GLOBALLY-CONSISTENT 3D SIMULTANEOUS LOCALIZATION AND MAPPING WITH MULTI-SENSOR FUSION

by

Peter Pifu Zhang

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
July 2007

Your file Votre référence
ISBN: 978-0-494-31508-8
Our file Notre référence
ISBN: 978-0-494-31508-8

To the people who helped

Table of Contents

List of Tables

List of Figures

Abstract

This thesis addresses the problem of globally-consistent localization and mapping simultaneously for autonomous mobile robots in an unknown and unstructured 3D environment by sensor fusion. It belongs to the research area of simultaneous localization and mapping (SLAM) in a mobile robot community. The main contribution presented in this thesis is the development of a set of new algorithms for an autonomous mobile robot with full 3D SLAM ability by multi-sensor fusion. It can be classified in the following aspects: (1) *A measurement system architecture designed for mobile robot localization and mapping in a large and unknown environment.* Based on the general SLAM method, a simple structure is designed by using a stereo camera and a set of range sensors to solve the 3D SLAM problem in an unknown environment. (2) *Registration uncertainty for robot self-localization in 3D.* This is an approach to estimate registration uncertainty where the feature correspondence-based method is used during the process of robot pose estimation. (3) *Algorithms for efficient map building in large area.* By using set theory, a set of new algorithms for efficient mapping building is designed during the SLAM solution processing. (4) *An algorithm design of a globally-consistent 3D SLAM by sensor fusion.* One sensor such as stereo camera will be used for local SLAM and another sensor such as buoys or GPS will be used for global path estimation. The estimation results from both sensors will be fused in a global coordinate system to form a globally-consistent map and path for the mobile robot. (5) *A mobile robot simulation system designed for full 3D SLAM application.* In this 3D animation system, a robot would navigate in a virtual space and measure the features with all equipped sensors in its view field at every time. These measurements are used to evaluate algorithms for mobile robot in any environment.

The strategies and methods provided in this dissertation can be used for any autonomous mobile robot. There are a number of directions in which the work presented here could be extended to many other challenging areas and the same strategies could be applied.

List of Abbreviations and Symbols Used

C_k	set of 3D points of the environment in a camera view field at time k
$F(\cdot)$	robot process model
F_k	Image at time k or Frame at time k
$G_k^{p_o,m}$	sub-map from $\Omega_k^{p_o}$ with size of m
$H(\cdot)$	measurement model for a robot system
I_k	rectified image at time k
L	Landmark
L_i	position of a landmark i
$L_k^g(i)$	global position of landmark i in local map k
$L_k^l(i)$	local position of landmark i in local map k
M_v	robot rotation matrix
N	number
P_b	covariance of X_b
P_c	covariance of X_c
P_v	covariance of X_v
P_{vv}	covariance of p_v
Q_v	covariance of μ
R_i	covariance of Z_i
R_k	rotation at time k
R_v	covariance of Z
S	B-spline function
$SIFT$	Scale Invariant Feature Transform
$SLAM$	Simultaneous Localization And Mapping
S_v	covariance of ν
T	Transpose of a matrix
T_k	translation at time k
U	Input of a robot
X_b	robot position estimated from buoys

X_c	robot position estimated from camera
X_v	robot state vector
Z	measurements of landmarks
Z_i	measurement of a landmark i
$\Omega_k^{p_o}$	features appeared at $F_k, \ldots, F_{k-p_o}$
$\hat{Z}$	predicted measurement of Z
μ	robot input noise
ν	innovation in measurements Z
ω	process noise
θ_v	robot orientation
$h(\cdot)$	measurement function for a landmark
m_f	number of landmarks(features) selected for a map
$p(\cdot)$	probability function
p_k^i	measured position for landmark i in C_k
p_o	number of overlaps among frames
p_v	position of a robot

Acknowledgements

I would like to thank all of the people who helped, supported, and encouraged me during the undertaking of this research.

First, I would like to thank my supervisor, Professor Evangelos Milios, for giving me the opportunity to pursue a PhD research on machine learning and mobile robot at Dalhousie University. His support, encouragement, and guidance have proved invaluable at every step of this work. I would also like to thank my co-supervisor, Dr Jason Gu, who provided great help, support, and encouragement during difficult times.

Thanks also to NSERC Canada and IRIS NCE for providing the financial support for the project "Underwater Walking Robot – Aqua". Thanks to all the people who were involved with the project: Dr G. Dudek, Dr M. Jenkin, Dr E. Milios, Dr C. Prahacs, A. Hogue, J. Sattar, P. Giguere, A. German, H. Liu, S. Saunderson, A. Ripsman, S. Simhon, L.-A. Torres, and I. Rekletis. Thanks to Prof. Alonzo Kelly of CMU, who read the thesis carefully and provided a long list of detailed comments that greatly improved the final version of the thesis.

It is also the time to acknowledge my thanks to Weiming Shen, Dr. Hao Meng, Hui Liu, Yongzhen Zhang, Zheyuan Yu, and Gang Wei for help in collecting image data for our experiments.

Perhaps my greatest debt is to those people who have supported me during this entire process: my family and friends. Special thanks are due to my wife, Biwu Fan, and my parents and sisters and brother for providing strong support on every occasion.

Chapter 1

Introduction

1.1 Thesis Statement

3D SLAM is a very important research topic in the mobile robot community, especially in the area of underwater application. A set of new algorithms is designed to solve this 3D SLAM problem by using multi-sensor fusion. These new algorithms have been verified by simulations and experiments in indoor and outdoor environments.

1.2 Motivation

The earth consists of one-third land and two-thirds water. From the beginning of human existence until recently, most of the human activities have taken place on land. Although many brave pioneers have already explored the underwater world, more equipment and tools are needed to expand man's ability for underwater activities such as underwater exploration, fish farming, underwater timber harvesting, environmental assessment, and underwater disaster salvage.

To accomplish this, an autonomous aquatic walking Robot is being designed by research groups from McGill University, York University, Dalhousie University and will have the ability to swim, and walk on land or the ocean floor. Many challenges have to be surmounted: basic vehicle design and function, motion control, modelling and processing of sensor data, pose estimation, map building, scene reconstruction, and exploration and autonomous navigation within 3D environments. Of them all, Simultaneous Localization And Mapping (SLAM) for this underwater robot will be the central topic of this thesis.

Simultaneous Localization and Mapping (SLAM) has been a difficult challenge and an active research topic in the autonomous robot research community. It is based on a fusion process that includes fusing observations of features or landmarks with dead-reckoning information to estimate the location of a robot in a unknown environment and to build a map which includes the landmark locations. This method has many significant advantages, described in [19]: "*(1) it does not need* a priori *topological map of the environment; (2) it does not need to set landmarks in the environment; and (3) it greatly enhances the robot's ability*". The applications of the SLAM technique have attracted great attention from underwater autonomous robotic researchers, aquaculture businesses, autonomous planetary explorers, and developers of autonomous air-borne vehicles, autonomous battle surveillance vehicles, and autonomous all-terrain vehicles for tasks such as mining and construction. These are the situations where absolute position or precise map information is inaccessible or too expensive to obtain.

The conventional measurement system for SLAM used on land autonomous vehicles is not suitable for the Autonomous Underwater Vehicle (AUV) since the environment in water is different from the environment in air. The main aspects which affect the robot's measurement system include the following: firstly, no GPS information can be directly used in the water; secondly, odometry information measured by the encoder installed on the robot is not reliable and is useless because of unpredictable currents in the open ocean area; and thirdly, visibility in water is not as good as in air, this will influence the camera's function. In order to ensure the AUV's SLAM mission is successful, designing a new measurement system is the first step. We will use off-the-shelf sensors, such as video camera, hydrophone, strap-down inertial measurement unit (IMU), global positioning system (GPS), and digital compass for this observation measurement system, which will be efficient in terms of time and cost. Hence, a sophisticated multi-sensor fusion algorithm will be designed for optimal location estimation and map building.

1.3 The Problems

Simultaneous localization and mapping has been one of the challenging and critically important problems in the mobile robot community. Most previous research addressed the 2D context of navigation in a plane, which involves three degrees of freedom (two translational and one rotational). In the underwater environment, the problem is inherently a 3D one, which involves six degrees of freedom (three translational and three rotational). These challenges for underwater robots can be classified into the three fields listed below.

1.3.1 Pose Estimation

Only in the last few years have methods been developed to allow reliable pose estimation and navigation in natural terrestrial environments. Doing this in an underwater environment remains a substantial challenge and yet this is a critical ability for a large range of basic navigation tasks. For terrestrial and airborne applications this can be accomplished using technologies including Global Positioning System (GPS), artificial visual landmarks, or extraction of known structures in the environment such as door frames or other environmental artifacts. The existence of many recently published papers [40, 56] indicates that this is still an active research topic, while most of the papers are still focused on the indoor environment and terrestrial outdoor environment. On the other hand, in the underwater environment, GPS signals are unavailable, artificial landmarks are often impractical or impossible to put into place, and techniques for selecting or detecting naturally invariant landmarks have not been examined, therefore, it is certainly a major research challenge.

1.3.2 Efficient Map Building

A precise map will enable a robot to navigate its environment autonomously. Since the last decade, several kinds of maps have been established, such as grid occupancy map, topological map, and geometric map. Some approaches for consistent map building have been used widely. Recently, hierarchical maps have been introduced for large environment map establishments. Most of the previous map building was focused on 2D maps; however, in an underwater situation, the environment is unstructured, landmarks are sparse, and maps should be in 3D. Techniques to establish consistent 3D maps in these situation are still a big challenge.

1.3.3 Globally-Consistent 3D SLAM

The scenario of SLAM is briefly described as follows: a robot is placed in an unknown location and environment, and the robot must estimate its whereabouts and establish a map using only observations from dead-reckoning sensors, at the same time. The map produced will be used by the robot for navigation. This was a problem posed several years ago, and it remains a very active research topic in the mobile robot community. In real application, especially for the underwater environment, there are many challenges. The first is that the SLAM is 3D, which means it has 6 degrees of freedom (three translational and three rotational); this is much more difficult than SLAM in 2D, which has only 3 degrees of freedom (two translational and one rotational). The second is that the environment is not compatible with some sensors or algorithms. This will cause the SLAM to not be able to process correctly in this environment. For instance, if the sensor cannot observe any landmarks for several steps, the path is only estimated based on unreliable odometry information; if the sensor then observes some new landmarks in the following steps, these landmarks will be added to the map, but their positions are dependent on the unreliable path. In turn, these unreliable landmark positions will be used to estimate the subsequent robot path based on the sensor's observation. As a consequence, the path and 3D map are not reliable and consistent. This is one of the biggest challenges in SLAM.

1.4 Objectives

This thesis solves the SLAM problem faced by the underwater mobile robot. When applied in a real situation, SLAM can be divided into 2D path and 2D map estimation, 2D path and 3D map estimation, and 3D path and 3D map estimation. The first estimation is called 2D SLAM, the second 3D SLAM, and the last full 3D SLAM. In my thesis, I focus on the full 3D SLAM problem. In the proposed set of new algorithms, a globally-consistent 3D localization and map is achieved with multi-sensor fusion.

1.5 Contribution

The main contribution of this thesis is the development of a set of new algorithms for an autonomous mobile robot with full 3D SLAM ability by multi-sensor fusion. The interrelated problems the new algorithms solve are the following:

(1)***A system architecture designed for underwater mobile robot localization and mapping***. Based on the general SLAM method, a simple measurement system structure is developed by using a stereo camera and a set of range sensors to solve the 3D SLAM problem in an unknown environment, such as underwater. A complementary fusion algorithm takes advantages of two types of sensors: range sensor for long range, and camera for short range and map building.

(2) ***Registration uncertainty for robot self-localization in 3D***. An approach to obtain uncertainty of registration is proposed while the correspondence-based method is used during the process of robot pose estimation.

(3) ***Efficient map building in large areas***. By using set theory, a new algorithm for efficient map building is designed based on the overlap of measurements and the numbers of landmarks required for ego-motion estimation. The algorithm for the efficient map building is robust to withstand noise.

(4) ***Globally-consistent 3D SLAM by sensor fusion***. In natural environments, it is difficult to establish a global map with the measurements from a stereo camera since there may not be enough landmarks in certain areas. Measurements from a camera will solve the SLAM problem based on a local coordinate system, and the measurement from range sensor, such as buoys [1] or GPS, will be used to estimate a global path of the robot. A new algorithm integrates the global path and local SLAM results and generates a globally-consistent map and path for the mobile robot.

(5) ***A mobile robot simulation system designed for full 3D SLAM application***. In a virtual 3D robot working space, landmarks in 3D space or on the

[1]Buoy is a small boat which is equipped with many sensors such as GPS and hydrophone array.

floor can be preset, the robot speed and robot path are designed, sensor type (camera, laser, sonar, and radar) can be selected, and working environment (underwater, space, and land) can be defined, before a simulation starts. A robot would navigate in the virtual space and sensors measure the landmarks in its view field at all times. These measurements would be used to evaluate the algorithm for full 3D SLAM estimation.

1.6 Outline of the Thesis

This thesis consists of eight chapters. Besides the introduction in chapter 1, the other chapters will be briefly presented in the sequence.

Chapter 2 is the summary of previous work on the SLAM problem.

Chapter 3 introduces the sensors which are used to solve the 3D SLAM problem in this thesis. These sensors are stereo camera, GPS, and range finder (a set of buoys in water). The measurements from these sensors are used to estimate the position of the robot which is employed. Since these sensors are heterogenous, the algorithm used by each sensor is different. An important aspect of this chapter is that the method for registration uncertainty for mobile robot self-localization in 3D is proposed based on a stereo camera. Various results for each sensor type are also presented in this chapter.

Chapter 4 presents some general procedures and algorithms to solve the SLAM problem. This includes the problem definition, models which will be applied for 3D SLAM, data association mechanism, and the algorithm to solve general SLAM problems, such as particle filter and fast SLAM. A multi-sensor configuration is designed and an associated algorithm is developed to solve the full 3D SLAM problem for mobile robots, especially for underwater robots.

Chapter 5 presents the new algorithm designed for efficient map building. It starts with the problem definition of a real map for mobile robot, then gives the requirements for map establishment. Finally, three new algorithms designed for map building are introduced. Each of the algorithms is verified by simulation results.

Chapter 6 is the simulation of a mobile robot in 3D space in different environments with the general algorithms for SLAM. Before the implementation of simulation, this chapter briefly introduces the design and structure of a 3D mobile robot simulation system which is developed for 3D SLAM and other operations.

Chapter 7 presents experiments results in a water environment, robot lab, and outdoor environments. In the water environment, we focused on the sensor testing such as camera and buoys. In the robot lab, we did the experiment with an arm robot, and the data were applied to validate the SLAM algorithms in a controllable environment. The outdoor experiment took place in an unknown environment; the purpose of the experiment was to check the suitability of our developed algorithms in an unknown environment.

Chapter 8 discusses open issues which need to be addressed to make the system complete and robust. Finally, it discusses the results of this thesis.

Chapter 2

Background

SLAM includes two main tasks: robot localization and map building. Both of them are not new, but accurate solutions for both of them at the same time is a hot topic in the autonomous mobile robot community. In this chapter, a survey in this area is provided, which consists of four main topics: SLAM for autonomous mobile robot in section 2.1, SLAM for underwater mobile robot in section 2.2, maps used for mobile robot in section 2.3, and globally-consistent SLAM in section 2.4. The chapter is then summarized.

2.1 SLAM for Autonomous Mobile Robot

SLAM was started when the concept of stochastic map [79] was presented about twenty years ago. This map included not only the position of landmarks, but also the spatial relationships and their uncertainties, and the inter-dependencies of landmarks for mobile robots. A little later, a method to calculate the features' uncertainty by geometric transformation between coordinate systems with efficient and consistent ways was developed in the paper [24]. At the same time, another method was presented in [4] [11], which was a Kalman filter-based algorithm for mobile robot with visual sensor. Based on these contributions, the term Simultaneous Localization And Mapping (SLAM) was introduced [48]. Since then, this field has attracted a considerable amount of attention in the last ten years.

2.1.1 Model-based Robot SLAM

Model-based robot SLAM is one of the most widely used methods. It has three basic models: robot model, features model and sensor measurement model. During the robot navigation in an unknown environment, the measurement of the features will be integrated with the prediction of the position of the robot to estimate the optimal robot position and features position. Since the state vector includes both the robot state vector and the features state vector, the robot pose and features position can be estimated at the same time.

Based on previous work, the models which are used for SLAM have been explicitly explained in [19]. The solution for the SLAM problem has been proved in this paper such that the estimated map converges monotonically to a relative map with zero uncertainty, and the absolute accuracy of the map and the vehicle location reach a lower bound defined only by the initial vehicle uncertainty. It is also shown that a mobile robot can incrementally establish a perfect map of the environment and compute a bounded estimation of the robot location simultaneously, starting from an unknown location in an unknown environment.

Given the robot model, landmark model, and measurement model, the Kalman filter [70, 19, 79] can be used to construct an estimator of the robot and all the landmarks at every process time. In real application, the robot model, landmark model, and measurement model would not be linear equations. Therefore, the EKF [19, 79] is a solution using a linear approximation of the system to maintain a state vector containing the locations of the robot and landmarks, as well as an approximation of correlated uncertainty in the form of a covariance matrix. But the EKF would become prohibitive in large environments because of the growth of complexity due to the update step requiring computation time proportional to the square of the number of landmarks. In this case, the particle filter [32, 83] should be used.

Each of these feature-based methods has its advantages and disadvantages. However, the similarity among all is the increase in computational complexity with the size of the environment and the number of landmarks. Several techniques have been proposed to alleviate this problem, such as the unscented Kalman filter [43], fast-SLAM [60, 78], and submaps [74, 6]. Among all the above listed methods, fastSLAM is the most prominent and widely applied in the autonomous mobile robot community.

2.1.2 Vision-based SLAM

Vision-based robot localization also belongs to the feature-based method, where features extracted from an image are registered with a model by establishing correspondence between the two models. An example of line segments as the features which are extracted from images to register with the model was presented in [4]. It applied probabilistic predictions of feature locations and their uncertainties. In order to improve the estimate reliability, the features with small error covariance or features that could be matched with unambiguous were registered first. The uncertainty in the position of the other features could be decreased as soon as the robot pose was updated by the correctly matched features. Therefore, other features could be registered more reliably. The line segments as features from image to register for the robot navigation were also used in [29].

For the SLAM problem, the environment is unknown; therefore, there is no model which can be employed to be registered with the features extracted from images. In this case, if two consecutive images have enough overlap and corresponding matched features extracted from both of the images, the ego-motion of the robot equipped with a camera can be estimated by registering these corresponding image features.

Approaches for extracting motion information from image sequences can be classified as *correspondence-based* and *flow-based.* Correspondence-based methods [63, 59, 33, 72] track distinct features such as corners, lines, high curvature points, SIFT, etc., through the image sequence, and compute 3D structure by triangulation. Flow-based methods [14, 88, 55, 3] treat the image sequence as a function $f(x, y, t)$, where (x, y) are image pixel coordinates and t is time. It restricts the motion between frames to be small, and computes shape and motion in terms of differential changes in the function f.

In the correspondence-based method, the most important step is to register the two sets of data according to the correspondence. Once we obtain the correct registration parameter, we face the question of how precise it is. The least-squares minimization procedure to obtain the robot pose as well as its covariance was presented in [51]. Based on this method, a method for mobile robot localization and mapping with uncertainty by fusion of robot odometry information and visual estimation information with Kalman filter was applied in [73]. A very similar approach to solving the robot pose and its uncertainty problem with a single camera was introduced in [17].

Another way to estimate the robot pose uncertainty is the statistical analysis method which is based on the implicit function theorem [28, 14]. This method was used to derive a general strategy to analyze the propagation of measurement and calibration errors in [42] and to estimate the registration parameters and to reconstruct the model of the scene consisting of planar patches. The issues of sensitivity and robustness in their motion recovery algorithm from the image velocities was addressed in [54]. An error characterization of the factorization method for 3-D shape and motion recovery from image sequences using matrix perturbation theory was proposed in [81]. All of the above methods are applied to the flow-based method. Using the same idea, a sophisticated procedure to estimate robot's pose uncertainty was derived from a function of maximum likelihood in the paper [92], which is applied to the correspondence-based method.

If the registration can be completed between two consecutive images by using the previous methods, and assuming that the start position of the robot is the original point of a global frame, the robot pose in the global frame can be obtained by the ego-motion information incrementally [33, 90, 78]. With the robot's global pose information and features extracted from images, a map can be built simultaneously. For example, a volumetric 3D map of its workspace was established with the data structure of octree in [33]. Since most of the features are located in some small area while most of the other spaces lay empty, all the previous data structures cannot solve the efficient search problem for the map with a large number of features.

2.2 SLAM for Underwater Robot

Compared with the land and/or air-based mobile robot, the problems of autonomous underwater robots are similar in theory, but much more difficult in reality, as they are 3D problem with no available GPS. They share common navigation principles such as robot modelling, observation modelling, prediction and observation update. The difference among them is that there are no accurate odometric sensors and the robot models for the challenging water environment.

There are two approaches to the autonomous underwater vehicle (AUV) navigation. One of them is to use Doppler Velocity Logs (DVLs) to estimate vehicle position by dead reckoning. In order to mitigate the dilution in accuracy with mission time, integrated DVL with an inertial measurement unit (IMU) is adopted by most systems [46]. Another approach is to use acoustic ranging equipment to provide distance measurements to transponders at known locations [86]. This approach relies on GPS-aided calibration of pre-deployed acoustic transponders before the mission begins.

The Australian Centre for Field Robotics has developed a submersible vehicle that can be used for navigation and SLAM operation. The results of the application of a SLAM algorithm to estimate the motion of a submersible vehicle were presented in [87]. Scans obtained from an on-board sonar are processed to extract stable point features in the environment. These point features are then used to build a map of the environment while simultaneously providing an estimation of the vehicle location. Sonar and camera sensors for the SLAM problem are also used for this problem [57]. Blob-like patches were applied to separate feature information from background noise and other errors. By using multilevel feature spaces with the probability theory, a map built from many consecutive scenes is more reliable and the estimated path is more accurate than the results obtained by using a single sensor.

2.3 Maps for Mobile Robot

Maps are a very general tool for navigation and exploration. In the early human exploration, our ancestors had to build a map for their activities. Since this early time, maps played a very significant role, such as for airplane navigation, ship navigation, and travelling.

2.3.1 Geometric Map: Occupancy Grid Map

A geometric map represents the distance between obstacles or landmarks. It is a very accurate approach to describe the surrounding environment. In this case, a certain area is modelled as some geometric primitives.

The most common geometric map is an occupancy grid map in which the working space is divided into grid cells, and the value associated with each cell represents its degree of occupancy. Various methods have been used to generate occupancy grid maps, such as Bayesian [62, 12, 18] and fuzzy logic [26]. An advanced example of an occupancy grid map is the map of the Smithsonian museum with a successful track of over 2km [18]. While early occupancy grids are primarily two-dimensional, 3D occupancy grids have also been used [61, 71, 20].

In order to obtain an map, the grid should be fine, which is inefficient in terms of both storage and computation.

2.3.2 Geometric Map: Stochastic Feature Map

Another geometric map is a feature-based map. Leonard et al. [47] indicated that robot navigation requires a precise and concise feature-based map which can generate efficient prediction of what the robot should see from a given location. Many features, such as points, lines, line segments, planes, semi-planes, and corners, have been used in mobile robot navigation [10, 11, 15, 28].

The feature-based map is expressed by the stochastic map. A stochastic map consists of uncertain spatial relationships which are tied together in a representation [79]. It includes the spatial relationships and their uncertainties, and the inter-dependencies of landmarks. For instance, if there are two landmarks $l_1 = \{x_1, y_1\}$ and $l_2 = \{x_2, y_2\}$, their stochastic map is expressed by

$$L = \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{x}_2 \\ \tilde{y}_2 \end{bmatrix} \tag{2.3.2.1}$$

and

$$C(L) = \begin{bmatrix} \sigma_{x_1x_1} & \sigma_{x_1y_1} & \sigma_{x_1x_2} & \sigma_{x_1y_2} \\ \sigma_{y_1x_1} & \sigma_{y_1y_1} & \sigma_{y_1x_2} & \sigma_{y_1y_2} \\ \sigma_{x_2x_1} & \sigma_{x_2y_1} & \sigma_{x_2x_2} & \sigma_{x_2y_2} \\ \sigma_{y_2x_1} & \sigma_{y_1y_1} & \sigma_{y_2x_2} & \sigma_{y_2y_2} \end{bmatrix} = \begin{bmatrix} C(l_1) & C(l_1l_2) \\ C(l_1l_2) & C(l_2) \end{bmatrix} \tag{2.3.2.2}$$

where the diagonal elements in the covariance matrix are the variances of the spatial variables, and the off-diagonal elements are the covariances between the spatial variables, which indicate the interdependence of the landmarks. This expression for the feature uncertainty has greatly improved the implementation of transformation from one coordinate system to another among the sensors [24].

The key to the stochastic map is that it solves the problem of uncertainty for the map, especially for the autonomous mobile robot. The sensor installed on a robot is used to measure relative information between the robot and landmarks. No matter how accurate the sensor is, the measurement error always exists, therefore, the estimated landmark belongs to a certain type of distribution. The first moment of this distribution is the estimated location, and the second is its uncertainty.

2.3.3 Topological Map

A topological map is an abstraction of an area which omits details of the environment, but captures distinctive places or landmarks. This map only keeps elements of the area and their relative locations or paths to each other. It describes the area using a graph that connects distinctive places or landmarks in the world [23]. The vertices of the graph are the distinctive landmarks, and edges are the direct paths which are accessible. A topological map is expressed by

$$G = (V, E) \tag{2.3.3.1}$$

where V is a set of vertices which is denoted by

$$V = \begin{bmatrix} v_1 & \cdots & v_N \end{bmatrix}^T$$

and E is a set of edges which is denoted by

$$E = \begin{bmatrix} e_1 & \cdots & e_M \end{bmatrix}^T$$

The key issue for topological map is to identify landmarks or distinctive places in the area. In an indoor situation, the structured environment provides edges, corners, and frames for natural landmarks. While in some cases a set of artificial landmarks might be used, unfortunately, in the outdoor environment, there is no structure, and it is hard to add in artificial landmarks. In this case, designing and selecting an invariant landmark is a big challenge.

A topological SLAM approach in an indoor environment was presented in [13]. The researchers used sensors to scan the surroundings and located meet point as distinct point to construct a generalized Voronoi graph (GVG) which includes some metric information about the robot environment. The GVG is the best robot path for navigation to avoid obstacles, and it was used as a high-level topological map for navigation [49] because of its efficiency. This map was based on a set of low-level feature based maps.

Topological maps are very efficient for storage and operation. However, they require accurate obstacle sensing for navigation. Furthermore, robustly abstracting places and connections from sensor data is a non-trivial problem.

2.3.4 Hierarchical Map

The stochastic map [79] has been used very widely in the autonomous mobile robot community since its development in 1986. This map is built by estimation algorithms (Kalman filter [85] and particle filter [69, 34, 19]) as soon as observation data is obtained. If the number of landmarks in an environment is large, the map-building process will take a long time even though a very capable computer is applied. Therefore, hierarchical maps are introduced to solve this computational complexity problem.

The Atlas system [7] is an approach in which current small-scale mapping algorithms can be applied to obtain real-time performances in large-scale and cyclic environments. In this approach, there is no single global coordinate system while a set of interconnected local coordinate systems is maintained. Each of the local maps has a limited size. This interconnected local coordinate system forms a graph in which the vertices express the local map, and the edges express the transformation between connected local maps. Similar approaches to build independent local stochastic maps were also applied in [82] and [65]. These local maps are the vertices of a graph.

In order to build an efficient hierarchical map, three criteria to determine the ending of a current local map and the beginning of a new local map were defined as follows [25]:

- *"The total number of features or landmarks in the current local map reaches a threshold;*
- *The uncertainty of the estimated robot location with respect to the base reference of the current local map reaches a limit;*
- *The number of matchings of landmarks between two consecutive measurements found by data association is less than a model requirement for the last sensor measurements."*

An important issue for hierarchical maps is how to decide or estimate the position for a high-level map reliably. Loop-information in the robot path was used to optimize the global map to make it consistent; however, in the event that there is no loop in the robot path, the path error and map error would be unlimited.

2.4 Consistent SLAM

In the model-based SLAM approaches, the estimate locations and maps at step k only used the estimation of locations and maps at step $k-1$ and the measurement at step k. This estimation is a local optimal result. If a loop exists in the map, the estimate result will not be consistent. Therefore, a SLAM algorithm with the ability to solve the revisiting problem is very important [83, 53, 35, 45].

Several groups have proposed algorithms that modify estimates backwards when it is possible. Lu and Milios [53] studied consistent global estimation of a metric map constructed using laser range data. Their method maintained a history of all the local frames of sensor data used to construct the map and the network of spatial relations between the frames. The spatial relations were obtained either by odometry or pairwise matching of the laser range data in adjacent consecutive overlapping frames. A maximum likelihood algorithm was used to get a position estimate for each of the frames by minimizing the Mahalanobis distance between the actual and estimated relations over the entire network of frames. This method had been extended to build maps on the site with large cycles in a more computationally efficient manner [35].

Another approach is based on the probabilistic framework with the Baum-Welch algorithm for estimation of the revisiting problem [45]. A more generalized approach for mapping in the absence of prior information was presented in [76] . This approach uses local geometric information to disambiguate different locations. Both of the previous approaches built topological maps. A probabilistic approach to build large-scale maps of indoor cyclic environments with mobile robots was developed in [83], where an efficient algorithm was designed to estimate localization and mapping alternatively with the probabilistic constraints from the robot motion and sensor perception.

Image-based SLAM can use overlapped information and path loop information to solve the globally consistent mobile robot localization. A good example is the paper [44] in which constraints were established at every intersection of a mobile path by using large scale appearance image mosaics, and a global optimization processing was implemented. Another example for this approach is based on the two steps optimization [89]: multi-overlapped information for local optimization, and loop information for globally consistent estimation. A little different approach presented in [77] is by organizing a sequence of images acquired from well-separated locations which has limited prior odometric information. This approach employs a feature-based method derived from a probabilistic pose estimation framework.

All the previous SLAM methods are used for 2D cases. An approach for 3D SLAM from stereo vision in an indoor environment was introduced in [72] and [33]. The researchers used the loop information to establish a globally consistent map, but this map is only in 2D plan because of its simplicity.

2.5 Summary

SLAM problem has been solved successfully in different situations at an indoor environment by many researchers. In outdoor environments, there are also many positive achievements in many cases. In order to make sure that the algorithms can be used in real application, such as in an underwater environment, there are still a few things that need to be done: (1) globally-consistent 3D SLAM; (2) efficient map building; (3) best system architecture for underwater mobile robot. These issues will be discussed and solved later on in this thesis.

Chapter 3

Measurement from Sensors

During a robot navigation process, when the sensor installed on the robot observes its environment and obtains some measurement, it should be able to estimate robot pose and landmark position immediately based on the SLAM strategies and algorithms in chapter 4. Different sensors will provide measurements that are different functions of the robot pose. How to use these different sensors for SLAM is the main topic of this chapter.

In this chapter, stereo camera, GPS, and range finder will be introduced, all of which will be applied for a large area consistent robot SLAM. The content of this chapter include the sensor structure, measurement property, feature extraction, and related models. This chapter is arranged as follows: section 3.1 is the trinocular camera, section 3.2 is the GPS, section 3.3 is the range measurement finder. Last, a summary of this chapter is presented.

3.1 Camera

Stereo camera is a popular sensor which can be used by a robot for its navigation. The advantage of using a camera is that it can obtain 3D information about the environment. During the stereo image matching step between the left image features and right image features, if the matching is incorrect, the 3D information will not be correct for the 3D points corresponding to the mismatched features. In order to increase the camera system's reliability, a trinocular camera will be used in this thesis.

For a stereo or trinocular camera used for robot localization, there are three types of correspondence which will be applied in the following section. First, correspondence of features of the left, right and top image, at a particular time, giving a set of 3D points for that time; second, correspondence of 3D points from different times,

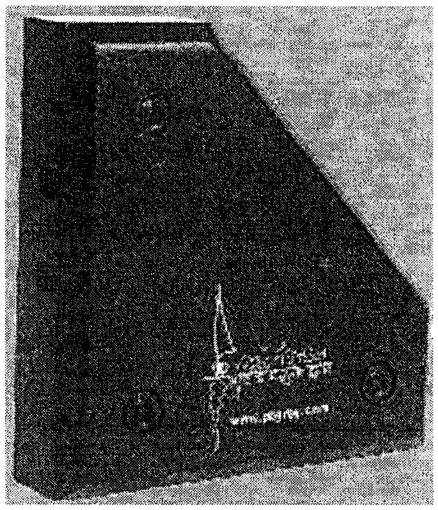


Figure 3.1: The trinocular camera system developed by Point Grey Research.

assuming small movement of the robot, and therefore high overlap between 3D points obtained at those times; third, features of images at different times can be matched.

3.1.1 Introduction of Trinocular Camera

A trinocular camera system consists of three standard cameras which are aligned so as to have overlapping fields of view. A widely used trinocular system in robotic area is the Triclops developed by Point Grey Research (Fig. 3.1). The Triclops has several features that make it versatile and easy to use. The image transmission to the host computer is completely digital via IEEE-1394, resulting in a no frame grabber jitter or analog-to-digital conversion noise. All images are synchronized internally, removing sources of uncertainty in multiple camera systems. It is completely controllable through a C/C++ API that allows quick prototyping and integration into computer vision applications.

The relationship of the three lenses in a trinocular system is shown in Fig. 3.2. In a well-calibrated trinocular system, it is possible to assume that the cameras have parallel image planes, aligned with epipolar lines. The position of a point (x, y, z) in a scene can be determined through triangulation based on corresponding points in the images, $p_t(x_t, y_t)$, $p_r(x_r, y_r)$ and $p_l(x_l, y_l)$, obtained by the top, right, and left cameras.

$$x = b_h x_l/(x_l - x_r) \tag{3.1.1.1}$$

$$y = b_h y_l/(x_l - x_r) \tag{3.1.1.2}$$

$$z = b_h f/(x_l - x_r) \tag{3.1.1.3}$$

where b_h and b_v are horizontal and vertical baseline displacement between the camera respectively. f is the focal length of the cameras.

Assume that the measured image coordinates in 2D and inferred 3D points by triangulation from stereo images have normally distributed (Gaussian) errors. The Gaussian distribution model to express the error of image coordinates is a common and convenient approximation that will give an adequate performance [8]. The true distribution of the 3D points is non-Gaussian because triangulation is a nonlinear operation [59]. If the distance from the camera to the point is not extreme, a Gaussian distribution for 3D is an acceptable approximation.

Due to the error in image coordinates, the covariance for the 3D point, P_v ($p = f(p_l, p_r)$), will be

$$P_v = J \begin{bmatrix} cov(p_l) & 0 \\ 0 & cov(p_r) \end{bmatrix} J^T \tag{3.1.1.4}$$

where J is the Jacobian matrix of first partial derivatives of p respecting to the random vectors p_l and p_r. It is expressed as

$$J = \begin{pmatrix} -b_h x_r/d^2 & 0 & b_h x_l/d^2 & 0 \\ -b_h y_l/d^2 & b/d & b_h y_l/d^2 & 0 \\ -b_h f/d^2 & 0 & b_h f/d^2 & 0 \end{pmatrix} \tag{3.1.1.5}$$

where $d = x_l - x_r$. Usually $cov(p_l) = [\sigma_x, 0; 0, \sigma_y]$, σ_x is in the range from 0.01 to 0.5 and σ_y is in the range from 0.01 to 0.5; and $cov(p_r) = cov(p_l)$.

By using a trinocular camera system, three images (left, right, and top images) can be obtained. The disparity among the images can be calculated by the correlation between the top and right images and between the right and left images. The 3D

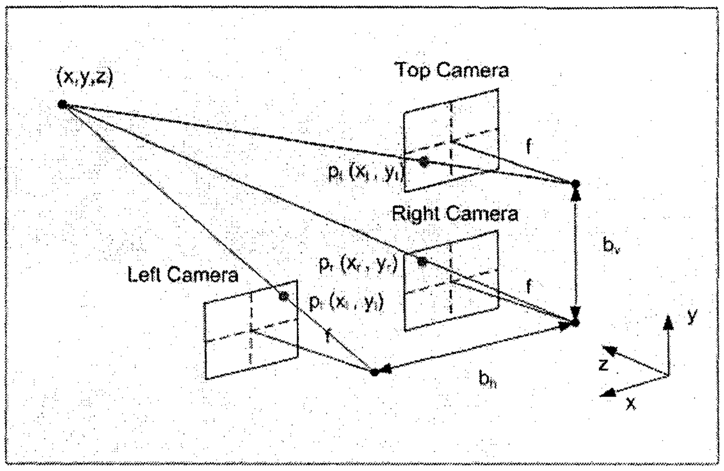


Figure 3.2: The relationship of the three lenses in a trinocular system. where b_v is the base distance between top camera and right camera, b_h is the base distance between right camera and left camera; p_t, p_r, and p_l are the image points of the real world point $p_3(x, y, z)$ in top, right, and left camera images, respectively.

information is called 3D cloud, which is a set of estimated 3D points corresponding to matched features in the three images and it can be obtained using the disparity and camera model. Therefore, three images and one 3D cloud file will be output from the trinocular camera system for its real application. The correspondence of features used here belongs to the first type.

3.1.2 Robot Pose Estimation with Stereo Camera

From the trinocular stereo camera system, three overlapping images can be obtained every time. By using the relationship of the three lenses in Fig. 3.2, a set of 3D points (or 3D cloud) can be calculated. If the images taken from two consecutive steps have enough overlap, then corresponding points between the two consecutive images can be established (this is the second type of the feature's correspondence). Therefore, 3D points of these images' points can be extracted easily from their associated 3D cloud. These two sets of associated 3D points can be used to estimate the robot's movement by data registration.

Assume a rectified image I_k, in which the distortion in the original image has been corrected, also, a 3D cloud C_k, which is a set of 3D data of the environment in the camera view field, are obtained at time k, for all k. From the image I_k, a set of SIFT features can be extracted. In a sequence of images, SIFT feature correspondence can be established between any two consecutive images. RANSAC method [31] is implemented to delete the outliers in the previous initial matching. For any matched features, its 3D points could be obtained from its associated 3D cloud. Since the SIFT feature gives a position in the image with sub-pixel accuracy, it is necessary to use bilinear interpolation to obtain an accurate 3D position. Therefore, it is possible to obtain, for a 3D point p_k^i in C_k, a corresponding 3D point $p_{k-1}^i \in R^3, i = 1, \ldots, n$, in C_{k-1}. Since the two 3D point data sets are derived from different image frames, their covariance will change with the depth. Hence, the corresponding points will have different error covariances. We assume that the error for every point i at time k can be expressed as $\sigma_{p_k^i}$.

In order to obtain the robot movement information between time step $k-1$ and k, Gaussian based Maximum Likelihood (ML) method can be used in this problem [2, 59]. We can directly register the cloud C_{k-1} to cloud C_k by using the corresponding 3D point data p_{k-1}^i and p_k^i. Ideally, by some rotation R_k and translation T_k for the robot, the 3D point p_{k-1}^i should be at the position of the 3D point p_k^i. Due to the errors from camera sensor and data processing, they will not be in the same position. The difference of the position is expressed by

$$\nu_i = p_k^i - R_k p_{k-1}^i - T_k \tag{3.1.2.1}$$

where i is in $1, \cdots, n$.

First, the 3D points p_k^i and p_k^j $(i, j = 1, \cdots, n$ and $i \neq j)$ are extracted from image corresponding SIFT features in the image, which are assumed independent. Therefore, the difference of the position ν_i and ν_j are mutually independent. Following the standard assumption that the measured positions of SIFT image features in the images are zero-mean Gaussian, the linearity of Eq. (3.1.2.1) justifies the assumption that v_i is zero-mean. Based on the Gaussian distribution and mutual independence of ν_i, the approach of ML is equivalent to minimizing the following Mahalanobis distance. Therefore, we obtain the optimization formula:

$$\min_{R_k,T_k} \quad E(R_k,T_k) = \sum_{i=1}^{n} \nu_i^T S_i^{-1} \nu_i \tag{3.1.2.2}$$

where $S_i = R_k \sigma_{p_{k-1}^i} R_k^T + \sigma_{p_k^i}$ is its associated covariance.

Before the registration process, the centroid of the two sets of data can be acquired as $p_k^c = \sum_{i=1}^{n} p_k^i/n$ and $p_{k-1}^c = \sum_{i=1}^{n} p_{k-1}^i/n$. By subtracting the centroid from each point, two new data sets $\hat{p}_k^i = p_k^i - p_k^c$ and $\hat{p}_{k-1}^i = p_{k-1}^i - p_{k-1}^c$ can be obtained. Substituting these data sets $\hat{p}_k^i$ and $\hat{p}_{k-1}^i$ into the innovation ν_i, the objective function, Eq. (3.1.2.2), can be changed to

$$\min_{R_k} \quad E = \sum_{i=1}^{n} (\hat{p}_k^i - R_k \hat{p}_{k-1}^i)^T S_i^{-1} (\hat{p}_k^i - R_k \hat{p}_{k-1}^i) \tag{3.1.2.3}$$

This is a non-linear function. Optimized solution can be calculated through linearization and iteration, which was used by Olson *et al.* [66]. To simplify E, the rotation R_k is expressed in the form of quaternion $R_k = R(q_k)$ and $q_k = (q_0^k, q_1^k, q_2^k, q_3^k)$. The objective function is linearized by taking first-order expansion with respect to the rotation in the quaternion expression. Suppose q_0 is the initial rotation estimate and R_0 is the corresponding rotation matrix, the first-order expansion of Eq. (3.1.2.3) is:

$$E = \sum_{i=1}^{n} (G_k^i - J_k^i q_k)^T S_i^{-1} (G_k^i - J_k^i q_k) \tag{3.1.2.4}$$

where $J_k^i = [\frac{\partial R_k}{\partial q_0^k}\hat{p}_{k-1}^i, \frac{\partial R_k}{\partial q_1^k}\hat{p}_{k-1}^i, \frac{\partial R_k}{\partial q_2^k}\hat{p}_{k-1}^i, \frac{\partial R_k}{\partial q_3^k}\hat{p}_{k-1}^i]$ (see Appendix A), and $G_k^i = \hat{P}_k^i - R_0 \hat{P}_{k-1}^i - J_k^i q_0^k$. Differentiating the objective function with respect to q_k and setting the derivatives to zero yields a closed form solution for q_k:

$$q_k = \sum_{i=1}^{n} ({J_k^i}^T S_i^{-1} J_k^i)^{-1} \sum_{i=1}^{n} ({J_k^i}^T S_i^{-1} G_k^i) \tag{3.1.2.5}$$

After acquiring the solution of Eq. (3.1.2.5), this estimated rotation is used as an initial estimate of the next step, and the process is iterated until it converges. Then the translation can be obtained by

$$T_k = R_k p_{k-1}^c - p_k^c \tag{3.1.2.6}$$

3.1.3 Uncertainty of Robot Movement Estimation

From the objective function Eq. (3.1.2.4), it is obvious that the rotation q_k and the measurement M_k $(= [p^i_{k-1}, p^i_k]^T)$, $(i = 1, \ldots, N)$ are related through an implicit function.

$$\Psi(q_k, M_k) = 0 \tag{3.1.3.1}$$

According to the implicit function theorem [28, 30], we get

$$\frac{\partial q_k}{\partial M_k} = -(\frac{\partial \Psi}{\partial q_k})^{-1} \frac{\partial \Psi}{\partial M_k} \tag{3.1.3.2}$$

where the rotation is expressed as a function of the measurements

$$q_k = f(M_k) \tag{3.1.3.3}$$

Expanding f as a Taylor series around $E[M_k]$ yields

$$f(M_k) = f(E[M_k]) + (M_k - E[M_k])\frac{\partial q_k}{\partial M_k} + O(M_k - E[M_k])^2 \tag{3.1.3.4}$$

where $O(.)^2$ denotes terms of order 2 or higher in M_k and $\frac{\partial q_k}{\partial M_k}$. Up to a first-order approximation, then covariance of q_k can be obtained as

$$\sigma_{q_k} = \frac{\partial q_k}{\partial M_k} \sigma_{M_k} (\frac{\partial q_k}{\partial M_k})^T \tag{3.1.3.5}$$

If we define $\Psi = \frac{\partial E^T}{\partial q_k}$, then we will have

$$\frac{\partial \Psi}{\partial q_k} = \frac{\partial^2 E}{\partial q_k^2} \quad and \quad \frac{\partial \Psi}{\partial M_k} = \frac{\partial^2 E}{\partial M_k \partial q_k} \tag{3.1.3.6}$$

Substituting Eq. (3.1.3.6) into Eq. (3.1.3.5), the covariance σ_{q_k} will be

$$\sigma_{q_k} = (\frac{\partial^2 E}{\partial q_k^2})^{-1} (\frac{\partial^2 E}{\partial M_k \partial q_k}) \sigma_{M_k} (\frac{\partial^2 E}{\partial M_k \partial q_k})^T (\frac{\partial^2 E}{\partial q_k^2})^{-T} \tag{3.1.3.7}$$

From the definition of the objective function (Eq.(3.1.2.4)), we have

$$\frac{\partial E}{\partial q_k} = \sum_{i=1}^{n} (-\frac{\partial R_k}{\partial q_k} \hat{p}^i_{k-1})^T S_i^{-1} (\hat{p}^i_k - R_k \hat{p}^i_{k-1}) \tag{3.1.3.8}$$

$$\begin{aligned} \frac{\partial^2 E}{\partial q^k_s \partial q^k_r} = &\sum_{i=1}^{n} \Big[(-\frac{\partial R^2_k}{\partial q^k_s \partial q^k_r} \hat{p}^i_{k-1})^T S_i^{-1} (\hat{p}^i_k - R_k \hat{p}^i_{k-1}) \\ &+ (\frac{\partial R_k}{\partial q^k_s} \hat{p}^i_{k-1})^T S_i^{-1} \frac{\partial R_k}{\partial q^k_r} \hat{p}^i_{k-1} \Big] \end{aligned} \tag{3.1.3.9}$$

where s and r equal to $0, 1, 2, 3$.

$$\frac{\partial^2 E}{\partial p^i_{k-1} \partial q^k_s} = (-\frac{\partial R_k}{\partial q^k_s})^T S_i^{-1}(\hat{p}^i_k - R_k \hat{p}^i_{k-1}) + (\frac{\partial R_k}{\partial q^k_s} \hat{p}^i_{k-1})^T S_i^{-1} R_k \quad (3.1.3.10)$$

$$\frac{\partial^2 E}{\partial p^i_k \partial q^k_s} = (-\frac{\partial R_k}{\partial q^k_s} \hat{p}^i_{k-1})^T S_i^{-1} \quad (3.1.3.11)$$

$$\frac{\partial^2 E}{\partial p^i_k \partial q^k_s} = \begin{pmatrix} \frac{\partial^2 E}{\partial p^i_{k-1} \partial q^k_s} & \frac{\partial^2 E}{\partial p^i_k \partial q^k_s} \end{pmatrix} \quad (3.1.3.12)$$

where $i = 1, \cdots, n$ and $s = 0, 1, 2, 3$. Substituting the Eq. (3.1.3.9) and Eq. (3.1.3.12) into Eq. (3.1.3.7), the covariance of q_k can be obtained. The covariance for the translation can be calculated by

$$\sigma_{T_k} = \sigma^c_{p_{k-1}} + R_k \sigma_{p^c_k} R^T_k \quad (3.1.3.13)$$

where $\sigma_{p^c_{k-1}}$ and $\sigma_{p^c_k}$ are the covariance of the centre point of data set p_{k-1} and p_k, respectively.

Input: two adjacent images I_{k-1} and I_k, and their $3D$ cloud C_{k-1} and C_k and Image pixel error

Output: transformation q_k, T_k; variance σ_{q_k}, σ_{T_k}

```
1: Extract SIFT feature from images $I_{k-1}$, $I_k$
2: Establish feature correspondences
3: Implement RANSAC to delete outliers
4: Estimate robot pose by Eq. (3.1.2.5) and Eq. (3.1.2.6)
5: Calculate rotation uncertainty by Eq. (3.1.3.7)
6: Calculate translation uncertainty by Eq. (3.1.3.13)
7: Output $q_k$, $T_k$, and $\sigma_{q_k}$ and $\sigma_{T_k}$
```

Algorithm 1: Algorithm for 3D data registration and its uncertainty

The computational complexity of calculating the uncertainty of robot movement estimation (step 5 and step 6) in this algorithm is $O(n)$, where n is the number of corresponding points. The algorithm for the registration of 3D data and its uncertainty estimation is shown in Alg. 1.

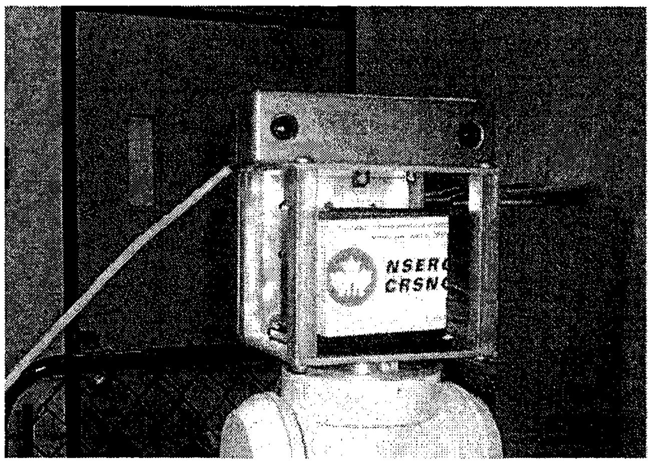


Figure 3.3: The Bumblebee stereo camera from Point Grey Research mounted at the tip of the PA10-7CE robot arm.

3.1.4 Lab Experiment

The lab experiment was performed with a BumbleBee camera system from Point Grey Research mounted on a Mitsubishi PA10-7CE Robot arm. The robot arm has a maximum speed of 3.33 meters per second and a payload of 10 kilograms (Fig. 3.3). The camera connects via an IEEE 1394 link to a PC. The stereo camera captures two 320 × 240 color images when the robot is stationary. The raw image which was distorted by the camera was corrected by implementation of some algorithms automatically provided by the company. The output is a set of rectified color images and a list of 3D cloud points associated with its rectified pixels. Points farther than four meters apart are discarded during the stereo processing in this experimental environment.

Self-localization Estimation

During the lab experiment, no artificial landmarks were used. The features used in this paper are SIFT features [52], which are extracted from the image in every step. Two adjacent images are matched for robot self-localization. RANSAC method [31] was used to delete the outliers. From the matched image points, their associated 3D points could be obtained from the associated 3D cloud. After this processing, two sets of 3D points which are matched correctly (with some error tolerance) were acquired, then the data registration was implemented to get rotation R and translation T.

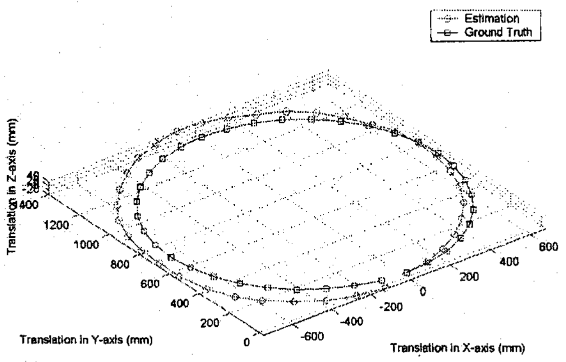


Figure 3.4: Robot trajectory displayed in 3D. Ground truth is obtained by the internal position sensors of the robot arm.

Suppose that the robot's start position is p_0, and the translation at time k is T_k, and rotation is R_k, then the absolute position of the robot can be obtained by

$$p_k = p_{k-1} + R_k * T_k \tag{3.1.4.1}$$

where $k = 1, \cdots, N$, and N is the number of measurements in the circle. The robot's built-in high-precision positioning system provides ground truth of the robot motion trajectory. The estimated trajectory in 3D is shown in Fig. 3.4 The image-based self-localization estimation is a 6 DOF problem. In our experiment, we observe that even though the robot moves in a plane, the estimated trajectory is not planar (Fig. 3.4) due to the errors in the process of estimation.

3.1.5 Analysis of Uncertainty

The uncertainty of robot localization in every step can be obtained together with the robot's pose estimation by using Eq. (3.1.3.7) and Eq. (3.1.3.13). For the rotation, the uncertainty is a 4×4 matrix since the rotation is expressed as quaternion during the pose estimation. And the uncertainty of translation is a 3×3 matrix, which can be expressed graphically with ellipsoid [16]. In our experiment, we only took the x-y plane to show the estimation uncertainty with the associated ellipse in every robot position. For example, the small ellipses are the estimated uncertainty in every robot position in Fig. 3.5. We knew from the results that in every step, the location

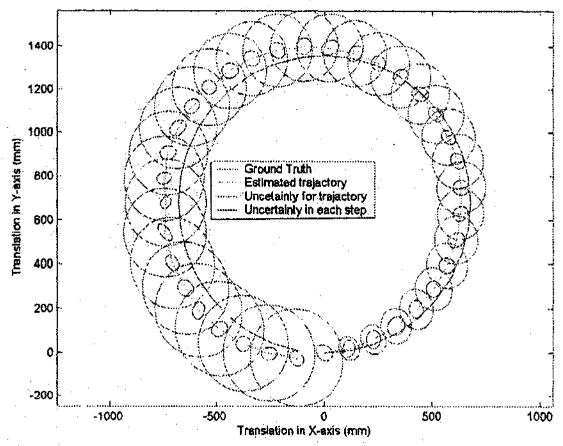


Figure 3.5: Estimated robot trajectory in x-y plane with associated uncertainty in test case 1.

uncertainties are almost similar, but their directions change in different positions.

The robot trajectory calculation by Eq. (3.1.4.1) is an iterative process. The uncertainty of absolute position can be calculated as

$$\sigma_{p_k} = \sigma_{p_{k-1}} + R_k \sigma_{T_k} R_k^T \tag{3.1.5.1}$$

where $k = 1, \cdots, N$. We assumed that the σ_{p_0} equals to I_3 and I_3 is an identical matrix with a dimension of 3. Therefore, the uncertainty of the robot position increases with time. This is displayed as ellipses, as shown in Fig. 3.5. In order to control the uncertainty growth with time, some advanced algorithms should be used. This will be discussed in Chapter 6 and Chapter 7.

Another experiment result was shown in Fig. 3.6. In this case, we took more images in the circular trajectory of the robot than in the previous case. The uncertainty in every robot position (small ellipses in Fig. 3.6) are almost similar to that of Fig. 3.5, since both of the experiments had the same environment set-up and used the same camera with the same calibration parameters. But the uncertainty of the robot position is larger than in Fig. 3.5. This resembles a random walk process whose standard deviation grows with the square root of the number of measurements added. As stated previously, this kind of error can be reduced by using advanced algorithms.

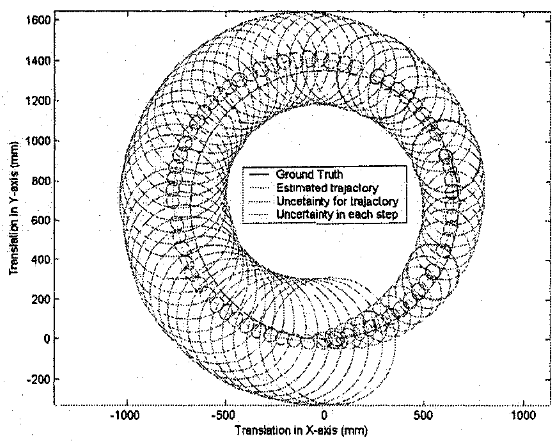


Figure 3.6: Estimated robot trajectory in x-y plane with associated uncertainty, where the density of sampling along the circular trajectory is higher than in test case in Fig. 3.5.

3.1.6 Discussion for Stereo Camera

From the experiment with the stereo camera system, we can see that it is possible to estimate robot position from the measurement of the stereo camera only if enough corresponding matched features can be extracted from overlapping consecutive images. In this estimation step, we did not use any artificial information, and we only used the relative measurement information from two consecutive measurements. It must be pointed out that the position estimation will fail if a sufficient number (usually at least is 7) of matched features cannot be found for two consecutive images. In order to solve this problem in this situation, a new algorithm will be discussed in Chapter 4.

Another issue is the uncertainty of the robot estimation. In the same working area, we can slow down the movement of the robot so that we can obtain more images over the same path. This will increase the overlap between any two adjacent images, and more numbers of matched features may be extracted. As a result, the uncertainty of the estimation should be improved (the value of the uncertainty should be small) for each step. However, from the experiment results, the uncertainty of the estimation for the robot path is increased with more images due to the accumulation of the errors. To avoid excessive big errors during the estimation, the multi-map technique will be used in Chapter 4 and Chapter 7, where a new estimation will be started as soon as the estimation error reaches certain threshold values.

3.2 GPS

The rapidly expanding use of the Global Positioning System (GPS) enables commercial navigation devices to be more popular and attainable for non-military users. GPS provides the absolute positioning information covering any part of the earth, in day and night. However, the visibility from the recipient to the number of GPS satellites is still critical in using GPS alone as a navigation device, such as under trees, inside buildings, in tunnels, between tall buildings, and under water [67]. Therefore, for a reliable navigation system, other types of sensors are needed as a complementary measurement device.

3.2.1 GPS Principles

The NAVSTAR GPS (NAVigation System with Time and Ranging Global Positioning System) is a satellite-based radio navigation system designed and operated by the U.S. DoD (Department of Defense). It provides three-dimensional position, navigation, and time information to the users with proper equipment. It became fully operational in 1994 with 21 satellites (plus 3 active spares) on 6 orbital planes within approximately 20,200 km altitude above the earth's surface with a 12-hour orbiting period worldwide. GPS has been designed so that at least 4 satellites can be available visibly above the horizon anywhere on the earth, 24 hours a day.

GPS is primarily a navigation system. The fundamental navigation principle is based on the measurement of so-called pseudoranges between the user and at least the four satellites. Starting from the known satellite coordinates in a suitable reference frame (WGS 84), the coordinates of the user's antenna can be determined. From the geometrical point of view, three range measurements are sufficient. A fourth observation is necessary because GPS uses the one-way ranging technique, and the receiver clock is not synchronized with the satellite clock. This synchronization error is the reason for the term pseudoranges [67].

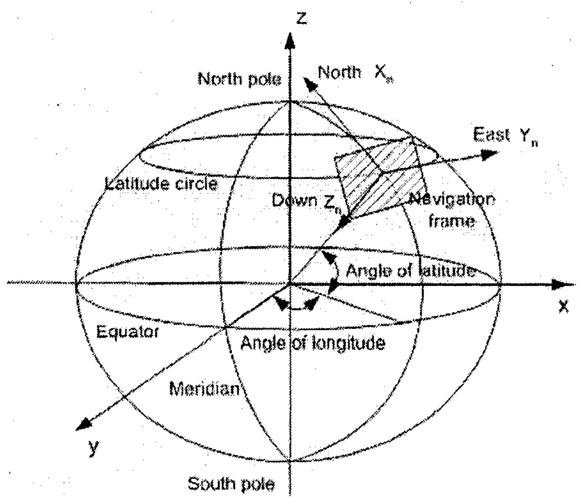


Figure 3.7: Frames for robot navigation. Geodetic frame (λ,φ,h), where λ is the angle of longitude, φ is the angle of latitude, and h is the altitude (the height from the earth surface to the interesting point in space in direction from the centre of earth to the interesting point.); Earth frame (x, y, z); and navigation frame (x_n, y_n, z_n).

3.2.2 GPS Measurements

As mentioned previously, at least 4 satellite measurements are acquired to determine the recipient position slaved to the coordinate frame of reference such as WGS 84. Usually, a GPS Receiver receives positional data from satellites orbiting the Earth. Using this data, the GPS Receiver can provide latitude, longitude, signal strength, number of satellites observed, deviations, satellite information, and ground speed. The GPS logger outputs this information in a variety of NMEA 0183 "sentences" containing different types of information.

The "sentences" used by this GPS Receiver are: GGA (Global Positioning System Fix Data), GSA (GPS DOP and Active Satellites), GSV (GPS SVs (Satellite Vehicles) in View), RMC (Recommended Minimum Specific GPS/TRANSIT Data), and VTG (Track Made Good and Groundspeed). Since the positioning information of an object is concerned, GGA sentence is used to obtain all the necessary data, such as longitude, latitude, and height in a geodetic coordinate system in earth-frame. These coordinates should be transformed to a local fixed navigation-frame for the AUV navigation application. The details of GGA are explained in Appendix B.

Assume the GPS receiver obtains a point in geodetic coordinate systems, such as (λ, φ, h). Its associated coordinates (x, y, z) in a rectangular coordinate system in earth-frame can be obtained by

$$\begin{aligned} x =&(N+h)cos(\lambda)cos(\varphi) \\ y =&(N+h)cos(\lambda)sin(\varphi) \\ z =&[N(1-e^2)+h)]sin(\lambda) \end{aligned} \tag{3.2.2.1}$$

where $e = \sqrt{f(2-f)}$, $f = (a-b)/a)$, and $N(\lambda) = a/\sqrt{1-e^2 sin(\lambda)^2}$. According to the WGS-84 ellipsoid [1] definition, a is the length of semimajor axis($a = 6378137.0m$), and b is the length of semiminor axis ($b = 6356752.3142m$). The definition for the frames which are used in this thesis is shown in Fig. 3.7 [27].

For a batch of GPS data $\lambda_i, \varphi_i, h_i$ or (x_i, y_i, z_i), $i = 1, \cdots, n$, if we take the x_1, y_1, z_1 in earth-frame as the origin of a local navigation-frame, the coordinate of $(x_n(i), y_e(i), z_d(i))$ in navigation-frame can be obtained by

$$\begin{pmatrix} x_n(i) \\ y_e(i) \\ z_d(i) \end{pmatrix} = \begin{pmatrix} -sin(\lambda_1)cos(\phi_1) & -sin(\lambda_1)sin(\phi_1) & cos(\lambda_1) \\ -sin(\phi) & cos(\phi) & 0 \\ -cos(\lambda_1)cos(\phi_1) & -cos(\lambda_1)sin(\phi_1) & -sin(\lambda_1) \end{pmatrix} \begin{pmatrix} x_i - x_1 \\ y_i - y_1 \\ z_i - z_1 \end{pmatrix} \tag{3.2.2.2}$$

Here is an example of measurements at the Halifax Citadel with GPS. This is a very high altitude area in Halifax. At every point, the GPS could receive signals from seven satellites. The measurement in the geodetic coordinate system is shown in Fig. 3.8. These data were transformed to a local navigation-frame, and the result is displayed in Fig. 3.9.

3.3 Range Finder

There are several range finders which can be applied to different application areas. In the underwater environment, acoustic sensors such as sonar and array of hydrophones are the choice for range measurement. On land or in a space environment, radar and laser range finders are suitable for range measurement. A range finder installed on a robot is used to measure the distance from the robot to a set of landmarks. If

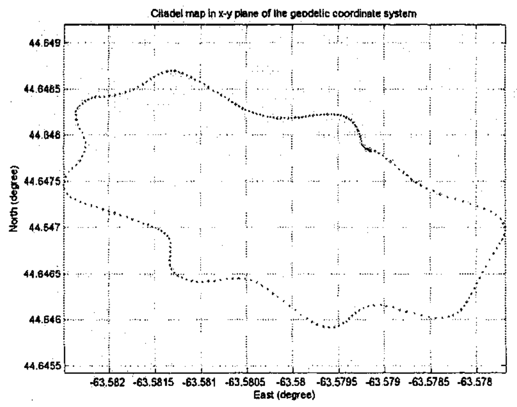


Figure 3.8: The robot path obtained from the Halifax Citadel in geodetic coordinate system.

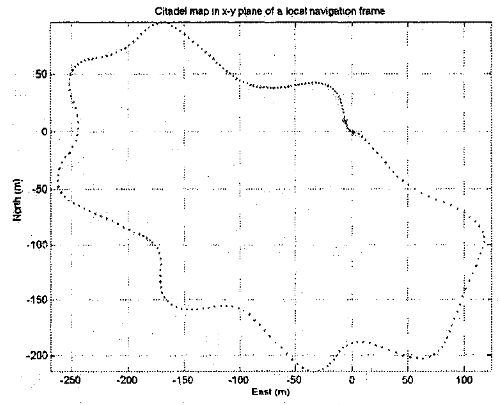


Figure 3.9: The robot path obtained from the Halifax Citadel in local fixed navigation frame.

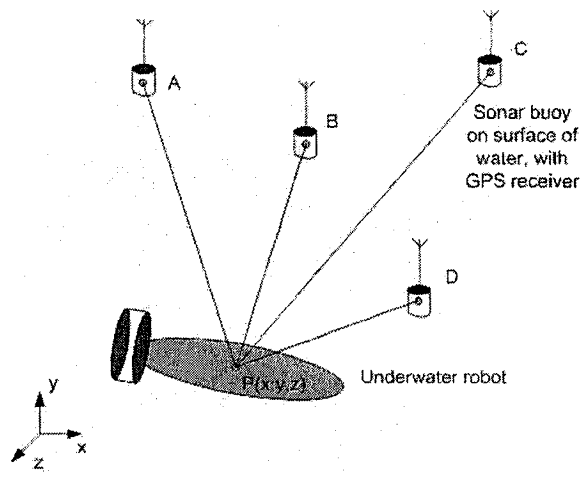


Figure 3.10: **Underwater robot localization by range finder.** A, B, C, and D are buoys which are equipped with a GPS receiver. The robot is equipped with sonar to measure the range from the robot to all the buoys.

the positions of the landmarks in a world-centered coordinate system are known or have been estimated, the problem is to estimate the robot pose and the estimate's reliability..

An instance of a robot position estimation in an underwater environment is shown in Fig. 3.10. We have assumed that there are m buoys (features) $L_i(x_i, y_i, z_i)$ $(i = 1, \cdots, m)$ and the measurement from each buoy to the robot $(p(x, y, z))$ is Z_i, then the measurement model is

$$Z_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} \tag{3.3.0.3}$$

It is possible to obtain the position of the robot by solving a set of non-linear equations from the Eq. (3.3.0.3), if there are more than 3 measurements. Many methods such as the Newton Method can be used for this solution, but the question remains as to how reliable this solution is.

Probability based on sensor fusion mechanism can be used for this problem. In this case, the process model is

$$X_v(k+1) = X_v(k) \tag{3.3.0.4}$$

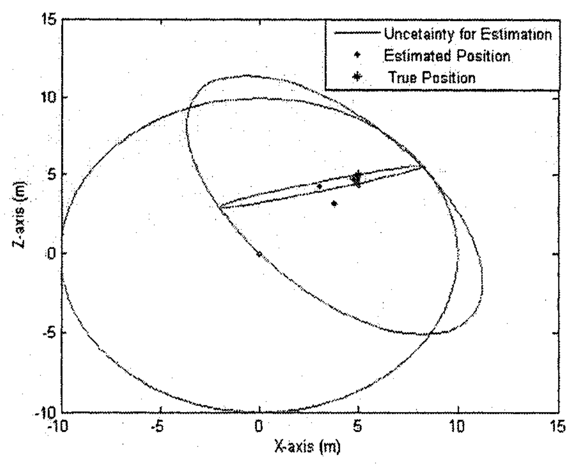


Figure 3.11: Uncertainty of measurement and its related estimation by Extended Kalman Filter.

and the measurement model is

$$Z_i(k+1) = h(X_v(k+1), L_i(k+1)) = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} \quad (3.3.0.5)$$

Since the measurement is a non-linear model, the Kalman filter can not be applied directly; therefore, the Extended Kalman filter is required.

The Jacobian of the measurement function (Eq.(3.3.0.5)) is

$$\begin{aligned} J_h &= \frac{\partial h}{\partial X_v} \\ &= \begin{pmatrix} \frac{\partial h}{\partial x} & \frac{\partial h}{\partial y} & \frac{\partial h}{\partial z} \end{pmatrix} \\ &= \begin{pmatrix} \frac{x - x_i}{r} & \frac{y - y_i}{r} & \frac{z - z_i}{r} \end{pmatrix} \end{aligned} \quad (3.3.0.6)$$

where $r = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}$. The Monte Carlo simulation result is shown in Fig. 3.11 and Fig. 3.12. The ellipse is the estimation error in Fig. 3.11, and its centre is the robot location estimated. During the simulation, the robot location converges to a point with very small estimation error. From the upper left graph in Fig. 3.12, it is easy to see that as soon as five measurements are obtained, a robust robot position estimation is possible by EKF.

The GPS and range sensor used in this thesis can only be used to determine the position of the robot, not its orientation. It is possible to estimate the robot's

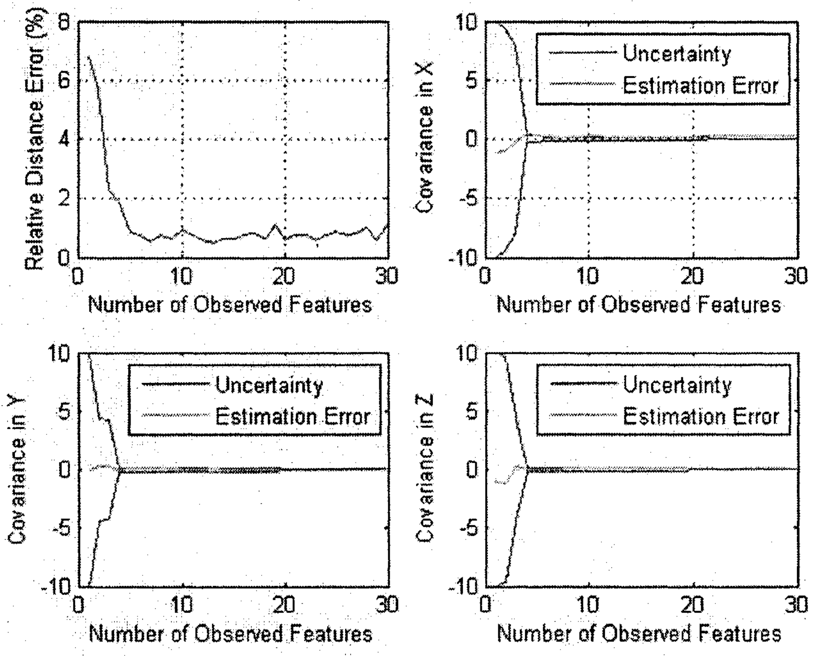


Figure 3.12: Estimation error via the number of features by Extended Kalman Filter. The uncertainty is the error estimated by the algorithm and the estimation error is the error from actual measurement in this simulation.

orientation by using the components of Jacobian during the linear processing for the non-linear measurement function, which is based on the assumption that the robot model permits motions only in the direction in which the robot is pointing. For the underwater mobile robot, the robot will move not only in the direction in which the robot is pointing, but also in other directions because of environmental factors such as water currents.

3.4 Summary

The three sensors that are used in this thesis are introduced in this chapter, including their properties, measurements, and related algorithms for mobile robot estimate. For every type of sensor which will be installed on a robot, its measurement can be used to estimate the robot position; but each one has drawbacks.

A stereo camera cannot be applied for long range measurement, and it should be possible to obtain at least 7 matched features in every pair of consecutive images during navigation. This is a very strict requirement, which cannot be guaranteed in a real application.

GPS cannot work in the underwater environment. For a mobile robot which will be working in water, a buoy, which is a small boat equipped with GPS and acoustic sensors, will be used to measure the distance between the buoy and the robot. This range measurement can locate the robot's position. If GPS measurement can be used at the same time for estimating the buoy's position, then the absolute robot position can be estimated.

The contribution in this chapter is a new method which was developed to estimate registration uncertainty based on the implicit function theorem for correspondence-based approach from stereo images. Lab experiments have investigated the reliability and robustness of the algorithm.

Chapter 4

General Concept of 3D SLAM

Simultaneous localization and mapping is a process that fuses sensor observations of features or landmarks with dead-reckoning information over time to estimate the location of the robot in an unknown area and to build a map that includes feature locations. In this chapter, a general model and its related solving algorithm for 3D SLAM are established. The method can be used for all of the situations in the mobile robot community. An underwater mobile robot is used as an example.

This chapter is organized as follows: section 4.1 is the problem definition; Section 4.2 establishes all the models for $3D$ SLAM, including the robot process model, the landmark model, and the measurement model; section 4.3 is the method for data association; section 4.4 presents the algorithms to solve the SLAM; section 4.5 describes the multi-sensor related issues based on the underwater mobile robot cases; and section 4.6 is the globally-consistent 3D SLAM for mobile robot in real environment.

4.1 Problem Definition

Assuming a 3D environment with randomly distributed landmarks and an autonomous mobile robot equipped with sensors (stereo camera, laser range finder, or sonar) which will move in this environment, by providing some proper input (robot speed and orientation), we need to determine the robot pose (position and orientation) and the position of detected landmarks during the robot navigation.

Because of measurement noise and robot input noise, it is very difficult to compute a deterministic value for the robot pose and landmark position. We can only estimate their approximate value by using algorithms such as the Kalman filter, the Particle filter, and the Unscented Kalman filter. By using these algorithms, it is also possible to calculate the confidence of the estimation.

In some areas of the robot's working environment, significant landmarks are sparse, especially in the underwater environment. A robot equipped with only one type of sensor may not obtain sufficient effective measurements, which would greatly affect the accuracy of the robot pose; therefore, more than one sensor will be used for the robot navigation in a real application.

In this thesis, the SLAM problem in the 3D environment will be solved with multiple heterogenous sensors. A general strategy will be proposed and related algorithms will be developed.

4.2 Models for 3D SLAM

4.2.1 Robot Process Model

Robot process model is a dynamic differential equation to describe the movement of a robot in a given environment and system input. It is related to the robot pose. The robot pose can be determined by its position and orientation. In a global coordinate system $OXYZ$, a robot position (p_v) is expressed by $(x, y, z)^T$, and its orientation can be expressed by Euler angles, rotation matrix, axis and angle, or quaternions. From any one of the orientation representations, it is possible to compute the other representations (Appendix A). For simplicity, euler angles, $(\theta_z, \theta_y, \theta_z)$, are selected as a robot orientation state vector. Therefore, the state vector of the robot X_v can be expressed as

$$X_v = \begin{bmatrix} p_v^T \\ \theta_v^T \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \tag{4.2.1.1}$$

where T is the transpose of a matrix. Assuming that the robot moves relative to its current pose with speed v and changes direction with Euler angles $(\delta\theta_x, \delta\theta_y, \delta\theta_z)$, the input to the robot can be expressed by

$$U = \begin{bmatrix} v \\ \delta\theta_x \\ \delta\theta_y \\ \delta\theta_z \end{bmatrix} \tag{4.2.1.2}$$

where v is the robot speed (a scalar value), and the direction of the motion is always along the robot's forward pointing. In order to simplify its implementation, the Euler angles need to be expressed in the form of a rotation matrix M_v

$$M_v = R_z(\theta_z) \cdot R_y(\theta_y) \cdot R_x(\theta_x) \tag{4.2.1.3}$$

where R_z, R_y, and R_x are the rotation matrices which are the rotation around the z, y, x-axis, respectively, in right hand coordinate system with positive angle θ_x, θ_y, and θ_z. The positive angle is at counter-clockwise direction (Appendix A). Then, the robot process model can be expressed as

$$\theta_v(k+1) = \begin{bmatrix} \theta_x(k+1) \\ \theta_y(k+1) \\ \theta_z(k+1) \end{bmatrix} = \begin{bmatrix} f_1(\theta_x(k), \delta\theta_x, \delta\theta_y, \delta\theta_z) \\ f_2(\theta_y(k), \delta\theta_x, \delta\theta_y, \delta\theta_z) \\ f_3(\theta_z(k), \delta\theta_x, \delta\theta_y, \delta\theta_z) \end{bmatrix} \tag{4.2.1.4}$$

and

$$p_v(k+1) = \begin{bmatrix} x(k+1) \\ y(k+1) \\ z(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ z(k) \end{bmatrix} + M_v(k) \begin{bmatrix} cos(\alpha) \\ cos(\beta) \\ cos(\gamma) \end{bmatrix} v\delta t \tag{4.2.1.5}$$

where δt is the sampling time. $M_v(k)$ is the rotation matrix, which corresponds to the Euler angles $(\theta_x(k), \theta_y(k), \theta_z(k))$ at time k. In Eq.(4.2.1.4), the angle $\theta_v(k+1)$ corresponds to the matrix $M_v(k+1)$, given by following equation

$$M_v(k+1) = M_v(\delta\theta) \cdot M_v(k) \tag{4.2.1.6}$$

where $M_v(\delta\theta)$ is a matrix which corresponds to the Euler angle $\delta\theta$. And in Eq.(4.2.1.5), the α, β, γ are direction angles corresponding to the Euler angles $(\theta_x(k), \theta_y(k), \theta_z(k))$.

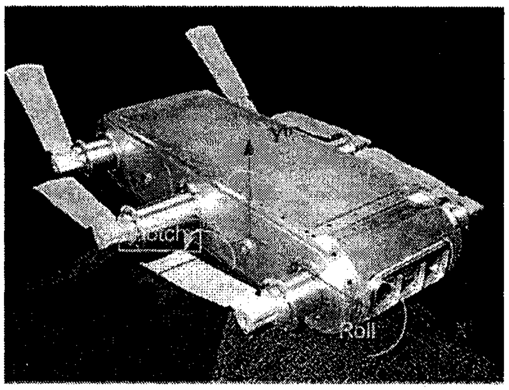


Figure 4.1: Coordinate systems of an autonomous mobile robot.

By combining Eq.(4.2.1.4) and (4.2.1.5), the process model can be written as a non-linear equation

$$X_v(k+1) = F(X_v(k), U(k) + \mu(k)) + \omega(k) \tag{4.2.1.7}$$

where $\mu(k)$ is the input noise, and $\omega(k)$ is the process noise, at the sample time k . The noise is assumed to be independent for different k, white, and with zero mean and covariance $Q_v(k)$.

4.2.2 Landmark Models

Landmarks can be classified into two types, artificial and natural. In a newly-visited natural environment, there is no artificial landmark for mobile robot navigation, therefore the natural landmarks are the only choice.

A robot map consists of a set of landmarks. In order to provide enough information for robot navigation, every landmark should include position information and attribute information (Fig. 4.2). If the landmark position is known, a sensor's measurement of it can be used in the robot pose estimation with algorithms such as the extended Kalman filter or the particle filter. If the landmark position is unknown, an algorithm for SLAM will be applied to estimate the robot pose and landmark position by the aid of measurements. The attribute information will provide knowledge about the landmark which distinguishes it from other features, which is very useful for data

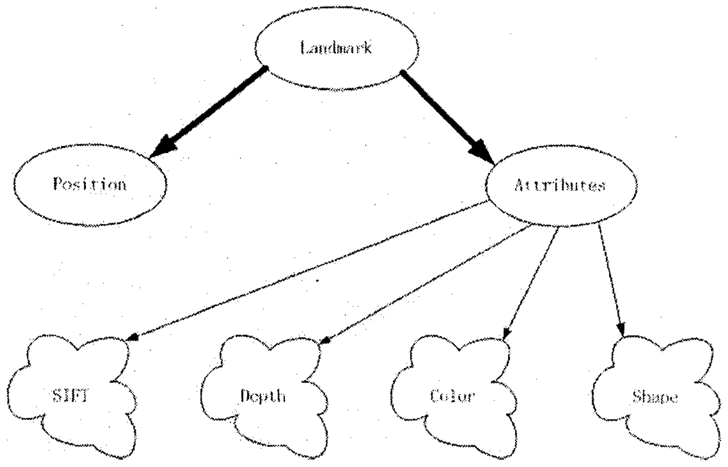


Figure 4.2: General landmark expression.

association; therefore, landmark L_i can be expressed as

$$L_i = \begin{bmatrix} L_{i,position} & L_{i,attribute} \end{bmatrix} \tag{4.2.2.1}$$

where

$$L_{i,position} = L_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{4.2.2.2}$$

During robot navigation, even though its environment is unknown, the landmarks for a map establishment are always assumed to have a static position. It is also assumed that attributes (or features) of a landmark will not change. In reality, features of a landmark may change with lighting conditions and sensor view point, therefore, the landmark i has the following evolution equation

$$L_i(k+1) = L_i(k) \tag{4.2.2.3}$$

where $i = 1, \cdots, m$, which means there are m landmarks which will be used; k is the time which is used during the robot navigation.

4.2.3 Measurement Model

A mobile robot is always equipped with some type of sensors for its navigation. The sensors can obtain measurements of the relative location of the observed landmarks with respect to the robot. This observation can be expressed by a set of non-linear functions of the landmark's position relative to the robot position, which is called measurement model. Assuming the position of landmark i in the global coordinate system $OXYZ$ is (x_i, y_i, z_i). At time k, the robot has the pose $X_v(k)$. The measurement of the landmark i at this time can be computed by

$$Z_i = \begin{bmatrix} Z_{x_i}(k) \\ Z_{y_i}(k) \\ Z_{z_i}(k) \end{bmatrix} = M_v(k) \begin{bmatrix} x_i - x(k) \\ y_i - y(k) \\ z_i - z(k) \end{bmatrix} \tag{4.2.3.1}$$

The observation model in the non-linear equation is

$$Z_i = h(X_v(k), x_i, y_i, z_i) + \eta(k) = h(X_v(k), L_i(k)) + \eta(k) \tag{4.2.3.2}$$

where $\eta(k)$ is the observation noise, which is assumed with zero mean and covariance R_i. $h(\cdot)$ is the non-linear measurement function.

Without loss of generality, it is assumed that the measurement from every sensor is independent. If there are m features observed at time k, the measurement model is obtained by simply stacking Eq. (4.2.3.2) as

$$Z(k) = \begin{bmatrix} Z_1(k) \\ Z_2(k) \\ \cdots \\ Z_m(k) \end{bmatrix} = H(X_v(k), L(k)) + \eta(k) \tag{4.2.3.3}$$

An instance of a robot and five landmarks in 3D space is shown in Fig. 4.3. When the robot moves in space, its sensor detects landmarks which are located in the view field of the sensor, then the robot pose and landmark position estimation can be performed. The estimated landmark position will be used to build a map which can be used by the robot for future navigation.

4.3 Data Association

There are two types of data associations - measurement between sensors received from multi-sensors, and measurement between adjacent times from a single sensor. In this thesis, only the data association between adjacent times from a single sensor will be

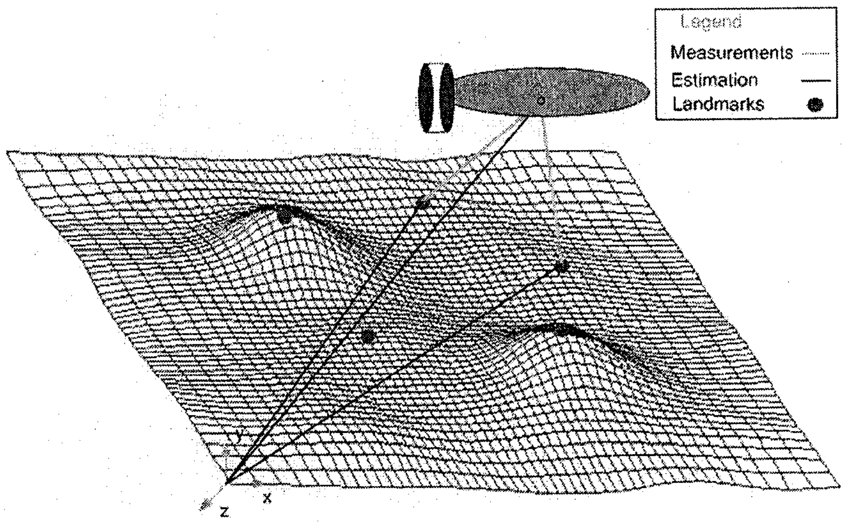


Figure 4.3: An instance of a robot and features in 3D experiment case.

addressed.

During robot navigation, if the sensor on the robot only observed one landmark, there would be no need for data association. Most sensors, such as camera, radar, laser, and sonar, will detect not only many real landmarks, but also many spurious landmarks; therefore, data association is a necessary step for landmark-based robot localization and object tracking.

A landmark defined in the previous section 4.2.2 includes position and attributes (feature) components. If the attribute tuple is available, then data association can be implemented with this information; otherwise, maximum likelihood of measurement will be used.

4.3.1 Data Association by Feature Attribute

Data association by using the landmark's attribute is simple. A very good example is the image feature registration with SIFT (Scale Invariant Feature Transforms) features [73]. SIFT features in an associated image are among the best representations for the natural unstructured environment. The SIFT features are invariant to image scaling, translation, and rotation, and partially invariant to illumination changes and

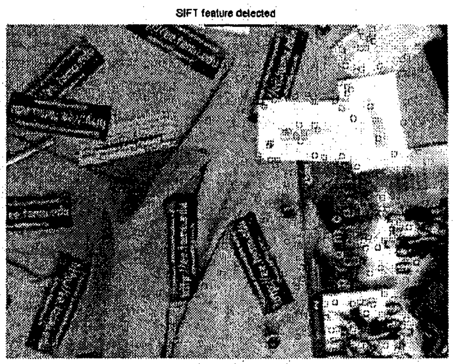


Figure 4.4: SIFT features in an image.

affine or 3D projection. The structure of the SIFT feature is as follows:

$$[u, v, gradient, orientation, descriptor_1, \cdots, descriptor_M]$$

where M is the number of the descriptor in SIFT feature. The position of landmark and the position of its related SIFT features (in an image) of a landmark have a non-linear relation. If a camera's physical position and orientation are given, the position of the landmark associated with the SIFT feature can be calculated by using its associated camera model [80].

An example of SIFT features in an image is shown in Fig. 4.4. The data association for SIFT features can be carried out by using their feature descriptor directly. SIFT features correspondence between two adjacent images obtained from a moving camera at different view points after the implementation of data association is displayed in Fig. 4.5.

4.3.2 Data Association by Maximum Likelihood of Measurement

If the measurements only provide a landmark's position information, but the landmark's attribute information is empty, the method presented by Bar-Shalom *et al.* [5] for the data association with innovation will be used here. Their method can be briefly described as follows:

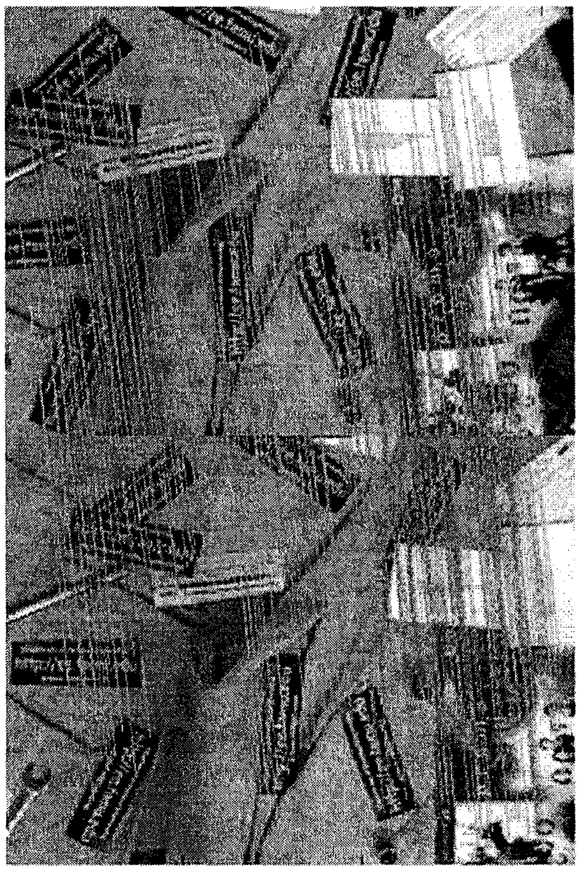


Figure 4.5: Result of SIFT feature correspondence between two adjacent images obtained from a moving camera at different view points after data association.

Innovation is the value of difference between measurement $Z(k)$ and predicted measurement $\hat{Z(k)}$, and expressed by $\nu(k)$

$$\nu(k) = Z(k) - \hat{Z(k)} \tag{4.3.2.1}$$

In order to define a measurement validation region, the innovation needs to be normalized as follows:

$$\varepsilon_{\nu}(k) = \nu(k)^T S_v(k)^{-1} \nu(k) \tag{4.3.2.2}$$

where $S_v(k)$ is the innovation covariance matrix, and it is defined as

$$S_v(k) = H(k)P_v(k)H(k)^T + R_v(k) \tag{4.3.2.3}$$

where $P_v(k)$ is the state vector's estimated covariance matrix at step k. The $\varepsilon_{\nu}(k)$ has a χ^2 distribution with n_z degrees of freedom, where n_z is the dimension of the measurement Z. The validation technique is based on this innovation. If a measurement is inside a fixed region of a χ^2 distribution, then this observation is accepted; otherwise the observation is rejected.

4.4 Estimation of Robot Pose and Landmark Positions

In the SLAM problem, a robot pose and landmark positions at time $k+1$ are unknown. They need to be estimated by using input information U, measurement information Z, and robot pose and feature position information, at time k. Stacking Eq. (4.2.1.4) and Eq. (4.2.1.5), the system process model can be expressed as

$$\begin{aligned} X_s(k+1) &= \begin{bmatrix} X_v(k+1) \\ L_1(k+1) \\ L_2(k+1) \\ \vdots \\ L_m(k+1) \end{bmatrix} = \begin{bmatrix} X_v(k+1) \\ L(k+1) \end{bmatrix} \\ &= \begin{bmatrix} F(X_v(k), U(k), \mu(k)) + \omega(k) \\ L(k) \end{bmatrix} \end{aligned} \tag{4.4.0.4}$$

and the measurement model is

$$Z(k) = H(X_v(k), L(k)) + \eta(k) \tag{4.4.0.5}$$

Both process model (Eq.(4.4.0.4)) and measurement model (Eq.(4.4.0.5)) are non-linear equations. A straightforward method to solve this problem is the Extended Kalman Filter (EKF). Due to the high dimensions of the state vector and the need for linearization of the non-linear models, the EKF is not computationally attractive. The Particle filter-based fast SLAM approach will be applied to solve this problem.

4.4.1 Particle Filter

From the view point of probability, the estimation of a robot pose and landmark positions involves computing their *posterior* probability density function (PDF), $p(X_v(k), L(k)|Z(k), X_v(k-1), U(k-1))$, based on the *prior* probability density function $p(Z(k)|X_v(k))$ and $p(X_v(k-1)|Z(k-1), X_v(k-2), U(k-2))$, where $X_v(k)$ is the robot state, $L(k)$ is the landmark state, $Z(k)$ is the measurement, $U(k)$ is the input to the system, at time k. This is the well-known Bayesian approach.

According to the definition of a robot model and landmark model in the previous section 4.2, the PDF for a SLAM problem $Bel(X_v(k+1))$ at time $k+1$ can be defined as [83]

$$Bel(X_v(k+1)) = p(X_v(k+1)|Z(k+1), X_v(k), U(k)) \tag{4.4.1.1}$$

The solution for the SLAM problem is to estimate the maximum of the $Bel(X_v(k+1))$. Bayes' formula can be used on Eq. (4.4.1.1) to simplify its implementation.

$$\begin{aligned}Bel(X_v(k+1)) &= \frac{p(Z(k+1)|X_v(k+1), X_v(k), U(k))p(X_v(k+1)|X_v(k), U(k))}{p(Z(k+1)|X_v(k), U(k))} \\ &= \xi p(Z(k+1)|X_v(k+1), X_v(k), U(k))p(X_v(k+1)|X_v(k), U(k))\end{aligned} \tag{4.4.1.2}$$

Where ξ is the value of the inverse denominator, and is assumed to be a constant. It is known that the measurement $Z(k+1)$ is only dependent on the current pose $X_v(k+1)$ and is not influenced by previous pose $X_v(k)$ and robot movement $U(k)$. Therefore, Eq. (4.4.1.2) can be simplified into

$$Bel(X_v(k+1)) = \xi p(Z(k+1)|X_v(k+1))p(X_v(k+1)|X_v(k), U(k)) \tag{4.4.1.3}$$

By applying the total probability theorem to the second item of the right part in

Eq. (4.4.1.3), then

$$\begin{aligned} Bel(X_v(k+1)) &= \xi p(Z(k+1)|X_v(k+1)) \\ &\int p(X_v(k+1)|X_v(k), U(k))p(X_v(k)|Z(k), X_v(k-1), U(k-1))dX_v(k) \\ &= \xi p(Z(k+1)|X_v(k+1)) \\ &\int p(X_v(k+1)|X_v(k), U(k))Bel(X_v(k))dX_v(k) \end{aligned} \tag{4.4.1.4}$$

where $p(Z(k+1)|X_v(k+1))$ is the sensor observation model, which can be calculated from Eq. (4.4.0.5); $p(X_v(k+1)|X_v(k), U(k))$ is the system evolution model, which can be calculated from Eq. (4.4.0.4). The integration in the Eq. (4.4.1.4) is a difficult challenge to solve the SLAM problem efficiently; therefore, a new algorithm must be designed.

The Monte Carlo based particle filter can be used to overcome the implementation challenge in Eq. (4.4.1.4). In the particle filter, $Bel(X_v(k))$ is expressed as a set of particles, and every particle is propagated in time according to the state process model such as Eq. (4.4.0.4). The weight of every particle is calculated based on the observation model from the Eq. (4.4.0.5). The robot pose and landmark position can be computed from the sum of the weighted samples. The particles should be re-sampled for the next step's estimation. Implementation of a particle filter is summarized in Alg. 2.

Particle filtering can be used for any process and observation models. In the Kalman filter and the extended Kalman filter, the basic requirement is that the error of process model and observation model should be Gaussian distribution. In most cases, this requirement is too restrictive. The particle filter has been called bootstrap filter [34], condensation [41], or Monte Carlo filter [18]. In recent years, this method has been successfully used in problems of object tracking [39] and mobile robot localization [18, 84].

Input: Robot movement $U(k)$, sensor measurement $Z(k+1)$, and sample number N
Output: Robot pose and features position

```
1: initialize state with $p(X_v(0))$
2: repeat
3:   for every particle $i$ do
4:     assign distribution using $p(X_v(k+1)|X_v(k), U(k))$
5:   end for
6:   for every particle $i$ do
7:     compute weight $w_i$ using $p(Z(k+1)|X_v(k+1))$
8:   end for
9:   calculate robot pose & landmark position from particles & associated weight
10:  re-sample the particles
11: until robot stop navigation
```

Algorithm 2: Particle filter implementation for robot pose and feature position

4.4.2 Fast SLAM

Fast SLAM is an approach to separate the SLAM problem into a robot pose and landmark position estimation that is conditioned on the robot pose. The term was first introduced in [60]. The implementation of FastSLAM is an example of the Rao-Blackwellised particle filter [64, 21].

From the previous definition of a SLAM problem, the system state estimate could be written as

$$Bel(X_v(k+1)) = p(X_v(k+1), L(k+1)|Z(k+1), X_v(k), U(k)) \tag{4.4.2.1}$$

This expression can be factored into two parts according to [60].

$$\begin{aligned} Bel(X_v(k+1)) &= p(X_v(k+1), L(k+1)|Z(k+1), X_v(k), U(k)) \\ &= p(X_v(k+1)|Z(k+1), X_v(k), U(k)) \\ &\prod_{i=0}^{m} p(L_i(k+1)|X_v(k+1), Z(k+1), U(k)) \end{aligned} \tag{4.4.2.2}$$

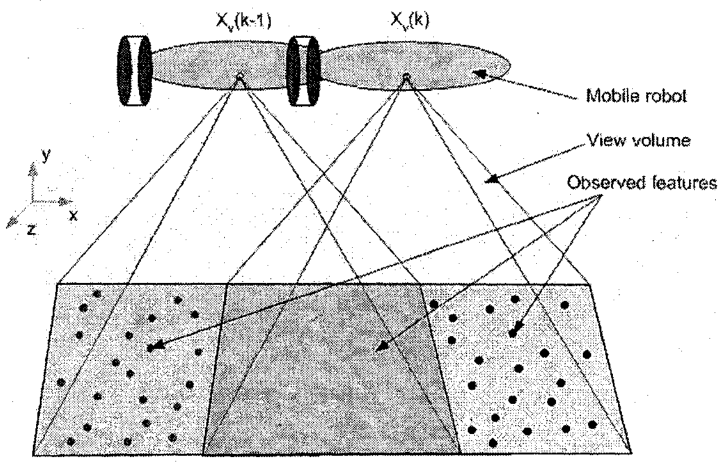


Figure 4.6: SLAM implementation with features observed by a sensor.

The estimate expression is decomposed into $m+1$ estimations. One of them is for the robot pose estimation, and m of them are for landmark estimation based on the estimated robot pose. The implementation of FastSLAM is summarized in Alg. 3. In this thesis, this algorithm is applied for the SLAM problem. The particle filter is implemented to calculate the conditional densities for robot and landmarks.

The idea for FastSLAM can be obtained from Fig. 4.6. We assume that all the observed landmarks by a sensor at robot position $X_v(k-1)$ exist in a map. Then, the observed landmarks at robot position $X_v(k)$ can be divided into two groups. Some of them are already in the map and are labeled as small yellow squares in Fig. 4.6, which are called old landmarks; some of them are new landmarks and do not yet exist in the map and are expressed with small black dots in Fig. 4.6. The measurements from the old landmarks are applied for the robot pose estimation at time k. The measurements from the new landmarks are used to estimate the new landmark positions in the global coordinate system based on the robot position. Then, all the new landmarks are added into the map.

In the fast SLAM approach, the factorizing assumption step turned the high dimension $6+3\cdot m$ of the SLAM problem into the low dimension (6 and 3) problem's combination, which greatly improves the computation efficiency, but it is assumed that the estimated robot pose has an accurate value. This assumption is not true and

Input: Robot movement $U(k)$, sensor measurement $Z(k+1)$, and sample number N

Output: Robot pose and detected features position

```
1: initialize state with $p(X_v(0)$
2: repeat
3:    for every particle $i$ do
4:       proposal distribution using $p(X_v(k+1)|X_v(k), U(k))$
5:    end for
6:    obtain observations $Z(k+1)$
7:    data association for the observation data
8:    for every particle $i$ do
9:       compute weight using $p(Z(k+1)|X_v(k+1))$
10:   end for
11:   re-sample the particles
12:   if current observed feature exists in the map then
13:      for every particle $i$ do
14:         for every observed feature do
15:            update the state of the robot
16:         end for
17:      end for
18:   end if
19:   if current observed feature is not in the map (new detected features) then
20:      for every particle $i$ do
21:         add the new detected features to the map based on the robot pose and
            observation to the features
22:      end for
23:   end if
24: until robot stop navigation
```

Algorithm 3: FastSLAM implementation

will cause errors in the step for the landmark position estimation in a global frame. This is the trade-off between efficiency and accuracy.

Another assumption in the fast SLAM is that the measurement of every detected landmark is independent of the other landmarks in the working area of the robot; therefore, the covariance between two landmarks will be zero. In other methods, such as the EKF or Particle filter, robot pose and all the detected features position are estimated in one state vector, and this may cause the covariance between two landmarks to be other than zero. In most of the cases, these values came from the algorithm design.

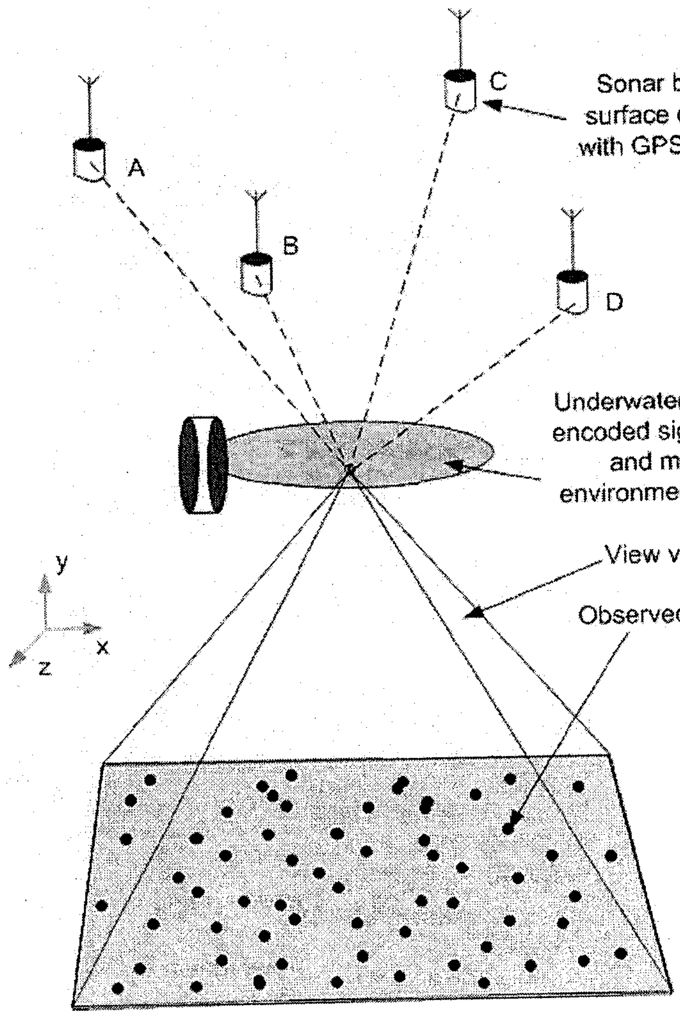
4.5 Multi-sensor Fusion for 3D SLAM

A mobile robot is usually equipped with many sensors which will work together. Fusing the measurements from more than one sensor will provide a more accurate estimation than by using only one sensor's measurement. An example of multi-sensor fusion is shown in Fig. 4.7, where an underwater mobile robot is equipped with a stereo camera and communicated with a set of buoys on the surface.

This kind of system structure has two advantages for the underwater robot. The buoys can provide a long range of measurements, while the measurements can be applied to estimate the robot position in a global frame. The stereo camera can provide detailed information of the immediate environment, which will be used for SLAM in a local frame. Integration of both can solve the SLAM problem in a large area for the underwater robot. The buoy system has been proposed using acoustic sensor by Liu and Milios [50].

4.5.1 Synchronization for Multi-sensor Fusion

All the sensors in a system cannot work at the same speed or frequency, such as in Fig. 4.7. Synchronization for multi-sensor fusion is an important issue. Usually, measurement frequency for each sensor is different, and their measurement time will not coincide. An instance of the estimated robot position from stereo camera and



C
Sonar buoy on surface of water, with GPS receiver
A
B
D
Underwater robot received encoded signal from buoys and measure the environment with a setero
View volume
Observed features
y
x
z

Figure 4.7: A sensor fusion example for a stereo camera and a set of buoys.

buoys with time stamp is shown in Fig. 4.8. The buoys will provide a long range estimate in a global frame, which has less accumulated errors than that with a camera. Therefore, when the estimate from buoys at time $t_{b,k}$ is obtained, the robot position from buoys at t_k should be estimated by interpolation as

$$X_b(t_k) = \xi_1 X_b(t_{b,k-2}) + \xi_2 X_b(t_{b,k-1}) + \xi_3 X_b(t_{b,k}) \tag{4.5.1.1}$$

where b means buoy and k means time stamp; $X_b(t_{b,k-2})$, $X_b(t_{b,k-1})$ and $X_b(t_{b,k})$ are robot position estimated by buoys at time $t_{b,k-2}$, $t_{b,k-1}$ and $t_{b,k}$, respectively. And ξ_1, ξ_2 and ξ_3 are coefficients related to the corresponding time stamps.

$$\xi_1 = \frac{(t_k - t_{b,k-1})(t_k - t_{b,k})}{(t_{b,k-2} - t_{b,k-1})(t_{b,k-2} - t_{b,k})} \tag{4.5.1.2}$$

$$\xi_2 = \frac{(t_k - t_{b,k-2})(t_k - t_{b,k})}{(t_{b,k-1} - t_{b,k-2})(t_{b,k-1} - t_{b,k})} \tag{4.5.1.3}$$

$$\xi_3 = \frac{(t_k - t_{b,k-2})(t_k - t_{b,k-1})}{(t_{b,k} - t_{b,k-2})(t_{b,k} - t_{b,k-1})} \tag{4.5.1.4}$$

The uncertainty $P_b(t_k)$ of the position $X_b(t_k)$ from buoys can be obtained from Eq. (4.5.1.1) based on the fact that all the estimated robot positions are independent variables since the robot position is estimated by using the measurement between the current robot position and buoys, instead of previous robot position, as discussed in Chapter 3. This is

$$P_b(t_k) = \xi_1^2 P_b(t_{b,k-2}) + \xi_2^2 P_b(t_{b,k-1}) + \xi_3^2 P_b(t_{b,k}) \tag{4.5.1.5}$$

where $P_b(t_{b,k-2})$, $P_b(t_{b,k-1})$ and $P_b(t_{b,k})$ are uncertainties of robot position estimated by buoys at time $t_{b,k-2}$, $t_{b,k-1}$ and $t_{b,k}$, respectively.

The estimated robot position from buoys at time t_k (see Fig. 4.8) can be obtained by using Eq. (4.5.1.1) and Eq. (4.5.1.5). At this time, fusing the robot estimation from the stereo camera and buoys is possible.

It is very important to ensure that all the sensors installed in the same system are synchronized. If there are more than two computers working for the same measurement system, it is necessary to calibrate them.

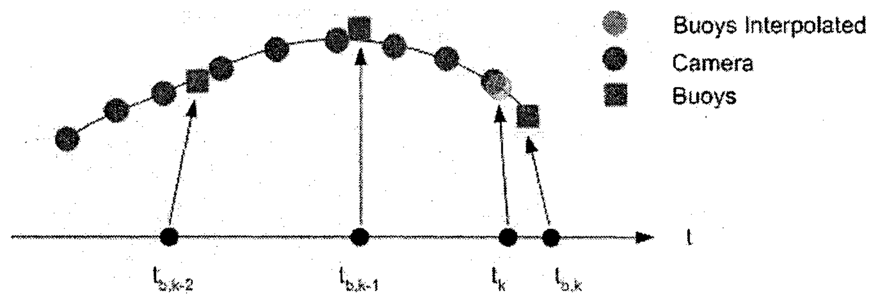


Figure 4.8: A stereo camera and buoys estimation with time stamps.

In the previous derivation, It must be pointed out that three points are used for the interpolation in this section. If there are more than three robot positions, a subset of three can be used, or higher order interpolation can be used. If only two robot positions (points) are available, linear interpolation could be applied.

4.5.2 General Sensor Fusion Mechanism

All the sensors in the system are applied for one purpose: reliable and accurate robot pose and map estimation. The sensors will not work properly all the time; therefore, a complementary fusion mechanism is designed in this thesis. Fig. 4.9 is a sensor fusion architecture for a stereo camera and a set of buoys.

Before sensor fusion is applied, each sensor in the system estimates the robot pose and/or builds a map independently. When the synchronized estimate is obtained, sensor fusion will be performed. Assuming that the error from each sensor follows the Gaussian distribution with zero mean and known covariance, from [5], the fusion is carried out by

$$X_v(t_k) = \frac{P_b(t_k)}{P_b(t_k)+P_c(t_k)} X_c(t_k) + \frac{P_c(t_k)}{P_b(t_k)+P_c(t_k)} X_b(t_k) \tag{4.5.2.1}$$

$$P_v(t_k) = \frac{P_b(t_k)P_c(t_k)}{P_b(t_k)+P_c(t_k)} \tag{4.5.2.2}$$

where $X_c(t_k)$ and $X_b(t_k)$ are the estimated robot pose at time t_k by the stereo camera and buoys, and $P_b(t_k)$ and $P_c(t_k)$ are their associated covariance, respectively.

It must be pointed out that the relative drift of the sensor estimate has a significant influence on the the sensor fusion result. Ideally, both of the sensor estimates

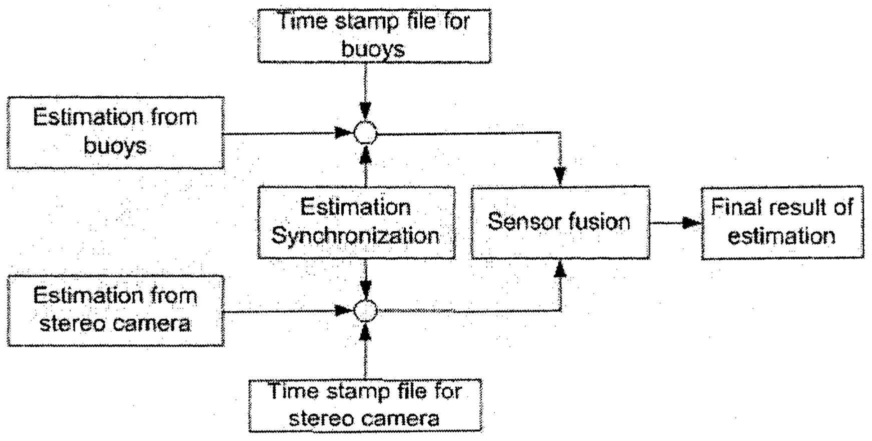

Time stamp file for buoys
Estimation from buoys
Estimation Synchronization
Sensor fusion
Final result of estimation
Estimation from stereo camera
Time stamp file for stereo camera

Figure 4.9: Sensor fusion architecture for camera and buoys.

should overlap in their estimation uncertainty's boundary. If the sensor estimates do not overlap in their uncertainty boundary, the fusion result will not be reliable. To avoid large relative drift of the sensor fusion, we used multi-map mechanism, which means that a new estimation sequence will start as soon as the estimation uncertainty reaches a threshold.

By using this sensor fusion mechanism and system designed for an underwater mobile robot, it is possible to estimate a globally-consistent robot position over a large work area. The following section discusses globally-consistent map building.

4.6 Globally-Consistent 3D SLAM for Mobile Robot in Application

The basic requirement for SLAM using stereo vision is that every two consecutive images have to provide enough overlapping features. In real applications, this requirement may be too strict. If this requirement can not be satisfied, the SLAM process based on image information will be stopped. In this case, even though the robot has another sensor installed (such as a GPS or buoys), it can not obtain a globally-consistent 3D path and a globally-consistent 3D map; therefore, it is necessary to design a new algorithm to solve the 3D SLAM problem in this more general case.

We assume that two sensors (stereo camera and buoys or GPS) will be employed, operating independently, the buoys estimate a globally-consistent robot 3D path and the stereo camera estimates a set of local maps and the robot's 3D paths in a local coordinate system. This means that when the two consecutive images provide enough overlapping features, the SLAM algorithm begins to work based on the local frames. The estimates also receive a global time stamp. The scenario of this processing is shown in Fig. 4.10. There are four local robot path and associated maps (not shown) based on local frame which is estimated by the stereo camera's measurement with the algorithm of SLAM in Fig. 4.10 (a). For simplicity, the maps are not drawn. These estimated path fragments are time stamped, which will be applied for sensor fusion. For the estimate by buoys measurement in Fig. 4.10 (b), the red dots are the robot's position at each time, and the blue squares are the time markers which are corresponding to the time stamps in Fig. 4.10 (a). The final globally-consistent robot path and map are expressed in Fig. 4.10 (c). There are several different steps required to obtain these: (1) robot path parameterizing; (2) robot position association; (3) transformation estimate from local coordinate system to global coordinate system; (4) globally-consistent map integration, and; (5) globally-consistent robot position.

4.6.1 Robot Path Parameterization

For 3D SLAM, connecting the robot position over time will form a curve in 3D space. A widely-used method to construct a 3D curve to smoothly pass all the discrete position points is the B-spline interpolation [68]. Assume that a sequence of robot 3D positions, $p_v(k)$, and its related covariance $P_{vv}(k)$ $(k = 0, \cdots, n)$, are obtained from a local coordinate system, and its corresponding time is t_k (Fig. 4.11). A parametric B-spline of degree 3 to pass these points is defined as

$$S(u) = \sum_{i=0}^{n} p_v(i) N_{i,3}(u) \quad u \in [0,1] \tag{4.6.1.1}$$

where $N_{i,3}(u)$ are the B-spline basis functions of degree 3, defined with respect to the knot vectors $u = \{u_0, u_1, \ \dots \ , u_{n+3+1}\}$, with $u_0 = u_1 = u_2 = u_3 = 0$ and $u_{n+1} = u_{n+2} = u_{n+3} = u_{n+3+1} = 1$, and its parameter u at any time can be obtained by

$$u = \frac{t - t_0}{t_n - t_0} \quad t \in [t_0, t_n] \tag{4.6.1.2}$$

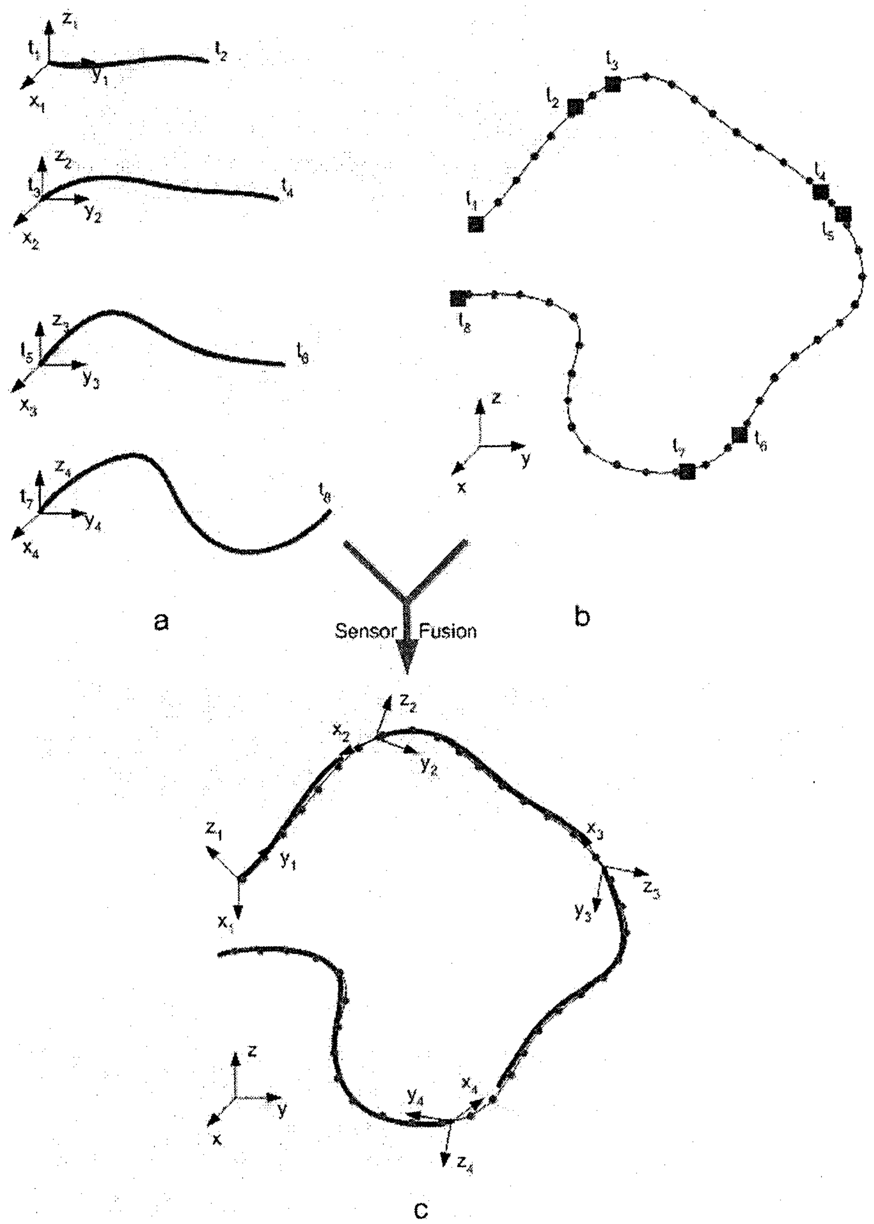


Figure 4.10: 3D SLAM by fusion estimation from camera and buoys. (a) Local robot position and map (for simplicity, map is not displayed in this figure) from camera with time stamps. (b) Robot position estimated from buoys with time stamps. (c) Final results after fusing the information from both camera and buoys.

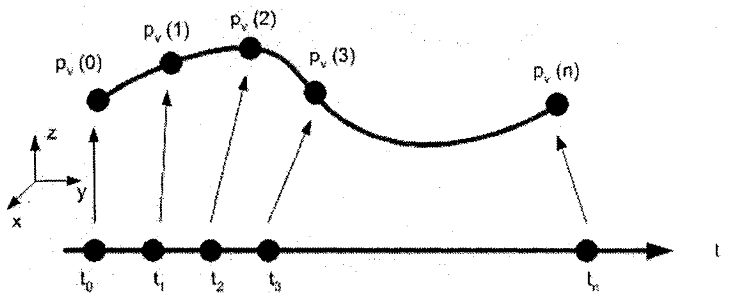


Figure 4.11: Robot path parameterizing with B-spline.

A B-spline base function $N_{i,p}(u)$ of degree p can be calculated by

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{any others} \end{cases} \tag{4.6.1.3}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u} N_{i+1,p-1}(u) \tag{4.6.1.4}$$

$$\text{define } \frac{0}{0} = 0$$

By using the B-spline function in Eq. (4.6.1.1), for any time t between time t_0 an t_n, it is easy to obtain the position of the robot. Its related covariance can be obtained by

$$P_{vv}(u) = \sum_{i=0}^{n} p_v(i)(N_{i,3}(u))^2 \quad u \in [0,1] \tag{4.6.1.5}$$

4.6.2 Robot Position Association

From the buoys, the robot position is estimated during its navigation. The positions, $X_b(k)$ $(k = 0, \cdots, m)$, are denoted by small black squares on the red curve in Fig. 4.12, and their corresponding times are shown on the time axis with tb_k, $(k = 0, \cdots, m)$. The robot path estimated from the camera is expressed by the blue curve, which is constructed in the previous subsection. With the time information, the corresponding robot position, $X'v(k)$, based on the local coordinate system$(x'y'z')$, can be calculated from the Eq. (4.6.1.1) and Eq. (4.6.1.5).

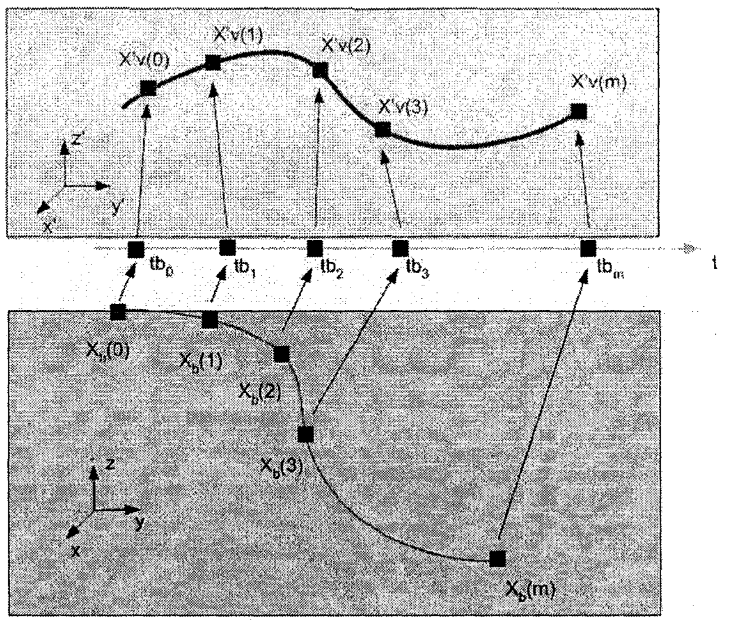


Figure 4.12: Robot position association. The top is in local coordinate system, and the bottom is in global coordinate system.

4.6.3 Transformation Between Frames

So far, two sets of corresponding 3D points, $X_b(k)$ and $X'_v(k)$ $(k = 0, \cdots, m)$, are obtained (We only use $X'_v(k)$ position information here since $X_b(k)$ only contain position information). If the number of corresponding points, m, is greater than 3 and not all the points in each coordinate system are in a straight line, it is then possible to estimate the transformation (rotation and translation) between these two data sets according to the method described in Chapter 3. If all the estimated 3D points in one of its local coordinate systems are located on a line or are very close to a line, it is impossible to estimate the orientation of this transformation. In this situation, other information such as a robot direction from a digital compass will be needed.

4.6.4 Globally-Consistent Map Integration

Assuming that the transformation for a local map k $(k = 1, \ldots, n)$ from a local coordinate system to a global coordinate system is expressed by R_k and T_k , for rotation and translation, respectively, a map established in the local coordinate system can be then transformed to the global coordinate system by

$$L_k^g(i) = R_k L_k^l(i) + T_k \tag{4.6.4.1}$$

where $L_k^g(i)$ is the landmark i's position in the global coordinate system, and $L_k^l(i)$ is the landmark i's position in the local coordinate system($i = 1, \cdots, M$).

If there are several different local maps established on one working area, it will be necessary to transform all of them from each local coordinate system to a global coordinate system, and then integrate all of them to form a globally-consistent map.

4.6.5 Globally-Consistent Robot Position

Due to the bandwidth limitation of the communication or the nature of the sensing, the measurement frequency of the buoys is lower than the stereo camera's frequency. In order to obtain a continuously consistent 3D path, the robot path in a local coordinate system should be transformed to the global coordinate system according to the same method from the previous map transformation.

The algorithm to establish a globally-consistent 3D map is based on the sensor fusion shown in Alg. 4. An overview of this algorithm is shown in Fig. 4.13. The centre of this diagram belongs to Alg. 4, and the left part for local SLAM is obtained by a camera with the previous FastSLAM, and the global robot position is obtained with the EKF algorithms by buoys and GPS.

In the processing of local SLAM from a camera, there are two issues which need to be considered: estimation error and efficient map. In order to control the estimation error in the local SLAM result, we always check the estimation error for robot path. If the estimated path error grows above a certain threshold, the current local SLAM processing is also stopped, in which case a new local SLAM processing will be started. For efficient map, we will discuss it in the next chapter.

Input: a sequence of images I_k, $k = 1, \cdots, N$, measurements from buoys, and a file of time stamps for each sensor

Output: a globally consistent map G and robot path p_v

```
1: k=2, numMap=0, flag=$newMap\_start$, Segment=[]
2: repeat
3:    robot path estimated from the measurements of buoys
4:    extract SIFT features from image $I_k$, $I_{k-1}$
5:    if matched features between image $I_{k-1}$ and $I_k$ are more than $m_f$ or the estimated position error is less than a certain threshold then
6:       if flag=$newMap\_start$ then
7:          numMap=numMap+1;
8:       end if
9:       SLAM estimation based on stereo camera's measurements
10:      Segment(k)=numMap
11:      flag=$map\_continue$
12:   else
13:      flag=$newMap\_start$
14:   end if
15: until robot stop navigation
16: robot path 3D curve from buoys estimation by B-spline
17: for each Segment do
18:    find corresponding point on 3D path curve
19:    estimate transformation parameters between local frame and global frame
20:    global map and path calculation
21:    integrate 3D map and path
22: end for
```

Algorithm 4: Globally Consistent 3D SLAM based on Sensor Fusion

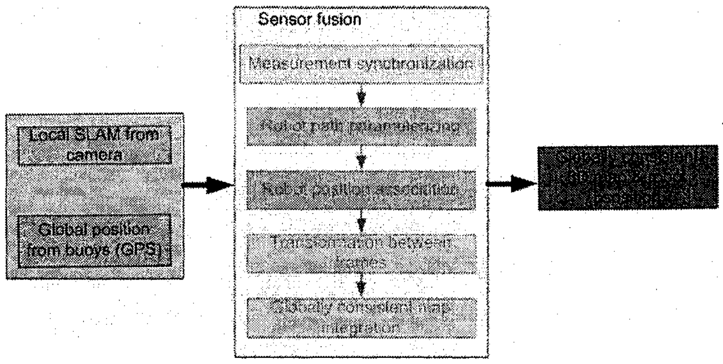

Sensor fusion
Measurement synchronization
Robot path parameterizing
Robot position association
Transformation between frames
Globally consistent map integration
Local SLAM from camera
Global position from buoys (GPS)

Figure 4.13: Diagram for the globally-consistent 3D SLAM.

4.7 Summary

This chapter established an approach to solve the full $3D$ SLAM problem, applied to an underwater environment. First, a general approach to the 3D SLAM problem was presented, which included the models in 3D case, data association and estimation algorithm. For an underwater mobile robot, a new measurement system was designed for large area's globally-consistent SLAM: buoys for long-range estimation, and camera for short-range estimation and map building. Globally-consistent results could be obtained by a complementary sensor fusion mechanism.

By carefully investigating all the algorithms used for SLAM, we designed a new sensor fusion algorithm for large area SLAM. Two types of sensors are needed for this algorithm: stereo camera (or laser scanner, radar) for local SLAM, and a sensor for robot path in the global coordinate system. Both the local and global paths were expressed by B-spline, and their corresponding 3D points of the associate robot path could be extracted. Transformation (translation and rotation) values from local coordinate system to global coordinate system could be estimated from these matched points, and, the local maps could be integrated to the globally-coordinate system, and a globally-consistent map and path could be obtained.

Chapter 5

Feature Selection for Map Building

It is well known that there are three basic maps in the mobile robot community - the stochastic landmark map [79], the occupancy grid map [12, 18, 62], and the topological map [23]. For a large area, a hierarchical map [25] is applied, which is a combination of the previous basic maps. In this thesis, the stochastic landmark map is used to build an efficient map for a mobile robot.

The organization of this chapter is as follows: the relation between landmarks and features is discussed in section 5.1; the problems of map building are defined in section 5.2; the requirements for selecting a subset of features to form a map are presented in section 5.3; a mechanism to construct an efficient and effective map is proposed in section 5.4; and the summary of the chapter is presented.

5.1 Landmark and Feature

Feature and *landmark* are two terms which are used frequently in the robot community. For better clarity, the definitions for them are listed below.

Definition 5.1.0.1 *A landmark is a a known visible solid physical object.*

Definition 5.1.0.2 *A feature is a specific structure in the image, for example an edge, corner or blob.*

Generally, a landmark existed in a 3D space has position (x, y, z), colour, size, shape, etc as its attributes. For a vision-based robot localization, the relation between *landmark* and *feature* and their detail components are shown in Fig. 5.1. A landmark contains position attributes in 3D and many image-based features.

A SIFT feature is a special feature of a landmark which is extracted from its associated image. For vision-based robot navigation, we extract SIFT feature from an

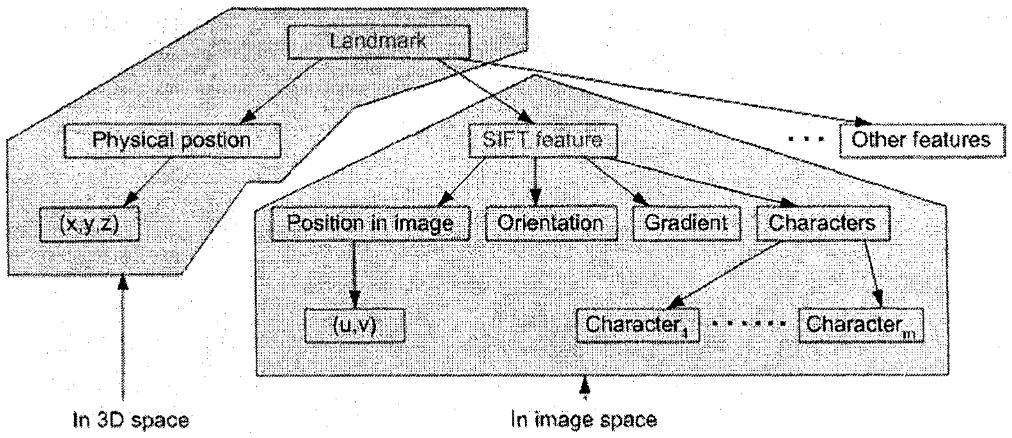


Figure 5.1: Relation between landmark and feature for vision-based robot localization.

image, and then determine the 3D position of the SIFT feature by the camera model. This determines the landmark for the robot's navigation.

For the map building in this chapter, *feature selection* means to select a subset of SIFT features from a set of SIFT features. Landmark selection and feature selection have the same functionality for map building (to build a concise map), but the difference is that they came from different spaces. Landmark selection happens in 3D physical space, and feature selection happens in image space (Fig. 5.1). Even though there are many features for a landmark, the term *feature* in the following sections means *SIFT feature.*

5.2 Problems of Map Building

For landmark based maps, one of the problems is how to select the appropriate landmarks from the observed landmarks to construct a map. For example, in a visual-based robot navigation system, SIFT features are widely used. From each frame of image, there are more than 1000 features for an image with 604 $\times$ 480 pixels in a rich featured area. For a small image with 240 $\times$ 320 pixels, there are more then 100 features in a sparse featured area.

Following is an example from our lab experiment in Chapter 3. An robot arm navigates by following a circular path in space, and a stereo camera system installed

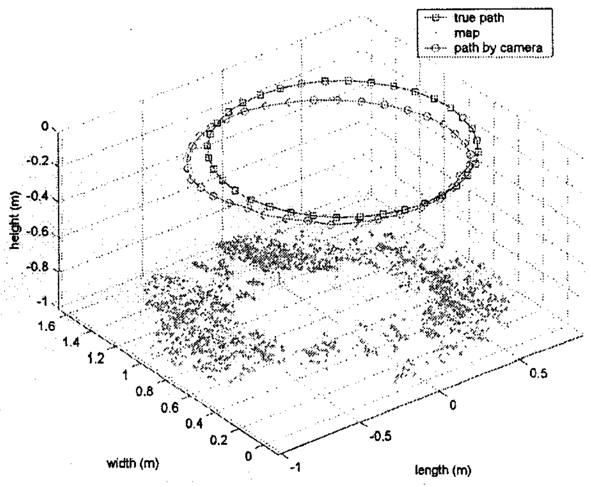


Figure 5.2: Robot path and map with SLAM algorithm in 3D.

on the robot takes a sequence of images by viewing the ground surface of the working area $3m \times 3m \times 1m$ in length, width, and height. All the landmarks observed two times in the robot working area are displayed in Fig. 7.5. In this case, the established map has 2025 features in the image sequence with 36 images, and every frame of image has about 56 features on average. The problem here is, do we really need so many landmarks to construct a map for the robot navigation?

5.3 Requirement for Map Establishment

A map for a mobile robot consists of a set of landmarks that have distinguishable feature attributes from other landmarks. The features of these landmarks should be widely visible (invariant to position, orientation, and illumination). We denote a map G with m landmarks $L_i, (i = 1, \ldots, m)$ as:

$$G = (L_1, \cdots, L_m) \tag{5.3.0.1}$$

The basic requirements for a map are defined as

(1) *Having a relatively small number of landmarks*;

(2) *Landmark's feature should be distinct*;

(3) *At any position on a path, at least m_f landmarks are visible.*

To meet the first requirement, it is necessary to keep a map as concise as possible. There are many advantages to having a concise map - first, it will save computation time; in a SLAM process, as soon as a landmark is observed, the map needs to be updated based on the new observation, and during the data association step, the smaller the map is, the faster the data association; second, it will save storage space. Each image frame from every time step, generally will contain more features than the algorithm requires. Only some of them will be needed to estimate the pose of a robot. Different algorithms require different minimum number of matched features for the robot's ego-motion estimation. Following are some examples:

With two consecutive images F_{k-1} and F_k from a moving camera which has calibration matrix K, if a set of corresponding points u_i, u'_i is determined, the relative translation T_k and rotation R_k from F_{k-1} to F_k can be computed by using the epipolar geometry in Fig. 5.3 from the following equation

$$u^T(K^{-1})^T S(T_k) R_k^{-1} K^{-1} u' = 0 \tag{5.3.0.2}$$

where matrix $S(T_k)$ is a skew symmetric matrix of the translation vector $T_k = [t_x^k, t_y^k, t_z^k]^T$, which is expressed as

$$S(T_k) = \begin{pmatrix} 0 & -t_z^k & t_y^k \\ t_z^k & 0 & -t_x^k \\ -t_y^k & t_x^k & 0 \end{pmatrix} \tag{5.3.0.3}$$

The solution of R_k and T_k in Eq. (5.3.0.2) needs at least 8 corresponding image points [80]. If there are more points, the estimation will be robust to noise and mismatched.

From a stereo camera system or a laser scan camera, it is possible to acquire a set of corresponding points in $3D$ space. These points are the landmarks in a real working area. The translation T_k and rotation R_k can be estimated by the registration method as described in chapter 4. In this method, at least 4 3D points are needed for a solution.

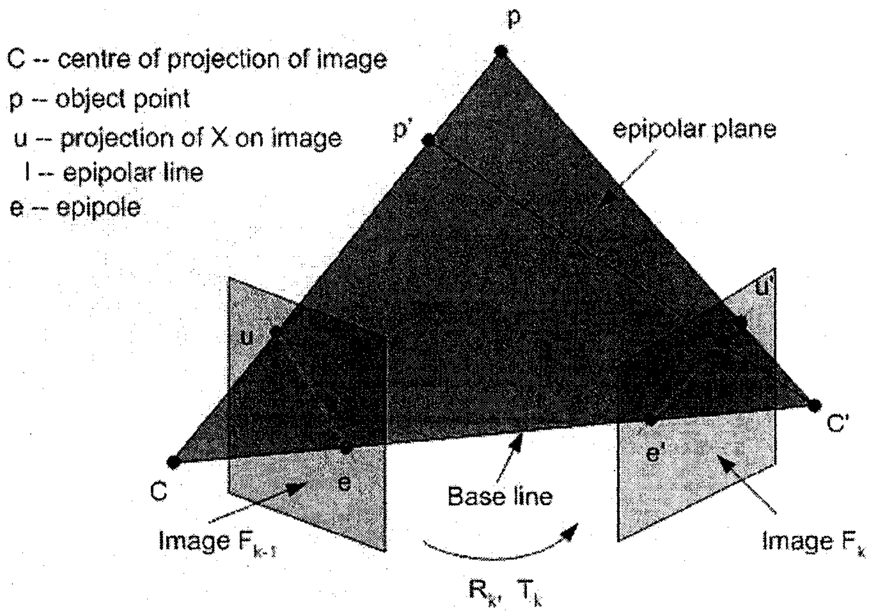


Figure 5.3: Epipolar geometry for robot's ego motion estimation.

Based on different algorithms employed, with at least m_f corresponding features or landmarks between two frames, it is possible to determine the robot's pose relative to its previous position. Of course, the more points used, the more robust the results. But, how many points will be enough for reliable results? A Monte Carlo simulation of registering two sets of $3D$ points is implemented with different numbers of features. The estimated distance error and angle error are shown in Fig. 5.4. From the simulation results, if there are more than 10 corresponding $3D$ points, the estimation accuracy will not change much with additional matched landmarks.

The simulation makes the assumption that the data association is perfect, i.e. there are no mismatches. In reality, because of noise, mismatching occurs, even though robust methods have been applied. In order to increase reliability of the estimation, more features should be selected at each frame. If the minimum feature number for ego-motion is m_f, the number of selected feature in 5 to 10 times of m_f would be a safe selection. This means that the appropriate selected features can be expressed by

$$m = (5 \sim 10) * m_f \tag{5.3.0.4}$$

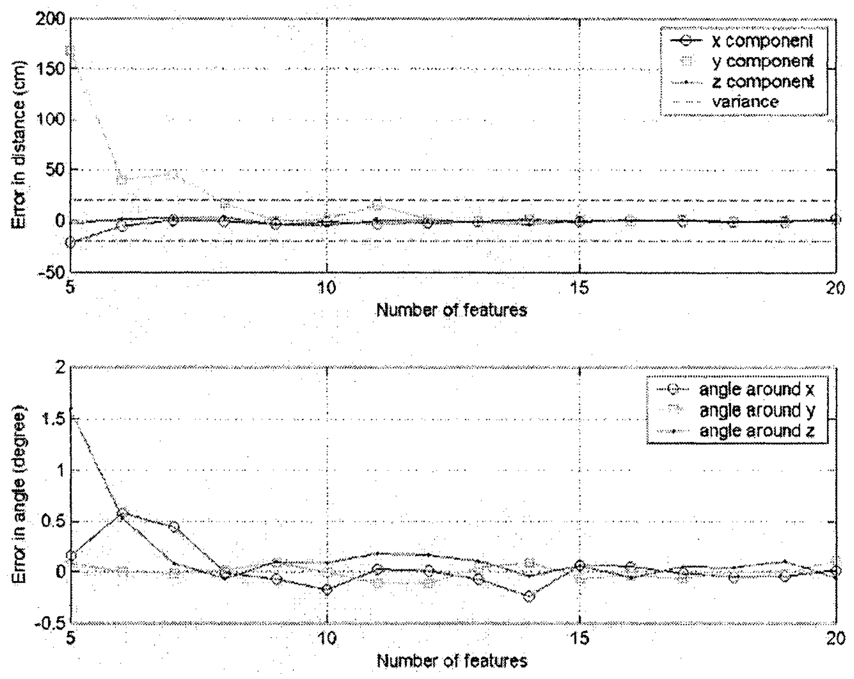


Figure 5.4: Monte Carlo simulation by comparison of the number of selected features.

5.4 Efficient Map Building

A map can be constructed in many different ways. The simplest method is to use all the observed features to form the map, as shown in Fig. 5.6. This is not an efficient method from the point of computation and storage. Three new strategies have been developed in this section. Before the algorithms are designed, we introduce some definitions.

5.4.1 Example for the Map Building

In the following chapter, we take an example from my simulation system [1]. A robot navigates by following a circular space path, and a stereo camera system installed on the robot takes a sequence of images by viewing the ground surface of the working area $100m \times 100m \times 20m$ in length, width, and height. It is assumed that there are 3000 landmarks randomly distributed on the surface floor. All the landmarks and observed landmarks in the robot working area are displayed in Fig 5.5, and all the landmarks observed during the robot's navigation are displayed at Fig. 5.6.

[1]Robot simulation system is introduced in Chapter 6.

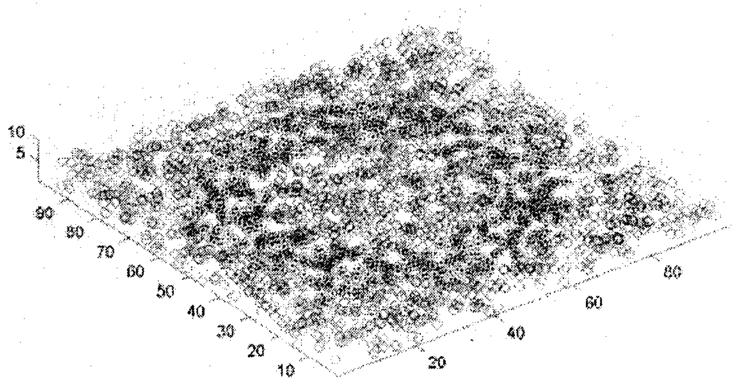


Figure 5.5: All the landmarks and observed landmarks in 3D. The red circles are the landmarks on the floor surface, the blue circles are observed landmarks, and the large green circle is the robot navigation path.

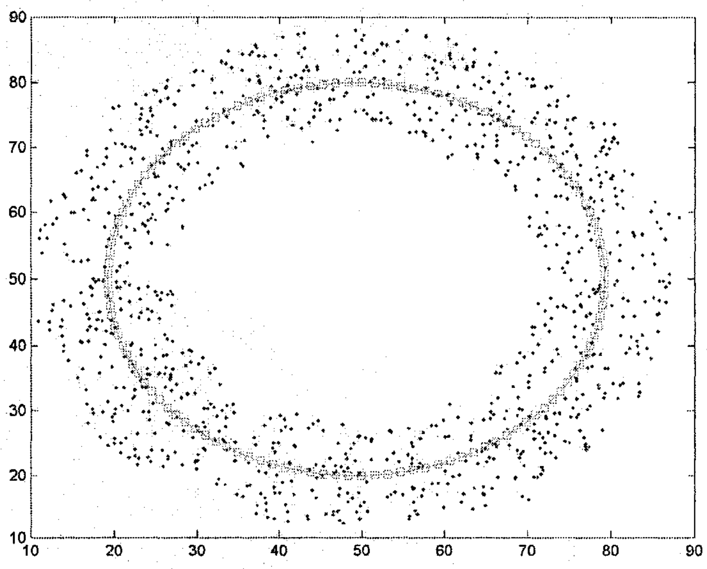


Figure 5.6: Observed features projected to 2D plane. The small blue dots are observed features, and large green circles are the robot navigation path.

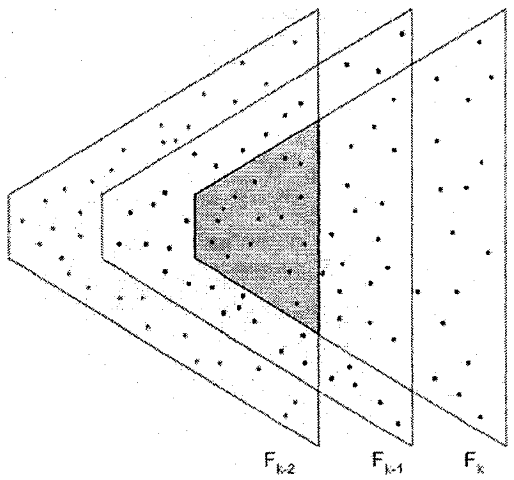


Figure 5.7: Overlapped features with the level of overlap 3. Different colour dots represent the features observed at time stamp $k-2, k-1$ and k. Features at the red area are the features included by the three frames F_k, F_{k-1}, and F_{k-2}.

5.4.2 Definitions

Definition 5.4.2.1 *F_k is a set of features which is observed by a sensor at time k. It is called frame at k. The size of F_k is expressed by $|F_k|$, which is equal to the number of features.*

Definition 5.4.2.2 *$\Omega_k^{p_o}$ is a set of features at k, which also appear at $F_k, \ldots, F_{k-p_o}$, where $p_o \geq 1$. $\Omega_k^{p_o}$ is called features on overlap p_o at k. It can be calculated by the set operation as*

$$\Omega_k^{p_o} = F_k \cap F_{k-1} \cap \cdots \cap F_{k-p_o} \tag{5.4.2.1}$$

For example, if the overlap level is 3, an instance of Ω_k^3 can be displayed at Fig. 5.7, where different colour dots represent the features observed at time $k-2, k-1$ and k. Features at the red area are the features included by the three frames F_k, F_{k-1}, and F_{k-2}, which is also expressed by Ω_k^3.

Definition 5.4.2.3 *G is a set of features which is stored in a database for mobile robot navigation, which is also called a* map. *A map can be expressed as*

$$G = \{L_1, L_2, \cdots, L_m\} \tag{5.4.2.2}$$

where L_i $(i = 1, \cdots, m)$ are the features in $\boldsymbol{R}^3$.

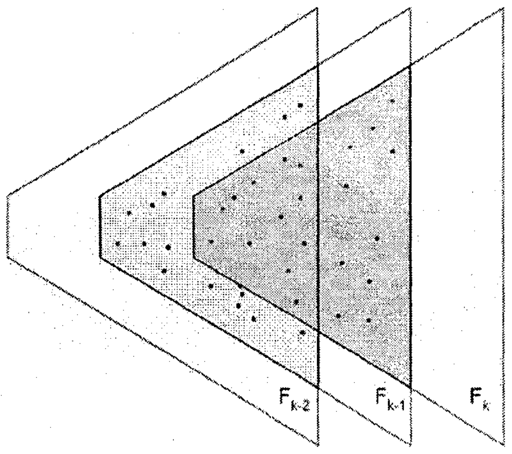


Figure 5.8: New features at time stamp k with the level of overlap 2. Features at the red area is Ω_k^2, and features in blue area is Ω_{k-1}^2.

Definition 5.4.2.4 $G_k^{p_o,m}$ *is a set of features in* $\Omega_k^{p_o}$*, and is also called as* sub map at k*, where the size of the sub map is* $|G_k^{p_o,m}| = m$.

Definition 5.4.2.5 $\Omega_k^{p_o,new}$ *is a set of features in* $\Omega_k^{p_o}$ *and not in* $\Omega_{k-1}^{p_o}$*, which is called* new features on overlap p_o at k. *It can be obtained by set operation as*

$$\Omega_k^{p_o,new} = \Omega_k^{p_o} \setminus \Omega_{k-1}^{p_o} \tag{5.4.2.3}$$

This definition can be understood by Fig 5.8, where Ω_k^2 is represented by the features in the red area, and Ω_{k-1}^2 is represented by the features in the blue area; And $\Omega_k^{2,new}$ represents the features in the red area which does not include the features overlapped by the blue area.

5.4.3 Feature Selected by Overlap

This is the simplest way to construct a map for a robot navigation. Suppose that the overlap level is p_o, then any features observed p_o consecutive times will be added into the map. Therefore the map G can be formed as

$$G = \Omega_{p_o}^{p_o} \cup \Omega_{p_o+1}^{p_o} \cup \cdots \cup \Omega_k^{p_o} \cup \cdots \tag{5.4.3.1}$$

or

$$G = G \cup \Omega_k^{p_o} \tag{5.4.3.2}$$

where k is the time stamp which is bigger than p_o. In this case, if a feature is observed more than p_o times, then it is added to the map. The algorithm to construct a map by this mechanism is shown in Alg. 5.

Input: A sequence of frames F_k, $k = 1, \cdots, N$, and overlap level p_o
Output: A map G

```
1: initialize map G=0
2: for each observation $k$ from $p_o$ to $N$ do
3:     data association among features in $(F_k, F_{k-1}, \cdots, F_{k-p_o})$, a feature observed
       by all $p$ times is added to $\Omega_k^{p_o}$ by Eq. (5.4.2.2).
4:     perform set operation by Eq. (5.4.3.2) to get map $G$.
5: end for
```

Algorithm 5: Map building by selecting features observed p_o consecutive times

Taking the observed data from Fig. 5.6 in the previous example, and implementing Alg. 5 for different overlap levels from 2 to 10, its statistic results are shown in Fig. 5.9 and Fig. 5.10. It is easy to see that the higher overlaps level p_o, the fewer features it will select each time to be added to map G. It must be pointed out that there are many parameters that will affect the number of features which will be added to a map; they include relative pose between two consecutive observations, and effective range of sensor.

It should be easy to obtain an appropriate map size by selecting the proper overlap level in a given situation from Fig. 5.9. Therefore, with a high level of overlap p_o, the features added to a map at each time should be small. There is another benefit to using a high level of overlap in map building - it is possible to form a local cycle during the robot pose and feature position estimation(Fig. 5.11), and this local cycle can be used to align the map to ensure it is consistent [91].

The disadvantage of using a high level of overlap is that in some areas there may not exist any features to add to a map, since in real applications, features in the working areas are not always evenly distributed. Therefore, the selected features

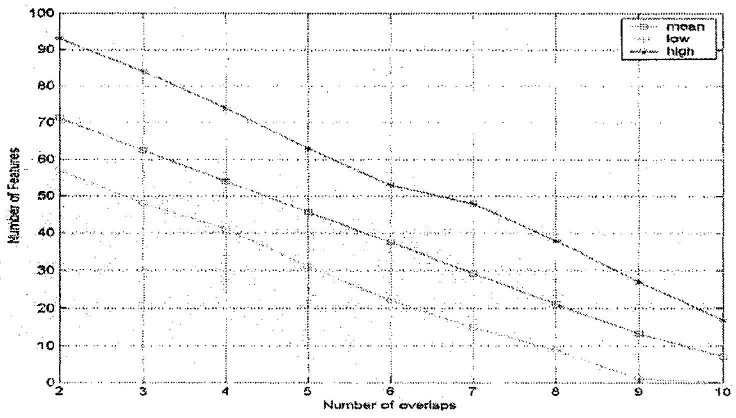


Figure 5.9: Average number of features for different overlap level.

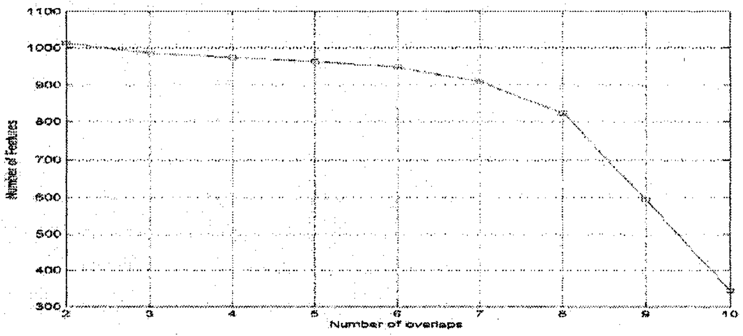


Figure 5.10: Total number of features in a map for different overlap levels.

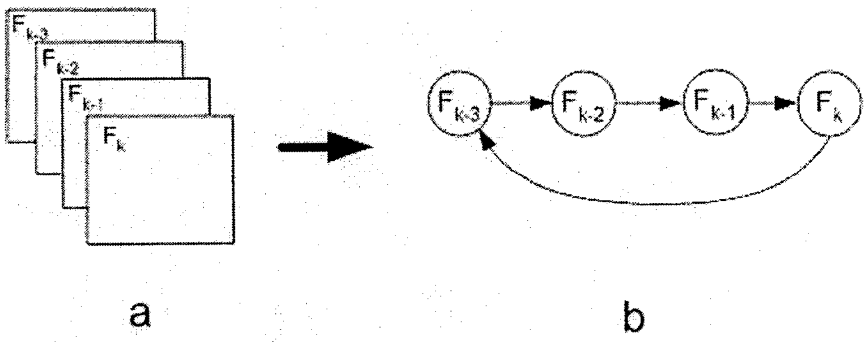


Figure 5.11: An instance of a local cycle from multi-frame, where the overlap level is 4.

observed by 3 time will be more safe and reasonable [78], but we can see that with overlap level 3 in Fig. 5.9, the average number features which will be added to a map each time is about 65. This is still much more than the algorithm needs; however, this problem will be solved by strategies in the following subsections.

5.4.4 Feature Selected with Fixed Number by Stratified Sampling

From the previous methods, we know that selecting a very high level of overlap is not the best way to construct an effective map. However, a low level of overlap will generate a map that is larger than required. At this point, a new algorithm is designed to construct an efficient and effective map.

The basic idea of this method is described as follows: with a low level of overlap p_o, a set of features (usually larger than we want, m) will be obtained each time. Then we randomly select m features from these for the map building. This algorithm can be implemented in four steps:

First, set a low level of overlap, p_o, such as 3 or 4, which will ensure that more than enough features will be obtained for most of the time. Second, compute the features at any time stamp k with the level of overlap, p_o, by Eq. (5.4.2.2), $\Omega_k^{p_o}$. Third, randomly select m features in $\Omega_k^{p_o}$, which is a sub map at k, $G_k^{p_o,m}$. Finally, the $G_k^{p_o,m}$ is added into map G by Eq. (5.4.3.2).

We must be careful in the third step to select features in $\Omega_k^{p_o}$. When the time step k is equal to p_o, this is the first time to select features for a map. If m features are needed from each frame F_k, m features can be selected randomly without replacement from $\Omega_k^{p_o}$, then we can obtain a sub map, $G_k^{p_o,m}$. When the time stamp k is bigger than p_o, we cannot select m features from $\Omega_k^{p_o}$ (where $k > p_o$) such as the case at $k = p_o$.

For example, assume that $m = 10$ features are needed for a robot localization computation. At time $k-1$, 10 features have been selected in the blue area for the map building, and these selected features are expressed with red dots at Fig. 5.12. If

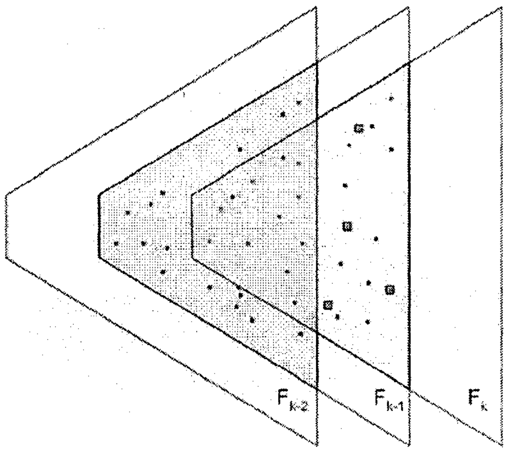


Figure 5.12: Features select on overlap level of 2 at time stamp k. F_{k-2}, F_{k-1}, and F_k represent the set of features projected to a $2D$ plane at time $k-2$, $k-1$, and k, respectively. The dots in the blue area represent the features on overlap 2 at $k-1$, Ω^2_{k-1}. The dots and squares in the yellow area represent the features on overlap 2 at k, Ω^2_k. Red dots are randomly selected features at time $k-1$ from Ω^2_{k-1}. Small blue squares are the features randomly selected at time k from $\Omega^{2,new}_k$.

another 10 features are selected at time stamp k with the same method, the selected features in the area where the blue and yellow areas are overlapped (Fig. 5.12) may be different from the features which are already in that area; we should avoid these duplications. At this time, we need to count the selected features from the overlapped area expressed by blue and yellow. This number is expressed by n; then, we only need to select $m-n$ features for the map from the yellow area which is not overlapped by the area of blue. The algorithm to select features in this case is summarized in Alg. 6.

Taking the observed data from Fig. 5.6 in the previous example, and implementing Alg. 6 by setting the overlap level to 3 and selecting 20 features from each frame, the final map after a robot finishes a circle path is shown in Fig. 5.13. The size of this map is 472. For the overlap level 3, the size of the map by Alg. 5 is about 1000 in Fig. 5.10; therefore this algorithm (Alg. 6) generates a much more concise and efficient map.

Input: A sequence of frames F_k, $k = 1, \cdots, N$, number of feature m, and overlap level p_o.

Output: A map G.

```
1: initialize map G=0
2: for each observation $k$ from $p_o$ to $N$ do
3:   data association among features in ($F_k$, $F_{k-1}$, $\cdots$, $F_{k-p_o}$), a feature observed
     by all $p$ times is added to $\Omega_k^p$ by Eq. (5.4.2.2)
4:   if k=p then
5:     randomly select $m$ features from $\Omega_{p_o}^{p_o}$, assigned to $\Omega_k^m$
6:   else
7:     find new features on overlap $p_o$ at $k$, $\Omega_k^{p_o,new}$, by Eq. (5.4.3.1)
8:     find the number of features in current map which is in $\Omega_k^{p_o}$, assigned to $n$
9:     randomly select $m - n$ features from $\Omega_k^{p_o,new}$, assigned to $G_k^{m-n}$
10:  end if
11:  do set operation by Eq. (5.4.3.2) to get map $G$
12: end for
```

Algorithm 6: Map building with Fixed Number by Stratified Sampling

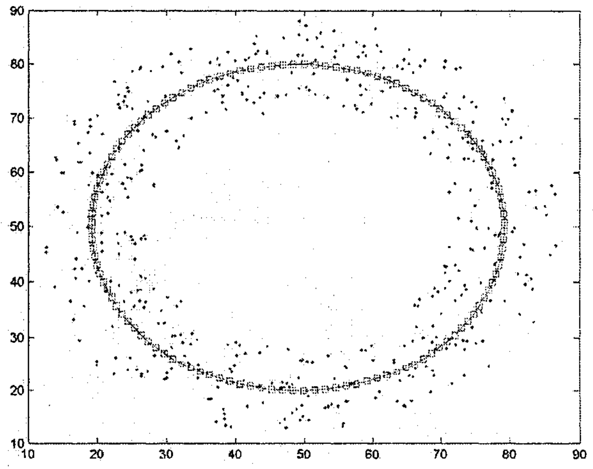


Figure 5.13: A map constructed by randomly selecting 20 features from each frame on overlap level 3. Total features in the map are 472.

5.4.5 Feature Selection for Even Distribution

Another problem in the previous method is that the two selected features may be too close. For this reason, a new algorithm is introduced to make sure that all the selected features are evenly distributed in the view space of camera in the map.

Assume that there are N features in $\Omega_k^{p_o}$, and m features are needed to be selected for a map. All the features in $\Omega_k^{p_o}$ can be grouped into m clusters where Euclidean distance is used as an evaluation parameter. The feature with the shortest distance to the centre of its cluster is selected for the map.

Input: A set of features in $\Omega_k^{p_o}$ with size N.
Output: m selected features, S.

```
1: initialize $S = 0$
2: $m$ clusters generated with agglomerative hierarchical clustering method
3: for each cluster do
4:     calculate its centre
5:     find a feature in the cluster which has the shortest distance to its centre
6:     add the feature into the map
7: end for
```

Algorithm 7: Selecting features with even distribution in space.

In the feature selection processing, some isolated features which are far away from other features should be selected with high priority, which will provide a more robust estimation [37]. In this case, the agglomerative hierarchical clustering method [36] will be used. An instance of clustering for this case is shown in Fig. 5.14, where feature A and feature B should be selected with high priority if more than 3 features need to be chosen in the set of features.

Integrating this algorithmAlg. 7 into Alg. 6, and implementing it with the observed data from Fig. 5.6 in the previous example, the final map after a robot finishes a circle path is shown in Fig. 5.15. The feature distribution in this map is much more

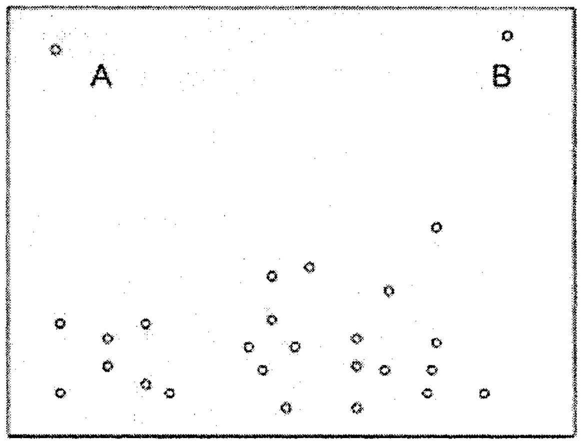


Figure 5.14: An instance for clustering, where feature A and feature B should be selected with high priority if more than 3 features need to be chosen in the set of features.

evenly distributed than in Fig. 5.13 with the algorithm of simply random selections.

It is hard to determine which of the maps in Fig. 5.13 and Fig. 5.15 is better. The Monte Carlo simulation method was used to check the average smallest distance D_{min} in different features from 10 to 30 selected from each frame, and overlap level $p_o = 3$ for both algorithms in subsection 5.4.4 and subsection 5.4.5. The Monte Carlo simulation results and their related variance is shown in Fig. 5.16. The R_{min} by red color represents the results from the method of feature selection randomly and the O_{min} by green color represents the results from the method of feature selection for even distribution. In all test cases, the average minimum distance of the features in the map by the algorithm in subsection 5.4.5 is larger than in subsection 5.4.4. By using statistic hypothesis testing, we can say that the average smallest distances are different for these two methods with 95% confidence.

5.5 Summary

Map building is a very important component of 3D SLAM. For a large work area in an outdoor environment, organizing some features for an efficient and reliable map is the issue we have discussed in this chapter. Based on the requirement for reliable robot pose estimation, several algorithms to build an efficient map were developed for mobile robot navigation. These algorithms were tested by using the data from our

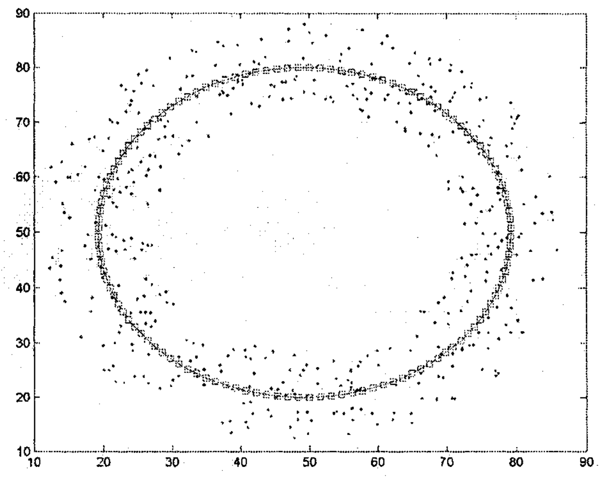


Figure 5.15: A map constructed by selecting 20 features evenly distributed from each frame on overlap level 3. Total features in the map are 472.

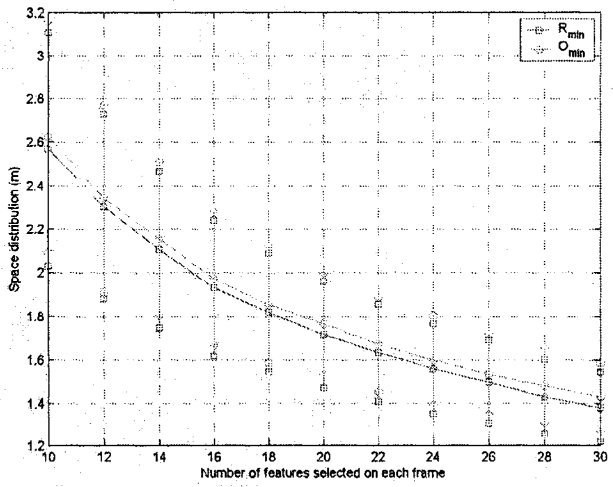


Figure 5.16: Monte Carlo simulation from selected features minimum distance comparison between random or even selection algorithm.

mobile simulation system (which will be introduced in the next chapter). Of the two optimized efficient maps, the map with evenly distributed features is much better for expressing features in the robot's working area, as it will provide a more robust pose estimation than the map with randomly-selected features.

Chapter 6

Simulation

In order to provide a convenient tool for evaluating 3D SLAM algorithms, a 3D simulation system (SLAM_3DSim) was developed, in which a robot equipped with sensors such as stereo camera, radar, sonar, laser range finder, and/or IMU can navigate in a pre-defined environment such as space, indoors, outdoors, and underwater by input. The sensors will take measurements during the robot's navigation at various intervals, and the measurements will be saved to a database or a file for the robot and landmarks position estimation.

In this chapter, an introduction of the SLAM_3DSim is presented in section 6.1. By using this simulation system, a robot pose estimation for different environments is presented in section 6.2. SLAM simulation with only stereo camera is presented in section 6.3. SLAM simulation with multi-sensor fusion is presented in section 6.5. Summary is provided in section 6.6.

6.1 Introduction to a 3D Simulation System

SLAM_3DSim is a software system developed to simulate a robot, equipped with multi-sensors, to measure its surroundings while navigating in an unknown environment. The measurement results are stored in a database and used to estimate the robot's pose and establish a map of the environment. This map can be used for robot navigation in the future. Below is a brief introduction of the system. Further details are provided in Appendix C.

6.1.1 Structure of the System

SLAM_3DSim was developed by strictly following the control-view-model design pattern [9] to guide the system design. Control processes and responds to events, such

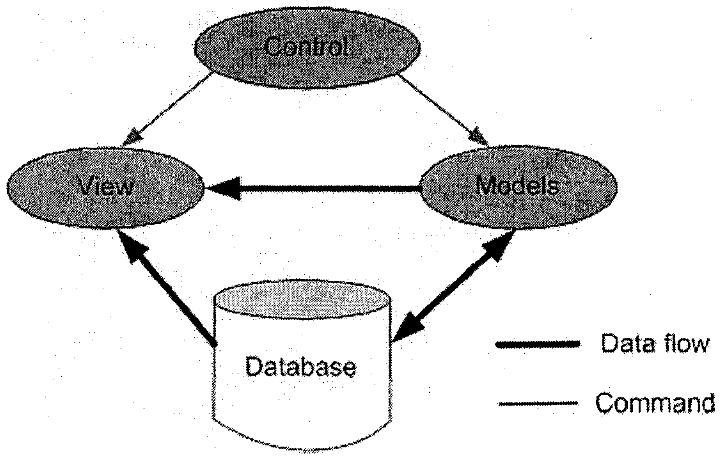


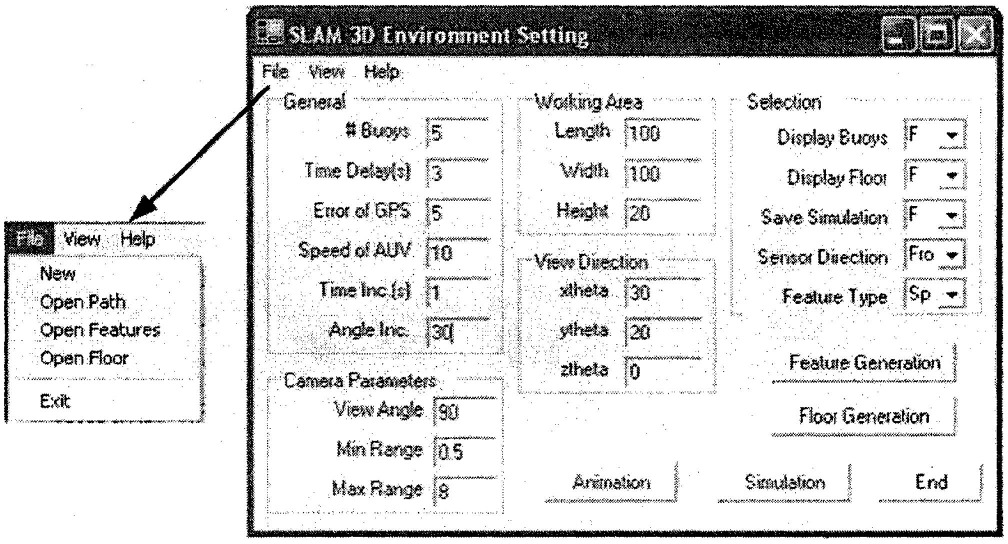
Figure 6.1: The structure of SLAM_3DSim simulation system.

as animation or simulation performance, input and output selections. View renders the model into a form or graph for interaction, such as robot animation displayed in $3D$ which projects to $2D$ in a predefined direction. Model represents the information on which the application operates. In this simulation system, the model includes the predefined robot path, environment setting (floor surface and landmarks), sensor modelling, and sensor measurement. The structure of the system is shown at Fig. 6.1.

6.1.2 Interface of the System

The interface of SLAM_3DSim provides the control and environment settings for different simulation cases. The main interface of the system is shown in Fig. 6.2. There are five groups for the input setting - general setting, working area, camera parameter, view direction, and other selections; and five buttons to implement their corresponding tasks.

An example of the simulation system application is displayed in Fig. 6.3, which is a robot navigating in an underwater environment, with stereo camera on the robot and buoys on the water surface. The small dots are the detected landmarks on the floor surface. The view volume of the sensor, such as a camera, is expressed by the blue cone, and there are five buoys on the water surface which send range information and GPS information to the robot. The detected landmarks are located on the floor



General
Buoys 5
Time Delay(s) 3
Error of GPS 5
Speed of AUV 10
Time Inc.(s) 1
Angle Inc. 30

Camera Parameters
View Angle 90
Min Range 0.5
Max Range 8

Working Area
Length 100
Width 100
Height 20

View Direction
xtheta 30
ytheta 20
ztheta 0

Selection
Display Buoys F
Display Floor F
Save Simulation F
Sensor Direction Fro
Feature Type Sp

Figure 6.2: Main interface of SLAM_3DSim simulation system.

of the bottom (Fig. 6.4). The bottom surface (in Fig. 6.4) is expressed as a wire frame, which is not displayed in Fig. 6.3 for simple and clear reason.

6.2 Simulation Case 1: 3D Robot Pose Estimation

Before implementing the complicated 3D SLAM problem, we start by a simple experiment in which we only want to estimate the robot's pose during its navigation. In this case, the most basic assumption is that there is a complete and accurate map of the environment. This implies that all the features' positions in the working area are known. This assumption is appropriate for real application in the following two situations: (1) all landmarks are artificial landmarks, which have fixed positions; or (2) a robot has navigated the area and implemented SLAM algorithm previously. Based on this assumption, the map is available, and the robot pose will need to be estimated during its navigation.

In this test case, it is assumed that it is an underwater environment with length, width, height of $100m \times 100m \times 20m$. There are 3000 landmarks on the floor surface which are randomly distributed in $2D$ domain space, and five buoys are placed on the surface of water in the experimental area. A robot navigates with speed $2m/s$ and direction of $(0, 3^0, 0)$ in euler angle relative to its current orientation. A stereo camera

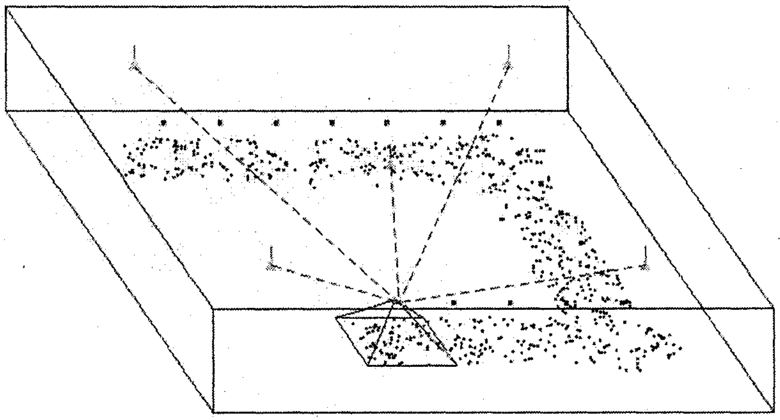


Figure 6.3: Robot navigates in an underwater environment with stereo camera and buoys on surface. The small dots are the detected landmarks on floor surface, view volume is the blue cone, and there are five buoys on the water surface which send range information and GPS information to the robot.

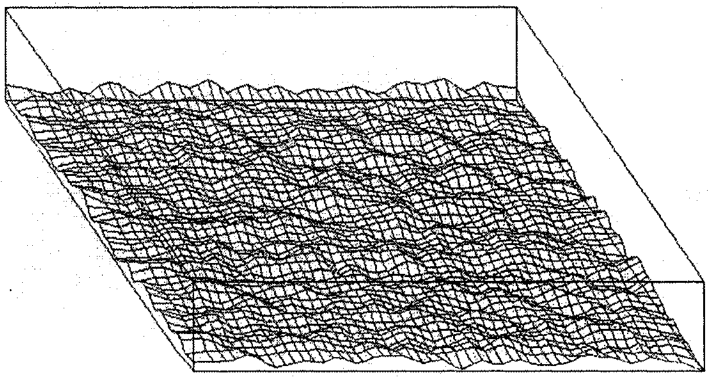


Figure 6.4: The bottom surface which is corresponding to the simulation case in Fig. 6.3. The surface is expressed by wire-frame model.

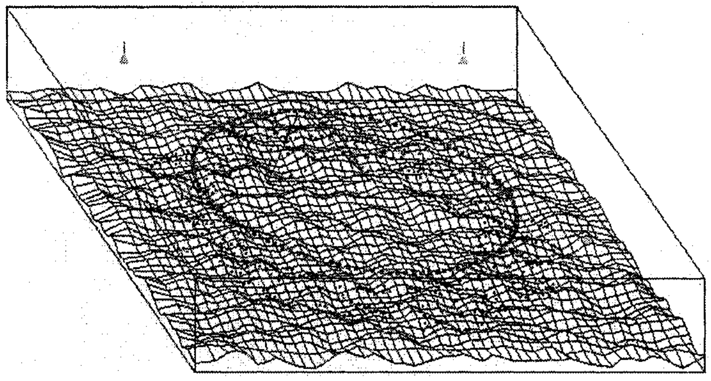


Figure 6.5: Measurement information of a robot after a circle navigation is completed. All the detected landmarks are located on the floor surface. Five buoys on the water surface transmitted range information to the robot every 5 seconds.

sensor installed on the robot views down to the bottom. Once per second, the stereo camera takes a photo to measure the environment; every 5 seconds, the buoys send distance information to the robot. The snapshot of the simulation results for this case is shown in Fig. 6.5. where the detected landmarks located on the floor surface are represented as small red dots. Five buoys on the water surface transmit range information to the robot every 5 seconds, and the floor surface is represented as wire-frame.

During the simulation, it is assumed that the sensors and input error distribution are Gaussian. It is also assumed that the stereo camera's measurement error covariance is expressed by a matrix with diagonal $(0.1m, 0.1m, 0.1m)^2$ and zeros for other entities, the buoy's range measurement error covariance is $0.01m^2$, and the robot input error covariance is $(0.1m, 0.1^0, 0.1^0, 0.1^0)^2$ for speed and three Euler angles.

With the stereo camera's measurement and system input, the robot position is estimated by implementing the particle filter with the process model and measurement model described in chapter 4. With the measurement of buoys on the water surface, the robot position is estimated by using the extended Kalman filter with the method described in chapter 3. The robot position is also estimated by using sensor fusion from both camera and buoy measurements. The estimated path with these three different approaches is shown in Fig. 6.6.

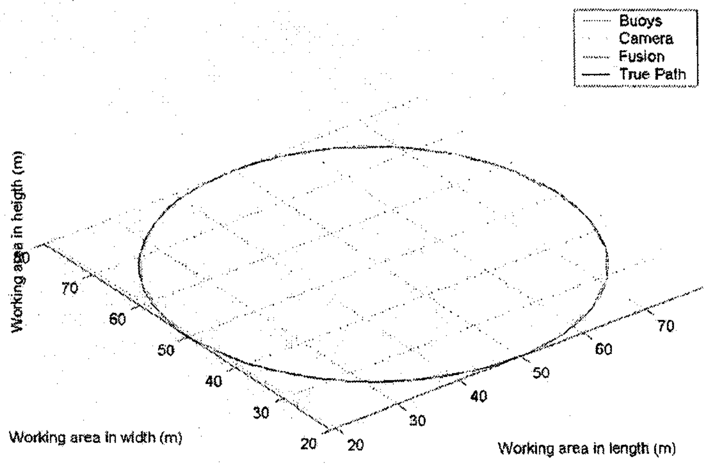


Figure 6.6: Robot path estimation in 3D with different methods in case 1.

It is very difficult to see the differences in these three paths in Fig. 6.6. Robot pose includes position and orientation. The position has components (x, y, z), and the orientation is expressed in Euler angle $(\theta_x, \theta_y, \theta_z)$ around x axis, y axis, and z axis. In order to have an overview of the robot's pose information, the errors for each component for a camera estimation and for a fusion of the camera and buoys are displayed at Fig. 6.7. From this figure, we can see that there was not much difference in the position components, but the orientation components showed obvious differences.

In order to determine whether the results from the three methods are different or not, a position error was used to evaluate the different approaches. The position error was computed between the actual robot position and the estimated position as soon as each measurement was taken. These errors are shown in Fig. 6.8. We observe that buoys provided the highest accuracy estimation among the three approaches. For most times, the camera's measurement is used for estimation of the robot pose at regular intervals. When the buoy's measurement has been determined, EKF is used to estimate the robot position. Then, this position information is fused with the position information from the camera. From the results in Table 6.1, the mean errors for camera and sensor fusion are very close. We can use statistical hypothesis testing of two populations [58] to check whether the mean error from sensor fusion is smaller than the mean error from the camera. The sample size (steps of navigation) is 121,

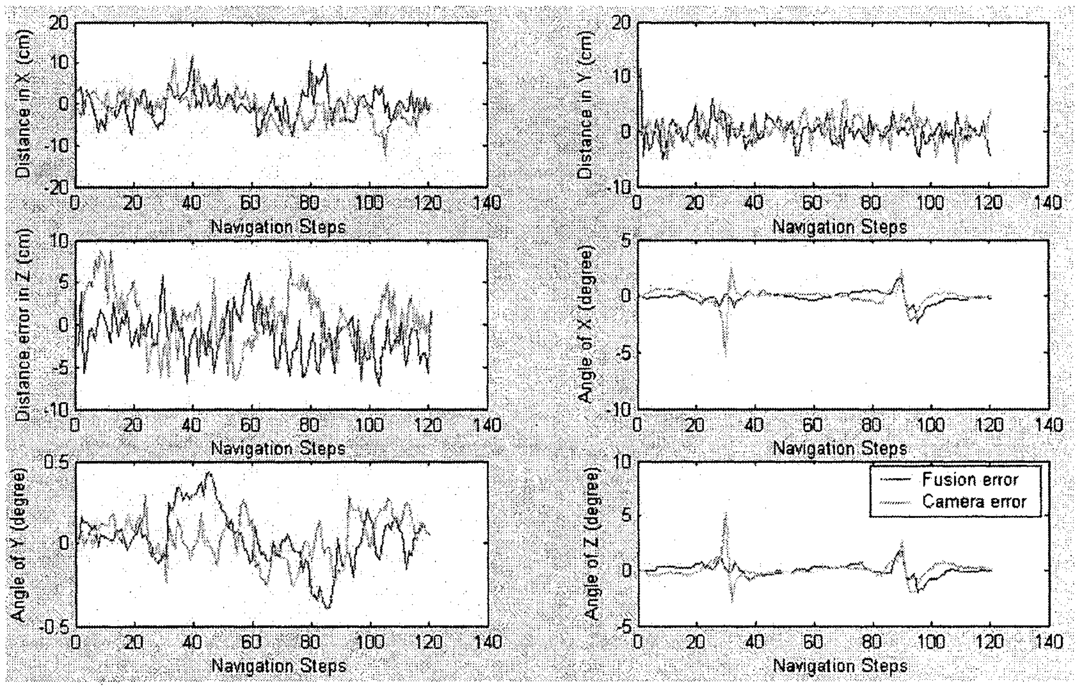


Figure 6.7: Robot position error with respect to actual position for each component in simulation case 1.

Table 6.1: Statistics of the position estimates obtained from different sensors.

	Buoys	**Camera**	**Sensor Fusion**
Mean(cm)	1.2	5.1	4.5
Covariance	0.25	0.23	0.17

and we assume the confidence level is 99%. The $p-value$ is smaller than 0.001, which is much smaller than 1%. Therefore, we can say that the mean error from the sensor fusion is smaller than the mean error from the camera with a confidence level of 99%.

An explanation is needed for the data displayed in Table 6.1, where the the fusion result from camera and buoys estimate is not better than the estimates based on camera and buoys separately. Let us look at the sampling intervals for each sensor in Fig. 6.8. The sampling interval for the camera is 1s, while for the buoys it is 5s. This means that, the fusion estimates are generated only every 5s, compared to every 1s for the camera estimates, i.e. five times less often.

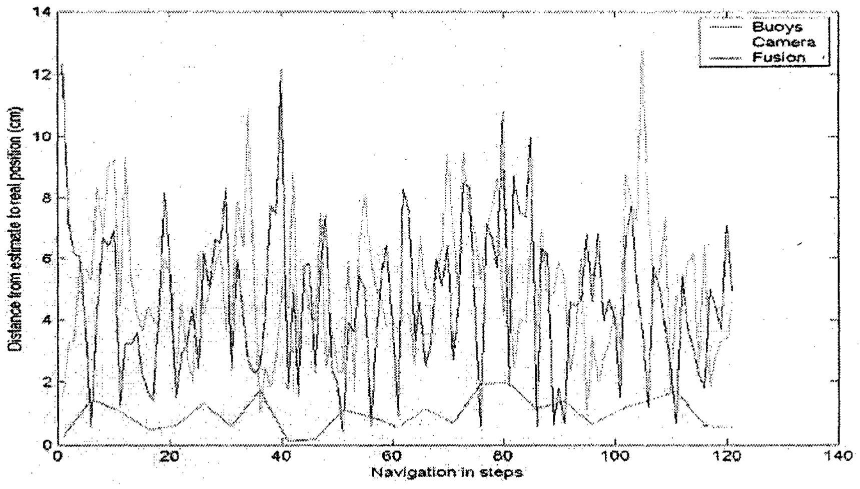


Figure 6.8: Robot position error with respect to actual position in case 1.

6.3 Simulation Case 2: 3D Robot SLAM with Small Measurement Noise

SLAM works in a situation where there is no map of the environment, and the sensor's measurement must be used to estimate both the employed robot pose and landmarks position. Fast SLAM algorithm in Chapter 4 is applied for this simulation. Before it is implemented, several issues need to be determined:

First, how to organize a map for this case. Based on the knowledge from the previous chapter, not all the observed landmarks will be used for the robot's pose estimation. We only select at most m_f landmarks, which are observed over three consecutive time intervals. These landmarks are selected by stratified sampling(Alg. 6 in Chapter 5).

Second, how to add the newly-observed landmarks to a map. we stated in section 4.4 of Chapter 4 that all the observed landmarks at time k ($k > 1$) can be divided into two group: old landmarks and new landmarks. The new landmarks need to be added to the map. From the measurement model in section 4.2 of Chapter 4, the position of landmark i in a global coordinate system can be obtained by

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = M_v(k)^{-1} \begin{bmatrix} Z_{x_i}(k) \\ Z_{y_i}(k) \\ Z_{z_i}(k) \end{bmatrix} + \begin{bmatrix} x(k) \\ y(k) \\ z(k) \end{bmatrix} \tag{6.3.0.1}$$

where $x(k), y(k), z(k)$ are the position of the robot at time k. $M_v(k)$ is the rotation matrix at time k, which is determined by its related euler angle. $Z_{x_i}(k), Z_{y_i}(k), Z_{z_i}(k)$ is the measurement of landmark i at time k by a sensor on the robot. The covariance of the landmark position can be obtained by

$$P_i = P_{vv}(k) + M_v(k) R_i M_v(k)^{-1} \tag{6.3.0.2}$$

where $P_{vv}(k)$ is the covariance of the position of the robot at time k, and R_i is the covariance of measurement error for landmark i which is determined based on the employed sensor.

Environment setting for 3D SLAM is almost the same as in case 1 of section 6.2 of this chapter. The only difference is that the environment for SLAM is unknown, which means that only observation information and robot input can be applied.

The simulation result is displayed in Fig. 6.9, which includes the estimated path, true path, all the landmarks in the map, and the errors for each landmark. The robot started to move at the point $(50m, 20m, 12m)$ in the working area in an anti-clockwise direction.

The map established by SLAM will be used for the mobile robot's navigation. If the map is not accurate enough, it is too difficult to locate the robot's true position; therefore, building a high-accuracy map is one of the most important issues. The path and landmarks errors in this simulation are shown in Fig. 6.10. The path error and landmarks error increased with the processing of the robot navigation.

The newly-observed landmarks at time k, which will be added to a map, are computed based on the robot pose and sensor measurement at that time. If the robot pose has large errors, consequently the newly-added landmarks in the map will have large errors. Furthermore, the inaccurate map built at time k will be used for the robot pose estimation at time $k+1$ and this will increase the error in the robot pose estimation at time $k+1$. Therefore, the most important issue for high-accuracy map building is to have a mechanism or method to estimate a robot's pose accurately.

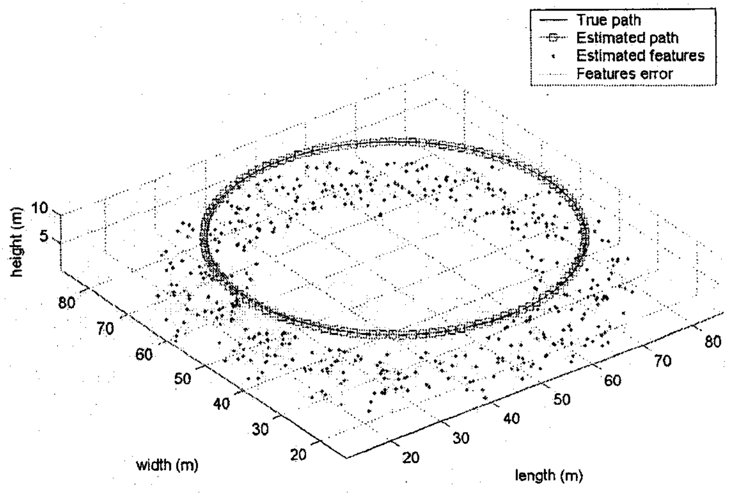


Figure 6.9: Simulation results for robot pose and landmarks position in 3D space in case 2.

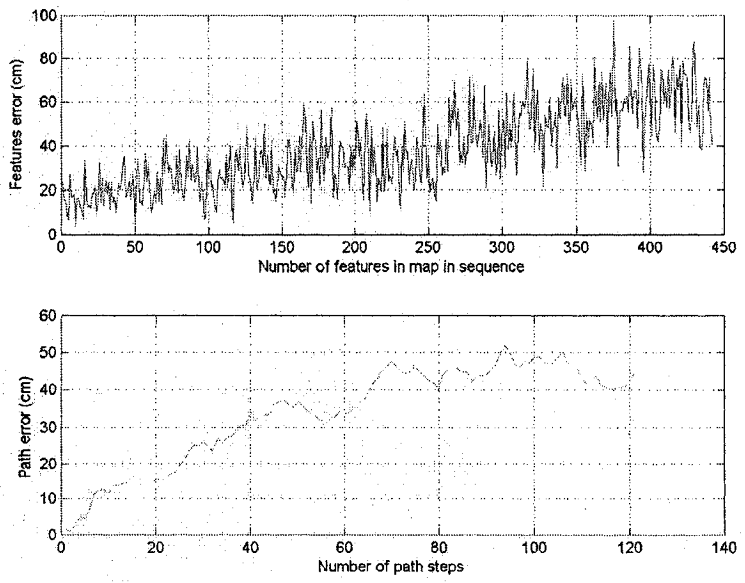


Figure 6.10: Robot pose error and landmark position error of simulation results of SLAM in case 2.

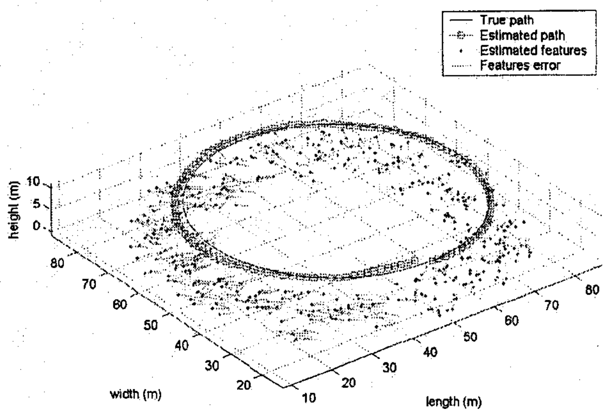


Figure 6.11: Simulation results for robot pose and landmarks position in 3D space in case 3.

6.4 Simulation Case 3. 3D Robot SLAM with Large Measurement Noise

Case 3 has almost the same environment settings as in case 2, and the only difference is that the sensor noise and input noise are bigger than in case 2. In case 2, the error covariance for the sensor is assumed to be expressed by a square matrix with $(0.1m, 0.1m, 0.1m)^2$ as diagonal and 0 for other entities, and the error covariance for robot input is $(0.1m, 0.1^0, 0.1^0, 0.1^2)^2$ for speed and three Euler angles. In this case (case 3), the error covariance for the sensor is $(0.5m, 0.5m, 0.5m)^2$ as diagonal for a square matrix, and the error covariance for the robot's input is $(0.5m, 0.5^0, 0.5^0, 0.5^2)^2$ for speed and three Euler angles. This will be used to investigate the influence of the SLAM results by different sensor errors and system input errors.

The estimated robot poses and landmarks positions in $3D$ for this case is shown in Fig. 6.11. The robot pose and landmarks position projected in horizontal plane is shown in Fig. 6.12. The robot's position error and landmarks position error are shown in Fig. 6.13.

To compare the simulation results in case 2 (Fig. 6.10) with case 3 (Fig. 6.13), the largest robot pose error is about $0.5m$ in case 2, but about $4m$ in case 3. we have

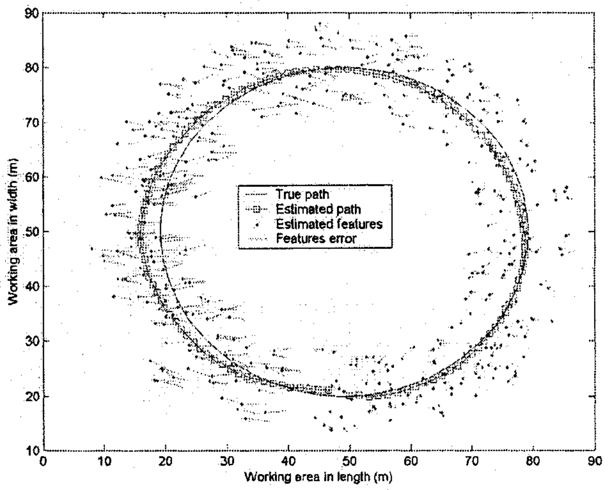


Figure 6.12: Simulation results for robot pose and landmarks position in 2D in case 3.

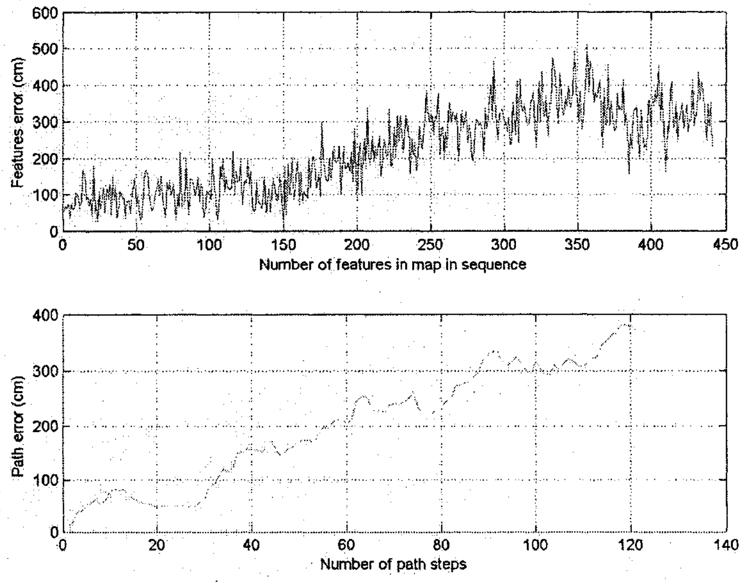


Figure 6.13: Robot pose error and features position error of simulation results of SLAM in case 3.

observed that the robot pose error will directly influence the landmarks position. This claim can be inferred from the estimated landmark error: in case 2, the biggest landmark error is about $0.8m$, while in case 3, it is about $5m$. From these results, we know that it is very important to control the sensor's error and system input's error for a reliable and accurate solution to the SLAM problem.

6.5 Simulation Case 4: 3D Robot SLAM by Sensor Fusion with Small Measurement Noise

Fast SLAM is a very efficient algorithm for a mobile robot in computation, but a robot using only one sensor(such as camera, laser scanner, radar, sonar) will cause its built map to incur large errors when the employed robot navigates for a large area, since the robot pose estimation and landmarks position estimation are based on relative information. For relative information, cumulative errors will be increased with navigation time. In the following case, another sensor, the sonar buoys, will be fused with a camera during the process of Fast SLAM implementation.

In case 4, the simulation setting is the same as in case 2, but there are five buoys placed on the surface of the water (see Fig. 6.3). The buoys send measurement of GPS information and distance measurement to the robot every 5 seconds. This information can estimate the robot's position with EKF. The camera will observe the environment downward and take one photo per second. Fast SLAM algorithm is implemented with the camera's observation. Each time the robot's position is estimated by using the buoys measurements, the information will be fused with the robot's position from fast SLAM. This will cause the fused robot position to be more reliable than the position estimated from only one sensor. The fusion results in a 3D situation are shown in Fig. 6.14.

To investigate the errors in this simulation, the path error and landmarks error are shown in Fig. 6.15 . The largest path error is less than $15cm$, and the largest landmarks error is less than $40cm$. In case 3 where only a camera sensor is used (Fig. 6.10), the largest path error is about $70cm$, and the largest landmarks error is about $80cm$. Therefore, we can say that sensor fusion results in much fewer errors than when only one sensor is used.

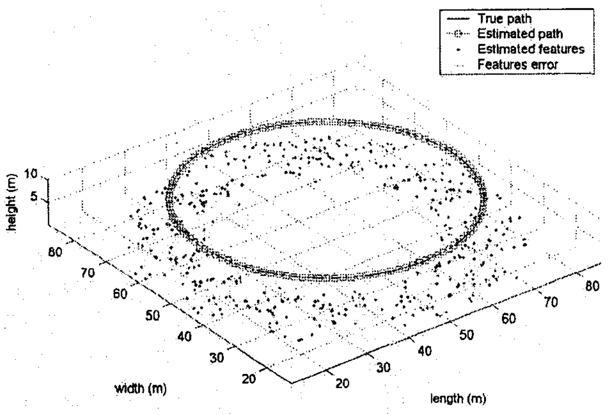


Figure 6.14: Simulation results for robot pose and landmarks position in 3D space in case 4.

Even though the path error and landmarks error in case 4 are decreased greatly, they are not very small. The results in Fig. 6.8 indicate that the largest path error from buoys measurement is less than $2cm$. Why does the buoys' high accuracy estimation not make the fusion results smaller than these results? In Chapter 4, the state vector definition for a robot has six components, three in position and three in orientation. The robot's pose estimated from the camera's measurement includes position and orientation, but the estimation from the buoys measurements only provides the position of the robot. During the fusion stage, only the robot's position fused and its orientation remains unchanged. As we know, the robot's pose estimation and the landmarks position estimation are related to the robot's orientation. The large error in orientation will definitely cause a large error in the landmarks position estimation. The landmark's large error will in turn cause a large error for the robot's pose estimation in its subsequent steps.

It must be pointed out that the cameras implement visual odometry with unbounded error growth, while the buoys implement triangulation with bounded error growth. These may affect the final fusion results.

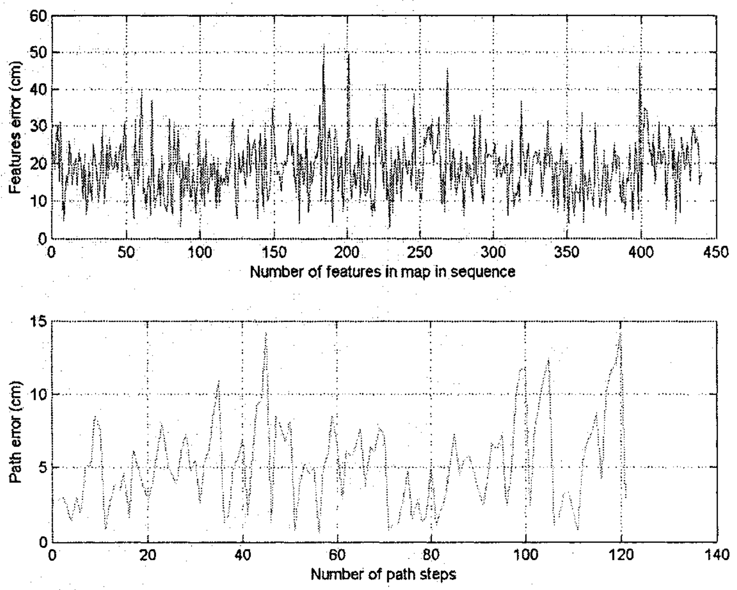


Figure 6.15: A robot and landmarks position error of simulation results of SLAM with sensor fusion in case 4.

6.6 Summary

A mobile robot simulation system developed for 3D SLAM is introduced at the beginning of this chapter. It includes the system design, system structure, interface and environment setting. By using this system, three types of robot navigation cases are simulated: mobile robot pose estimation, SLAM simulation with only stereo camera, and SLAM simulation with sensor fusion.

From the simulation results, we notice that measurement noise has a significant influence on the robot pose and landmark position estimates. To limit sensor error to a small value will greatly improve the estimation accuracy. If the sensor error can be limited to 0.1m, the robot position and landmark position error will be in the range of 1m. If the sensor error is in the range of 0.5m, the robot position and landmarks position error will be in the range of 5m. This error is not acceptable in an environment of the scale of a soccer field (e.g. $100m \times 100m \times 20m$).

Sensor fusion is one of the potential approaches to improve the accuracy of robot pose and features position. By comparing the accuracy using both a camera and buoys with using a camera only, at the same level of camera noise, the former performs much better than the latter.

Chapter 7

3D SLAM in Real Application

Mobile robot experiment in an underwater environment is very difficult and expensive. In order to validate the algorithms developed in this thesis, we designed several experiments in different environments which are focused on different objectives, as follows:

First, we experimented in an indoor swimming pool and an open ocean of Barbados where we focused on the vision visibility test and buoy operation test. The experiment results will guide the whole measurement system design.

The second experiment was the indoor test with arm robot and stereo camera, the objective of which was to validate the SLAM algorithm with sensor fusion. The benefit of this test was that we knew the real position of the robot, and we could control the landmarks in the working environment.

The third experiment was the outdoor test with multi-sensor to check whether the SLAM algorithms for sensor fusion could work in a real and unknown working environment - at a parking lot with on a clear day in summer. We installed several sensors, such as the trinocular camera, GPS, inertial measurement unit, and digital compass, on a garden cart, which we pushed while moving through the parking lot.

This chapter is organized as follows: the water environment experiment is introduced in section 7.1; the indoor experiments of 3D SLAM with an arm robot are presented in section 7.2; and the outdoor experiment of 3D SLAM is presented in section 7.3, ending with the summary.

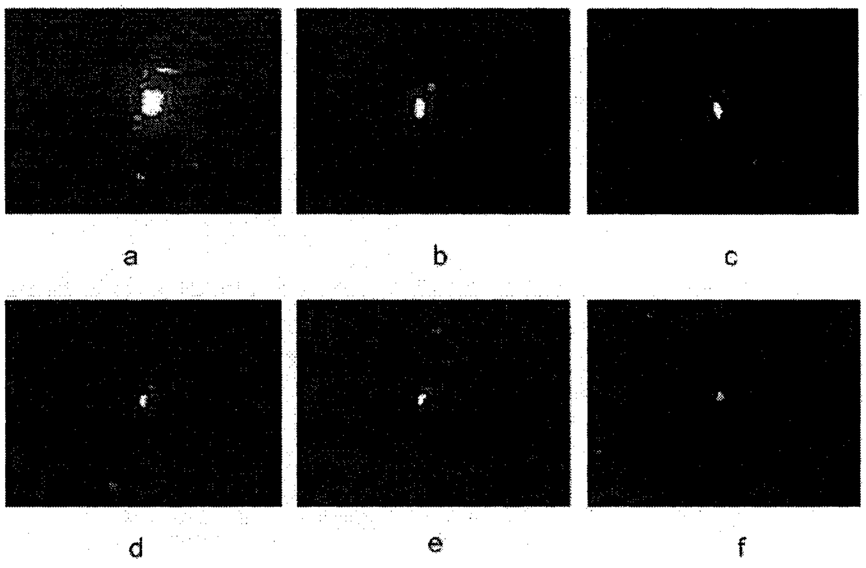


Figure 7.1: Camera visibility experiment in swimming pool with LED. The distance from *a* to *f* are 4.65m, 7.10m, 9.55m, 12.07m, 14.52m, and 16.34m. The view direction is parallel to the surface of water, and center of camera is at 0.6m deep in water.

7.1 Water Environment Experiment

7.1.1 Visibility Test with Camera in Water

The visibility of a camera in water is a very important issue for camera-based robot localization. In open ocean, many things will affect the visibility, such as weather, depth of water, or rough level of ocean. Some measurements were done for the camera visibility in the swimming pool of Dalhousie University. A LED was placed in a transparent plastic box, which was affixed to a wall in the pool. The pictures were taken from various distances in front of the box. The results are shown in Fig. 7.1. From this experiment, it is easy to see that in good light conditions and clear water, a camera can measure the environment in the distances of up to 10 meters.

Some experiments for camera visibility were done in the Atlantic Ocean off western Barbados. An image of the ocean floor, as shown in Fig. 7.2, was taken in clear weather at a depth of approximately 3 metres from a distance of approximately 4 metres. From this image, the landmarks on the ocean floor in that coral reef area are clear and distinct. This means it is possible to select enough landmarks in this area

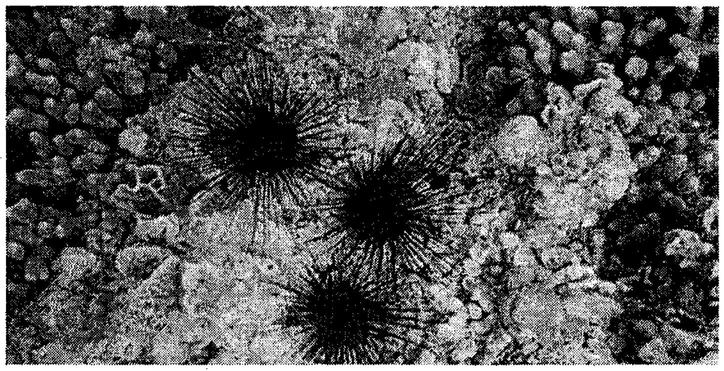


Figure 7.2: Camera visibility in ocean.

to which image-based robot localization can be applied. However, not all areas of the ocean floor have distinct features, and not all the images are clear enough to extract distinct features.

7.1.2 Configuration of an Underwater Robot System for SLAM

An autonomous mobile robot is usually equipped with many different sensors. To-date, the most popular sensors which are used in the mobile robot community are stereo cameras, laser range finders, sonar detector sets, strap down inertial navigation sensors, and digital compasses. The underwater robot system, in addition to the previously mentioned sensors, should be equipped with an acoustic system which can measure the distance from the robot to the buoys on the surface of the water. The buoys are equipped with GPS, compass, and inertial navigation sensor so that their position and orientation can be measured in a global coordinate frame. Therefore, integrating all the sensors equipped on a robot is a key issue for a reliable and robust SLAM solution.

A typical system configuration of SLAM for an underwater autonomous robot is shown in Fig. 7.3. All the sensors used by this system are divided into two groups according to their employed subjects: robot and buoy. The list of sensors and their associated purposes are explained in Table 7.1.

From our experiment in water, we know it is possible to solve the SLAM problem in an underwater environment with information fusion from multi-sensor measurement

Table 7.1: Sensor for SLAM of an underwater robot

Name	Name of Sensors	Purpose
Robot	Trinocular stereo camera	Local SLAM
	Inertial measurement unit (IMU)	robot pose estimation
	Digital compass	IMU's calibration
	Projector	Communicating with hydrophone
Buoy	Hydrophone	Receiving the signal from projector
	GPS receiver	buoy's absolute position
	Inertial measurement unit (IMU)	buoy's orientation
	Digital compass	IMU's calibration

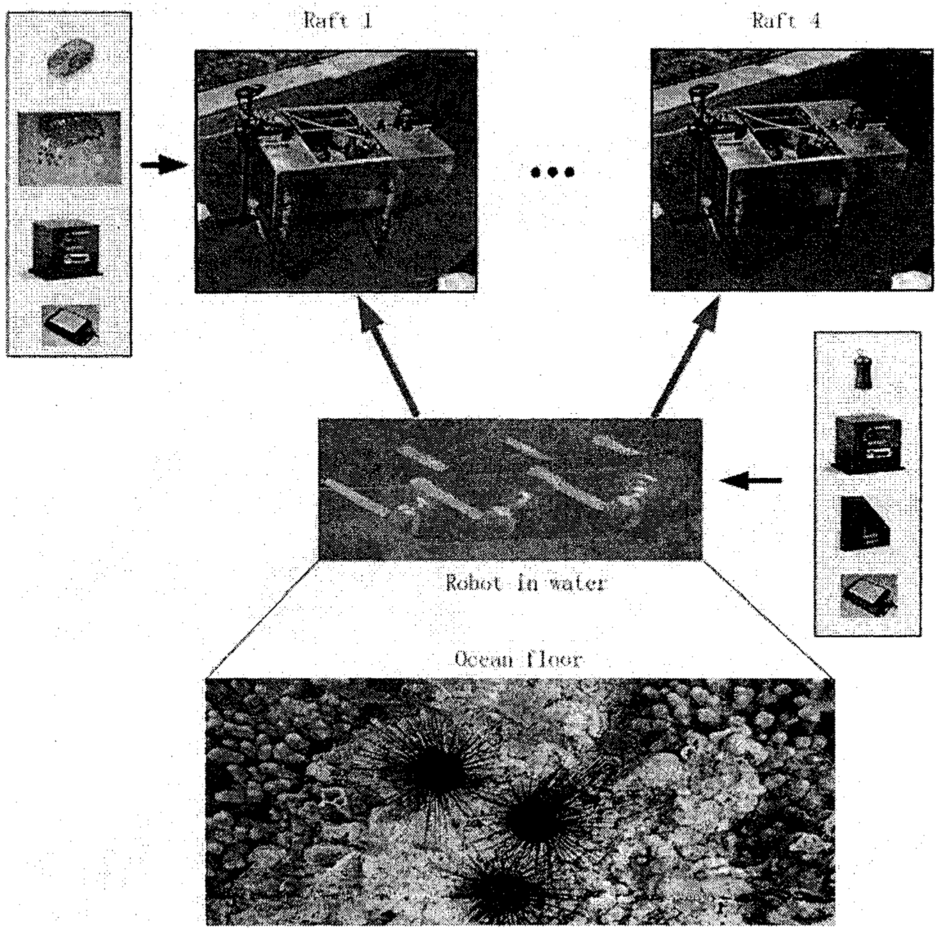


Figure 7.3: System configuration for underwater robot 3D SLAM. Four sensors installed on a raft list on the right of the raft, which are a GPS receiver, a hydrophone set, an inertial measurement unit and a digital compass. Four sensors installed on the robot list on the right of the robot, which are low frequency projector, an inertial measurement unit, a digital compass and a trinocular stereo camera.

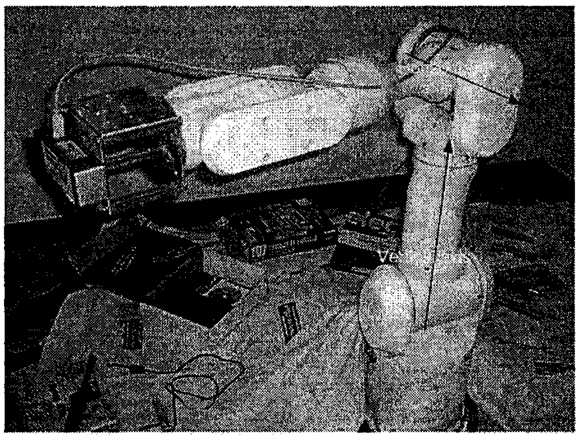


Figure 7.4: PA10-7CE Arm robot and environment set-up.

if the vision sensor can obtain sufficient features in its working environment. The sensors used in this case are the stereo camera system installed on the robot, and the buoys on the surface of water. During the robot navigation, the buoys will measure the robot's position each time, according to the buoys' working frequency. The measurements from the stereo camera will estimate the robot's position and establish a map, using the SLAM algorithm, in a local frame when the environment provides enough matched landmarks. The globally-consistent robot localization and map can be estimated by the algorithm presented in Chapter 4.

7.2 Indoor Experiment with Arm Robot

Some experiments were done in the robot lab at Dalhousie University, using an arm robot, equipped with sensors as for the underwater robot.

7.2.1 Environment Setting

The lab experiment was performed with a BumbleBee camera system mounted on a Mitsubishi PA10-7CE Robot. The set-up for this experiment is shown in Fig. 7.4. In the indoor environment, we could control the light appropriately for a vision test. For the robot's position and map estimate, the working environment would have enough landmarks. As shown in the Fig. 7.4, we placed many objects on the lab floor such as tools, boxes, and books as the landmarks for the experiment.

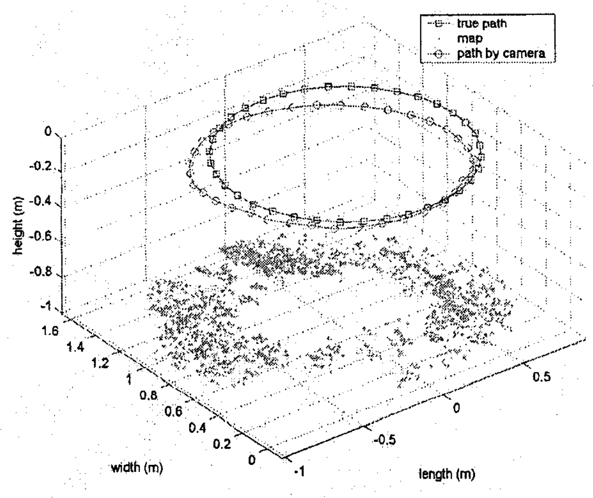


Figure 7.5: Robot path and map with SLAM algorithm in 3D.

7.2.2 SLAM with Stereo Camera

In this case, the robot rotated in a circle within a radius of 0.678m, every 10 degrees for the camera to take a pair of images.

By using the method to solve the SLAM problem introduced in the Chapter 4, the estimated robot 3D position and 3D map are displayed in Fig. 7.5. The results projected to 2D plane are displayed in Fig. 7.6. In this implementation, we used all the features which had been observed twice in the image sequence. In this case, the established map has 2025 features in the image sequence with 36 images.

The error of estimated robot path can be obtained by comparison of the values between the true path and the estimated path, which is shown in Fig. 7.7. From the results, it can be observed that the path error between the beginning point and step 9 or step 26 is very large, while the path error between the beginning point and the ending point from 3D SLAM is very small. We note that from the robot's beginning point to ending point the robot path forms a loop as shown in Fig. 7.5. Because of the error in orientation, the error at the points where the path of the loop is achieved is much less than at some other point in the loop. From this viewpoint, if we only use the closed loop information to optimize the path and map estimation by the method

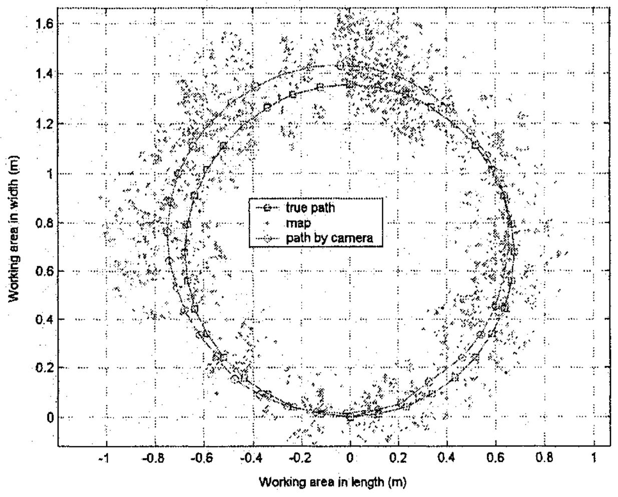


Figure 7.6: Robot path and map with SLAM algorithm in 2D.

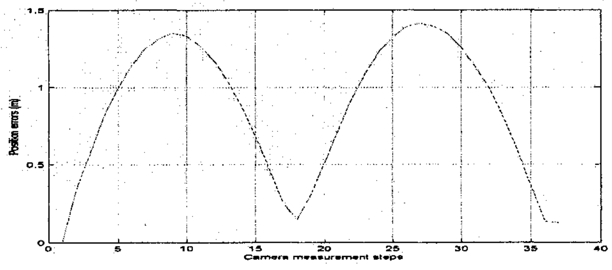


Figure 7.7: Error of the estimated robot path.

of Lu & Milios [53], it will not achieve the desired results with few errors. This will be discussed in the summary of this section.

7.2.3 SLAM with Sensor Fusion

Besides the stereo camera, another sensor is used to measure the position of the head of the arm robot every five seconds. This is similar to the underwater robot which has buoys to measure its position with low frequency.

The environment set-up for sensor fusion is somewhat different from that of SLAM with only the stereo camera, where in all the working areas of the mobile robot, the environment set-up ensured that in every position of the robot, the images taken by

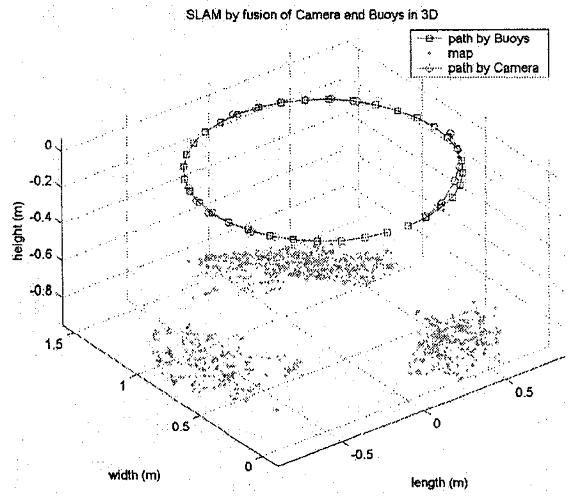


Figure 7.8: Globally-consistent 3D robot path and map with sensor fusion.

the camera would have enough SIFT features. With sensor fusion, it is not necessary that every image have enough SIFT features.

In this case, the measurements from the stereo camera cannot continuously estimate the pose of the mobile robot because in some areas there are not enough SIFT features in the images. Therefore, the stereo camera can only be applied effectively to estimate its local pose and to build a local map. The measurement from the range sensors can be used to estimate the robot's position in the global coordinate system. From the algorithm 4, by fusing the results from the stereo camera and range sensors, it is possible to build a globally-consistent map. The SLAM result from fusing the measurements from a stereo camera and range sensor is shown in Fig. 7.8. From the results, we can see that a global map is integrated from three local maps. Projecting all the paths and map information on the ground plane, we obtained Fig. 7.9.

The estimated path error with sensor fusion is much less than with using only the camera. The greatest path error is approximately 0.24m in Fig. 7.10, while in Fig. 7.7, the greatest path error is approximately 1.3m. Therefore, sensor fusion can provide much more reliable estimation than with using only one sensor.

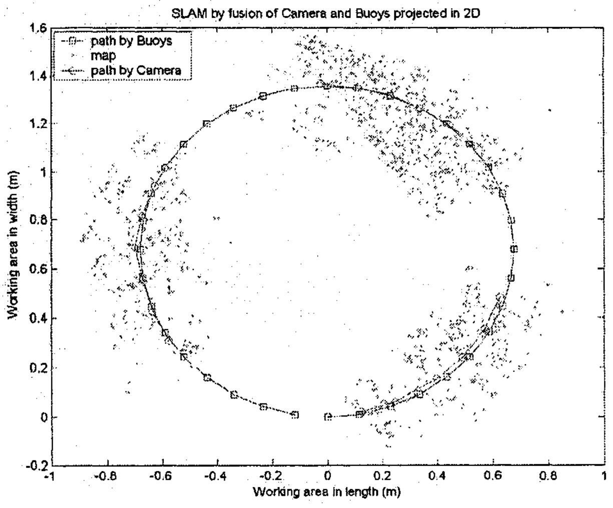


Figure 7.9: Globally-consistent 3D robot path and map with sensor fusion projected in 2D.

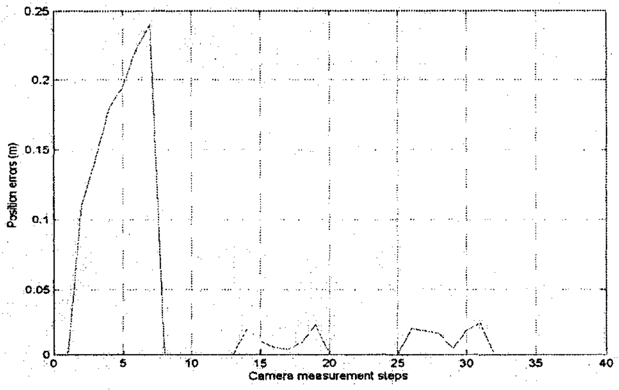


Figure 7.10: Estimated path error with sensor fusion.

7.2.4 Summary and Discussion for SLAM from Lab Experiment

The algorithm for globally-consistent SLAM is appropriate in a lab environment. In the lab environment, we can set the working space according to our design. For instance, we could not obtain enough matched SIFT features in two consecutive images if we did not place objects in some areas. In this case, the SLAM algorithm 4 developed in Chapter 4 needs to be applied. The results (Fig. 7.8) from the lab experiment data set indicate that the new algorithm can be used in a lab environment.

Sensor fusion can obtain accurate SLAM result. By comparing the results from single stereo camera and from sensor fusion, the error for estimated position from sensor fusion is much smaller than the error from single sensor.

Sensor fusion can avoid the limitation of graph-based optimization method for globally-consistent SLAM. For the robot trajectory which forms a loop in the navigation, from the sensor's measurement, estimated results will not form a loop because of the measurement error, model error, and computational error. Usually, a graph-based optimization is applied to build a globally-consistent robot path and map [53] [75] [91]. This is a constrained optimization problem, where the objective function to be minimized is the likelihood function between the estimated robot pose and the consistent robot pose with the constraints that the first and last positions of the robot should be the same, and the first and last orientations of the robot should be in the same direction. The solution of this optimization is the globally-consistent robot path and map. The requirement in this method is that the robot path should form a loop or an approximate loop, but the initial estimated result that does not involve loop closure may have a large error. Then, this method will distribute the final error of the initial estimation into each step of the path.

In the case where the estimated robot path forms a loop or an approximate loop (which means the estimated robot path error between the first and last positions is very small), but the estimated robot path error in the other positions is very large, the previous optimization method cannot decrease this error. An example of the case is shown in Fig. 7.7. By using the sensor fusion approached developed in this thesis, the limitation of the graph-based optimization method can be avoided, such as in Fig. 7.10,where the robot's path error is very small in every position. Another example is where the consistent 2D map is built by using a compass to measure orientation [22].

Environment set-up is very important for SLAM in the lab. There are many issues which will affect the quality of the robot SLAM in the lab via environment set-up. First, we should place enough landmarks in the working space to make sure that more than enough SIFT features can be extracted from the associated images. Second, we should control the moving speed of the robot to make sure that any two consecutive images will have enough overlap, making it possible to extract enough matched features. Third, we should maintain the environment light at an appropriate level. Strong lights will increase the reflection of certain objects, which will decrease the accuracy of feature position. If the lighting is too weak, the intensity of the image will be very low, making it difficult to extract enough features. For these reasons, we did not use directly exposed lamps, but kept the environment bright enough.

7.3 Outdoor Large Area SLAM

For the indoor experiment in the previous section, even though natural features were used, we had set-up the working area with randomly-placed objects and provided adequate illumination. In this section, we present experiments that validate the Alg. 4 in an outdoor environment for 3D SLAM.

7.3.1 System Configuration

A trinocular camera, DIGI-COL-60 Color Digiclops System with 6mm lenses and 10cm baselines in both directions, a GPS receiver, and a digital compass are used for this experiment. The sensors are fixed on a rigid frame which is installed on a garden cart (Fig. 7.11). The garden cart is pushed at unknown speeds, and the sensors installed in the cart take measurements continuously.

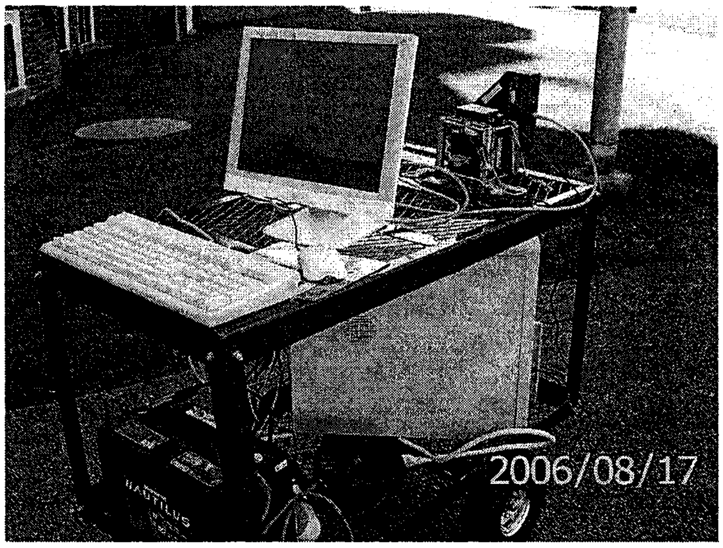


Figure 7.11: System setting for field test.

7.3.2 3D SLAM in Real Application

We did the experiment at a parking lot at Dalhousie University. The working area is 20 meters in length and 15 meters in width. During the robot navigation, 190 images for each camera are taken one second apart, and half of these images are displayed in Fig. 7.12. These images represent many different terrains, such as rocks, grass, paved ground, and curb. Different terrains will generate different numbers of SIFT features. We can see that in the rock area, the number of features from the related image are more than in other areas (Fig. 7.13), such as in the paved area.

The trinocular camera provides not only three images (right,left and top) at each time, but also a data file which includes the 3D points of the view field based on the camera coordinate system (3D cloud). Generally, if an image has more SIFT features, the trinocular system will provide more 3D points (Fig. 7.13). The number of SIFT features and the number of 3D points will influence the robot pose estimation directly. If there are not enough 3D points, the related algorithm may not be able to find enough matched features and 3D points for the robot pose estimation.

The number of matched features between two consecutive images in the image sequence of the experiment is displayed in Fig. 7.14. We knew that there are many

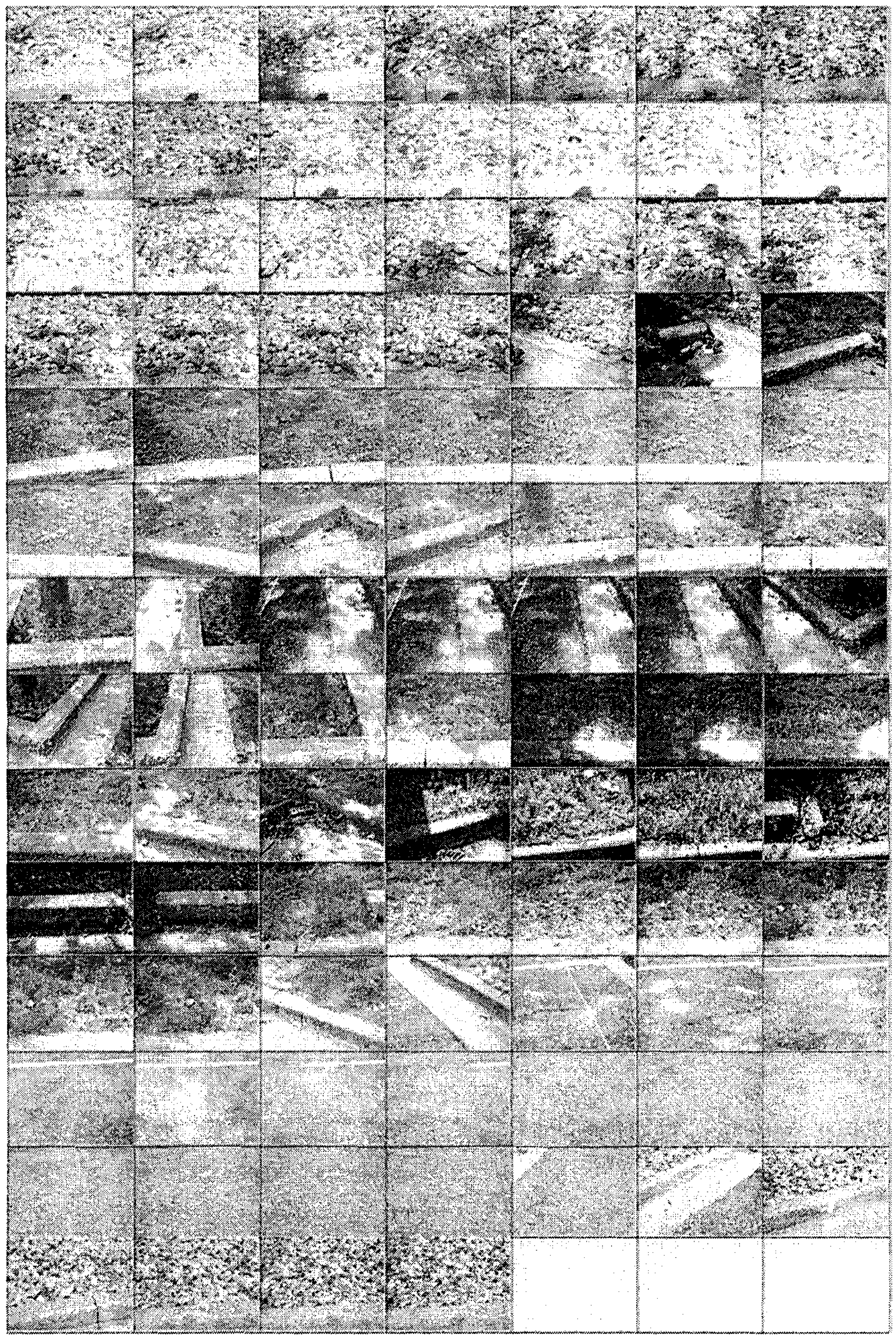


Figure 7.12: Images from field test (in order of 1,3,...,189), from right to left, and top to bottom.

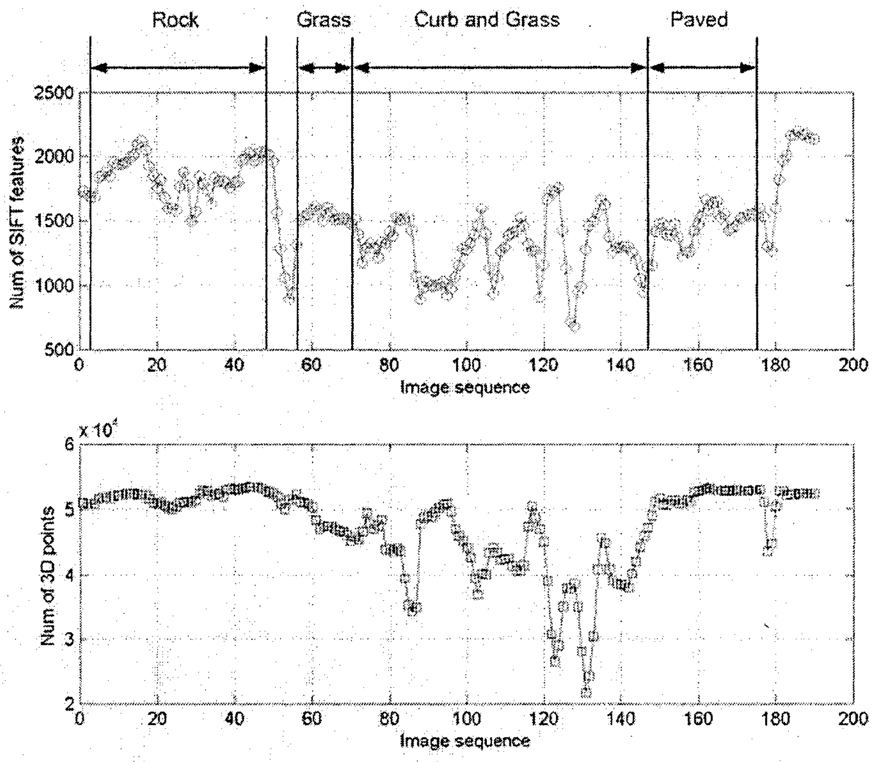


Figure 7.13: Number of features and 3D points from each image.

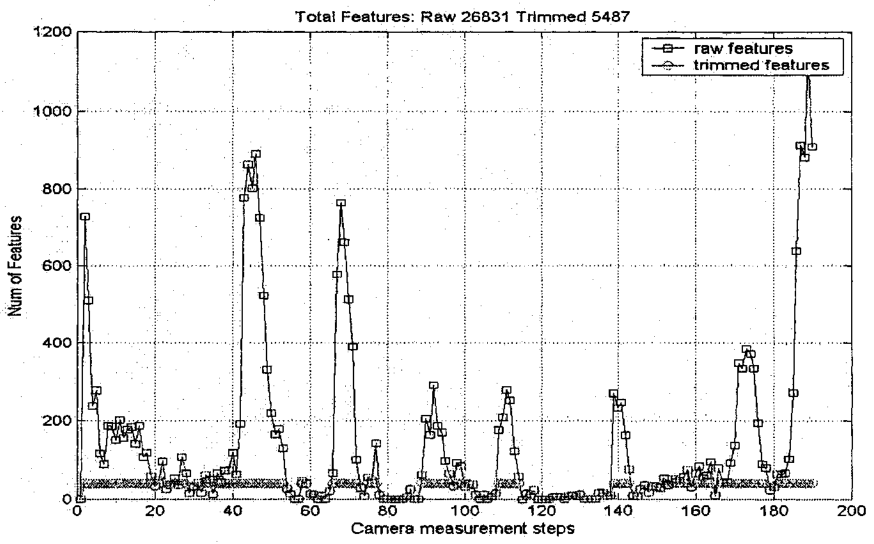


Figure 7.14: Number of matched SIFT features in each image with its previous consecutive image.

issues which will influence the number of matched features. In the outdoor environment, overlap is the most important issue for the number of matched features. There are eight peak points in Fig. 7.14 which correspond to images number 4, 45, 67, 92, 111, 138, 173, 188. Except for the peak point at image number 4, all the peak points correspond with the turning points on the robot path, since the robot moved very slowly at the turning points and this caused the consecutive images to have more overlaps than that of other places.

If there is only a camera sensor, it is impossible to estimate a robot path and establish a map continuously for the whole navigation period, since, in some places, there are not enough matched features for the estimation, such as at steps 60, 80, 105, 145, and 165 in Fig. 7.14. Therefore, we have to solve the local SLAM problem first for the case where there is a sequence of images which has enough matched features in two consecutive images. In our experiment, there were 190 images which generated 24 local maps and robot path segments in Fig. 7.15, Fig. 7.16, Fig. 7.17, and Fig. 7.18. In these figures, the line with circle is the path estimated by camera, and the dots are the detected features for map; x axis in the direction from lower left to upper right represents length in the unit of meter, and y axis in the direction from lower left to upper left represents width, and z axis in vertical line represents the height, all of the axises have unit in meter.

It must be pointed out that, even though there are enough matched features in two consecutive images at certain points, we did not continuously generate that local map since the error of estimate path exceeds a certain value. There are two ways to trigger the algorithm to stop the local SLAM in this case: the uncertainty of the estimated robot position is larger than a certain threshold; or the number of steps in a local map building is more than a certain number. In this implementation, we used the second, where the threshold is such that when the number of steps in a local SLAM reaches 10, the local map is terminated, and a new local map is started.

The GPS, like the buoys for the underwater mobile robot, is used to estimate the robot's position in the global coordinate system. We fused the estimate from the GPS, local maps and robot paths from the stereo camera by using Alg. 4. A globally-consistent 3D map and robot path are obtained, as shown in Fig. 7.19, and the result projected in 2D is shown in Fig. 7.20.

7.3.3 Efficient Map Building in Real Application

In the previous result, we used all the matched features to solve the local SLAM problem, and the final map has 26,831 features. We know that there is no need to have so many features in the map. An efficient map building algorithm (Alg. 7) was used in the Alg. 4, and the final globally-consistent 3D map and robot path with efficient map are displayed in Fig. 7.21. In this case, we selected at most 50 matched features from two consecutive images to solve the local SLAM problem, and the final map has 2281 features. The size of the efficient map is less than 10% of the size of the normal map. This saves much storage space and also speeds up the computation for the related map operation.

An interesting question here is whether the efficient map will affect the estimation results. We used statistical hypothesis testing to check the estimated path position

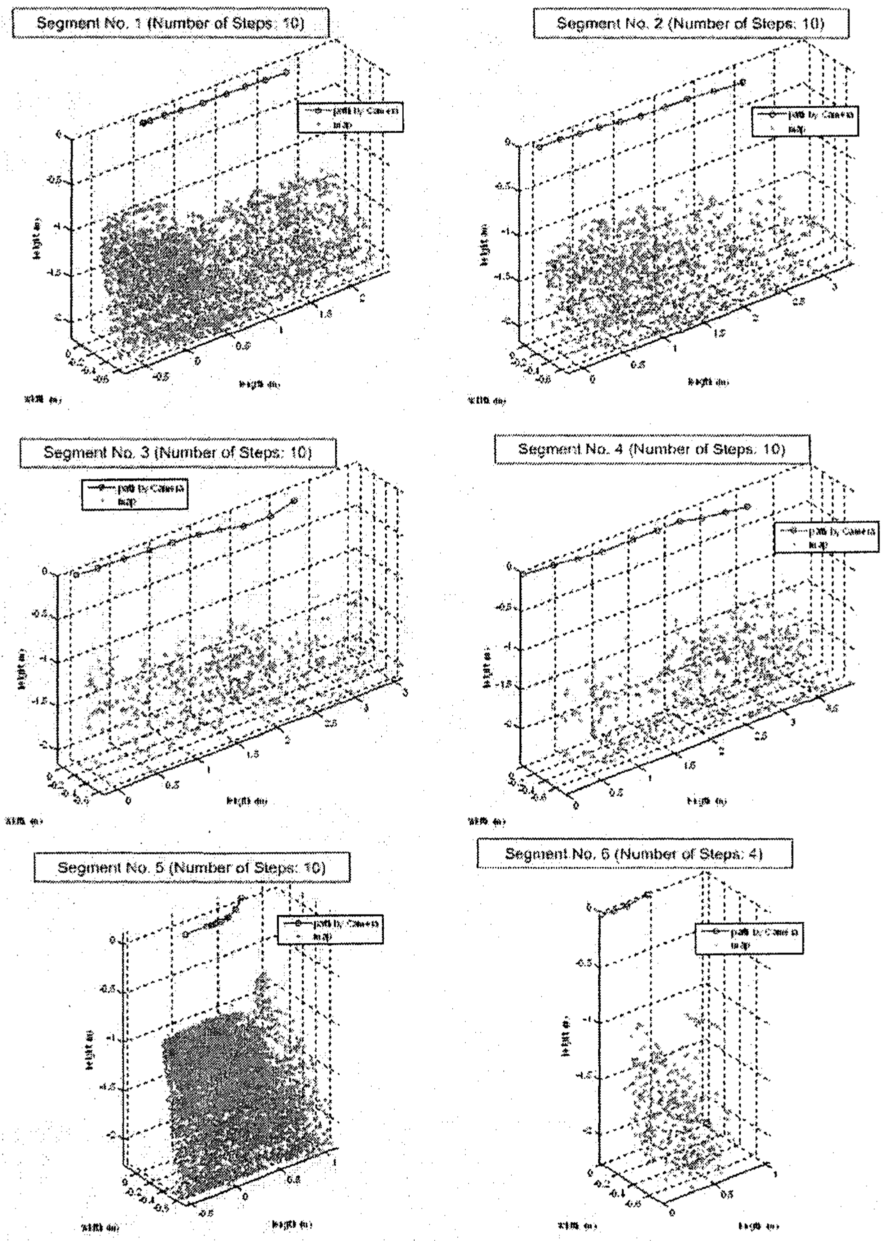


Figure 7.15: Local maps from field test (1,...,6).

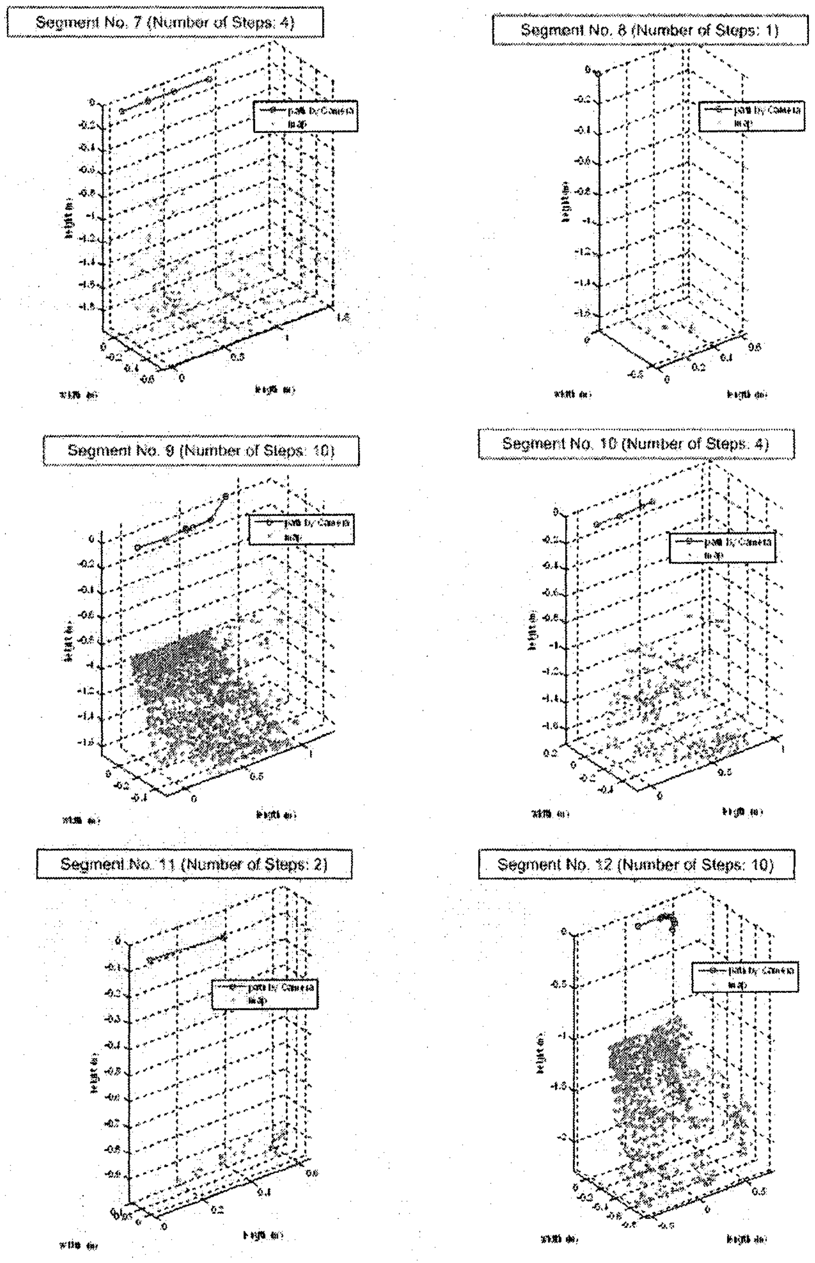


Figure 7.16: Local maps from field test (7,...,12).

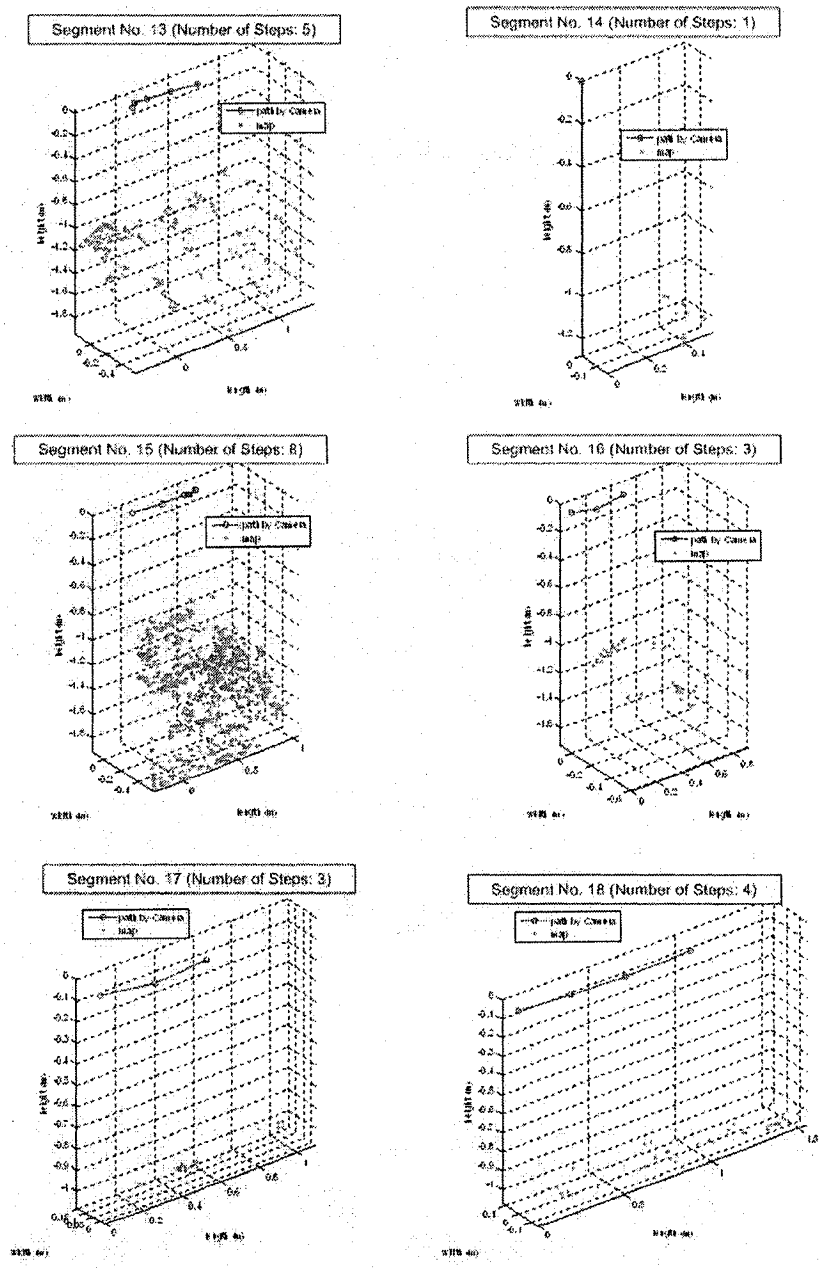


Figure 7.17: Local maps from field test (13,...,18).

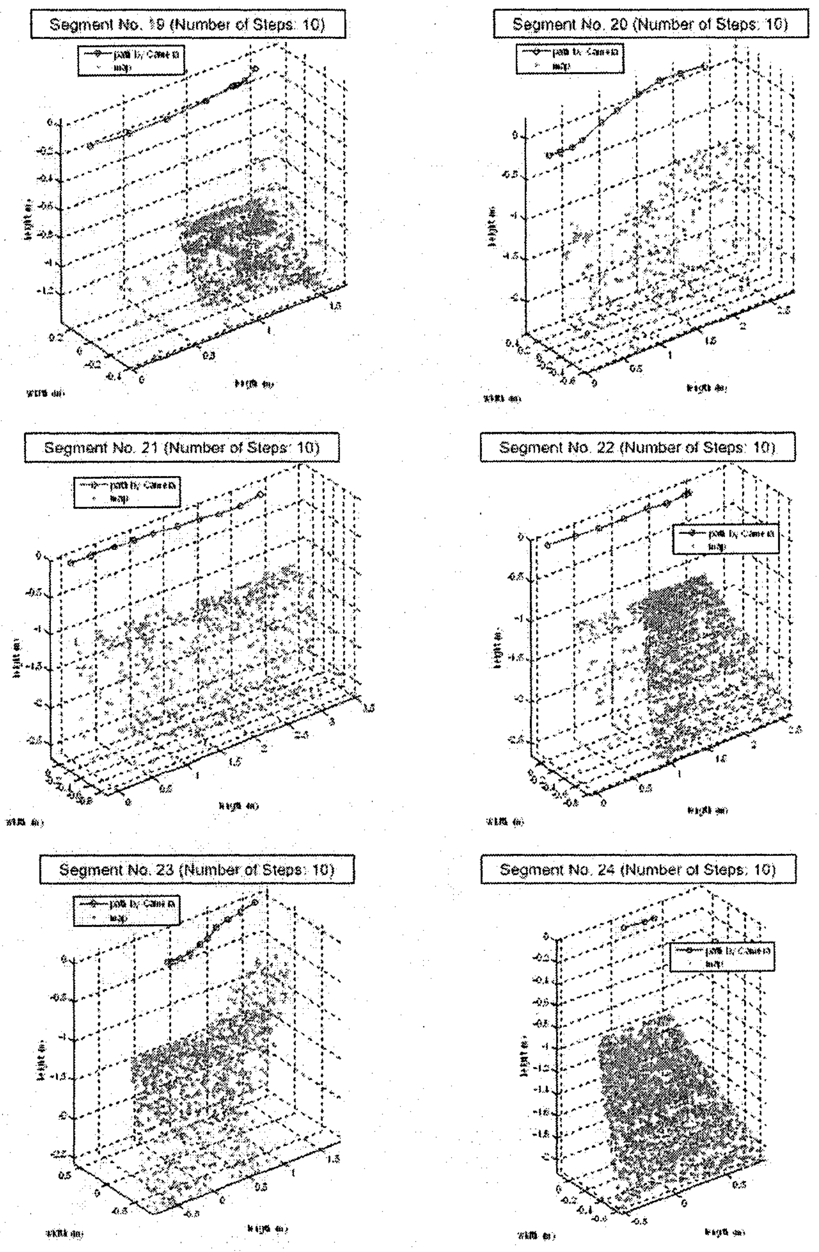


Figure 7.18: Local maps from field test (19,...,24).

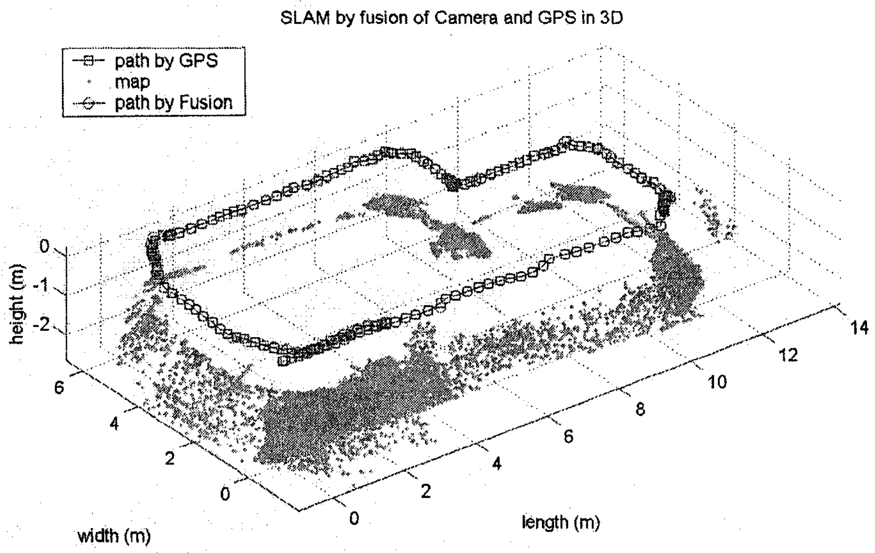


Figure 7.19: Global map and robot path in 3D from field test.

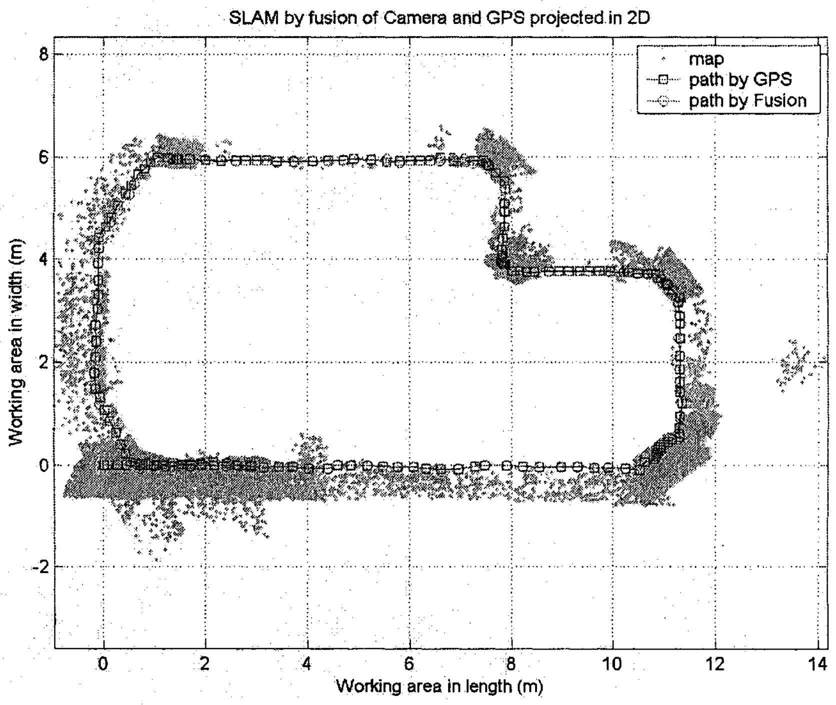


Figure 7.20: Global map and robot path projected to 3D from field test.

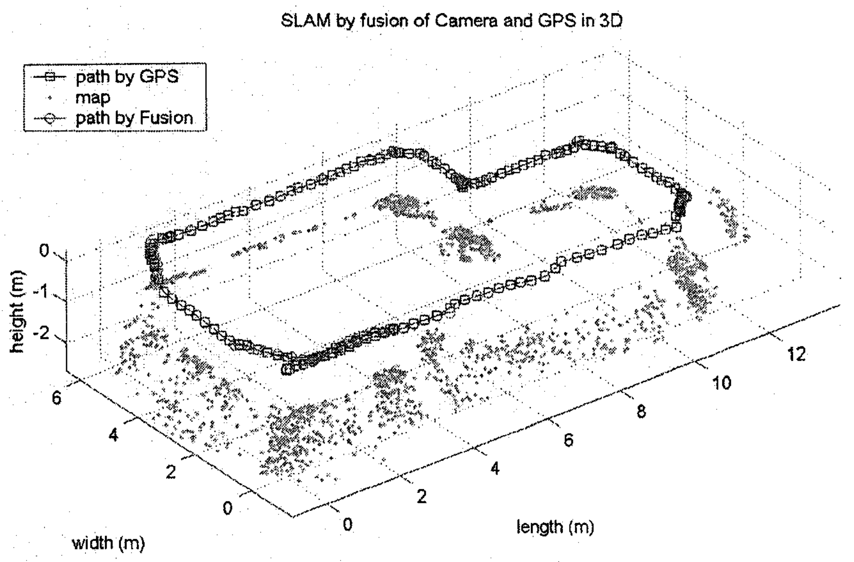


Figure 7.21: **Global map and robot path with efficient map from field test in 3D.**

and features position of the map for both cases: normal map and efficient map. We assumed that there is no significant difference between them, with a confidence level of 95%. The test results indicated that the hypothesis is true and accepted. That means there is no accuracy deterioration for SLAM solution by using the efficient map.

7.3.4 Discussion for SLAM from Outdoor Experiment

The globally-consistent SLAM results from the outdoor experiment showed the success of the algorithms and the measurement system. In order to have a better understanding of the implementation of the related algorithms, some issues need to be clarified.

During the fusion step that integrates both results from local SLAM and the estimate from the buoys in Alg. 4, extra measurements from another sensor are needed if the robot path in the local SLAM is a straight line or an approximate straight line. In

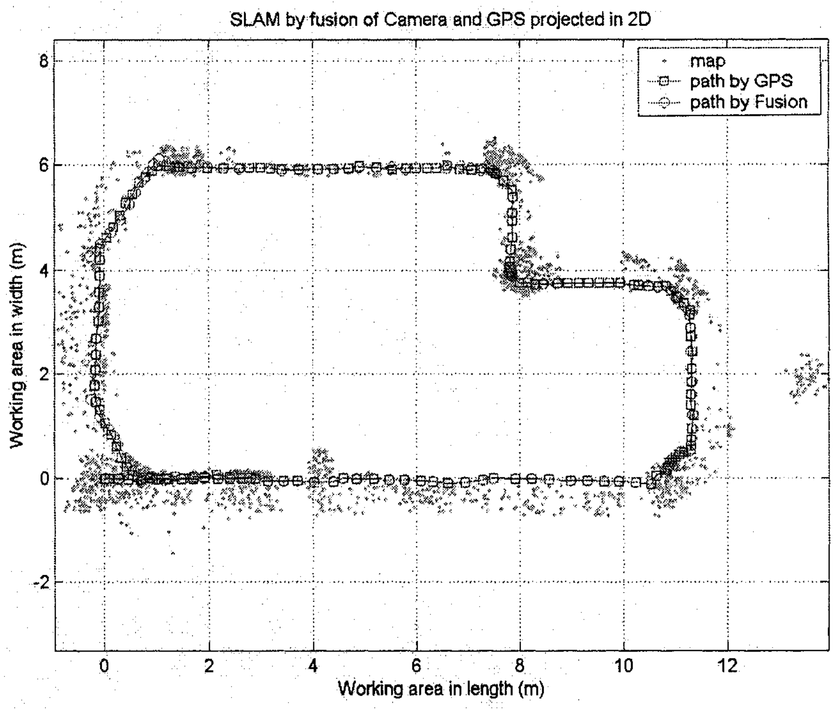


Figure 7.22: Global map and robot path with efficient map from field test in 2D.

our experiment, half of the 24 local SLAM results have a straight line or an approximate straight line in their associated robot paths. In these cases, the measurement from a digital compass (which obtains the heading, pitch, and roll of the attached robot) will be used to determine the map orientation during the fusion step.

In all the 24 local SLAM results from our experiment, there are 6 which did not have 4 position points in their associate paths. If the position points in a robot path are fewer than 4, it is impossible to form a B-spline curve in Alg. 4. If this occurs, we simply ignore the local results. Of course, some important landmarks may be lost in the final map, this can be avoided by using the following method.

If there is only one position point in the associate robot path from a local SLAM result, we can simply ignore it. If there are two or three, we can use a linear interpolation method to construct the robot path curve, and then use this curve to replace the B-spline curve in Alg. 4, which will decrease the likelihood of losing landmarks in the final map.

Another issue is the efficient map building. In the algorithm (Alg. 7) used to build an efficient map, we consider that every landmark has the same opportunity to be selected for the final map. Therefore, we only apply the position information of a landmark, and we do not apply its attribute information such as colour, shape, volume, and dimension. In real application, some landmarks are more important than others. For example, a large tree is more important than a small tree as a landmark in a map. The future task should consider this issue and the efficient algorithm should be modified.

7.4 Summary

A stereo camera is a very important sensor for SLAM applied by a mobile robot. From the experiment in a water environment, we know that it is possible to use a stereo camera to solve a SLAM problem if the environment provides enough matched features in every two consecutive images.

During the processing for local SLAM, in order to avoid too large estimation error (which is accumulated from sensor noise, measurement noise, computational noise, and model noise), we set a threshold such that when the estimation error or steps in the estimation reach certain values, the local SLAM will be stopped, and a new local SLAM will be started. Therefore, there are two reasons for having local SLAM: unsuitable environment and estimation error.

For large area SLAM with SIFT features, another problem is that the established map is bigger than actually needed, which will cause storage and computational burdens. By using the agglomerative hierarchical clustering method, we can select the most appropriate features from a large set of features, then an efficient map can be built. In the results from our field experiment, the size of the efficient map is less than 10% the size of the original map. Statistic testing results indicated that the estimation value for path and map has no difference when using the two kinds of maps.

This strategy and the algorithm have been tested in two ways: indoor lab and outdoor field. The results show the potential of the algorithm in establishing a globally-consistent 3D map and robot path in the environment which only has some features for vision sensor in part of the working area.

Chapter 8

Conclusion

8.1 Conclusion

In this thesis, we have presented a new strategy and algorithm to solve the full 3D SLAM problem based on the sensor fusion mechanism, which can be used on land, in space and in an underwater environment.

The main contribution in this thesis is the development of a set of algorithms for globally-consistent localization and mapping for a mobile robot with multi-sensor fusion. To implement these algorithms, two types of sensors are needed: one which can be used to measure the position of the landmarks in the environment, such as stereo camera, radar, laser range finder and sonar; and another which can be used to measure the absolute position of the mobile robot, such as GPS and buoys. The first sensor can be used to establish local map and robot pose, the second can be used to estimate global robot pose. A sensor fusion algorithm is designed to build a consistent 3D map and estimate consistent 3D robot pose. The contribution of this thesis can be classified into five categories.

(1) ***A system architecture designed for underwater mobile robot localization and mapping***. The SLAM algorithm based on stereo camera developed in an indoor environment is not suitable in an outdoor environment since it is impossible to continuously obtain enough matched features in the image sequence, and the odometry information is not reliable. How to design a system for a mobile robot to obtain globally consistent 3D map and robot path is a significant challenge.

Based on the general SLAM method, we developed a simple structure by using a stereo camera and a set of buoys to solve the 3D SLAM problem in the underwater environment. A complementary fusion algorithm uses the advantages of both the sensors: buoys for long range, and camera for short range.

(2) ***Registration uncertainty for robot self-localization in 3D.*** A stereo camera is a very basic sensor used by a mobile robot. It has been successfully applied to indoor environments to solve the SLAM problem. There are two approaches to solve it: flow-based and correspondence-based. The correspondence-based approach extracts distinct features such as corners, lines, high curvature points, SIFT, etc., through an image sequence, and computes shape and motion by registering these features. The flow-based approach treats the image sequence as a function $f(x, y, t)$, where (x, y) are image pixel coordinates and t is time; this approach computes shape and motion in terms of differential changes in the function f, but restricts the motion between two consecutive frames to be small. In the outdoor and underwater environments, it is very difficult to guarantee that this requirement will be met. Therefore, the correspondence-based approach is selected for this thesis.

Based on the correspondence-based approach, we designed a new method for estimating the uncertainty of the registration parameter. This method uses the implicit function theorem to derive the pose uncertainty from a maximum likelihood formulation. This result can be used to estimate the robot pose and map position's uncertainty. Results from the lab experiment have provided strong support for this method.

(3) ***Efficient map building in a large area.*** When a stereo camera is used for the SLAM problem in an unknown natural environment for a large area, a challenge is the size of the established map. In some areas that have rich features, there are too many features which are needed for the robot pose estimation and which will cause the size of the map to be too big for efficient operation. By using a set theory, a set of new algorithms for an efficient map building is designed, which are based on the level of overlapping, stratified sampling, and select centre points from a set of clusters by the agglomerative hierarchical clustering method.

(4) ***Globally-consistent 3D SLAM by sensor fusion.*** In a natural environment, it is impossible to establish a global map with the measurements from a

stereo camera only, because there may not be enough features in some areas and the odometry information is very poor. Therefore, the measurements from the stereo camera can only be used to solve the SLAM problem based on a local coordinate system where enough matched features can be obtained from two consecutive images in an image sequence, while the measurement from buoys or GPS will be used to estimate a global path of the robot. The difficult challenge here is how to establish a globally-consistent 3D map and estimate a globally-consistent 3D robot pose.

By using time information for each measurement, it is possible to establish correspondence from the local robot path (by camera) to the global robot path (by buoys) which is expressed by a 3D B-spline function. The corresponding information can be applied to estimate the 3D transformation from each local frame to a global frame. Finally, the globally-consistent 3D map will be constructed. Results from indoor and outdoor experiments demonstrated that the new fusing algorithm is convenient and successful.

(5) ***A mobile robot simulation system designed for full 3D SLAM application.*** For an unknown area, it is impossible to accomplish experiments with a real mobile robot. We developed a mobile robot simulation system with two types of sensors installed on it: the relative measurement sensor (stereo camera, laser canner, radar, and sonar) and the absolute position measurement sensor (GPS and buoys). The robot can navigate according to a given path in a working environment which is filled with designed landmarks, and the environment can be in space, water, or on land. The system is developed with a friendly interface and real time 3D animation ability. The measurements by the sensors will be saved to a series of files that shall be applied to validate the globally-consistent 3D SLAM algorithms.

8.2 Future Work

The strategy and algorithms developed in this thesis have successfully solved the globally-consistent 3D SLAM problem for the mobile robot, especially for the underwater mobile robot; however, there are still many tasks needed to be done for a complete system.

(1) ***Real-time local SLAM with stereo camera.*** The SLAM algorithm based on the stereo camera is unable to be performed in real time; the bottleneck is the data association. For two consecutive images, the matched SIFT features are obtained by the shortest distance among all the pair features, but many spurious matched features will also be generated. The RANSAC method was applied to delete the outliers in the matched features. This step is very time consuming in some cases. We need to use epipolar geometry information and odometry information to detect the outliers.

(2) ***Underwater experiment to validate the algorithms.*** We did some experiments in water and designed equipment for the project, but we still could not do the experiment for the 3D SLAM. There are many challenges for this task: First, experiments in an open ocean area require the equipment to be reliable and suitable. Our equipment can only be used in a swimming pool, but there are no natural landmarks in the swimming pool. Setting landmarks in the pool at Dalhousie is not allowed. Second, experiments in water are very expensive: requiring a lot of equipment, much manpower, and a lot of time. The best way to complete this task is to cooperate with potential users in the future.

(3) ***New feature development for underwater environment.*** In this thesis, we used SIFT features for the SLAM. Even though this feature is widely used successfully and is reliable in most cases, it is also difficult to find a corresponding point by human means, which means that the features can only be understood by robot.

(4) ***Enhancement for simulation system.*** The models in our 3D mobile robot system are wire-frame, and satisfactory for all the functions in this system. If good visualization is required, advanced tools, such as OpenGL, are necessary to render the robot and the 3D environment. Another type of enhancement is to integrate SLAM algorithms with this system. This software can be commercialized when these enhancements are completed.

Bibliography

[1] World Geodetic System 1984 (WGS-84) its definition and relationships with local geodetic systems. Technical report, Defense Mapping Agency, Fairfax, VA, 1985.

[2] A. Adam, E. Rivlin, and I. Shimshoni. Computing the sensory uncertainty field of a vision-based localization sensor. *IEEE Transactions on Robotics and Automation*, 17:258–267, 2001.

[3] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:384–401, 1985.

[4] N. Ayache and O. Faugeras. Maitaining a representation of the environment of a mobile robot. *IEEE Transaction on Robotics and Automation*, 5, 1989.

[5] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association.* Academic Press, 1988.

[6] S. Betge-Brezetz, P. Hebert, R. Chatila, and M. Devy. Uncertain map making in natural environments. In *In Proc. IEEE Int. Conf. Robotics and Automation*, pages 1048–1053, April 1996.

[7] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *International Conference on Robotics and Automation*, pages 1899–1906, Sep. 2003.

[8] M. J. Brooks, W. Chjnacki, D. Gawley, and A. Hengel. What value covariance information in estimation vision parameters? In *IEEE Proceedings of the ICCV*, 2001.

[9] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture.* John Wiley and Sons, 1996.

[10] J. A. Castellanos and J. D. Tardos. *Mobile robot localization and map building: A multisensor fusion approach.* Kluwer Academic Publishers, 1999.

[11] R. Chatila and J. P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 138–145, 1985.

[12] D. W. Cho. Certainty grid representation for robot navigation by a bayesian method. *Robotics*, 8:159–165, 1990.

[13] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): Towards exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17:125–137, 2001.

[14] A. R. Chowdhury and R. Chellappa. Statistical error propagation in 3D modeling from monocular video. *IEEE Workshop on Statistical Analysis in Computer Vision*, 2003.

[15] I. J. Cox. Blanche: Position estimation for an autonomous robot vehicle. In *Proceedings of the International Workshop on Intelligent Robots and Systems, IEEE/RSJ*, pages 432–439, Japan, 1989.

[16] G. Csurka, C. Zeller, Z. Zhang, and O. Faugeras. Characterizing the uncertainty of the fundamental matrix. In *Techinical report*, INRIA, June 1995.

[17] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV2003*, 2003.

[18] F. Dellaert, D. Foxt, W. Burgard, and S. Thrunt. Monte Carlo Localization for Mobile Robots. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, pages 1322–1328, Michigan, May 1999.

[19] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[20] J. Dominik, S. Cyrill, and B. Wolfram. Autonomous exploration for 3D map learning. *Fachgesprche Autonome Mobile Systeme (AMS)*, 2007.

[21] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[22] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12:287–300, 2002.

[23] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics.* Cambridge University Press, Cambridge CB2 2RU.

[24] H. F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Transaction on Robotics and Automation*, 4:23–31, 1988.

[25] C. Estrada, J. Neira, and J. D. Tardos. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics and Automation*, 2005.

[26] E. Fabrixi and A. Safflotti. Extracting topology-based maps from gridmaps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2973–2978, San Francisco, CA, USA, 2000.

[27] J. A. Farrell and M. Barth. *The global positioning system and inertial navigation.* McGraw-Hill, 1998.

[28] O. D. Faugeras. *Three-Dimensional Computer Vision: A Geometric viewpoint.* MIT Press, 1993.

[29] C. Fennema, A. Hanson, E. Riseman, J. R. Beveridge, and R. Kumar. Model-directed mobile robot navigation. *IEEE Transaction on Systems, Man, and Cybernetics*, 20(6):1352–1369, 1990.

[30] J. Fessler. Mean and variance of implicitly defined biased estimators (such as penealized maximum likelihood): Application to tomography. *IEEE Transaction on Image Processing*, 5:493–506, 1996.

[31] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24, 1981.

[32] D. Fox, S. Thrun, F. Dellaert, and W. Burgard. Particle filters for mobile robot localization. In *A. Doucet, N. de Freitas, and N. Gordon, editors, Sequential Monte Carlo Methods in Practice*, Spinger Verlag, New York, 2000.

[33] M. A. Garcia and A. Solanas. 3D simultaneous localization and modeling from stereo vision. In *Proceedings of the 2004 IEEE International Conf. on Robotics and Automation*, pages 847–853, New Orleans, LA, April 2004.

[34] N. J. Gordon. A hybrid bootstrap filter for target tracking in clutter. *IEEE Transaxtions on Aerospace and Electronic Systems*, 33:353–358, 1997.

[35] J. S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 318–325, Nov. 1999.

[36] J. Han and M. Kamber. *Data Mining - Concepts and Techniques.* Morgan Kaufmann Publishers, 2006.

[37] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, The Edinburgh Building, Cambridge CB2 2RU, UK, 2000.

[38] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of Optical Society of America*, 4:629–642, 1987.

[39] C. Hue, J. L. Cadre, and P. Perez. Tracking multiple objects with particle filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 38:791–812, 2002.

[40] A. Huster. *Relative position sensing by fusion monocular vision and inertial rate sensors.* Ph.D. thesis, Stanford University, 2003.

[41] M. Isard and A. Blake. Condensation-conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29:5–28, 1998.

[42] O. Jokinen and H. Haggren. Statistical analysis of two 3-d registration and modeling strategies. *Journal of Photogrammetry and Remote Sensing*, 53:320–341, 1998.

[43] S. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear system. In *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Rescource Management II, SPIE*, 1997.

[44] A. Kelly. Mobile robot localization from large scale appearance mosaics. *International Journal of Robotics Research*, 19(11), 2000.

[45] S. Koenig and R. Simmons. Passive distance learning for robot navigation. In *Proceedings of the Thirteenth International Conference on Mach ine Learning*, pages 266–274, San Mateo, 1996.

[46] M. B. Larsen. High performance doppler-inertial navigation - experimental result s. *IEEE Oceans*, 2000.

[47] J. J. Leonard, H. F. Durran-Whyte, and I. J. Cox. Dynamic map building for an autonomous mobile robot. *The International Journal of Robotics Research*, 11(4):286–298, 1992.

[48] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 1442–1447, May 1991.

[49] B. Lisien, D. Morales, D. Silver, G. Kantor, I. Rekleitis, and H. Choset. Hierarchical Simultaneous Localization and Mapping. In *Proceedings of the International Conference on Robotics and Automation*, 2004.

[50] H. Liu and E. Milios. Acoustic positioning using multiple microphone arrays. *Journal of the Acoustical Society of America*, 117(5):2772–2782, May 2005.

[51] D. G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8:113–122, 1992.

[52] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[53] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

[54] Y. Ma, J. Kosecka, and S. Sastry. Linear differential algorithm for motion recovery: A geometric approach. *International Journal of Computer Vision*, 36:343–350, 2000.

[55] H. Madjidi and S. Negahdaripour. Global alignment of sensor positions with noisy motion measurements. In *IEEE International Conference on Robotics and Automations*, New Orleans, April 2004.

[56] S. Majumader. *Sensor Fusion and Feature Based Navigation for Subsea Robots.* Ph.D. thesis, The University of Sydney, Australian Centre for Field Robotics, School of Aerospace, Mechanical and Mechatronic Engineering, 2001.

[57] S. Majumder. *Sensor Fusion and Feature Based Navigation for Subsea Robots.* Ph.D. thesis, The University of Sydney, Australian Centre for Field Robotics, School of Aerospace, Me chanical and Mechatronic Engineering, 2001.

[58] P. S. Mann. *Introductory Statistics.* John Wiley and Sons, INC, 1998.

[59] L. Matthies and S. A. Shafer. Error modeling in stereo navigation. *IEEE Journal of Robotics and Automotion*, RA-3:239–248, 1987.

[60] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fast SLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002.

[61] H. Moravec. Robot spatial perception by stereoscopic vision and 3D evidence grids. Technical Report CMU-RI-TR-96-34, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 1996.

[62] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 116–121, May 1985.

[63] H. P. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover.* PhD thesis, Stanford University, CA, Sep. 1980.

[64] K. Murphy. Bayesian map learning in dynamic environments. In *Neural Information Processing Systems (NIPS)*, 1999.

[65] P. M. Newman, J. J. Leonard, and R. R. Rikoski. Towards constant-time SLAM on an autonomous underwater vehicle using synthetic aperture sonar. In *11th International Symposium on Robotics Research*, Sienna, Italy, 2003.

[66] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone. Stereo egomotion improvements for robust rover navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, Korea, May 2001.

[67] M. Park. *Error analysis and stochastic modeling of MEMS based inertial sensors for land vehicle navigation applications.* Master of science, The University of Calgary, Calgary, Alberta, 2004.

[68] L. A. Piegl and W. Tiller. *The Nurbs Book.* Springer, 1997.

[69] I. Rekleitis. *Cooperative Localization and Multi-Robot Exploration.* Ph.D. thesis, McGill University, Montreal, Quebec, Canada, 2003.

[70] S. I. Roumeliotis and G. A. Bekey. Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization. In *Proc. 2000 IEEE International Conference on Robotics and Automation*, pages 22–28, San Francisco, California, April 2000.

[71] J. Ryde and H. Hu. Cooperative mutual 3D laser mapping and localization. In *IEEE International Conference on Robotics and Biomimetics Kunming*, China, Dec 2006.

[72] J. Saez and F. Escolano. A global 3D map-building approach using stereo vision. In *IEEE International Conference on Robotics and Automation*, New Orleans, April 2004.

[73] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *Int. Journal of Robotics Research*, 21:735–758, 2003.

[74] S. Se, D. Lowe, and J. Little. Vision-based global localization and mapping for mobile robots. *IEEE TRANSACTIONS ON ROBOTICS*, 21(3):364–375, Jun 2005.

[75] G. C. Sharp, S. W. Lee, and D. K. Wehe. Toward multiview registration in frame space. In *IEEE Internatinal Conference on Robotic and Automation*, pages 3542–3547, Seoul, Korea, May 2001.

[76] H. Shatkay and L. Kaelbling. Learning topoligical maps with weak local odometric information. In *Proceedings of the Fifteenth Int. Joint Conf. on Artificial Intelligence*, pages 920–927, San Mateo, CA:Morgan Kaufmann, 1997.

[77] R. Sim and G. Dudek. Self-organizing visual maps. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Jose, CA, 2004.

[78] R. Sim, P. Elinas, M. Griffin, and J. J. Little. Vision-based SLAM using the Rao-Blackwellised particle filter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[79] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *Int. Journal Robotics Research*, 5:56–68, 1986.

[80] M. Sonka, V. Hlavac, and R. boyle. *Image Processing, Analysis, and Machine Vision.* PWS Publishing, 1999.

[81] Z. Sun, V. Ramesh, and A. M. Tekalp. Error characterization of the factorization method. *Computer Vision and Image Understanding*, 82:110–137, 2001.

[82] J. D. Tardos, J. Neira, P. M. Newman, and J. J. Leonard. Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research*, 2002.

[83] S. Thrun and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robot. *Machine Learning*, 31:29–53, 1998.

[84] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128:99–141, 2001.

[85] G. Welch and G. Bishop. An introduction to the Kalman filter, SIGGRAPH 2001 course 8. In *Computer Graphics, Annual Conference on Computer Graphics & Interactive Techniques*, Los Angeles, CA, Aug. 2001.

[86] M. B. Whitcomb, D. R. Yoerger, and H. Singh. Combined Doppler/LBL based navigation of underwater vehicles. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, 1999.

[87] S. B. Williams, P. Newman, G. Dissanayake, and H. Durrant-Whyte. Autonomous underwater simultaneous localisation and map building. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 1793–1798, San Francisco, CA, April 2000.

[88] G. J. Young and R. Chellappa. Statistical analysis of inherent ambiguities in recovering 3-D motion from a noisy flow field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:995–1013, 1992.

[89] P. Zhang, E. E. Milios, and J. Gu. Graph-based automatic consistent image mosaicking. In *IEEE International Conference on Robotics and Biomimetics*, page paper no. 332, Shenyang, China, Aug. 22-26 2004. http://www2.acae.cuhk.edu.hk/~robio/.

[90] Pifu Zhang, Jason Gu, and Evangelos Milios. Registration uncertainty for robot self-localization in 3D. In *Second Canadian Conference on Computer and Robot Vision, CRV 2005*, pages 490 – 497, Victoria, BC, May 9-11 2005.

[91] Pifu Zhang, Evangelos E. Milios, and Jason Gu. Graph-based automatic consistent image mosaicking. In *IEEE International Conference on Robotics and Biomimetics*, pages 558 – 563, Shenyang, China, Aug. 22-26 2004.

[92] Pifu Zhang, Evangelos E. Milios, and Jason Gu. Vision data registration for robot self-localization in 3D. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2315 – 2320, Edmonton, Alberta, Canada, August 2005.

Appendix A

Euler Angle

A.1 Euler Angle and Rigid Body Transformation

We wish to define a rotation matrix by using euler angles ψ, θ, and ϕ that correspond to rotations of the principal axes x, y, and z respectively. Each principal axis rotation forms a 3x3 matrix where the direction of rotation for a positive angle is defined by the right-hand rule of the axis of rotation, as follows

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & -sin(\psi) \\ 0 & sin(\psi) & cos(\psi) \end{pmatrix} \tag{A.1.0.1}$$

$$R_y(\theta) = \begin{pmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{pmatrix} \tag{A.1.0.2}$$

$$R_z(\phi) = \begin{pmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.1.0.3}$$

Therefore a positive ψ, θ, and ϕ correspond to rotations of the principal axes from Y-to-Z, Z-to-X, and X-to-Y respectively, corresponding to the right-hand rule. We form the composite rotation matrix R by applying the rotation in order of x, y, and then z (using post-multiplication/column vectors):

$$R = R_z * R_y * R_x \tag{A.1.0.4}$$

where

$$\begin{aligned}
r_{00} &= cos(\phi) * cos(\theta) \\
r_{01} &= -sin(\phi) * cos(\psi) + cos(\phi) * sin(\theta) * sin(\psi) \\
r_{02} &= sin(\phi) * sin(\psi) + cos(\phi) * sin(\theta) * cos(\psi) \\
r_{10} &= sin(\phi) * cos(\theta) \\
r_{11} &= cos(\phi) * cos(\psi) + sin(\phi) * sin(\theta) * sin(\psi) \\
r_{12} &= -cos(\phi) * sin(\psi) + sin(\phi) * sin(\theta) * cos(\psi) \\
r_{20} &= -sin(\theta) \\
r_{21} &= cos(\theta) * sin(\psi) \\
r_{22} &= cos(\theta) * cos(\psi)
\end{aligned} \tag{A.1.0.5}$$

Therefore a point P_1 defined in a cartesian coordinate system rotates ψ around x-axis, θ-axis, and z-axis, respectively, in that order. The new point P_2 in the same coordinate system is

$$P_2 = RP_1 \tag{A.1.0.6}$$

Another definition of rotation is the case where point P will not rotate, but its coordinate system will rotate from $xyzo$ to $x'y'z'o'$. The problem is calculating the position of P in the new coordinate system.

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & sin(\psi) \\ 0 & -sin(\psi) & cos(\psi) \end{pmatrix} \tag{A.1.0.7}$$

$$R_y(\theta) = \begin{pmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{pmatrix} \tag{A.1.0.8}$$

$$R_z(\phi) = \begin{pmatrix} cos(\phi) & sin(\phi) & 0 \\ -sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.1.0.9}$$

$$\begin{aligned}
r_{00} &= cos(\phi) * cos(\theta) \\
r_{01} &= sin(\phi) * cos(\psi) + cos(\phi) * sin(\theta) * sin(\psi) \\
r_{02} &= sin(\phi) * sin(\psi) - cos(\phi) * sin(\theta) * cos(\psi) \\
r_{10} &= -sin(\phi) * cos(\theta) \\
r_{11} &= cos(\phi) * cos(\psi) - sin(\phi) * sin(\theta) * sin(\psi) \\
r_{12} &= cos(\phi) * sin(\psi) + sin(\phi) * sin(\theta) * cos(\psi) \\
r_{20} &= sin(\theta) \\
r_{21} &= -cos(\theta) * sin(\psi) \\
r_{22} &= cos(\theta) * cos(\psi)
\end{aligned} \tag{A.1.0.10}$$

Therefor a point P at coordinate system $xyzo$ will have a coordinate P' at coordinate system $x'y'z'o'$ after the rotation matrix R

$$P' = RP \tag{A.1.0.11}$$

It is very important to distinguish the difference between the two rotation definitions; otherwise, the operations in $3D$ will be confusing.

A.2 Euler Angles from Matrix

Extracting the euler angles from a given matrix is neither unique nor easy. In our simulation, a basic assumption is that there is no roll during the robot's navigation. In this case, $\psi = 0$ is always correct as an input parameter. Because of errors in input, a small value for ψ is possible. The expected euler angles in our simulation system are

$$\begin{aligned}
\psi &= (-5, +5) \\
\theta &= [0, 360) \\
\phi &= (-45, +45)
\end{aligned} \tag{A.2.0.12}$$

From the previous rotation definition, if a matrix is known, its corresponding euler angle can be obtained by

$$\theta = \begin{cases} asin(-r_{20}) & \text{if } r_{00} \geq 0 \\ \pi - asin(-r_{20}) & \text{if } r_{00} < 0 \end{cases} \tag{A.2.0.13}$$

and if $|cos(\theta)| \geq Tol$, then

$$\psi = atan(r_{21}/r_{22}) \tag{A.2.0.14}$$

$$\phi = atan(r_{10}/r_{00}) \tag{A.2.0.15}$$

and if $|cos(\theta)| < Tol$, this is the gimbal lock case

$$\psi = 0 \tag{A.2.0.16}$$

$$\phi = atan(r_{12}/r_{11}) \tag{A.2.0.17}$$

where Tol is a small value which is used to avoid machine error and computation error; usually it is 1×10^{-6}.

A.3 Relationship between Rotation Matrix and Quaternion

Quaternion q consists of four components (q_0, q_1, q_2, q_3). It has the following relationship with the rotation matrix [38]

$$R = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix} \tag{A.3.0.18}$$

where

$$\begin{aligned}
r_{00} &= q_0^2 + q_1^2 - q_2^2 - q_3^2 \\
r_{01} &= 2(q_1q_2 - q_0q_3) \\
r_{02} &= 2(q_1q_3 + q_0q_2) \\
r_{10} &= 2(q_1q_2 + q_0q_3) \\
r_{11} &= q_0^2 - q_1^2 + q_2^2 - q_3^2 \\
r_{12} &= 2(q_2q_3 - q_0q_1) \\
r_{20} &= 2(q_3q_1 - q_0q_2) \\
r_{21} &= 2(q_3q_2 + q_0q_1) \\
r_{22} &= q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{aligned} \tag{A.3.0.19}$$

The derivative of matrix R with respect to each quaternion's components are as

$$\frac{\partial R}{\partial q_0} = \begin{pmatrix} 2q_0 & -2q_3 & 2q_2 \\ 2q_3 & 2q_0 & -2q_1 \\ 2q_2 & 2q_1 & 2q_0 \end{pmatrix} \tag{A.3.0.20}$$

$$\frac{\partial R}{\partial q_1} = \begin{pmatrix} 2q_1 & 2q_2 & 2q_3 \\ 2q_2 & -2q_1 & -2q_0 \\ 2q_3 & 2q_0 & -2q_1 \end{pmatrix} \tag{A.3.0.21}$$

$$\frac{\partial R}{\partial q_2} = \begin{pmatrix} -2q_2 & 2q_1 & 2q_0 \\ 2q_1 & 2q_2 & 2q_3 \\ -2q_0 & 2q_3 & -2q_2 \end{pmatrix} \tag{A.3.0.22}$$

$$\frac{\partial R}{\partial q_3} = \begin{pmatrix} -2q_3 & -2q_0 & 2q_1 \\ 2q_0 & -2q_3 & 2q_2 \\ 2q_1 & 2q_2 & 2q_3 \end{pmatrix} \tag{A.3.0.23}$$

Appendix B

GPS Measurement Decoding

The GPS receiver outputs information in a variety of NMEA 0183 sentences containing different types of information. The "sentences" used by this GPS Receiver are: GGA (Global Positioning System Fix Data), GSA (GPS DOP and Active Satellites), GSV (GPS SVs (Satellite Vehicles) in View), RMC (Recommended Minimum Specific GPS/TRANSIT Data), and VTG (Track Made Good and Groundspeed). Since we are concerned with the positioning information of an object, we can use the GGA sentence to obtain all the necessary data, such as longitude, latitude, and height, in geodetic coordinate system. Following is an example of a GGA "sentence" received by the GPS receiver:

$GPGGA, 171935.214, 4438.0472, N, 06335.5476, W, 1, 08, 1.0, 8.2, M, -23.0, M, 0.0,$
$0000 * 49$

The explanation of this "sentence" is listed in the following table.

Table B.1: Explanation of the GGA Sentence from the GPS

Name	**Data**	**Description**
Identifier	GPGGA	Global Positioning System Fix Data
Time	171935.214	UTC time of 17:19:35.214
Latitude	4438.0472	$43^0 38.0472'$
Direction	N	N(north or S(south)
Longitude	06335.5476	$63^0 35.5476'$
Direction	W	W(west) or E(east)
Fix Quality	1	$0 - Invalid, 1 - GPSfix, 2 - DGPfix$
Number of Satellites	08	8 satellites are observed
HDOP	1.0	Horizontal dilution of precision
Altitude	8.2	8.2 meters above mean sea level
Mean	M	mean sea level
Meters	-23.0	Height of geoid on WGS84 ellipsoid
Mean	M	mean value
Time update	0.0	time since last DGPS update, 0.0 - no update
DGPS Ref. Station ID	0000	No DGPS stations
Checksum	49	Used by program to check for transmission errors

Appendix C

Simulation System Design

C.1 Environment Design for Simulation

Environment settings are a very important initial step. On the interface of Fig. 6.2, all the required input parameters have default values. For different simulation cases, the user can adjust the default value according to his/her requirements. On the file menu, three files needs to be opened before the simulation starts. Among all settings, the following are some of the most important.

C.1.1 Selection Group

There are five choices in the selection group on the interface of Fig. 6.2. The first three only have true or false selections. If only for animation purposes, there is no need to save the sensor observation to a database or files; therefore, the selection in this case should be false.

For sensor directions, there are three choices: front, bottom, and upper, for camera, laser range finder, sonar, and radar. For different applications, appropriate sensor directions need to be selected. In Fig. 6.4, an underwater environment the equipped camera views the bottom and measures the detected landmarks.

C.1.2 Landmark Generation

Landmarks are $3D$ points which can be generated for different cases, such as in space, on a surface, or on a wall. In space, given the number of landmarks, the system will randomly generate all the landmarks in the working volume with defined distribution.

If the landmarks need to be generated on the floor surface, a corresponding surface needs to be loaded first (the surface is expressed by B-spline model). The number of landmarks should be given initially, then they will be generated randomly on the parameter domain of the surface with defined distribution and mapped to the 3D surface. If the generated landmarks are satisfactory, they can be saved to a database or file.

C.1.3 Bottom Surface Generation

Floors generally are of standard shape and size, the simplest being a flat plane, as is the case for most of the indoor environments. In outdoor environments, however, floors vary, for example, the floor of a lake or an ocean. To express any floor shape or size, the B-spline (bi-cubic spline) surface is applied to construct a floor for the robot simulation system.

The B-spline surface is constructed from a set of control points and a knot vector, which decide the shape and continuity of the surface. The advantages of B-spline are that they are: (1) easy to construct on composite surfaces; (2) easy for shape control; (3) efficient at computation.

A parametric tensor product B-spline surface S of degree p in u direction and degrees q in v direction is defined as

$$S(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} d_{i,j} N_{i,p}(u) N_{j,q}(v) \quad (u,v) \in [0,1] \times [0,1] \tag{C.1.3.1}$$

where $d_{i,j}$ are 3D control points, $N_{i,p}(u)$ and $N_{j,q}(v)$ are B-spline basis functions of degrees p and q, respectively. $N_{i,p}(u)$ and $N_{j,q}(v)$ are defined with respect to the knot vectors $\tau = \{\tau_0, \tau_1, \ \dots \ , \tau_{m+p+1}\}$, and $\sigma = \{\sigma_0, \sigma_1, \ \dots \ , \sigma_{n+q+1}\}$, respectively, with $\tau_0 = \dots = \tau_p = \sigma_0 = \dots = \sigma_q = 0$ and $\tau_{m+1} = \dots = \tau_{m+p+1} = \sigma_{n+1} = \dots = \sigma_{n+q+1} = 1$.

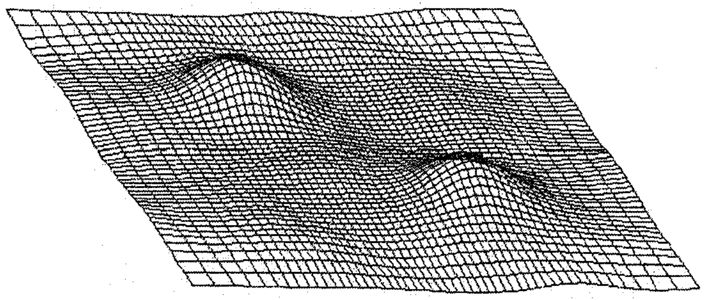


Figure C.1: A floor surface generated by using B-spline function

A B-spline base function $N_{i,p}(u)$ of degree p can be calculated by

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{any others} \end{cases} \tag{C.1.3.2}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u} N_{i+1,p-1}(u) \tag{C.1.3.3}$$

$$define\ \frac{0}{0} = 0$$

Given 10×10 control points in $3D$, with evenly distributed knot, except the end knots as defined in Eq. (C.1.3.1), and both directions u and v having degree 3. A surface generated by the B-spline function is shown in Fig. C.1. By changing the control points and knots, it is possible to generate any shape floor.

In this system, the user needs to input the number of control points in two directions, and the largest value of the height of the surface; then a random surface will be generated and be saved to the database or file. If the user wants to have special landmarks on the surface, he/she needs to change the control point to achieve the desired results.

C.2 Camera View Field Design

The robot in any place of the working area is expressed by position (x, y, z) and the speed (v_x, v_y, v_z). At this point, the view field of the sensor should be decided in 3D

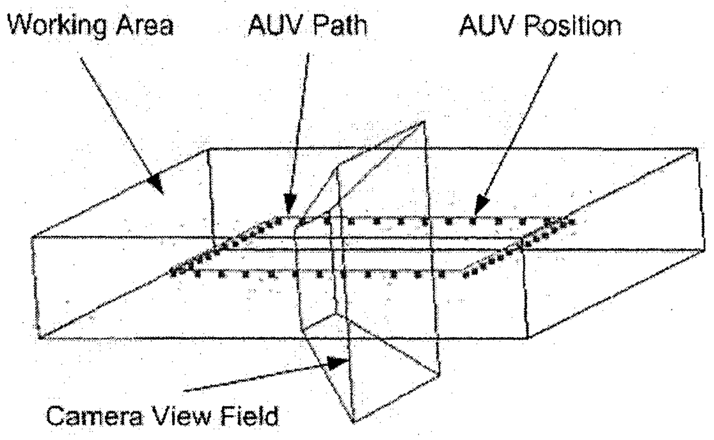


Figure C.2: The Augmented Camera View Field in Working Area

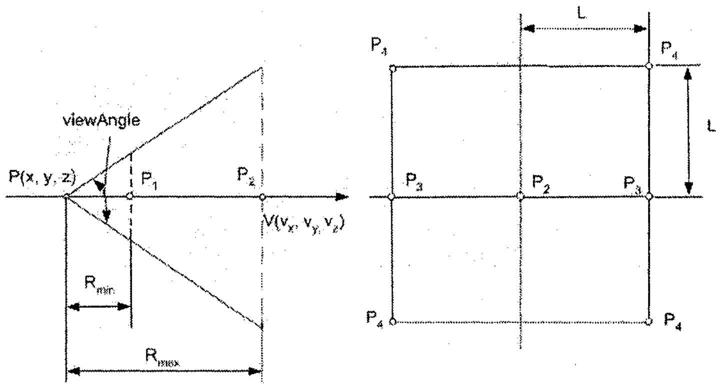


Figure C.3: Camera View Field in Detail

space. This is a difficult task. The camera view field in a real simulation system is displayed in Fig. C.2.

Assume a sensor, such as a camera, has the following parameters: view angle (α), minimum range (R_{min}), and maximum range (R_{max}). The structure of the view volume is displayed in Fig. C.3.

There are eight points which define the camera view field, and the constructor of the view field must obtain the coordinates of the points, which are actually formed by two squares, front determined by maximum range, and back by minimum range. When one square is determined, another can be calculated by a simple change in the range data of the sensor. We display the details for the front square as follows: The steps to calculate the view volume is as shown in Alg. 8. The basic assumption for this algorithm is that the robot will not rotate around $x - axis$. That means there is no roll during the robot navigation.

Input: Robot position P and direction $V(v_x, v_y, v_z)$, maximum range of the sensor R, view angle of the sensor α (See Fig. C.3).
Output: Four Points which decide the front square of view field.

```
1: Determine $P_2$ on the line which pass $P$ with direction $V(v_x, v_y, v_z)$ and the distance between $P$ and $P_2$ is $R$.
2: Determine $P_3$ on the front plane, where the distance between $P_2$ and $P_3$ is $L$. There are two $P_3$ points
3: Determine $P_4$ on the from plane, there $\bar{P_2P_4} = \sqrt{2}L$ and $\bar{P_3P_4} = L$. There are four such points, which consist of the front plane of the the view field
```

Algorithm 8: Compute the front square of view volume.

C.3 Landmark Detected Information Check

Sensor view field or view volume is a convex space with six faces, and can be simplified to a polyhedron (Fig. C.4). The algorithm to decide whether a landmark will be located in the view volume or not is shown in Alg. 9.

C.4 Rotation Matrix Interpolation

In this simulation system, a robot navigates along a given path at a given speed. At a turning point, the robot only rotates from one direction to another direction. How to decide the pose of the robot during the turn process is a big challenge.

At any time, a robot's orientation can be decided by a rotation matrix or a direction vector related to the original orientation. Assuming that a robot has a direction v_1 at position P. At this point, the robot will only rotate from direction v_1 to v_2, where v_1 and v_2 are space vectors. The problem is deciding the robot's direction

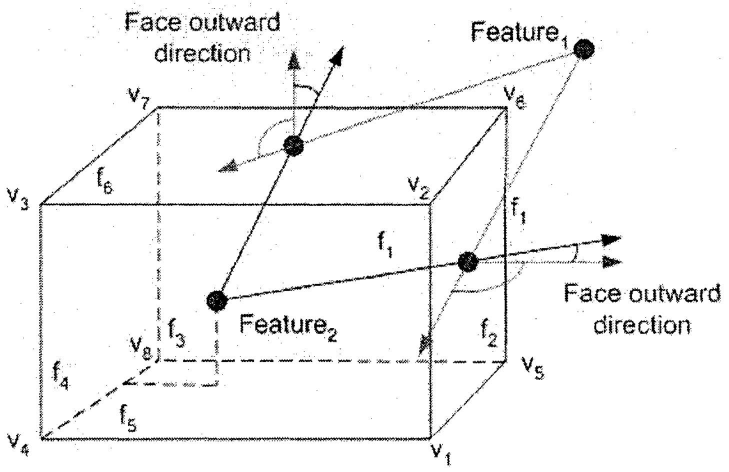


Figure C.4: Approach to determine a landmark detected information given by a sensor view field. Where v_i is the vertex, f_i is the face ($i = 1, \cdots, 6$)

Input: A landmark L, and view volume of a sensor V (See Fig. C.4).
Output: The landmark is in view volume or not

```
1: for i=1 to 6 do
2:   calculate the centre point $p_i^c$ of face $f_i$
3:   compute the outward face direction $d_i$
4:   construct a vector $d_L$ from landmark $L$ to centre point $p_i^c$
5:   calculate the angle $\alpha$ between vector $d_i$ and $d_L$
6:   if $\alpha > 90$ then
7:     Return false
8:     Stop checking
9:   end if
10: end for
11: return true
12: stop checking
```

Algorithm 9: Check whether a landmark is located at view volume or not.

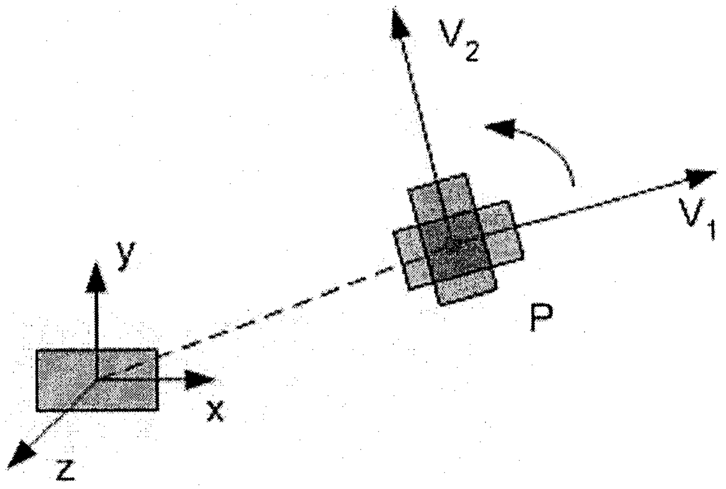


Figure C.5: A robot rotates at a turning point P. Where v_1 is the start direction of the robot, and v_2 is end direction of the robot at position P

at any time between v_1 and v_2 (Fig. C.5). From the robot's direction vector, it is possible to obtain its corresponding rotation matrix relative to its original direction, supposing they are M_1 and M_2. The problem is finding a rotation matrix M_t between matrix M_1 and M_2. The solution can be obtained by matrix interpolation.

By applying interpolation techniques, the rotation matrix M_t can be generated using a blending function with the parameter t, $t \in [0, 1]$. At $t = 0$, the matrix M_t is equal to the matrix M_1. At $t = 1$, the matrix M_t is equal to the matrix M_2. Then the rotation matrix (M_t) is specified as:

$$M_t = f(M_1, M_2, t) \tag{C.4.0.4}$$

where f is a blending function. The algorithm to compute an interpolated matrix is shown in Alg. 10.

Input: matrix M_1 and M_2, parameter t
Output: a matrix M_t

```
1: compute a rotation matrix $M_T$ that will transform $M_1$ to $M_2$ by $M_T = M_2 M_1^{-1}$
2: transfer matrix $M_T$ to its corresponding quaternion $Q_T$
3: transfer quaternion $Q_T$ to its corresponding rotation axis $V$ and angle $\theta$
4: compute a new angle $\theta_t$ by blending function $f$, then $\theta_t = f(t, \theta)$
5: transfer the rotation axis $V$ and angle $\theta_t$ into a quaternion $Q_t$
6: transfer the quaternion $Q_t$ into rotation a matrix $M_s$
7: $M_t = M_s M_1$
```

Algorithm 10: Compute an interpolated matrix between two matrixes