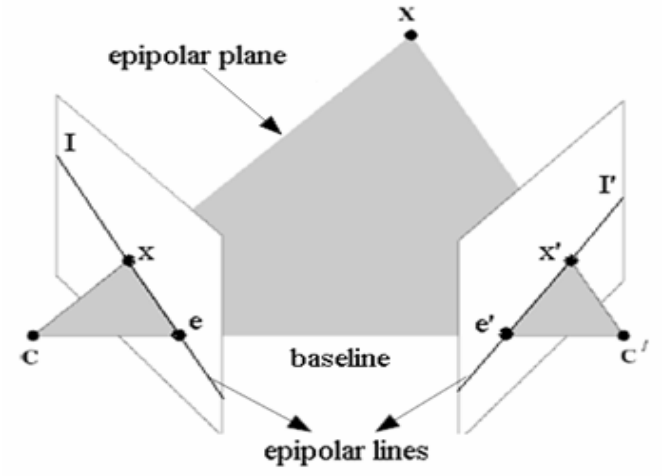

Figure 3-9 The Epipolar Geometry

There are some terms to be defined in epipolar geometry in order to explain the geometric relations. These terms are epipolar plane, epipolar line and epipole. In the Figure 3-9, X is the object point with its projection points x and x' on the left and right image planes, C and C' points are the camera centers. The epipolar plane is the plane defined by the points X , C and C' and contains the baseline. The epipole is the point at which the baseline intersects with the image plane. Moreover, it is the image of the camera center belonging to other camera. The epipolar line is the intersection of the epipolar plane with the image plane. The epipolar plane intersects the left and right image planes at the corresponding epipolar image planes and a correspondence is defined between the epipolar lines. The object point, corresponding image point and epipolar lines are all on the same plane, the epipolar plane, which means the corresponding image point of an image point on the epipolar line is on the other epipolar line. By utilizing this fact, the image search and matching tasks for stereo images become much simpler.

In order to recover the depth of a 3D object point X, it is needed to find the x and x' values. When these values are known, the depth can be found by back projecting the lines defined by the camera centers and the corresponding image points transforming the problem of depth determination into the problem of finding the corresponding image point of the known image point. For this purpose, the relation between the epipolar lines and the image points needs to be known. The fundamental matrix is the algebraic representation of the epipolar geometry and constructs this relationship. According to this relationship given an image pair, for each image point x in one image, there is a corresponding epipolar line I' in the second image. In this way, the fundamental matrix provides a projective mapping from points to lines. This mapping is indicated as follows:

$$I' = Fx \tag{3-5}$$

where x is the image point, F is the 3×3 fundamental matrix and I' is the corresponding epipolar line. The fundamental matrix F is a matrix of rank 2.

Since the image point x' lies on the epipolar line I',

$$x'^{T} I' = 0 \tag{3-6}$$

From the equations 3-5 and 3-6, the following relation can be obtained, which is called the epipolar constraint:

$$x'^{T} Fx = 0 \tag{3-7}$$

Since the fundamental matrix is a representation of the epipolar geometry and the epipolar geometry depends on the intrinsic parameters and relative poses of the cameras, this matrix contains this information also. After some calculations the fundamental matrix F is computed as follows:

$$F = K'^{-T} [t]_x R K^{-1} \tag{3-8}$$

where $[t]_x$ is the skew-symmetric matrix and indicated as follows for $t = (t_x, t_y, t_z)^T$.

$$[t]_x = \begin{bmatrix} 0 & -t_z & -t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{3-9}$$

3.3.2 Image Planes with Parallel Optical Axes

This is the simplest case for a stereo camera system, in which the two cameras are separated by a baseline distance b with their parallel optical axes. In this coplanar system, the image of an object occurs on different locations at the planes determined by the baseline distance and depth of the object.

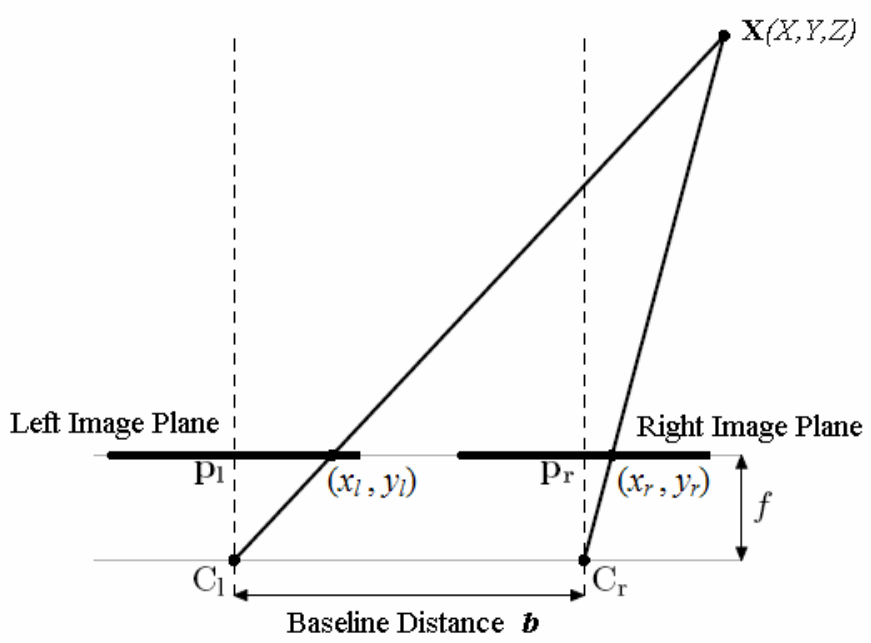

Figure 3-10 Coplanar Image Planes with Baseline Distance *b*

The following equations can be obtained by using the basic triangular similarity rules:

$$x_l = f\frac{X+b}{Z} \tag{3-10}$$

$$x_r = f\frac{X}{Z} \tag{3-11}$$

$$\frac{y_l}{f} = \frac{y_r}{f} = \frac{Y}{Z} \tag{3-12}$$

$$x_l - x_r = f\frac{b}{Z} = d \tag{3-13}$$

$$Z = f\frac{b}{d} \tag{3-14}$$

where (x_l, y_l) and (x_r, y_r) stand for the left and right image point coordinates. In this model, the displacement of corresponding points between two images is called disparity and indicated by d. The disparity is inversely proportional to the depth of the object, whereas it is proportional to the focal length and baseline distance. Therefore the knowledge of disparity, focal length and baseline distance is sufficient to determine the depth of the object.

Considering the epipolar geometry presented in the section 3.3.1, the resulting geometry for parallel image planes is viewed as follows:

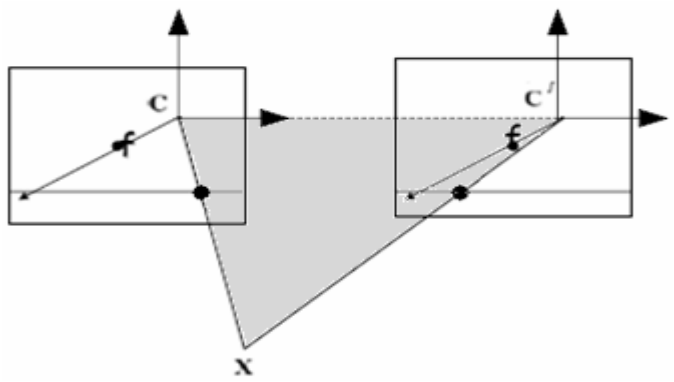

Figure 3-11 Epipolar Geometry for Parallel Image Planes

In this case, the epipolar lines are parallel to the baseline and the epipoles are at the infinity since the baseline and image planes intersect at the infinity. Moreover, epipolar lines become parallel also and they correspond to image rows. As a result, the point correspondance problem simply becomes searching the corresponding point on the other image with the same vertical coordinate of the image point.

3.3.3 Triangulation

In order to compute the depth of an object, the image points are back projected to lines which intersect at the depth of the object ideally. However, in real applications these lines do not intersect due to the noise present in the correspondence task. There are a number of suggestions for this problem one of which is the finding mid-point of the common perpendicular to the two rays, the mid-point of the line segment perpendicular to both lines from the closest points. Unfortunately, this procedure does not give the optimum results in most cases such as the case where the angles are not equal [41]. The method using the projection matrices presents more accurate results; therefore this procedure is used in this thesis.

We defined the P and P' matrices as the projection matrices of the corresponding images so that the images of an object point X are represented as $x = PX$ and $x' = P'X$. The cross products $x \times PX$ and $x' \times P'X$ are equal to zero and we get two linearly independent equations from each cross product. Furthermore, in order to get a unique solution, we put the constraint $\|X\| = 1$ into the problem. If we define the projection matrices as follows:

$$P = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}, \qquad P' = \begin{bmatrix} P_1' \\ P_2' \\ P_3' \end{bmatrix} \tag{3-15}$$

where P_i and P_i' are the 1×4 row vectors of the matrices P and P' respectively, the equations obtained from the cross products are given as:

$$x(P_3 X) - (P_1 X) = 0 \tag{3-16}$$

$$y(P_3 X) - (P_2 X) = 0 \tag{3-17}$$

$$x(P_2X) - y(P_1X) = 0 \tag{3-18}$$

$$x'(P_3'X) - (P_1'X) = 0 \tag{3-19}$$

$$y'(P_3'X) - (P_2'X) = 0 \tag{3-20}$$

$$x'(P_2'X) - y'(P_1'X) = 0 \tag{3-21}$$

By selecting the first two linearly independent equations of the each equation set, the equation $AX = 0$ is formed as follows:

$$AX = \begin{bmatrix} xP_3 - P_1 \\ yP_3 - P_2 \\ x'P_3' - P_1' \\ y'P_3' - P_2' \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = 0 \quad \text{subject to } \|X\| = 1 \tag{3-22}$$

where the matrix A is:

$$A = \begin{bmatrix} xP_3 - P_1 \\ yP_3 - P_2 \\ x'P_3' - P_1' \\ y'P_3' - P_2' \end{bmatrix} \tag{3-23}$$

As stated earlier, the back projected rays do not intersect usually and optimal solutions are tried to be found. An optimal solution can be attained using the Singular Value Decomposition of the matrix A, which is expressed as $A = UDV^T$

where U and V are orthogonal and D is a diagonal matrix. Solving this equation, the optimal solution is found as the last column of V in least square sense.

3.4 Landmark Detection and Matching

In order to achieve robot localization, the objects existing in the environment are measured somehow and their distance information is extracted from these measurements. These distance values are incorporated in the current map of the environment with unique labels. The distance information of the objects is used for determining the robot location in the environment. In order this scenario to be correct, the environment should contain distinct objects recognizable via vision-based methods, which are called as *feature* or *landmark*. In the following subsections the landmark type used in this thesis and its properties, detection methods and matching task are described.

3.4.1 Landmark Type and Characteristics

The map building and localization system presented in this thesis uses point features that exist in the environments such as corners of walls, doors or other objects. These features exist in the environment naturally, that's why they are called natural landmarks. These are discrete features and do not have continuous appearance, so that recognition of them can be achieved unambiguously. In order for a robot to have long-term localization ability, these landmarks should have invariant characteristics with respect to some criteria such as viewing angle, measuring distance etc. The most important elements of these characteristics are repeatedly detectability and measurability, reliability and longevity. First of all, the landmarks should be stationary in the environment. In this way a landmark can present reliable localization information and the robot can accurately update its position. Landmarks that have regular motion patterns in the environment may also present reliable

information, but derivation of this information is of course more challenging and only static landmarks are used for SLAM operation in this thesis. Another property that affects the reliability of the landmarks is that they should not be occluded frequently for long time periods. Otherwise, they cannot be seen and detected by the robot system although they have good characteristics for detection and matching task. Furthermore, the landmarks should be recognizable from a varying set of distances and viewing angles, since the robot moves in the environment and probably does not view the landmarks from the same distance and angle with the distance and angle that it first measures them.

In order to represent the landmarks, we use simple image patches with dimensions 15×15. The selection of this patch size is due to a tradeoff between computational complexity and probability of mismatch. When the size is smaller, the probability of mismatch increases. On the other hand, higher patch size slows down the matching task by increasing the computation time. In a robot system with more powerful processor and utilizing a steerable camera head, a larger image patch is more appropriate reducing the probability of mismatch. Since our system has a non-steerable camera system and moderate processing properties, the specified patch size is selected to give quite good results.

Apart from the patch size, there is another important factor that increases the computational cost of the SLAM operation: number of landmarks. In fact, this number dominates the computational complexity of the algorithm, when it exceeds a particular threshold, because the Kalman Filter estimation used in this thesis has covariance calculations which get more complex with increasing number of elements. Therefore the number of landmarks must have a reasonable value, in order to achieve a real-time operation. For this reason, a real-time efficiency algorithm will be presented in the section 6.3. Moreover, various experiments with different number of landmarks can be conducted with our tool SLAM Suite, since it presents a parameterization infrastructure for various SLAM characteristics. The experimental and analysis results of these observations will also be presented in the experiments section.

3.4.2 Detecting Landmarks

In order to make use of the salient features in the environment for localization and map building purposes, they must first be detected via a feature detection method. We use the Shi and Tomasi's feature detection method in this thesis, which is essentially a corner detection algorithm [42]. This method is similar to the Harris corner detector, but it is better in detecting salient features that can easily be distinguished from their surroundings. The image patches with high intensity variation can be identified using this detector.

The method introduced by Shi and Tomasi first calculates the horizontal and vertical gradients of the image intensity for each image pixel and sums up the values belonging to the current searching patch as follows:

$$Z = \sum_{P} \begin{bmatrix} g_x^{\,2} & g_x g_y \\ g_x g_y & g_y^{\,2} \end{bmatrix} \tag{3-24}$$

where g_x and g_y are horizontal and vertical image intensity gradients and P is the image patch. After the summation both eigenvalues of the matrix Z are calculated and used as selection criteria. When the smaller of them is greater than a predefined threshold, the patch is accepted to be corner. The case in which there is a single large value among the eigenvalues indicates that the patch contains a one directional variation such as an edge.

In order to find good features, this operator is applied all over the image and the patches with high smaller eigenvalues are chosen. When the patches are selected, some criteria are also considered one of which is the quality level. This criterion is used to determine the acceptable quality of the image patches to be selected. Another factor affecting the patch selection is the minimum possible distance between corners. When the Euclidean distance between two adjacent corners is below this value, the less strong corner is eliminated. To achieve real-time operation

this distance should be kept large, so that the vision system works on a sparse set of strong features and does less processing. All of these criteria can also be changed and tuned by our tool SLAM Suite. OpenCV library has a function called *cvGoodFeaturesToTrack* which implements this algorithm and this function is used as a feature detector in this thesis.

Of course all of the corners detected by this detector are not good features. The selected image patches may include some bad features such as light reflections and shadows. But the algorithm presented in the next section detects these bad features, when the robot cannot reobserve them for several times. When a bad feature is detected, the SLAM system gets rid of it and updates the map.

3.4.3 Landmark Searching and Matching

Landmarks are searched on the taken camera image for the purpose of adding new landmarks to the current map and detecting the existing landmarks to use them in system update. Searching for new landmarks needs to be done all over the image surely, but searching the existing features all over the image is a computationally complex burden. In fact, in a state estimation system this is unnecessary and every landmark has an uncertainty region to search for. The probability of detecting the pre-known feature in this area is quite high, so that only this region is searched. This subject will be discussed with technical details in section 6.1.5.

After finding the landmarks in the image, it is time for matching task. Matching is required for several reasons. One reason is to detect an existing landmark in the image. The features added to the map are associated with the image patches taken when they are first observed. The associated image patch is matched with the corners found in the search region. When a high correlation is detected, the feature is said to be reobserved. Another reason for matching is to determine the stereo image pair of the same landmark. When an existing or new landmark is detected on the right camera image, it also needs to be found in the left camera image by

matching process. In this way, the depth of the landmark in question can be extracted and fed to the system for correction.

In order to find corresponding pair during matching, several similarity criteria can be utilized using the matching methods such as sum of squared differences, cross correlation and correlation coefficient, also with their normalized forms. The normalized sum of squared differences is defined as follows:

$$R(x,y)=\sum_{x',y'}\frac{(T(x',y')-I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'}T(x',y')^2\sum_{x',y'}I(x+x',y+y')^2}} \tag{3-25}$$

where $T(x,y)$ is the image patch and $I(x,y)$ is the image which is being searched. This similarity checking method compares all of the image patches of the same size with image patch for each pixel on the main image by taking difference of image patch pairs. These differences are added up and the result is divided by a normalizing factor. For a perfect match, the result must be zero. However, this is not achievable for real cases and a specified threshold is used generally for decision. If the most matching patch has also a similarity values below the threshold, it is accepted to be matched. The OpenCV function that we use detects strong features on the image, so it is enough to apply the similarity checking for only those detected corners.

Another method used for similarity checking is cross correlation. The equation that belongs to the normalized cross correlation method is as follows:

$$R(x,y)=\sum_{x',y'}\frac{(T(x',y')I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'}T(x',y')^2\sum_{x',y'}I(x+x',y+y')^2}} \tag{3-26}$$

where $T(x,y)$ is the image patch and $I(x,y)$ is the image being searched. The template image is compared with the image blocks constructed around the corners

detected by the feature detection algorithm. These image blocks are constructed to be of the same dimensions with the template image. Ideally, the result must be 1 for a perfect match; however this cannot be attained in real life because of viewpoint variances, illumination changes and noise, so a pre-determined threshold should be employed to be checked for decision. The values below this threshold are then discarded.

The last similarity checking method utilized in this thesis is correlation coefficient. The normalized version of this algorithm has the following form:

$$R(x, y) = \sum_{x', y'} \frac{(T'(x', y')I'(x + x', y + y'))^2}{\sqrt{\sum_{x', y'} T'(x', y')^2 \sum_{x', y'} I'(x + x', y + y')^2}}, \tag{3-27}$$

where $T(x, y)$ is the image patch and $I(x, y)$ is the image being searched,

$$T'(x', y') = T(x', y') - \frac{1}{(w \bullet h)} \sum_{z'' y''} T(x'', y'') \tag{3-28}$$

and

$$I'(x + x', y + y') = I(x + x', y + y') - \frac{1}{(w \bullet h)} \sum_{z'' y''} I(x + x'', y + y'') \tag{3-29}$$

While searching a stereo pair of a landmark, it is enough to search it in the image regions proximate to the epipolar line. If the corner with highest similarity value also passes the threshold control, then it is selected and the depth information of that feature is extracted by triangulation.

CHAPTER 4

STATE ESTIMATION

Estimating the state of the robot and its environment is a fundamental problem. An efficient state estimator can compute the current state of the robot recursively based on the previous state. This computation can be done using Bayesian Filter. In this chapter, we will first introduce the basic Bayesian Filter. Then, we will present the Kalman Filter which is a form of Bayesian Filter. Finally, we will explain the extended version of the Kalman Filter for nonlinear systems, namely the Extended Kalman Filter.

4.1 Bayesian Filter [44]

Bayesian Filter recursively calculates the posterior distribution:

$$Bel(x_T) = P(x_T \mid Z_T) \tag{4-1}$$

Estimation of the robot state given the data is

$$Bel(x_t) = p(x_t \mid Z_T) \tag{4-2}$$

The robot's data Z_T includes observations o_i and actions a_i.

$$Bel(x_t) = p(x_t \mid o_t, a_{t-1}, o_{t-1}, a_{t-2}, ..., o_0) \tag{4-3}$$

Using Bayesian Theorem we get:

$$Bel(x_t) = \frac{p(o_t \mid x_t, a_{t-1}, ..., o_0)\, p(x_t \mid a_{t-1}, ..., o_0)}{p(o_t \mid a_{t-1}, ..., o_0)} \tag{4-4}$$

Since the denominator is constant with respect to x_t, we can assume it constant with value $1/\eta$. So the resulting equation is,

$$Bel(x_t) = \eta p(o_t \mid x_t, a_{t-1}, ..., o_0)\, p(x_t \mid a_{t-1}, ..., o_0) \tag{4-5}$$

First-order Markov assumption shortens first term:

$$Bel(x_t) = \eta p(o_t \mid x_t)\, p(x_t \mid a_{t-1}, ..., o_0) \tag{4-6}$$

Using the theorem of total probability, we expand the last term,

$$Bel(x_t) = \eta p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}, ..., o_0) p(x_{t-1} \mid a_{t-1}, ..., o_0) dx_{t-1} \tag{4-7}$$

First-order Markov assumption again shortens middle term:

$$Bel(x_t) = \eta p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}) p(x_{t-1} \mid a_{t-1}, ..., o_0) dx_{t-1} \tag{4-8}$$

Finally, substituting the definition of $Bel(x_t - 1)$ we obtain the probability distribution estimated from the robot's data:

$$Bel(x_t) = \eta p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1} \tag{4-9}$$

For Bayesian Filter iteration, motion model and sensor model can be considered as follows:

- **Propagation of motion model:** Current state estimate is computed before taking a sensor reading by integrating over all possible previous state estimates and applying the motion model.

$$Bel\_(x_t) = \int P(x_t \mid a_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1} \tag{4-10}$$

- **Sensor model update:** Current state estimate is computed by taking a sensor reading and multiplying by the current estimate based on the most recent motion history.

$$Bel(x_t) = \eta P(o_t \mid x_t) Bel\_(x_t) \tag{4-11}$$

There are some requirements to be realized for the implementation of the Bayesian Filter as follows:

- Representation for the belief function
- Update equations
- Motion model
- Sensor model
- Initial belief state

Representation of the belief function can be sample-based or parametric. Particle filter is an example for the sample-based representations. An example for a parameterized Bayesian Filter is the Kalman Filter.

4.2 Kalman Filter [44]

Kalman Filters represent posterior belief by a Gaussian (normal) distribution. Initial belief $Bel(x_0)$ is a Gaussian distribution.

State at time $t+1$ is a linear function of state at time t:

$$x_{t+1} = Fx_t + Bu_t + \varepsilon_{t(action)} \tag{4-12}$$

Observations are also linear:

$$z_t = Hx_t + \varepsilon_{t(observation)} \tag{4-13}$$

Error terms are zero-mean random variables which are normally distributed. Motion model and sensor model are Gaussian. Each belief function is uniquely characterized by its mean μ and covariance matrix Σ. Computing the posterior state implies to compute a new mean μ and covariance Σ from old data using actions and sensor readings.

The motion model of a linear dynamic system can be expressed as:

$$x_{t+1} = F_t x_t + B_t u_t + G_t w_t \tag{4-14}$$

where F_t is state transition function, B_t is control input function, G_t is noise input function with covariance Q, x_{t+1} and x_t are posterior and prior states respectively, u_t is control input and w_t is process noise.

The measurement equation of the system can be expressed as:

$$z_{t+1} = H_{t+1} x_{t+1} + n_{t+1} \tag{4-15}$$

where H_{t+1} is sensor function, z_{t+1} is sensor reading and n_{t+1} is sensor noise with covariance R.

Noise components for the above equations, namely G_t, w_t and n_{t+1}, are introduced to deal with the uncertainties that the deterministic system models cannot manage. These uncertainties are imperfections in the models, the environmental effect that are out of control and the noise in sensor measurements.

According to the Fundamental Theorem of Estimation, the state and covariance will be:

$$\begin{aligned} \hat{x}^{MMSE} &= E[x \mid Z_{t+1}] \\ P^{MMSE} &= E[(x - \hat{x})^2 \mid Z_{t+1}] \end{aligned} \tag{4-16}$$

The following notation will be used for next:

$$\begin{aligned} \hat{x}_{t+1|t+1} &= E[x_{t+1} \mid Z_{t+1}] \\ \hat{x}_{t|t} &= E[x_t \mid Z_t] \\ \hat{x}_{t+1|t} &= E[x_{t+1} \mid Z_t] \end{aligned} \tag{4-17}$$

Using the notation described in 4-17, we obtain the following equations for the state estimates:

- Predicted state:

$$\hat{x}_{t+1|t} = F_t \hat{x}_{t|t} + B_t u_t \tag{4-18}$$

- Predicted measurement:

$$\hat{z}_{t+1|t} = H_{t+1} \hat{x}_{t+1|t} \tag{4-19}$$

- Predicted state covariance:

$$P_{t+1|t} = F_t P_{t|t} F_t^{\ T} + G_t Q_t G_t^{\ T} \tag{4-20}$$

- Innovation covariance:

$$S_{t+1|t} = H_{t+1} P_{t+1|t} H_{t+1}^{T} + R_{t+1} \tag{4-21}$$

- Residual:

$$r_{t+1} = z_{t+1} - \hat{z}_{t+1/t} \tag{4-22}$$

- Kalman gain:

$$K_{t+1} = P_{t+1/t} H_{t+1}^{T} S_{t+1}^{-1} \tag{4-23}$$

- Corrected state estimate:

$$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1} \tag{4-24}$$

- Corrected state covariance:

$$P_{t+1/t+1} = P_{t+1/t} - K_{t+1} H_{t+1} P_{t+1/t} \tag{4-25}$$

4.3 Extended Kalman Filter [44]

It is one of the first probabilistic SLAM algorithms and solves the Online SLAM problem using linearized Kalman Filter. It is an extension of Kalman Filter to apply it to the non-linear systems.

In mobile robot applications the odometry estimate is rather treated as a sensor than a reflection of the robot's control system. Mobile robot dynamics are not linear as in the case of many other systems. A linear process model should be built out of the non-linear system dynamics to model such systems. Extended Kalman Filter is an extended version of Kalman Filter, where state and sensor models of the non-linear systems can be linearized at the current state estimate with the cost of state error residual increase since it is not the best estimate.

Non-linear dynamic model indicates the change of robot state with time:

$$x_{t+1} = f(x_t, u_t) + w_t \tag{4-26}$$

where f is a non-linear function and $w_t = N(0, \sigma_t)$. Non-linear measurement model predicts the measurement value given the robot state:

$$z_t = h(x_t) + v_t \tag{4-27}$$

where h is a non-linear function and $v_t = N(0, \sigma_t)$. Since Gaussian probability distribution functions are not preserved under non-linear transformations, resulting vectors for above equations are not Gaussian and state estimation cannot be done recursively. In order to linearize the systems, the non-linear functions f and h are expanded in Taylor series around the previous estimate $\hat{x}_{t|t}$. In this expansion, if the higher order terms are neglected, the resulting equations behave linearly and computational cost for solving these equations reduces. But these equations do not give the best estimates and they are suboptimal.

Based on the linearized model equations, Extended Kalman Filter is a suboptimal state estimator and its equations can be obtained similarly with linearization assumption above in two phases.

The equations for the prediction phase are given below:

- Predicted State

$$\hat{x}_{t+1|t} = f(\hat{x}_{t|t}) + u_t \tag{4-28}$$

- Predicted Measurement

$$\hat{z}_{t+1|t} = h(\hat{x}_{t|t}) \tag{4-29}$$

- Predicted State Covariance

$$P_{t+1|t} = \frac{\partial f}{\partial x} P_{t|t} \frac{\partial f}{\partial x}^T + Q_t \tag{4-30}$$

- Innovation Covariance

$$S_{t+1} = \frac{\partial h}{\partial x} P_{t|t} \frac{\partial h}{\partial x}^T + R_t \tag{4-31}$$

The equations for the correction phase are as follows:

- Residual

$$r_{t+1} = z_{t+1} - \hat{z}_{t+1/t} \tag{4-32}$$

- The filter gain

$$K_{t+1} = P_{t+1|t} \frac{\partial h}{\partial x} S_{t+1}^{-1} \tag{4-33}$$

- Corrected State

$$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1} \tag{4-34}$$

- Corrected State Covariance

$$P_{t+1/t+1} = P_{t+1/t} - K_{t+1} H_{t+1} P_{t+1/t} \tag{4-35}$$

Due to linearization for extension to Kalman Filter, some limitations are introduced such as non-zero mean of state prediction error and singular innovation state covariance matrix. As in Kalman Filter, process and measurement covariances specify the reliability of the filter and both should be small. Because at each time step Jacobian matrices are calculated, system becomes time-variant and this increases the computational cost compared to case where Kalman Filter is used for a time-invariant system.

There are some problems with EKF. First, it uses uni-modal Gaussians to model non-Gaussian probability density function. To cope with this problem, multiple EKFs can be used. Multi-Gaussian approach permits to represent arbitrary probability densities. In such multiple hypothesis tracking, consistent hypothesis are

tracked while inconsistent ones are dropped. In this manner, it is similar to particle filters except that the number of filters to track is much smaller. Second, only one set of measurement-feature associations is considered and maximum likelihood association is used. This reduces the recovery chance for inconsistent associations. Another problem with a Kalman Filter is that if the uncertainty of the robot becomes too large because of a collision or another reason, filter fails and the position is definitely lost.

CHAPTER 5

THE ROBOT SYSTEM

In this chapter, the hardware and software components of our robot system are described first. Then, the robot models used in this thesis are discussed.

5.1 System Composition

There are three main components that our robot system is composed of:

- The robot vehicle
- The camera system
- The processing unit

The camera system is mounted on the vehicle platform and they are the mobile components of the robot system. This mobile part is controlled by a stationary processing and coordination unit. In the following subsections, these components are described in detail.

The composition of our robot system is shown in Figure 5-1.

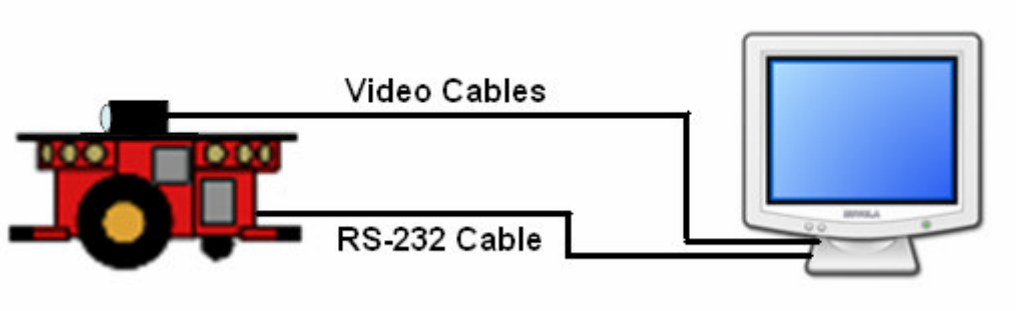

Figure 5-1 Robot System Composition

5.1.1 The Robot Vehicle

The robot vehicle used in our thesis is a Pioneer 2 DX8 model manufactured by ActiveMedia Robotics Company. It has a differential 3-wheeled robot base with two bigger wheels at the front and the third one is at the back. The two wheels at the front of the vehicle are able to move independently by driving two servo motors, whereas the wheel at the back of the vehicle is just a free-running one and used for stabilization purpose. The servo motors are controlled by a Hitachi microcontroller and controller board. The vehicle has 3 degrees of freedom for motion, which are translation in two axes and rotation around the axis perpendicular to the motion plane. The wheel revolutions of the vehicle are read by shaft encoders and these readings are used to specify the robot location with respect to its starting point. The midpoint between two wheels at the front is the center of the vehicle and the robot location is specified considering this point. However, to determine the robot location in this way, namely odometry, is error-prone; because the cases such as slippage and skidding cannot be handled. We will show the capability of our system for handling these cases in Experiment 3 of chapter 7, by comparing with the odometry readings.

At the front of the robot there is a sonar arc composed of several sonar sensors, which is used for proximity sensing. Although it can be used for depth perception also, it is not in this thesis; because its depth range is small and uncertainty of depth

measurement is high. The sonar arc of the robot vehicle is used for only proximity sensing and so for obstacle avoidance in this thesis, whereas the depth perception task is achieved by visual tools. The sonar arc of the vehicle is illustrated in Figure 5-2.

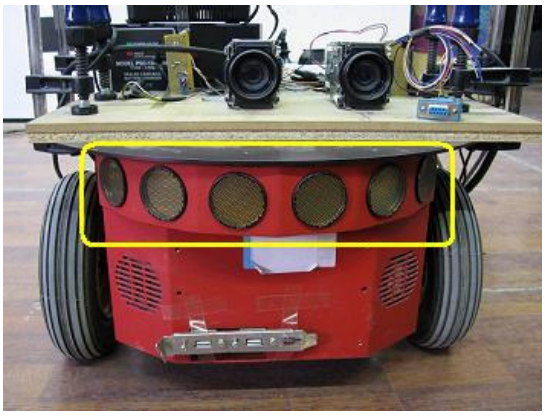

Figure 5-2 The Sonar Arc of the Robot Vehicle

The communication between the robot vehicle and the processing unit is established by a RS-232 serial cable, which has Rx, Tx and GND pins for signal transmission. The vehicle is supplied power via the rechargeable 12V batteries.

The vehicle is controlled by the software provided by the ActiveMedia Robotics, which is called *Aria*. The Aria library involves a complete set of functions in order to control the vehicle and make it perform the desired commands. Moreover, the library provides an infrastructure for reading vehicle current state characteristics such as odometry values, velocity and acceleration of the vehicle, battery voltage etc. We will utilize the odometry readings at most in chapter 7 in order to compare and show our experimental results.

5.1.2 The Camera System

There is a stereo camera system mounted on our robot vehicle, which was constructed previously for the theses [36 and 40]. The system has two identical Sony color board cameras, composite video cables and a board on which the cameras are mounted. The stereo camera system is depicted in Figure 5-3.

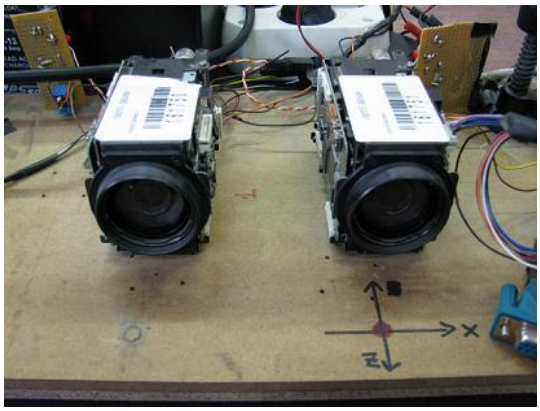

Figure 5-3 The Stereo Camera System

The cameras give analog interlaced outputs and they are transferred via composite video cables. Cameras are fixed on the board and their positions do not change significantly in time. So, the camera calibration parameters obtained can be used for long time periods. The cameras have resolution of 768×576 in PAL standard. They have also 18× optical zoom and 10× digital zoom capabilities. But they are used in wide angle mode in this thesis, in order to view wider scene and determine better features to track in this way. Moreover the opportunity of tracking a feature for longer periods is gained, since it exists in the field of view longer. Therefore, the camera calibration procedure is also done in wide angle mode. The grayscale outputs are taken from the cameras and used, since color is not needed for our system. The general characteristics of the cameras are given in Table 5-1.

Table 5-1 Technical Characteristics of the Cameras

	Right Camera	**Left Camera**
Model	Sony FCB-IX47AP	Sony FCB-IX47AP
Serial No	1001512	1001511
Resolution	768×576	768×576
Focal Length	3.1 – 31 mm	3.1 – 31 mm
Optical Zoom	18×	18×
Digital Zoom	10×	10×

The right camera is the reference camera in the stereo construction. Its center is the origin of the camera coordinate frame. The camera coordinate frame and the robot vehicle coordinate frames coincide, therefore the depth information gained by the camera system is also obtained with respect to the vehicle and there is no need for any transformation between these frames. The right camera is also selected as the active camera, when our SLAM system is operating in single camera mode.

5.1.3 The Processing Unit

The processing and coordinating unit of our robot system is a stationary PC host. It has a Pentium 4 - 2.4 GHz processor and 512 MB ram. These processing and memory specifications are sufficient for image processing tasks. PC has also a hard disk with enough storage capacity for the necessary programs and intermediate files, Matrox Meteor II frame grabber card and 64 MB GeForce 4 MX440 video graphics card. Although the video graphics card is not directly influential on image processing tasks, it is needed for visualizing our system via its graphical user interface.

As stated earlier, the cameras give analog outputs which are needed to be converted into digital form in order to perform image processing. The conversion process is achieved by the help of Matrox Meteor II frame grabber card. It samples the analog

output of the cameras and forms digital data. The card supports up to 12 video inputs, which is quite sufficient for our system. But, there is a time limitation for the stereo case, since the card processes the video inputs by multiplexing them one at a time.

The operating system of our PC is Microsoft Windows XP. This operating system with necessary hardware components can satisfy the time requirements of our SLAM system. The main reason for this selection is that all of the software tools which we use for developing our system are compatible with this operating system.

We have developed our system in Visual C++ .NET 2003 environment using the programming language C++. Aria and MIL (Matrox Imaging Library) C++ libraries are integrated into our code and used for controlling the robot vehicle and the frame grabber card respectively. Aria library is free but MIL is a commercial one. Luckily, there is a free version with limited features called MIL-Lite. This version allows only image and video acquisition. Therefore an open source and free library, namely Intel Open Source Computer Vision Library (OpenCV) is used for image processing tasks. The graphical user interface (GUI) of our system is developed using Windows Forms framework provided by our development environment. We have also built and used C++ shared libraries from the MATLAB code that we have formed, for plotting the maps constructed by our system via its GUI.

5.2 Robot System Models

In this section, the mathematical models associated with the robot vehicle and the camera system are presented.

5.2.1 The Vehicle Model

Our robot is assumed to be running always on a ground plane without inclination. With this assumption a world coordinate frame is defined to determine the robot location at the time that robot starts its motion, which has x and z axes forming

the ground plane and y-axis perpendicular to that plane. Since the plane has no inclination, the robot has a constant position which is 0 in the y-axis. Therefore, the translational position of the robot with respect to the world coordinate frame is defined by only the x and z-axis components. The rotational position of the robot is defined by a single angle θ, since it has one degree of freedom from the rotational point of view. As a result, the robot position with respect to the world coordinate frame is represented by $(x, y, \theta)^T$, where x and z are the coordinates of the robot center and θ is the robot's orientation with respect to the z-axis. The representation of the robot location in the world coordinate frame is illustrated in Figure 5-4.

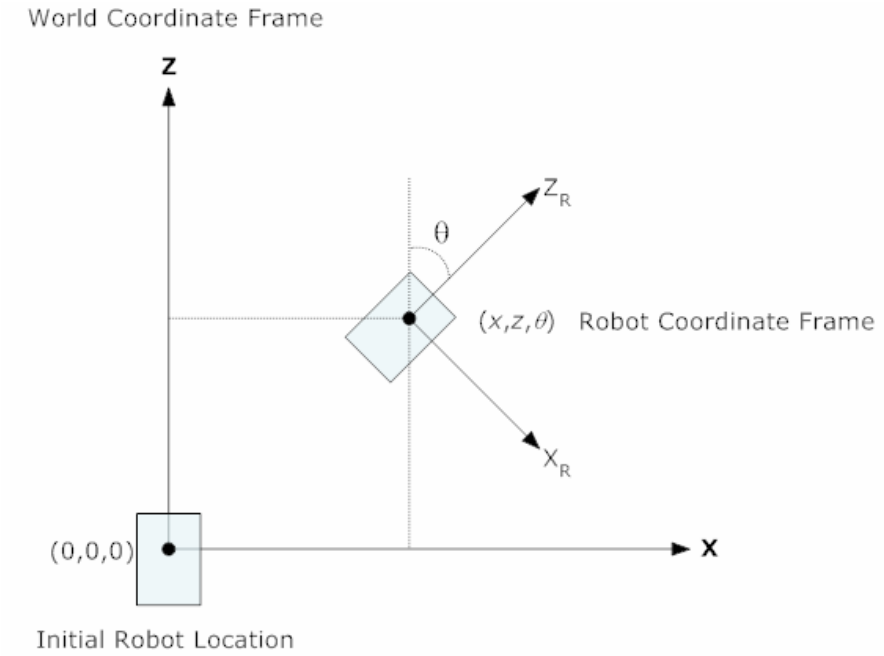

Figure 5-4 The Location of the Robot Vehicle in the World Coordinate Frame

At starting position, all of the robot position parameters have zero values and the robot coordinate frame coincides with the world coordinate frame.

In our thesis, translational and rotational displacements are taken as control inputs instead of velocities. In the latter case displacement cannot be calculated exactly during acceleration and deceleration and incorporating acceleration in the model makes calculations complicated which is not desired. The other reason for this selection is the time difference between the channel switching of the frame grabber card. The switching time is about 100 ms for our system, and this causes an extra positional difference at the time of grabbing image proportional to the speed between the cameras. The depth calculations can still be made by taking account this positional difference, but the calculation of the displacement is not certain as the one in the first case and in the latter case camera model becomes dependent on the vehicle model which is also not desired. While the second reason applies only to stereo operation, the first reason applies both stereo and mono modes of our system. Because of these reasons, the motion types of our vehicle are moving some amount of distance d forward or backward and turning some amount of angle ψ in the requested direction. Fortunately Aria library includes exactly these displacement commands, so our control input vector is $(d,\psi)^T$. At a single time step one of these control inputs can be applied to the robot vehicle.

The new robot location can be calculated as follows:

$$x(k+1) = x(k) + d(k)\sin\theta(k) \tag{5-1}$$

$$z(k+1) = z(k) + d(k)\cos\theta(k) \tag{5-2}$$

$$\theta(k+1) = \theta(k) + \psi(k) \tag{5-3}$$

Surely the robot vehicle cannot attain the exact displacements and come to final positions specified above due to wheel slippages, surface irregularities and internal imperfection of the vehicle itself; that's why we seek to utilize vision tools to correct the robot position. Therefore some uncertainty should be modeled for the motion defined. We model the uncertainty in the control inputs as Gaussian with zero mean. In this uncertainty model we selected standard deviations for the control inputs proportional to the inputs. The standard deviation values are as follows:

$$\sigma_d = kd \tag{5-4}$$

$$\sigma_\Psi = k\Psi \tag{5-5}$$

where $k = 0.2$.

We define the estimated robot position f_v and the control input u as follows:

$$f_v = \begin{bmatrix} x(k+1) \\ z(k+1) \\ \theta(k+1) \end{bmatrix} \quad , \quad u = \begin{bmatrix} d \\ \Psi \end{bmatrix} \tag{5-6}$$

Then we can calculate the covariance of f_v as follows:

$$Q(k) = \frac{\partial f_v}{\partial u} U(k) \frac{\partial f_v}{\partial u}^T \tag{5-7}$$

where $\frac{\partial f_v}{\partial u}$ is the Jacobian of f_v with respect to u and U is the covariance matrix of u:

$$\frac{\partial f_v}{\partial u} = \begin{bmatrix} \frac{\partial x(k+1)}{\partial d} & \frac{\partial x(k+1)}{\partial \Psi} \\ \frac{\partial z(k+1)}{\partial d} & \frac{\partial z(k+1)}{\partial \Psi} \\ \frac{\partial \theta(k+1)}{\partial d} & \frac{\partial \theta(k+1)}{\partial \Psi} \end{bmatrix} \quad , \quad U(k) = \begin{bmatrix} \sigma_d^{\,2} & 0 \\ 0 & \sigma_\Psi^{\,2} \end{bmatrix} \tag{5-8}$$

This uncertainty model is verified to be an appropriate one during the experiments and it gives good results. Therefore we do not need to add other higher order error sources which would certainly complicate the calculations.

5.2.2 The Camera Model

The finite projective camera model mentioned in section 3.2.2 is used in this thesis. The internal camera calibration parameters are found as follows in chapter 3, where the skew is zero.

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5-9}$$

After substituting parameters by the actual values found in the calibration procedure in chapter 3, we obtain the following projection matrices to be used in depth calculations. The projection matrix for the right camera is as follows:

$$P = \begin{bmatrix} 901.47 & 0 & 359.91 \\ 0 & 907.71 & 302.26 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \tag{5-10}$$

And the projection matrix for the left camera is as follows:

$$P' = \begin{bmatrix} 904.69 & 0 & 368.30 \\ 0 & 911.04 & 307.804 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} 0.9997 & 0.0036 & -0.0253 & -94.4883 \\ -0.0034 & 1 & 0.0094 & 0.2030 \\ 0.0253 & -0.0093 & 0.9997 & -0.2843 \end{array}\right] \tag{5-11}$$

As stated earlier, the rotation matrix is an identity matrix and the translation matrix is a zero matrix for the right camera since it is the reference camera. By the help of these projection matrices and the triangulation method defined in 3.3.3, the depth of the detected features can be calculated.

For the measurement model, scalar measurement noise matrices are taken as follows for the stereo and mono cases:

$$R_{stereo} = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 625 \end{bmatrix} \tag{5-12}$$

$$R_{mono} = \begin{bmatrix} 1600 & 0 & 0 \\ 0 & 2500 & 0 \\ 0 & 0 & 15625 \end{bmatrix} \tag{5-13}$$

As it can be seen, the single camera SLAM operation has much more uncertainty in the measurement model, since it depends on the vehicle model.

CHAPTER 6

SIMULTANEOUS LOCALIZATION AND MAPPING SYSTEM

The knowledge of its own location is a fundamental requirement for a robot, in order to navigate autonomously in an environment and perform necessary tasks. Moreover, the knowledge of environment enables robot to fulfill various tasks such as path planning and obstacle avoidance. Since inference of self-location and map of environment entails each other, they are performed sequentially and this process is called *Simultaneous Localization and Map Building*.

In this chapter, we will explain our simultaneous localization and mapping system by defining its components, operational modes and work flows. Our SLAM system has several components such as state estimation, map management, environment perception and graphical user interface. These components will be presented in detail in the following subsections. Furthermore, our system has various operating modes some of which are active concurrently. These concurrent modes are robot modes (*real-simulator*), SLAM modes (*stereo-mono*) and motion modes (*predefined-manual-autonomous*). Various combinations of these working modes can be realized via the user interface component.

Our simultaneous localization and map building system is composed of several functional components each performing different tasks. These various components are presented in Figure 6-1 as follows:

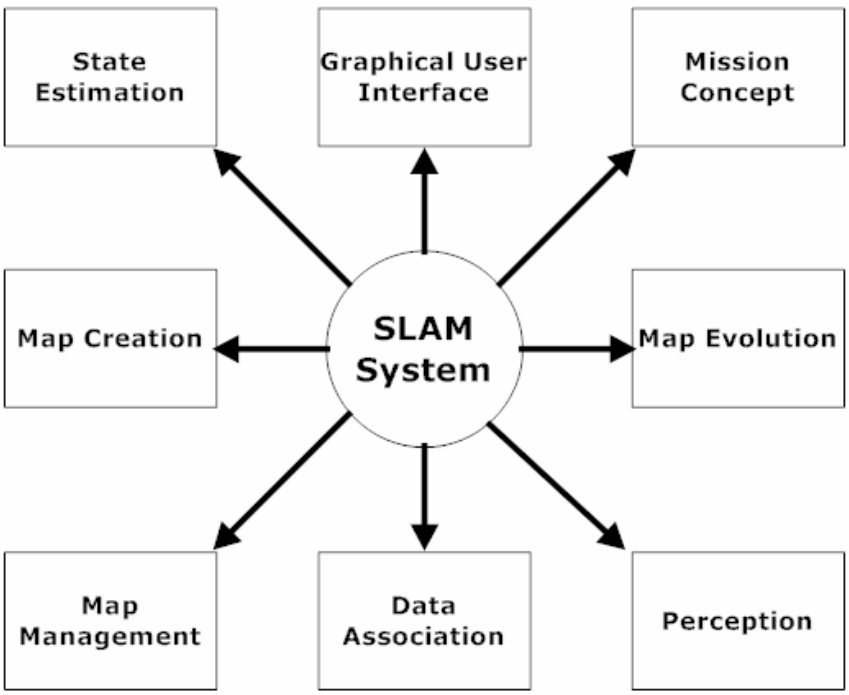

Figure 6-1 SLAM System Components

6.1 State Estimation

The first main part of the SLAM algorithm is state estimation. Continuously estimating its next state, the robot can localize itself in the map that it is creating.

6.1.1 The State Vector

The current estimates of the robot position and the landmark positions are maintained in the state vector $\hat{x}$, which is defined as follows:

$$\hat{x} = \begin{bmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \\ \mathrm{M} \end{bmatrix} \tag{6-1}$$

where $\hat{x}_v$ is the robot position estimate and the $\hat{y}_i$'s are the landmark position estimates belonging to subscripted features. These vectors are defined as:

$$\hat{x}_v = \begin{bmatrix} \hat{x} \\ \hat{z} \\ \hat{\theta} \end{bmatrix} \quad , \quad \hat{y}_i = \begin{bmatrix} \hat{X}_i \\ \hat{Y}_i \\ \hat{Z}_i \end{bmatrix} \tag{6-2}$$

The state vector $\hat{x}$ contains $3(n+1)$ elements, where n is the number of landmarks. These elements indicate positions with respect to world coordinate frame, which is set at the start of the robot motion. The state vector has a dynamic nature, so it is expanded as new landmarks are added to the map and shrinked as existing landmarks are deleted from the map.

6.1.2 The State Covariance Vector

This vector maintains the uncertainties at the robot and landmark position estimates. The state covariance matrix P is defined as follows:

$$P = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & \mathrm{K} \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & \mathrm{K} \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & \mathrm{K} \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} \end{bmatrix} \tag{6-3}$$

Each of the elements of the state covariance matrix indicated above is a 3×3 covariance matrix belonging to the subscripted vectors. The state covariance vector is also dynamic and its size changes parallel to the changes in the state vector.

6.1.3 System Initialization

Since the starting position of the robot is taken as the origin of the map to be constructed, in system initialization, the robot position variables are all set to zero. Furthermore, as we know these values definitely, the state covariance matrix also has all entries equal to zero as below:

$$\hat{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad , \quad P = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6-4}$$

6.1.4 State Prediction Phase

The robot is able to make linear motion or turn motion at each time step. These discrete motion steps are denoted by k and the state estimation due to the motion is made according to the robot motion model given in section 5.2.1. The predicted state vector after a motion is given below:

$$\hat{x}(k+1 \mid k) = f_v(\hat{x}(k \mid k), u(k)) \tag{6-5}$$

where f_v is the state transition function defined in section 5.2.1 taking the current corrected robot state $\hat{x}(k \mid k)$ and the input motion vector $u(k)$ as arguments. Since the features are assumed to be stationary in our system, their predicted positions are taken as the current corrected landmark positions. So, the resulting equation is:

$$\hat{x}(k+1 \mid k) = \begin{bmatrix} f_v(\hat{x}_v(k \mid k), u(k)) \\ \hat{y}_1(k \mid k) \\ \hat{y}_2(k \mid k) \\ \mathrm{M} \end{bmatrix} \tag{6-6}$$

In prediction phase, the new state covariance is constructed according to the general EKF covariance prediction equation given below:

$$P(k+1 \mid k) = \frac{\partial f}{\partial x} P(k \mid k) \frac{\partial f}{\partial x}^T + Q(k) \tag{6-7}$$

where $\frac{\partial f}{\partial x}$ is the full state transition Jacobian matrix, $P(k|k)$ is the current corrected state covariance matrix and the $Q(k)$ is the process noise covariance matrix. These matrices have the following forms for our system:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_v}{\partial x_v} & 0 & 0 & \Lambda \\ 0 & I & 0 & \Lambda \\ 0 & 0 & I & \Lambda \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} \end{bmatrix} \quad , \quad Q(k) = \begin{bmatrix} Q_v(k) & 0 & 0 & \Lambda \\ 0 & 0 & 0 & \Lambda \\ 0 & 0 & 0 & \Lambda \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} \end{bmatrix} \tag{6-8}$$

Since the landmarks observed and used to construct the map are assumed to be stationary, the full state transition Jacobian matrix has identity matrices and the process noise covariance matrix has zero matrices for the corresponding entries. Therefore, the resulting predicted covariance matrix for our system is as follows:

$$P(k+1|k)=\begin{bmatrix} \frac{\partial f_v}{\partial x_v}P_{xx}(k|k)\frac{\partial f_v}{\partial x_v}^T+Q_v(k) & \frac{\partial f_v}{\partial x_v}P_{xy_1}(k|k) & \frac{\partial f_v}{\partial x_v}P_{xy_2}(k|k) & \mathrm{K} \\ P_{y_1x}(k|k)\frac{\partial f_v}{\partial x_v}^T & P_{y_1y_1}(k|k) & P_{y_1y_2}(k|k) & \mathrm{K} \\ P_{y_2x}(k|k)\frac{\partial f_v}{\partial x_v}^T & P_{y_2y_1}(k|k) & P_{y_2y_2}(k|k) & \mathrm{K} \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} \end{bmatrix} \quad (6\text{-}9)$$

where f_v and Q_v are defined in section 5.2.1.

6.1.5 Measurement Prediction Phase

The landmark observations and measurement of their locations are done with respect to the robot coordinate frame. Since we maintain the robot and landmark positions in world coordinate frame in the constructed map, the landmark observations must be associated with global landmark and robot positions, so that the landmark measurement predictions can be done by using this relation as follows:

$$h_i=\begin{bmatrix} h_{ix} \\ h_{iy} \\ h_{iz} \end{bmatrix}=\begin{bmatrix} (x_i-x)\cos\theta-(z_i-z)\sin\theta \\ y_i \\ (x_i-x)\sin\theta+(z_i-z)\cos\theta \end{bmatrix} \quad (6\text{-}10)$$

where x_i, y_i, z_i are the positional values of the ith landmark and x, y, z are the positional values of the robot with respect to the world coordinate frame. By this equation, the predicted measurement for each landmark is calculated and in this way $h(\hat{x}(k+1|k))$ prediction is constructed which is given below:

$$h(\hat{x}(k+1|k)) = \begin{bmatrix} h_1 \\ h_2 \\ \mathrm{M} \end{bmatrix} \tag{6-11}$$

When a measurement is predicted, its innovation covariance that defines the amount of deviation from the predicted values is also calculated using the general innovation covariance equation as follows:

$$S(k+1) = \frac{\partial h}{\partial x} P(k|k) \frac{\partial h}{\partial x}^T + R(k) \tag{6-12}$$

where $\frac{\partial h}{\partial x}$ is the Jacobian matrix of h with respect to state vector x.

The innovation covariance defines a potential volume around the predicted measure of the landmark in which the probability of finding the landmark is high. In order to utilize this information, that volume must be projected on to the camera image planes for the sake of forming a search region for the associated landmark. By this way, we convert the information defined in metric sense into the one described by pixels. This conversion can be achieved using the projection equation and internal camera matrix defined in chapter 3 as follows:

$$u_R = \frac{f_{Rx} h_x + s_R h_y}{h_z} + u_{R0} \quad , \quad v_R = \frac{fyh_y}{h_z} + v_{R0} \tag{6-13}$$

$$u_L = \frac{f_{Lx} h_x + s_L h_y}{h_z} + u_{L0} \quad , \quad v_L = \frac{fyh_y}{h_z} + v_{L0} \tag{6-14}$$

for right and left cameras respectively. Since s_R and s_L values are zero for our camera system, these equations take the following forms:

$$u_R = \frac{f_{Rx} h_x}{h_z} + u_{R0} \quad , \quad v_R = \frac{fyh_y}{h_z} + v_{R0} \tag{6-15}$$

$$u_L = \frac{f_{Lx} h_x}{h_z} + u_{L0} \quad , \quad v_L = \frac{fyh_y}{h_z} + v_{L0} \tag{6-16}$$

The innovation covariance for the image vector $\begin{bmatrix} u_R \\ v_R \end{bmatrix}$ is calculated by using the following equation:

$$U_R = \frac{\partial u_R}{\partial h} S(k+1) \frac{\partial u_R}{\partial h}^T \tag{6-17}$$

where the Jacobian matrix of the u_R with respect to h is as follows:

$$\frac{\partial u_R}{\partial h} = \begin{bmatrix} \frac{f_{Rx}}{h_z} & 0 & -\frac{f_{Rx} h_x}{h_z^2} \\ 0 & \frac{f_{Ry}}{h_z} & -\frac{f_{Ry} h_y}{h_z^2} \end{bmatrix} \tag{6-18}$$

In the same way, the predicted image point and the image covariance is calculated for left image. These calculated U_R and U_L covariances define elliptic search regions on the right and left images by specifying a number of standard deviations.

By using these search regions, the computational cost of template matching is pretty much reduced. Moreover, chance for mismatch of the image templates is decreased. While selecting small number of standard deviations may cause not to find the feature in the constructed region, large numbers increase the number of potential mismatches.

6.1.6 State Vector and Covariance Correction Phase

After prediction of the state vector and its covariance, these predictions are corrected by taking measurements. This update is done according to the Extended Kalman Filter correction rules. Therefore, the Kalman gain is computed as a first step as follows:

$$K(k+1) = P(k+1)\frac{\partial h}{\partial x}^{T} S(k+1)^{-1} \tag{6-19}$$

After gain calculation, the filter update is performed:

$$\hat{x}(k+1 \mid k+1) = \hat{x}(k+1 \mid k) + K(k+1)(z(k+1) - h(\hat{x}(k+1 \mid k))) \tag{6-20}$$

$$P(k+1 \mid k+1) = P(k+1 \mid k) - K(k+1)S(k+1)K(k+1)^{T} \tag{6-21}$$

By the help of these correction equations the state vector and state covariance are updated.

6.2 Map Creation and Evolution

The second main part of the SLAM algorithm is map building. In this process, the observed landmarks with respect to the estimated robot position are inserted into the map under construction, after their coordinates are converted into the world frame. At system initialization, the observed landmarks are added into the empty map and

in this way the map building process starts. Since we want to use our resources efficiently, the number of features to be tracked at the same time is tried to be kept below a threshold which can be defined by the user interface component. So, when the number of tracked features is enough, the usual prediction and correction cycle goes on without any landmark insertion. However this condition does not last for long time, especially for a moving system. After the number of active features, features being tracked currently, goes down below this threshold, the new landmark searching and inserting mechanism takes place.

On the other hand, some landmarks added to the map previously may be the bad ones that they cannot be measured most of the time, even though they are predicted to be observed. In these cases, it is needed to have a method for deletion of these landmarks. We propose a deletion condition similar to one in [45] for the landmarks by defining some variables. In order to figure out if a feature is useless, some number of measurement attempts has to be done. We define this number as 15 by default, however this variable with all other ones can be altered via the user interface in order to adapt to the current conditions. When this number is reached, the landmarks are started to be continuously checked for deletion. If the ratio of immeasurable attempts to the measurable ones is greater than a specified threshold which we define as 0.5 by default, then the landmark is deleted by the system. Of course the measurement attempt is done for a landmark if the robot state satisfies the condition for that landmark as stated earlier, which is if the robot distance to the landmark and the view direction are in the ranges specified when the landmark is first initialized. The deletion of landmarks is also required when a total number of features exceed a threshold and the real time efficiency control is active, which will be described in section 6.3.

Having explained the landmark addition and deletion conditions, we can now present how these processes are performed.

6.2.1 Addition of New Landmarks

At system initialization, new landmarks are searched and added into the map. The addition process is also required, when the number of active landmarks to track and correct the state prediction, becomes below a threshold. New landmark addition enlarges the state vector and the state covariance with new elements associated with the landmark added. Since the state vector contains landmark positions with respect to world frame and landmark measurements are obtained with respect to robot frame, these measurements are converted to appropriate frame as follows:

$$y_i = \begin{bmatrix} x_v + h_{ix}\cos\theta + h_{iz}\sin\theta \\ h_{iy} \\ z_v - h_{ix}\sin\theta + h_{iz}\cos\theta \end{bmatrix} \tag{6-22}$$

After obtaining the position of the new landmark with respect to world frame, this vector is simply added at the end of the state vector:

$$\hat{x}_{new} = \begin{bmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_i \end{bmatrix} \tag{6-23}$$

The new state covariance is constructed after calculating the Jacobians $\frac{\partial y_i}{\partial x}$ and $\frac{\partial y_i}{\partial h}$ as follows:

$$P_{new} = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & P_{xx}\frac{\partial y_i}{\partial x}^T \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & P_{y_1x}\frac{\partial y_i}{\partial x}^T \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & P_{y_2x}\frac{\partial y_i}{\partial x}^T \\ \frac{\partial y_i}{\partial x}P_{xx} & \frac{\partial y_i}{\partial x}P_{xy_1} & \frac{\partial y_i}{\partial x}P_{xy_2} & \frac{\partial y_i}{\partial x}P_{xx}\frac{\partial y_i}{\partial x}^T + \frac{\partial y_i}{\partial h}R\frac{\partial y_i}{\partial h}^T \end{bmatrix} \quad (6\text{-}24)$$

where R is the scalar measurement noise covariance.

6.2.2 Deletion of Landmarks

When a feature existing in the map is figured out to be a bad one, it is removed from the map in order to make room for more strong and detectable landmarks. Deletion process is simply removing the row and column associated with the feature to be deleted from the state vector and associated covariance matrix. Deletion of the i^{th} feature is illustrated below:

$$\begin{bmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_i \end{bmatrix} \Rightarrow \begin{bmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} \quad (6\text{-}25)$$

$$\begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & P_{xy_i} \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & P_{y_1y_i} \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & P_{y_2y_i} \\ P_{y_ix} & P_{y_iy_1} & P_{y_iy_2} & P_{y_iy_i} \end{bmatrix} \Rightarrow \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} \end{bmatrix} \quad (6\text{-}26)$$

6.3 Map Management

During the map building process, a map management strategy is required in order to run the algorithm effectively and get rid of useless features. One part of this strategy

is explained in the previous section. The other part of the strategy takes role when the total number of landmarks reaches a threshold. Even though a deletion strategy is defined for bad features, the number of landmarks can reach high levels especially when the robot runs in large environments. In this case, the algorithm's real time performance degrades, since it conveys a covariance matrix for the state vector which becomes very huge with high number of elements and calculations get more complex. In order to avoid this situation, some landmarks need to be removed from the current map. The selection of landmarks for deletion is done by considering the uncertainties and the geometric distribution of them in the current map. Since the landmarks with low uncertainties provide more reliable information they must be preserved. However, this protection is limited by the geometric conditions. To state more clearly, after the deletion process the remaining landmarks should still represent the whole environment. Therefore the smallest rectangular prism which encapsulates all of the landmarks is formed. Then this prism is divided into sub blocks, the number of which is defined via the user interface. For division of the map we defined division parameters x_{count}, y_{count} and z_{count}, which designate the number of intervals in the corresponding axes. After parceling the environment, the deletion process of the most uncertain landmarks is started. At the end of this process, each parcel containing at least one landmark at the beginning should still contain at least one landmark, so that the geometric distribution of the landmarks is preserved. An example geometric division of the mapped environment and the landmarks in it are shown in the following figure, where $x_{count} = 3$, $y_{count} = 3$ and $z_{count} = 4$:

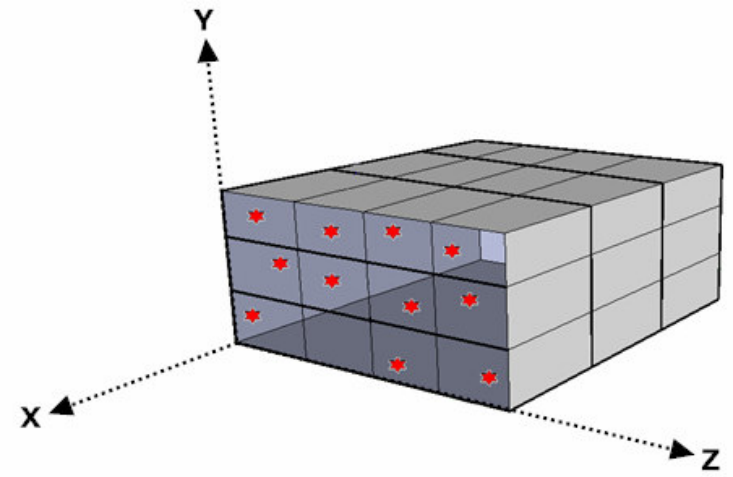

Figure 6-2 Division of the map into sub blocks

6.4 Data Perception and Association

This part of the system is discussed in detail in chapter 3.

6.5 Graphical User Interface

This part of the system is presented in detail in appendix.

6.6 Mission Concept

Having gained the localization capability, the robot can complete the given tasks and missions such as goal reaching. The mission concept for our system is about the capability of the robot for reaching an assigned location. This task is achieved by the robot by simply calculating the angle it must turn and the distance it must go using the line defined by its current position and the target position. After these calculations have been made, robot can arrive to the goal location making necessary movements. However, there may be an obstacle on the path that the robot has planned. In this case robot just avoids the obstacle by the help of its sonar sensors, and then new angle and distances are calculated to plan the new path.

6.7 Operational Modes of the SLAM System

The Simultaneous and Mapping System has several concurrent modes. The system can run various operations with combination of different operating modes.

First of all, there are two main robot modes which are simulation and real modes. In simulation mode the system connects the simulator robot, which is named *MobileSim* in this application. In this mode, the motion control of the robot can be run and tested without a real robot. In real mode, a real Pioneer robot is tried to be connected. After connection has been established, one can perform various operations that SLAM system provides. The simulator and real robots are illustrated in the following figures:

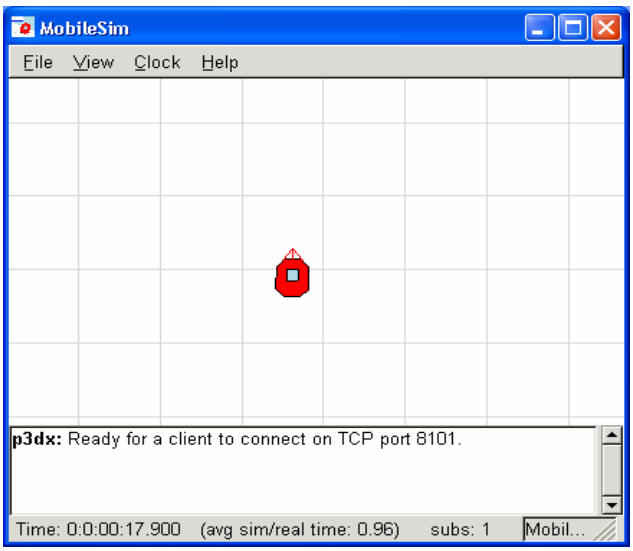

Figure 6-3 Simulator Robot: MobileSim

Figure 6-4 Real Robot: Pioneer

Another working mode is SLAM mode which has two alternatives, namely Stereo SLAM and Mono SLAM. If the system has two calibrated cameras, a Stereo SLAM operation can be done by selecting Stereo mode. In Stereo mode, the depth perception for the landmarks is achieved immediately, since the calibrated stereo camera system allows triangulation by using the left and right camera images.

However, the system also allows running with single camera if selected. While operating in Mono mode, the system uses only the right camera for image grabbing task. Since one image is not enough for depth perception quantitatively, the system takes grabs of environment two times in a specified consequent time steps, compared to Stereo case. By making use of the motion parallax obtained during robot movement, system can detect the landmark depths again by triangulation. For mono operation, normally the camera optical axis should be perpendicular to the motion direction in order to obtain motion parallax and derive the depth information. But the cameras are mounted parallel to the motion direction in our robot system, so the robot has to grab an image, then turn some angle (usually 90º) and move some distance to provide a baseline distance for parallax, and again turns to its initial direction to grab the second image. Having grabbed two images from

different positions, the robot provides itself a motion parallax and makes usual triangulation calculations explained in chapter 3. Since the uncertainty in the robot motion to get parallax is much higher than the stereo calibration parameters, the depth information obtained in this way has more error rate than the one obtained by the stereo camera system. Nevertheless, the depth measurement experiment results of the mono operation are not bad and presented in chapter 6. In [17], also two kinds of SLAM approaches, stereo and monocular, are presented. Their EKF based stereo SLAM solution is similar to ours. However for single camera SLAM, which they called $Bearings-Only\ SLAM$, they use sum of Gaussians approach. In this approach, they initialize the representation of a feature with a sum of Gaussians. Then this representation is updated as robot moves, until a single hypothesis remains. In this method the camera principal axis is nearly perpendicular to the motion direction, so a smooth hypothesis tracking is possible which is not valid for our static camera system.

To show using a single camera for the SLAM operation is important due to several reasons. These reasons are related with the system requirements to be implemented of course. For instance, if a system that localizes itself roughly in the environment while mapping also is enough and there is no need for a much accurate localization, SLAM with a single camera will be the first choice due to its lower cost. Another reason may be a requirement for an error recovery in the system. System may operate in stereo mode normally, to localize itself in the constructed map. However, the system stops working when one of the cameras fails to work if there is no mono operation capability. Having the ability to operate with a single camera, a robot system can just switch to mono operation mode in the case of an error in one of the cameras. In this way, more robust robot systems can be built up.

The main SLAM algorithm, that is state estimation and map building parts, are same for both stereo and mono modes. Only the depth perception ways differ in these operational modes. We have also mentioned that the camera construction in our robot system is not appropriate for mono operation and our robot does not have unfortunately moving head. Therefore in the experiments chapter, only the depth

perception and “backward-forward” motion performances of mono and stereo modes are compared, in which they only differ. The other experiments are related with the common components of both modes and the results are obtained using the stereo mode.

The parallel operational modes of our SLAM system are shown as a statechart representation in Figure 6-5:

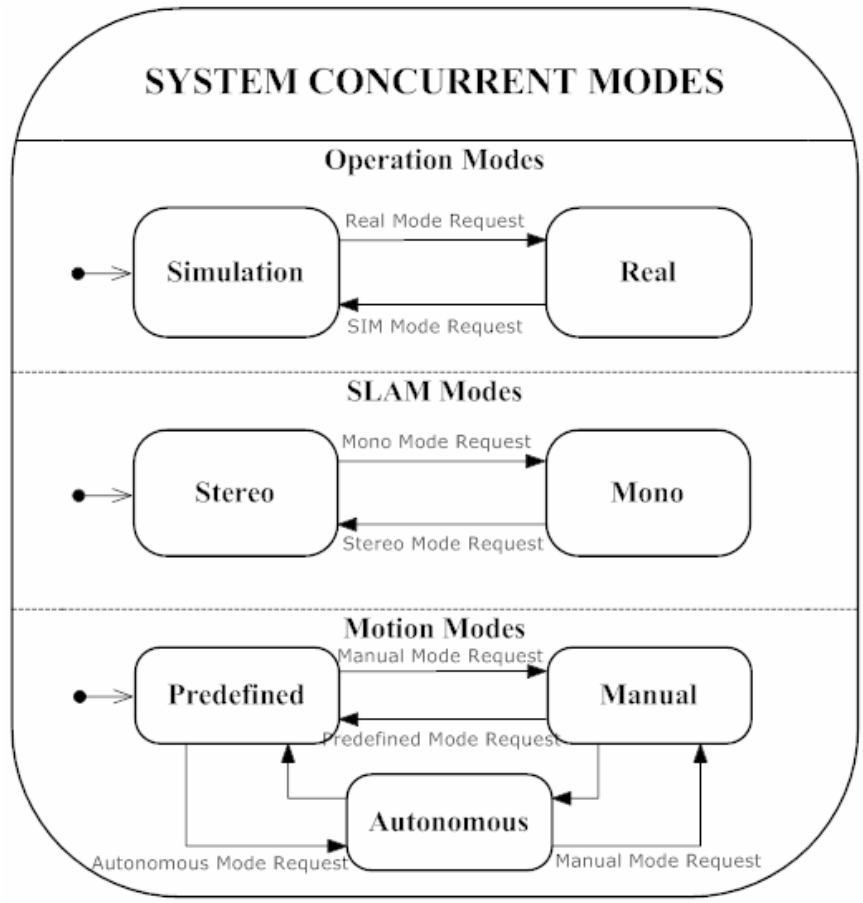

Figure 6-5 Parallel System Modes

The motion control of the SLAM system can also be determined with respect to the motion mode choice. There are three motion modes for our system. In Predefined mode, the user has to prepare a predefined list of motion commands and load it to the system via the graphical user interface. Then the robot operates with these commands, until the last command is processed. As motion commands are executed, the Extended Kalman Filter prediction and correction phases are fulfilled. The detailed flowchart of this operational mode is depicted in Figure 6-6.

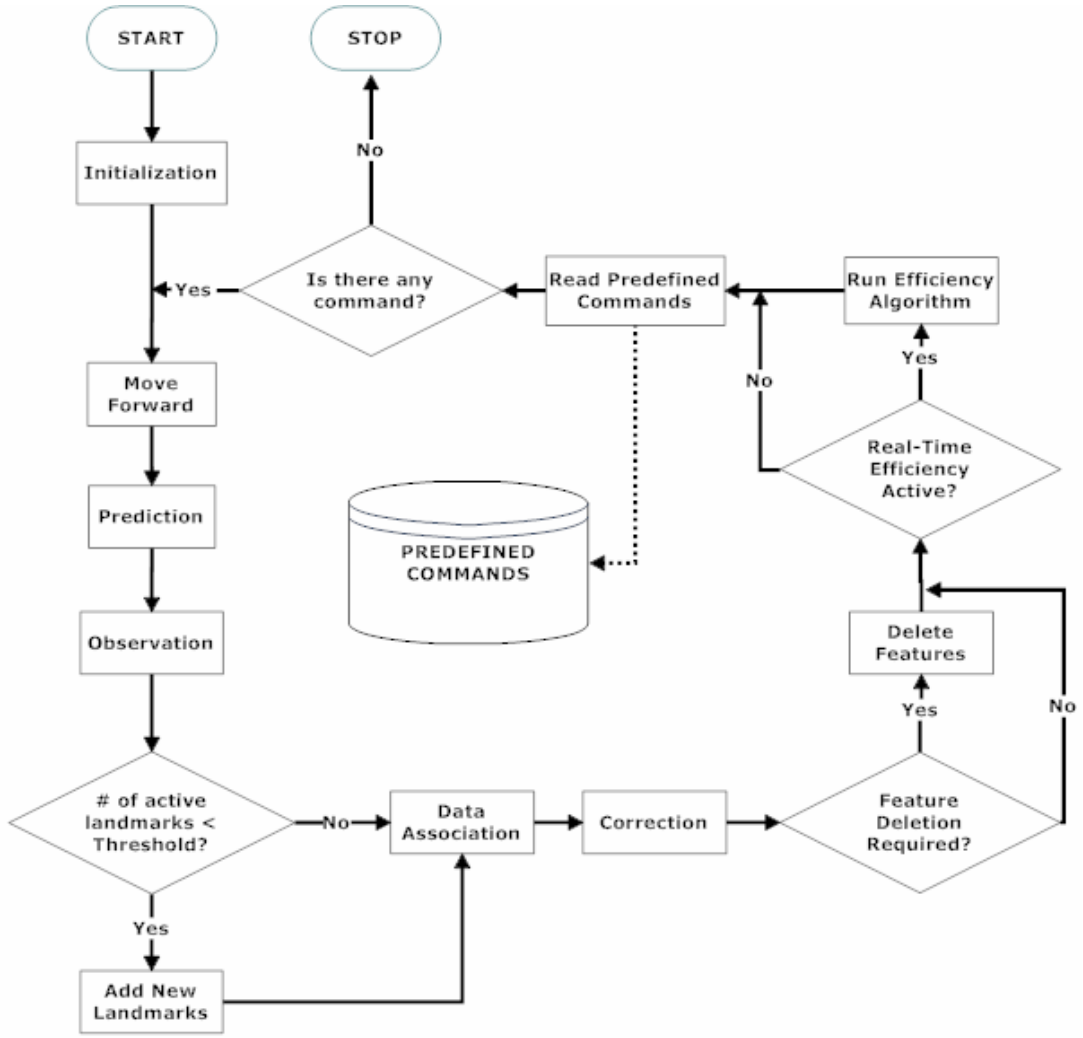

Figure 6-6 Flowchart of SLAM with Predefined Motion

In Manual mode, the system waits for keyboard requests from the operator in order to control robot. These commands then construct the robot's path and SLAM operation continues, until no more user request is received. The corresponding flowchart is given below.

Figure 6-7 Flowchart of SLAM with Manual Inputs

Lastly, the system has an autonomous control capability, which permits it to control the robot without any external intervention. In this case, the robot freely moves forward until it reaches an obstacle. Then the obstacle is avoided by changing movement direction, and the movement is carried on in the new direction. A fully autonomous SLAM operation can be achieved in this way and corresponding flowchart is illustrated in Figure 6-8.

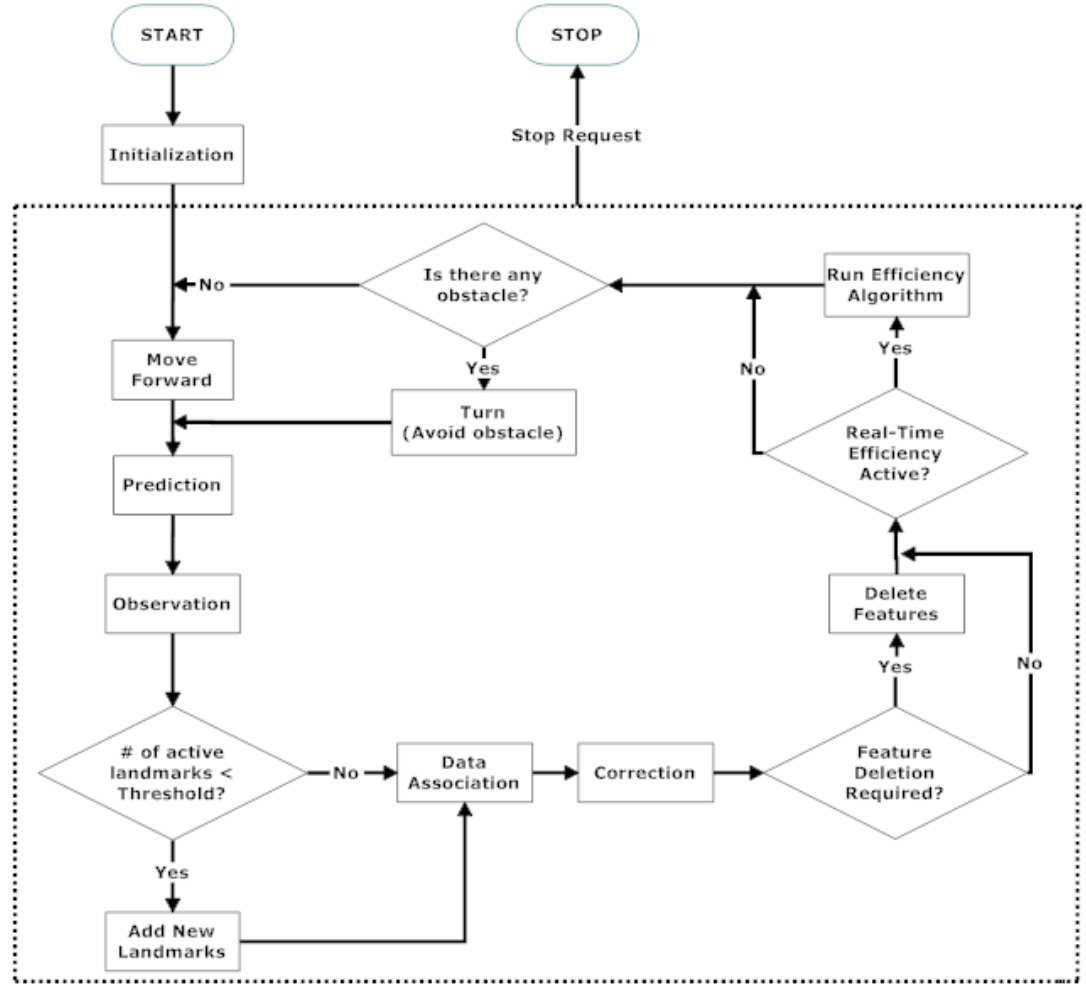

Figure 6-8 Flowchart of SLAM in Autonomous Mode

CHAPTER 7

EXPERIMENTS AND RESULTS

As stated earlier our system has both simulator and real robot modes, so that it can be tested under both modes. The simulation mode provides us a great deal of opportunities for testing our system especially during development time. Nevertheless, the actual performance of the system can be seen exactly in the real mode. Therefore, a number of experiments have been carried out in our laboratory in order to evaluate the system performance and discuss the results.

In the middle of our laboratory there is necessary space for the robot to operate, and there are tables, chairs, windows and bookshelves which the robot can make use of their strong features as its landmarks for map construction. In order to evaluate the robot localization and map building performance we need a ground truth reference. Luckily our laboratory has a grid which was marked previously. Although some parts of the grid are erased it still gives necessary information. The grid contains regularly spaced squares with dimensions of 50 millimeters. By the help of this grid, we can measure the position of the robot metrically. Of course some detected landmarks have been out of the grid range and the ground truth measurements of these features have been made by using an extra ruler.

The colors of the squares that designate the found corners on the image have the following meaning. If the color of the square is green, the landmark it surrounds is observed for the first time and added to the map. However the landmark is an existing one and said to be tracked by the system, if the color of the square is blue.

Our experimental setup is illustrated in Figure 7-1:

Figure 7-1 Experimental Setup

Experiment 1: Depth Measurement Tests

In this experiment, we conduct depth test with both stereo and mono camera operational modes of our system. In order for the SLAM algorithm operates accurately, the measurements should be accurate enough and error range must not be high. Of course the measurement errors are tried to be compensated by defining some noise and uncertainty terms, but it is certain that more accurate measurements lead a more accurate system.

For the test setup, a white board with a black filled square pasted on it is placed in front of the camera. For the stereo depth measurements we take the right camera as reference and the measurements are conducted with respect to the principal axis of

the right camera. The different distances are measured by changing the position of the robot.

The depth measurement test results for the stereo case are given below:

Table 7-1 Stereo Camera Depth Measurement Test Results

Actual Depth (mm)	**Measured Depth (mm)**	**Error Percentage (%)**
500	506	1.2
1000	1011	1.1
1500	1540	2.66
2000	2060	3
2500	2572	2.8
3000	3100	3.3
3500	3581	2.3
4000	4142	3.5
4500	4653	3.4
5000	5182	3.6

The results are quite good since the measured values are close to the actual values. The error percentages are also given in the table. We think that the irregularities in the error percentages originate from the placement errors of the robot in front of the white board.

For the single camera depth measurement tests we use the right camera which is also used during the mono mode in the real system operation. The measurements are conducted as follows:

1. The first image is grabbed by the right camera
2. The robot turns a right angle and goes 100 mm straight ahead
3. The robot comes back to its initial direction and grabs second image

The depth measurement test results for the single camera case are given below:

Table 7-2 Single Camera Depth Measurement Test Results

Actual Depth (mm)	**Measured Depth (mm)**	**Error Percentage (%)**
500	560	12
1000	1152	15.2
1500	1670	11.3
2000	2212	10.6
2500	2763	10.5
3000	3377	12.5

The results are worse than that of the stereo case as we expected, since the measurements are done by assuming that the robot makes the specified motions exactly without errors and the surface that the robot stands on does not have local inclinations. But of course, our robot makes its movements with some errors, that's why we are implementing a state estimation algorithm, and the surface may have some little irregularities. Although these results are worse than the stereo case, they still indicate the applicability of the mono SLAM algorithm within some error boundary.

Experiment 2: Backward and Forward Motion

In this experiment the robot is again placed in front of the white board with black square and forced to move forward or backward at each time step. The distance request to travel at one step is 100 mm. At specific time steps, we have read the estimated results by our system, odometry values and the ground truth values.

The test results for this experiment are given below. Since our ground truth measurements are not accurate to measure the magnitudes shorter than millimeters, we have not specified values after the decimal point. Moreover we have set the ground truth values to zero for the x-axis position and angle of the robot, again because we cannot measure too little changes.

The notes in parentheses next to time steps in which the experimental results are illustrated indicate the motion pattern that is made between the associated step and the time step in which the last experimental results are presented.

The legend for the constructed maps for the experiments is shown in Figure 7-2.

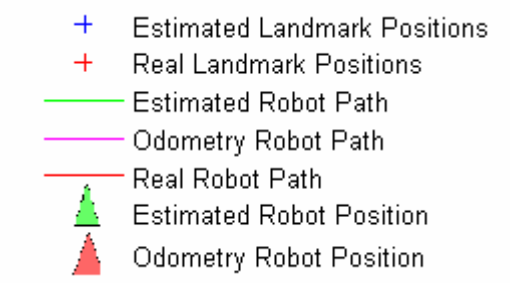

Figure 7-2 Legend for Constructed Maps

Initialization:

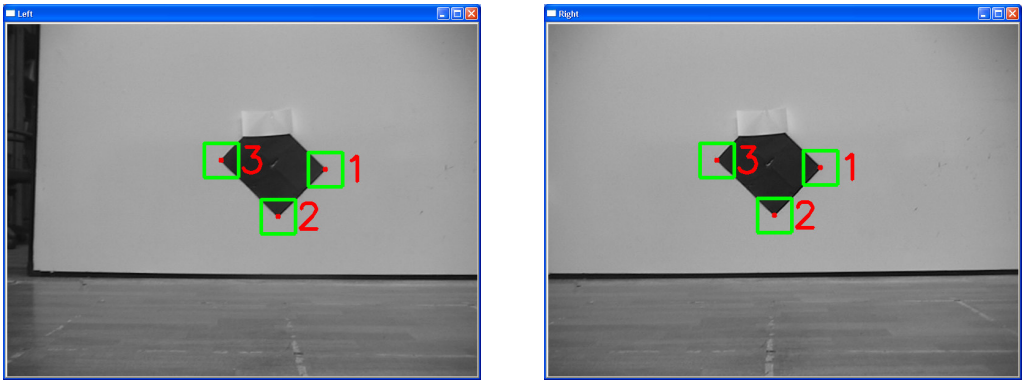

Figure 7-3 Camera Images for Initialization of Experiment 2

Table 7-3 Real Landmark Positions

	X (mm)	**Y (mm)**	**Z (mm)**
Landmark 1	185	130	2050
Landmark 2	10	-50	2000
Landmark 3	-195	160	2050

Table 7-4 Estimated Landmark Positions

	X (mm)	**Y (mm)**	**Z (mm)**
Landmark 1	187.39	144.28	2008.67
Landmark 2	20.72	-27.46	2053.65
Landmark 3	-188.44	173.10	2000.71

5th Step (After 5 Movements Forward):

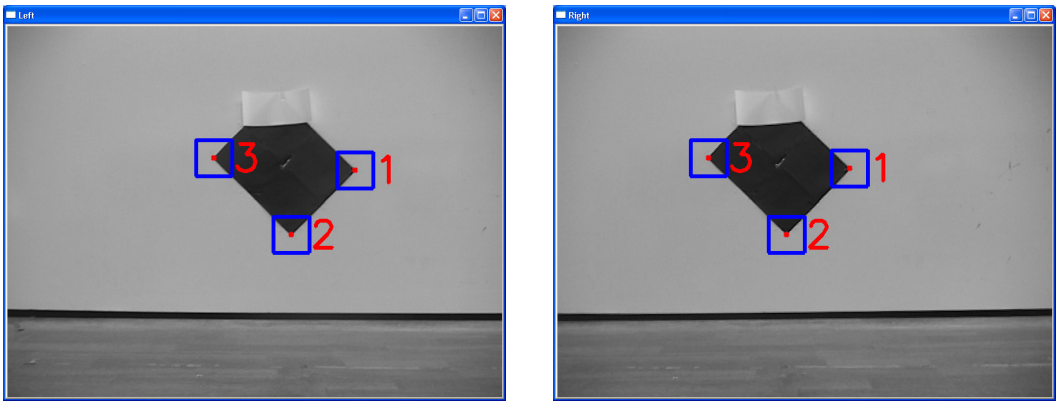

Figure 7-4 Camera Images for 5th Step of Experiment 2

Table 7-5 Estimated Landmark Positions at 5th Step for Experiment 2

	X (mm)	**Y (mm)**	**Z (mm)**
Landmark 1	188.15	127.98	2022.59
Landmark 2	19.52	-46.55	2037.80
Landmark 3	-188.17	153.51	1987.45

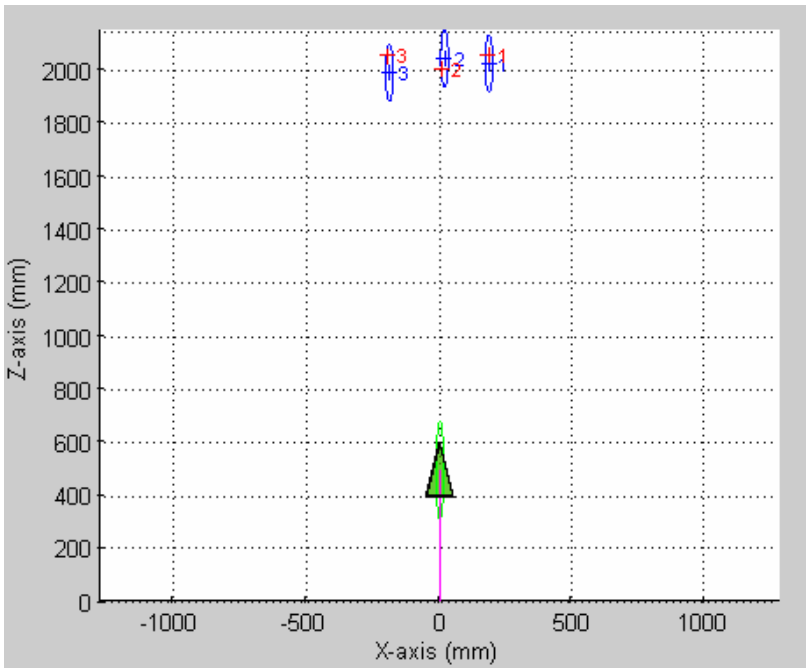

Figure 7-5 Predicted 2D Map for 5th Step of Experiment 2

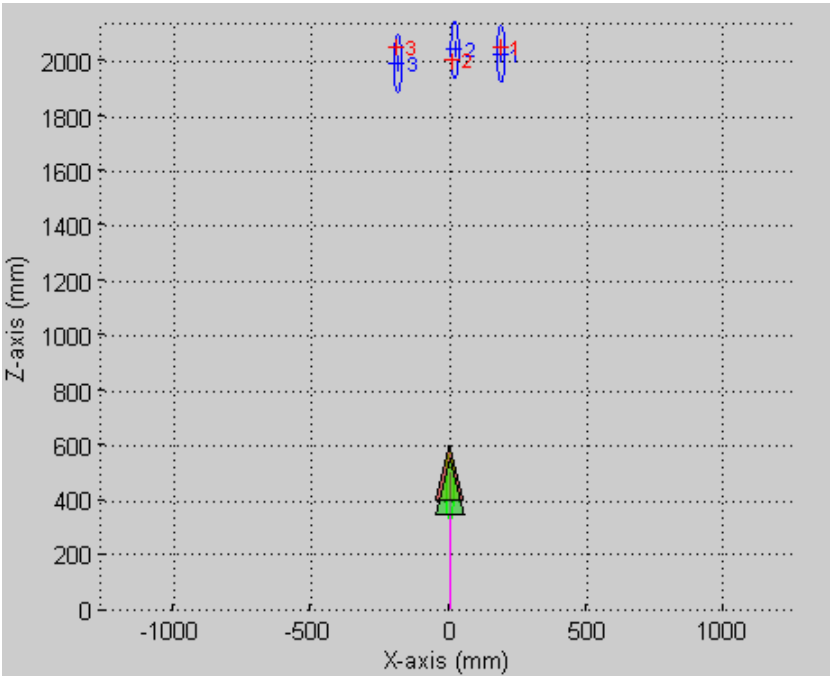

Figure 7-6 Corrected 2D Map for 5th Step of Experiment 2

10th Step (After 5 Movements Forward):

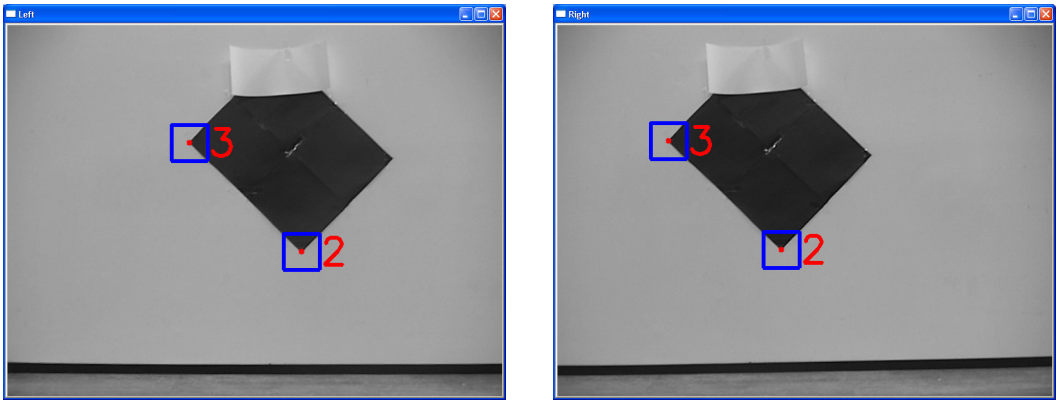

Figure 7-7 Camera Images for 10th Step of Experiment 2

Since the similarity threshold in this experiment is 0.9 and similarity value obtained for landmark 1 is 0.87, it cannot be observed.

Table 7-6 Estimated Landmark Positions at 10th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	189.75	125.52	2025.09
Landmark 2	19.24	-50.99	2030.32
Landmark 3	-189.55	150.03	1992.43

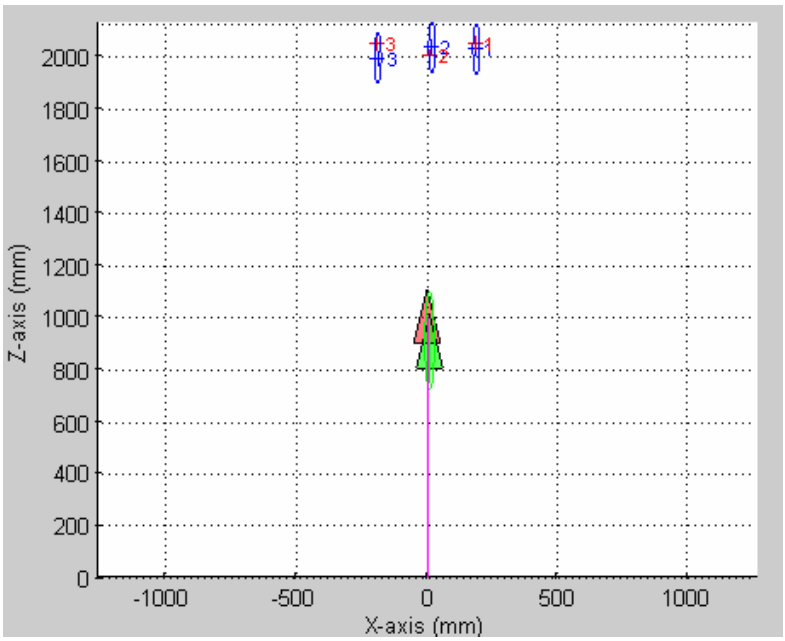

Figure 7-8 Predicted 2D Map for 10th Step of Experiment 2

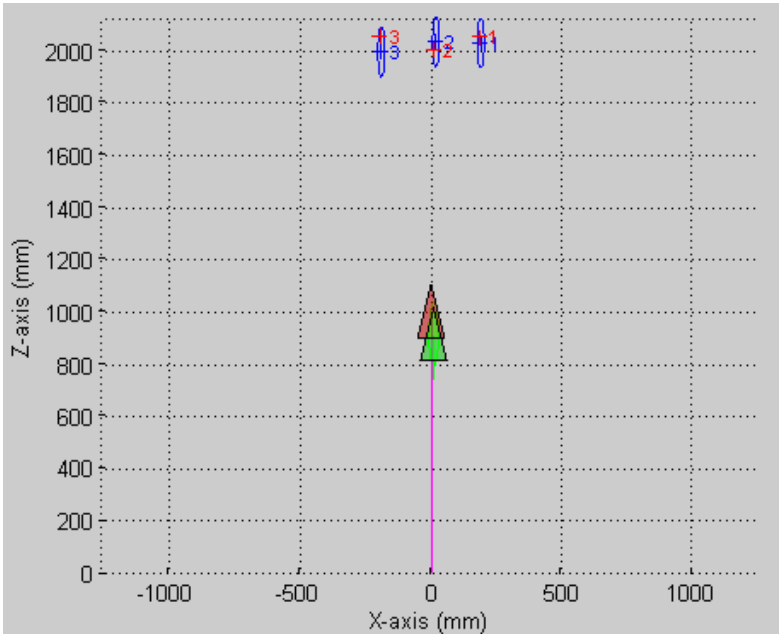

Figure 7-9 Corrected 2D Map for 10th Step of Experiment 2

15th Step (After 5 Movements Backward):

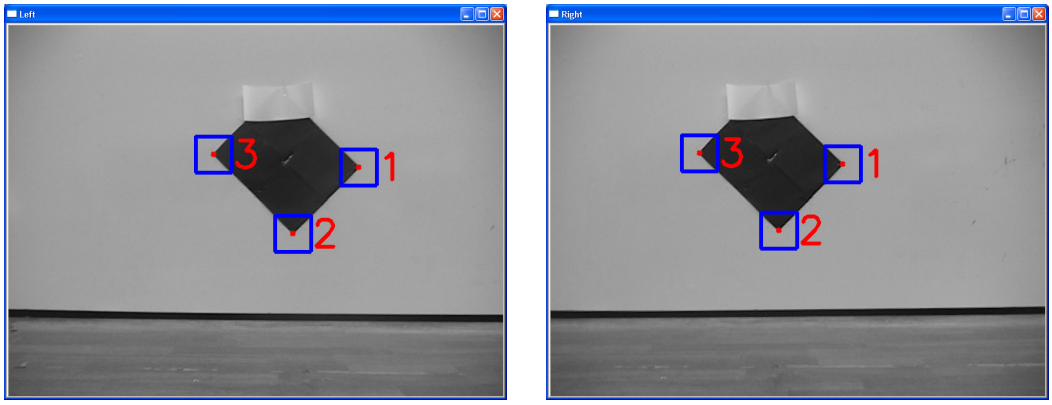

Figure 7-10 Camera Images for 15th Step of Experiment 2

Table 7-7 Estimated Landmark Positions at 15th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	188.77	123.20	2016.04
Landmark 2	19.76	-51.25	2026.37
Landmark 3	-189.08	149.91	2005.43

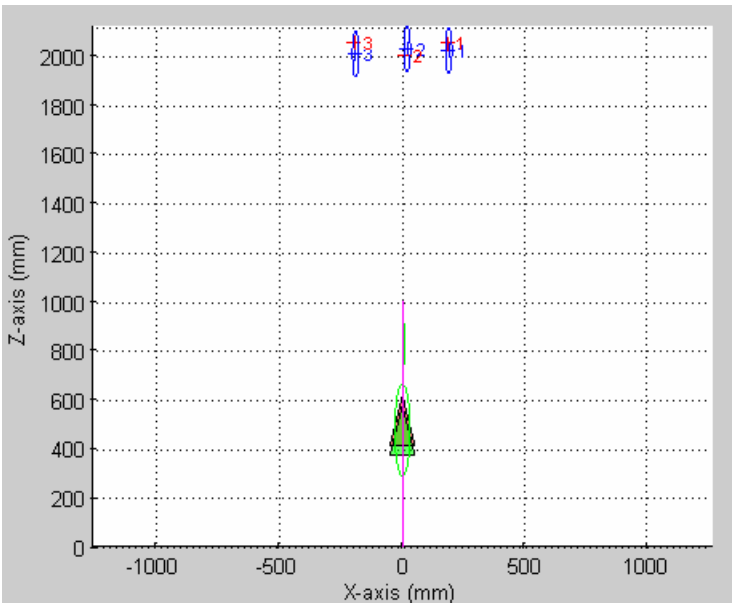

Figure 7-11 Predicted 2D Map for 15th Step of Experiment 2

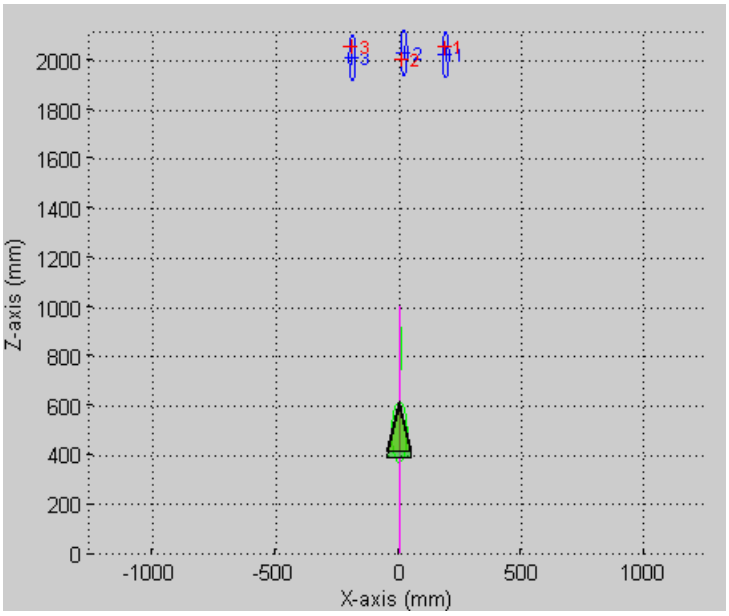

Figure 7-12 Corrected 2D Map for 15th Step of Experiment 2

20th Step (After 5 Movements Backward):

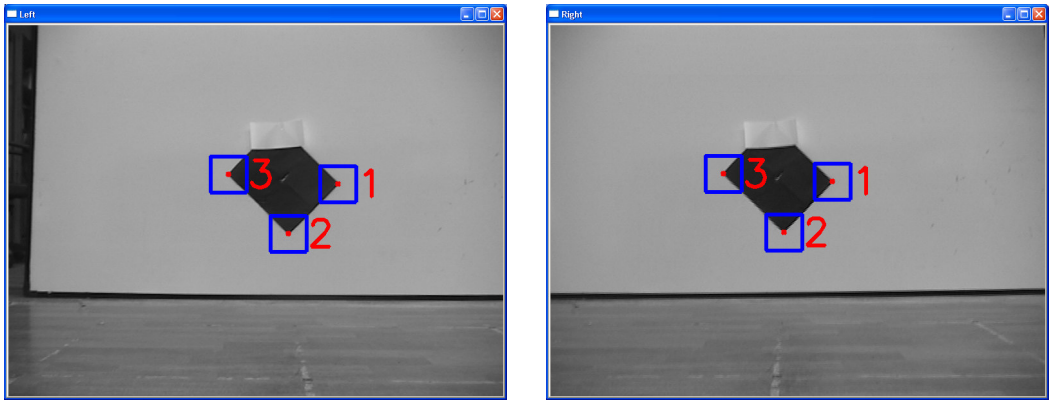

Figure 7-13 Camera Images for 20th Step of Experiment 2

Table 7-8 Estimated Landmark Positions at 20th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	189.04	123.88	2009.07
Landmark 2	20.40	-50.76	2027.11
Landmark 3	-189.84	151.93	2011.65

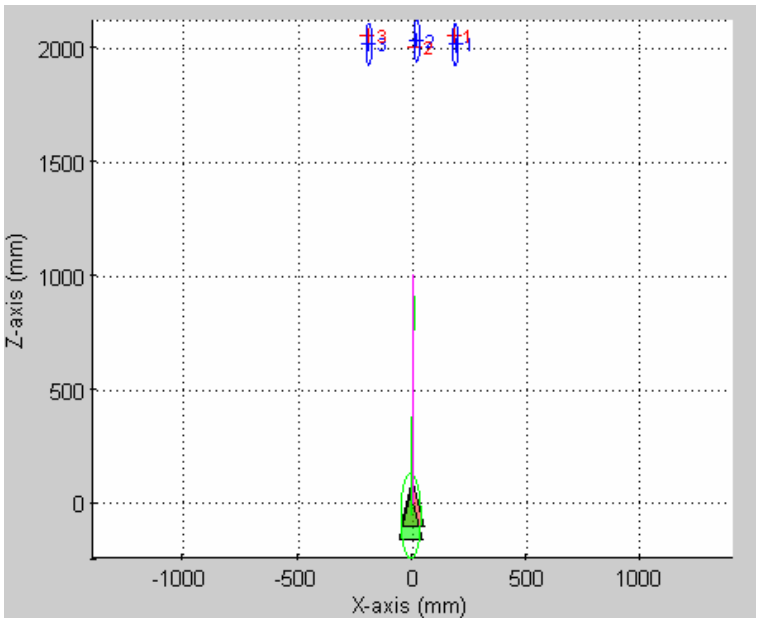

Figure 7-14 Predicted 2D Map for 20th Step of Experiment 2

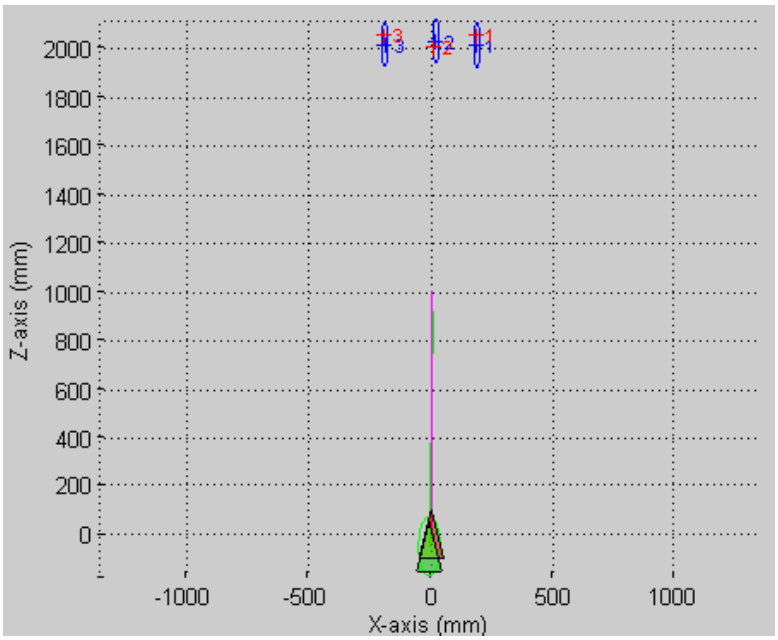

Figure 7-15 Corrected 2D Map for 20th Step of Experiment 2

25th Step (After 5 Movements Backward):

Figure 7-16 Camera Images for 25th Step of Experiment 2

Table 7-9 Estimated Landmark Positions at 25th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	188.92	124.29	2004.13
Landmark 2	20.72	-50.12	2028.13
Landmark 3	-189.99	153.23	2015.57

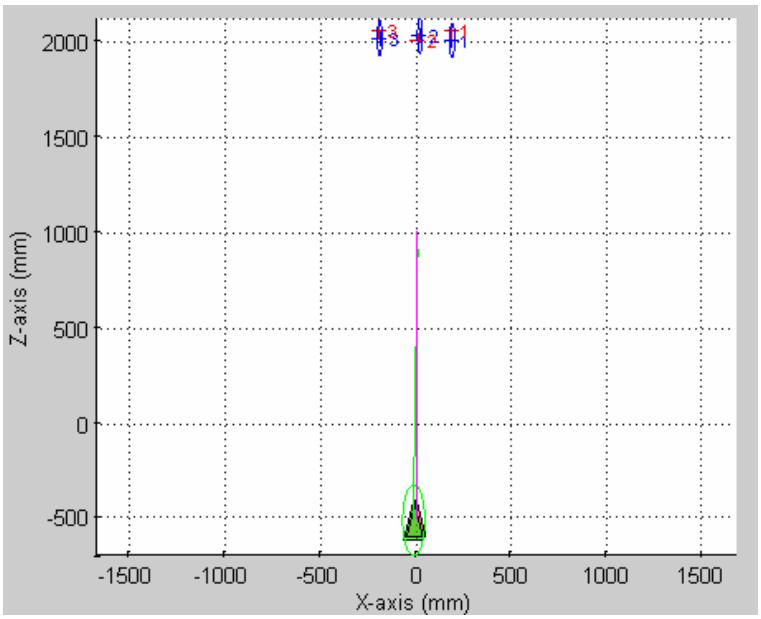

Figure 7-17 Predicted 2D Map for 25th Step of Experiment 2

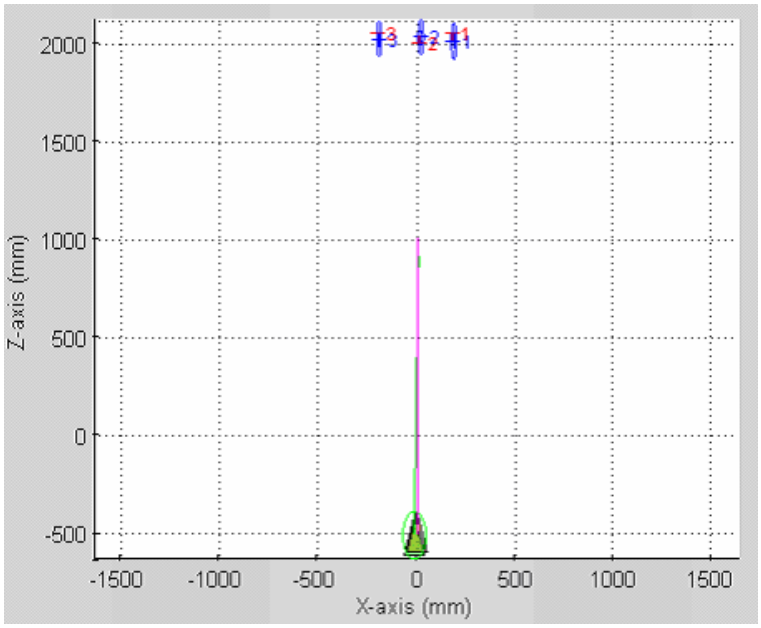

Figure 7-18 Predicted 2D Map for 25th Step of Experiment 2

30th Step (After 5 Movements Forward):

Figure 7-19 Camera Images for 30th Step of Experiment 2

Table 7-10 Estimated Landmark Positions at 30th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	187.27	124.28	2005.04
Landmark 2	20.32	-48.81	2031.08
Landmark 3	-188.54	152.65	2011.77

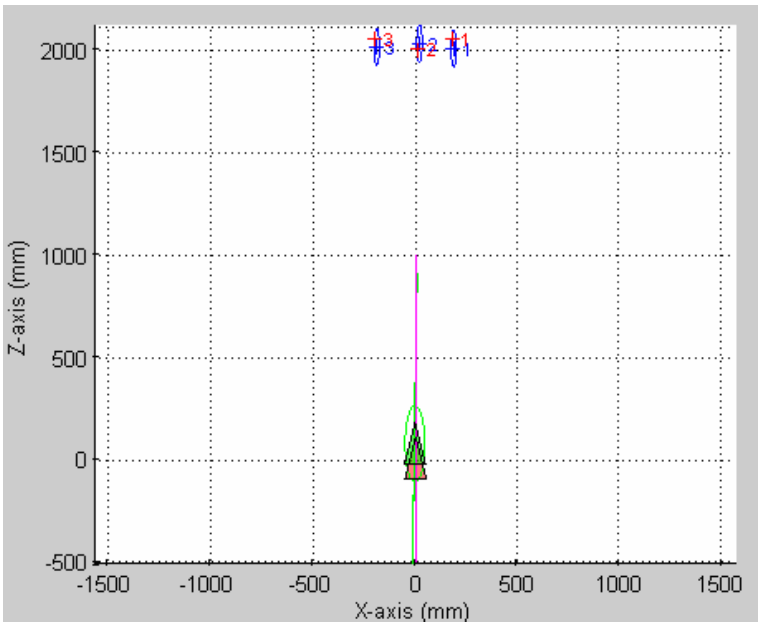

Figure 7-20 Predicted 2D Map for 30th Step of Experiment 2

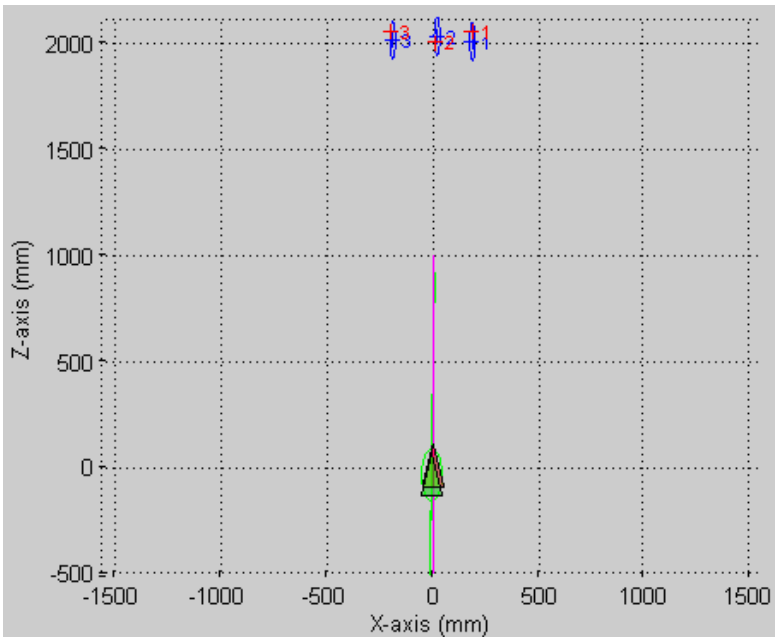

Figure 7-21 Predicted 2D Map for 30th Step of Experiment 2

After final step, it is seen that the estimated robot position is close to the real one, whereas the odometry reading deviates some amount from the actual position even though the motion of the robot on the laboratory surface is regular and not much slippage is encountered. The impressing results for the case where some slippage is introduced to the system will be shown in the next experiment. The landmark position estimations are also close to real ones.

The camera images for left and right cameras while robot is tracking the landmarks are as follows:

Table 7-11 Robot Positional Values for Experiment 2

Time Step	**Estimated (mm)**	**Odometry (mm)**	**Ground Truth (mm)**
0	0	0	0
5	447.45	497.61	475
10	911.30	1000.07	940
15	489.90	512.65	490
20	-51.32	0.49	10
25	-512.65	-506.34	-470
30	-39.07	5.82	10

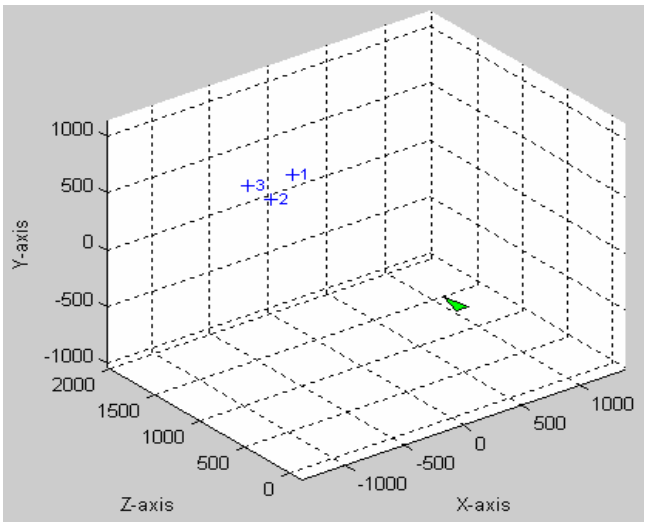

Figure 7-22 Estimated 3D Map of Experiment 2

The robot positional covariance variation is shown in Figure 7-23. Robot position uncertainty increases with time. But the uncertainty decreases at time step 30, where the robot is at its initial position.

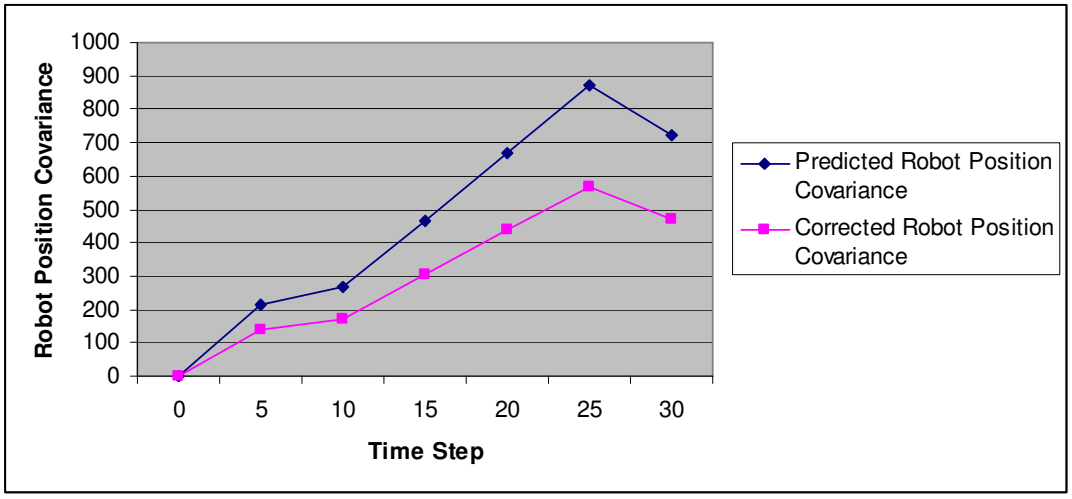

Figure 7-23 Robot Position Covariance Variation of Experiment 2 (mm^2)

The robot estimated and odometry position errors are nearly similar for this experiment.

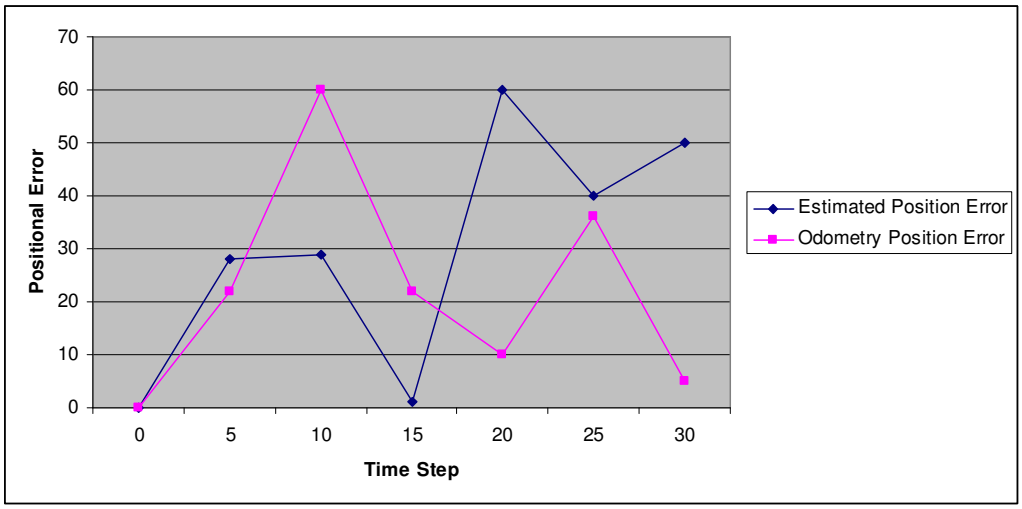

Figure 7-24 Robot Positional Error Variation of Experiment 2 (mm)

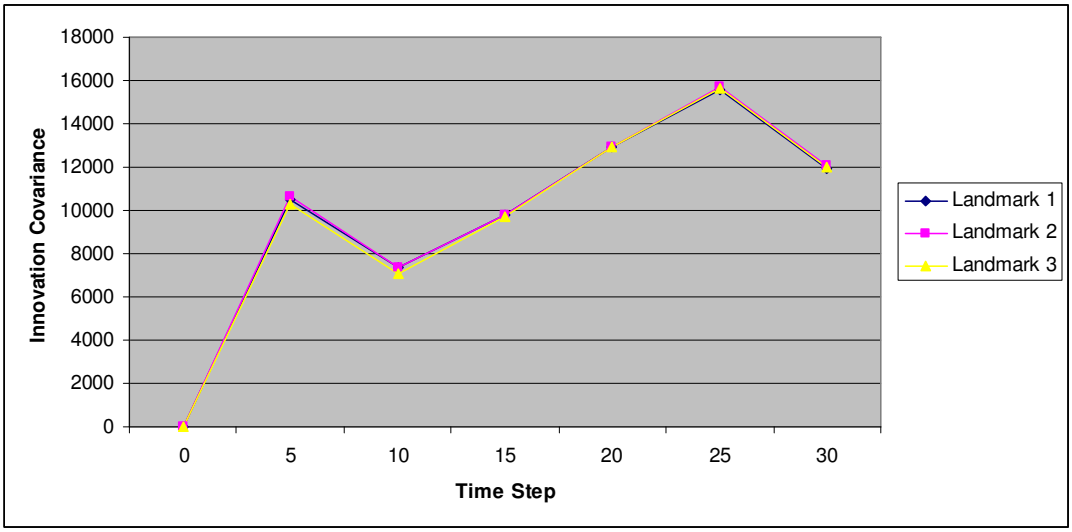

Figure 7-25 Innovation Covariance Variation of Experiment 2 (mm^3)

The innovation covariances of landmarks first increase and then decrease after some time steps.

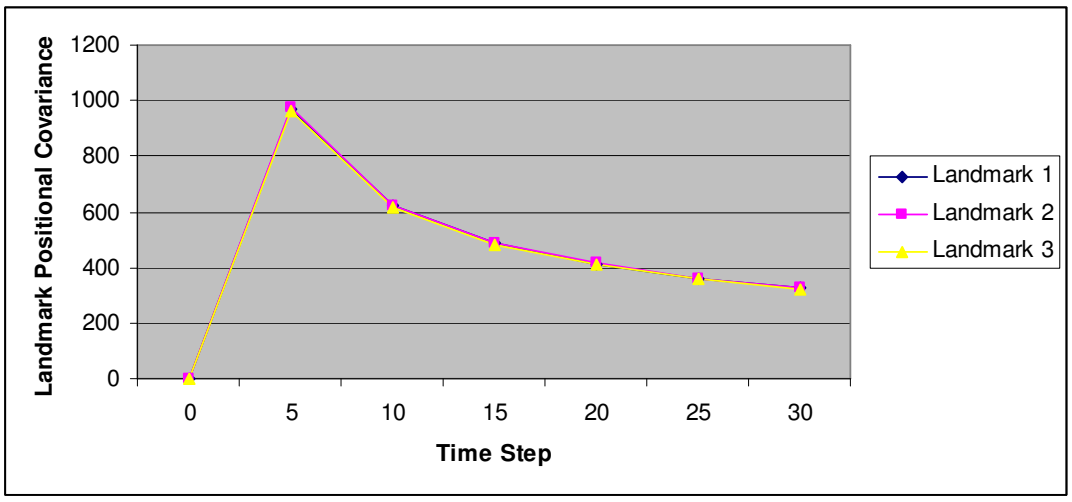

Figure 7-26 Landmark Positional Covariance Variation of Experiment 2 (mm^3)

The landmark positional uncertainties decrease with time by reobserving them as depicted in Figure 7-26.

Experiment 3: Backward and Forward Motion with Slippage Introduced

In this experiment the robot is again placed in front of the white board with black square and forced to move forward or backward at each time step. The distance request to travel at one step is 100 mm. At specific time steps, we have read the estimated results by our system, odometry values and the ground truth values. The difference of this experiment from the Experiment 2 is that after 15th time step we introduce 200 mm slippage to the system by simply carrying the robot 200 mm forward without rotating the wheels. In this way we can simulate as if the system runs long period so that the slippage errors are accumulated to 200 mm. Also this

experiment tests the system performance, when a sudden disturbance is applied to the system such as a collision with an obstacle and slips some distance as a result.

The test results for this experiment are given below:

Initialization:

Figure 7-27 Camera Images for Initialization of Experiment 3

Table 7-12 Real Landmark Positions for Experiment 3

	X (mm)	Y (mm)	Z (mm)
Landmark 1	185	130	2050
Landmark 2	10	-50	2000
Landmark 3	-195	160	2050

Table 7-13 Estimated Landmark Positions at Initialization for Experiment 3

	X (mm)	Y (mm)	Z (mm)
Landmark 1	189.90	150.88	1965.47
Landmark 2	26.93	-20.20	2007.19
Landmark 3	-181.87	181.99	2001.69

5th Step (After 5 Movements Forward):

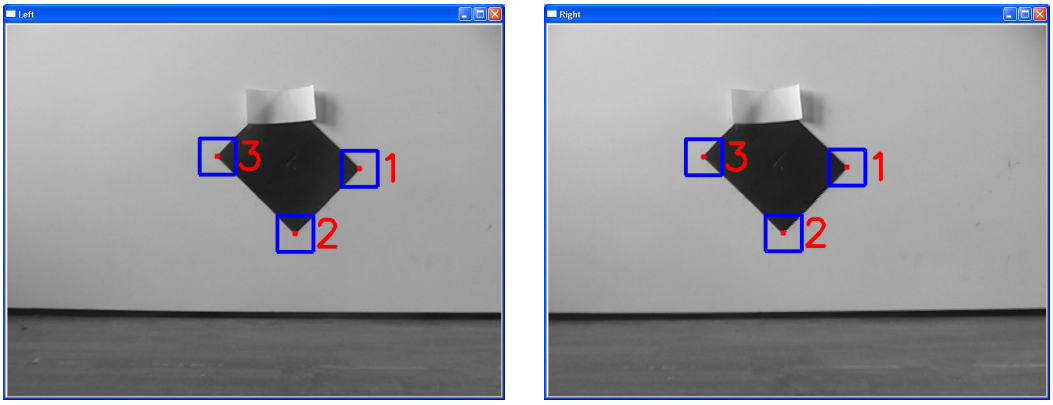

Figure 7-28 Camera Images for 5th Step of Experiment 3

Table 7-14 Estimated Landmark Positions at 5th Step for Experiment 3

	X (mm)	Y (mm)	Z (mm)
Landmark 1	191.55	128.87	1992.31
Landmark 2	25.32	-43.82	2010.29
Landmark 3	-181.98	155.14	1968.32

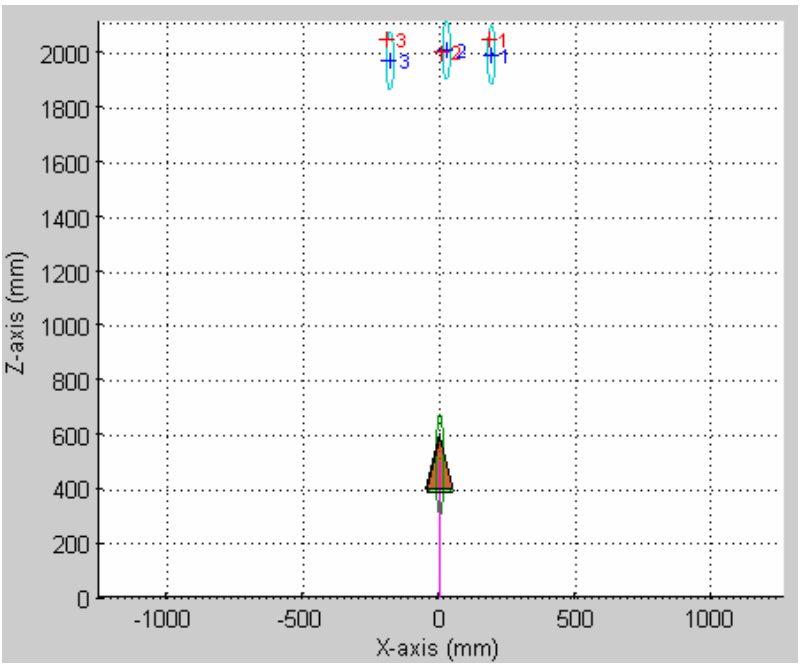

Figure 7-29 Predicted 2D Map for 5th Step of Experiment 3

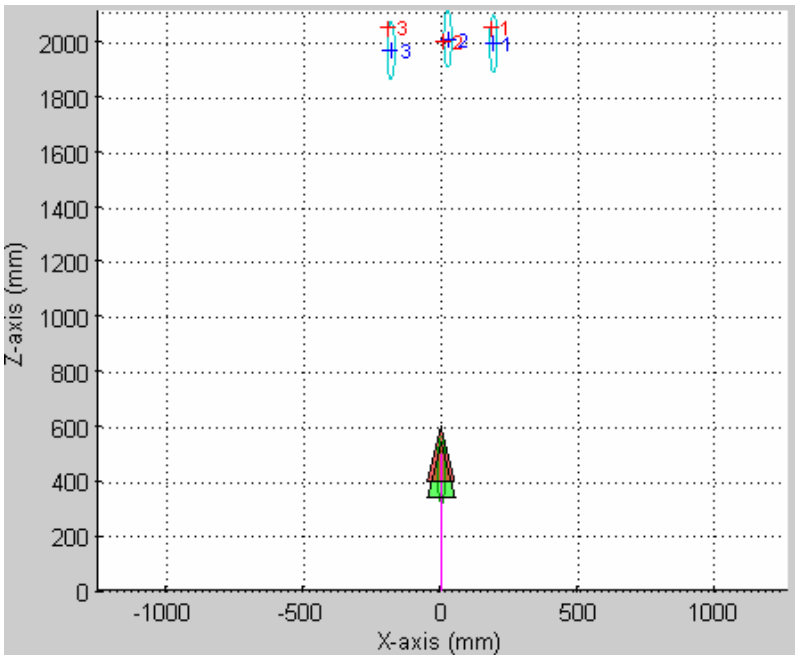

Figure 7-30 Corrected 2D Map for 5th Step of Experiment 3

10th Step (After 5 Movements Forward):

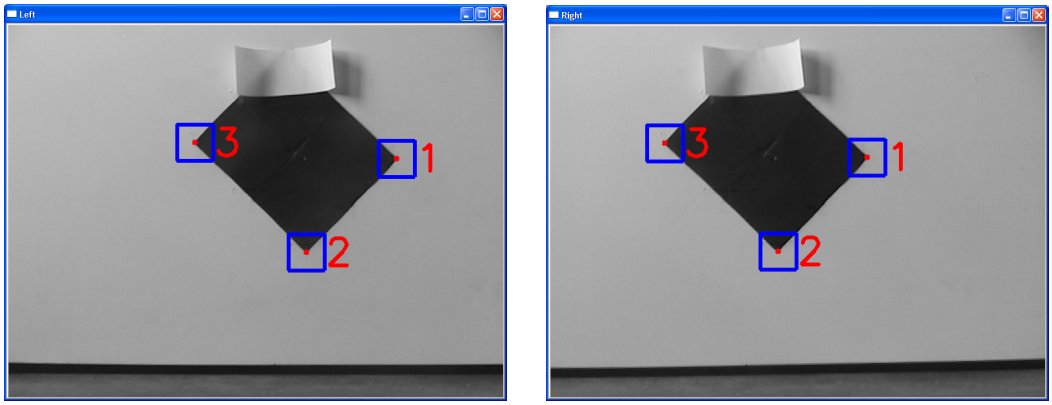

Figure 7-31 Camera Images for 10th Step of Experiment 3

Table 7-15 Estimated Landmark Positions at 10th Step for Experiment 3

	X (mm)	Y (mm)	Z (mm)
Landmark 1	193.35	124.72	1995.60
Landmark 2	25.09	-49.75	2004.28
Landmark 3	-183.68	150.49	1971.07

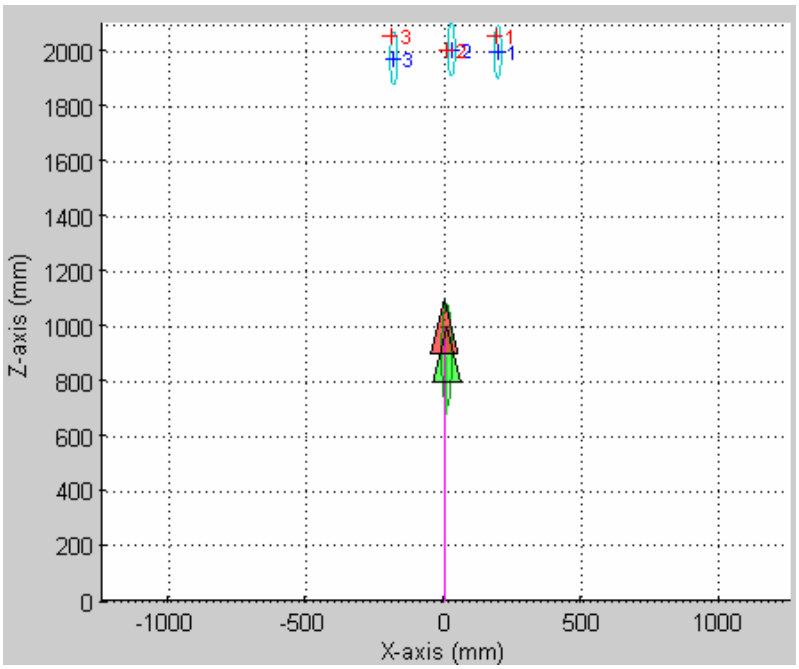

Figure 7-32 Predicted 2D Map for 10th Step of Experiment 3

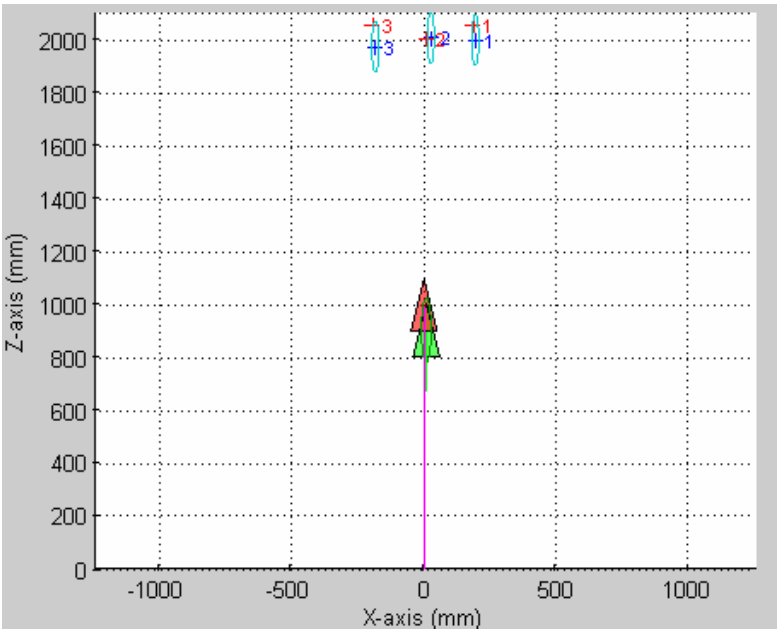

Figure 7-33 Corrected 2D Map for 10th Step of Experiment 3

15th Step (After 5 Movements Backwards):

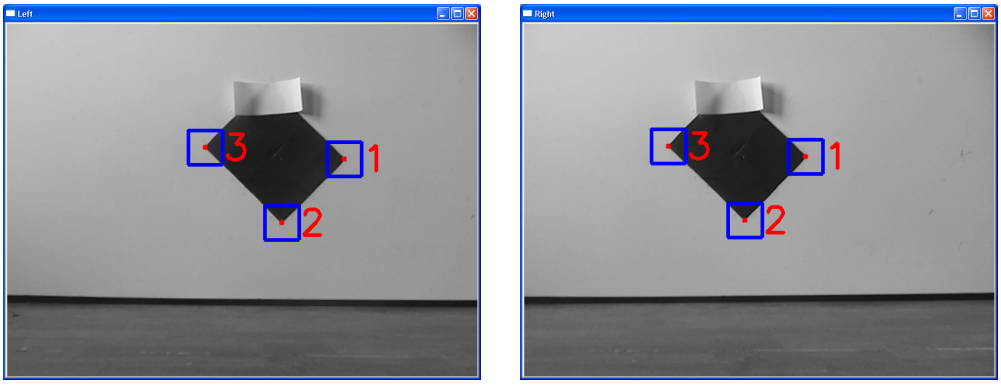

Figure 7-34 Camera Images for 15th Step of Experiment 3

Table 7-16 Estimated Landmark Positions at 15th Step for Experiment 3

	X (mm)	Y (mm)	Z (mm)
Landmark 1	192.15	122.10	1988.36
Landmark 2	25.90	-51.11	2001.17
Landmark 3	-183.23	149.13	1981.42

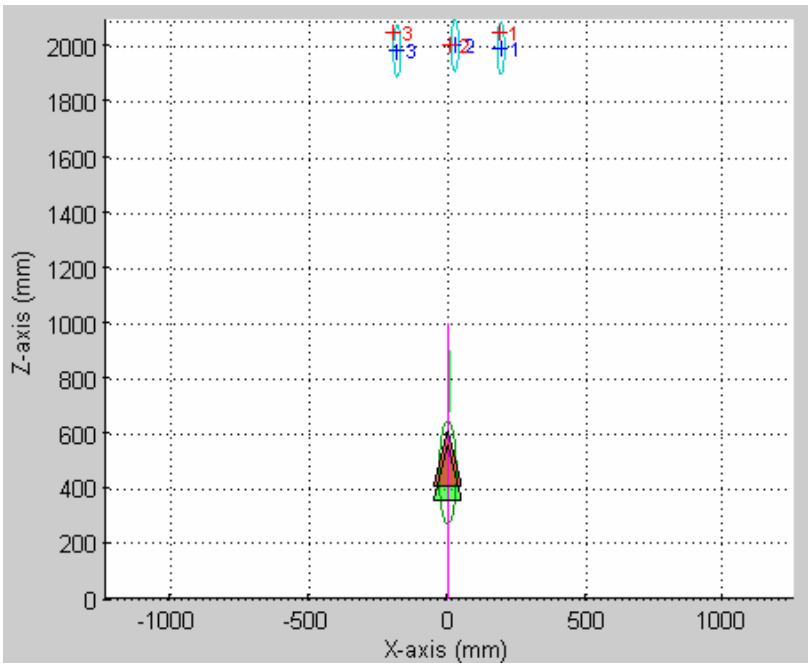

Figure 7-35 Predicted 2D Map for 15th Step of Experiment 3

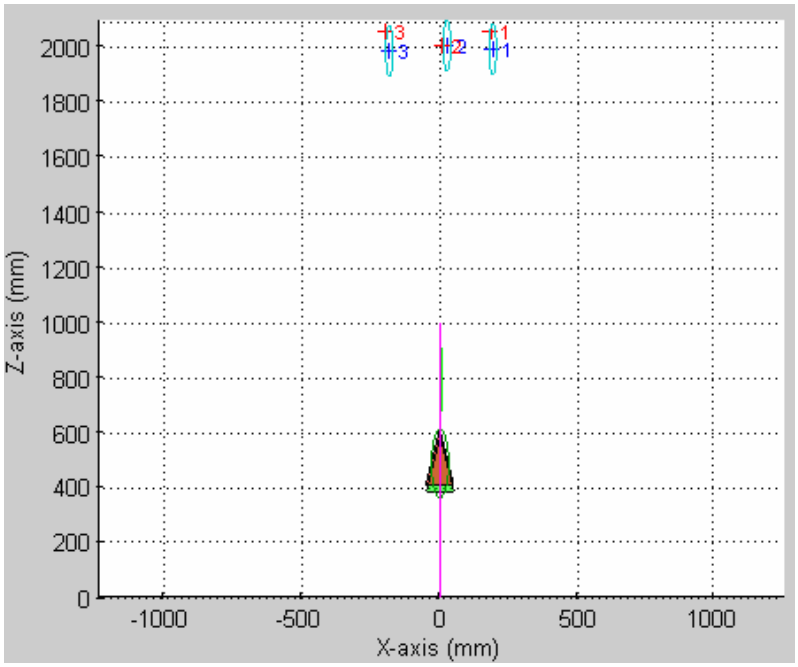

Figure 7-36 Corrected 2D Map for 15th Step of Experiment 3

At the end of 15th step the robot is carried 200 mm forward to introduce slippage.

20th Step (After 5 Movements Backward):

Figure 7-37 Camera Images for 20th Step of Experiment 3

Table 7-17 Estimated Landmark Positions at 20th Step for Experiment 3

	X (mm)	Y (mm)	Z (mm)
Landmark 1	191.65	122.76	1983.72
Landmark 2	26.08	-50.06	2001.55
Landmark 3	-182.94	150.42	1985.68

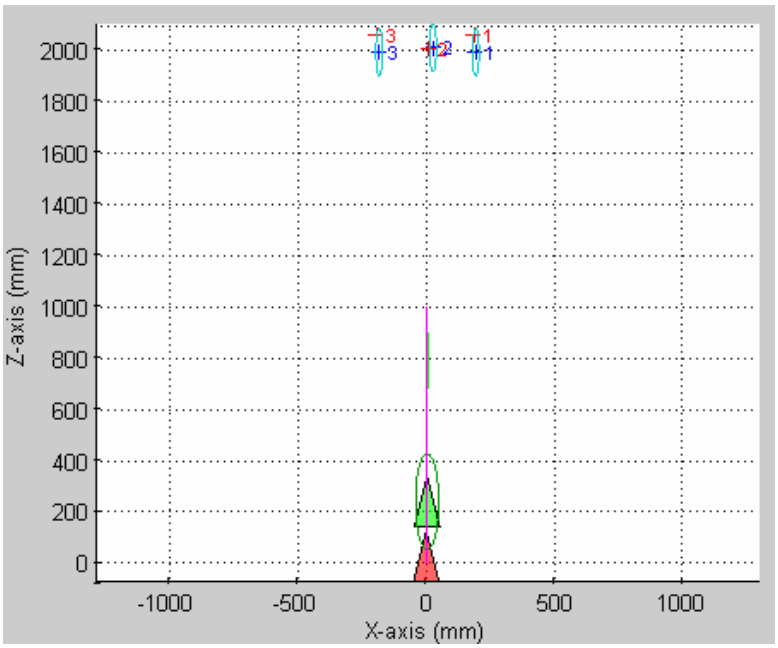

Figure 7-38 Predicted 2D Map for 20th Step of Experiment 3

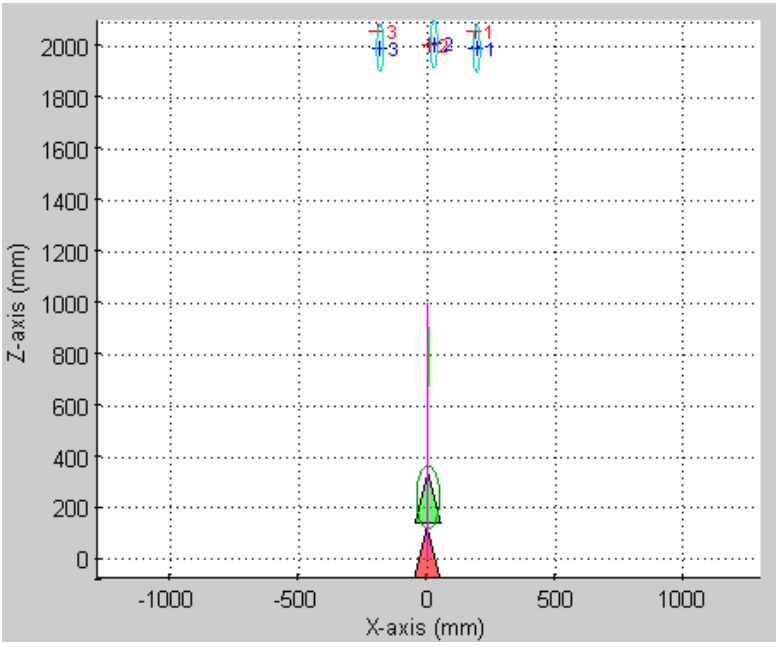

Figure 7-39 Corrected 2D Map for 20th Step of Experiment 3

25th Step (After 5 Movements Backward):

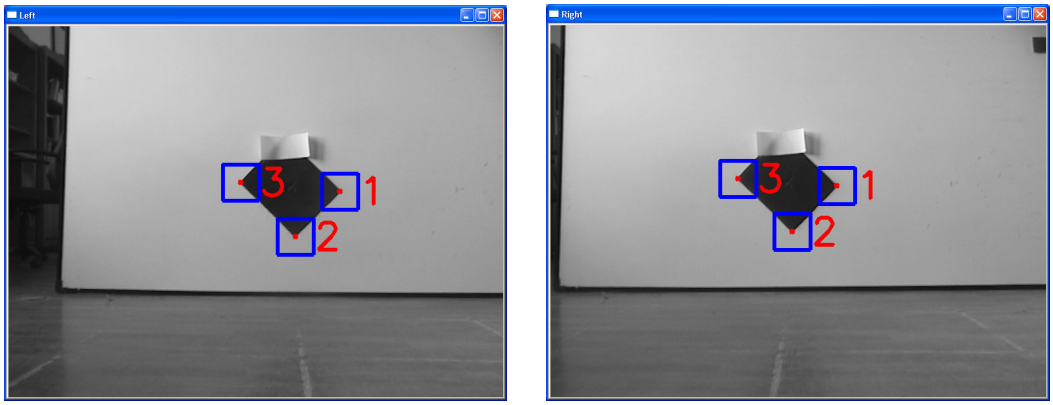

Figure 7-40 Camera Images for 25th Step of Experiment 3

Table 7-18 Estimated Landmark Positions at 25th Step for Experiment 3

	X (mm)	Y (mm)	Z (mm)
Landmark 1	190.78	122.08	1980.18
Landmark 2	26.03	-49.69	2001.80
Landmark 3	-182.19	150.39	1988.97

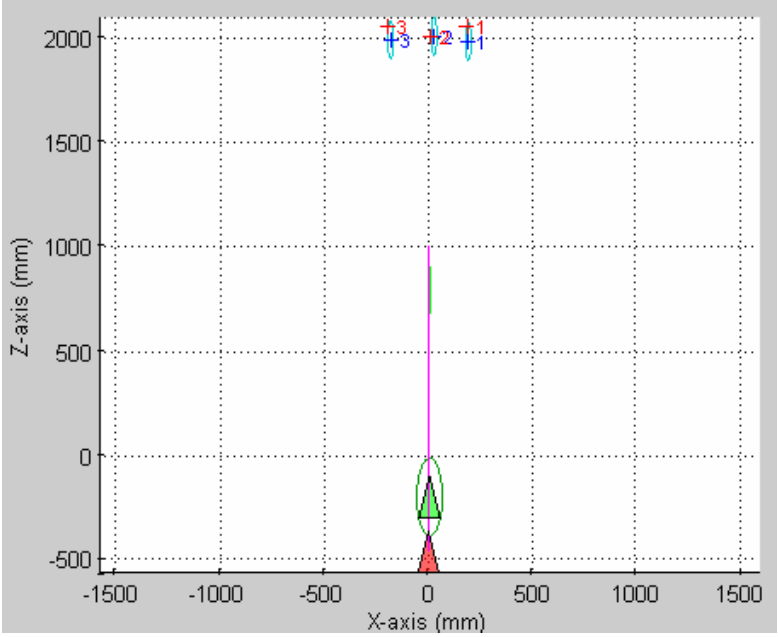

Figure 7-41 Predicted 2D Map for 25th Step of Experiment 3

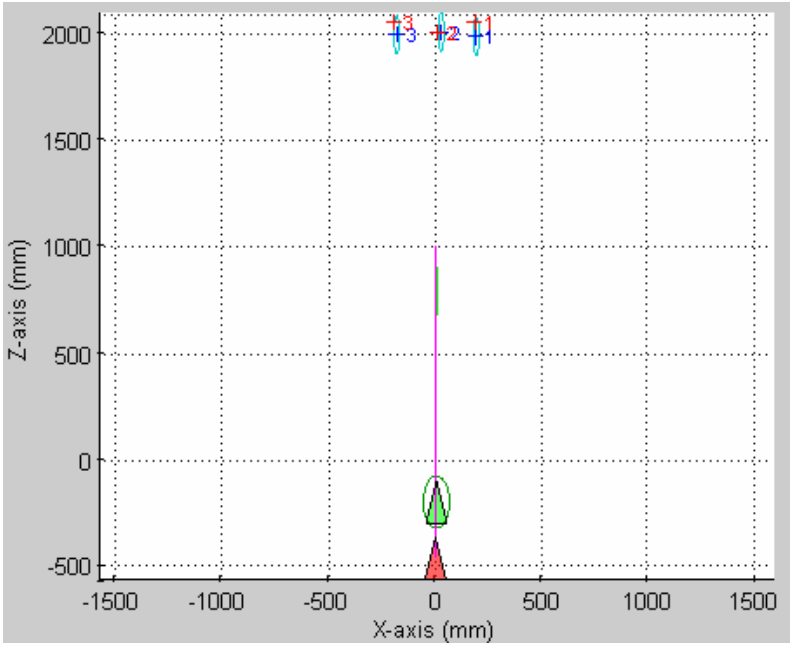

Figure 7-42 Corrected 2D Map for 25th Step of Experiment 3

30th Step (After 5 Movements Forward):

Figure 7-43 Camera Images for 30th Step of Experiment 3

Table 7-19 Estimated Landmark Positions at 30th Step for Experiment 3

	X (mm)	Y (mm)	Z (mm)
Landmark 1	191.66	123.34	1983.56
Landmark 2	25.86	49.16	2002.26
Landmark 3	-182.72	151.27	1985.13

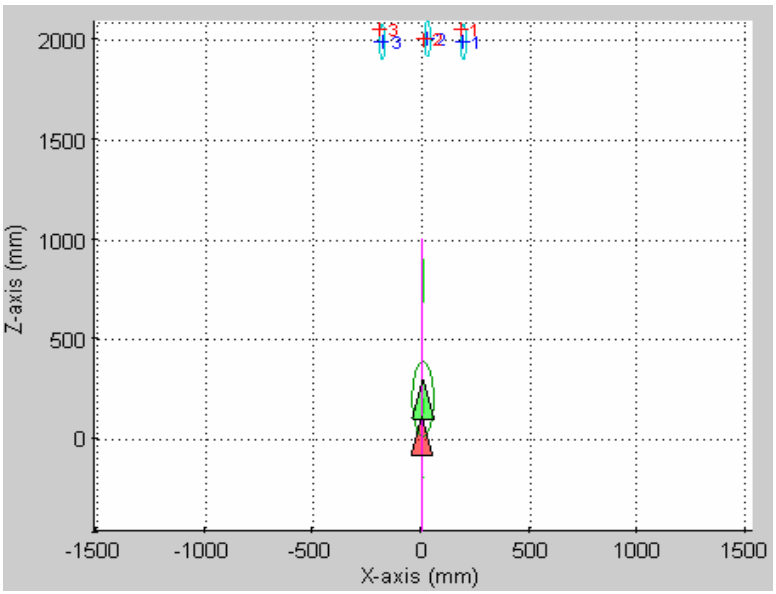

Figure 7-44 Predicted 2D Map for 30th Step of Experiment 3

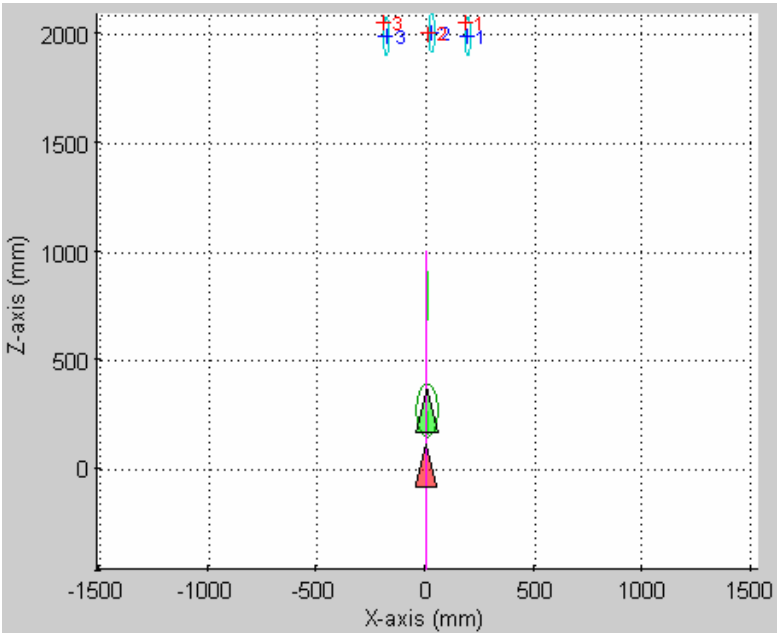

Figure 7-45 Corrected 2D Map for 30th Step of Experiment 3

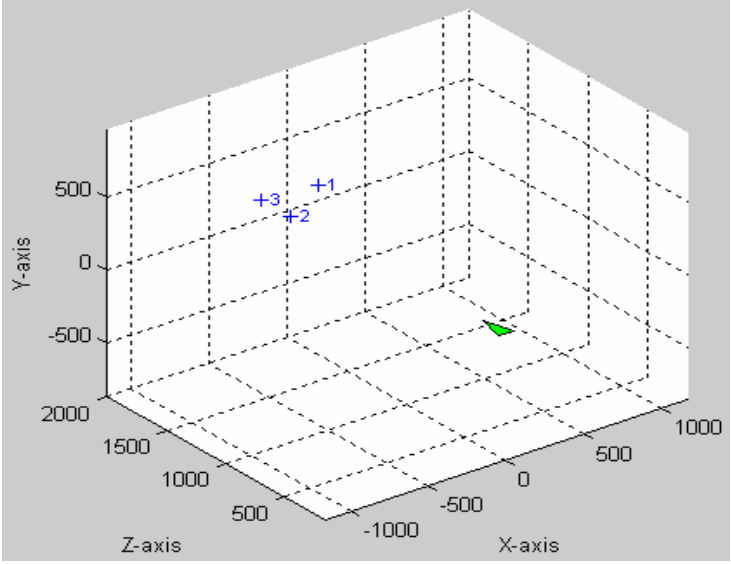

Figure 7-46 Estimated 3D Map of Experiment 3

After final step, we have obtained really impressing results which are so close to the real values, whereas odometry value for z-axis position of robot excessively deviates from the actual value. Our state prediction and correction algorithm has easily recovered from this error introduced to the system, after few time steps.

Table 7-20 Robot Positional Values for Experiment 3

Time Step	**Estimated (mm)**	**Odometry (mm)**	**Ground Truth (mm)**
0	0	0	0
5	440.40	498.58	470
10	900.71	996.68	940
15	485.19	510.22	490
20	240.25	21.34	260
25	-201.94	-468.51	-205
30	264.87	9.70	260

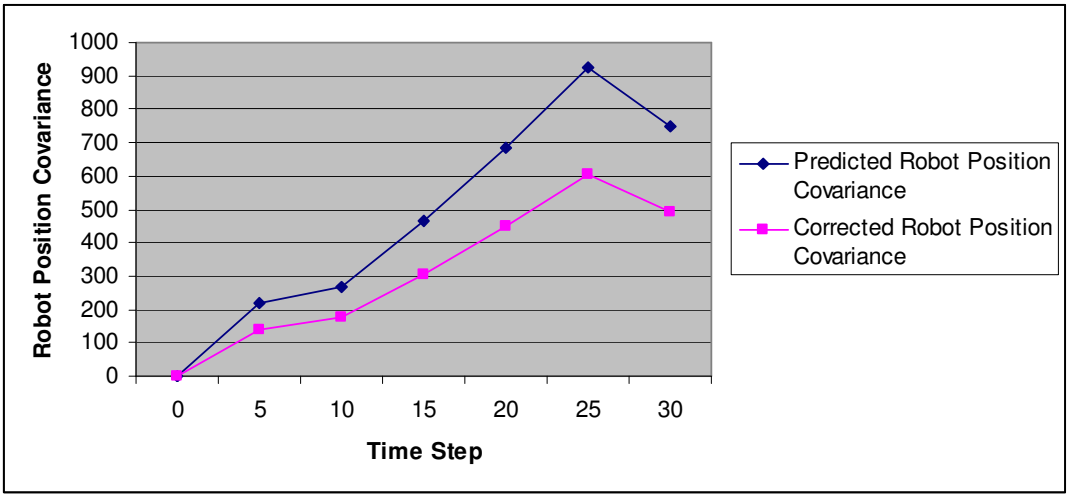

Figure 7-47 Robot Positional Covariance Variation of Experiment 3 (mm^2)

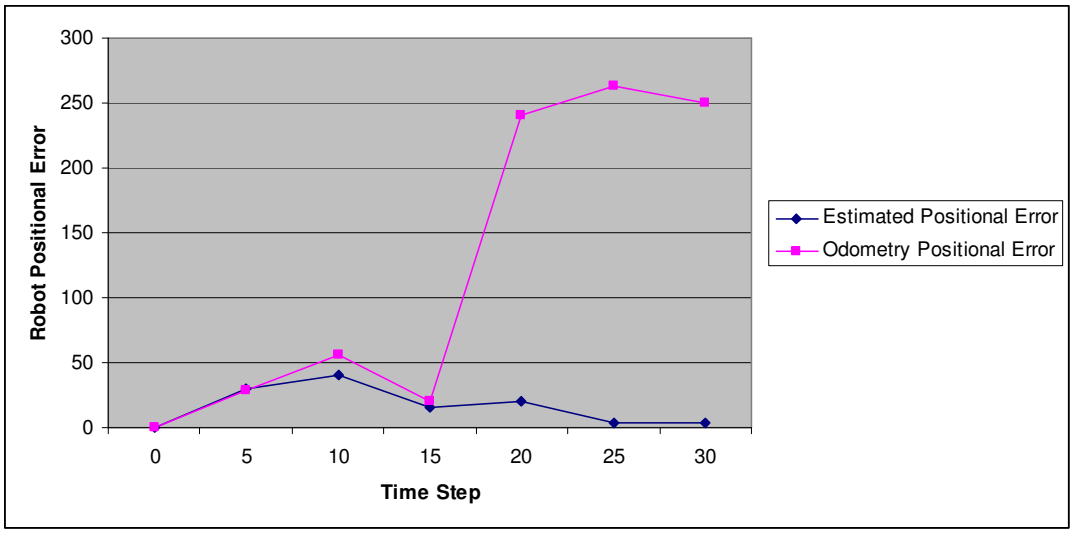

Figure 7-48 Robot Positional Error Variation of Experiment 3 (mm)

The robot odometry positional greatly deviates from the ground truth, after the slippage has been introduced to the sytem, whereas our SLAM system still estimated the robot position with little error.

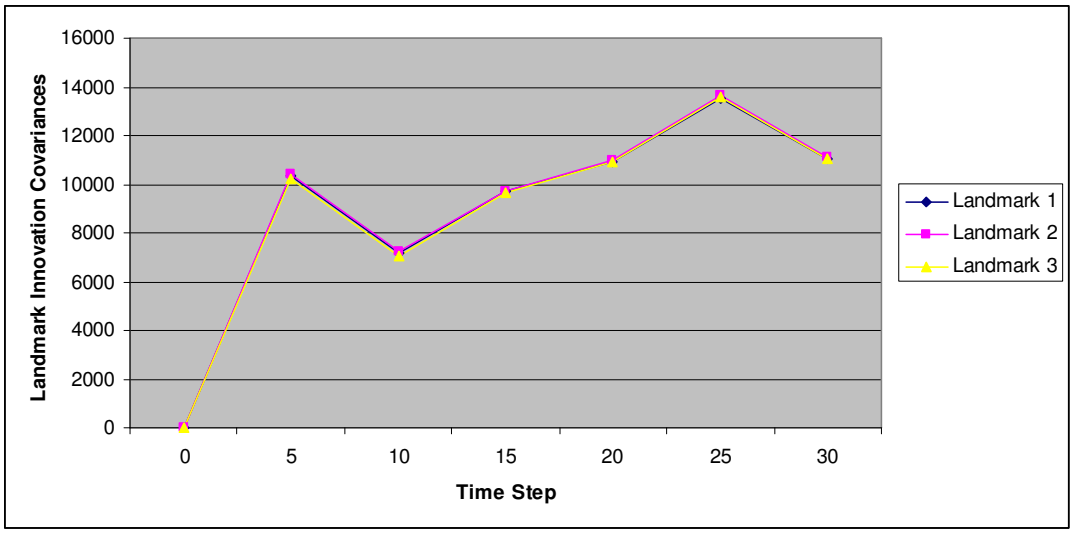

Figure 7-49 Landmark Innovation Covariance Variation of Experiment 3 (mm^3)

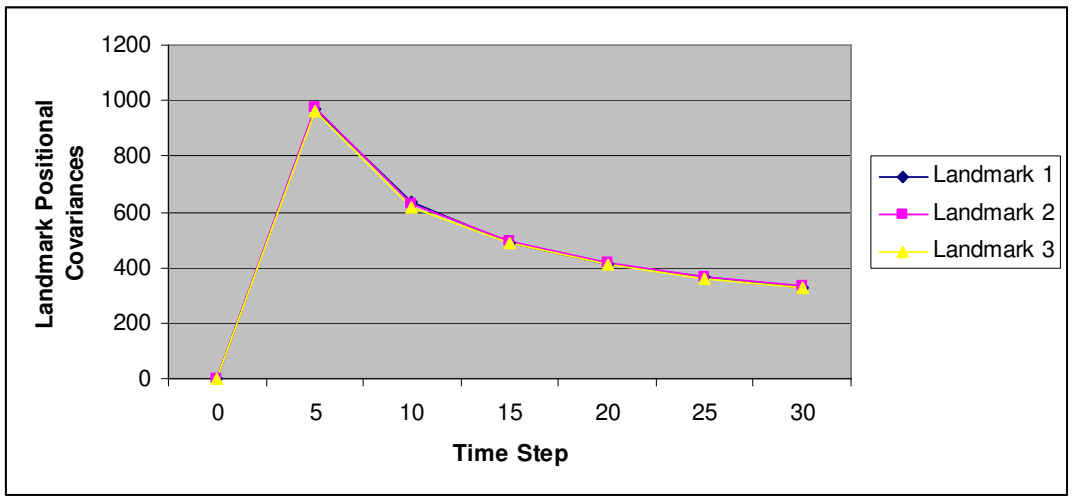

Figure 7-50 Landmark Positional Covariance Variation of Experiment 3 (mm^3)

Experiment 4: Backward and Forward Motion in Mono Mode

In this experiment the Experiment 2 is repeated in Mono SLAM mode.

Initialization:

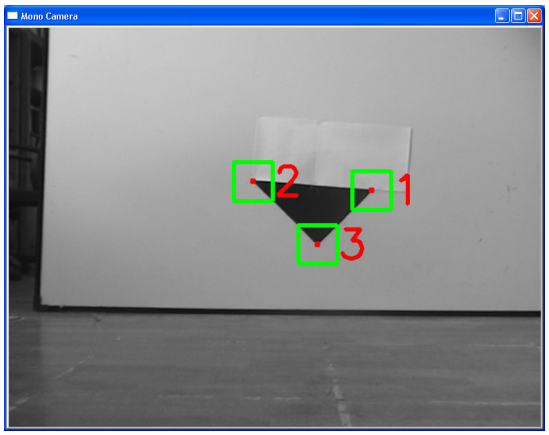

Figure 7-51 Camera Image for Initialization of Experiment 4

Table 7-21 Real Landmark Positions

	X (mm)	Y (mm)	Z (mm)
Landmark 1	185	130	2050
Landmark 2	-195	160	2050
Landmark 3	10	-50	2000

Table 7-22 Estimated Landmark Positions

	X (mm)	Y (mm)	Z (mm)
Landmark 1	259.93	132.64	1962.38
Landmark 2	-112.93	167.25	2024.78
Landmark 3	91.20	-36.49	1971.76

5th Step (After 5 Movements Forward):

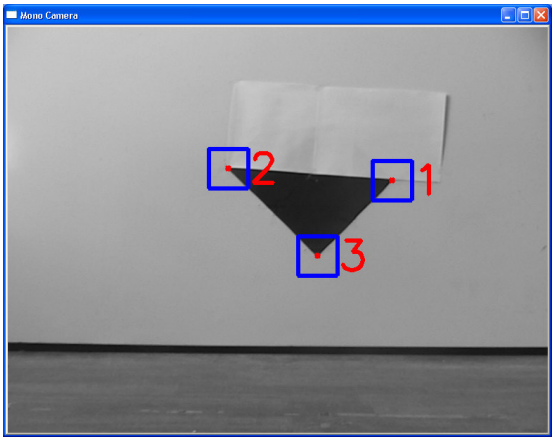

Figure 7-52 Camera Image for 5th Step of Experiment 4

Table 7-23 Estimated Landmark Positions at 5th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	264.38	131.89	1970.24
Landmark 2	-117.38	159.67	1972.66
Landmark 3	89.76	-41.11	1958.01

Since our static camera construction does not allow smooth mono operation, the robot continuously changes its x coordinate also in order to create motion parallax for depth measurements. Therefore the robot paths have the form shown in Figure 7-53.

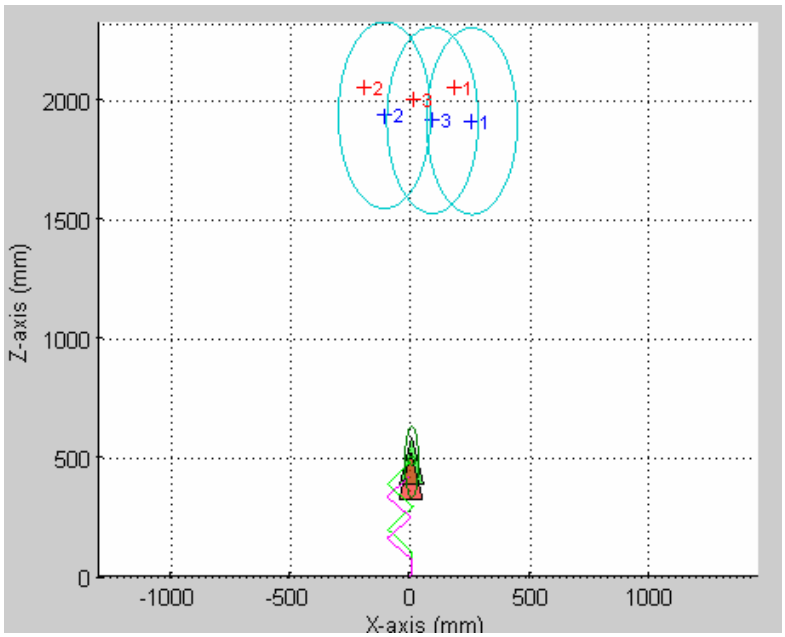

Figure 7-53 Predicted 2D Map for 5th Step of Experiment 4

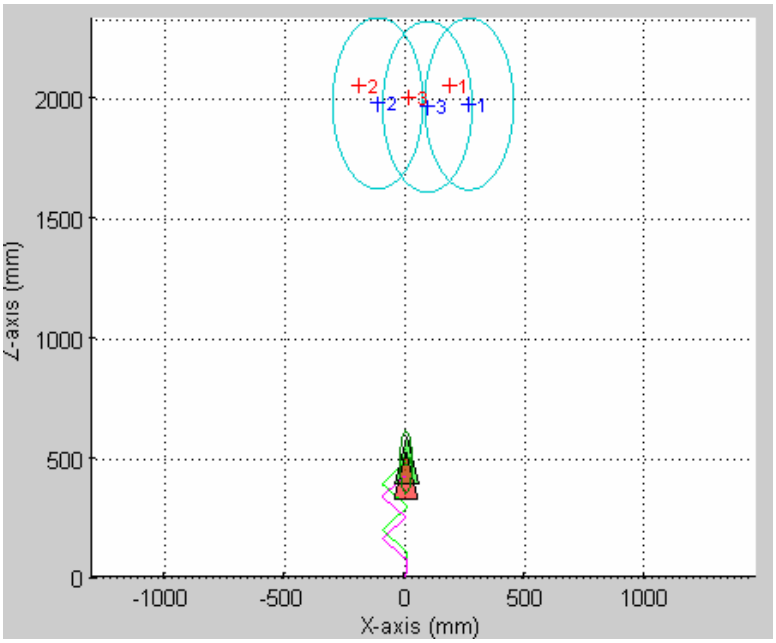

Figure 7-54 Corrected 2D Map for 5th Step of Experiment 4

10th Step (After 5 Movements Forward):

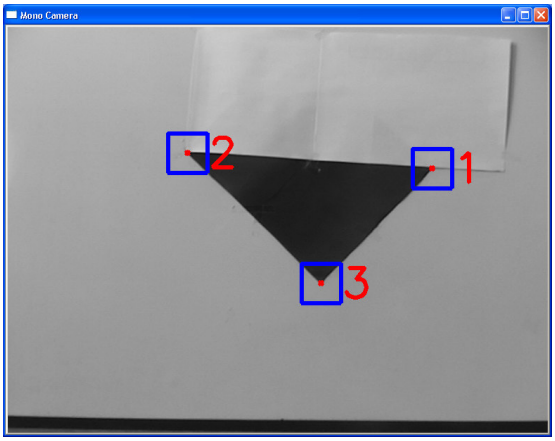

Figure 7-55 Camera Image for 10th Step of Experiment 4

Table 7-24 Estimated Landmark Positions at 10th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	269.20	126.42	1955.92
Landmark 2	-120.06	155.13	1978.22
Landmark 3	87.57	-47.83	1950.58

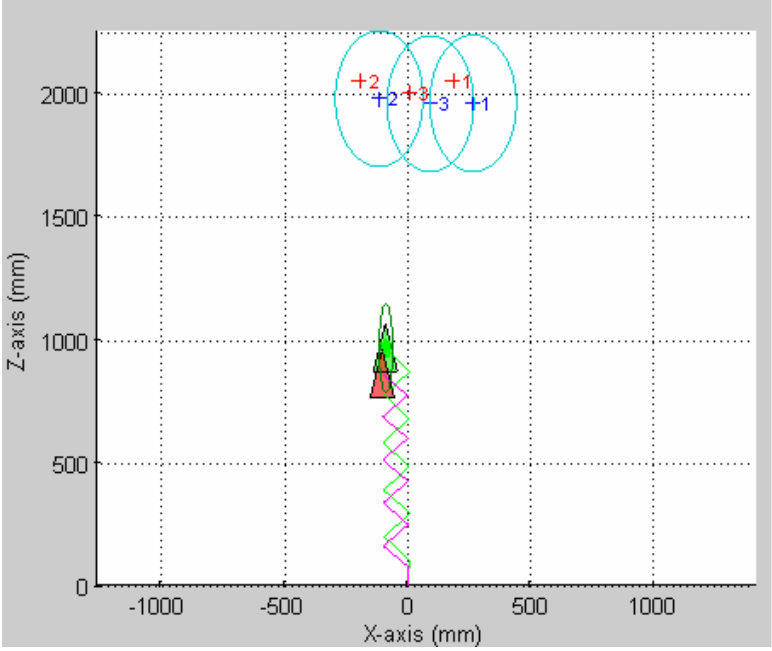

Figure 7-56 Predicted 2D Map for 10th Step of Experiment 4

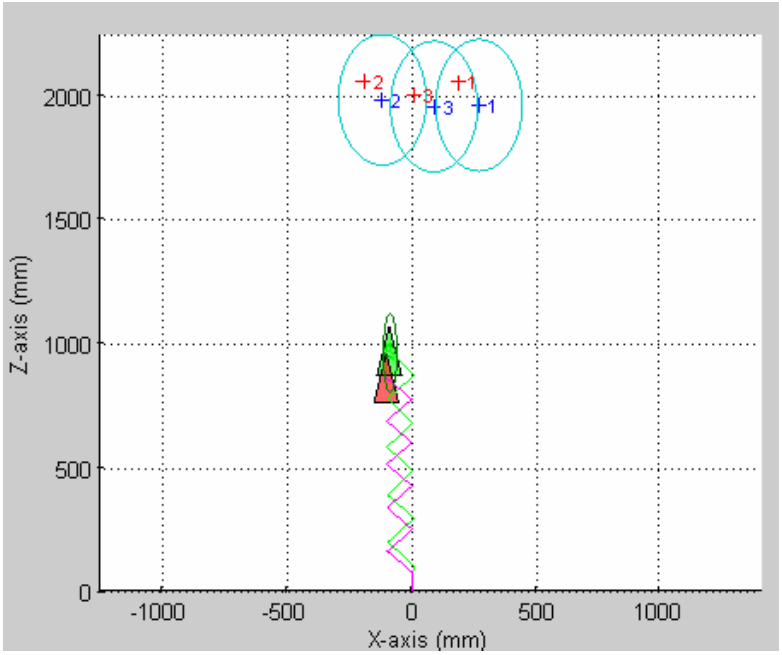

Figure 7-57 Corrected 2D Map for 5th Step of Experiment 4

15th Step (After 5 Movements Backward):

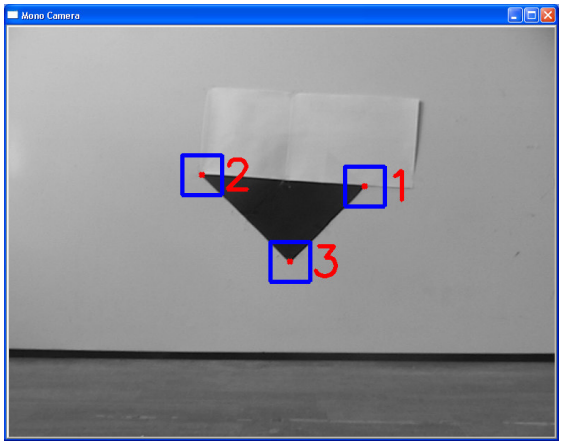

Figure 7-58 Camera Image for 5th Step of Experiment 4

Table 7-25 Estimated Landmark Positions at 15th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	264.90	122.58	1896.55
Landmark 2	-112.40	150.87	1924.59
Landmark 3	90.52	-47.88	1895.22

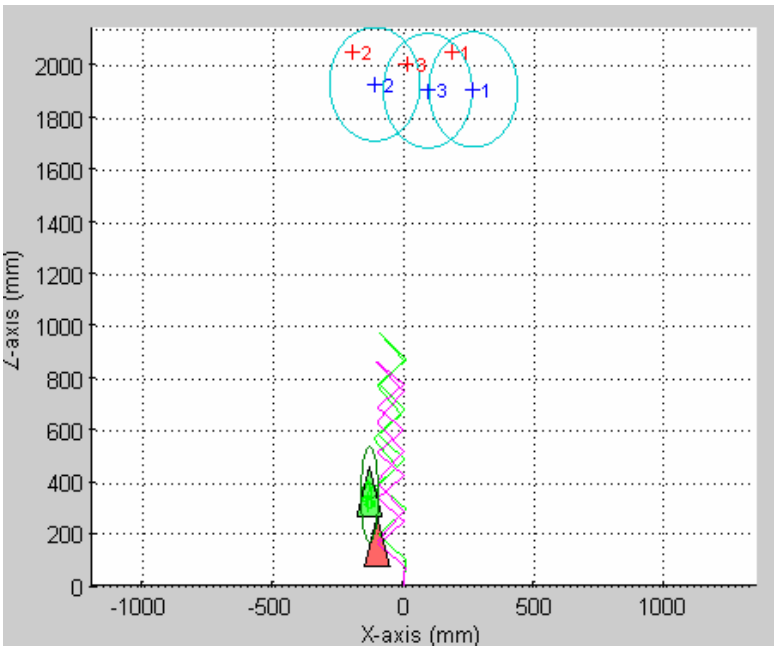

Figure 7-59 Predicted 2D Map for 15th Step of Experiment 4

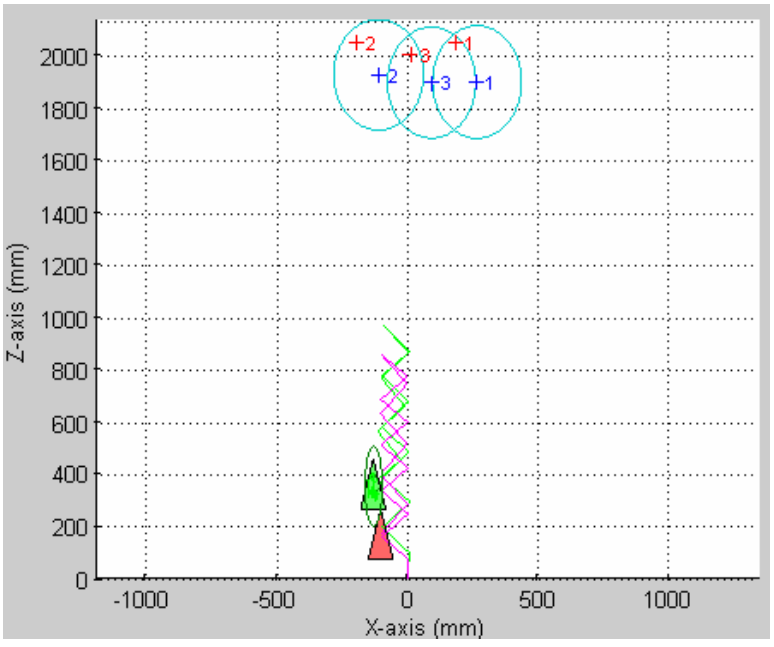

Figure 7-60 Corrected 2D Map for 5th Step of Experiment 4

20th Step (After 5 Movements Backward):

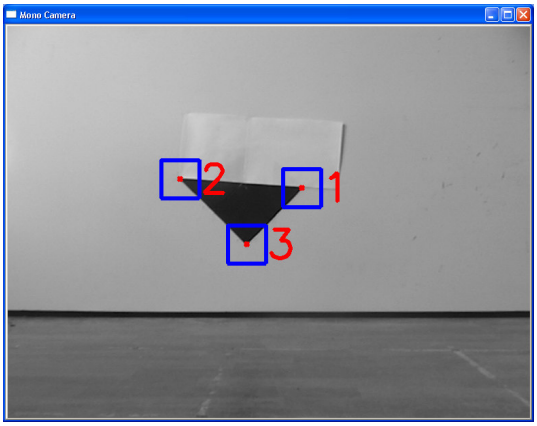

Figure 7-61 Camera Image for 20th Step of Experiment 4

Table 7-26 Estimated Landmark Positions at 20th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	253.98	117.17	1778.32
Landmark 2	-102.97	144.39	1802.88
Landmark 3	89.62	-44.45	1778.39

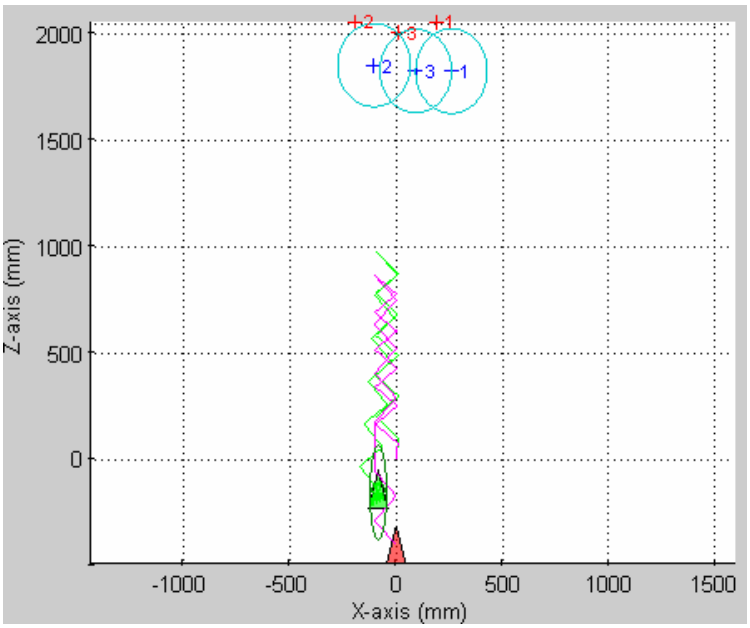

Figure 7-62 Predicted 2D Map for 20th Step of Experiment 4

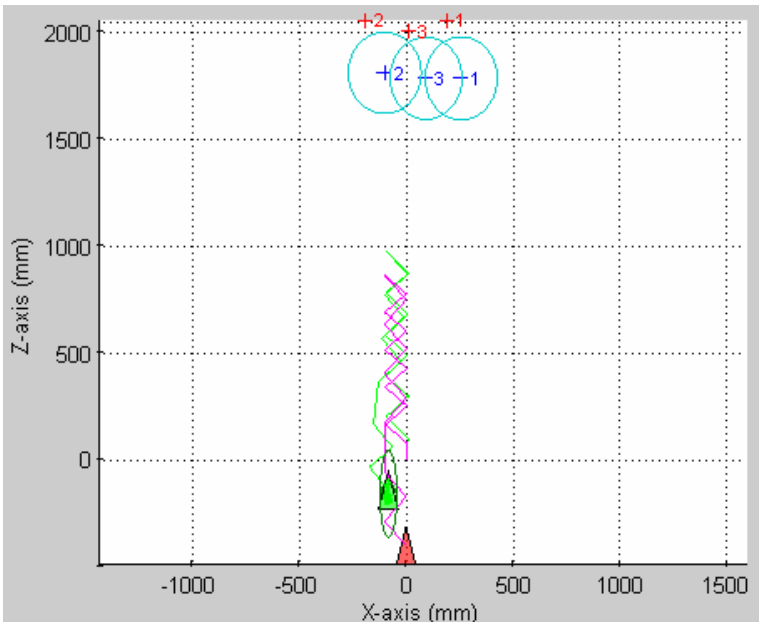

Figure 7-63 Corrected 2D Map for 20th Step of Experiment 4

25th Step (After 5 Movements Backward):

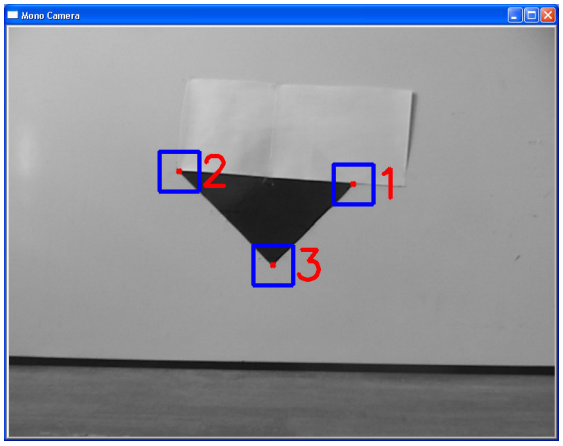

Figure 7-64 Camera Image for 25th Step of Experiment 4

Table 7-27 Estimated Landmark Positions at 25th Step for Experiment 2

	X (mm)	Y (mm)	Z (mm)
Landmark 1	258.36	121.40	1779.22
Landmark 2	-107.50	150.06	1821.34
Landmark 3	90.17	-44.85	1787.49

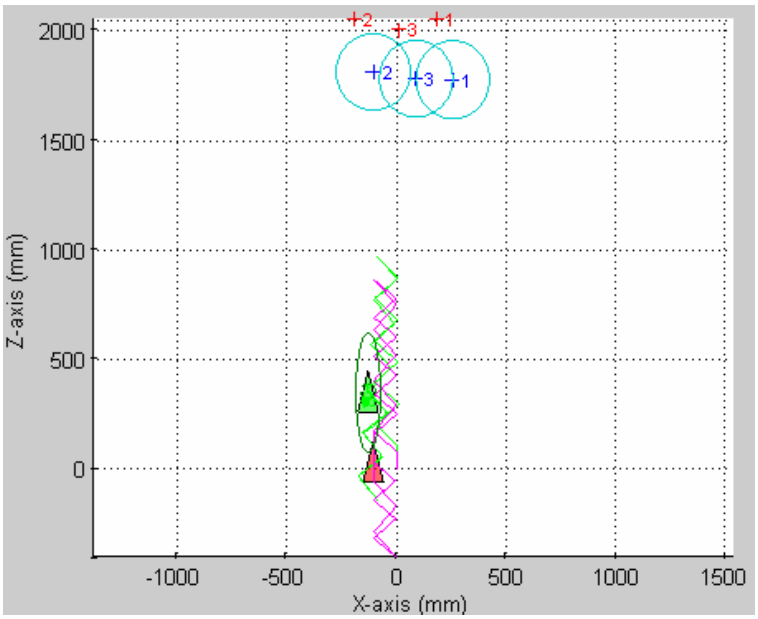

Figure 7-65 Predicted 2D Map for 25th Step of Experiment 4

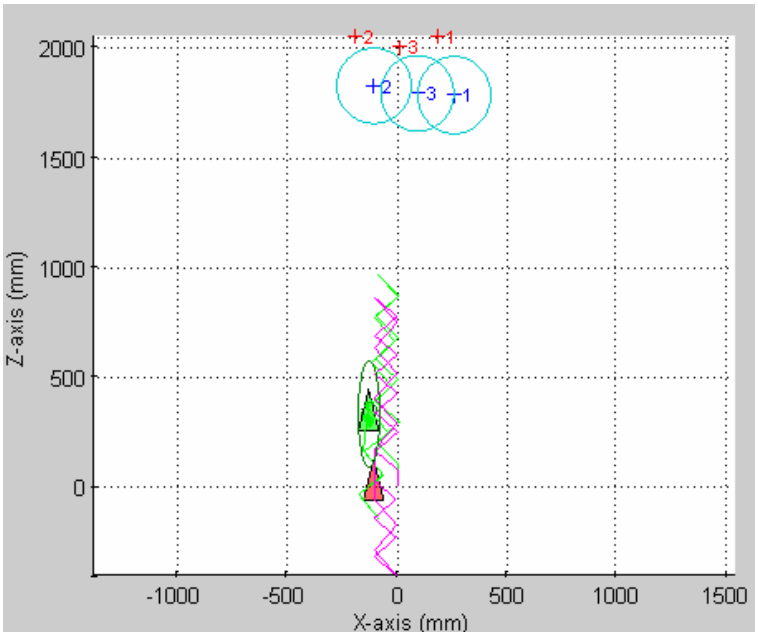

Figure 7-66 Corrected 2D Map for 25th Step of Experiment 4

After final step, it is seen that the estimated robot position is close to the real one, whereas the odometry reading deviates some amount from the actual position even though the motion of the robot on the laboratory surface is regular and not much slippage is encountered. In the last time steps the system estimates the landmark positions over confidently, so that the real landmark positions are outside the uncertainty ellipses of the estimated values.

Table 7-28 Robot Positional Values for Experiment 4

Time Step	Estimated (mm)	Odometry (mm)	Ground Truth (mm)
0	0.00	-14.55	0
5	485.23	423.89	530
10	968.30	858.94	1045
15	362.64	169.75	520
20	-144.48	-404.49	90
25	349.08	29.58	630

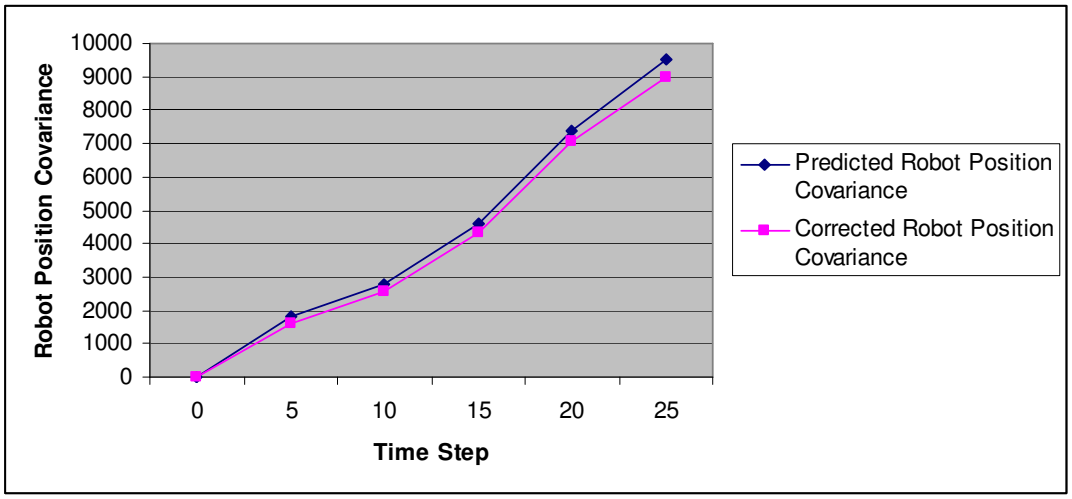

Figure 7-67 Robot Position Covariance Variation of Experiment 4 (mm^2)

The robot positional uncertainty increases with time as shown in Figure 7-67. The increasing measured robot positional error in Figure 7-68 justifies this situation.

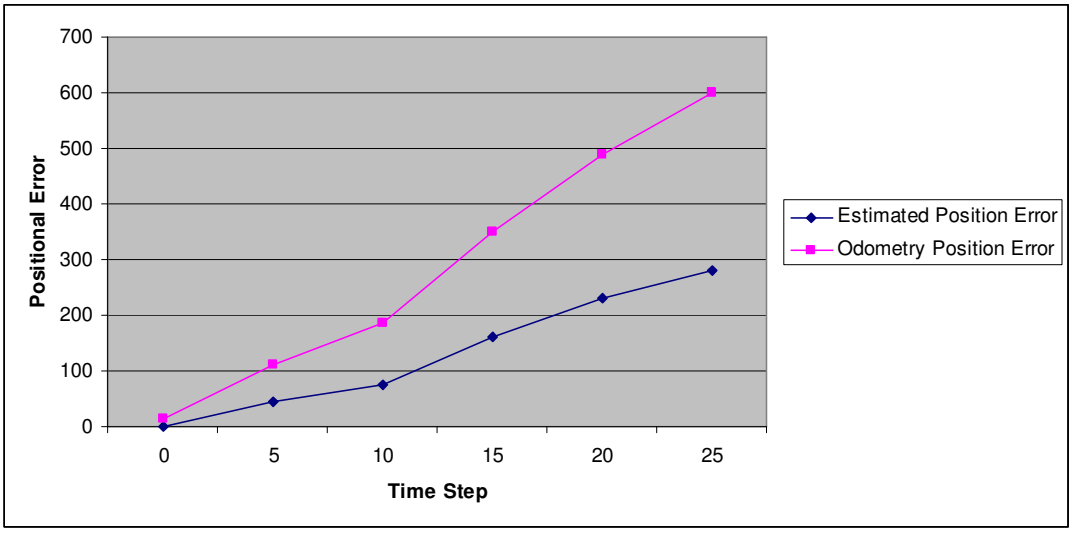

Figure 7-68 Robot Positional Error Variation of Experiment 4 (mm)

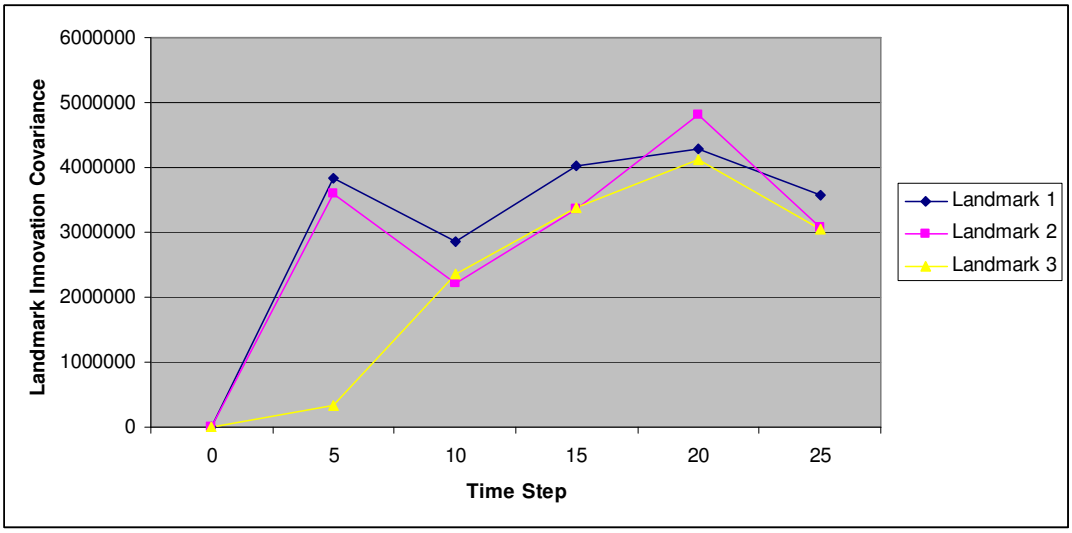

Figure 7-69 Landmark Innovation Covariance Variation of Experiment 4 (mm^3)

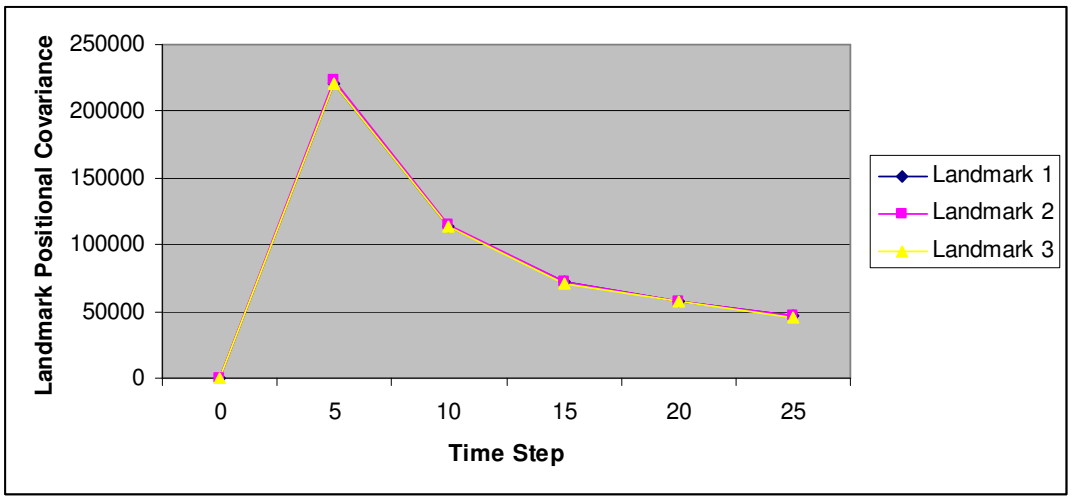

Figure 7-70 Landmark Positional Covariance Variation of Experiment 4 (mm^3)

Experiment 5: A Complete Run of the SLAM algorithm by Closing Loop

In this experiment, the robot travels a loop in the laboratory by moving forward by 100 mm requests and turning the corners by 90º. At specific time steps, we have read the estimated results by our system, odometry values and the ground truth values. Since this experiment is a full run of our algorithm by making the robot go around the laboratory and build its map, its value is high. Therefore, we will show each important steps of this experiment to get insight of our SLAM system.

The test results for this experiment are given below:

Initialization:

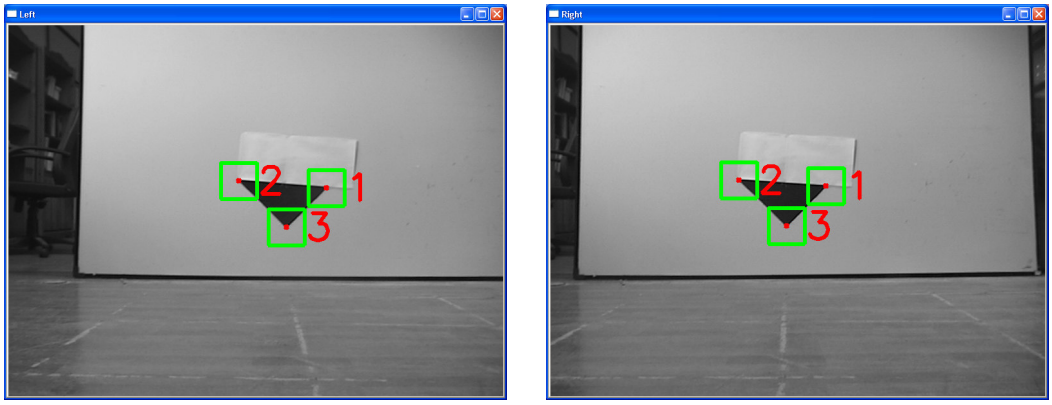

Figure 7-71 Camera Images for Initialization of Experiment 5

Table 7-29 Position Values for Experiment 5

	Estimated (mm)			Odometry (mm)			Ground Truth (mm)		
Time Step	**X**	**Z**	**Φ**	**X**	**Z**	**Φ**	**X**	**Z**	**Φ**
0	0	0	0	0	0	0	0	0	0
10	-2.04	905.37	0.92	0.48	995.22	0.00	0	950	0
12	167.16	904.87	89.76	99.91	987.46	99.05	167.16	904.87	89.76
19	815.05	879.49	91.68	795.89	876.88	99.05	810	910	92
21	800.06	696.30	-177.37	767.76	772.60	-161.72	790	720	-175
25	776.65	299.10	-177.18	640.69	387.03	-161.81	720	330	-175
36	159.61	-70.27	-87.15	-15.52	267.72	-62.93	110	70	-85
38	65.54	-64.46	3.71	-110.10	317.19	35.68	75.54	100	5

Table 7-30 Estimated Landmark Positions at Initialization for Experiment 5

	X (mm)	Y (mm)	Z (mm)
Landmark 1	187.78	139.65	2522.93
Landmark 2	-190.89	167.74	2534.06
Landmark 3	17.17	-31.19	2539.40

Table 7-31 Real Landmark Positions at Initialization for Experiment 5

	X (mm)	Y (mm)	Z (mm)
Landmark 1	185	130	2550
Landmark 2	-195	160	2550
Landmark 3	10	-50	2500
Landmark 4	4200	800	1750
Landmark 5	4200	-220	1800
Landmark 6	2500	-35	1050
Landmark 7	3500	-200	1950
Landmark 8	2570	125	700
Landmark 9	100	1000	-3000
Landmark 10	100	900	-3100
Landmark 11	-50	900	-3100
Landmark 11	-1850	500	600

10th Step (After 10 Movements Forward):

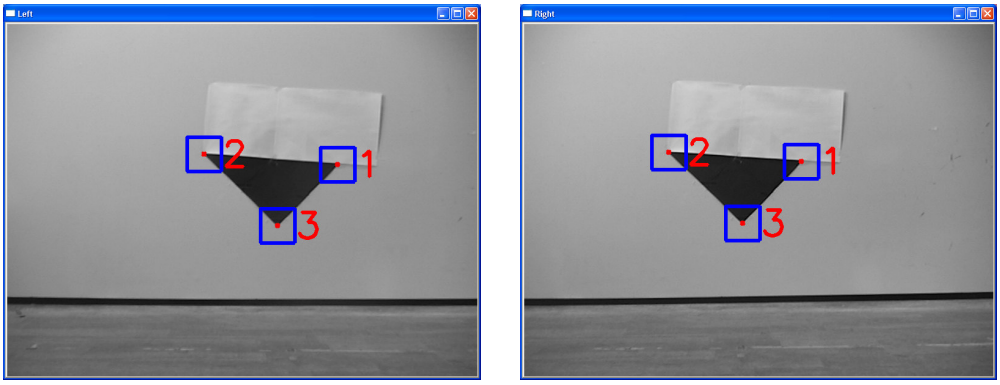

Figure 7-72 Camera Images for 10th Step of Experiment 5

Table 7-32 Estimated Landmark Positions at 5th Step for Experiment 5

	X (mm)	Y (mm)	Z (mm)
Landmark 1	191.77	133.04	2531.16
Landmark 2	-194.59	156.08	2468.63
Landmark 3	16.59	-44.98	2533.02

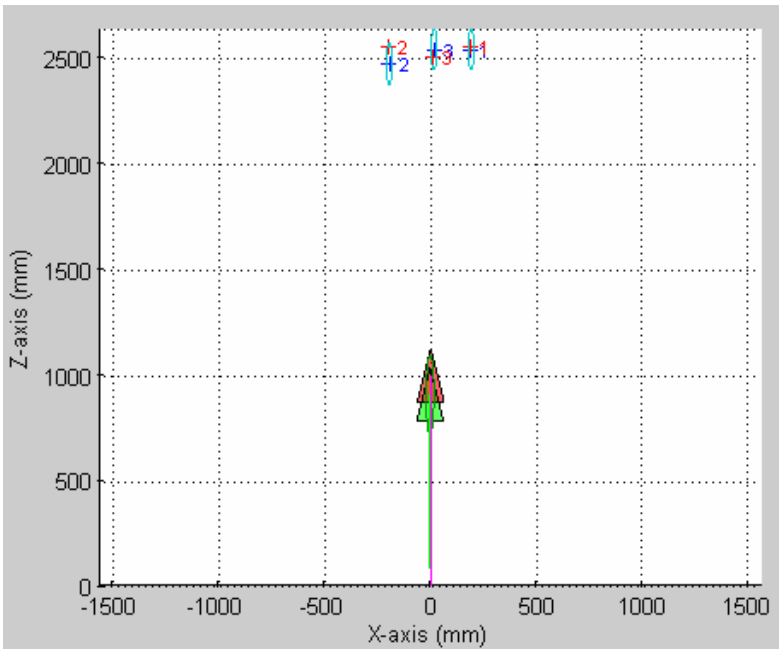

Figure 7-73 Predicted 2D Map for 10th Step of Experiment 5

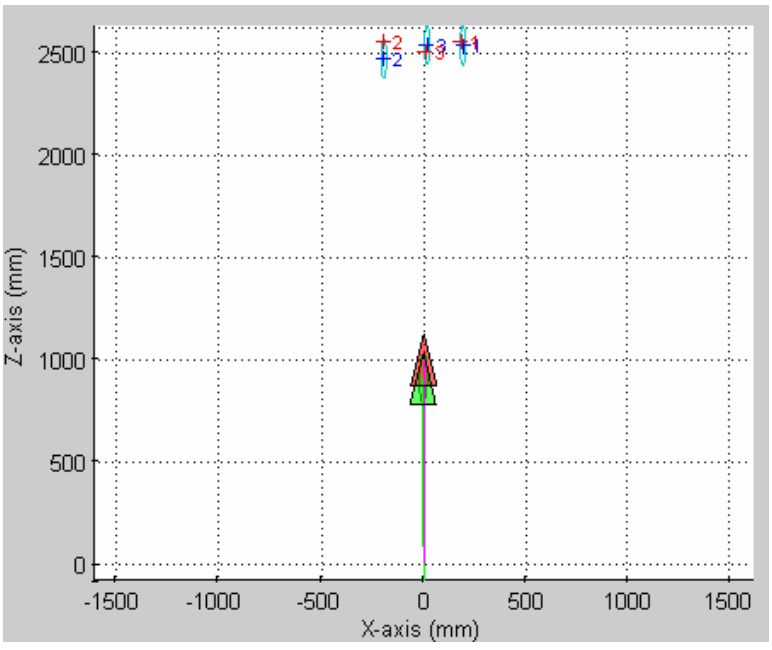

Figure 7-74 Corrected 2D Map for 10th Step of Experiment 5

11th Step (After 90º Turning to Right):

Figure 7-75 Camera Images for 11th Step of Experiment 5

12th Step (After 1 Movement Forward):

Figure 7-76 Camera Images for 12th Step of Experiment 5

Table 7-33 Estimated Landmark Positions at 14th Step for Experiment 5

	X (mm)	Y (mm)	Z (mm)
Landmark 1	191.77	133.04	2534.71
Landmark 2	-194.59	156.08	2464.87
Landmark 3	16.59	-44.98	2533.25
Landmark 4	3827.11	778.62	1719.59
Landmark 5	4446.26	-165.79	1933.54
Landmark 6	2533.04	-4.01	1074.91
Landmark 7	3432.67	-123.14	1964.64

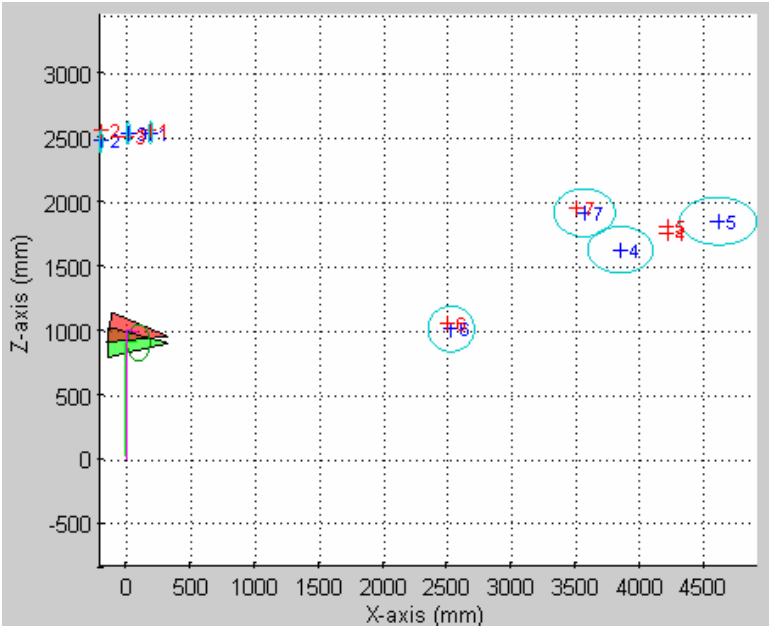

Figure 7-77 Predicted 2D Map for 11th Step of Experiment 5

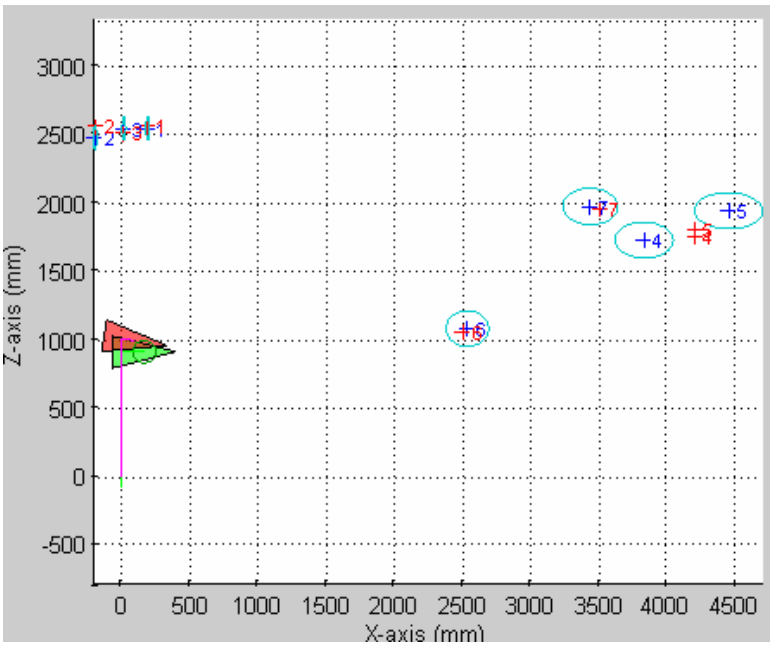

Figure 7-78 Corrected 2D Map for 11th Step of Experiment 5

14th Step (After 2 Movements Forward):

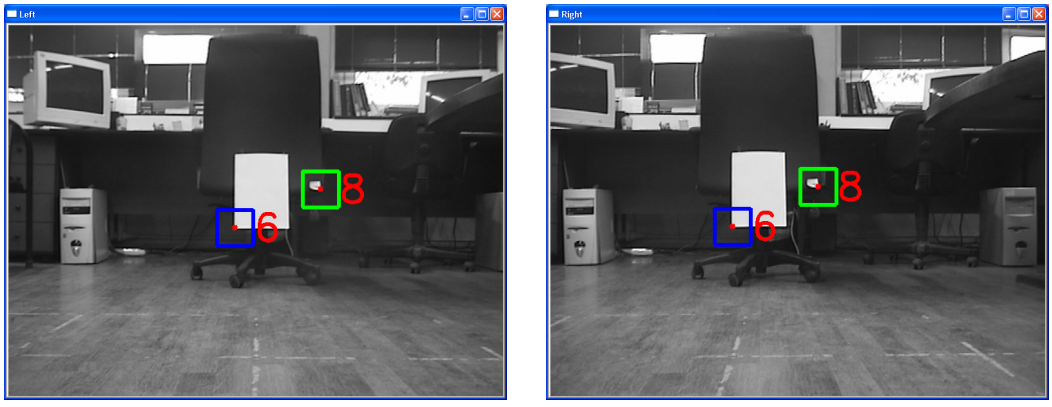

Figure 7-79 Camera Images for 14th Step of Experiment 5

19th Step (After 5 Movements Forward):

Figure 7-80 Camera Images for 19th Step of Experiment 5

Table 7-34 Estimated Landmark Positions at 24th Step for Experiment 5

	X (mm)	Y (mm)	Z (mm)
Landmark 1	192.72	133.04	2529.53
Landmark 2	-193.73	156.08	2470.50
Landmark 3	17.53	-44.98	2532.98
Landmark 4	3826.48	762.13	1620.36
Landmark 5	4384.64	-171.84	1793.28
Landmark 6	2552.34	-21.32	1028.37
Landmark 7	3408.51	-127.98	1855.77
Landmark 8	2677.82	127.77	696.94

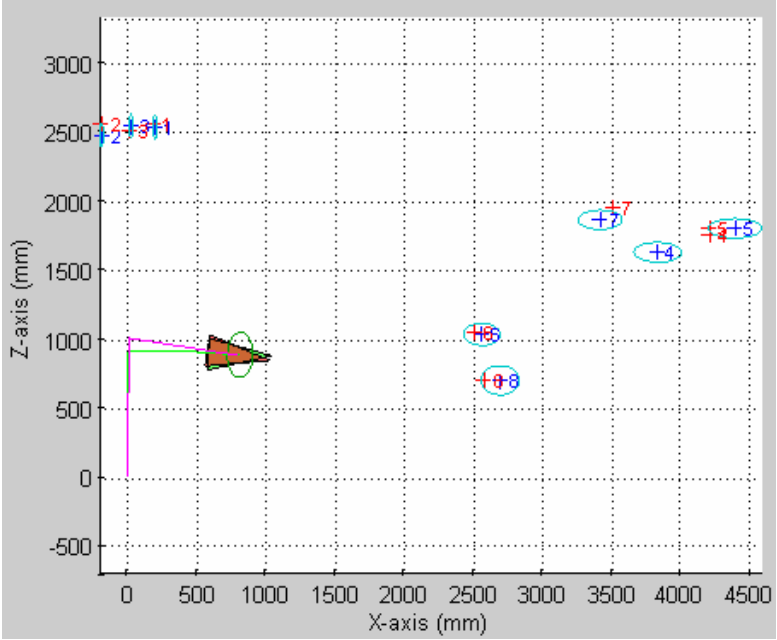

Figure 7-81 Predicted 2D Map for 19th Step of Experiment 5

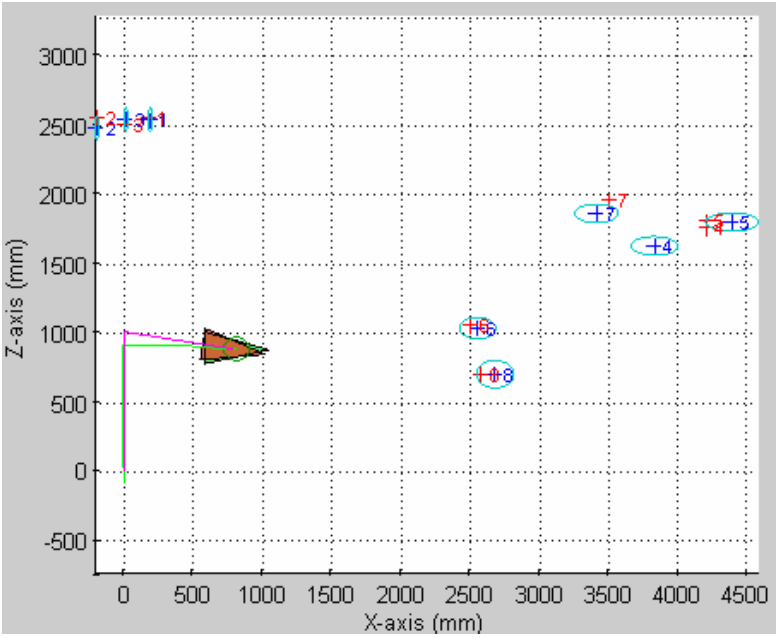

Figure 7-82 Corrected 2D Map for 19th Step of Experiment 5

21st Step (After 1 Movement Forward):

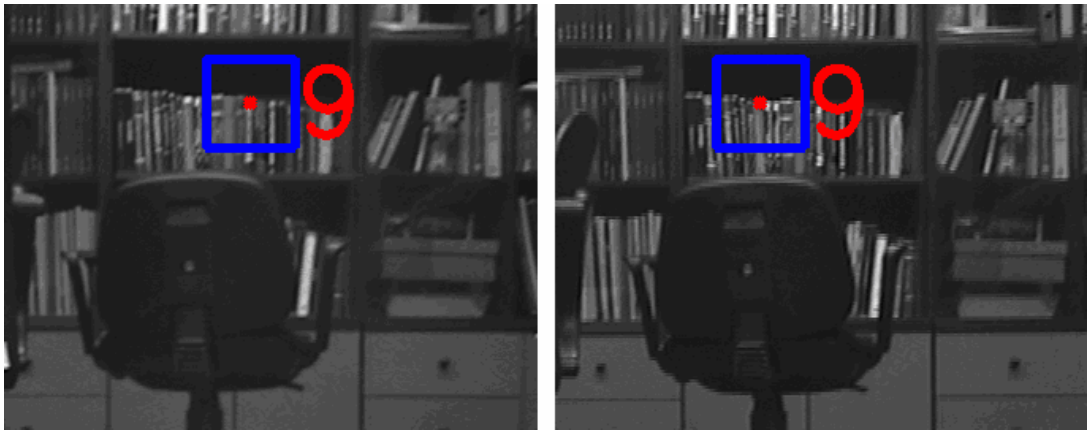

Figure 7-83 Wrong Data Association in 21st Step of Experiment 5

Table 7-35 Estimated Landmark Positions at 27th Step for Experiment 5

	X (mm)	Y (mm)	Z (mm)
Landmark 1	192.72	133.04	2529.54
Landmark 2	-193.73	156.08	2470.50
Landmark 3	17.53	-44.98	2532.98
Landmark 4	3826.43	762.13	1620.46
Landmark 5	4384.60	-171.84	1793.40
Landmark 6	2552.24	-21.32	1028.44
Landmark 7	3408.47	-127.98	1855.86
Landmark 8	2677.69	127.77	697.01
Landmark 9	466.27	620.65	-1831.47

Since the detected image patches on the left and right camera images are different, the estimated position for landmark 9 is wrong.

25th Step (After 4 Movement Forward):

Table 7-36 Estimated Landmark Positions at 28th Step for Experiment 5

	X (mm)	Y (mm)	Z (mm)
Landmark 1	192.77	133.04	2529.36
Landmark 2	-193.71	156.08	2470.70
Landmark 3	17.59	-44.98	2532.97
Landmark 4	3825.51	762.13	1616.67
Landmark 5	4383.88	-171.84	1788.99
Landmark 6	2550.67	-21.32	1025.93
Landmark 7	3407.81	-127.98	1852.44
Landmark 8	2675.75	127.77	694.33
Landmark 9	461.05	620.65	-1831.67
Landmark 10	220.44	874.79	-3126.67
Landmark 11	-2.80	915.17	-3279.54

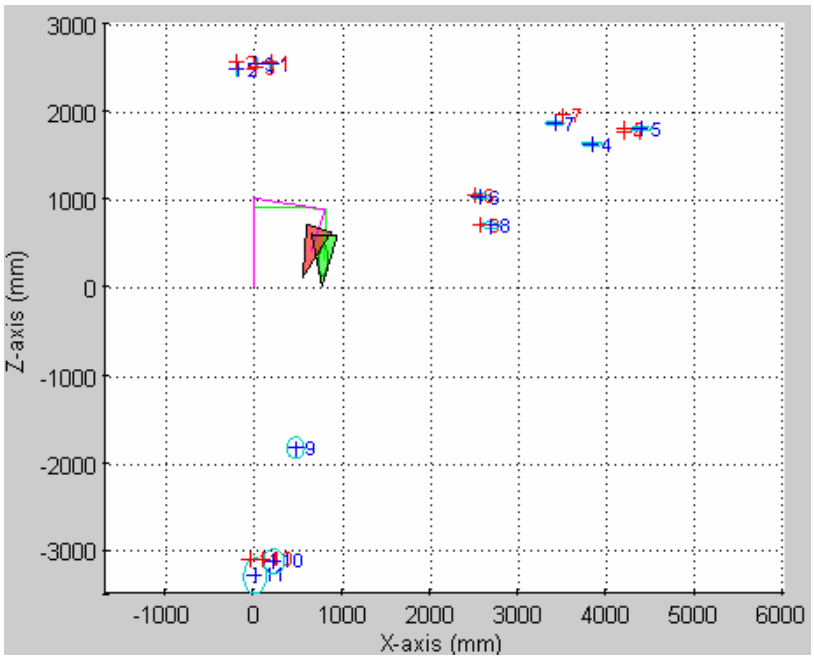

Figure 7-84 Predicted 2D Map for 25th Step of Experiment 5

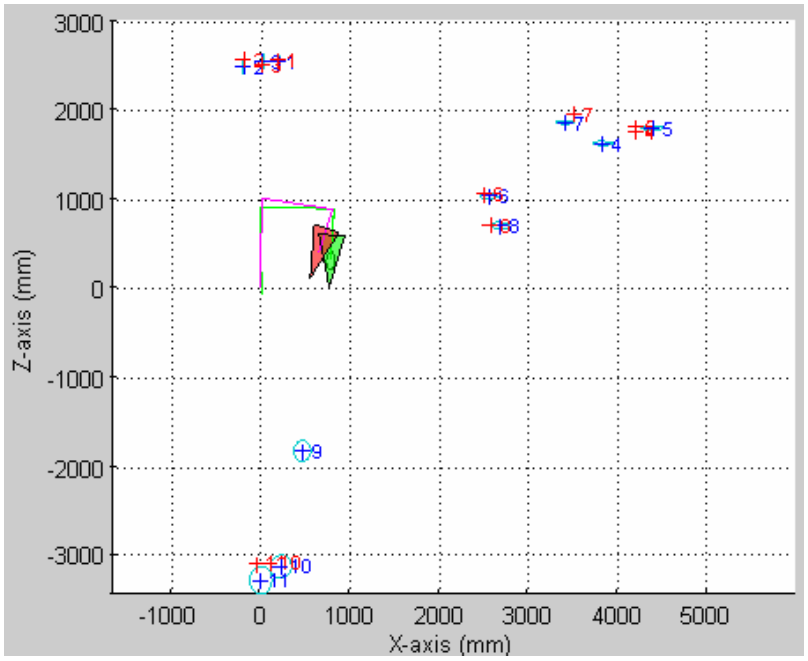

Figure 7-85 Corrected 2D Map for 25th Step of Experiment 5

32nd Step (After 4 Movements Forward, 90º Turning to Right and 2 Movements Forward):

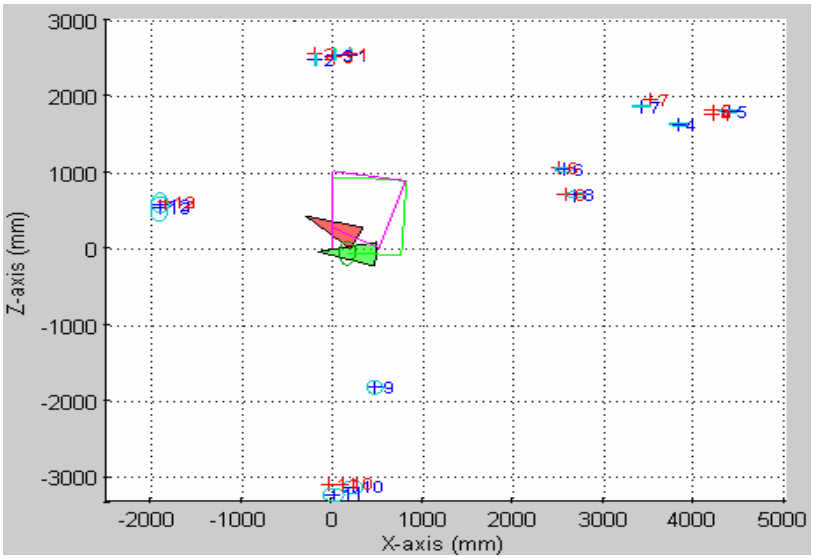

Figure 7-86 Predicted 2D Map for 32nd Step of Experiment 5

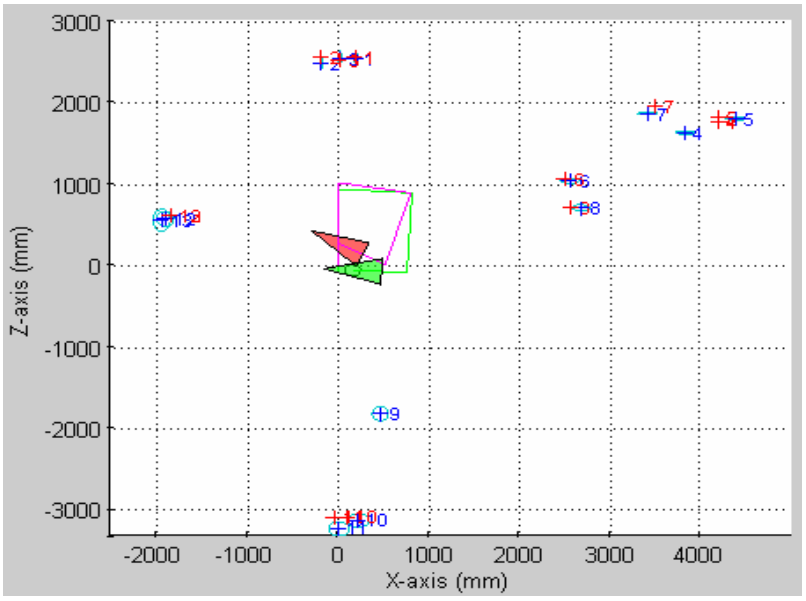

Figure 7-87 Corrected 2D Map for 32nd Step of Experiment 5

38th Step (After 90º Turning to Right):

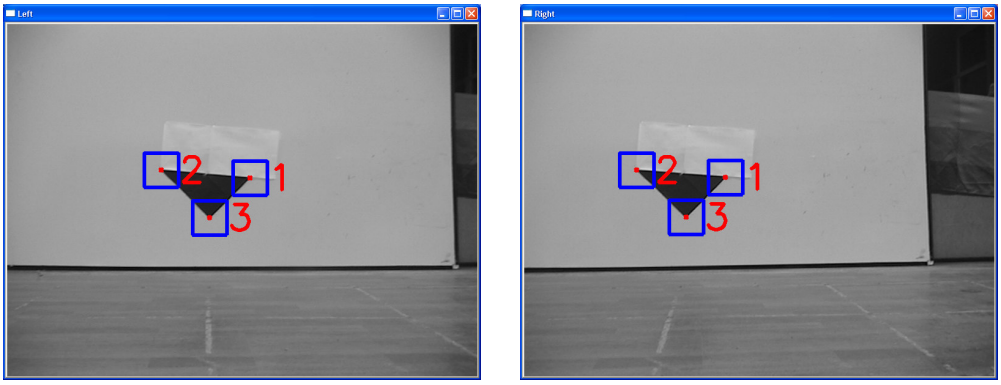

Figure 7-88 Camera Images for 38th Step of Experiment 5

Table 7-37 Estimated Landmark Positions at 31st Step for Experiment 5

	X (mm)	Y (mm)	Z (mm)
Landmark 1	192.82	132.94	2528.37
Landmark 2	-193.46	156.19	2471.71
Landmark 3	17.57	-44.85	2532.97
Landmark 4	3827.36	762.13	1597.04
Landmark 5	4386.60	-171.84	1766.35
Landmark 6	2549.50	-21.32	1012.88
Landmark 7	3410.84	-127.98	1834.85
Landmark 8	2672.91	127.77	680.51
Landmark 9	443.45	620.65	-1833.86
Landmark 10	192.86	881.29	-3144.14
Landmark 11	-22.25	911.75	-3242.23
Landmark 12	-1899.30	574.89	570.74

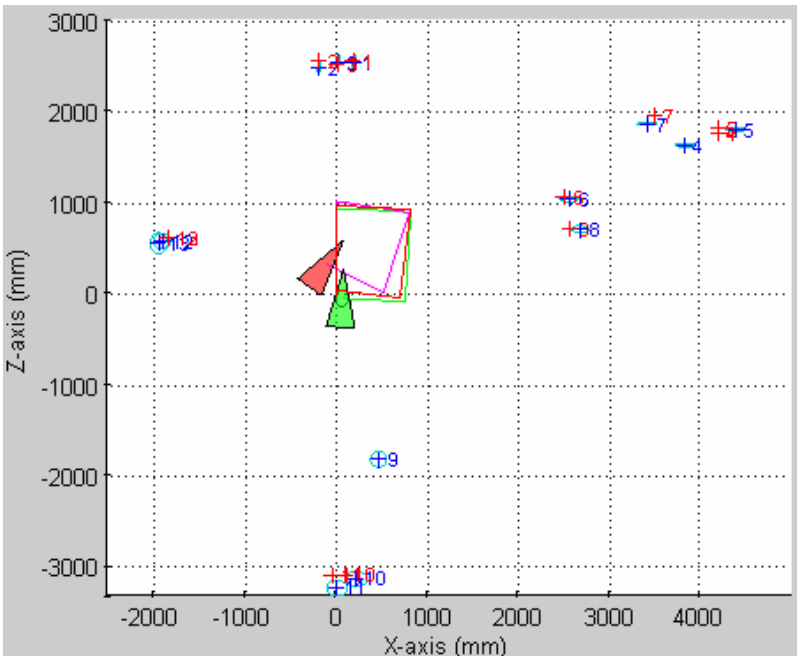

Figure 7-89 Predicted 2D Map for 38th Step of Experiment 5

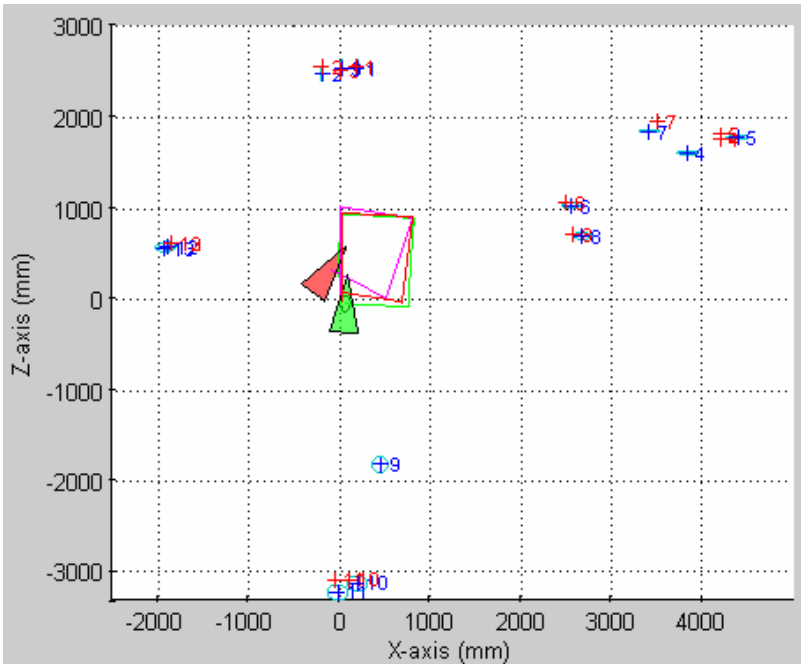

Figure 7-90 Corrected 2D Map for 38th Step of Experiment 5

In last step of the experiment robot completes the loop and reobserves the landmarks that were added in the initialization time. Therefore the robot position uncertainty decreases. It is clearly seen that the robot estimated path is more close to the actual path compared to odometry case.

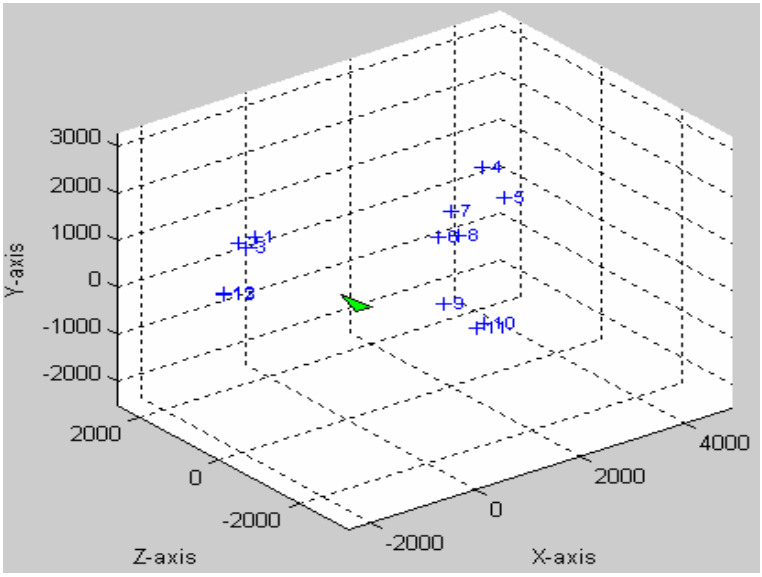

Figure 7-91 Estimated 3D Map of Experiment 5

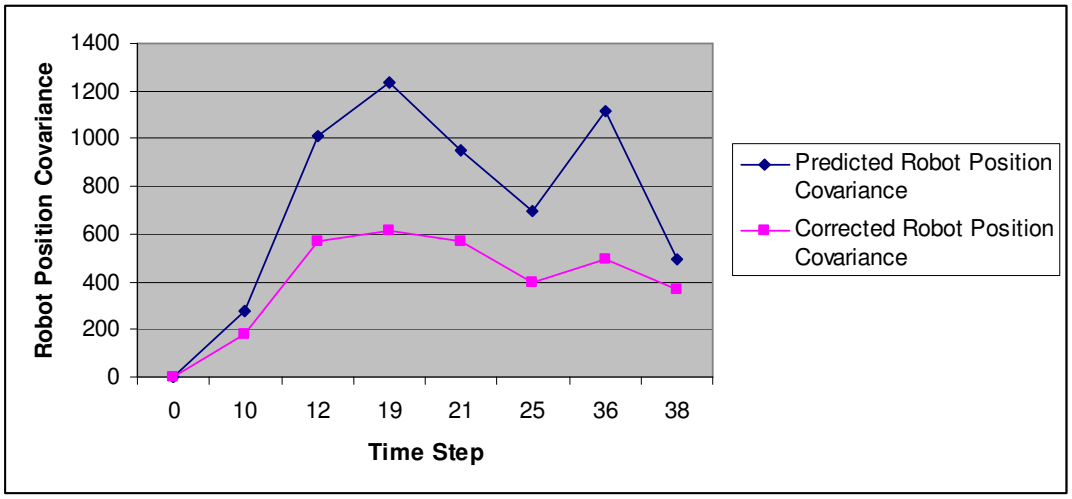

Figure 7-92 Robot Position Covariance Variation of Experiment 5 (mm^2)

The robot positional error increases firstly, the starts to decrease with reobservations. At time step 38, the robot position uncertainty diminishes greatly where the robot reobserves the landmarks at initialization.

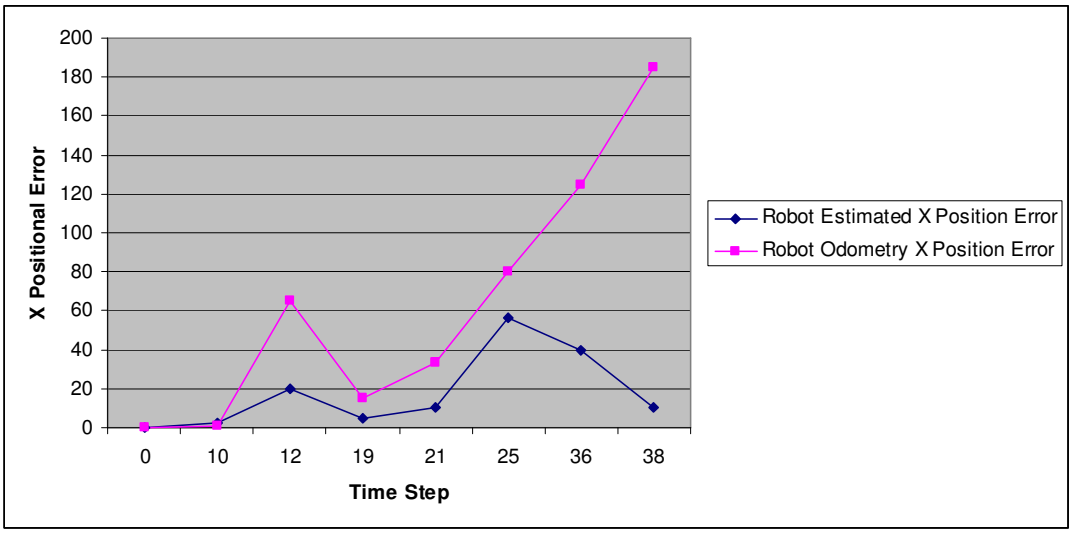

Figure 7-93 Robot X Positional Error Variation of Experiment 5 (mm)

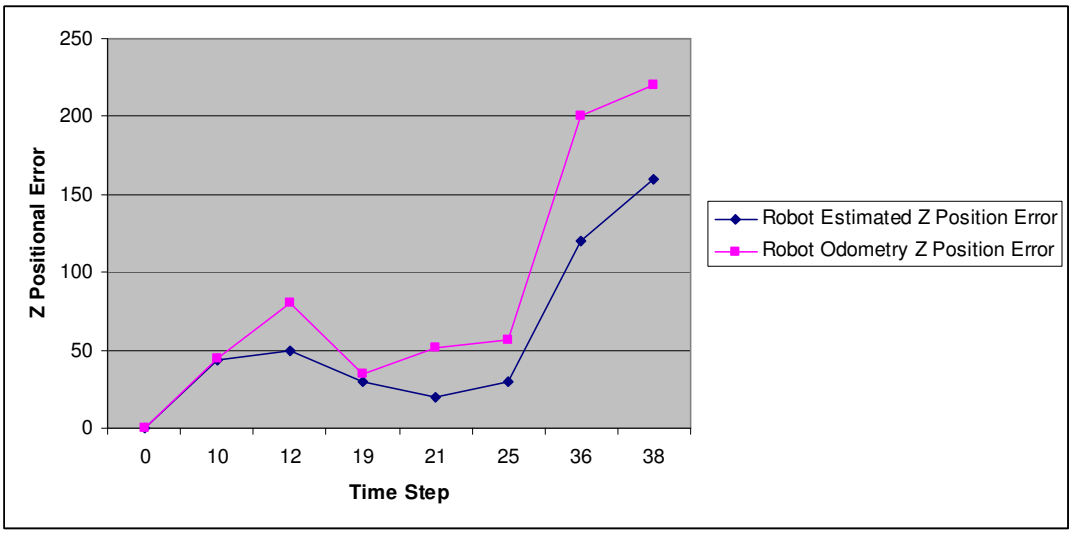

Figure 7-94 Robot Z Positional Error Variation of Experiment 5 (mm)

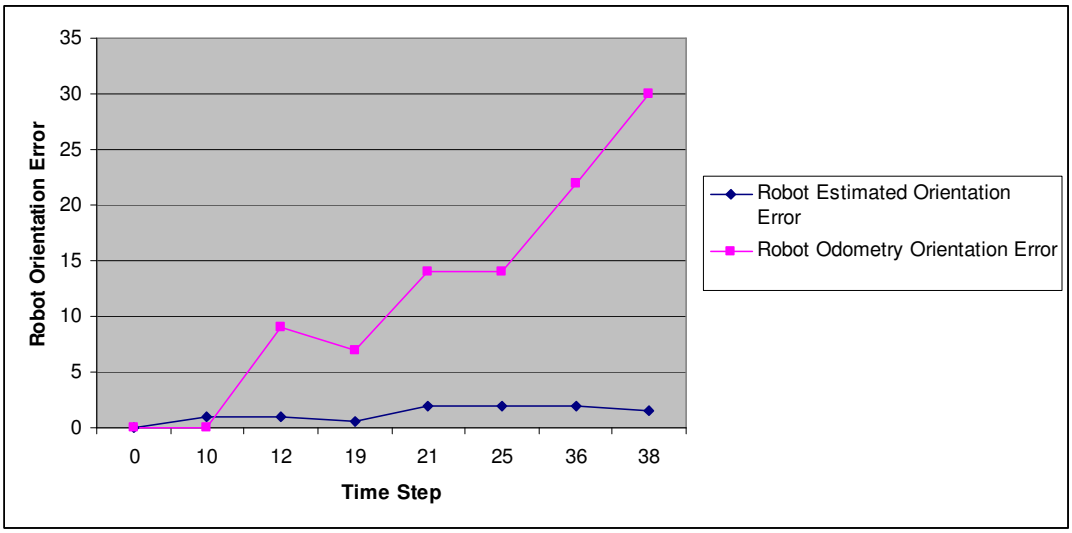

Figure 7-95 Robot Orientation Error Variation of Experiment 5 (degrees)

To summarize the experimental results, it is seen that in almost all steps the position and orientation of the robot is estimated better than the odometry readings. The robot position uncertainty increases in first steps, and then begins to decrease reobserving landmarks in the map. The uncertainties of landmark positions also decrease with time by reobserving them. The uncertainties of some landmarks shrinks to much that the system becomes over confident about their places and the ground truth positions of these landmarks stands outside the uncertainty ellipses.

Experiment 6: Map Management Strategy for Real-Time Efficiency

When the number of landmarks in the current map exceeds a threshold, typically 100, the algorithm cannot run in real time. Therefore in order to preserve the real-time efficiency for our system an algorithm is implemented which is explained in the previous chapter. In this experiment, we just want to show how the efficiency algorithm works. Therefore, we have not waited for 100 landmarks to be added to the map but specify this number as 10. Moreover, we specify the percentage of landmarks to be deleted as 50% of all the landmarks. We define the number of

blocks that the 3D map is divided into as 4 by specifying both the number of x-axis and z-axis intervals as 2.

At some step, the robot has added total of 11 landmarks into the map as follows:

Table 7-38 Estimated Landmark Positions for Experiment 7 Before Real-Time Efficiency Algorithm Is Run

	X (mm)	**Y (mm)**	**Z (mm)**
Landmark 1	115.07	24.67	1752.03
Landmark 2	109.98	147.69	1762.22
Landmark 3	264.60	157.75	1783.05
Landmark 4	268.84	32.22	1776.09
Landmark 5	2225.49	177.66	1058.02
Landmark 6	2230.46	-19.86	902.54
Landmark 7	1384.36	-144.84	-1998.74
Landmark 8	907.60	1164.17	-4058.31
Landmark 9	1180.43	-88.64	-2111.05
Landmark 10	-15.25	920.88	-4092.15
Landmark 11	-74.19	1083.47	-4017.29

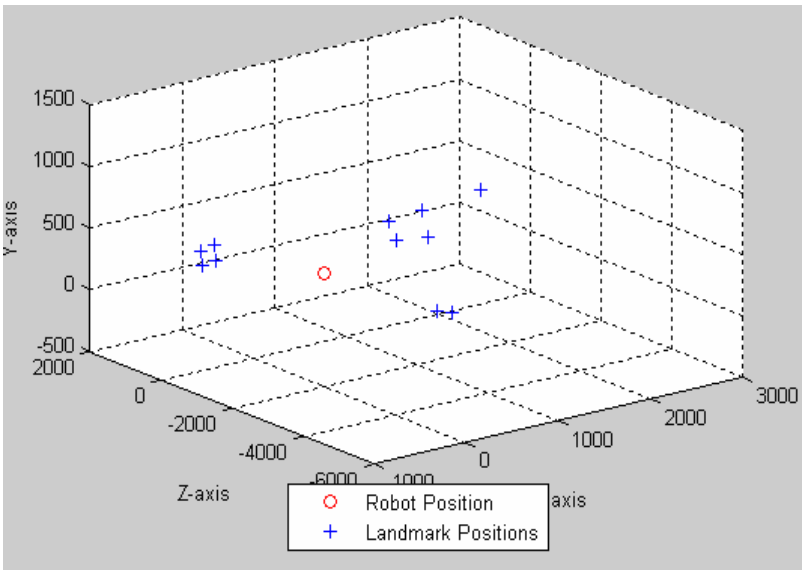

Figure 7-96 Constructed 3D Map for Experiment 6 Before Real-Time Efficiency Algorithm Is Run

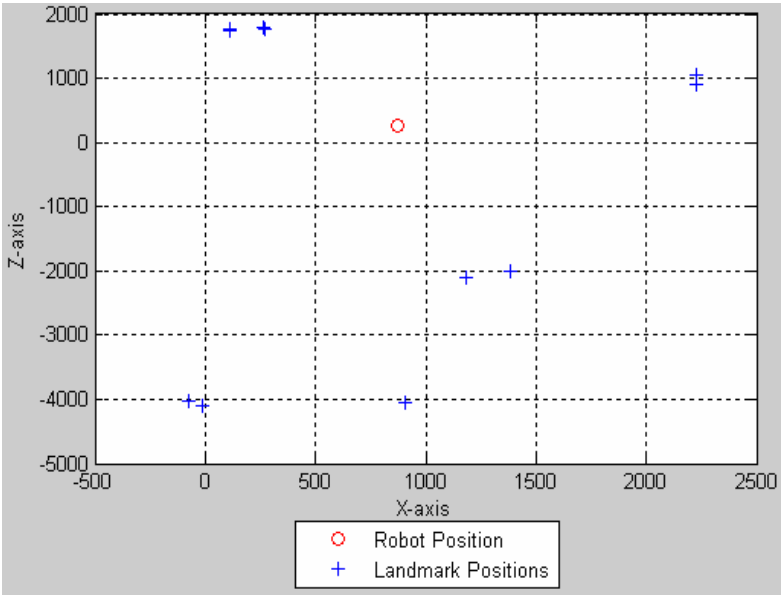

Figure 7-97 Constructed 2D Map for Experiment 6 Before Real-Time Efficiency Algorithm Is Run

After the algorithm has been run with the specified parameters, the number of landmarks has been reduced to 5, and then immediately a new landmark has been initialized with ID 6. In this step, the landmarks seen by the robot have also been tracked and correction is done, so that the remaining landmarks have their position estimates with little changes.

Table 7-39 Estimated Landmark Positions for Experiment 6 After Real-Time Efficiency Algorithm Is Run

	X (mm)	**Y (mm)**	**Z (mm)**
Landmark 1	115.18	24.67	1753.19
Landmark 2	110.17	147.69	1763.47
Landmark 3	264.99	157.75	1781.87
Landmark 4	2212.28	177.66	1025.83
Landmark 5	1104.69	-88.64	-2121.04
Landmark 6	-114.68	906.71	-4026.69

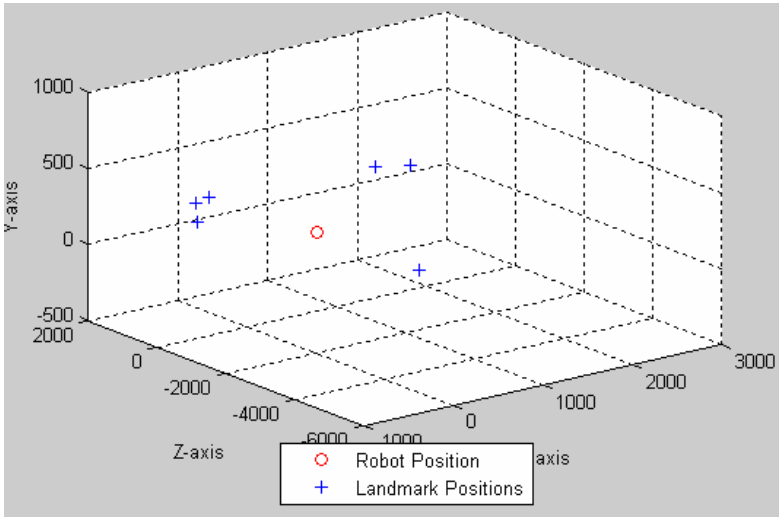

Figure 7-98 Constructed 3D Map for Experiment 6 After Real-Time Efficiency Algorithm Is Run

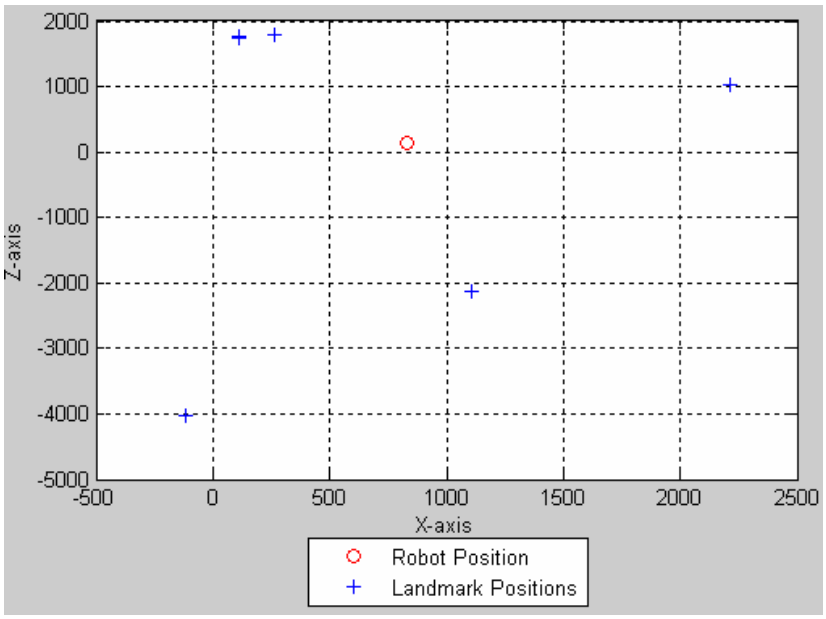

Figure 7-99 Constructed 2D Map for Experiment 6 After Real-Time Efficiency Algorithm Is Run

As it can be seen clearly from the Figure 7-99, the most uncertain landmarks have been deleted from the second, third and fourth quarters of the map until only there remains only one landmark at each sub block. On the other hand, the first quarter block has 2 landmarks just after the algorithm has finished. Since the newly initialized feature after the algorithm has been run, exists also in the first block, that part of the map contains three landmarks eventually.

CHAPTER 8

SUMMARY AND CONCLUSIONS

In this chapter of the thesis we will summarize what we have done throughout this work, review the conclusions that we have gained from the experiments and present some possible extensions and potential future work for our thesis.

We will first review our contributions and mention the conclusions.

- We have implemented an accurate algorithm for simultaneous localization and map building problem by utilizing Extended Kalman Filter. The results of the experiments indicate that our algorithm works accurately and determines the robot and landmark positions within small error ranges. But it has been seen that the error amount has increased when wrong feature associations were made. Because wrong data associations cause depth measurements to be incorrect which are used during the correction phase of the algorithm. We have testified that the performance of the SLAM algorithm's localization capability improves over the odometry in time, which is expected due to accumulation errors in the odometry. This difference is clearly seen, when a manual wheel slippage is introduced to the system. After some time steps the SLAM algorithm recovered the error, whereas the odometry values had drift from the ground truth proportional to the slippage amount.

- In our work, we have defined a search region on the camera images similar to [36] for remeasurement of the existing features. We have seen that while the

probability of finding a feature on the region decreases with decreasing number of standard deviations, the probability of the mismatches increases with increasing number of standard deviations on the other hand. Therefore a compromise should be attained between these two conditions and 5 standard deviations are found to be good choice for this situation.

- We have defined a set of conditions for the remeasurement of the existing features similar to [45]. In this way deletion of the strong features is avoided, because even strong features would be deleted when tried to be measured from a position that is too different from the robot position of initializing that feature. We have also seen that while robot is making small angular motions the features can still be tracked, they cannot be tracked for large angular motions since they disappear from the field of view.

- Stereo and single camera modes of SLAM are defined. In [17], these two modes of SLAM are also implemented. Our single camera implementation differs from that one in order to be appropriate for our static camera construction. The measurement model of the stereo mode is more certain since its calibration parameters are obtained from the static camera calibration procedure. Thus, the depth measurement test results are much better for the stereo case. In the mono case the uncertainties in the measurement model are high, since the camera model depends on the vehicle model and the uncertainties in the robot motion are much greater than the measurements of the static camera calibration procedure. Nevertheless, it is good to have both operation modes in case of emergency. Although the mono operation does not give as good results as the stereo one, it can be needed when one of the cameras has problems. In this situation, the mono operation can be activated and robot may be said to run in a degraded mode. Furthermore, the mono operation can be the first choice for some systems that do not require high accuracy and where the cost is more important.

- We have built a map management strategy like in [45] that takes care of the quality and the size of the map. The strategy eliminates the bad features from

the map by defining a rule set for remasurement of the existing landmarks. The landmarks that cannot achieve the requirements of the strategy are removed from the map and the map quality is kept high in this way. We have extended the strategy in order to provide a mechanism that takes role when the size of the maps becomes too large for efficient operation. So, the size of the map is kept in acceptable values and performance of the algorithm remains high. These mechanisms have been tested during the experiments and seen to work well.

- We have also defined a mission concept in order to see the beyond of localization and map building operation. When a reliable SLAM algorithm is run in an unknown environment for enough time, the robot will have built the map of the environment and localized itself in it. After this point, the robot is now ready for a mission to accomplish. One of the basic missions that can be assigned to the robot is obviously to reach a goal point. Having learned its goal point coordinates, robot simply calculates the necessary parameters and plans its path. We believe that this extra feature of our system is important; since it takes our vision one step further and indicates how the results of the SLAM algorithm can be utilized for further operations.

- Lastly, we have constructed a complete SLAM system which has various operating modes and various components including a graphical user interface, *SLAM Suite*. By the help of this GUI, it is easy to interact with the SLAM system, to change the necessary parameters related to the SLAM operation and to see the results of the algorithm visually.

We have also some suggestions for the future to extend this work:

- We mention that there may be switching between the stereo and mono operational modes in the case of a camera error. Some intelligence can be given to the SLAM system to decide the operational modes by considering the states of its elements. In this way, a more autonomous system can be formed. The

similar intelligence can also be provided for selection of the parameter values. During the operation time, the robot system can evaluate its current performance somehow and make a decision for specifying a new set of operating parameters.

- We also mention that tracking of the existing features during large angular movements cannot be achieved. However this situation can be dealt with, if the robot system has a movable camera system. In this way, the features can be tracked more longer periods which means that more accurate results can be obtained.

- In this thesis, only the strong corners and cornerwise features are used. In order to widen the utilizable range of landmarks, other features such as edges and planes can be incorporated to the system in addition to the corners.

- Finally, the obstacle avoidance task in the autonomous operational mode and the mission concept is achieved by making use of the sonar information. By constructing maps visually in the form of occupancy grids as stated in 2.2.1.1, the obstacle avoidance can be achieved by considering the occupancies in the map.

REFERENCES

[1]: Ingemar J. Cox "Blanche: An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle". *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 2, April 1991

[2]: Ingemar J. Cox "Blanche: Position Estimation for an Autonomous Robot Vehicle". *IEEE/RSJ International Workshop On Intelligent Robots and Systems '89*, Tsukuba, Japan, Sep. 4-6, 1989

[3]: Jens-Steffen Gutmann, D. Fox "An Experimental Comparison of Localization Methods Continued". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, EPFL, Lausanne, Switzerland, October 2002

[4]: Jens-Steffen Gutmann, W. Burgard, D. Fox, K. Konolige "An Experimental Comparison of Localization Methods". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B.C., Canada, October 1998

[5]: Jens-Steffen Gutmann "Markov-Kalman Localization for Mobile Robots". *16th International Conference on Pattern Recognition (ICPR'02)*, Vol.2, 2002

[6]: M. I. Ribeiro, P. Lima "Markov Localization". *Robotica Movel*, Portugal, May 2002

[7]: Henry W. Stone "Mars Pathfinder Microrover: A Low-Cost, Low-Power Spacecraft". *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics,* Madison, WI, August 1996

[8]: L. Whitcomb, D. Yoerger, H. Singh, J Howland "Advance in Underwater Robot Vehicles for Deep Ocean Exploration: Navigation, Control and Survey

Operations". *Proceedings of the Ninth International Symposium of Robotics Research (ISRR '99)*, Snowbird, Utah, USA, October 9-12, 1999

[9]: Rodney A. Brooks "Achieving Artificial Intelligence Through Building Robots". *A.I. Memo 899*, May, 1986

[10]: E. Krotkov, M. Hebert, R. Simmons "Stereo Perception and Dead Reckoning for a Prototype Lunar Rover". *SpringerLink*, Vol.2, No.4, Dec 1995

[11]: E. Freund, J. Rossman "Intelligent Autonomous Robots for Industrial and Space Applications". *Proceedings of the IEEE/RSJ/GI Intelligent Robots and Systems, IROS*, Munich, Germany, September 1994.

[12]: Andrew J. Davison "SLAM with a Single Camera". *SLAM/CML Workshop at ICRA*, May 9, 2002

[13]: Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, O. Stasse "Mono SLAM: Real-Time Single Camera SLAM". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No.6, June 2007

[14]: B. Williams, P. Smith, I. Reid "Automatic Relocalization for a Single-Camera Simultaneous Localization and Mapping System". *IEEE International Conference on Robotics and Automation*, Roma, Italy, 10-14 April, 2007

[15]: Andrew J. Davison, David W. Murray "Simultaneous Localization and Map Building Using Active Vision". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.24, No.7, July 2002

[16]: T. N. Thanh, Y. Sakaguchi, H. Nagahara, M. Yachida "Stereo SLAM Using Two Estimators". *IEEE International Conference on Robotics and Biomimetics*, Kunming, China, December 17-20, 2006

[17]: T. Lemaire, C. Berger, I. Jung, S. Lacroix "Vision-Based SLAM: Stereo and Monocular Approaches". *International Journal of Computer Vision*, 2007

[18]: F. Dellaert, D. Fox, W. Burgard, S. Thrun “Monte Carlo Localization for Mobile Robots”. *IEEE International Conference on Robotics and Automation*, Vol.2, Detroit, MI, USA, 1999

[19]: Hans-Joachim v. d. Hardt, D. Wolf, R. Husson “The Dead Reckoning Localization System of the Wheeled Mobile Robot ROMANE”. *IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 1996

[20]: Tim Bailey “Mobile Robot Localization and Mapping in Extensive Outdoor Environments”. *PhD Thesis*, The University of Sydney, August 2002

[21]: R. Baczyk, A. Kasinski, P. Skrzypczynski “Vision-Based Mobile Robot Localization with Simple Artificial Landmarks”. *Prepr. 7th IFAC Symp. on Robot Control*, Pages 217-222, Wroclaw, 2003

[22]:Michael Csorba “Simultaneous Localization and Map Building”. *PhD Thesis*, Balliol College, Oxford, 1997

[23]: John Folkesson “Simultaneous Localization and Mapping with Robots”. *PhD Thesis*, School of Computer Science and Communication, Stockholm, Sweden, September 2005

[24]: David Ribas “Report on the Research Project Towards Simultaneous Localization & Mapping for an AUV Using an Imaging Sonar”. *Project Report*, Universitat de Girona, June 2005

[25]: Jose E. Guivant “Efficient Simultaneous Localization and Mapping in Large Environments”. *PhD Thesis*, The University of Sydney, May 2002

[26]: Stefan Bernard Williams “Efficient Solutions to Autonomous Mapping and Navigation Problems”. *PhD Thesis*, The University of Sydney, September 2001

[27]: Paul Michael Newman “On the Structure and the Solution of the Simultaneous Localization and Map Building Problem”. *PhD Thesis*, The University of Sydney, March 1999

[28]: Kristopher R. Beevers “Mapping with Limited Sensing”. *PhD Thesis*, Rensselaer Polytechnic Institute, Troy, New York, May 2007

[29]: J. Gasos, A. Saffiotti “Integrating Fuzzy Geometric Maps and Topological Maps for Robot Navigation”. *3rd International ICSC Symposium on Soft Computing (SOCO ‘99)*, pages 754-760, Genova, Italy, June 1999

[30]: Giorgio Grisetti “Towards a PhD Thesis on Simultaneous Localization and Mapping”. *PhD Thesis Report*

[31]: A. Arleo, Jose d. R. Millan, D. Floreano “Efficient Learning of Variable-Resolution Cognitive Maps for Autonomous Indoor Navigation”. *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 6, December 1999

[32]: E. Remolina, B. Kuipers “Towards a General Theory of Topological Maps”. *Artifical Intelligence*, Elsevier B. V., 2003

[33]: P. Buschka, A. Saffiotti “Some Notes on the Use of Hybrid Maps for Mobile Robots”. *8th International Conference on Intelligent Autonomous Systems*, Amsterdam, The Netherlands, March 2004

[34]: G.Q. Huang, A.B. Rad, Y.K. Wong “Online SLAM in Dynamic Environments”. *Proceeding of the 12th International Conference on Advanced Robotics, ICAR ’05*, Pages 262-267, 18-20 July 2005

[35]: M. W. M. Gamini Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, M. Csorba “A Solution to the Simultaneous Localization and Map Building (SLAM) Problem”. *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 3, June 2001

[36]: Zafer Arıcan "Vision-Based Robot Localization Using Artificial and Natural Landmarks". *Master Thesis*, Middle East Technical University, August 2004

[37]: Misbaur Rehman Saad "State Estimation Technique for a Planetary Rover". *Master Thesis*, Helsinki University of Technology, July 2007

[38]: Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/, last visited on August 2008

[39]: Hynek Bakstein "A Complete DLT-based Camera Calibration with a Virtual 3D Calibration Object". *Diploma Thesis*, Charles University, 1999

[40]: Mustafa Özuysal "Manual and Auto Calibration of Stereo Camera Systems". *Master Thesis*, Middle East Technical University, August 2004

[41]: Richard I. Hartley, Peter Sturm "Triangulation". *Computer Vision and Image Understanding*, Vol.68, No.2, Pages 146-157, 1997

[42]: J. Shi, C. Tomasi "Good Features to Track". *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '94*, Pages 593-600, Seattle, WA, USA, 21-23 Jun 1994

[43]: S. Thrun, W. Burgard, D. Fox "Probabilistic Robotics". *Book*, The MIT Press, September 2005

[44]: Paul E. Rybski, "Mobile Robot Localization and Mapping using the Kalman Filter" *Lecture Notes*, Carnegie Mellon University

[45]: Andrew J. Davison, "Mobile Robot Navigation using Active Vision". *PhD Thesis*, University of Oxford, June 1998

APPENDIX A

SLAM SUITE

SLAM Suite allows the user to change various parameters related with feature detection, feature correlation, feature deletion, feature remeasurement, robot configuration, real-time efficiency parameters, camera parameters, motion patterns in addition to capability of different mode selections such as robot operation mode, SLAM mode and motion type. The SLAM operation can be paused and continued in any time during execution, and analysis can be done via graphical and textual information provided by SLAM Suite. Since lots of various SLAM tasks are allowed to be performed with changing several parameters, the program can be seen as a suite of SLAM operations and that's why it is called SLAM Suite.

A.1 The Default View

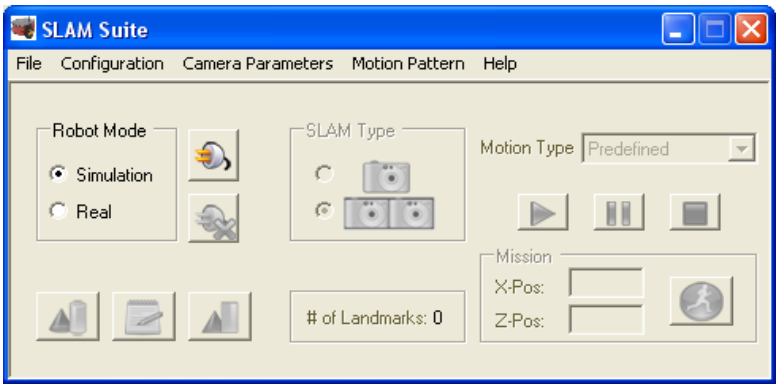

Figure A-1 SLAM Suite: Default View

The default view of SLAM Suite is illustrated in Figure A-1. The connection is not established in this view, only connection and menu items are active. The user can connect to the simulator robot or actual Pioneer robot by selecting the appropriate option and pressing the “Connect” button shown by  icon.

A.2 The Main View

After the connection is established, the view becomes as shown in Figure A-2. In this view, SLAM mode selection, motion type selection, mission concept and map plotting functionalities are active. The user can choice the SLAM mode and motion modes and then begin the operation by just clicking the “Run” button designated by  icon.

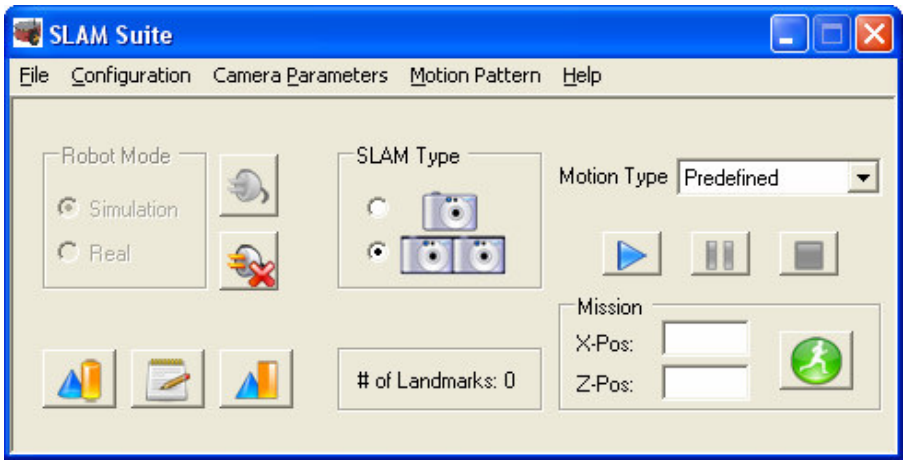

Figure A-2 SLAM Suite: Robot Connection Established

After the SLAM algorithm has started, the operation can be paused in predefined and autonomous modes by clicking the “Pause” button labeled by  icon or terminated by “Terminate” button labeled by  icon as shown in Figure A-3. Since the system waits for keyboard input request from the operator for next motion

in manual mode, pausing is not needed but termination process also applies to this mode. The parameter menus, mode selections, mission concept ad map plotting interface becomes disabled during the operation.

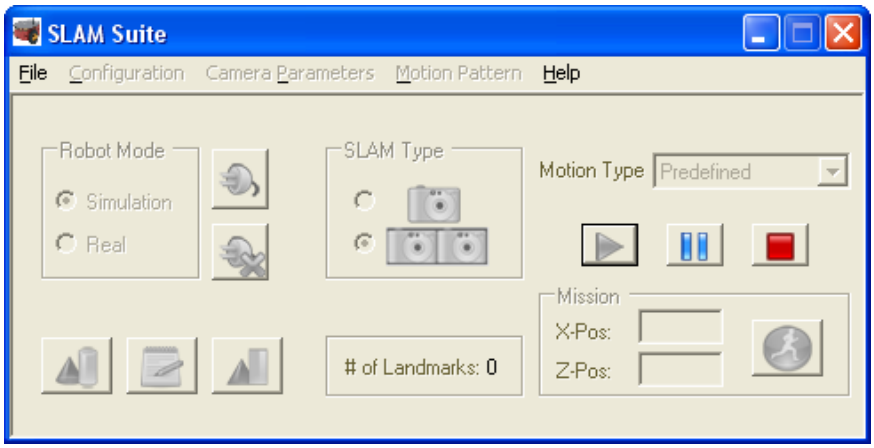

Figure A-3 SLAM Suite: Operation Started

A.3 Map Drawing Interface

When the desired operation is completed or by pausing the algorithm at the middle of the operation information about the robot location and current map contents can be obtained by the map drawing interface of SLAM Suite. The map drawing interface is shown in Figure A-4, when the SLAM operation is paused. The 2 dimensional representation of the current map can be plotted by clicking the "Draw 2D Map" button labeled by icon. In the 2D representation the x and z-axis components of the robot and landmark locations are plotted. When the 3 dimensional representation of the current map is desired to be seen, "Draw 3D Map" button labeled by icon can be clicked. Lastly, SLAM Suite also presents a textual way of gathering the location data. When the user clicks "Export to File" button labeled by icon, the system forms a file named "ConstructedMap.txt" in the current directory, which contains the information about the robot and landmark

locations. If a file with the same name already exists, it is overridden with the current values. The number of the landmarks existing in the current map is also shown in the main view.

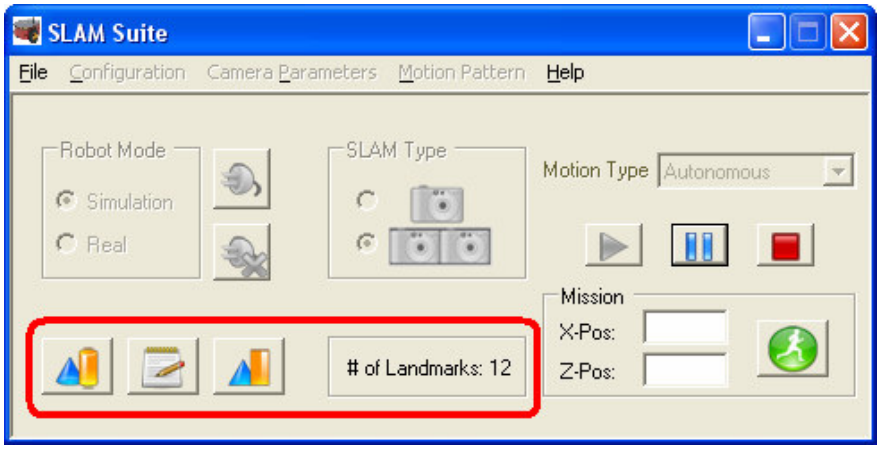

Figure A-4 SLAM Suite: Map Drawing Interface

A.4 Mission Interface

The robot can be stopped in any time during the operation and given a goal point to arrive. The goal point is defined by its x-axis and z-axis coordinates. These values can be entered into the system by the mission interface of SLAM Suite as shown in Figure A-5. After specifying the coordinates of the goal point, the user can assign the mission to the robot by simply clicking the “GO” button labeled by  icon.

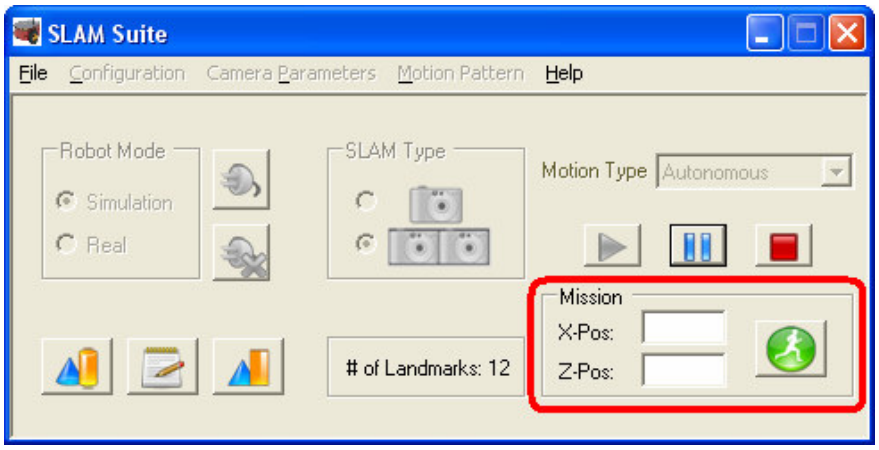

Figure A-5 SLAM Suite: Mission Interface

A.5 Configuration

The configuration of the SLAM system by specifying various parameters is achieved by the configuration menu interface of SLAM Suite. The sub items of the configuration menu are depicted in Figure A-6.

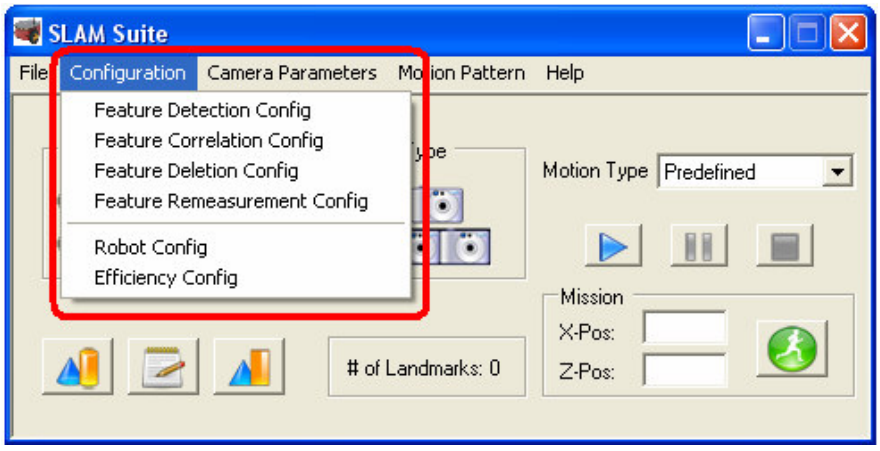

Figure A-6 SLAM Suite: Configuration Menu

Various parameters of our SLAM algorithm can be changed via the sub menus of the configuration menu.

A.5.1 Feature Detection Configuration

In this menu, parameters associated with the feature detection algorithm that we use in this thesis, namely “Good Features to Track”, can be adjusted. These parameters are number of features, quality level and minimum distance. Number of features is the number of corners to be detected in the image. Quality level specifies the minimum accepted quality of the image corners. Lastly, minimum distance specifies the minimum possible distance between the detected corners. Other parameters such as minimum number of active features and image showing can also be changed via this menu interface. While minimum number of active features specifies the minimum number of features which are tracked by our SLAM system, image showing attribute determines whether the images grabbed by the cameras are shown on the screen. The feature detection Configuration menu is shown in Figure A-7.

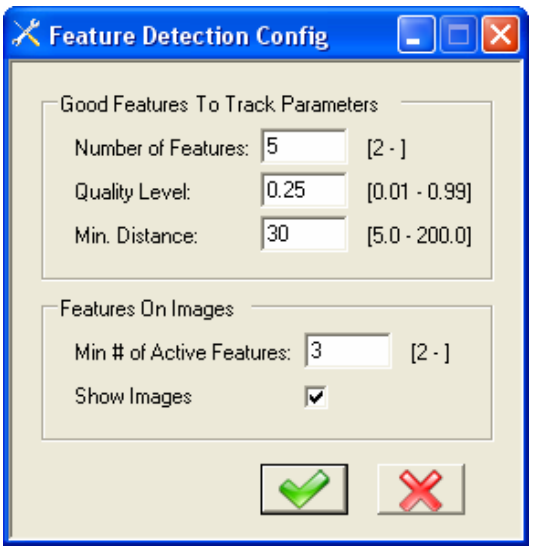

Figure A-7 SLAM Suite: Feature Detection Configuration

A.5.2 Feature Correlation Configuration

The correlation method to be used in feature correlation is selected via this interface. The possible options are “Squared Difference”, “Normalized Squared Difference”, “Cross Correlation”, “Normalized Cross Correlation”, “Correlation Coefficient” and “Normalized Correlation Coefficient”. The correlation threshold value can also be specified under this menu. This value must be chosen between 0 and 0.5 for the first two options, whereas it must be chosen between 0.5 and 1 for other methods. The feature correlation configuration menu is illustrated in Figure A-8.

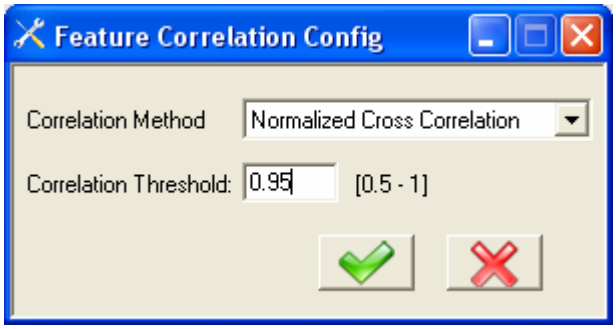

Figure A-8 SLAM Suite: Feature Correlation Configuration

A.5.3 Feature Deletion Configuration

In the SLAM operation, weak features are deleted after some conditions are met. In this menu, these conditions are defined. One of the parameters for the conditions is the minimum number of attempts. After this value is reached for a landmark, the feature is started to check for correlation performance with a ratio for the next steps. If the ratio of the number of measurement attempts with failure is to total number of attempts is greater than the delete ratio, the feature is deleted by the algorithm. This configuration menu is shown in Figure A-9.

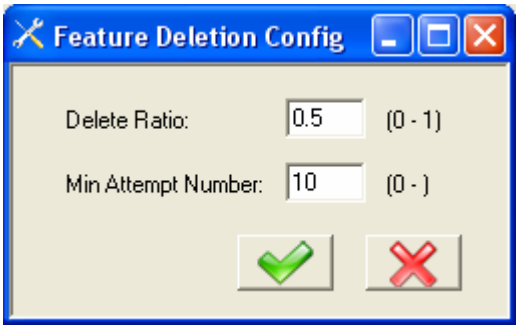

Figure A-9 SLAM Suite: Feature Deletion Configuration

A.5.4 Feature Remeasurement Configuration

There are some conditions to be met for the existing features to be remeasured. The limiting factors for the feature remeasurement are the initial distance between the robot and the landmark and the initial robot angular position. When a feature is initialized (measured for the first time and added to the map), the distance between itself and robot vehicle and the angular position of the robot vehicle are also saved to be used for comparison in the next steps. In the prediction step, feature measurements are also predicted. Current distance between the vehicle and the current landmark is calculated by the help of these predictions. If the distance is in the distance range one of the conditions is met. The range is defined by multiplying the numbers minimum distance ratio and maximum distance ratio with the initializing distance of the landmark. Second condition parameter is the angular robot position. If the absolute difference of the current angular position and initial angular position during initializing the landmark is below the maximum angle difference value, then the second condition is also met. After both of these conditions are met, a search region is formed on the image for the purpose of remasurement. This region is defined by the measurement uncertainties and the specified number of standard deviations. The number of standard deviations can be

set via this menu interface. The feature remeasurement configuration menu is depicted in Figure A-10.

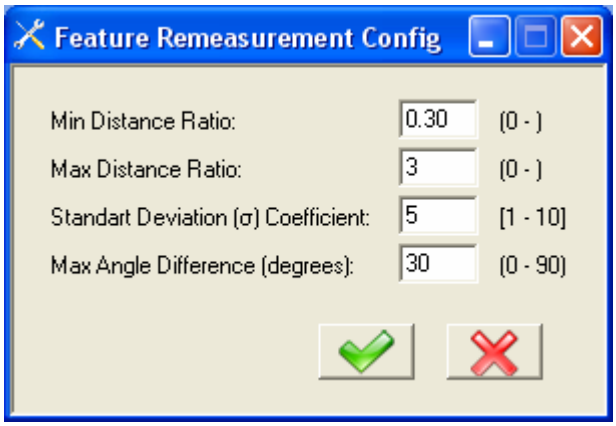

Figure A-10 SLAM Suite: Feature Remeasurement Configuration

A.5.5 Robot Configuration

There are various operating parameters for also the robot itself. For instance the moving distance and turning angle values for control inputs can be defined via this menu. The avoidance distance parameter which specifies distance of the robot to an obstacle that the robot must turn and avoid in autonomous mode is also valued in this menu. Moreover camera system parallelism and active camera channels for the right an left cameras are specified here. Camera system parallelism defines the alignment of the camera coordinate frame and the robot coordinate frame. For example in our case, the robot and camera coordinate frames coincide and the angle between them is 0. But the camera system can be placed in different positions due to some reasons and this parameter must be specified in this situation. The robot configuration menu is shown in Figure A-11.

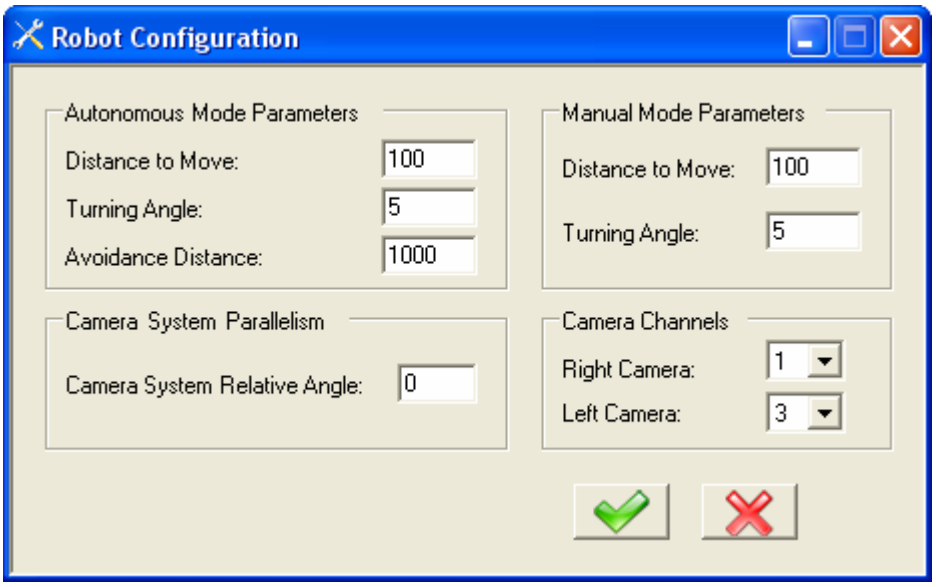

Figure A-11 SLAM Suite: Robot Configuration

A.5.6 Real-time Efficiency Configuration

The last configuration menu is the efficiency configuration. In this menu, the parameters associated with the map management strategy which takes place at the time total number of landmarks reaches a specified threshold are specified. First of all, the attribute that specifies whether the real-time efficiency strategy is active or not can be determined. Then the total number of landmarks can be defined as the threshold for the strategy to take control. The percentage of the landmarks to delete, when the efficiency strategy runs is also specified via this interface. When it is time to run the real-time efficiency, the map of the environment is divided into equal sized blocks firstly. The number of blocks is determined by the number of intervals along the x-axis, y-axis and z-axis dimensions of the smallest 3D rectangular prism that contains all of the landmarks. These interval numbers of the corresponding axes can be defined in this menu. This menu is illustrated in Figure A-12.

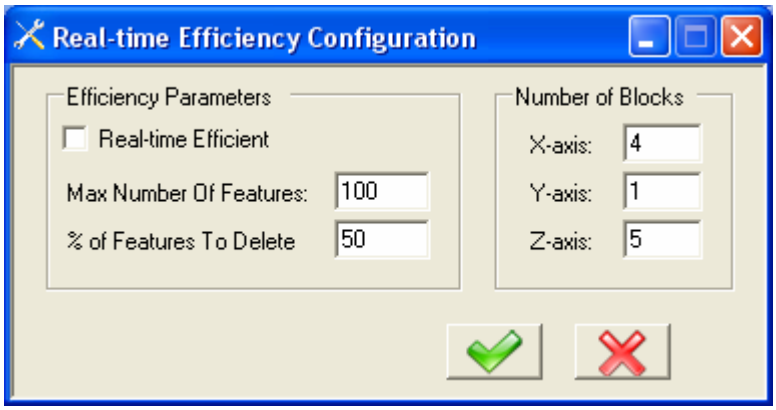

Figure A-12 SLAM Suite: Real-time Efficiency Configuration

A.6 Camera Parameters Loading Interface

The intrinsic and extrinsic camera calibration parameters can be written in a file and it can be loaded via this menu interface. The camera parameters loading interface is depicted in Figure A-13.

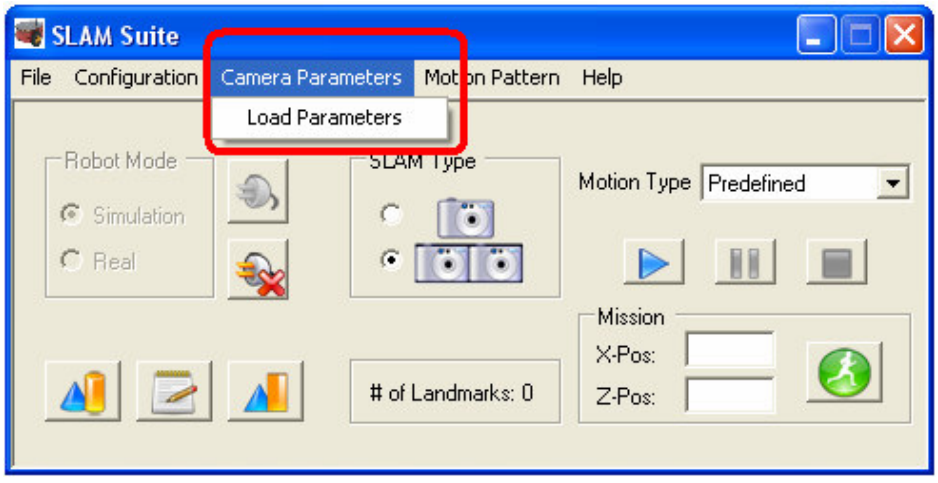

Figure A-13 SLAM Suite: Camera Parameters Loading Interface

A.7 Motion Pattern Loading Interface

The motion pattern can be defined in a file and loaded to the system for the predefined motion mode. The motion pattern loading interface is shown in Figure A-14.

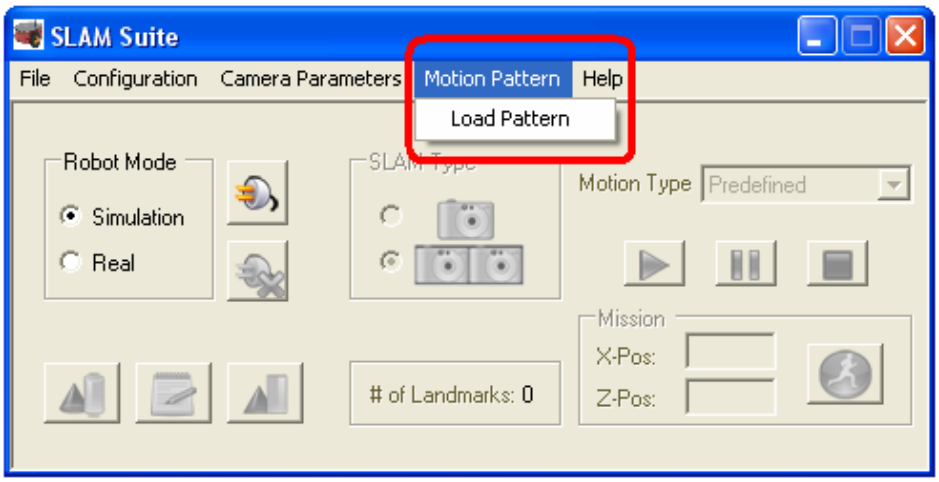

Figure A-14 SLAM Suite: Motion Pattern Loading Interface