

University of
South Australia

TOWARDS SCENE UNDERSTANDING USING A COOPERATIVE OF ROBOTS WITH LIMITED SENSING

Stephen Jeyapaul Franklin

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Systems Engineering

Knowledge-based Intelligent Engineering Systems Centre
School of EIE
University of South Australia
May, 2010

© Stephen J. Franklin 2010

To my family and friends…

Author’s Declaration

I declare that this thesis is based on my work during my Ph.D. study at University of South Australia, which has not been submitted for a degree or diploma in any university; and to my best knowledge it does not contain any information written or submitted by other authors except where due reference is made in the text.

Stephen Jeyapaul Franklin

Adelaide, SA

Abstract

Scene understanding is an essential element involved in the exploration of an unknown territory using mobile robots. In this regard, scene understanding refers to the identification and localization of elements in the scene. When the scene is dynamic by nature or has objects that move around, they need to be characterized and discriminated from the static elements, which is essential for the SLAM (Simultaneous Localization And Mapping) process when external global localization information is not available. We present some techniques and algorithms to accomplish these goals with regard to a scenario where we have a co-operative of robots equipped with a monocular camera having three degrees of freedom (x,y,θ), observing a dynamic scene. The objectives are to use the static elements of the scene for self-localization of robots and try to estimate the velocity, trajectory and structure of moving objects. The latter is much more difficult to solve than the former. We discuss some metrics that have been identified that could be used for the co-ordination of robots and develop optimized steering mechanisms, which would help reduce the errors associated with the estimated parameters.

Acknowledgements

Firstly, I would like to thank Prof. Lakhmi Jain, Head of the Knowledge based intelligent Engineering Systems (KES) centre and the University of South Australia for giving me this opportunity. Special thanks to Prof. Jain for his guidance, support and encouragement all through my candidature. I would like to thank my supervisors, Dr. Anthony Finn and Mr. John Pattison for their wonderful guidance and giving me the liberty to conduct this study on my own terms. I would also like to thank Dr. Samuel Drake for his help and comments on the mathematics. This thesis was funded in part by the Australian Defence Science and Technology Organisation (DSTO). Their support is gratefully acknowledged.

I would like to thank my colleagues, Beulah, Jinsong, Mohammed, Angela and everyone else in my group for all the conversations and help. Special thanks to Mohammed for supporting me with my rather ambitious project, the details of which I wouldn't discuss here! Thanks to all my students at the electronics laboratory, I have always enjoyed the time I spent there and I have learnt a lot from you guys through your numerous queries.

Finally, I would like to thank my parents for their support. Thanks to all my family and friends for their encouragement and belief in me when I wasn't so sure. Special mention to my periamma for teaching me maths in a unique way; the Ramanathan family for giving me a welcome break during the weekends and keeping me company; Denie, Aravind and Sakthi, you guys have been great and been there always; Sundar and Aravind for making my stay in Adelaide memorable with all the trips to the eateries, the laughs and what not; Navi and Liz for patiently listening to everything I had to say about my research and experiences. Finally, thanks to the bunch of guys and girls who call themselves 'Tamizhaga Aussies!', your company has been amazing, to say the least!

Table of Contents

Author's Declaration ..... **iii**
Abstract ..... **iv**
Acknowledgements ..... **v**
Table of Contents ..... **vi**
List of Figures ..... **ix**
List of Abbreviations ..... **xi**
List of Symbols ..... **xii**

CHAPTER 1 INTRODUCTION 1
1.1 Motivation 2
1.2 Problem and Application Domain 3
1.2.1 Applications 3
1.2.1.1 Unmanned Ground Vehicles 3
1.2.1.2 Unmanned Air Vehicles 3
1.2.1.1 Autonomous Underwater Vehicles 4
1.2.2 Problem Domain 4
1.3 Bearing-only SLAM 5
1.4 Moving Object Tracking 6
1.4.1 SLAM with Moving Object Tracking 7
1.5 3D Structure Estimation 7
1.5.1 Why 3D? 8
1.5.2 3D Representation of Objects 10
1.6 Multi-robot Coordination 11
1.7 Optimization 13
1.8 Thesis Statement 13
1.9 Aims and Objectives 14
1.10 Thesis Organization 14
1.11 Publications 15

CHAPTER 2 THEORETICAL FRAMEWORK 16
2.1 Background 16
2.2 Stochastic Estimation 16
2.2.1 Bayesian Estimation 17
2.2.2 Batch Estimation 18
2.2.2.1 Projective Factorization 18
2.2.2.2 Bundle Adjustment 19
2.2.3 Recursive Bayesian Estimation 20
2.2.4 Kalman Filtering 21
2.2.5 Extended Kalman Filter 23
2.2.6 Other Techniques 24

2.2.6.1 Computational Efficiency 25
2.2.6.2 Unscented Kalman Filter 25
2.2.6.3 Robust Estimation Schemes 26
2.3 Motion Optimization and Control 26
2.3.1 Control Structure 27
2.4 Summary 28

CHAPTER 3 SYSTEM MODELLING 29
3.1 Development and Testing Scenario 30
3.1.1 Simulation with Webots 31
3.2 Simulated Indoor GPS for Ground Truth Verification 32
3.3 Kalman Filter for the Individual Robots 34
3.4 Moving Object Tracking 41
3.5 Optimal Sensor Placement 46
3.6 Summary 48

CHAPTER 4 COORDINATION FOR COOPERATIVE STRUCTURE ESTIMATION 50
4.1 Cost Function 51
4.1.2 System Model 51
4.1.3 Information based cost functions 52
4.1.4 Nonlinear Model Predictive Control 57
4.1.5 Cost function significance 59
4.2 Constraints 60
4.2.1 Obstacles 61
4.2.1.1 Static obstacles 61
4.2.1.2 Dynamic obstacles 63
4.2.2 Field of view 64
4.2.3 Velocity and Acceleration 67
4.3 A Solution to the Optimal Control Problem 67
4.3.1 Dynamic Programming 67
4.3.2 Principle of Optimality 68
4.3.3 General Form 69
4.3.4 Nonlinear Model Predictive Control (NMPC) 70
4.3.5 Branch-and-Bound 71
4.4 Summary 73

CHAPTER 5 RESULTS AND DISCUSSION 74
5.1 The experiment 74
5.2 Localization 75
5.3 Moving target estimation 76
5.4 Optimization 78
5.5 Summary 81

CHAPTER 6 CONCLUSION 82
6.1 Conclusion and Summary 82
6.2 Future Scope 83
6.2.1 Situation Awareness/Scene Understanding 83

6.2.2 SIFT/SURF Integration 84
6.2.3 Complexity 84
6.2.4 GPS Integration 85
6.2.5 Implementation 85
References 86

List of Figures

Figure 1 – (Left) Single robot SLAM – recursive estimation process, (Right) Localization from two views from two observing robots (θ_1 and θ_2 measured anticlockwise from the x-axis). 2

Figure 2 – Framework for an image based feature evaluation system: from [3]. 9

Figure 3 – View from different angles and a rendered image from the 3D model: from [19]. 10

Figure 4 – Top row shows several input images and the bottom row shows the rendering of the object using the patches used for representation. It can be seen that the models are incomplete but this is sufficient for the purpose of recognition: from [20]. 11

Figure 5 – Kalman filtering process 22

Figure 6 – Linearization. 24

Figure 7 – Controller structure. 28

Figure 8 – Webots Model (2x2m area) 32

Figure 9 – A view from the overhead camera sharing the orientation and localization of the robots. 33

Figure 10 – Vehicle model. 35

Figure 11 – Feature extraction (blue dots – moving object, red dot – landmark) 39

Figure 12 – Functional block diagram for information flow. 39

Figure 13 – Trajectory estimation using dead reckoning and SLAM along with ground truth. Magenta – Dead reckoning, Blue/Cyan – Ground truth, Green/Red – SLAM and Cyan/Red – Sections where no landmarks are seen. 41

Figure 14 – Functional block diagram. 42

Figure 15 – Information sharing/update scheme on the supervisor. 45

Figure 16 – Moving object track along with the trajectories of the robot. Green – Moving object track (estimated), Magenta - Overlaid moving object ground truth, Red – Ground truth for the robot trajectories and Blue – EKF SLAM estimation of the robot trajectories..... 46

Figure 17 – Illustration of the effect of different observation points on localization error. 47

Figure 18 – Sensor-target geometry with the measured angles. 53

Figure 19 – Control and estimation structure. 58

Figure 20 – Representation of obstacles. 62

Figure 21 – The matrix after convolution 62

Figure 22 – A discrete 2D point spread function of size 13×13 and $\sigma = 1$..... 64

Figure 23 – Camera field of view and measurement range. 65

Figure 24 – A plot of the function 10^x for $x[-10,10]$. 66

Figure 25 – A plot of the double exponential function with $A = 100$ and $x[-1,1]$..... 66

Figure 26 – Structure of the problem. 73

Figure 27 – Estimated trajectory with the error ellipses; arrows indicates the regions where the uncertainty increases and then decreases as soon as a landmark (or map feature) is observed. 75

Figure 28 – Robot trajectories with error ellipses along with the moving object track.... 76

Figure 29 – Error ellipses associated with the red beacon. 77

Figure 30 – Error ellipses associated with the green beacon. 78

Figure 31 – Centralized motion optimization. 79

Figure 32 – Distributed motion optimization..... 81

List of Abbreviations

Abbreviation	Augmentation
AOA	Angle Of Arrival
AUV	Autonomous Underwater Vehicles
BB	Branch-and-Bound
BIBO	Bounded Input Bounded Output
BRL	Behavioural Primitives Level
CI	Conditionally Independent
CPL	Conceptual Primitives Level
CRLB	Cramer-Rao Lower Bound
DATMO	Detection And Tracking of Moving Objects
DP	Dynamic Programming
EKF	Extended Kalman Filter
FIM	Fisher Information Matrix
GDOP	Geometric Dilution Of Precision
GPS	Global Positioning System
ISE	Image Sequence Evaluation
ISL	Image Signal Level
LQR	Linear Quadratic Regulator
MIMO	Multiple Input Multiple Output
MMSE	Minimum Mean Squared Error
MO	Moving Object
MPC	Model Predictive Control
MSE	Mean Squared Error
NMPC	Nonlinear Model Predictive Control
PDL	Picture Domain Level
PSF	Point Spread Function
SAL	Sensor-Actuator Level
SDL	Scene Domain/3D data Level
SFM	Structure From Motion
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous localization and mapping
SLAM with GO	SLAM with Generic Objects
SO	Stationary Object
SURF	Speeded Up Robust Features
UAV	Unmanned Air Vehicles
UGV	Unmanned Ground Vehicles

List of Symbols

Symbol	Explanation
$E[x]$	expectation of x
$\hat{\Theta}(x)$	an estimator of x
$V_{r/l}$	robot right/left wheel velocity
k	time-step
b	robot wheel axle length
$N_{r/l}$	robot right/left wheel encoder values
D_n	robot wheel diameter
X	state vector
U	control command vector
f	state transition matrix
F	Jacobian of the state transition matrix
B	controllability matrix
Z	observation/measurement vector
h	measurement matrix
H	Jacobian of the measurement matrix
K	feedback factor or Kalman gain
Q	error covariance of the process noise
R	error covariance of the observation/measurement noise
P	covariance of estimation error
σ	standard deviation
x, y, θ	robot pose
T_x, T_y	target beacon location
L_x, L_y	landmark location
I	cost function
μ	control command sequence
μ^*	optimal control command sequence

Chapter 1

Introduction

Autonomous mobile robots have generated a lot of interest over the past two decades. This is due to the fact that they have a wide range of applications. One such application is the exploration of unknown spaces such as ocean floors, other planets, hostile regions in the context of military surveillance, etc. One of the major objectives of such exploration is scene understanding and situational awareness. Situational awareness as defined by Endsley [1] is, “the perception of elements in the environment, the comprehension of their meaning in terms of task goals, and the projection of their status in the near future”. In our case, the basis for situational awareness lies in the determination of the spatial relationships between the observer (the robot) and the objects/features in its vicinity and the spatial relationships between the features. In the case of dynamic objects the changes over time also need to be tracked [2]. These essentially constitute localization, mapping, moving object tracking (trajectory and velocity estimation) and 3D structure estimation of the artefacts in the scene. This kind of information is derived from sensors onboard the robot, which in our case include odometers for each wheel and a fixed CCD camera, although it should be noted that the measurements from all of these sensors have uncertainties associated with them.

The Simultaneous Localization And Mapping (SLAM) problem and motion tracking problem have been solved separately and have also been integrated under the assumption that both range and bearing information is available with respect to an observation point and direction. When range information is not available, it is not possible to determine the trajectory of a moving object uniquely. We intend to solve this problem using a co-operative of two robots whereby, at any instant a moving object can be localized using bearing measurements from two robots and their own localization. Figure 1 (right)

illustrates this. It should be noted that all the measurements used have uncertainties attached to them and hence the location of the moving object is estimated as a region where it is expected to be. The recursive estimation process used to keep track of the location of the robots is illustrated in Figure 1 (left). Further, the pose of each robot relative to each other has a bearing on the estimated region [3]. Therefore, coordination of the two robots is also undertaken such that the location uncertainty of the moving object is minimized. We also discuss the integration of such a system with existing SLAM algorithms. 3D structure estimation stationary objects and SLAM have also been integrated [4]. However, complete 3D information about all objects (moving or stationary) needs due consideration, which is the main focus of this research.

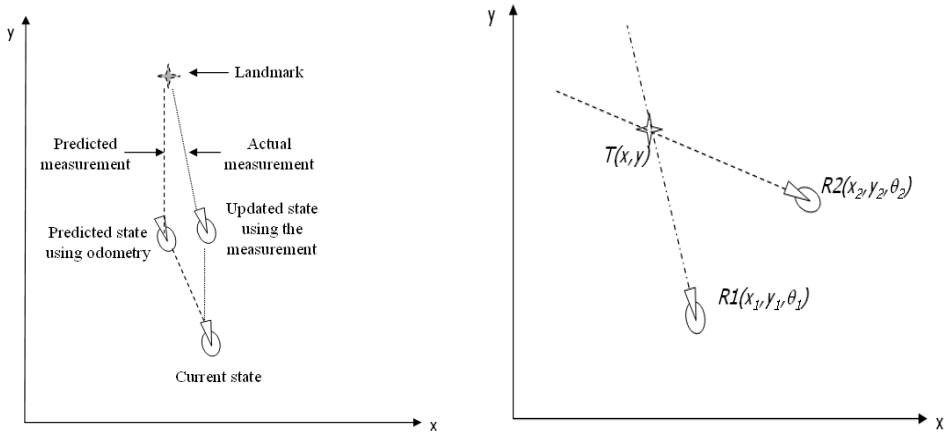


Figure 1 – (Left) Single robot SLAM – recursive estimation process, (Right) Localization from two views from two observing robots (θ_1 and θ_2 measured anticlockwise from the *x*-axis).

1.1 Motivation

Studies have suggested that the availability of skilled human resources to the defence forces is expected to fall in coming years [5]. The need to maintain operational effectiveness means that some work will require robots that can act autonomously and cooperatively. When a single skilled human operator can supervise a group of robots appropriately, the contribution to the operation is likely to be maximized. Also, this

provides a subjective human input to the operations, allowing the system to be less predictable.

Having accurate and up-to-date situational awareness is critical to both human assisted and automated decision making. Although this is critical to the decision making process, accurate situational awareness has been found to be elusive in applications such as air traffic control [6], military command and control operations [7]. One of the objectives of this work is to provide a better situational awareness for applications in the context of the use of autonomous vehicles and automated decision making.

1.2 Problem and Application Domain

1.2.1 Applications

This research can be applied to a vast domain of applications that utilize autonomous vehicles. The primary reason for the use of autonomous vehicles is the prevention of human personnel from being exposed to hazardous environments that might put their life at risk. The following are some of the specific domains where this research can be used.

1.2.1.1 Unmanned Ground Vehicles

There are two types of Unmanned Ground Vehicles (UGVs); tele-operated and autonomous. Our research focuses on autonomous vehicles. UGVs are designed to perform the same operations as inhabited vehicles such as surveillance, reconnaissance, mapping unknown territory, etc. The difference is that they are designed to operate in hazardous environments where direct human intervention may not be possible. UGVs are often required to make their own decisions, workout strategies and execute them.

1.2.1.2 Unmanned Air Vehicles

Unmanned Air Vehicles (UAVs) are used extensively by the military to perform various operations. They are primarily used for remote sensing using different type of sensors such as infrared, radar, biological sensors, etc. Fixed-wing UAVs are used for the transport of goods required for military operations. UAVs are also used for precision

ground target missile strikes. UAVs are also being used for civilian applications such as urban search and rescue, and meteorology applications for the collection of real-time data in hazardous environments such as the hurricanes.

1.2.1.1 Autonomous Underwater Vehicles

Autonomous Underwater Vehicles (AUVs) have three major applications, 1. They are used by the military to monitor protected areas, anti-submarine warfare, detection of other underwater vehicles, search for unexploded mines etc. [8]. 2. They are used commercially to map the seafloor by the petroleum industry before the installation of infrastructure [9]. 3. They are also used for research by marine biologists to monitor reef activity, mapping and profiling the reef, determine the chemical composition of sea water, etc.

1.2.2 Problem Domain

In order to get a complete understanding of an observed scene, it is essential that we characterize the various static and dynamic objects present in it. From our perspective, the key to performing this is to find the spatial relationship between the static features in a scene and find the temporal relationship of the dynamic features. We use dynamic robots to observe a given scene; therefore it is also necessary to localize the robot. For fully autonomous operation, it is essential that this is done without any external localization infrastructure. This calls for the integration of three well researched problems, which include SLAM, moving object tracking and 3D structure estimation. In our case, it should be noted that these three processes are highly interlinked and therefore, an integrated estimation process can be expected to provide much better results than if all of these processes run separately. Finally, we have assumed the use of a cooperative of robots with a very limited sensor suite, namely, a monocular camera and wheel encoders, keeping in perspective our aim of providing an affordably expendable solution. This requires actions of the robot team to be coordinated rather than each member of the team doing what it thinks is best from its own perspective. This also means that the robots

move on paths that provide maximum information about the scene or, in other words, work towards following an optimal path considering the various constraints as a team. The following sections provide some insight into each of these three areas.

1.3 Bearing-only SLAM

Simultaneous localization and mapping (SLAM) [10] provides a solution to the mapping and localization problem when external localization techniques (such as GPS) are not available or are unreliable. However, this has been implemented with the use of laser scanners that provide both range and bearing information [10, 11]. In our case, we have the constraint of using a single monocular camera with a limited field of view. This means that we get bearing information only and no range information unless there is some prior information available about the tracked objects, but this cannot be assumed. There has been a lot of interest on image based SLAM and it is possible to generate a map of the environment and estimate camera ego-motion simultaneously [12]. Scene structure estimation has also been added to image-based SLAM by fusion of vision and inertial information[4].

Image-based SLAM has made significant progress in the recent past and it is now solvable in real time [12]. Distinctive image features are used as landmarks on the map and, when the landmarks are revisited, it aids localization. This process goes on recursively. However, it should be noted that in order for the features to be used as landmarks, they need to be stationary. In both [12] and [4] non-stationary objects, or dynamics in the scene, have been identified to be a hindrance for the algorithm. However, if the dynamics of the scene are handled (identified correctly) and then characterized, this has been found to aid SLAM and vice versa [2]. In this work we have assumed that the SLAM technique has already been initialized [13] and that loop closure has already been performed. Initialization and loop closure for bearings-only SLAM is non-trivial, although, a solution for the problem using variable state dimension filters is available [14].

1.4 Moving Object Tracking

There has been a lot of work on motion estimation using bearing measurements alone. Motion estimation, as opposed to trajectory estimation, estimates the velocity and orientation of the tracked object whereas trajectory estimation also includes localization information. There are two distinct approaches for motion estimation from image sequences (i) feature-based analysis and (ii) optic flow based analysis [15]. There are a lot of variants of these two techniques that have been implemented for different applications and both approaches have their own merits and demerits. Stereoscopy has been known to aid motion analysis but it is beyond the scope of this study.

Feature-based analysis [16] [17] relies on the extraction of sparse and distinct image features such as lines, boundaries of objects and corners. Such features are extracted from each image in a sequence and correspondences are determined. The displacements of the observed features are then used to solve a set of non-linear equations that are formed using constraints that govern the displacement. The constraints are generally based on assumptions such as the tracked object being rigid, laws of motion that govern the movement of the object, etc. A solution to the set of non-linear equations leads to the estimation of motion parameters of the tracked object. The extraction of distinct image features and the establishment of correspondences between images are non-trivial and computed using specialized algorithms such as Scale Invariant Feature Transform (SIFT) [16] and Speeded Up Robust Features (SURF) [17]. Among other issues, the performance of many of these algorithms degrades greatly when there are variations in ambient lighting. The assumptions that are used to form the non-linear equations do not always hold and it has also been known that the development of a stable and accurate recursive solution to this problem is difficult.

Optical flow based analysis[18] can also be used for the same purpose but has a very different formulation. Feature extraction and the establishment of correspondences between images are completely avoided. A velocity vector is calculated at each pixel on the image using the instantaneous brightness variation over time in the local neighbourhood. The technique relies only on the local spatial and temporal derivative of

the brightness. The computed velocity map is generally a function of the relative motion between the camera and the objects of the scene. Constraints can be used as before to estimate the motion parameters of the objects in the scene. Although optical flow based analysis avoids issues faced by feature based analysis, it has issues of its own. A technique like this is always susceptible to sensor white noise, which is present in the frames of a video sequence and, since the analysis is limited to a local spatial domain, there is a velocity constraint on the dynamic object to be tracked. If objects moving with high velocity are to be tracked, very high frame rates and/or high resolution cameras are usually required and the computational time of the algorithm also becomes critical.

1.4.1 SLAM with Moving Object Tracking

In the recent literature [2] two approaches have been proposed to incorporate motion tracking with SLAM. Although the implementation is with the use of laser scanners, the framework for our application is very similar as the only thing that differs is the measurement model. The first approach described is called SLAM with Generic Objects (SLAM with GO). It is very similar to the basic SLAM algorithm, except for the addition of motion mode information for the landmark. The motion mode (stationary, moving, move-stop-move, etc) has to be learned from the measurements. It is very simple but has been found to be computationally expensive and unsuitable for real-time implementation. The second approach, aimed at solving this issue is called SLAM with Detection And Tracking of Moving Objects (DATMO). In this approach each new moving object gets its own statistical estimator (a Kalman filter) and motion mode and state estimation takes place separately and is then used to update the Extended Kalman Filter (EKF) used for SLAM. This has been found to make the process run faster and hence more suitable for real-time implementation.

1.5 3D Structure Estimation

The area of 3D profile generation from 2D images is a little more than a decade old. There have been a lot of techniques proposed in this area [36]. In the initial stages, the

techniques required special equipment and a large number of images of the object taken from different angles. After some progress new techniques were developed which could manage with a limited number of images and did not require special equipment. However, camera calibration was still a requirement and user interaction was also required [12]. Today, techniques exist that do not need the cameras to be calibrated and can work without user interaction[17].

1.5.1 Why 3D?

While processing 2D images is more convenient and computationally less intensive compared to 3D, it misses out in the area of robustness because objects tend to appear very differently on images when captured from different perspectives. Recent studies looking at the development of cognitive vision systems [4] support this and suggest the use of model based scene domain processing for tasks such as, object recognition, behaviour recognition, etc. Also, 3D model based systems greatly improve performance of automated object recognition systems in the presence of occlusions, clutter, changes in the environment, etc., because of the fact that structural information is the same irrespective of the sensor used to generate the information. They also open the possibility to find 3D pose estimates of the object.

Work in the area of image based feature description and recognition has raised the need for a systems approach to solve this problem. A coarse system architecture or framework for such a system can be found in [3] which look at the problem of Image Sequence Evaluation (ISE) for natural language text description of a given video sequence. A very similar framework can be adopted for our problem.

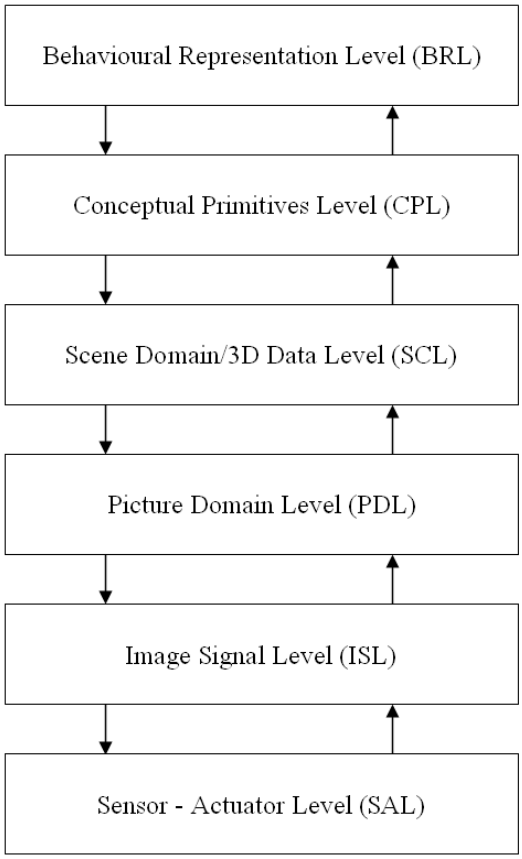


Figure 2 – Framework for an image based feature evaluation system: from [3].

The lower layers, SAL and ISL deal with lower level functions such as image acquisition, camera parameter management, noise removal, auto focus, etc. However, the focus of this research will be on the SCL and CPL with a bit of input from the PDL as well. At the SCL we look at the generation of a good 3D profile using cues from the PDL. The CPL looks at features like recognition and generating end results such as maps and feedback data to the lower levels as well. The BRL specifies how the system would react to signals or alerts from the lower layers. Work at this layer depends on the performance of the lower layers.

1.5.2 3D Representation of Objects

From our perspective 3D object representation techniques can broadly be classified into complete 3D rendering and sparse 3D rendering. Complete 3D rendering refer to techniques commonly used in virtual reality type applications where a near complete 3D profile of an object is developed from a set of 2D images. This does not suit our purpose because of numerous problems associated with it. Firstly, our application is not that of virtual reality modelling but rather to autonomous derive situational awareness information and make online decisions based on that. Secondly, this process is computationally intensive and requires a lot of memory to store the 3D profile of an object and therefore not suited well for real-time implementation. "Shape from Silhouette" [19] is one such method used for complete 3D rendering and Figure 3 shows a typical reconstruction using a set of thirty six 2D images, two of which are shown on the left.

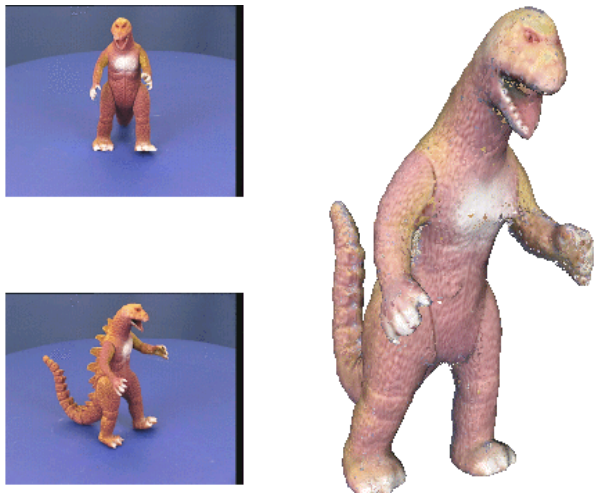


Figure 3 – View from different angles and a rendered image from the 3D model: from [19].

Alternatively, a scheme for the representation of objects using scale and affine invariant interest regions has been proposed by Rothganger et al. in [20]. A recognition scheme using multi-view geometry has also been proposed in [20], which imposes pose-based constraints on the matching process. Images of the object taken from different angles can

be used to first match the overlapping patches, and then this can be used progressively to form a 3D model of the object. The output will be in the form of rectified patches (typically parallelograms) and their rectification transformations. This specifies the orientation of the patch in reality. Figure 4 shows some results from this algorithm.



Figure 4 – Top row shows several input images and the bottom row shows the rendering of the object using the patches used for representation. It can be seen that the models are incomplete but this is sufficient for the purpose of recognition: from [20].

1.6 Multi-robot Coordination

It has been recognized that a lot of tasks can be performed more effectively and robustly using multiple robots [5, 21]. A review of the primary research areas and the tasks in this area can be found in [22]. The research areas of interest to us are mapping, exploration and motion co-ordination. The single-robot version of this problem is fairly well researched and understood. However, the multi-robot version cannot be solved using the same techniques as information sharing between the robots and co-ordination needs to be added.

There are two distinct approaches to this problem. One of them is the centralized approach [23] where the leader controls all the other robots. The leader is aware of its own parameters like speed, orientation, localization, as well as that of the other units. This approach becomes computationally expensive as the number of robots increases. The other approach is the distributed approach [24] where each robot has its own

decision-making ability. We use a combination of the distributed and centralized approach because, as the robots are autonomous, initially they will be in a search mode and looking for targets. Once a robot finds a target, this information needs to be shared with at least one other robot so that it can join the pursuit and move towards getting images of the target from different angles in order to generate the required parameters in the best possible way. The system then switches to a centralized approach where, information is shared through a base station that is also responsible for estimating the parameters of the moving target and co-ordination of the robots.

Coordination can also be classified based on the degree of interaction between the robots. Classification based on the degree of interaction falls into two categories; loose coordination and tight coordination [25]. Loose coordination is used in applications where the task can be completed by an individual robot but a team would complete the same task more efficiently. Examples include surveillance, exploration of unknown territory, etc. Coordination is generally performed by assigning each team member a subtask that can be carried out by itself without any assistance from the team members. Tight coordination is used in applications where an individual robot does not have the ability to perform the task by itself and requires the assistance of one or more of the team members. Examples include cooperative object manipulation, following a formation pattern, movement with constraints such as communication range, etc. In these kinds of applications, the task cannot be split into subtasks that can be performed independently by a team member. The actions of the robots need to be correlated for the task to be completed successfully. Our application falls into the latter category. Finding the trajectory of a moving object cannot be done by an individual robot equipped with a bearing-only sensor. Bearing measurements from two robots are required to localize an object (see Figure 1 (right)). Therefore, a two robot team needs to be tightly coordinated for our application.

1.7 Optimization

Mathematical optimization refers to the process of selecting the best possible element from a set of feasible alternatives. The selection is based on finding the maximum or minimum of a given objective function. The choice thus made is said to be optimal with respect to the given objective function. Trajectory optimization is a variant of mathematical optimization that refers to the process of choosing the trajectory for a moving vehicle that is optimal with respect to some measure of performance and simultaneously satisfy the constrains associated with the dynamic system. Given a trajectory, the vehicle then needs to be driven or controlled in order to follow the trajectory; this makes it an optimal control problem. Any autonomous vehicle should have the capability to generate reference trajectories that are optimal in some sense and follow them in the presence of uncertainty and re-plan whenever there is a change in the environment or mission. Given a reference trajectory, this is generally done by constructing a state-space trajectory and then a feed forward compensator can be used to find the corresponding control input for the vehicle to follow the reference trajectory taking into account the vehicle dynamics [26]. Feedback control is then used to reduce the error, which denotes the difference between the reference trajectory and the actual trajectory. This can be termed as the classical approach and it has to be noted that the trajectory generation stage of the process and the control stage are completely decoupled. The other approach, termed as optimization-based control [27], is slightly different from the classical approach and it has integrated trajectory generation and control. This is the approach used in our study; more details are given in section 2.3 and chapter 4.

1.8 Thesis Statement

Performing SLAM, moving object tracking and the estimation of the sparse 3D structure of a moving object collectively with active control will produce better parameter estimates than each of them performed independently for obtaining real-time autonomous situational awareness with minimal sensing capability.

1.9 Aims and Objectives

- Study and review the current state of the art in the broad area of autonomous systems with regard to the derivation of situational awareness/scene understanding.
- Develop an estimation methodology that can be used to estimate various parameters that constitute situational awareness in a dynamic environment for a multi robot cooperative that has limited capabilities.
- Develop an information based optimization scheme where, the actions of the robot team are coordinated and near-optimal with regard to the information gained.
- Test the proposed estimation methodology and optimization scheme in a synthetic environment that would validate the claims made in this thesis and demonstrate the effectiveness of this study.

1.10 Thesis Organization

Chapter 2 presents a more detailed review of the existing literature on the problems outlined in the previous sections and their integration. First, we look at the SLAM problem in general with a focus on the mathematical formulation and present some differences and issues that arise when performing bearing-only SLAM. Then, we consider the problem of tracking a moving object and discuss how this can be integrated with SLAM and the extensions required on the mathematical formulation. Then, we look into the coordination aspects of the robot team and the coordination model used for our application. Finally, we review some of the motion optimization techniques and present some metrics that can be used for the optimization that would facilitate a near-optimal estimation of the required parameters in the shortest possible time (Since the metrics are predictive by nature and optimality cannot be guaranteed).

Chapter 3 presents the mathematical modelling of the system. We present the mathematical model used to perform SLAM and the various steps involved in the

process. This includes perception modelling, motion modelling, data association, the recursive estimation process and the information sharing scheme between the robots. Then we present the extensions required to accommodate moving objects in the model and their structure.

Chapter 4 presents the coordination scheme used for the system and describes the level of autonomy that the robots have at the different stages of the operation, supervisory control and how they can cope with issues such as, communication link failures, etc. Finally, we discuss how motion planning has been implemented and the implementation of the motion optimization techniques.

In Chapter 5, we discuss the simulation test scenarios used and the results obtained. The results present a verification of the theoretical model developed and show its effectiveness. We present the initial results and then show the effect of motion planning and optimization. We then discuss the suitability of our approach to real applications and review the limitations.

Chapter 6 presents an overview of the study that been undertaken and conclusions that can be drawn from it. Possible future extensions to this study are also outlined.

1.11 Publications

- S. Franklin, A. Finn, J. Pattison, and L. C. Jain, "Towards scene understanding using a co-operative of robots," J. Intell. Fuzzy Syst., vol. 21, pp. 101-112.
- S. Franklin, A. Finn, J. Pattison, and L. C. Jain, "A Distributed Information-based Motion Optimization Scheme for Cooperative Mobile Robots with Bearing-only Sensors," in Innovations in Defence Support systems: Springer-Verlag, Germany, 2010. In press.

Chapter 2

Theoretical Framework

2.1 Background

There are two distinct phases in solving the problem outlined in section 1.9. One is the problem of parameter estimation and the other is the problem of motion optimization. Although these two processes are deeply intertwined, there are distinct mathematical frameworks available to solve these problems. Parameter estimation refers to the process of deducing the system state based on a system model and sensor measurements, both of which are known to be noisy. Motion optimization refers to the process of developing a feedback control law that guides the robots on a path that is optimal in some sense (i.e., minimize a given cost function). These two processes are intertwined because the metrics used in the development of the cost function come from the state estimation process. Bayesian estimation theory provides a solid basis for this kind of inference problem and the techniques considered here for parameter estimation work on the basis of providing minimum mean squared error. The optimization problem is solved using the concept of dynamic programming wherein, a complex problem is broken down sequentially into several simpler problems over time which in turn leads to a solution to the original problem. In this chapter, we will review some of the common techniques used to solve these problems and provide a rationale for the choice of techniques used in this work.

2.2 Stochastic Estimation

For the SLAM and moving object tracking problems, we use a state space model where the system state consists of a set of random variables, $X = \{x_1, x_2 \ldots x_n\}^T$, with a known

probability distribution represented in terms of a covariance matrix. However, these random variables are not readily observed. Instead, the observations from the sensors consist of another set of random variables which are related to the system state by a given relationship. The observed random variables, $Y = \{y_1, y_2 \ldots y_n\}^T$, also have a known probability distribution specified by a covariance matrix. This class of problems falls under the domain of estimation theory [28-30]. Estimation theory basically deals with the development of an estimator, $\hat{\Theta}$, which is a function of the observation vector, $Y = \{y_1, y_2 \ldots y_n\}^T$ that estimates the state vector, X but is independent of X. There are different types of estimators available and some are better than others. Estimators can be compared with each other in terms of their properties. Two of the estimator properties that are of interest to us are that the estimator is unbiased and that is provides estimates that have minimum mean square error. This is given by the following conditions,

- The term, $\left| E[\hat{\Theta}] - X \right|$ tends to 0 as time approaches infinity and,
- $E[(\hat{\Theta} - X)^2] \leq E[(\hat{\Theta}' - X)^2]$ where, $\hat{\Theta}'$ is any other estimator.

There are several other properties that are desirable such as linearity, consistency, efficiency, etc. but they are not discussed here in detail for the sake of simplicity and continuity. In the following sections, we will discuss ways of realizing such estimators and the selection of the most appropriate choice.

2.2.1 Bayesian Estimation

In the context of decision and estimation theory, Baye's estimator is an estimator or decision rule that minimizes the a posteriori expected value of a loss function. Here, posterior refers to the new conditional probability assigned to the random variable being estimated in the light of new evidence (observations). The most common loss function used in Bayesian estimation is Mean Squared Error (MSE), also referred to as the squared error risk. MSE is given by,

$$E[(\hat{\Theta}(Y) - X)^2] \qquad \ldots(2.1)$$

An estimator is said to be Bayesian if it minimizes the loss function which, in our case is MSE. The estimator that minimizes the posterior expected loss, $E(MSE \mid X)$ for each x also minimizes Baye's risk and is therefore a Bayesian estimator. Using MSE as the loss function, the Baye's estimate is simply the mean of the posterior distribution, given by

$$\hat{\Theta}(Y) = E[X \mid Y] = \int X.p(X \mid Y)dX \quad \ldots(2.2)$$

Where, $p(X \mid Y)$ refers to the conditional probability. This is a Minimum Mean Squared Error (MMSE) estimator of *X*.

2.2.2 Batch Estimation

Batch estimation takes all measurements over a time period to solve all the unknown state variables at the same time. It allows us to find a maximum a posterior estimate, given by,

$$\arg\max.(p(Y \mid X)p(X)) \quad \ldots(2.3)$$

Where, $p(Y \mid X)$ is the observation likelihood and $p(X)$ is the prior probability.

There are several ways to do this and brief descriptions of some of the methods are given below. A more detailed review can be found in [13].

2.2.2.1 Projective Factorization

This technique [31, 32] comes from the area of Structure From Motion (SFM) estimation in the field of computer vision. This technique can also be used for the SLAM problem too [13]. The measurement model for projective factorization is given by,

$$Y = MS \quad \ldots(2.4)$$

Where, Y is the set of all measurements, M is the parameterized robot motion projection matrix and S contains the locations of the landmarks. The problem here is to find

matrices M and S, given Y. This can be performed using singular value decomposition. The details of this technique and its implementation are beyond the scope of this thesis and the interested reader is directed to the references given above.

This is a linear technique and does not require initial guess on the solution but the iterations are quite slow and the solutions cannot be guaranteed to be physically feasible. Due to this, a solution may not be optimal in the least squares sense. Besides, there is no easy way to incorporate odometry measurements into this model which can cause problems with the scale of the map.

2.2.2.2 Bundle Adjustment

This technique [33, 34] is also from SFM and it is widely used in feature based 3D reconstruction problems. Given a set of images from different viewpoints with feature points (landmarks), bundle adjustment can be used to solve the problem of refining the scene geometry (location of the features), finding the camera motion parameters and camera characteristics simultaneously. It basically translates to an optimization algorithm when there is noise associated with the measurements and it becomes a maximum likelihood estimator under the assumption that the error is zero-mean Gaussian.

Bundle adjustment requires an initial guess of the solution that is refined iteratively until convergence. The objective of the algorithm is to minimize the error, which is the difference between the actual projections of the feature points on the images and the predicted projection points based on the current solution. This difference is expressed as a sum of a set of non-linear functions. Minimization is achieved with the help of non-linear least squares algorithms. The Levenberg–Marquardt algorithm [35] is one such algorithm that is most commonly used for this purpose. Mathematically, if n 3D points are projected onto m images, let, $x_{i,j}$ represent the projection of the ith feature on the jth image and vectors a_j and b_i represent the parameters of the j^{th} camera and i^{th} feature respectively, then, bundle adjustment minimizes the projection error which is given by,

$$\min_{a,b} \sum_i \sum_j d(Q(a_j, b_i), x_{i,j})$$

…(2.5)

Where, Q is the predicted projection, x is the measured projection and d is the Euclidean distance between them.

This algorithm produces optimal solutions based on least squared error and the iterations are fast. It requires an initial guess which can lead to problems such as getting stuck in local minima when the initial guess does not fall into the basin of the globally optimal solution. Batch estimation, in general, suffers from the problem of computational complexity when m and n become high. This makes them unsuitable in applications where there are computational constraints. Recursive solutions can handle these problems better and are better suited for online implementations.

2.2.3 Recursive Bayesian Estimation

Recursive Bayesian estimation is a general probabilistic technique for the estimation of a probability density function recursively over time using the current measurements and a system process model. With recursive problem formulations, it is important that the formulation is computationally feasible as well as mathematically correct. Recursive estimation is the paradigm for processing observations as and when they are available and then, once they are processed, the measurements are thrown away and only the posterior probability distribution is stored. This is essential because most real-time systems require the state estimation process to be continuous since the estimates may be used for other processes such as feedback control, which happens to be our case.

Under some realistic assumptions, recursive estimators can provide as much information as their batch estimation counterparts. The assumption is that the distribution used by the filter be able to represent every possibility of the state given the observations so far. The assumption that the noise that corrupts the observed and unobserved random variable is zero-mean Gaussian is a very reasonable one. Under these assumptions, we can get the same performance at a lesser cost in terms of computation and memory requirements.

Besides, recursive estimation is well suited for Markov processes and, in our case the state evolution can be modelled as a Markov process.

2.2.4 Kalman Filtering

The Kalman filter is based on linear optimal control system theory [36]. Optimal control of dynamic systems refers to the operation of a dynamic system at a minimal cost. This is generally formulated as a linear feedback control law,

for a linear dynamic system, $\dot{X} = FX + BU$

find a control law, $U = -KX$

that minimizes the quadratic cost function, $J = \int_0^\infty (X(t)^T QX(t) + U(t)^T RU(t))dt$

…(2.6)

Where,

X,U – state and control vectors respectively.

F, B – state transition and controllability matrices respectively.

K – feedback factor or gain.

Q,R – weight matrices that determine K.

The control law that minimizes the cost function J is said to be optimal with respect to J, and this is called a Linear Quadratic Regulator (LQR). It should be noted here that U and X are vectors and hence represent a Multiple Input Multiple Output (MIMO) system. Interestingly, it is possible to model the state estimation process in the same way. In 1960, R.E. Kalman published his celebrated paper [37] describing a recursive solution to the estimation problem. He examined the problem of constructing a system state estimator that minimized the effects of the process noise and the sensor noise to produce the most likely estimate of the system state. He found that the estimator design process is the same as the LQR design process if the system process and the observations are corrupted by zero-mean additive white Gaussian noise. Thus the Kalman filter provides a

solid theoretical basis for the state estimation of a system with uncertain dynamics. Kalman filtering is a two step process. The first step is called prediction, where we predict the future state of the system based on the process model. The second step is called the update or correction, where the prediction is adjusted to fit the current observations (see Figure 5). For a detailed derivation of the Kalman filtering algorithm refer to [37, 38]. The details of the Kalman filter design for our problem are given in the next chapter.

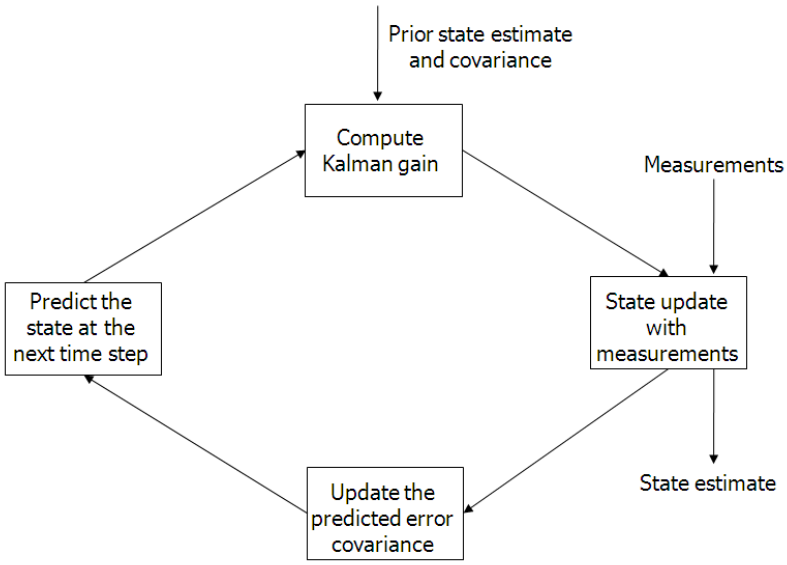


Figure 5 – Kalman filtering process.

The computations are made in the following way:

1. System state prediction at the next time step.

$$X_{k+1|k} = f.X_{k|k} + w_k$$

   Where, w_k is the process noise with intensity Q.

2. Error covariance prediction

$$P_{k+1|k} = f.P_k.f^T + Q$$

3. Compute Kalman Gain

$$K = P_{K+1|k}\left(h.P_{k+1|k}.h^T + R\right)^{-1}$$

Where, R represents the intensity of the measurement noise and H is the measurement matrix that relates the measurements to the system state.

4. System state update

$$X_{k+1|k+1} = X_{k+1|k} + K\left(Z_{k+1} - h.X_{k+1|k}\right)$$

5. Error covariance update

$$P_{k+1|k+1} = \left(I - K.h\right)P_{k+1|k}$$

…(2.7)

This process continues recursively.

2.2.5 Extended Kalman Filter

The Kalman filter discussed in the previous section makes some assumptions that are essential to ensure optimality. One of them is that the variable that form the state vector and observations have a linear relationship and the evolution of the state variables over time is also linear. In practice, this is hardly the case. In our case, the measurements and the state vector have a non-linear relationship. When non-linearity is present in a system, an extended Kalman filter (EKF) is used, which incorporates a suitable linear approximation. It has to be noted that the estimates produced by an EKF cannot be guaranteed to be optimal since an approximation is involved. Without a linear approximation, nothing can be said about the estimates. This is because, the Kalman filter produces optimal estimates based on the assumption that all variables have a Gaussian distribution. When an observation has a non-linear relationship with the system state, although the observation itself has a Gaussian error profile, the nonlinear relationship alters the Gaussian profile of the state variable that the observation relates to during the update or correction step. However, with a good approximation near optimal estimates can be produced. Linearization refers to finding a linear approximation of a function at a given point. Linearization can be performed using a Taylor series expansion by discarding the higher order terms or, we can also use the first derivative of the function. The latter approach is used here. The matrices *f* and *h* are the main sources of linearity in the system. The function *f* describes the temporal evolution of the system and *h* relates

the measurements to the system state. For example, if we consider the nonlinear function $y = f(x)$ plotted in Figure 6, let's say we want to linearize this function at the point $x = a$. Then the linear approximation is given by $f'(a)$ (see Figure 6 for an illustration). We perform the same operation on the matrices *f* and *h* and once we find the linear approximation, the filtering process is then the same as the standard linear Kalman filter. Since we are dealing with multiple variables, we use a Jacobian matrix and the linearization point is the predicted state of the system. The matrices after linearization are represented by,

$$F = \left.\frac{\partial f}{\partial x}\right|_{X_{k|k}}$$

$$H = \left.\frac{\partial h}{\partial x}\right|_{X_{k+1|k}}$$

…(2.8)

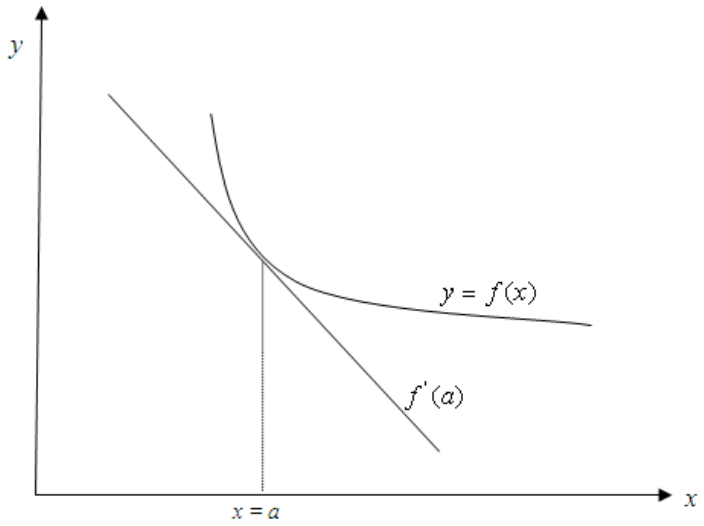


Figure 6 – Linearization.

2.2.6 Other Techniques

The EKF is a robust technique and is used in many applications but, in some circumstances it may not provide satisfactory performance. It is limited by assumptions

such as, linearity and the requirement of the error profile to be Gaussian. Other problems include computational complexity and the lack of provision in the filtering model to handle multiple measurements over time (one such problem is outlined in section 2.2.6.3). There are a few other specialized techniques that are used to tackle these problems. Some of them are discussed below.

2.2.6.1 Computational Efficiency

One of the problems associated with EKF-based SLAM is the computational complexity. Although it performs well, the iterations take more time to compute as the state vector grows as new landmarks are added to it. There have been some algorithms proposed to address this issue. The most significant of these include FastSLAM [39, 40] and SLAM with Conditionally Independent (CI) local maps [41, 42]. FastSLAM uses particle filters [43] in place of the EKF. Particle filters are also used as an alternative to the EKF and Unscented Kalman Filter (UKF) and they are the recursive counterparts of the Markov chain Monte Carlo batch estimation method. They have been known to produce results comparable to those produced by EKFs. A version of particle filters called the Rao-Blackwellized particle filter [44] that combines the particle filter and the EKF has been known to produce very good estimates and it also exhibits very good time efficiency [45]. In the case of bearing-only SLAM all of these methods need to be matched with a good landmark initialization and management scheme. Though particle filter based methods perform very well, the simplicity of the EKF based methods is hard to overlook. SLAM with CI-local maps is a method that performs just as well and it uses the EKF.

2.2.6.2 Unscented Kalman Filter

In theory, an Unscented Kalman Filter (UKF) [46, 47] performs the same operation as an EKF. The difference is that UKF performs much better than an EKF when the measurement and state transition models have functions that are highly non-linear. Instead of the derivative based linearization used by the EKF at the predicted state, the UKF uses a deterministic sampling technique referred to as the Unscented transform [48]

to compute a set of sample points on the in the vicinity of the predicted system state. These points then undergo a nonlinear transformation. The mean is given by the weighted average of the transformed points and the covariance is given by the weighted outer product of the transformed points [46]. Though the UKF performs well, we found that an EKF performs well enough in our case as it produced good estimates.

2.2.6.3 Robust Estimation Schemes

Recently, specialized algorithms have been developed for bearing-only SLAM. The algorithm proposed in [14] addresses all issued pertaining to bearing-only SLAM. The technique is based on the Variable State Dimension Filter (VSDF) [49] which is a nonlinear smoothing technique. This filter has the capability of updating the state vector using past measurements as well. The algorithm used a statistical linearization technique which has similar features as the Unscented transform and it performs well in the presence of non-linearity. This is also a recursive algorithm.

2.3 Motion Optimization and Control

Dynamic Programming (DP) algorithms can be used to obtain an optimal control policy for the robots. However, DP algorithms have the problem of computational complexity, referred to as the ‘curse of dimensionality’ by Bellman [50]. The computational complexity of the numerical solution to the DP problem increases exponentially as the size of the problem increases or in other words, as the state space grows [51]. Therefore, a closed form solution to the DP problem may not be feasible and practically, in some cases where the behaviour of the system is hard to predict an optimal solution may not be possible. Also, when a system being controlled is dynamic; planning needs to be on-line, taking into consideration the latest data as and when it becomes available. This limits the time available to compute the numerical solution. As a result, we need to strike a balance between the ease of implementation, available computational power and satisfactory performance. This generally means having to settle for a suboptimal control paradigm. Two popular paradigms used in this regard are certainty equivalent control and open loop

feedback control. Limited look ahead control is a technique that combines these to paradigms; Model Predictive Control (MPC) is one of the variants of limited look ahead control and when coupled with a receding horizon formulation fits best for the optimal control problem considered in this study. A feasible optimal trajectory (optimal with respect to a cost function) is generated from the current location of the robot to the expected future location over a finite time horizon of length, T. This reference trajectory is followed for a short time, $\delta<T$, and then, the reference trajectory is recalculated again based on the current parameters. The advantages of using the MPC approach is that, the controller becomes adaptive in nature and therefore, is well suited for robot control in unpredictable dynamic environments, which is our case. The changes in mission and the dynamic addition of constraints are also easily accommodated into the control loop. Our objective is to track moving objects and also to estimate the structure of objects. For this, the cost functions used in this research are Geometric Dilution Of Precision (GDOP) [3] and the Fisher Information Matrix (FIM) [52, 53] for motion tracking and structure estimation respectively. A mathematical description of this control structure is given in chapter 4. The following section contains an overview of the low level steering mechanisms and how it is integrated with the high level coordination and control.

2.3.1 Control Structure

The controller enables the robot to behave in a certain way depending on the mission and the environmental perception. It consists of a set of prioritized steering mechanisms, each one designed to accomplish a certain task or execute a manoeuvre. If required, two such mechanisms can be combined to create a more interesting behaviour based on the guidance from the supervisory controller. The block diagram in Figure 7 illustrates this. Some of the steering mechanisms include avoid obstacle, wander, follow target, etc. The implementation of the steering mechanisms have been inspired from the texts in [54] and [55].

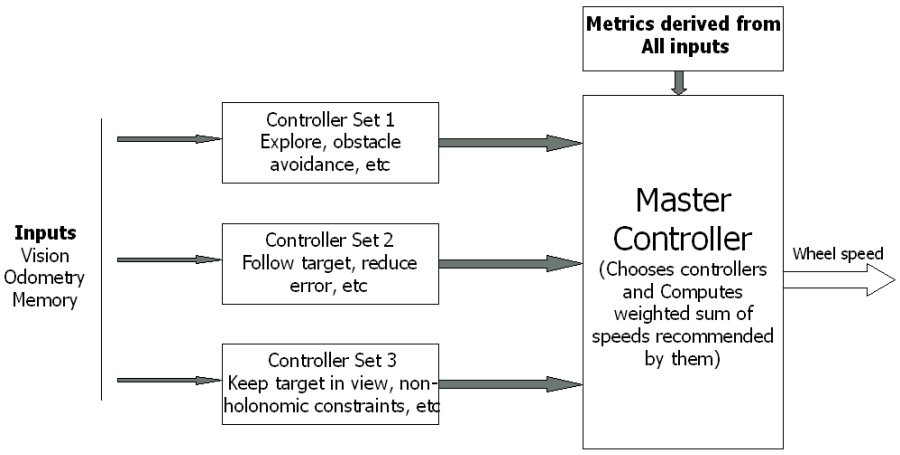


Figure 7 – Controller structure.

2.4 Summary

In this chapter we have established the theoretical foundations for the estimation process and the basis for motion optimization. Chapter 3 provides a detailed overview of the specific estimation process used and chapter 4 presents the mathematical modelling of the optimal control scheme.

Chapter 3

System Modelling

In the previous chapters, the rationale for doing this kind of work and the mathematical background for the modelling problems such as the one considered in this thesis were discussed. In this chapter, we discuss some of the specific implementation aspects and how the individual components come together from both the theoretical and practical perspectives.

Discussion centres on the design of the Extended Kalman Filters for the estimation of the parameters that form the system state vector. Two EKFs are developed, one that resides on the individual robots and one that resides on the supervisory controller. Synthetic data is used to validate the design of the filters and some of the results are presented. The EKF is used here as an estimation tool and, also, as a tool for data fusion. Here, data fusion refers to the fusion of data from the two sensors used; namely, bearing measurements from the monocular camera and odometry measurements from the wheel encoders. When bearing measurements to the landmarks alone are available, odometry measurements are very important. Although there have been instances where robot localization and mapping has been carried out without odometry [56] they require range measurements. Without odometry, the estimates will be out of scale. Therefore, the EKF is used to fuse the data from the two very different sensors and model them appropriately. The inclusion of data from sensors such as GPS receivers, magnetic compasses and inertial measurement units are not considered in this study. However, those sensors can also be readily included in this framework if required [38].

This study looks at the issue of gaining situation awareness with a cooperative of robots with limited sensing and, therefore, we also look at issues pertaining to the modelling of a scenario to test the proposed methodology, the autonomy of the individual robots and the information flow between the team members. Situation awareness measurement is a relatively new area of study spanning a little more than a decade, and team situation awareness is still in its infancy [57]. Studies that explore the factors affecting group decision making and performance are mostly tested on small artificially created teams performing simple tasks under controlled laboratory conditions [58]; a few studies have been conducted under more realistic conditions [59]. The data used in this work is synthetic, but it is generated using accurate models that resemble the real world very closely, and they are designed so that the factors affecting the decision making process are readily observed.

3.1 Development and Testing Scenario

The model used for testing consists of a closed 2×2m field with level surface, where the robots can roam around freely (see Figure 8). For the sake of simplicity obstacles have not been added in the area at present. Three Khepera II [60] robots are added to the field, two of those are used for surveillance and the other is the modelled moving target. The robot's sensor suite consists of a monocular colour camera, radio communication and eight IR range sensors. The IR range sensors have a limited range (~5cm) and, therefore, can be used only for local obstacle and wall avoidance. The camera is fixed and does not have any degrees of freedom of its own. It has a 45^0 field of view. The views from the robots can be simulated using the Webots simulation tool [61]. The robot designed to be the moving target has two coloured beacons (green and red) separated by a small distance to aid it's detection. They can also be used for the 3D structure estimation problem by looking at the beacons as the end points of a 3D line segment and then progressively move on to more complex structures such as in [17]. The existence of static landmarks is essential for the SLAM algorithm and, therefore, we have placed four white cubes at the four corners of the field. The image processing algorithm used is a simple colour based

thresholding algorithm designed to pick out white landmarks and the red and green beacons on the moving object. A supervisory controller has also been setup that can communicate with, and coordinate the actions of both robots. The supervisory controller can in principle, be set up on either or both of the robots, but the robots have very limited memory and processing power to accommodate the image processing. Therefore, for practical reasons, the supervisory controller resides on a separate computer with ample computational power and wireless link to the robots and the video stream. The image processing relating to an overhead camera, which is used to provide accurate georegistered information runs on the same computer (see section 3.2).

3.1.1 Simulation with Webots

Webots is a mobile robot simulation software. It has a VRML-based prototyping tool, which allows us to create virtual 3D worlds with physical properties. The model can include active objects such as robots and inactive objects such as walls. Robots can be configured to have a sensor suite, actuators and a locomotion model. The robots can be programmed as required. The robot supervisory controllers can be programmed in Java, C/C++ or MATLAB. In our case, the controllers have been written in C/C++ using the visual studio IDE. The Kalman filtering and image processing has been written in MATLAB and linked to these controllers such that they can be invoked as library functions. Webots also include numerous models of commercially available robots such as the Khepera II. It also has interfaces to connect to these robots so that once tested in simulation, the software can be directly transferred to the real robots. Figure 8 shows the top view of the simulation created for testing our algorithms.

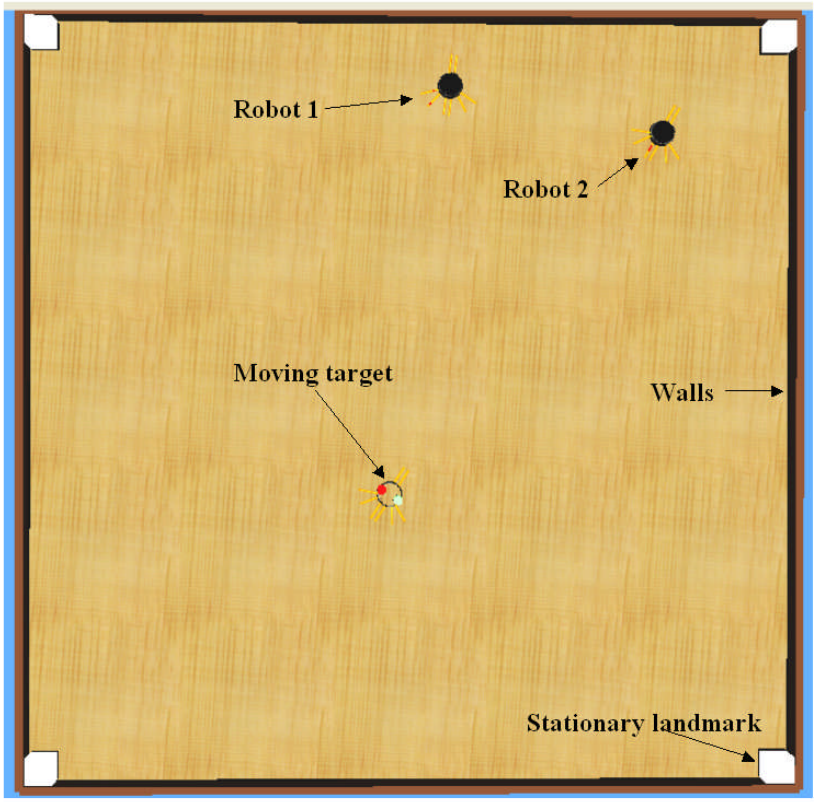


Figure 8 – Webots Model (2x2m area)

3.2 Simulated Indoor GPS for Ground Truth Verification

Although we only use synthetic data in this work, real-time testing is expected to be carried out on the 'Khepera' miniature mobile robots in due course as a continuation of this work. The test bed consists of a fully configurable robot playing field. Partitions can be put in and the set up can be changed to create different scenarios. Little cubes or blocks can be put in to simulate obstacles. The Khepera robots are also configurable. The base unit has a Motorola 68331 processor clocked at 25MHz with 512 KB of RAM and 512Kb flash memory. They are powered by the on-board batteries and are run by two brushed DC servomotors with differential steering using the PWM (Pulse Width Modulation) scheme. They is capable of fully autonomous operation. The add-on features include a Bluetooth wireless radio, wireless monocular colour video and a gripper for

moving objects. The robot command is through a Bluetooth wireless radio. Another setup is an overhead camera looking at the whole playing field. This is primarily used to find the orientation of robots and localize them in absolute terms, which in turn serves as the ground truth. The robots have a coloured triangular marker on top, which facilitates orientation calculation (see Figure 9 for a view from the overhead camera). In the simulated environment though, Webots have a simulated GPS feature which replaces the image based system used in the laboratory.



Figure 9 – A view from the overhead camera sharing the orientation and localization of the robots.

3.3 Kalman Filter for the Individual Robots

A mobile robot needs to be aware of its environment. One of the first things it needs to know is its current position, its orientation and, possibly, information on where it has been. SLAM allows a robot starting at an arbitrary position, and in an unknown environment, to progressively build a map of its environment while simultaneously using that map to find localization and orientation of the observing robots [10, 11].

As stated in section 2.2.4, a Kalman filter is basically a statistical estimation tool that is designed for modelling linear Markov chains[1] that are corrupted by white Gaussian noise. When there is non-linearity present in the model, i.e. the system state and the observables have a non-linear relationship, which is the case in our model, an Extended Kalman Filter (EKF) must be used. The EKF uses a linear approximation of the non-linear functions involved. The Kalman filter is well suited for the implementation of the SLAM system as the evolution of the system states (robot pose, target pose, etc.) can all be modelled as Markov chains. The EKF algorithm for our problem is described as follows,

[1] Markov chains refer to stochastic processes where the future state of the system depends only on the current state and not on any previous states.

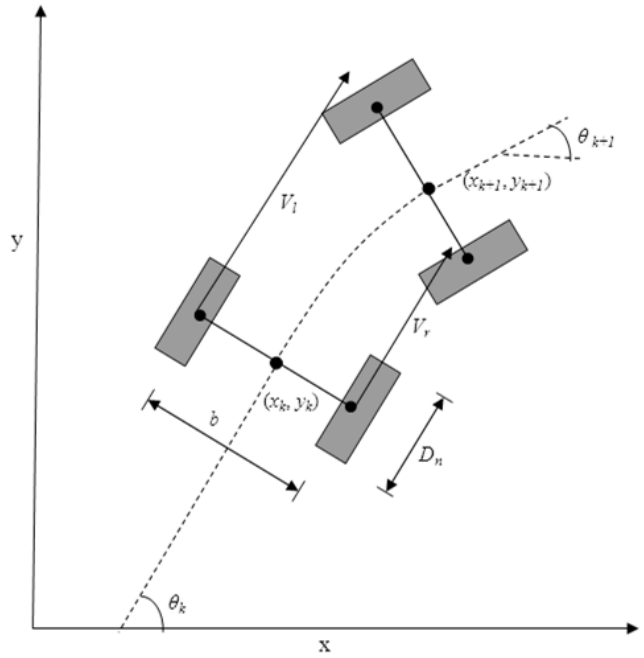


Figure 10 – Vehicle model.

The kinematic model for a differential drive planar vehicle (see Figure 10) is given by,

$$x_{k+1} = x_k + \left(\frac{V_r + V_l}{2}\right)\cos\theta_k$$

$$y_{k+1} = y_k + \left(\frac{V_r + V_l}{2}\right)\sin\theta_k$$

$$\theta_{k+1} = \theta_k + \left(\frac{V_r - V_l}{b}\right)$$

…(3.1)

Where,

Wheel displacement, $V_r = C_m.N_r$

$V_l = C_m.N_l$

x,y – location of the robot

θ – orientation of the robot

k – time-step

b – wheel axle length

$N_{l/r}$ – left and right wheel encoder values respectively

$C_m = \pi D_n / C_e$

D_n – wheel diameter

C_e – wheel encoder resolution

The EKF is a predictor-corrector system. It uses a system process model, which is based on the kinematic model, to predict the future state of the system given the current state of the system. One set of measurements is provided by the wheel encoders, which indicates how fast the wheels are spinning. This is modelled as a control input to the system. Based on the previous state of the system, the state transition matrix and the control inputs, the Kalman filter predicts the state of the system at the next time step. The problem with odometry is that the errors are cumulative and increase with time. Depending on the quality of the sensors and how accurately they have been modelled, it is only a matter of time before the measurements become meaningless. The system state vector and the state transition model is given by,

$$\text{State vector, } X = \left\{ \underbrace{x, \dot{x}, y, \dot{y}, \theta, \dot{\theta}}_{\text{Robot states}}, \underbrace{x_1, y_1 ..... x_4, y_4}_{\text{Landmarks}} \right\}^T$$

$$X_{k+1|k} = f(k, X_{k|k}, U_k) + w_k \qquad \ldots(3.2)$$

Where, f denotes the state transition matrix, U_k denotes the control inputs and w_k is the process noise.

Let,

$$\Delta V = \left(\frac{V_r + V_l}{2} \right) \qquad \ldots(3.3)$$

$$\Delta \theta = \left(\frac{V_r - V_l}{b} \right) \qquad \ldots(3.4)$$

$$U_k = \begin{bmatrix} \Delta V.\cos\theta_k \\ \Delta V.\sin\theta_k \\ \Delta\theta \end{bmatrix}$$

…(3.5)

The Jacobian of the state transition matrix (only the terms relating to *x,y,θ*) are,

$$F = \begin{bmatrix} 1 & 0 & -\Delta V.\sin\theta \\ 0 & 1 & \Delta V.\cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

…(3.6)

$$\text{Where, } F = \left.\frac{\partial f}{\partial x}\right|_{X_{k|k},U_k}$$

The predicted state and the associated covariance matrix given the current state of the system are given by,

$$X_{k+1|k} = F.X_{k|k} + w_k$$
$$P_{k+1|k} = F.P_k.F^T + Q$$

…(3.7)

Where, *Q denotes the covariance of the process noise* w_k

The system state estimation process will integrate the errors involved in the process and the odometry measurements over time. In order to keep the errors bounded, some form of absolute measurement is required. In our case we use the re-observed landmark bearing measurements from the monocular camera. The measurement model is given by,

The measurement at time *k*, $Z_k = h(k, X_{k|k}) + v_k$

When h is a non-linear function, Jacobian, $H = \left.\frac{\partial h}{\partial x}\right|_{X_{k+1|k}}$

Predicted measurement, $Z_{k+1|k} = H.X_{k+1|k}$

Where, v_k denotes the measurement noise with covariance *R*.

…(3.8)

The following are the steps involved in obtaining the bearing measurements and updating the EKF,

- Landmark extraction – The measurements used in our model are derived from the video stream from the robot's on-board camera. The vision algorithm is designed to pick out three different colours using a simple colour based thresholding scheme. The stationary landmarks are white and the two beacons on the moving target that are green and red. Upon invocation, the algorithm returns the centroid *(x,y)* of the colour patches on the image plane (see Figure 11). The *x* value is then used along with the robot pose, θ, and transformed into bearing measurements. The information on the moving target is not used locally by the robot but is forwarded to the supervisory controller (see Figure 12). The artificial landmarks and beacons can be replaced by real ones by modifying the feature extraction algorithm that is designed to extract different imagery on the video stream. For example, more sophisticated algorithms such as, SIFT[16, 62, 63] or SURF[17, 64] may be used.

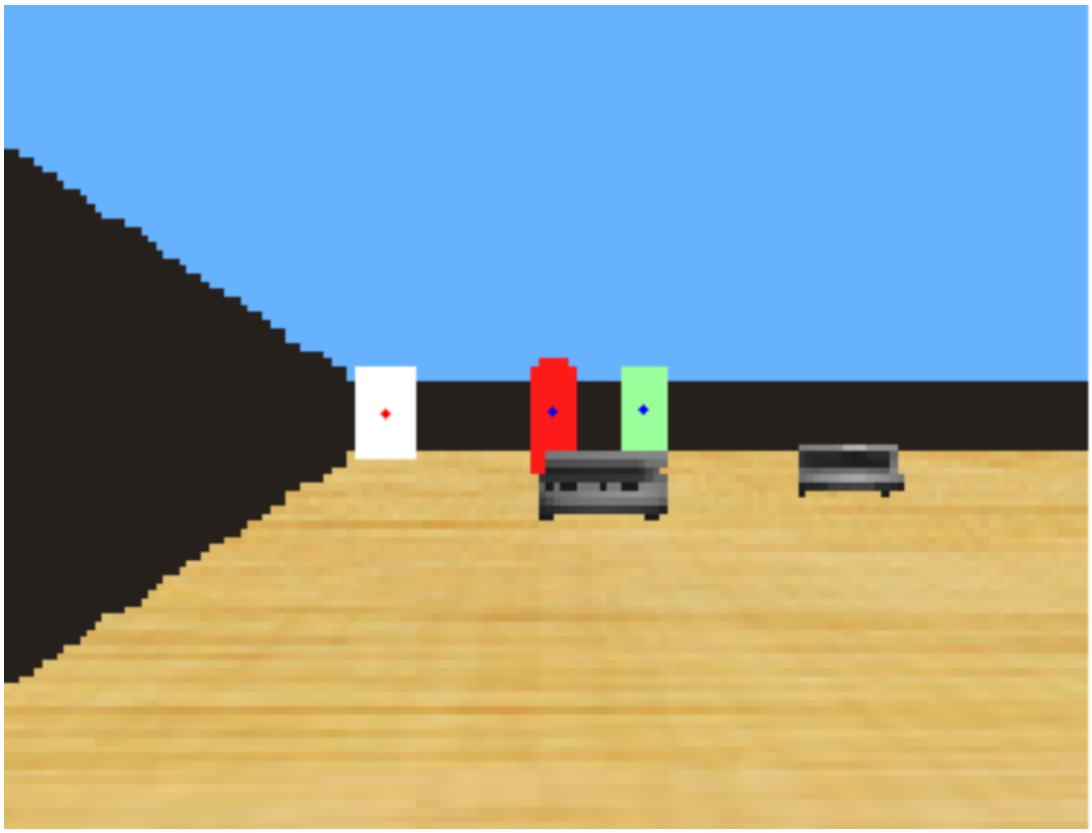


Figure 11 – Feature extraction (blue dots – moving object, red dot – landmark)

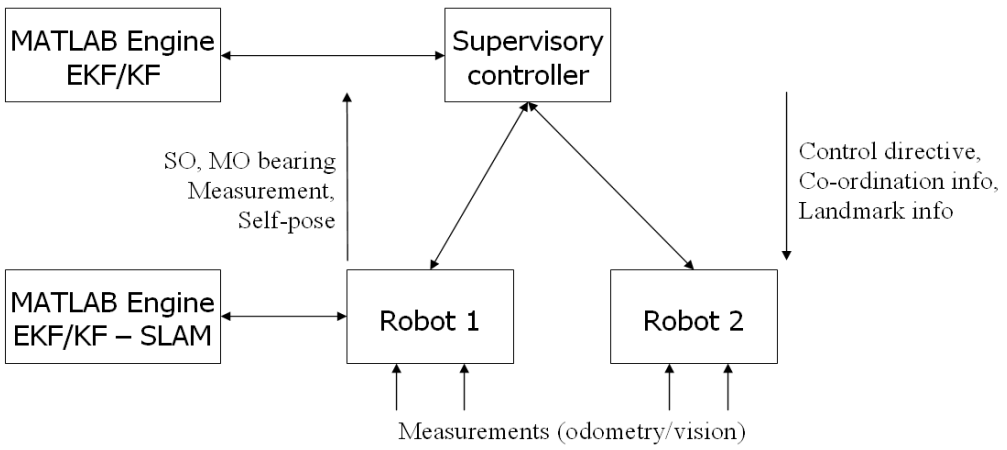


Figure 12 – Functional block diagram for information flow.

- Data association – There are four stationary landmarks which are all identical in shape, size and colour. Therefore, when a bearing measurement is observed, we need to determine which of the four landmarks it belongs to. Since we already have a state estimate, we use that to predict where the landmarks might be. Using the prediction and the measurement we can devise a similarity measure to match it to the landmark that gives the smallest error estimate. The measures used for this vary depending on the sensors used and the environment. In our case, we found that a simple nearest neighbour search works well. The equations relating to the landmark bearing measurements are,

$$\text{Bearing, } [\theta] = \left[\tan^{-1}\left(\frac{L_y - y}{L_x - x} \right) - \theta + v_k \right]$$

$$\text{Jacobian, } H = \left[\frac{(L_y - y)}{r^2} \quad \frac{-(L_x - x)}{r^2} \quad -1 \right]$$

Similarly, the columns corresponding to the landmarks will be,

$$= \left[\frac{-(L_y - y)}{r^2} \quad \frac{(L_x - x)}{r^2} \right]$$

…(3.9)

all other terms will be zero and, (L_x, L_y) represent the location of the corresponding landmark and, $r^2 = (L_x - x)^2 + (L_y - y)^2$

- State update with landmark measurements – This occurs when the robot has prior information on the location of the landmarks and then revisits the same landmarks. Based on the current state, (x,y,θ) the landmark location can be predicted and then compared with the measurement. This is called innovation and is used to update the system state and hence reduce the error. This assumes that the measurements are accurate and the associated uncertainty is less than the uncertainty introduced by the odometric measurements. The EKF update equations are,

$$\textit{Kalman filter gain, } K = P_{k+1|k}.H^T \left(H.P_{k+1|k}.H^T + R \right)^{-1}$$

...(3.10)

Updated system state and covariance matrices,

$$X_{k+1|k+1} = X_{k+1|k} + K\left(Z_{k+1} - H.X_{k+1|k}\right)$$
$$P_{k+1|k+1} = \left(I - K.H\right)P_{k+1|k}$$

…(3.11)

Figure 13 shows the trajectory estimate using odometry alone, and then the effect of bearing measurement updates on the estimation process.

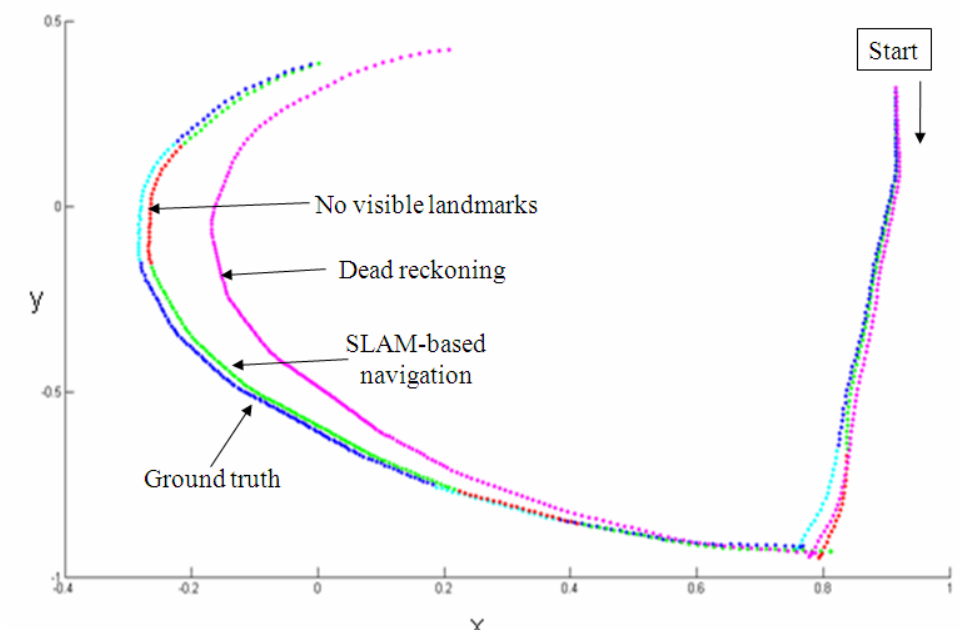


Figure 13 – Trajectory estimation using dead reckoning and SLAM along with ground truth. Magenta – Dead reckoning, Blue/Cyan – Ground truth, Green/Red – SLAM and Cyan/Red – Sections where no landmarks are seen.

3.4 Moving Object Tracking

In the scenario described here, the two robots have the same moving object in view, and corresponding points in the environment are known. The moving object track determines

the moving object's location at a given time and also find it's motion characteristics. In order to perform this, the pose of the two robots and the corresponding bearing measurements are required. This has been modelled in the form of an EKF performing both SLAM and moving object tracking on the supervisor computer, which can communicate with both robots and retrieve measurements from them. Since the two robots are in the same area, the static landmarks are common to both of them and we get two bearing measurements for the landmarks which helps reduce the error associated with them. With regards to moving object tracking, the moving object is modelled as a dynamic system with a constant velocity. The velocity is basically determined as the first derivative of the location with respect to time. Figure 14 shows the functional block diagram of the system

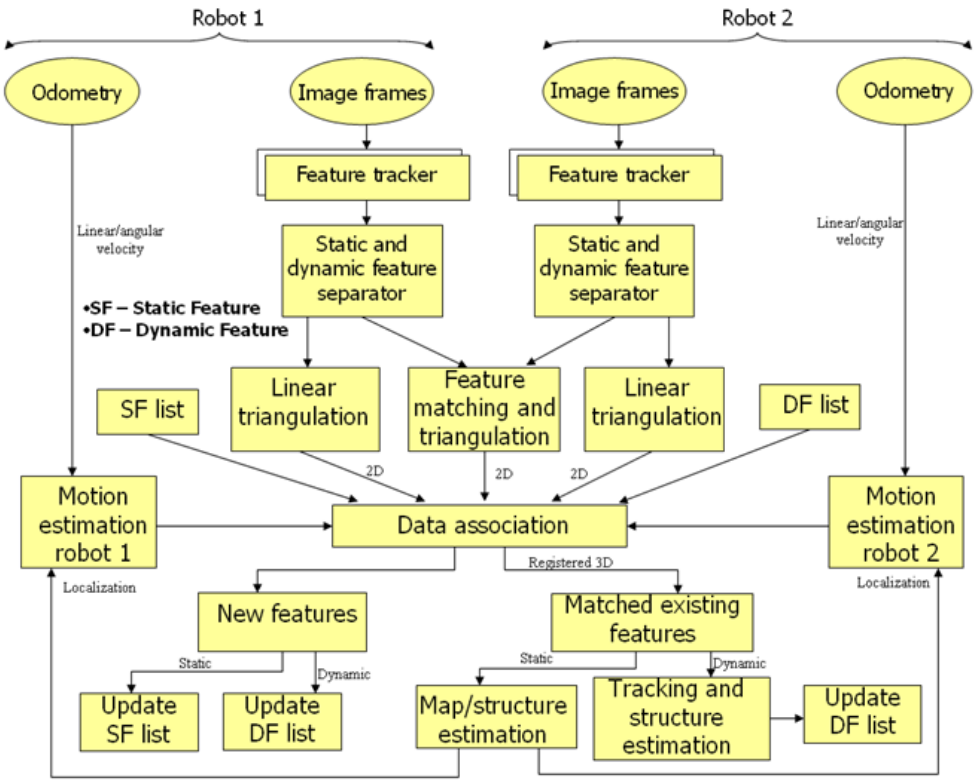


Figure 14 – Functional block diagram.

The reason for having an EKF performing SLAM in the individual robots is to give the robots a high level of autonomy. Even if the communication with the supervisor is lost, the robots can still keep track of their individual locations and wait for the communication to be restored. Since the supervisor should have better estimates of the system states, every 30 seconds it sends an update of the system states to the robots. The robots treat these updates as measurements and the state vector is updated accordingly. Figure 15 shows the information sharing scheme used by the supervisor. The EKF on the supervisory controller is an extended version of the EKF used by the robots. The state vector of the supervisory controller has been modified to include the states of both of the robots and the moving object. Therefore, the process model and the measurement model pertaining to the moving object tracking alone is given below.

State vector,

$$X = \left\{ \underbrace{x, \dot{x}, y, \dot{y}, \theta, \dot{\theta}}_{\text{Robot1}}, \underbrace{x, \dot{x}, y, \dot{y}, \theta, \dot{\theta}}_{\text{Robot 2}}, \underbrace{x_1, y_1 ..... x_4, y_4}_{\text{Landmarks}}, \underbrace{T_x, \dot{T}_x, T_y, \dot{T}_y}_{\text{Moving object}} \right\}^T$$

…(3.12)

Kinematic model of the moving target,

$$\begin{bmatrix} T_x \\ \dot{T}_x \\ T_y \\ \dot{T}_y \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_x \\ \dot{T}_x \\ T_y \\ \dot{T}_y \end{bmatrix}_k + v$$

…(3.13)

Robot pose, *R1 (x_1,y_1), R2 (x_2, y_2)* and target angle measurements *(θ_1, θ_2)*

Measurement model for moving object tracking

$$\tan\theta_1 = \frac{y_1 - T_y}{x_1 - T_x}$$

$$\tan\theta_2 = \frac{y_2 - T_y}{x_2 - T_x}$$

$$\begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = \begin{bmatrix}\tan^{-1}\left(\frac{y_1 - T_y}{x_1 - T_x}\right) + v \\ \tan^{-1}\left(\frac{y_2 - T_y}{x_2 - T_x}\right) + v\end{bmatrix}$$

Jacobian,

$$H = \begin{bmatrix}\frac{-(y_1 - T_y)}{r^2} & \frac{(x_1 - T_x)}{r^2} & \frac{(y_1 - T_y)}{r^2} & \frac{-(x_1 - T_x)}{r^2}\end{bmatrix}$$

…(3.14)

are the partial derivatives of θ_1 w.r.t x_1, y_1, T_x, T_y respectively.

Where, $r^2 = (x_1 - T_x)^2 + (y_1 - T_y)^2$

Similarly, there will be another row of partial derivatives for θ_2 as well.

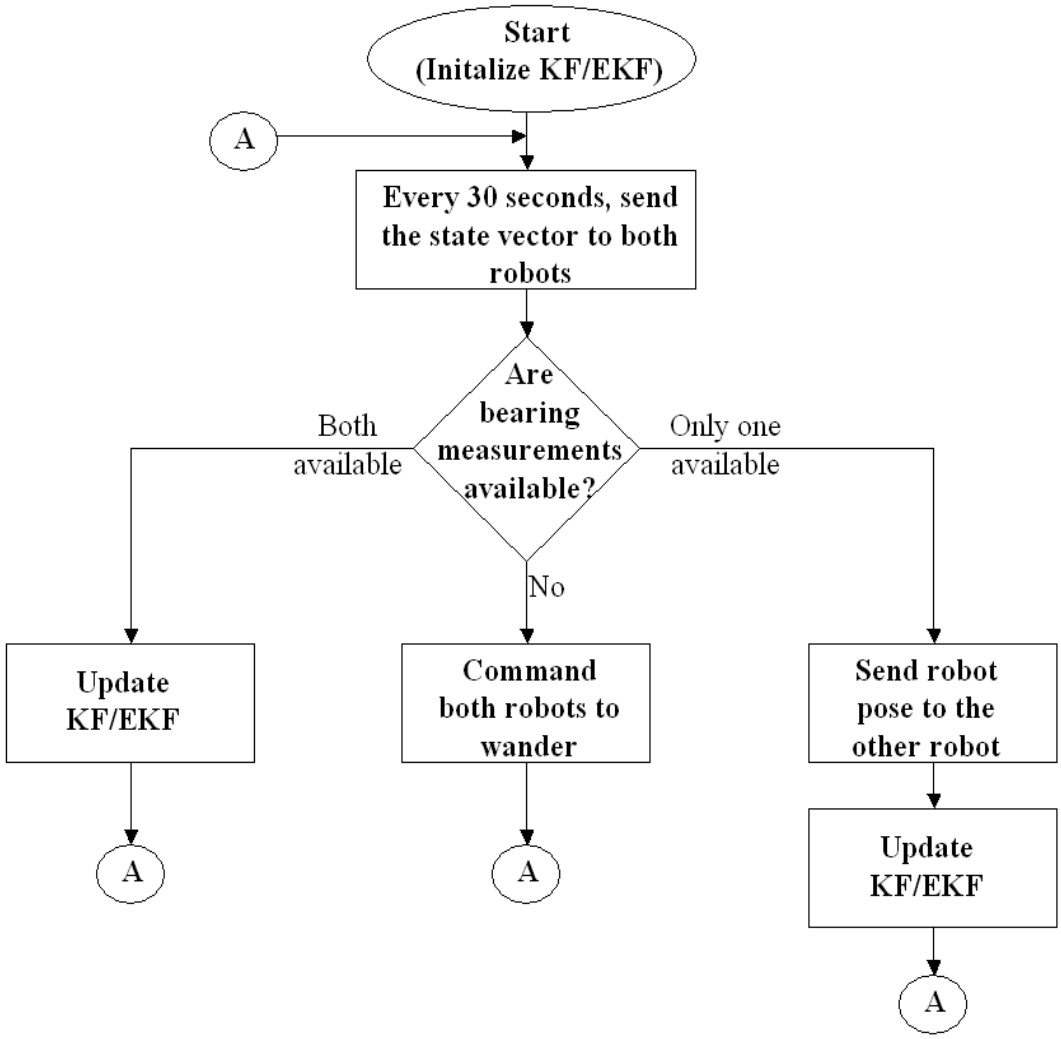


Figure 15 – Information sharing/update scheme on the supervisor.

Figure 16 shows the estimated trajectory of the two robots along with the estimated trajectory of the moving object. The ground truth of each is also provided for reference. This demonstrates that two robots with limited sensing can effectively coordinate their actions by exchanging relevant information to track moving objects. This also demonstrates that the robots can concurrently localize themselves using the landmark measurements and perform other operations required for situational awareness.

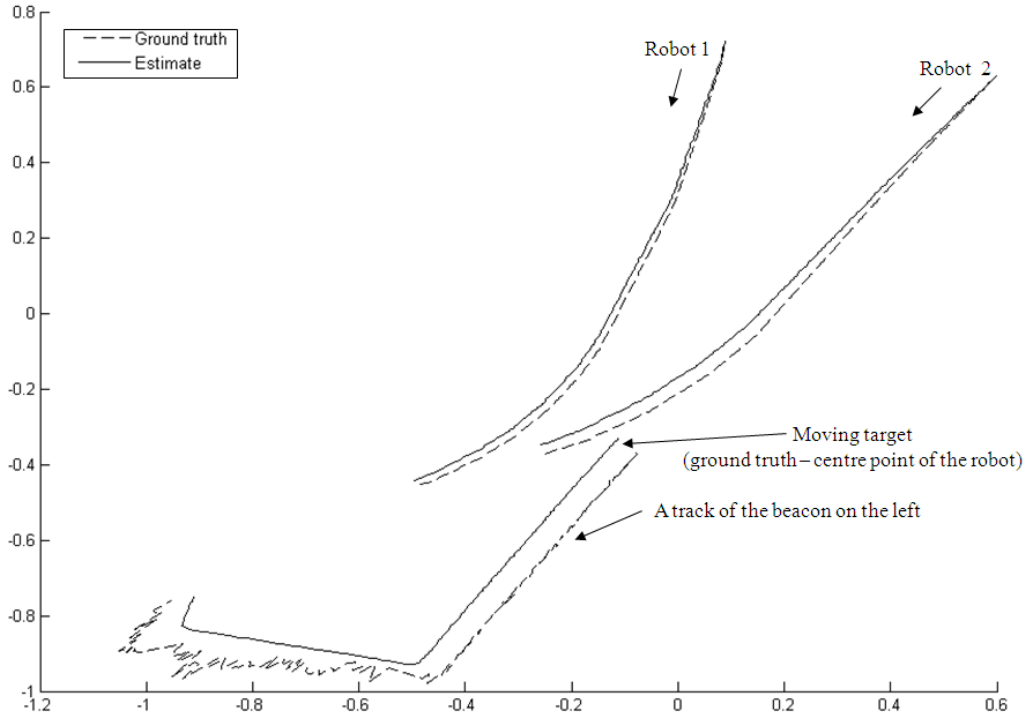


Figure 16 – Moving object track along with the trajectories of the robot. Green – Moving object track (estimated), Magenta - Overlaid moving object ground truth, Red – Ground truth for the robot trajectories and Blue – EKF SLAM estimation of the robot trajectories.

3.5 Optimal Sensor Placement

All the parameters estimated using the EKF have associated uncertainties. To locate the moving object we need measurements from the cameras and the locations of the robots. The uncertainties associated with the robot location and the various measurements may affect the estimated location of the moving object. Therefore we need to quantify the uncertainty as a function of the measurements and of the observing robot location. Geometric Dilution Of Precision (GDOP) is a measure used in localization systems to relate the localization accuracy to the measurement accuracy. It is used in GPS to find the accuracy of the localization as a function of the satellite geometry. GPS uses time of arrival measurements and converts them to range to perform the localization. Our system does not have any range measurements and is purely an Angle Of Arrival (AOA) system.

A mathematical expression for GDOP for an AOA system is given in equation 3.16. A more detailed explanation can be found in [3]. By looking at the pose of the two robots, their associated uncertainties, and the bearing measurement error, we can calculate the GDOP value for the location of the moving object (see Figure 17). For instance, when the robots are at more or less the same distance from the target, minimum error calls for a wider baseline and a relative angle tending to 90^0, but keeping the object in view is important to get a measurement in the first place. The wheel speed, therefore, should be a weighted sum of the speeds recommended by the two steering mechanisms, one for keeping the object in view and the other for movement towards a wider baseline. The algorithm will then predict the location of the moving object after unit time, and find the weights such that the object is still kept in view and simultaneously try to reduce the GDOP value.

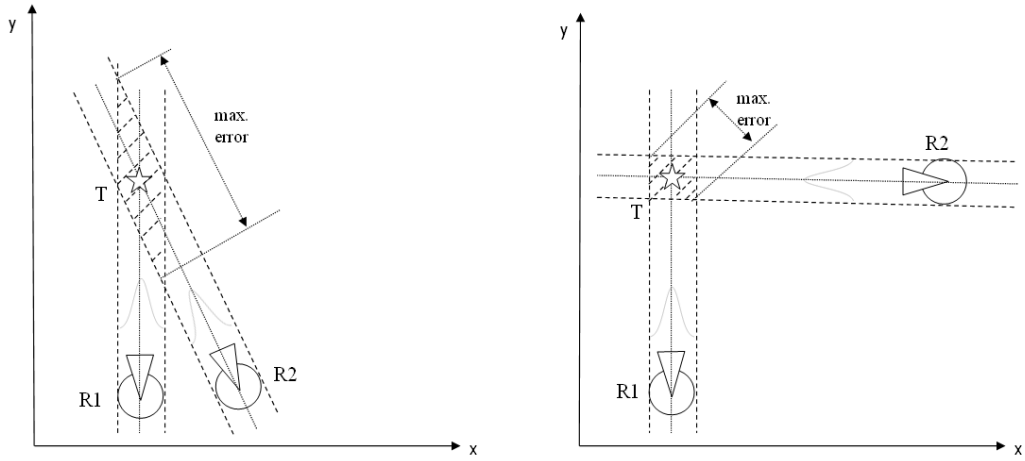


Figure 17 – Illustration of the effect of different observation points on localization error.

The robot pose and target location are represented as,

$$R1\ (x_1, y_1, \theta_1),\ R2\ (x_2, y_2, \theta_2),\ T\ (T_x, T_y)$$

θ_1 and θ_2 are the bearing measurement from robots *R1* and *R2* respectively,

$$\tan\theta_i = \frac{T_y - y_i}{T_x - x_i}$$

$$\begin{bmatrix} T_x \\ T_y \end{bmatrix} = \begin{bmatrix} \tan\theta_1 & -1 \\ \tan\theta_2 & -1 \end{bmatrix}^{-1} \begin{bmatrix} x_1 \tan\theta_1 & -y_1 \\ x_2 \tan\theta_2 & -y_2 \end{bmatrix}$$

…(3.15)

Let,

$$r = \sqrt{(T_x - x_i)^2 + (T_y - y_i)^2}$$

$$\begin{bmatrix} d\theta_1 \\ d\theta_2 \end{bmatrix} = \begin{bmatrix} \frac{-(T_y - y_1)}{r^2} & \frac{(T_x - x_1)}{r^2} \\ \frac{-(T_y - y_2)}{r^2} & \frac{(T_x - x_2)}{r^2} \end{bmatrix} \begin{bmatrix} dT_x \\ dT_y \end{bmatrix} = H. \begin{bmatrix} dT_x \\ dT_y \end{bmatrix}$$

$$GDOP = \sqrt{trace((H^T H)^{-1})}$$

…(3.16)

Figure 17 illustrates this concept but, in reality, the striped error regions correspond to range dependent ellipses as the errors derive from purely AOA sensors. The EKF has information regarding these ellipses in the form of the covariance matrices. Therefore, an information (inverse of covariance) theoretic approach could be used in place of a heuristic approach like GDOP since it fits readily into our framework. The Fisher information matrix [65] can be used for this purpose and is discussed in detail in the next chapter.

3.6 Summary

We have described an autonomous multi-vehicle robot cooperative that uses SLAM and that has the ability to track moving objects with very limited and constrained sensing capability. We have also discussed how a system like this can be constructed with a recursive estimation scheme along with the intra-team coordination features and demonstrated its basic functional performance. In the following chapter, we examine the issue of motion optimization with respect to the 3D structure estimation of the moving

object. Although we acknowledge that the ability of the system to handle more realistic landmarks and suitably managing them is non-trivial, it is not discussed here since it is not the focus of the present study. Detailed accounts relating to this further issue can found in [12, 66, 67].

Chapter 4

Coordination for Cooperative Structure Estimation

This chapter presents a multi-robot motion coordination strategy for the problem of estimating the location and 3D structure of a moving object. This can also be termed as ‘coordinated line-of-sight tracking and characterization’. The objective is to find the trajectory and structure of the moving object, and to optimize the motion of the observing robots towards maximizing the information gained by observing the moving object’s location and structure. We use a sparse 3D structure that consists of contrast-based interest points and their local descriptors to represent the moving object. The cost function used for the motion optimization problem is based on the Fisher Information Matrix (FIM). A predictive optimal reference trajectory that provides the maximum information gain is computed using the concept of model predictive control in the discrete time domain. A branch and bound algorithm is then used to search the solution space and produce optimal reference trajectories, and the closed loop control structure computes the control commands that drive the robots on the proposed optimal paths. The robots used are non-holonomic and equipped with monocular cameras that have a limited field of view and are fixed in the direction of travel. As a result, these constraints need to be embedded into the cost function. Also discussed is the integration of a high level coordination strategy with low level control mechanisms that facilitate reactive obstacle avoidance, keeping any object within the field of view, etc. The low level coordination schema is also used for relaying information between the robots and the base station

(where the supervisory controller resides) and is essential for the execution of the SLAM filters, as well as ensuring that the target/object of interest is located within the field of view of both robots prior to any cooperative tracking commencing.

4.1 Cost Function

4.1.1 Information Definition

The Fisher Information Matrix is named after its inventor, the statistician R. A. Fisher. It is a way of calculating the amount of information that an observable random variable carries about an unknown parameter upon which a likelihood function depends. It is used to determine the minimum error associated with the estimation of the random variable. If θ is the unknown parameter and X is the random variable, the Fisher Information is given by,

$$I(\theta) = E\left\{ \left[\frac{\partial}{\partial \theta} \ln\ f(X;\theta) \right]^2 \middle| \theta \right\}$$

…(4.1)

Which implies that $\theta \leq I(\theta) < \infty$. In other words, it is the expectation of the squared score of the likelihood function of θ, given X.

4.1.2 System Model

In our application, a location filter develops an estimate over time of the system states, which estimates the observer vehicle states and the target states. The location and velocity state transition model is given by,

$$X_k = F\, X_{k-1} + B_k\, U_k + w_k$$

…(4.2)

The measurement model is given by,

$$Y_k = HX_k + v_k$$

…(4.3)

Where,

X – State vector

F – State transition matrix

U – Control input vector

B – Controllability matrix

Y – Measurement vector

H – Measurement matrix

w, v represent additive white Gaussian noise with covariance, Q and R respectively.

During the process of state evolution, Q and R introduce an error in the estimation process, $e = X_k - \hat{X}_K$ (the difference between the predicted state and the actual state of the system) with a covariance matrix referred to as P_k. The inverse of this covariance matrix gives the information matrix.

4.1.3 Information based cost functions

The primary problem being dealt with here is co-operative surveillance. We also have a number of constraints, like limited field of view, which require a team of two or more robots to be tightly coordinated so that they track the same moving object in a manner that allows them to collaboratively determine its structure. The objective is to track a moving object, perform a characterization task (i.e. structure estimation) and, when a particular performance threshold is reached, store the collected information and move on to other objects of interest.

During the characterization process, we are interested in planning the paths of the observing robots based on a criterion to maximize the effectiveness of the operations. The

operation here refers to the state estimation process, where a Kalman Filter models the state evolution process and gives an estimate of the current state of the system along with the associated covariance matrix. Therefore, the FIM is a fitting measure for trajectory optimization. FIM has been used successfully to determine the optimal trajectory for the case of a single stationary target [65], and the coordinated control of multiple robots for the same purpose [68, 69]. It has also been used for the case where there are multiple moving objects [53]. Our case involves a single moving target, but we have the added issue of structure estimation. The main contribution of this study, therefore, is the development of a FIM-based cost function that takes into account the structure of the object and the constraint of having bearing-only measurements. The previous chapter discussed in detail the state estimation process. This chapter describes the criteria and cost function used for trajectory optimization. We then look at a solution for the optimized control problem. It should be noted that both are tightly coupled to the state estimation process described in the previous chapter.

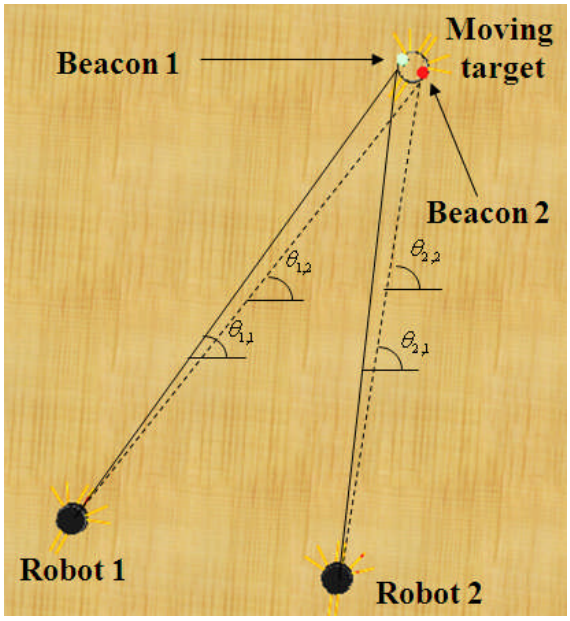


Figure 18 – Sensor-target geometry with the measured angles.

The test scenario (Figure 18) involves a team of two observing robots equipped with a monocular camera locating a moving target with two coloured markers to facilitate the tracking process. These coloured markers represent the end points of a 3D line segment, the structure of which we wish to estimate. Since two points on a line define it completely, the spatiotemporal evolution of these two points is what we need to estimate. The rationale for this is that more complex objects can be modelled using multiple points to provide a mesh-like representation. To keep the mathematics simple, we assume that all the movement is confined to a 2D plane. The extension to 3D movement is a straight forward process, where, both azimuth and elevation measurements are made independently to determine *(x,y,z)* instead of just the azimuth measurements to find *(x,y).*

Let the location of the two points be (x_{T1}, y_{T1}) and (x_{T2}, y_{T2}) such that these are points on a rigid moving target and hence their location and motion are correlated. In other words, a change in x_{T1} directly affects x_{T2} and the same goes for the y-coordinates. If there are n such points, on a known structure, one point would completely determine all others. In fact, the x-coordinates and y-coordinates would separately fit into a multivariable normal distribution. This is based on knowledge that a set of random variables $\{x_1, x_1, \ldots x_n\}$ follow a multivariable normal distribution if any linear combination of them are also normally distributed. Interestingly, the FIM for a set of random variables that follow a multivariate normal distribution takes a special form for a set of random variables $\{x_1, x_1, \ldots x_n\}$ that has an estimated of a set of parameters $\{\theta_1, \theta_1, \ldots \theta_n\}$[70]. The $(m,n)^{th}$ element of the $n \times n$ FIM is then given by,

$$I_{m,n} = \frac{\partial x}{\partial \theta_n} P^{-1} \frac{\partial x^T}{\partial \theta_n} + \frac{1}{2} \, trace \left(P^{-1} \frac{\partial P}{\partial \theta_m} P^{-1} \frac{\partial P}{\partial \theta_n} \right)$$

…(4.4)

Where,

$$\frac{\partial x}{\partial \theta_m} = \left[\frac{\partial x_1}{\partial \theta_m} \quad \frac{\partial x_2}{\partial \theta_m} \quad \cdots \quad \frac{\partial x_n}{\partial \theta_m} \right]$$

$$\frac{\partial P}{\partial \theta_m} = \begin{bmatrix} \frac{\partial P_{1,1}}{\partial \theta_m} & \frac{\partial P_{1,2}}{\partial \theta_m} & \cdots & \frac{\partial P_{1,n}}{\partial \theta_m} \\ \frac{\partial P_{2,1}}{\partial \theta_m} & \frac{\partial P_{2,2}}{\partial \theta_m} & \cdots & \frac{\partial P_{2,n}}{\partial \theta_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial P_{n,1}}{\partial \theta_m} & \frac{\partial P_{n,1}}{\partial \theta_m} & \cdots & \frac{\partial P_{n,n}}{\partial \theta_m} \end{bmatrix}$$

The covariance matrix *P* happens to be a constant derived from the state estimation process. In this case, the FIM reduces to

$$I_{m,n} = \frac{\partial x}{\partial \theta_m} \, P^{-1} \, \frac{\partial x}{\partial \theta_n}^{T}$$

…(4.5)

The elements of the matrix, $\partial x/\partial \theta$ reduce to unity because,

$$x_n = a.\theta_m + b$$

Where, *a,b* are constants (and *a=1* in our case). Therefore, $\frac{\partial x_n}{\partial \theta_m} = 1$.

Now we need to derive an estimate for the covariance matrix, *P*. Typically information is lost due to the dynamics of the system, its propagation and the process noise. Similarly, we gain information from the sensors and whenever measurements are made, although as information is added we must take into account the uncertainty associated with the sensor measurements. For the estimation of a single point target, two (bearing-only) sensor

measurements are required. Since we are tracking two points, typically we will require four bearing-only measurements. However, for simplicity, in the following equations we just show the procedure involved in the estimation of a single point, and then use a matrix representation that handles multiple points. The measurement,

$$Y_k = \begin{bmatrix} \theta_{i,1} \\ \theta_{i,2} \end{bmatrix}_K + \nu_k$$

$$= \begin{bmatrix} \tan^{-1} & \left(\dfrac{y_1 - y_{T1}}{x_1 - x_{T1}} \right) \\ \tan^{-1} & \left(\dfrac{y_2 - y_{T1}}{x_2 - x_{T1}} \right) \end{bmatrix} + \nu_K$$

…(4.6)

The measurement matrix, *H*, relates the measurement to the target state vector (x_T). It is given by,

$$H_k = \begin{bmatrix} \dfrac{\partial \theta_{1,1}}{\partial x_{T,1}} & \dfrac{\partial \theta_{1,1}}{\partial y_{T,1}} \\ \dfrac{\partial \theta_{1,2}}{\partial x_{T,1}} & \dfrac{\partial \theta_{1,2}}{\partial y_{T,1}} \end{bmatrix}_{\hat{X}_T}$$

…(4.7)

$\hat{X}_T$ Represents the predicted state and is used as the linearization point. The error associated with the measurements is given by,

$$R = \begin{bmatrix} \sigma_{\theta_{1,1}}{}^2 & 0 \\ 0 & \sigma_{\theta_{1,2}}{}^2 \end{bmatrix}$$

…(4.8)

The measurement error manifests itself on the state vector. The relationship between the measurements and the state vector is given by the measurement matrix. Therefore the effect of the measurement error on the state vector is given by,

$$FIM = H^T R^{-1} H \quad \ldots(4.9)$$

R^{-1} represents the inverse of error and hence the equation is transformed to the information domain.

We now have a system that evolves over time such that this evolution is modelled by the state transition matrices. Consequently, the evolution of information can be written as,

$$FIM = F_{k|k+1}{}^T H_k{}^T R^{-1} H_k F_{k|k+1} + Q^{-1} \quad \ldots(4.10)$$

This kind of representation can be used to model static as well as dynamic targets. In the case of a static target F_k will reduce to an identity matrix.

4.1.4 Nonlinear Model Predictive Control

The optional control problem at hand fits the Nonlinear Model Predictive Control (NMPC) [71] paradigm. NMPC is a control algorithm that uses a system process model along with the control input history. It also uses an optimization cost function over the receding prediction horizon. It is an iterative scheme that samples the system state at the current time,*'t'* and a cost-minimisation strategy is computed over a short time horizon in the future, t_h. Calculations are performed at each time-step, and an optimal cost-minimizing control sequence is then computed until time, $t + t_h$. The first control command alone is used and, at the next time time-step, the system state is re-sampled and the whole process repeated. This results in a fresh control command sequence. The final control sequence resulting from this approach is not optimal, but it has been shown to give near-optimal results [72]. It should also be noted that, since the control is based on

the predictions made by the system process model, the predictions cannot be assumed to be accurate due to the unpredictable nature of the dynamic systems that have been modelled. Therefore, the expectation of producing an optimal control sequence is unrealistic. Figure 19 gives a bit of a perspective into the overall control architecture. It shows the variables and parameters used and how they are fed into the various subsystems and how each subsystem is linked.

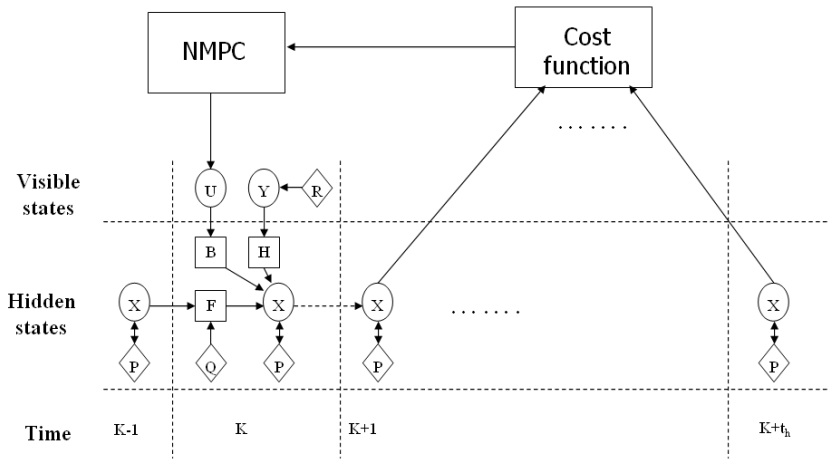


Figure 19 – Control and estimation structure.

Returning to the cost function, Fisher Information actually repeats the variance of a score. If '*n*' independent and identically distributed samples are drawn, the Fisher Information is '*n*' times the individual Fisher Information of each sample [73]. Consequently, The FIM based information cost over the discrete time horizon will be a summation that is given by,

$$\mathrm{FIM} = \sum_{\mathrm{K}=\mathrm{t}}^{\mathrm{t}+\mathrm{t_h}} \left[\mathrm{F}_{\mathrm{K|K+1}}^{T} \mathrm{H}_{\mathrm{K}}^{T} R^{-1} H_{K} F_{K|K+1} + Q^{-1} \right] \qquad \ldots(4.11)$$

Equation 4.11 holds good for tracking a single point. When more than one point needs to be tracked, equation 4.11 needs to be combined with equation 4.5. It should be noted here that the *x* and *y* coordinates follow the multivariate normal distribution separately. Although, both *x* and *y* coordinates can jointly follow the multivariate normal

distribution, this cannot always be assumed. Counter-intuitively, while the target is moving along the *x*-axis, there would be no change in the y-coordinates, making x and y independent. There can be other scenarios where they are dependent. Therefore, it is best to treat them as independent all the time. The cost function now becomes,

$$FIM = \sum_{k=t}^{t+th} \frac{\partial X}{\partial \theta} F_{k/k+1}^{T} H_k^{T} R^{-1} H_k F_{k/k+1} \frac{\partial X}{\partial \theta}^{T} + \theta$$

…(4.12)

Care should be taken to treat the x and y coordinates separately. This can be done either by carefully designing the $\partial X/\partial \theta$ matrix or by splitting the matrices and then perform the calculations and appending them later.

4.1.5 Cost function significance

Looking at the R and H matrices in detail,

$$\mathrm{R} = \begin{bmatrix} \sigma_{i,1,}^2 & 0 \\ 0 & \sigma_{i,2,}^2 \end{bmatrix}$$

$$\mathrm{R}^{-1} = \frac{1}{\sigma_{i1,}^2 \sigma_{i1,}^2} \begin{bmatrix} \sigma_{i2,}^2 & 0 \\ 0 & \sigma_{i1,} \end{bmatrix}$$

... (4.13)

The first row of the H matrix,

$$= \begin{bmatrix} \frac{Y_1 - Y_{T1}}{R^2} & -\frac{X_1 - X_{T1}}{R^2} \end{bmatrix}$$

...(4.14)

Where $R^2 = (X_1 - X_{T1})^2 + (Y_1 - Y_{T1})^2$ which represents the distance between the observing vehicle and the target point.

The effect of sensor–target geometry on localization accuracy using bearing only sensors have been analysed many times before [3, 74, 75]. Furthermore, it has been shown in [76] that simple heuristics such as equal angular separation are not a good approximation when the target range is not the same from each sensor. Therefore, optimal sensor

placement depends heavily on the target range and not just the relative angle between the sensors with respect to the target. As can be seen in equation (4.14) the cost function takes into account the target range as well. Similarly, it has been shown that an information based cost function used with different estimation methods [77], as well as other types of sensors performs well [68, 78]. It has also been shown that the methodology used here accurately models the information gathering process [53]. Hence, it can be argued that the information based cost function developed herein is appropriate for the optimization problem under consideration. The final scalar evaluation of the cost function is given by the determinant of the FIM.

$$\text{Cost,} \quad I = \underbrace{\left|FIM_t\right|}_{\text{current cost}} + \underbrace{\left|FIM_{t-th}\right|}_{\text{cost over the horizon}} \qquad \ldots(4.15)$$

The trace and its corresponding values can also be used for more complex evaluations, although the determinant suffices for our problem.

4.2 Constraints

The previous section dealt with the development of an information based cost function for the task of guiding the robots through information rich paths. However, we have a system that has a number of constraints that might stand in the way of taking those information rich paths. The major constraints in our system include the limited field of view of the camera that generates the bearing measurements, the non-holonomic nature of control of the robots, and the presence of obstacles such as walls and the other robots. The optimization cost function needs to be modified suitably such that these constraints are taken into account. We do this by adding terms to the cost function that increase the cost if the path taken would cause the robots to lose the target from the on-board camera's field of view and same goes for the case where the path goes through obstacles, walls and other robots. The non-holonomic constraint though is modelled through the state transition model used for the progression in time.

4.2.1 Obstacles

In our problem, we have static obstacles which are the walls/stationary landmarks and a dynamic obstacle, which is a robot's team mate. Each one is modelled slightly differently but the objective is the same. Static obstacles are modelled using Gaussian smoothing over the obstacle locations and dynamic obstacles are modelled using point spread functions.

4.2.1.1 Static obstacles

As discussed in section 1.3, we have assumed a pre-initialized system wherein the robot knows the location of the four static landmarks and the location of the walls. This information is available to the robots in the form of a 100 × 100 matrix which corresponds to their environment where the walls and landmarks are represented as a high value (1) and open spaces have a low value (0). The 100 × 100 matrix is shown in the form of the image in Figure 20. However, this information needs to be transformed into a metric that would represent a cost for visiting a given location. This typically needs to be very high where the obstacles are and progressively diminish at locations further away. This requires the introduction of a mechanism which allows a trade off between traversing through information rich paths and getting too close to obstacles (or risk hitting them). In our case, this would mean letting the low level obstacle avoidance scheme take over. We have achieved this by using a 2D Gaussian image kernel (similar to the one shown in Figure 22) to convolve over the 100 × 100 environment representation matrix. The matrix after convolution looks like the one shown in Figure 21. Moreover, this more closely resembles the probabilistic navigation maps [79] generated by robots and allows them to manoeuvre along trajectories.

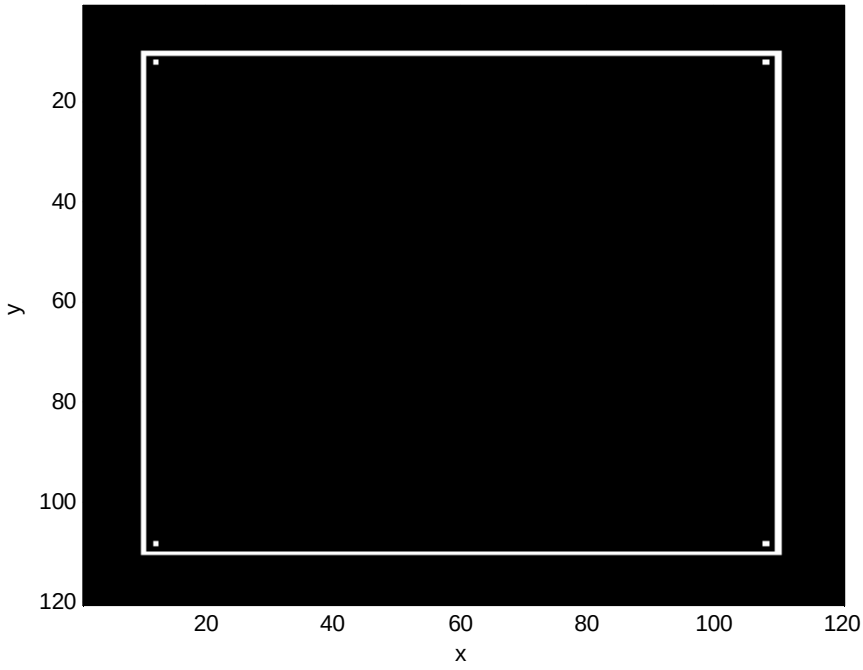


Figure 20 – Representation of obstacles.

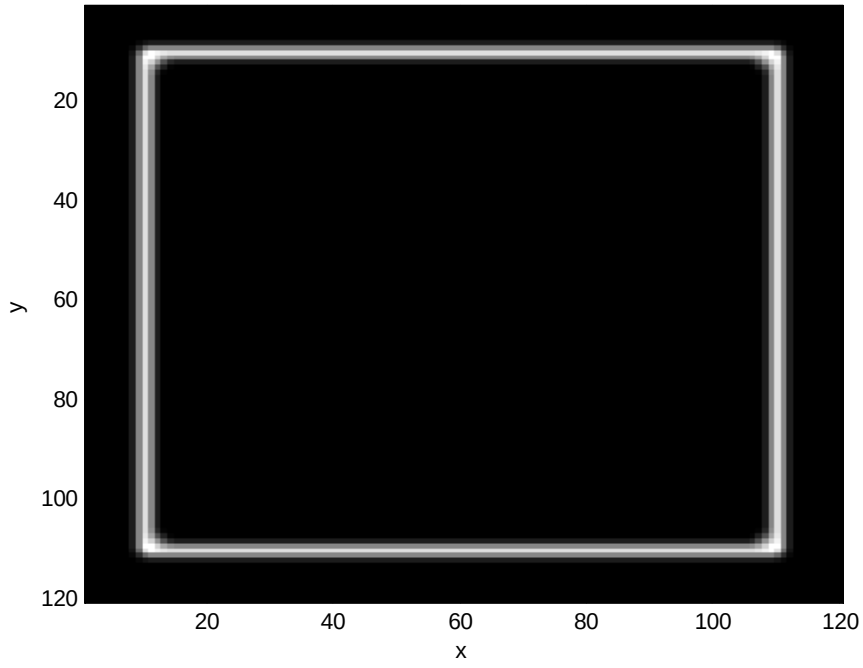


Figure 21 – The matrix after convolution

4.2.1.2 Dynamic obstacles

Dynamic obstacles include both the moving targets and any other robots that form part of the pursuit team. It is therefore important that the location of these objects at a given time be taken into account for the optimal trajectory generation. In the case of the moving target it is important that the observing robots do not get too close since it might be hostile. Similarly, although as the robots get closer they collect more information (the property a of bearing-only sensor) the effective target area covered by the sensor decreases, which means that some tracked features may fall outside the field of view. This would detract from the information gain. In regard to other robots in the team, trajectory de-confliction is important so that two members of the same team do not compete for the same space thereby avoiding the possible interference of low level control possible. This is performed using 2D discrete Point Spread Functions (PSF). These are basically matrices similar to the one shown in Figure 22. PSFs are generally used in imaging systems to model the response of the system to point sources. In more general terms, it can be defined as the impulse response of a system. In our constraint modelling, the robots and the target are approximated as point objects while in reality, they are not. Therefore, the constraint metric needs to take this into account and a Gaussian point spread function is used at the expected location of the objects at a given time *t*. This is simply added to the matrix shown in Figure 21. The PSF is generated in MATLAB using the following function,

$$h_g(n_1, n_2) = e^{-(n_1^2+n_2^2)/(2\sigma^2)}$$

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n_1}\sum_{n_2} h_g(n_1, n_2)}$$

…(4.16)

Where,

n_1,n_2 – Kernel size

σ – Standard deviation

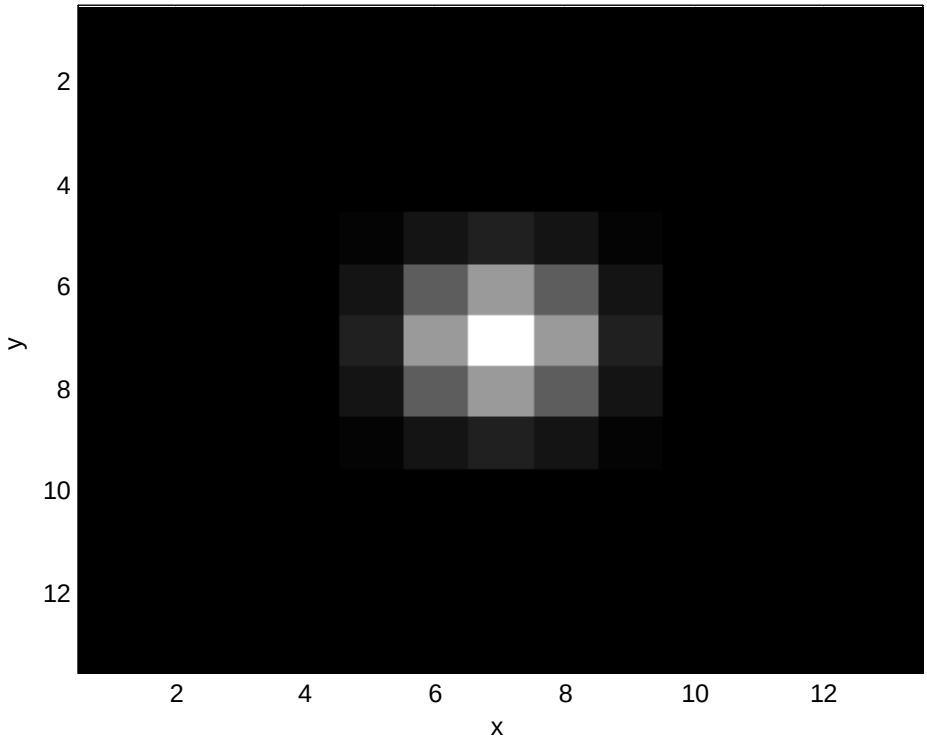


Figure 22 – A discrete 2D point spread function of size *13×13* and $\sigma = 1$.

4.2.2 Field of view

The cameras used by the robots have a limited field of view of only 45^0 (Figure 23). It is important that the target is kept in this field so that measurements are possible. To do this we add an exponential term to the cost function. This is because we want this term to have very little effect when the projection of the target falls near the centre of the field of view and increase rapidly as the projection starts to move towards the edges of the field. This fits an exponential profile and hence the choice. The measurements from the camera fall within the range, $-22.5^0 \leq \theta \leq 22.5^0$ (Figure 23). The measurements are first normalized to fit into the range, *[-1,1]*. The function used is actually a double exponential function. The reason for this can be seen in Figure 24 and Figure 25. There are several advantages in using a double exponential function. The function fits the measurement domain without any need for modification, there also is control over the maximum value

that this function can return at the extremities of *x* through the variable *A* and the flatness of the curve when *x* is small (close to 0) is ideal since the information based cost would get complete preference. When multiple points on the target are tracked, the centroid of these points is taken into account for this calculation representing the target as a point feature. The function is given by,

$$J_{FOV} = \begin{cases} A^{x^x} - 1 & -22.5^0 \le \theta \le 22.5^0 \\ high & otherwise \end{cases} \quad \ldots(4.17)$$

Where, *x* represents the normalized angle measurement and *A* can be varied as required to limit the range. This is adjusted based on the evaluation of the information cost such that the value of the function is high enough to effect a change in the trajectory. The minimum value of the double exponential is *1* at *x=0*. To make this zero, *-1* is added. This is because the function needs to have as little effect as possible when the target is well within the field of view.

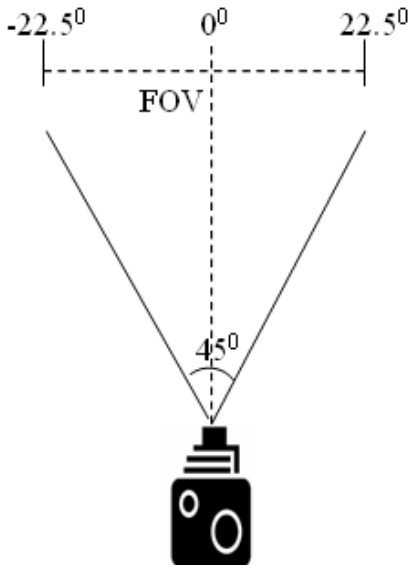


Figure 23 – Camera field of view and measurement range.

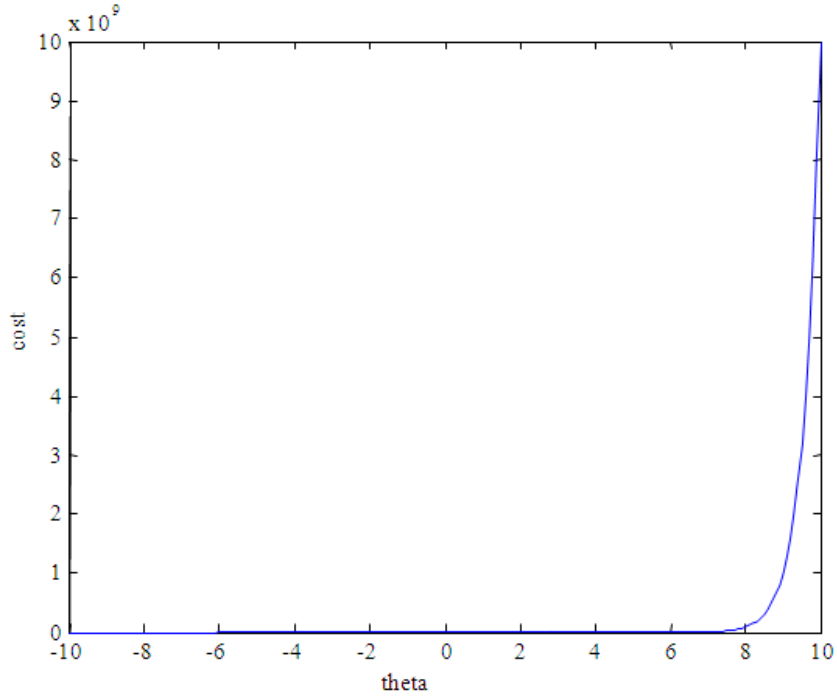


Figure 24 – A plot of the function 10^x for *x[-10,10]*.

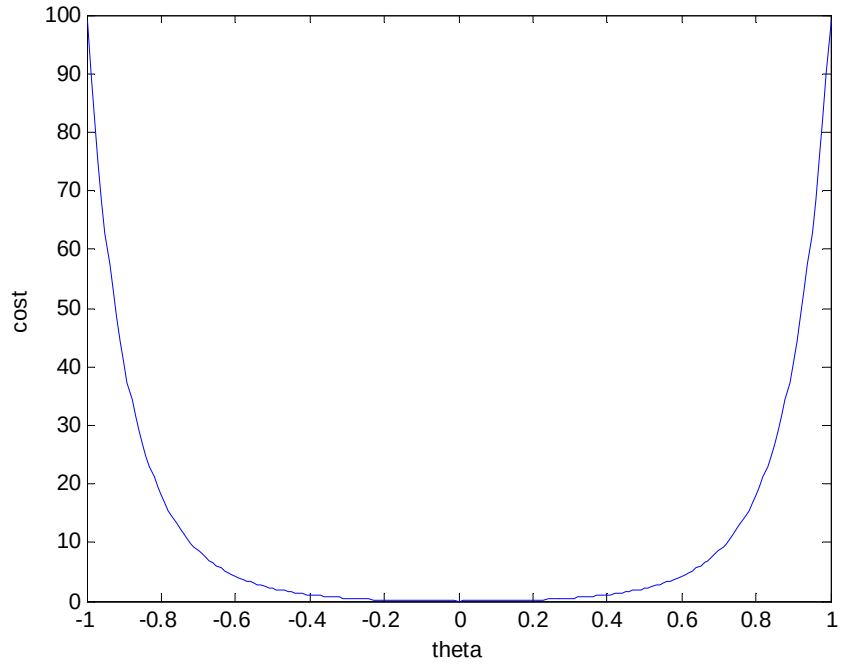


Figure 25 – A plot of the double exponential function with *A = 100* and *x[-1,1]*.

4.2.3 Velocity and Acceleration

As discussed before, the robots have a differential drive where each of the two wheels are run by two motors. The motors have Pulse Width Modulation (PWM) control and the duty cycle of the pulses determines the speed of the motors. As with any such dynamic system, there are constraints on the maximum speed the robots can travel and the change in velocity (acceleration) so that the motion of the robots is stable with minimal wheel slippage and skidding. The maximum limits for velocity and acceleration are determined practically. As a result, the following equations need to be satisfied always.

$$\begin{aligned} \left|V_{l|k}\right| &< V_{\max} \\ \left|V_{r|k}\right| &< V_{\max} \\ \left|V_{l|k} - V_{l|k+1}\right| &< a_{\max} \\ \left|V_{r|k} - V_{r|k+1}\right| &< a_{\max} \end{aligned} \qquad \ldots(4.18)$$

4.3 A Solution to the Optimal Control Problem

We have a system that is governed by a set of differential equations (state transition model) that has several constraints and an objective function that needs to be maximized given the current state of the system. We do not have a set terminal constraint and the robots are free to roam around. Therefore, we only have the current state of the system and the system model that can be used to predict the future state of the system with some degree of certainty. We can either keep the problem in the continuous domain and solve it as an initial value problem or solve it using dynamic programming in the discrete domain. The latter approach is used here.

4.3.1 Dynamic Programming

Solving this problem involves setting up the receding horizon formulation; that is set up the objective cost function and search for the path that gathers the most information over the short time horizon while satisfying the constraints. Apply the first control command of the generated sequence and then repeating the process until the information threshold

is reached. The path search can be performed in two ways, the top-down approach and the bottom-up approach. The following sections provide a detailed description of the process. The notation used is due to Bertsekas [51].

4.3.2 Principle of Optimality

Dynamic programming is based on the principle of optimality due to Bellman [80]. The control commands are represented by μ_k, which is a vector of all the command inputs of all the robots. The control command sequence from time, t = 0 to t = N is denoted by π, given by,

$$\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\} \quad \ldots(4.19)$$

π is referred to as the policy or the control law and it represents a set of all possible command sequences. Our objective is to find the instance of π that provides maximum information gain. This is denoted by,

$$\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\} \quad \ldots(4.20)$$

The principle of optimality is quite simple and it states the following: if Eqn. 20 represents the optimal policy for a given problem, considering a sub-problem wherein the system is at time, $t = i$, following the optimal policy, the system state is X_i. If we wish to find the optimal policy from i to N with respect to the same cost function, then the truncated policy,

$$\pi^* = \{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\} \quad \ldots(4.21)$$

is optimal for this sub-problem. In simpler terms, if an optimal route from Adelaide to Sydney goes through Melbourne with respect to some cost function, the principle of optimality states that the Melbourne to Sydney part of the route is also optimal (with regard to the same cost function). Thus, the principle of optimality implies that an optimal policy can be generated sequentially by solving simpler sub-problems.

4.3.3 General Form

The modelling of a problem in the dynamic programming paradigm consists of establishing a state space model of the system, the controls and the effect of control on the system states and the cost function. The dynamic system is represented by,

$$x_{k+1} = f_k\left(x_k, u_k, w_k\right) \quad \text{for, } k = 1,2,\ldots,N$$

…(4.22)

Here, x_k is an element of the state space denoted by S, u_k is the control command that is constrained to take values with respect to the various constraints discussed in the previous section. w_k is the random process noise with a probability P_k that depends on x_k and u_k. f_k is the function that maps the system states in time, given the control. It is referred to as the state transition model. N represents the horizon or the number of time-steps over which the control is applied.

Finally, the cost used is additive over time and it depends on the system state, control and the process noise. For a deterministic system, the cost function can be written as,

$$I = g_N\left(x_N\right) + \sum_{k=0}^{N-1} g_k\left(x_k, u_k, w_k\right)$$

…(4.23)

The first term represents the cost at the end of the horizon and the second term represents the cost incurred at every intermediate time-step. However, in our case, as we have a stochastic system where w_k is a random variable, the cost function also becomes a random variable. Therefore, we use the expected cost,

$$I = E\left\{ g_N\left(x_N\right) + \sum_{k=0}^{N-1} g_k\left(x_k, u_k, w_k\right) \right\}$$

…(4.24)

It should be noted here that the actual cost function used in this work is FIM based and, as described in section 4.1.1, Fisher information is the expectation of a score of a random variable that has been observed. Moreover, the system state modelling is exactly the same as that used in the previous chapter for the estimation process. As a result the problem is very well suited for a solution using dynamic programming.

4.3.4 Nonlinear Model Predictive Control (NMPC)

NMPC is a sub optimal control scheme that addresses the inadequacies of popular schemes such as the linear quadratic model, minimum variance control, etc. These methods become inadequate mainly for two reasons. One is the presence of non-linearity in the system, where, the system is linearized around some point, and the second is that the constraints cannot readily be modelled through a simple linear feedback controller. Furthermore, the presence of constraints often dictates that the system be run near the boundaries of the constraints and the solutions provided by linear quadratic control [81] are not suited for this.

The primary objective of NMPC is to produce a with a stable closed loop system while fulfilling the state and control constraints. However, the issue with this is in ensuring the stability of the controller. A deterministic dynamic system with input is deemed to be stable if any bounded input causes a bounded output (BIBO stability criterion). An NMPC based controller is deemed to be stable if the total cost over an infinite time horizon is finite. This holds only for a deterministic system and cannot be guaranteed for a stochastic system. For example, the constraint for guaranteed stability on a quadratic cost function used in [51] would be,

$$\sum_{k=0}^{\infty}\left(x_k^{'}Qx_k + \bar{\mu}(x_k)'R\bar{\mu}(x_k)\right) < \infty$$

…(4.25)

However, the same cannot be said about the cost function in eq. 4.24 because of the presence of the random variable w_k. A feasible solution to this problem is the introduction of a set-membership constraint on w_k similar to those used in [51] and [82]. Essentially, this process involves constraining the random variable w_k such that $w_k \in W(x,\mu(x))$. This in turn constrains the state space such that,

$$x_{k+1} = f_k\left(x_k, u_k, w_k\right) \in T$$

…(4.26)

Where, T is a subset of the state space called the target set and the controller is required to drive the system only into T. This constraint makes the system stable.

However, from a practical perspective, although there are dynamic programming algorithms for problems where the state estimates are not perfect, they are computationally more intensive than their deterministic counterparts. In fact, in practice, they are rarely ever used. Conceptually, the optimization process is the same for both cases. Therefore, it is common practice to reduce the stochastic system to a deterministic system and solve the problem sub-optimally. Since our problem is predictive, conceptually optimal solutions can only be found using hindsight and, therefore, the process is sub-optimal even without this approximation.

In our case, it is known that the noise that corrupts the state estimation process is Gaussian. The EKF produces least squared error state estimates on this assumption. Therefore, we have state estimates that are believed to be the most likely based on prior information. Hence, we base our policy generation process based on the most likely state for a given input. We must recall that the policy itself is not fully employed, only the first control command is applied and the process repeated at the next time step. As a result, stability is ensured locally at each time-step using the above method. Furthermore, since only the most likely states are considered, the stochastic system reduces to a deterministic system.

4.3.5 Branch-and-Bound

The preceding sections were about the modelling of the system and reducing it to a tractable and computational level. Having done that, we must now find the optimal policy that maximizes the information gathered. There are several methods to do this. We have a problem where the current state is known and the robots have a decision to make regarding the control input. Depending on the decision, the system can be driven into other states and for each choice the cost function evaluated and stored as, *I*. This forms an acyclic graph (shown in Figure 26). It can be seen that the number of feasible solutions is

going to be very large. Consequently, an exhaustive search of all possible solutions is going to be computationally intensive and quite possibly impractical to implement.

For these reasons, we use the Branch-and-Bound (BB) method to find the optimal solution. The key idea behind BB is to partition the solution space into smaller subsets and then use some bounds on the best possible solution that can be obtained through a subset to eliminate some of them. The BB algorithm can be summarized as follows,

For the problem of minimizing the cost function, *I(x)* over $x \in X$, where, X is the solution space. We first divide the space into two subsets X_1 and X_2. Then we need to set the upper and lower bounds (details in the next chapter). The subset X_2 may be chosen over X_1 if,

$$\text{Lower bound, } I_1 \leq \min_{x \in X_1} I(x) \text{ and,}$$

$$\text{Upper bound, } I_2 \leq \min_{x \in X_2} I(x)$$

…(4.27)

For each node on the graph, the subsequent subsets are identified and the upper and lower bounds are calculated. Whenever the above condition is satisfied the corresponding sub graph is disregarded. More detailed accounts of the BB algorithm can be found in [83].

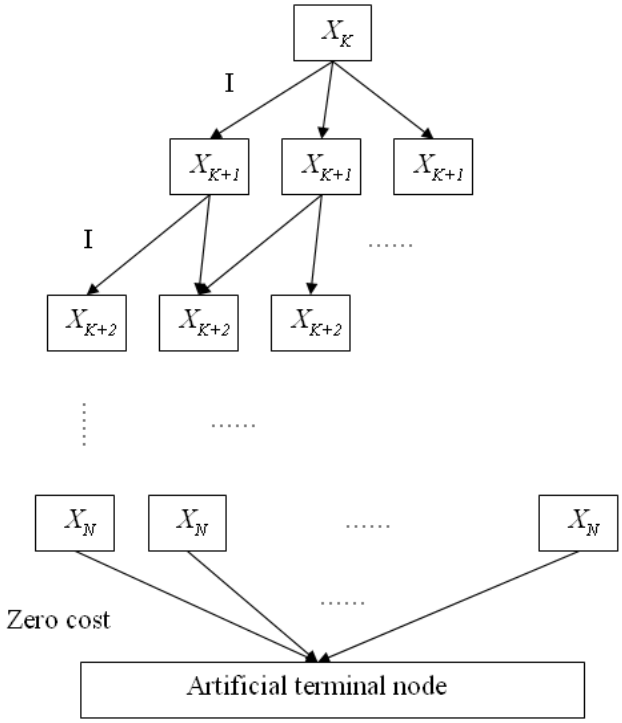


Figure 26 – Structure of the problem.

4.4 Summary

In this chapter we have looked at the development of a novel cost function for cooperative observer motion optimization from a sparse 3D structure estimation perspective. Subsequently, the various system constraints were also modelled and incorporated into the framework. Finally, a solution to the optimal control problem was proposed using the concepts of dynamic programming and receding horizon control. This concludes the theoretical modelling of the system. The following chapter will discuss the various implementation issues, the results obtained using this algorithm and its effectiveness.

Chapter 5

Results and Discussion

This chapter presents some of the results that were obtained during the course of this study. These results validate and support the claims made by this dissertation. The results are based on synthetic data generated using Webots. The simulated models used to generate the synthetic data have been meticulously designed to resemble the real-time environment as closely as possible.

5.1 The experiment

A simulation environment was created with the following objectives,

- Each robot in the environment needs to localize itself from an initial map of the environment, the measurements being made to features in the map through an on-board electro-optic sensor and odometry.
- There is a moving object in the environment with two distinct features on it (a red coloured and green coloured beacon) that define its structure. These features need to be tracked over time, so that information about the location, velocity and structure of the object may be obtained.
- As the previous task cannot be completed by a single robot two robots need to form a team and share information appropriately. As the solution to this problem is non-unique (there are multiple trajectories that the robots can follow), the robots need to optimize their paths in terms of the maximum information gained.

We have previously defined a mathematical model to achieve all of these objectives and the following section provides some illustration this model's effectiveness.

5.2 Localization

In section 3.3, it was shown that the SLAM based localization scheme improves the robot location estimate compared to the estimates based on odometry alone. When ground truth is not available, it is hard to validate this claim. However, the Kalman filter also keeps an estimate of the uncertainty in the estimation due to process noise and measurement noise. This can be represented in the form of an error ellipse at a given standard deviation (σ). Figure 27 shows the results with the 0.5σ uncertainty ellipses superimposed on the trajectory. This also shows that the error on the estimate is bounded as long as a landmark is observed every now and then (the maximum interval depends on the error introduced by the odometry). The block arrow in Figure 27 shows the nominal error level. The uncertainty information is also important when we move on to the tracking of the moving object because, the location of the moving object, from the estimation point of view, is a function of the robot location and the measurements made to the features on it. So, this directly affects the uncertainty in the estimation of the parameters describing the moving object.

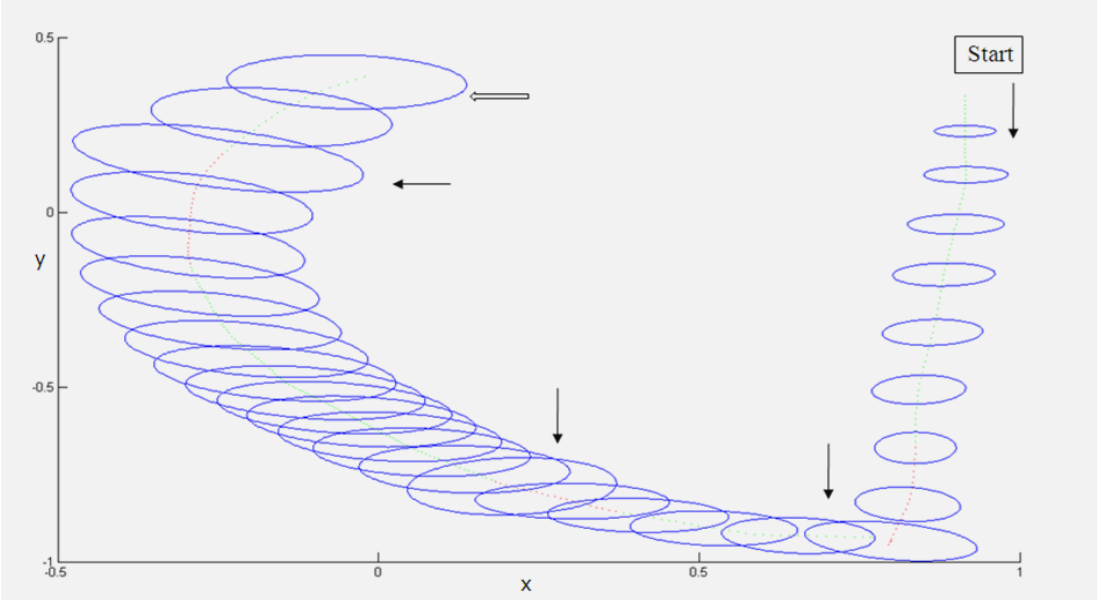


Figure 27 – Estimated trajectory with the error ellipses; arrows indicates the regions where the uncertainty increases and then decreases as soon as a landmark (or map feature) is observed.

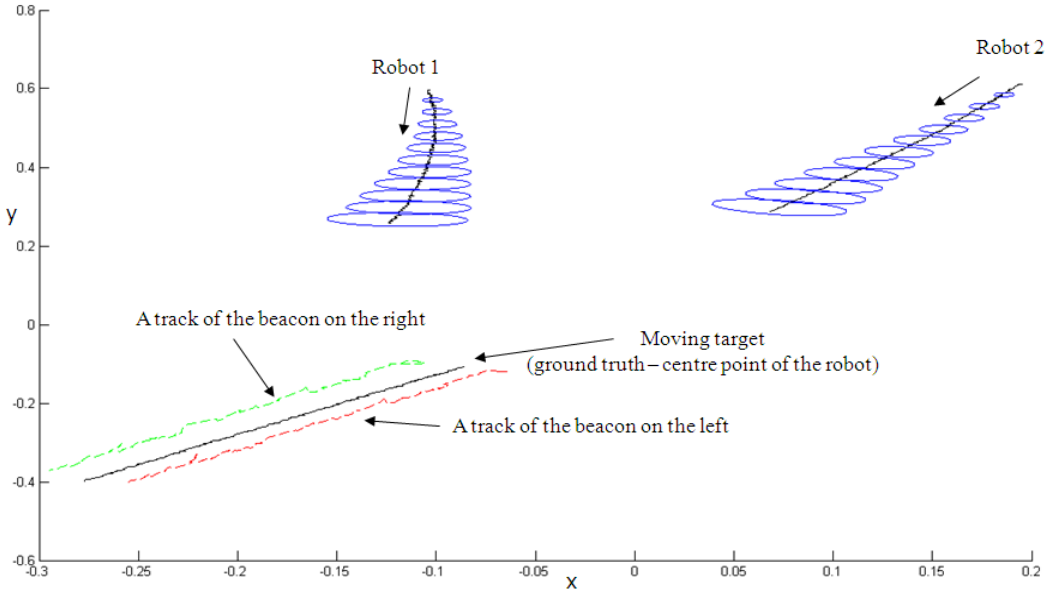


Figure 28 – Robot trajectories with error ellipses along with the moving object track.

5.3 Moving target estimation

To localize a moving object using bearing measurements alone, two concurrent measurements from two unique locations are required. The two robots working as a team provide these two measurements. Using the estimated location of the robots and the measurements made by them, the location of the moving target is estimated. In our case, the robots keep track of a red coloured and a green coloured beacon on the moving object. Figure 28 shows the estimated trajectory of the robots and the two beacons. Error ellipses show the error profiles on the surveillance robot location estimation. The pose of each robot is defined by three parameters, x,y and θ. Each of these parameters has a Gaussian error profile attached to it and so does each measurement made by the robots. So, the location of the robot depends on eight parameters and all of them are known only with a degree of certainty. This means that the estimates of the parameters pertaining to the moving target will inherit these error profiles. The error profiles attached to the estimates will be elliptical if the measurements, observing robot poses and the moving

object are related in a linear fashion. However, this is not strictly the case and in fact we have used linear approximations of the nonlinear functions that relate these parameters. Figure 29 and Figure 30 show the approximate elliptical error profile attached to estimate of the red and green beacons respectively.

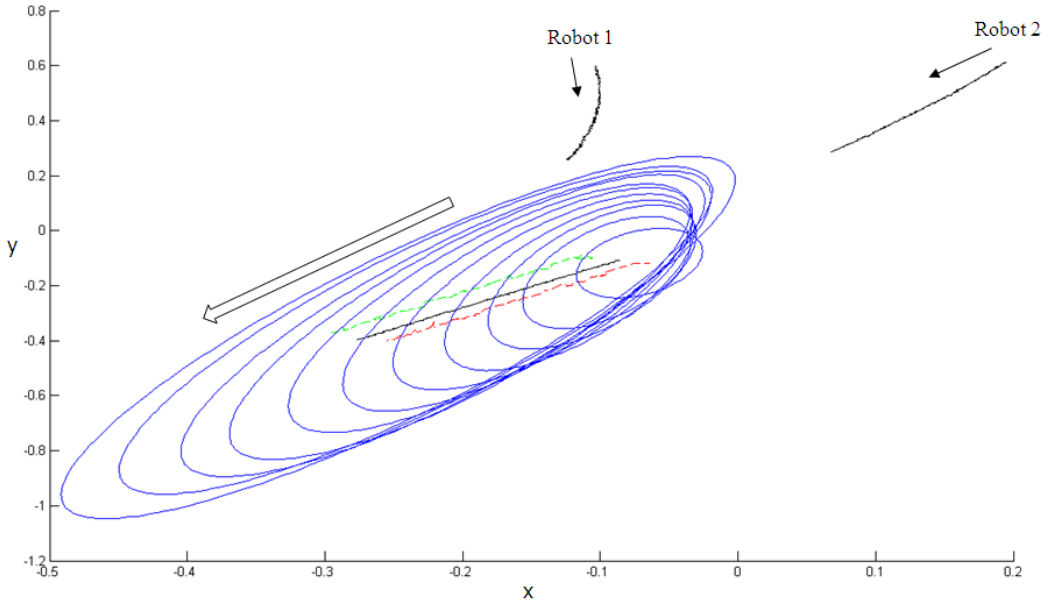


Figure 29 – Error ellipses associated with the red beacon.

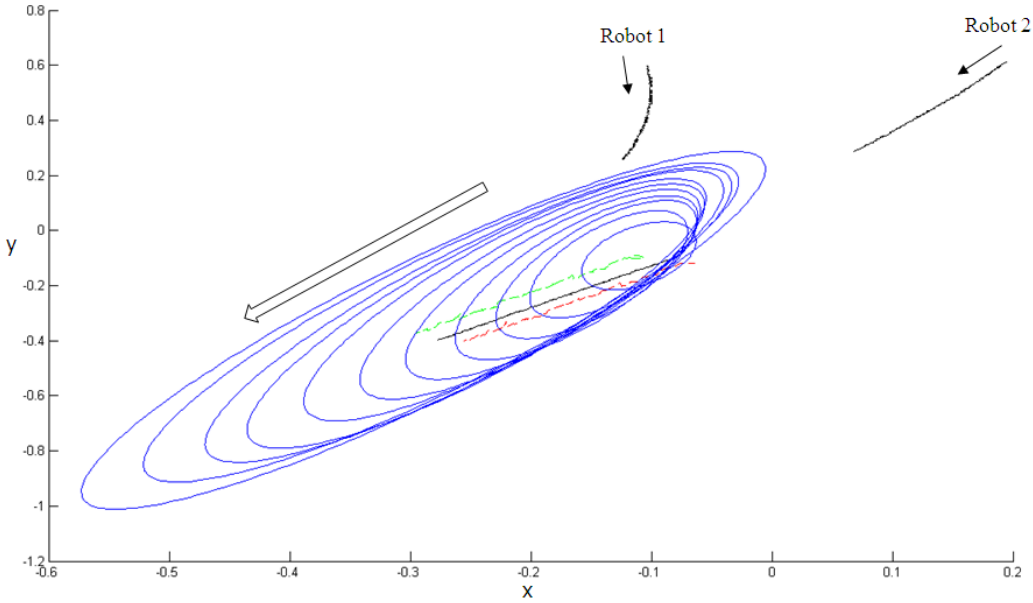


Figure 30 – Error ellipses associated with the green beacon.

5.4 Optimization

In the previous section, we have shown the initial results, the evolution of the error profiles on relevant parameters, and their effect on the estimation of the moving object location. The objective of the optimization process is to manipulate some of the system parameters in order to reduce some of the error estimates, particularly, the error on the moving object track. (The mathematical description of this process and the receding horizon formulation are available in Chapter 4). In our system model, the manipulated variables are the wheel speeds of the two robots. The optimization scheme alters these variables and subsequently changes the x,y and θ of each robot to affect the error profiles. Initially, the robots just follow the moving object for a short time before the optimization scheme turns on. This is to let the EKF build a reasonable estimate of the velocity of the moving object to facilitate the predictive control meaningfully. The optimization process is computationally very intensive; therefore, the state space is coarsely discretized. This will yield a coarse solution although experience has shown that the amount of

information gained by finding a fine solution is often not worth the extra computational resources. All possible input combinations are generated and the EKF is then run for horizon time (five time steps in our case). The cost function is also generated for each combination. A lower bound and an upper bound are required for BB (Branch and Bound) search. In theory, the lower bound is usually the Cramer-Rao Lower Bound (CRLB), but practically the lower bound is hardly ever reached. Therefore, as the search progresses the lowest cost is tracked and paths that incur more cost are rejected. Upper bound is handled in the same way. To make the computation faster, whenever the forcing functions introduced large penalties (PSF, FOV constraint, etc.), those paths were also rejected. Collectively, this process yields cooperative behaviour. Figure 31 shows the effect of the optimization process. Error ellipses are shown for both beacons and the maximum error (the major axis of the ellipse) was reduced by an average of about 16%.

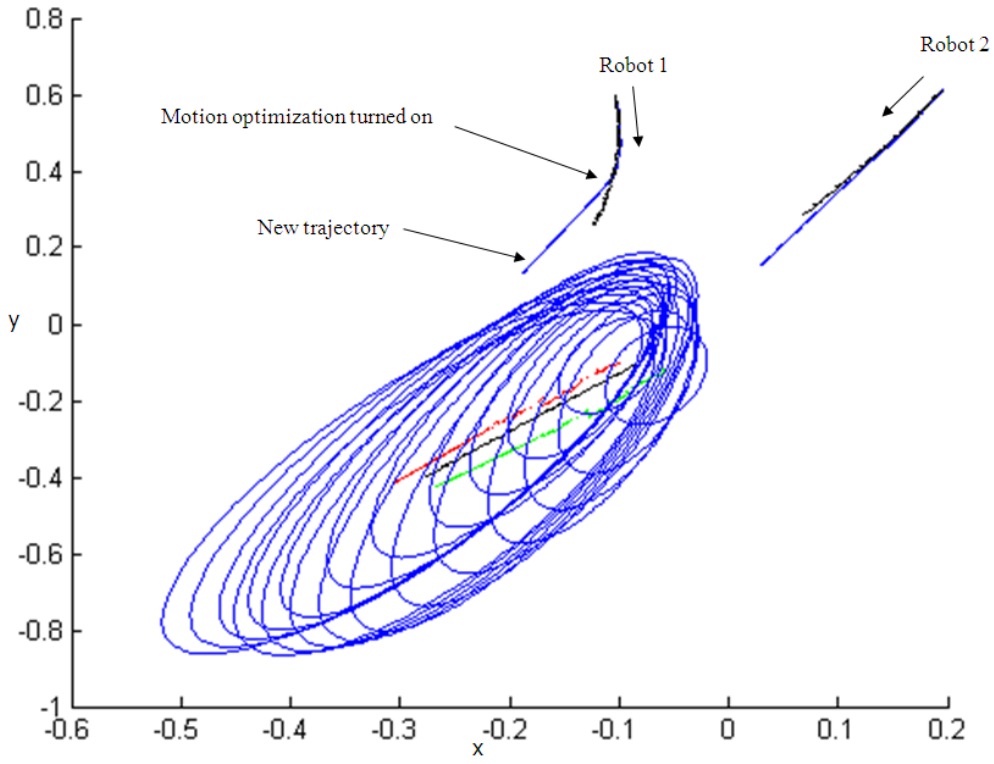


Figure 31 – Centralized motion optimization.

The problem with such a centralized optimization scheme is the computational cost. Even after the reduction of the state space, each iteration of this algorithm took an average of about 110 seconds on a computer equipped with an Intel Core Duo processor running at 2.0GHz with 2GB of RAM.

Consequently, the problem was broken down with the aim of finding a sub optimal solution that has a very low computational cost but at the same time provide reasonable performance. Figure 32 shows the result of a scheme that provides almost similar performance but at a highly discounted computational cost. This optimization scheme runs separately with one robot assuming a nominal trajectory for the other (EKF prediction) and then optimizing the trajectory of the second robot assuming that robot one follows the trajectory suggested by the optimization scheme in the first step[2]. This scheme only takes an average of 1 second to compute on the same computer. The reason for this is, if a robot has four command options (forward fast, forward slow, bank left, bank right), for a prediction horizon of five time steps, the number of command combinations are given by, $4^5 = 1024$. For two robots, the number of combinations is given by, $4^5 \times 4^5 = 1048576$. This explains the reason for the difference in computational cost. Although the two schemes deliver slightly different trajectories for the robots, the difference in terms of information gain is negligible.

[2] This is often referred to as passive or vicarious collaboration.

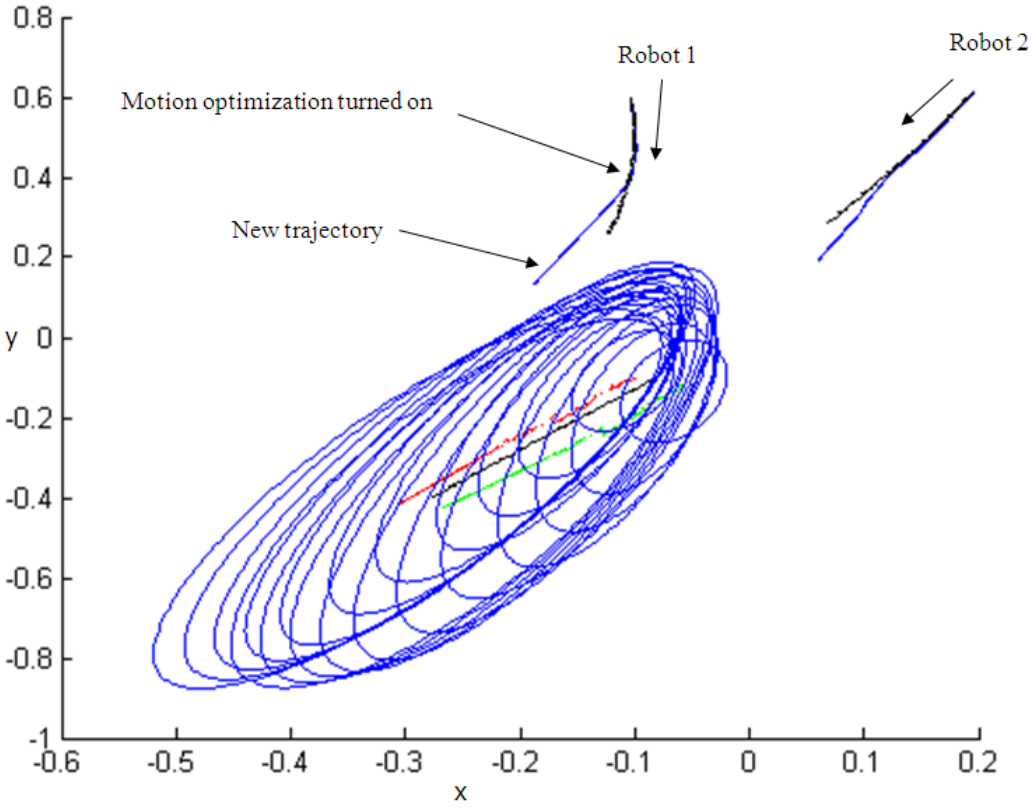


Figure 32 – Distributed motion optimization.

5.5 Summary

In this chapter, we have shown some results that demonstrate the effectiveness of the models developed in this thesis. We have shown how this study could serve as a precursor for obtaining complete situation awareness using a team of robots with very limited sensing capability. We have also discussed a motion optimization scheme that is computationally feasible without dimensionality problems. The following chapter provides a summary of this study and some possible future extensions.

Chapter 6

Conclusion

6.1 Conclusion and Summary

In this dissertation, a framework has been established that can be used to evolve low level situation awareness. Firstly, we have developed an estimation methodology that produces estimates of the location of the robots observing a scene from a map and the location and motion parameters of distinct points on a moving object that forms part of the scene. Secondly, we have developed a near optimal feedback control mechanism that integrates with the low level robot control, to manoeuvre the observing robots on information rich paths such that maximum information is collected about the structure of the moving object.

This work has been inspired from work published in recent literature; it includes, SLAM with Moving Object Tracking (SLAMMOT) [2], bearing-only localization and mapping [13] and Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) [22]. Our algorithm is a combination of these two approaches. The objective is to explore new areas and follow moving targets while keeping them under the sensor's field of view with trajectories planned and optimised against 3D structure. An indoor localization infrastructure is used that provides the location of the robots and the targets. We have developed a robot team with a sensor suite consisting of only a limited field of view monocular camera and odometry and we have demonstrated that we can track moving objects through the cooperative action of the robots without the aid of external localization infrastructure and satisfying field of view constraints. Although our test scenario represents a case where there are just two robots and one moving target it

can easily be extended for cases where there are multiple moving targets and more than two robots.

This work stands at the juncture of high level and low level situation awareness. In classic SLAM algorithms, data pertaining to dynamic features in the scene are considered a hindrance and hence discarded. Also, with a bearing-only sensor, it is not possible to determine all the parameters pertaining to dynamics features in the scene. Therefore, at the lower levels, we have the predictive control and the estimation of parameters pertaining to dynamic features of a scene by way of cooperation between team members, who cannot perform this task individually due to the constraints at the sensor level. This means that we have opened the doors for a variety of operations that enable the derivation of high level situation awareness.

This work has been conducted with a view to the concept of affordably expendable robots.. In the context of military surveillance, this means having to work with teams of inexpensive vehicles with very limited sensing and mobility features. The aim is to get these vehicles to work as a team and perform relatively complex tasks such that their cooperation leads to improvements over and above more expensive single-vehicle robots.

6.2 Future Scope

This work has evolved from the integration of a number of research areas in autonomous systems including SLAM, moving object tracking, cooperative activities, situation awareness, etc. therefore, there are a number of ways this work can be continued in the future. The following sections describe a few possibilities.

6.2.1 Situation Awareness/Scene Understanding

Based on model used for representation of a vision based cognitive system for scene understanding (Section 1.5, Figure 2), high level situation awareness is obtained through the top two levels consisting of the behavioural primitives level (BPL) and the conceptual primitives level (CPL). CPL refers to the conversion of raw data into a more descriptive representation. In our context, this can refer to the conversion of the location and speed of

a moving object over time into descriptors like, “moving object – friendly – searching for targets” or “moving object – unknown – taking evasive action”. At the BPL, all information from the CPL is collected and a global understanding of the scene can be generated and appropriate action can be taken. For example, if a scene is filled with moving objects that are deemed to be friendly and form part of the team, and if nothing of interest is found, it might be a good idea to label the area as “safe – nothing of interest” and move to other areas. Though there is a long way to go to be able to do this, our study proves that it is indeed possible to do this by providing the raw low level data.

6.2.2 SIFT/SURF Integration

In this study, the stationary landmarks and the moving targets were easily identifiable beacons. Practically, this will seldom be the case. Therefore, the feature extraction stage should be modified through the integration of sophisticated techniques such as SIFT and SURF. Although these techniques are quite robust, they can generate a lot of feature points and they would need to be managed appropriately. Also, the problem of classifying the feature points as stationary and non-stationary will be non-trivial. Furthermore, the feature extraction algorithms themselves are computationally quite expensive.

6.2.3 Complexity

Systems such as the one developed in this study always come at the expense of implementation complexity. As the system is used to solve larger and more complex problems, the complexity of the system itself increases. As the state vector grows due to the increase in the number of landmarks and the dynamic features; the computational time required for each EKF iteration increases as well. Though there are solutions to these problems, they are significantly more complex and hence harder to implement. The specific implementation of the motion optimization problem that we have solved for the case of a two robot team with one target becomes computationally intractable as the size of the team increases. There are specific issues of hardware and algorithmic

implementation that should be addressed with respect to feasible and infeasible solution determination.

6.2.4 GPS Integration

This work has been carried out under the assumption that no external localization infrastructure (such as GPS and UWB positioning) is available. However, when some sort of infrastructure is available, it is imperative that such infrastructure be utilized. It would be interesting to find out the effects of including such measurements whenever available. This is important because, SLAM algorithms that perform well in real systems are quite complex due to some of the issues faced, such as, data association in the large (city-sized SLAM [84]), CI maps used to reduce computational load, linearization of highly nonlinear functions in the estimation model, etc. Analysis of the integration of an external localization system will answer questions such as, the extent to which a simple EKF can handle these problems. For example, if a GPS update is available at least once every 30 minutes, it should be possible to get away with a very simple estimation methodology.

6.2.5 Implementation

The algorithms and techniques developed so far have been validated through a synthetic environment where everything in the lab has been recreated as closely as possible. The next step is to transfer them to the actual laboratory and validate the algorithms. Subsequently, this can be implemented in a more realistic environment instead of miniature robots.

References

1. Endsley, M.R., *Toward a theory of situational awareness.* Human Factors, 1995a. **37**(1): p. 32-64.
2. Chieh-Chih, W., et al., *Simultaneous Localization, Mapping and Moving Object Tracking.* Int. J. Rob. Res., 2007. **26**(9): p. 889-916.
3. Dempster, A.G., *Dilution of precision in angle-of-arrival positioning systems.* Electronics Letters, 2006. **42**(5): p. 291-292.
4. Peter, G., E. Peter, and V. Markus, *Simultaneous Motion and Structure Estimation by Fusion of Inertial and Vision Data.* Int. J. Rob. Res., 2007. **26**(6): p. 591-605.
5. Finn, A., K. Kabacinski, and S.P. Drake. *Design Challenges for an Autonomous Cooperative of UAVs*. in *Information, Decision and Control, 2007. IDC '07*. 2007.
6. Nullmeyer, R.T., et al., *Human Factors in Air Force Flight Mishaps: Implications for Change*, in *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2005*. 2005, National Training Systems Association.: Arlington, VA
7. Gorman, J.C., N.J. Cooke, and J.L. Winner, *Measuring team situation awareness in decentralized command and control environments.* Ergonomics, 2006. **49**(12): p. 1312 - 1325.
8. Hagen, P.E., N. Størkersen, and K. Vestgård, *The HUGIN 1000 military AUV system*, in *UDT Europe 2004*. 2004: Nice, France.
9. Bingham, D., et al., *The Application of Autonomous Underwater Vehicle (AUV) Technology in the Oil Industry - Vision and Experiences*, in *FIG XXII International Congress*. 2002: Washington, D.C. USA.
10. Leonard, J.J. and H.F. Durrant-Whyte. *Simultaneous map building and localization for an autonomous mobile robot*. in *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*. 1991.
11. Dissanayake, M.W.M.G., et al., *A solution to the simultaneous localization and map building (SLAM) problem.* Robotics and Automation, IEEE Transactions on, 2001. **17**(3): p. 229-241.
12. Davison, A.J., et al., *MonoSLAM: Real-Time Single Camera SLAM.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2007. **29**(6): p. 1052-1067.
13. Deans, M.C., *Bearings-Only Localization and Mapping*, in *School of Computer Science*. 2005, Carnegie Mellon University: Pittsburgh, Pennsylvania.
14. Deans, M., *Bearings-only localization and mapping*, in *Robotics institute*. 2005, Carnegie Mellon University: Pittsburg, PA.
15. Aggarwal, J.K. and N. Nandhakumar, *On the computation of motion from sequences of images-A review.* Proceedings of the IEEE, 1988. **76**(8): p. 917-935.

16. David, G.L., *Distinctive Image Features from Scale-Invariant Keypoints.* Int. J. Comput. Vision, 2004. **60**(2): p. 91-110.
17. Bay, H., et al., *Speeded-Up Robust Features (SURF).* Computer Vision and Image Understanding, 2008. **110**(3): p. 346-359.
18. Berthold, K.P.H. and G.S. Brian, *Determining Optical Flow*. 1980, Massachusetts Institute of Technology.
19. Kuzu, Y. and V. Rodehorst. *Volumetric Modelling Using Shape from Silhouette*. in *Fourth Turkish-German Joint Geodetic Day*. 2001.
20. Rothganger, F., et al. *3D object modeling and recognition using affine-invariant patches and multi-view spatial constraints*. in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. 2003.
21. Khatib, O., et al. *Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation*. in *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*. 1996.
22. Parker, L.E. *Current State of the Art in Distributed Autonomous Mobile Robotics*. in *Distributed Autonomous Robotic Systems 4*. 2000: Springer-Verlag Tokyo.
23. Mataric, M.J., *Issues and Approaches in the Design of Collective Autonomous Agents* Robotics and Automation Systems, 1995. **16**: p. 321-331.
24. Craig, W.R., *Flocks, herds and schools: A distributed behavioral model*, in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. 1987, ACM.
25. Kalra, N., *A Market-Based Framework for Tightly-Coupled Planned Coordination in Multirobot Teams*, in *The Robotics Institute*. 2007, Carnegie Mellon University: Pittsburgh, Pannsylvania.
26. Murray, R.M., *Nonlinear control of mechanical systems: A Lagrangian perspective.* Annual Reviews in Control, 1997. **21**: p. 31-42.
27. Richard M. Murray, J.H., Ali Jadbabaie, Mark B. Milam, Nicolas Petit, William B. Dunbar, Ryan Franz,, *Online Control Customization via Optimization-Based Control*, in *Software-Enabled Control*, G.B. Tariq Samad, Editor. 2004. p. 149-174.
28. Stark, H. and J.W. Woods, eds. *Probability, random processes, and estimation theory for engineers*. 1986, Prentice-Hall, Inc. 430.
29. Melsa, J.L. and D.L. Cohn, *Decision and Estimation Theory*. 1978, Mexico: McGraw-Hill.
30. Mendel, J.M., *Lessons in digital estimation theory*. 1986: Prentice-Hall, Inc. 304.
31. Mahamud, S. and M. Hebert. *Iterative Projective Reconstruction from Multiple Views*. in *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*. 2000.
32. Mahamud, S., et al., *Provably-Convergent Iterative Methods for Projective Structure from Motion.* Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 2001. **1**.

33. Triggs, B., et al., *Bundle Adjustment - A Modern Synthesis*, in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*. 2000, Springer-Verlag.
34. Hartley, R. and A. Zisserman, *Multiple View Geometry in Computer Vision*. 2003: Cambridge University Press. 700.
35. Gill, P.E. and W. Murray, *Algorithms for the Solution of the Nonlinear Least-Squares Problem.* SIAM Journal on Numerical Analysis, 1978. **15**(5): p. 977-992.
36. Kwakernaak, H., *Linear Optimal Control Systems*, ed. S. Raphael. 1972: John Wiley \& Sons, Inc. 575.
37. Kalman, R.E., *A New Approach to Linear Filtering and Prediction Problems.* Transactions of the ASME-Journal of Basic Engineering, 1960. **82**(1): p. 35-45.
38. Brown, R.G. and P.Y.C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering: With Matlab Exercises and Solutions*. 1996: John Wiley & Sons Inc.
39. Montemerlo, M. *FastSLAM: A factored solution to the simultaneous localization and mapping problem*. in *AAAI National Conference on Artificial Intelligence* 2002.
40. Montemerlo, M., et al., *FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges*, in *Proceedings of the 18th international joint conference on Artificial intelligence*. 2003, Morgan Kaufmann Publishers Inc.: Acapulco, Mexico.
41. Pinies, P. and J.D. Tardos. *Scalable SLAM building conditionally independent local maps*. in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. 2007.
42. Pinies, P. and J.D. Tardos, *Large-Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision.* Robotics, IEEE Transactions on, 2008. **24**(5): p. 1094-1106.
43. Arulampalam, M.S., et al., *A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking.* Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on], 2002. **50**(2): p. 174-188.
44. Doucet, A., et al., *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks*, in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. 2000, Morgan Kaufmann Publishers Inc.
45. Bekris, K.E., M. Click, and E.E. Kavraki. *Evaluation of algorithms for bearing-only SLAM*. in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. 2006.
46. Julier, S. and J. Uhlmann. *A New Extension of the Kalman Filter to Nonlinear Systems*. in *In Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*. 1997.
47. Julier, S. and J.K. Uhlmann, *A General Method for Approximating Nonlinear Transformations of Probability Distributions*. 1999, unknown.
48. Julier, S.J., J.K. Uhlmann, and H.F. Durrant-Whyte. *A new approach for filtering nonlinear systems*. in *American Control Conference, 1995. Proceedings of the*. 1995.

49. Mclauchlan, P.F., *The Variable State Dimension Filter applied to Surface-Based Structure from Motion*. 1999, University of Surrey: Guildford.
50. Bellman, R.E., *Adaptive Control Processes: A Guided Tour*. 1961: Princeton University Press.
51. Bertsekas, D.P., *Dynamic Programming and Optimal Control*. third ed. Vol. 1. 2005, Belmont, Massachusetts: Athena Scientific.
52. Kay, S.M., *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall Signal Processing Series 1993, London: Prentice Hall International.
53. Ousingsawat, J. and M. Campbell, *Optimal Cooperative Reconnaissance Using Multiple Vehicles.* Journal of Guidance, Control, and Dynamics 2007, 2007. **30**(1): p. 122-132.
54. Braitenberg, V., *Vehicles: Experiments in Synthetic Psychology*. 1984, Cambridge, MA: The MIT Press.
55. Reynolds, C. *Steering Behaviors for Autonomous Characters*. in *Game Developers Conference 1999*. 1999.
56. Lu, F. and E. Milios, *Globally Consistent Range Scan Alignment for Environment Mapping.* Auton. Robots, 1997. **4**(4): p. 333-349.
57. Endsley, M.R. and W.M. Jones, *A Model of Inter- and Intrateam Situation Awareness:Implications for Design, Training and Measurement*, in *New Trends in Cooperative Activities*, M. McNeese, E. Salas, and M. Endsley, Editors. 2001, Human Factors and Ergonomics Society: Santa Monica.
58. McGrath, J.E., *Time, Interaction and Performance(TIP): A Theory of Groups.* Small Group Research, 1991. **22**: p. 147-174.
59. Orasanu, J. and E. Salas, *Team Decision Making in Complex Environments*, in *Decision Making in Action: Models and Methods*, G.A. Klien, et al., Editors. 1993, Albex.: Norwood, NJ. p. 327-345.
60. Khepera-II. *http://www.k-team.com*. Miniature mobile robot.
61. Micheal, O., *Webots: Professional Mobile Robot Simulation.* Journal of Advanced Robotics Systems, 2004. **1**(1): p. 39-42.
62. Krystian, M. and S. Cordelia, *Scale \& Affine Invariant Interest Point Detectors.* Int. J. Comput. Vision, 2004. **60**(1): p. 63-86.
63. Lowe, D.G. *Object recognition from local scale-invariant features*. in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. 1999.
64. Bay, H., T. Tuytelaars, and A. Van Gool. *SURF: Speeded Up Robust Features*. in *9th European Conference on Computer Vision*. 2006.
65. Oshman, Y. and P. Davidson, *Optimization of observer trajectories for bearings-only target localization.* Aerospace and Electronic Systems, IEEE Transactions on, 1999. **35**(3): p. 892-902.
66. Jensfelt, P., et al. *A framework for vision based bearing only 3D SLAM*. in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. 2006.
67. Luis, G., et al. *vSLAM: vision-based SLAM for autonomous vehicle navigation*. 2004: SPIE.

68. Grocholsky, B., A. Makarenko, and H. Durrant-Whyte. *Information-theoretic coordinated control of multiple sensor platforms*. in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*. 2003.
69. Grocholsky, B., *Information-Theoretic Control of Multiple Sensor Platforms*. 2002, University of Sydney: Sydney.
70. Van Trees, H.L., *Detection, Estimation, and Modulation Theory, Part I*. 1968, New York: Wiley.
71. Allgöwer, F. and A. Zheng, *Nonlinear Model Predictive Control*. Progress in Systems and Control Theory, ed. C. Byrnes. Vol. 26. 2000: Birkhäuser Verlag.
72. Rawlings, J.B., *Tutorial overview of model predictive control.* IEEE Control Systems Magazine, 2000. **20**(3): p. 38-52.
73. Cover, T.M. and J.A. Thomas, *Elements of information theory*. 1991: Wiley-Interscience. 542.
74. Nardone, S., A. Lindgren, and G. Kai, *Fundamental properties and performance of conventional bearings-only target motion analysis.* Automatic Control, IEEE Transactions on, 1984. **29**(9): p. 775-787.
75. Taek Lyul, S., *Observability of target tracking with bearings-only measurements.* Aerospace and Electronic Systems, IEEE Transactions on, 1996. **32**(4): p. 1468-1472.
76. Bishop, A., et al., *Optimality analysis of sensor-target geometries in Passive localization: Part 1 - bearing-only localization* in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing (2007 : Melbourne, Vic.)* 2007, Institute of Electrical and Electronics Engineers (IEEE) Melbourne, Australia.
77. Wu, J.J., R. Sharma, and T.S. Huang, *Analysis of uncertainty bounds due to quantization for three-dimensional position estimation using multiple cameras.* Optical Engineering, 1998. **31**(1): p. 280-292.
78. Ousingsawat, J. and M.E. Campbell, *On-line estimation and path planning for multiple vehicles in an uncertain environment.* International Journal of Robust and Nonlinear Control, 2004. **14**(8): p. 741-766.
79. Burgard, W., et al. *Collaborative multi-robot exploration*. in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*. 2000.
80. Bellman, R.E., *Dynamic Programming*. 2003: Dover Publications, Incorporated. 366.
81. Athans, M., *The role and use of the stochastic linear-quadratic-Gaussian problem in control system design.* Automatic Control, IEEE Transactions on, 1971. **16**(6): p. 529-552.
82. Bertsekas, D. and I. Rhodes, *Recursive state estimation for a set-membership description of uncertainty.* Automatic Control, IEEE Transactions on, 1971. **16**(2): p. 117-128.
83. Wosley, L.A., *Integer Programming*. 1998: Wiley, N. Y.

84. Wang, C.-C. and C. Thorpe. *A Hierarchical Object Based Representation for Simultaneous Localization and Mapping*. in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. September 2004.