Trajectory Planning for Feature-Based Localisation and Mapping

Cindy Leung

A thesis submitted in fulfilment
of the requirements for the degree of
Doctor of Philosophy

The University of Technology, Sydney
2007

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student

____

Acknowledgements

I would like to thank everybody who have supported and encouraged me during my candidature.

Firstly I would like to thank my supervisor, Professor Gamini Dissanayake, for his invaluable guidance, support, and for providing me this opportunity. I would also like to thank my co-supervisor Dr Shoudong Huang, who took me under his wing and led me through optimal control research and for providing helpful insight and explaining difficult concepts.

Thanks goes to Dr Adel Al-Jumaily for his help early in my candidature and for encouraging me to pursue a research degree in robotics. Additional thanks goes to Dr Ngai Kwok and Mr Alen Alempijevic for the snippets of code, Mr Damitha Herath for the data set, Ms Sharon Ong for the UAV model and Dr Haye Lau for proof reading and providing templates for most documents including red-tape reports, begging letters and acknowledgements.

In general, I wish to thank the great minds in all three nodes of CAS for providing a fantastic culture in which to research and learn, and I would like to thank Professor Hugh Durrant-Whyte, Dr Tomonari Furukawa and others in particular for organising workshops to expressly share some of that collective knowledge.

Back at UTS, thanks goes to Dr Matthew Gaston for a high availability computing cluster. I wish to also thank Matt, Zhan, Weizhen, Rami, Ashod and Jeff for the fun, friendship, food and daily discussions during my residence in CAS and also on level 20.

I must also thank those closest to me. Beginning with Ashley for his help with experiments, proof reading, love and support, and for the late-night lifts home. Last but not least, I owe my gratitude to my family Mr and Mrs Leung, including you Connie, for the years of education, encouragement and care, for making it easy for me to concentrate on my research and for the untimely late-night dinners.

Table of Contents

List of Figures

List of Tables

Abstract

Localisation and mapping are two fundamental tasks required for many autonomous mobile robot applications. Robot motion has a significant impact on the quality of the outcomes of any localisation and mapping algorithm due to the fact that the information acquired through observing the environment depends on the viewpoint from which the observations are made. This thesis develops mobile robot trajectory planning strategies that maximise the information gain, thereby leading to more effective localisation and mapping outcomes.

In this thesis, the trajectory planning problems are formulated as optimal control problems where the control objective is to maximise the information gain or to minimise the uncertainty of the robot location and/or the map estimate. As not all the information can be predicted before observations are made, the optimal control problems involve gradually identified models. Model Predictive Control (MPC) is proposed to solve the control problems as it exploits updated information in the planning at each time step. Two optimisation strategies are implemented in the multi-step optimization of MPC planning. The first is an Exhaustive Expansion Tree Search (EETS) and the second combines this search with Sequential Quadratic Programming (SQP). Analyses of the results show that EETS provides a near optimal solution.

Three trajectory planning problems are considered. The first problem is trajectory planning for multiple robots in the task of target localisation. The robots are equipped with bearing-only sensors and their locations are assumed to be provided by an external source. Given initial location estimates with large uncertainty, the task is to pinpoint the locations of multiple targets within a prescribed terminal time. An Extended Information Filter (EIF) is used to estimate the locations of the targets.

The second trajectory planning problem considers the task of point feature based Simultaneous Localisation and Mapping (SLAM) for a single robot. Here, the robot is to map a given area within a prescribed terminal time. An Extended Kalman Filter (EKF) is used to estimate the robot pose and feature locations. The final trajectory planning problem extends the SLAM problem further to the mapping of line features. As simply applying EKF to line-feature SLAM produces inconsistent estimates, a Smoothing and Mapping (SAM) algorithm is used as a viable alternative.

In these SLAM problems, coverage is another important performance index. As MPC with a few steps look-ahead is predominately a local planner without any long term planning or any explicit strategy for exploration, an attractor-based strategy is developed to improve its performance. With the addition of the attractor, coverage is shown to be much improved. On a Pioneer2DX robot, real-time line-feature localisation and mapping with SAM is demonstrated.

Nomenclature

Commonly used symbols in this thesis are listed as follows.

$\boldsymbol{\lambda}$, $\hat{\boldsymbol{\lambda}}$	Target or feature location, target or feature location estimate
$j = 1 \ldots J$	Number of features
$i = 1 \ldots T$	Update time steps
$k = 1 \ldots K$	Number of observations
D, N_ω	Planning horizon, number of discrete control options
$h(.)$	Observation model
f(.)	Process model
r, θ, φ	Range, bearing, elevation measurement
α, d	Angle and distance of line measurement
$\hat{\boldsymbol{\zeta}}_i, \hat{\boldsymbol{\Theta}}$	EKF state vector, SAM state vector
$\boldsymbol{\Sigma}_k$	Covariance of observation
$w_{\mathrm{x},}$	Single step process noise
w_{z}	Observation noise
$\mathbf{P}_w$, $\boldsymbol{\Lambda}_i$	Covariance of process noise (single step, multiple steps)
$\mathbf{u}_i$	Controls inputs
v, ω, ψ	Velocity, turnrate, roll
$\boldsymbol{\Omega}$	Array of line segments
$\boldsymbol{\mu}_{i+1}$	Innovation
$\mathbf{I}$	Information matrix
$\mathbf{P}$	Covariance matrix
$\mathbf{K}$	Kalman gain
$\mathbf{S}$	Innovation covariance
β	Mahanalobis distance

Chapter 1. Introduction

1.1. Overview

This chapter introduces the trajectory planning problems addressed in this thesis. The challenges involved in solving these problems are outlined and the principal contributions that have emerged from studying the trajectory planning problems are described. Subsequently, a list of publications resulting from these contributions is given. The structure of the thesis is then outlined along with a brief description of the content of each of the following chapters.

1.2. Trajectory Planning Problem

Trajectory planning is critical to almost all autonomous mobile robot applications. One class of trajectory planning problems considers the case where a robot is to travel from point A to B in a given map. Common objectives include determining the shortest path or the path of quickest time. Typical path planning algorithms that constitute this class of problems, such as A* (Nilsson 1980), are deterministic and assume the robot to be a point and then with a given map, the shortest path to the destination is planned. When the map is partially known, D* planning algorithms (Stentz 1995), a dynamic version of A*, can be applied where part of the path may be replanned as more knowledge of the environment is acquired. There exists a subtle difference between path planning and trajectory planning. A trajectory is time dependent and may include additional parameters such as robot velocities, turn-rate or acceleration, whereas a path is not.

The trajectory planning problem becomes increasingly challenging when a map is not available. As robot sets out to explore an unknown environment, it needs to create a map simultaneous to determining an appropriate path for exploration. This belongs to a class of trajectory planning problems where the robot is not given a specific destination, but must instead determine a time dependent path to best achieve a certain task. If the robot location is not available from an external source, it must also localise itself within the map being built. This is referred to in literature as Simultaneous Planning, Localisation and Mapping (SPLAM) (Meger 2006) or Active Simultaneous Localisation and Mapping (Active SLAM) (Zhang et al. 2006).

When either the map or the robot locations are unknown, the robot pose and map features must be estimated through the information obtained from sensors. These sensors could include laser, cameras, or radar. Algorithms such as the Extended Information Filter (EIF) (Grocholsky 2006), the Extended Kalman Filter (EKF) (Maybeck 1979) and Smoothing and Mapping (SAM) (Dellaert and Kaess 2006) may be used for the estimation. Inherent in the estimations are uncertainties and these must be accounted for when performing the trajectory planning.

Specific factors to consider in the trajectory planning may include the robot dimension, maximum turn-rates and velocities, obstacles in the environment, the uncertainty of the location of the robot and the task to be performed. In optimal trajectory planning, all possible constraints in the system must be considered and the trajectory should be determined to maximise the achievement of the task objectives.

In the case of multiple robots, their coordination needs to be considered for efficiency. Redundant actions or interference between robots occur when a robot repeats a task or performs actions in common with another robot. Clearly, the trajectory planning needs to consider the paths and tasks of all the robots.

In this thesis, three trajectory planning problems are considered. The first trajectory planning problem deals with multi-agent bearing-only target localisation. For this problem, the objective is to coordinate multiple robots with given constraints to localise multiple targets in the environment. The second trajectory planning problem addresses point-feature Active SLAM. Here, the task of the robot is to map an unknown environment containing point features. This is then extended to line-feature Active SLAM where an unknown structured environment is mapped.

1.3. Problems Addressed in this Thesis

In each of the three problems addressed, the following relevant objectives are:

Trajectory Planning for Multi-agent Bearing-only Target Localisation (Chapter 3)

- To obtain the optimal path to maximise the information of the locations of the targets within a prescribed terminal time.
- To coordinate multiple agents.
- To incorporate new information to gradually modify the trajectory plans.

Trajectory Planning for Point-feature based SLAM (Chapter 4)

- To obtain the optimal path to map an unknown environment within a prescribed terminal time.
- To minimise uncertainty of the robot pose and point feature locations of the map.
- To incorporate long term goals and coverage into the trajectory planning.

Trajectory Planning for Line-feature based SLAM (Chapter 5)

- To obtain the optimal path to map an unknown structured environment within a prescribed terminal time.
- To extract and represent lines from range and bearing observations.
- To minimise uncertainty of the robot pose and line feature locations of the map.
- To implement the trajectory planning algorithm developed on a physical mobile robot.

1.4. Principal Contributions

The main contributions of this thesis are:

- Trajectory planning for the tasks of feature-based localisation and mapping are formulated as an optimal control problem with a gradually identified model. The objective of the planning is to maximise information gain or to minimise uncertainty. However, not all of the information required for the trajectory planning can be predicted before observations are made and thus the system models are gradually identified.
- Model Predictive Control (MPC) is applied to the trajectory planning problems for feature-based localisation and mapping. It is a recursive planning strategy that exploits the updated information available at each step. An optimisation strategy, Exhaustive Expansion Tree Search (EETS), is used to select the optimal control input from a discrete set of control options. A comparison of this strategy to other algorithms is conducted.
- An attractor based strategy is developed to improve the performance of MPC for SLAM. The main motivations for this technique include enabling high-level planning, incorporating long term goals, and to provide an incentive for exploration using the framework of the MPC strategy.
- The trajectory planning problem for line feature SLAM is considered. The developed planning strategies are demonstrated on a physical robot to perform real-time line-feature based SLAM, estimated using SAM, in a structured environment.

1.5. Publications

The following publications document some of the contributions of this thesis and related work.

Leung, C. Huang, S. and Dissanayake, G. 2008 "Active SLAM for Structured Environments", *IEEE International Conference on Robotics and Automation* (in press).

Leung, C. Huang, S. and Dissanayake, G. 2006 "Active SLAM using Model Predictive Control and Attractor Based Exploration", *Proceedings of the 2006 IEEE/RSJ*

International Conference on Intelligent Robots and Systems, Beijing, China, pp. 5026-5031, October 2006.

Leung, C. Huang, S. Kwok, N. M. and Dissanayake, G. 2006 “Planning under Uncertainty using Model Predictive Control for Information Gathering”, *Robotics and Autonomous Systems*, Volume 54, Issue 11, November 2006, pp. 898-910.

Leung, C. Huang, S. Dissanayake, G. and Furukawa, T. 2005 “Trajectory Planning for Multiple Robots in Bearing-Only Target Localisation”, *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Canada, pp. 2312-2317, August 2005.

Leung, C. and Al-Jumaily, A. 2004 “A Hybrid System for Multi-agent Exploration”, IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2004), Budapest, July 2004.

Leung, C. and Al-Jumaily, A. 2003 “Combining Wavefront Propagation and Possibility Theory for Autonomous Navigation in an Indoor Environment”, *Australasian Conference on Robotics and Automation (ACRA 2003),* Brisbane, Dec 2003.

1.6. Thesis Structure

Chapter 2 discusses the trajectory planning problems in the presence of uncertainty. Several developed solutions in the literature for trajectory planning in the tasks of target localisation and SLAM are reviewed along with related literature. A planning strategy, MPC, is also reviewed along with theoretical issues regarding the stability and computational complexity and with several works applying this strategy to trajectory planning.

Chapter 3 considers the trajectory planning problem for multi-agent bearing-only target localisation. The objective of the trajectory planning is to cooperatively maximise the information gathered by the agents. The EIF is used to estimate the location of the targets of interest and the information gathered is quantified using the information matrix. Trajectory planning is formulated as an optimal control problem for a nonlinear system with a gradually identified model and then solved using nonlinear MPC. The optimisation in MPC is via an Exhaustive Expansion Tree Search (EETS) and is shown to provide a near optimal solution. MPC for target localisation is demonstrated in both two-dimensional and three-dimensional scenarios.

Chapter 4 addresses the Active SLAM problem in the case of point features using a single robot. The robot pose and location of point features in the environment are estimated using an EKF. The objectives of the trajectory planning is to minimise uncertainty of the SLAM estimates and to maximise coverage and efficiency. It is first shown that the Active SLAM can be formulated as an optimal control problem for a nonlinear control system with a gradually identified model and then solved using MPC. To improve the performance of MPC, a novel technique that adds an attractor to facilitate exploration is introduced.

Chapter 5 extends the work of Chapter 4 to trajectory planning for line-feature based SLAM. The robot poses and line features are estimated using SAM which is found to provide more consistent estimates than EKF. To reduce the number of poses in the state vector and to reduce the computation time, the SAM algorithm is modified to use the relative poses of the robot between observation steps. Trajectory planning is performed using the estimates from SAM and for computational efficiency MPC is performed using estimates predicted by an EKF. Again, MPC with the addition of the attractor is used to obtain trajectories that minimises the uncertainty of the estimates and maximises coverage. The trajectory planning is demonstrated in real-time on a Pioneer2DX robot.

Chapter 6 summarises the main contributions of this thesis and notes some directions of interest for future research.

Chapter 2. Literature Review

2.1. Introduction

This chapter reviews a number of selected works in literature on mapping and trajectory planning. General strategies available for planning of mobile robot information gathering tasks under uncertainty are introduced. The discussion then focuses on the relevant research and progress in trajectory planning for target localisation and for Simultaneous Localisation and Mapping (SLAM). Finally, relevant developments from the Model Predictive Control (MPC) literature are studied, which form the foundations of the strategy proposed to solve the planning problems considered in this thesis.

2.2. Strategies for Planning Under Uncertainty

Information gathering via observations is fundamental for many autonomous robot tasks such as target localization, exploration, mapping, search and SLAM. These processes are subject to uncertainty; about the environment, in the sensor measurement and in the process model. To enable an autonomous robot to optimally exploit the sensor data, planning for information gathering needs to effectively mitigate the adverse effects of these uncertainties.

Several techniques currently exist for planning under uncertainty. For example, uncertainties can be viewed as random variables or as a stochastic process. The field of stochastic control has popular strategies such as Markov Decision Processes (MDPs) and their extensions, Partially Observable Markov Decision Processes (POMDPs) (Kaelbling et al. 1998). Uncertainties can also be treated as deterministic signals (with a known hard bound or a known energy bound) which leads to nonlinear l^{∞} bounded robust control (Huang and James 2003) and nonlinear H^{∞} control (Helton and James 1999). All these problems are theoretically solvable and optimal solutions can be obtained by computing the value functions defined for the belief states (Kaelbling et al 1998) or the information state (Helton and James 1999, Huang and James 2003). Although these global planning strategies all aim at providing global optimal control policies, the curse of dimensionality and curse of history (Brooks et al. 2005) are major concerns, where the size of the problem and possible outcomes grows with time. Some decomposition and approximation methods are available to obtain solutions, such as via clustering (Li et al. 2005), implementing a hierarchical POMDP (Foka and Trahanias 2005) and using parameters to represent approximate continuous states (Brooks et al. 2005). Nevertheless it is still very difficult to make them tractable in real-time large-scale applications with existing computational platforms.

Given the prohibitive computational expense for obtaining optimal solutions, computationally tractable sub-optimal solutions are desired for real-time implementations. One such approach involves the use of a single-step look ahead planning strategy, which has been proposed for various information gathering tasks. Feder et al. (1999) used the expected information gain from taking a step in a given direction in their Active SLAM algorithm. Stroupe and Balch (2005) selected robot movements to maximise the utility of the reduction of uncertainty of the next observation in multi-robot mapping. A similar strategy for information gathering in vision and image processing is called the Next Best View (NBV) and is applied in (Li and Liu, 2005) for the task of three-dimensional object reconstruction.

In terms of resultant plan quality, multi-step planning is clearly an improvement over single-step strategies. Model Predictive Control (MPC), or Receding Horizon Control (Allgower and Zheng 2000), is a multi-step planning strategy which can also be used in information gathering tasks. MPC has found many applications due to its ability to incorporate constraints in the planning process, its limited computational requirements and its capability of allowing feedback at each step. Recently, MPC has been applied in trajectory planning for multi-robot planning and control under input/state constraints for collision-avoidance (Shim et al. 2003), regulation of nonholonomic mobile robots (Gu and Hu 2005), multi-agent formations (Wesselowski and Fierro 2003), as well as target tracking and engagement (Tierno 2001), and unmanned aerial vehicle (UAV) navigation with passive noisy sensing (Frew 2005). In this thesis, MPC is used as the strategy for planning trajectories in the tasks of target localisation and SLAM.

2.3. Trajectory Planning for Target Localisation

Bearing-only target localisation has a wide range of applications in both military and civilian areas (Ristic and Arulampalam 2003; Grocholsky 2006). In Grocholsky (2006), the problem of multi-robot target localization was considered and an EIF was applied to the estimation process. The trajectories were planned using partial utility measures involving the local knowledge and influence of an agent. For the optimisation, the solution was approximated using parameterisation of the control trajectories and then solved using Sequential Quadratic Programming (SQP). This work focused on sub-optimal decentralised control through negotiation.

Passerieux and Cappel (1998) applied optimal control theory to determine the trajectory of a constant speed bearing-only observer by minimizing the uncertainty of the target estimate as described by the inverse of the Fisher Information Matrix (FIM). The target was assumed to move in a straight line with constant velocity with the target location and velocity to be estimated. Several assumptions were made to simplify the problem. These include assumption of a perfectly omni-directional bearing-only sensor and no constraint on the observer maneuverability. To maximize the accuracy of the estimation, a hill climbing technique was used to assign the control and in addition to this, several arbitrary initial guesses of the control were used to avoid convergence to a local optimum.

The optimal robot trajectory for bearing-only localisation, when a single robot is locating a single target, can be computed off-line when the exact locations or the trajectories of the targets are known. Oshman and Davidson (1999) presented three such techniques for target localization. All three techniques maximised the determinant of the information matrix. Two of the methods were gradient based, where the initial guess is arbitrarily selected. Differential inclusion (Seywald, 1994) is used in the third technique, which handles constraints by employing a description of the dynamical system in terms of its states and their sets of attainability. This involved the generation of finite-dimensional approximations to the solutions, which effectively eliminate the controls from the system model. Oshman and Davidson concluded that directly maximising the information matrix is superior to the approximation methods.

A related work on decentralised multi-agent trajectory planning of multiple UAVs for multi-target surveillance was recently published (Tang and Özgüner 2005). The objective considered was to minimize the average time between consecutive observations of each target. The multi-agent motion planning was achieved by dividing the targets between the robots and then solving several single-agent problems. These agents are modeled as a nonholonomic Dubins car thus the roll of the UAV was not considered. Therefore it was assumed that when the agents turn, the direction of the sensor remains constant i.e. pointed perpendicular to the ground. The targets were treated as a set of waypoints with a given traverse order and the trajectory planning connects these waypoints using lines, arcs and circles such that each target is visited in sequence. Several heuristics are used to improve the flight time such as reversing the entry and exit points of turn sequences in the path to avoid the path crossing over itself. It was also assumed that the speed of the agents is much faster than the targets. Moreover, sensor noise was not considered.

2.4. Trajectory Planning in SLAM

Several works have focused on the Active SLAM problem. In Stachniss et al. (2004), a frontier based exploration with SLAM is conducted. A Rao-Blackwellized particle filter is applied to localisation and mapping. Using two maps; occupancy grid and topological, active loop-closing is performed. Active loop-closing is based on determining minimum distances between the current robot location and the old nodes in the topological map. The algorithm forces the robot to revisit previously traversed areas in order to reduce the uncertainty in the map, although maximisation of information gain along the path is not considered.

Feder (1999) used the single-step planning strategy for a point feature-based Active SLAM problem, where an action is chosen given the current knowledge so as to maximise the information gain in the next measurement. This approach is applied to air and underwater experiments with sonar sensing and the author showed that adaptive sensing can save time, energy and reduce the amount of data that needs to be acquired. Planning with a longer horizon could clearly improve results.

Makarenko et al. (2002), considered multiple objectives during the SLAM exploration process as a form of utility. To aid the process, an occupancy grid is used in the mapping and a frontier method is used to generate several potential destinations for the robot. A destination is selected based on the evaluation of several utility functions, namely: cost of navigation, information gain and localizability at the destination. The solution, however, may be improved by considering these functions along each step in the path instead of only at the destination.

Bryson et al. (2005) developed an on-line guidance scheme for maximising vehicle navigation and map information for an UAV. SLAM was performed using the EKF. The guidance system obtains a discrete set of possible destinations from a frontier cells in a grid map. To facilitate exploration, a single feature is assumed to be at the destination and is given a large uncertainty. The utility of travelling to a destination was then determined by predicting the information gain from simulated observations along the path to the destinations. Behaviour-based decision rules were developed based on qualitative knowledge of the effect of manoeuvres on observability (Kim and Sukkarieh 2004). Using the information utility and decision rules, a trajectory consisting of lines and arcs to the destination was assigned. This provided a suboptimal solution in terms of navigation performance that is computationally practical.

The closest related work to that presented in this thesis is EKF based Active SLAM by Sim and Roy (2005), where an A-optimal global planning strategy for SLAM was proposed. It was pointed out that, in SLAM, the trace of the covariance matrix is a better measure of map quality than its determinant. An assumption was made that the approximate locations of all the features are available at the beginning, thus replanning was not critical. The planning algorithm does not take into account the robot dynamics or sensor limitations. A global solution for the path is determined for an environment discretised into a grid. This global solution was made feasible by using the estimated position of the robot and the trace of the covariance matrix during planning, which allows far lower computational cost but contains less information. Sim (2005) also encouraged a robot to explore by randomly placing virtual features in unexplored areas. A Voronoi graph was used for the path planning; with assumptions of perfect data association and an unlimited sensor field of view. Similarly, in Sim (2005a), an approach was presented for information-driven SLAM. The robot is driven to a globally optimal position for maximising information gain of the features. This work was for bearing-only SLAM and the main focus was to overcome the stability issues. Features that were too close to the robot which may cause filter instability were blocked by using a virtual minimum range sensor.

2.5. Related work on Model Predictive Control (MPC)

MPC, also referred to as Receding Horizon Control, is applied as the control strategy for the trajectory planning problems addressed in this thesis. In general, MPC is a recursive strategy where multiple control steps are planned and only the first step is executed. In information gathering tasks, observations from sensors are influenced by the control action and they involve uncertainty and disturbances. This makes feedback essential. The recursive characteristic of MPC makes it suitable for information gathering tasks where the knowledge of the environment is gradually improved.

2.5.1. Theoretical aspects of Nonlinear MPC (NMPC)

Local planning strategies, such as MPC, are well suited to systems with hard real-time constraints and systems with highly dynamic environments. MPC is simple and can be fast, thus enabling continuous replanning to incorporate feedback and up to date knowledge of the system state and the environment.

There have been over 2000 reported applications of Linear MPC, making it the industry standard in some areas. The use of non-linear MPC (NMPC) is also accelerating, from 1997-2000 the reported applications of NMPC increased from one to eighty six (Allgöwer and Zheng 2000). One of the main reasons for the popularity of MPC is its ability to incorporate hard constraints that are difficult to handle by other methods.

However, the real applications of nonlinear MPC (NMPC) are still limited as compared to linear MPC; one of the major difficulties lies in the stability issue (Allgower and Zheng 2000). It has been illustrated by Bitmead et al. (1990) that for non-linear systems, closed-loop stability is not guaranteed for a generic finite-horizon cost function as the local minima found within the planning horizon may not lead to the global minima at the terminal time. Stabilisation methods including the dual-mode receding horizon controller, the infinite horizon closed loop costing, the quadratic terminal penalty and the contractive constraint have been proposed (Allgöwer and Zheng 2000).

For the information gathering tasks, however, the stability is not critical when the features are stationary. In general, the only concern here is the stability of robot motion and the convergence of the estimation algorithms. The stability of the robot motion should be guaranteed as long as the control actions are admissible and the obstacles are avoided. The convergence of the SLAM algorithm is generally guaranteed for any robot motion (Dissanayake et al. 2001). For the multi-robot target localisation problem, the information about the targets can only be increasing (resulting in smaller uncertainties) when observations are made. Thus the uncertainty of the target location estimates is always bounded for arbitrary robot trajectories.

Large computational burden creates another practical issue that limits the applications of NMPC. Generally, the computational cost grows exponentially with the increase of the planning horizon or the number of control options. Several techniques can be used to reduce the computational complexity in the optimization process. To provide an example, one can compute the first control, which is implemented exactly, while approximating the remaining controls, which are not implemented, see (Zheng 1997).

2.5.2. Application of MPC to Mobile Robot Trajectory Planning

Several works have applied MPC to mobile robot trajectory planning. In Bellingham et al. (2002), a mobile robot is set to travel from one side of a two-dimensional map to a destination point on the other side, avoiding obstacles. It is demonstrated that MPC can lead to the robot being trapped in dead-ends due to the limited planning horizon. A strategy is introduced that gives penalties for undesirable paths in the MPC solution using a precomputed cost-map over the known map. The time-optimal cost map is generated using lines connecting edges of obstacles to the goal and searched using Dijkstra's algorithm. Mixed-Integer Linear Programming (MILP) is used as the optimisation strategy. The results show near time-optimal paths that avoid entrapment. However, noise in the system is not considered.

In (Mettler and Toupet 2005), trajectory planning is conducted for a single robot in a known three-dimensional environment using MPC. The robot is intended to traverse from its current position around obstacles to a specified destination point. MILP is used as the optimisation strategy. A cost-to-go function is computed off-line that decomposes the environment into obstacle free cells and then the graph representing the connectivity of the cells is searched for the shortest path to the destination, using dynamic programming. The cost-to-go function is incorporated with the MPC optimisation. The authors state that local-minima in the cost-to-go map may result in the robot becoming trapped.

Kuwata et al. (2006) presented a solution to the decentralised control of multiple homogeneous UAVs using MPC. A cost-to-go function is generated for intelligent trajectories around obstacles. In this function, a shortest path algorithm is applied to a graph-based representation of the environment to approximate the cost to fly from each obstacle corner to the destination. Vehicles exchange information with neighbouring vehicles and collisions are detected by checking for the overlap of loiter-circles in the UAV paths. It is argued that the robust feasibility of the entire fleet of UAVs can be guaranteed using only local knowledge of the environment and other vehicles. The optimisation technique is also performed using MILP. Although the disturbance in the vehicle model is considered, the map is entirely known.

In all three of the aforementioned works, the robots were set to travel from their current point to a predefined goal in a known map. MPC is applied and a cost-to-go function is incorporated to provide a good estimate of the path beyond the planning horizon. In each case, the uncertainties present in the measurements is not considered.

Frew (2005) proposed to use MPC in several different problems, including UAV waypoint navigation, trajectory following, safe exploration of unknown environment and aircraft see-and-avoid. The planning technique was developed for a single robot using a two-dimensional model. Constraints were set in the cost function with a large weight and the optimization technique generated the candidate control input set using breath-first random search. The cost of fuel, navigation, information ignorance and safety are all considered in the cost function. The trace of the covariance matrix is minimised as this is claimed to be the simplest scalar measure to compute. The robot location uncertainty is not considered in this work.

2.6. Summary

The literature reviewed addresses trajectory planning problems for target localisation and mapping. Current optimal control strategies such as MDP and POMDP can be applied to systems with uncertainty but are computationally expensive. MPC has been applied to trajectory planning problems where the information to be acquired beyond the planning horizon can be approximated by computing a cost-to-go function for a known map. The trajectory planning problems considered in this thesis requires a simple algorithm that can be implemented in real-time without the need for a known map. Chapter 3 of this thesis makes a contribution in that a strategy is developed for using MPC in multi-agent information gathering tasks where robot cooperation and the uncertainty of the system are predominant aspects of the tasks. For such tasks, the destination point is not defined and therefore creating a cost map or cost-to-go function is difficult. To further improve the performance of MPC, Chapter 4 introduces an attractor to encapsulate possible information available beyond the planning horizon and to allow for multiple objectives of the planning. This is applied to information gathering tasks where the robot localisation and feature detection are also considered. The following chapters describe how MPC can be used to; a) plan trajectories for multi-agent bearing-only target localisation, b) plan trajectories for active SLAM with point features, and c) plan trajectories for active SLAM with line features.

Chapter 3. Trajectory Planning for Multi-Agent Bearing-only Target Localisation

3.1. Introduction

This chapter is concerned with planning the trajectory for multiple agents attempting to locate targets using bearing-only information. The objective of the trajectory planning is to cooperatively maximise the information gathered by these agents. The Extended Information Filter (EIF) is used to estimate the location of the targets of interest and the information gathered is quantified using the information matrix.

The trajectory planning problem is formulated as an optimal control problem for a nonlinear system with a gradually identified model and then solved using nonlinear Model Predictive Control (MPC). A solution to the MPC optimization problem is provided using an Exhaustive Expansion Tree Search (EETS). Sequential Quadratic Programming (SQP) is used to refine the solution obtained from an EETS. MPC for target localisation is demonstrated in both two-dimensional and three-dimensional scenarios for locating targets on the ground with a camera mounted on multiple UAVs. The proposed methods are analysed and evaluated through simulations.

3.2. Background

Cooperation among a team of distributed agents enables tasks such as target search, localisation, tracking and engagement to be performed with greater efficiency than a single robot (Bourgault et al. 2004; Passino et al. 2002; Ristic and Arulampalam 2003; Furukawa et al. 2003). The accuracy of information available about the targets determines which task is to be performed. If the targets are not visible and the area within which the targets are present is known, the primary task is to search. If the targets are moving and observable, the primary task is target tracking. If the target location is known and the target is approachable then the primary task may be engagement. If the targets are stationary and the locations of the targets are partially known, then the primary task of the robots becomes target localisation (sometimes referred to as geolocation). This chapter focuses on multi-agent cooperation for target localisation.

One of the practical situations that motivate the work presented in this chapter is the scenario where the approximate locations of targets are available through satellite surveillance, for example, and the precise target location estimates are to be acquired through cameras mounted on a set of UAVs.

When a sensor on board the robot measures the relative bearing to the target without any range information, as in the case of a camera, the robot trajectory has a significant influence on the accuracy of the target location estimate. In many applications, the location of the targets needs to be estimated on-line and only the estimates of the target locations are available for computing the robot trajectories. With bearing-only localisation, the initial estimates of target locations may be fairly inaccurate. Undoubtedly these location estimates are gradually updated as more measurements are acquired. Given these considerations, the development of real-time optimal trajectory planning algorithms for multi-robot target localisation is a challenging problem.

3.3. Target Localisation Problem

In target localisation (Grocholsky 2006), the task is to localise the targets in an environment as quickly as possible from a sequence of observations.

Suppose there are N robots and J targets. The target localisation problem is to estimate the locations of the J targets using the information obtained from a sequence of observations made by the N robots. The sensors on the robots have limited viewing range, scope and precision and the targets are assumed to be stationary and can be abstracted as points in space, as illustrated in Figure 3-1. It is assumed that the initial estimates of the target locations are available and that the location of the robot is provided by an external source such as GPS. It is also assumed, for simplicity, that collision between the robots will not occur (for example, the robots may be a set of UAVs flying at different altitudes).

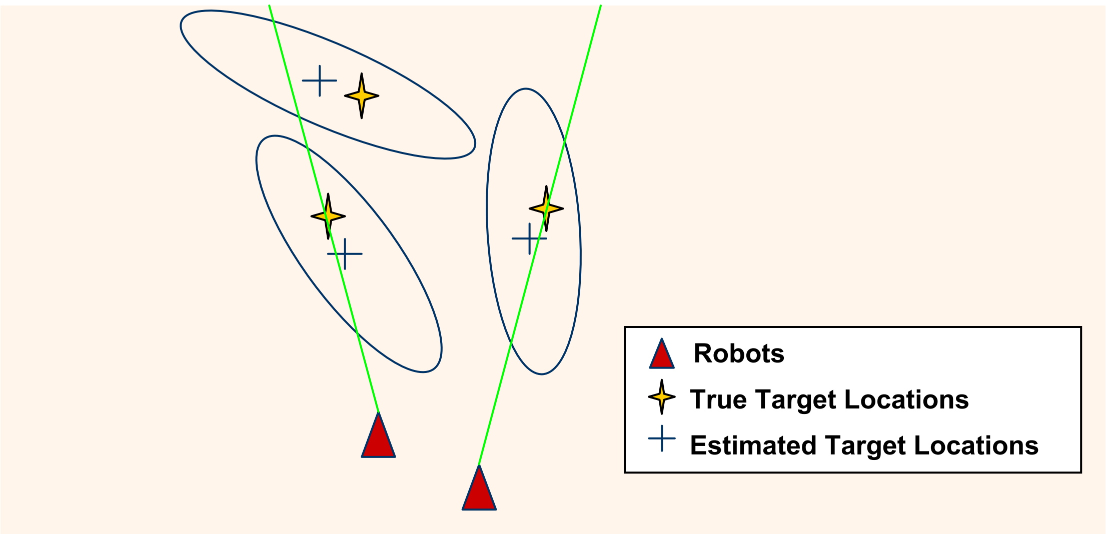


Figure 3-1 Problem Scenario, Multi-agent Bearing-only Target Localisation

3.3.1. Robot Process Model

Let $\mathbf{x}_i^n$ denote the robot pose of robot n at time i in a Cartesian coordinate frame. The discrete-time process model of the n^{th} robot is

$$\mathbf{x}_{i+1}^n = f_n\left(\mathbf{x}_i^n, \mathbf{u}_i^n\right) + w_{\mathbf{x}} \tag{3.1}$$

where $\mathbf{u}_i^n$ is the control input at time i and is constant during $[i; i+1)$, $w_{\mathbf{x}}$ is the process noise and f_n is a nonlinear function which depends on the dynamic model of the n^{th} robot, which may vary for heterogeneous robots.

Constraints of the robot's motion are to be incorporated into the planning process. The control constraints and state constraints can be expressed by

$$\mathbf{u}_i^n \in U_i^n,\ \mathbf{x}_t^n \in R_{i+1}^n,\ t \in [i, i+1) \tag{3.2}$$

where U_i^n is the set of admissible controls for the robot n at time i and R_{i+1}^n describes the safe region for robot n during time i to time $i + 1$.

3.3.2. Bearing-only Observation model

Following a discrete-time model, observations are made at each time step in a finite time horizon [0, T] where T is a given integer. The observation model, at time $i+1$, for the n^{th} robot observing the j^{th} target is

$$\mathbf{z}_{i+1}^{jn} = h_n(\mathbf{x}_{i+1}^n, \boldsymbol{\lambda}^j) + w_{\mathbf{z}}^n \tag{3.3}$$

where $\mathbf{z}_{i+1}^{jn}$ is the angular measurement from the n-th robot to the j-th target at time-step i+1, $\boldsymbol{\lambda}^j$ defines the location of the j-th target in a Cartesian coordinate frame, $w_{\mathbf{z}}^n$ is a zero-mean Gaussian noise measurement with covariance matrix $\boldsymbol{\Sigma}_n$ and h_n is a nonlinear function which describes the model of the sensor mounted on the n^{th} robot. Here, $j \in J_{i+1}^n$ where J_{i+1}^n denotes the set of the indices of the features that robot n can observe at time i+1, which is expressed by

$$J_{i+1}^n = \left\{ j_1, \ldots, j_{K_{i+1}} \right\}, \tag{3.4}$$

where the integer K_{i+1} depends on the pose of robot n at time i+1 and the range of the sensor.

3.4. Algorithm for Target Localisation

3.4.1. The Extended Information Filter

The observations made from various robot poses can be processed using an Extended Information Filter (EIF) (Grocholsky 2006; Maybeck 1979; Thrun et al. 2004) and compiled into estimates of the target locations. Theoretically, the EIF is equivalent to the Extended Kalman Filter (EKF). However, the EIF is computationally more efficient than the EKF when there is no prediction step. For the target localisation problem addressed in this chapter, the poses of robots are available and the targets are assumed to be stationary, thus a prediction step in the EIF is not required. For this reason, the EIF is used here as the underlying estimator.

3.4.2. New Information Obtained from Observations

Let the state estimate of the j^{th} target location at time i be $\hat{\boldsymbol{\lambda}}_i^j$ for j = 1, …, J. The information on the target location obtained from the observation $\mathbf{z}_{i+1}^{jn}$ can be represented by the information matrix

$$\mathbf{I}_{i+1}^{jn} = (\mathbf{H}_i^{jn})^{\mathrm{T}} \boldsymbol{\Sigma}_n^{-1} \mathbf{H}_i^{jn} \tag{3.5}$$

where $\mathbf{H}_i^{jn}$ is the Jacobian of h_n with respect to the target state $\boldsymbol{\lambda}_i^j$, evaluated at $(\mathbf{x}_{i+1}^n, \hat{\boldsymbol{\lambda}}_i^j)$. That is

$$\mathbf{H}_i^{jn} = \nabla_{\boldsymbol{\lambda}_j} h_n \,|_{(\mathbf{x}_{i+1}^n, \hat{\boldsymbol{\lambda}}_i^j)} \,. \tag{3.6}$$

If target j is out of the sensor range of robot n at time $i+1$, then no new information about the target is obtained from the observation and $\mathbf{I}_{i+1}^{jn}$ is returned as a zero matrix.

3.4.3. Target Location Estimate using Multi-agent Data Fusion

Since N independent observations (from the N robots) are made to target j at time $i+1$, the new information obtained about this target is

$$\mathbf{I}_{i+1}^{new,j} = \sum_{n=1}^{N} \mathbf{I}_{i+1}^{jn} \, . \tag{3.7}$$

The total information on target *j* at time $i+1$, after acquiring these observations, $\mathbf{I}_{i+1}^{j}$, can be calculated by

$$\mathbf{I}_{i+1}^{j} = \mathbf{I}_{i}^{j} + \mathbf{I}_{i+1}^{new,j} \tag{3.8}$$

where $\mathbf{I}_{i}^{j}$ is the total information available on the target *j* at time *i*.

At time *i* and $i+1$, the estimates of the locations of target *j* are denoted as $\hat{\boldsymbol{\lambda}}_{i}^{j}$ and $\hat{\boldsymbol{\lambda}}_{i+1}^{j}$, respectively. The update of the location estimate takes the form

$$\hat{\boldsymbol{\lambda}}_{i+1}^{j} = \hat{\boldsymbol{\lambda}}_{i}^{j} + (\mathbf{I}_{i+1}^{j})^{-1} \sum_{n=1}^{N} \mathbf{i}_{i+1}^{jn} \tag{3.9}$$

where

$$\mathbf{i}_{i+1}^{jn} = (\mathbf{H}_{i}^{jn})^{\mathrm{T}} \boldsymbol{\Sigma}_{n}^{-1} \mu_{i+1}^{jn} \, , \tag{3.10}$$

here $\mathbf{H}_{i}^{jn}$ is given by (3.6) and μ_{i+1}^{jn} is the innovation defined by

$$\mu_{i+1}^{jn} = \mathbf{z}_{i+1}^{jn} - h_{n}(\mathbf{x}_{i+1}^{n}, \hat{\boldsymbol{\lambda}}_{i}^{j}) \, . \tag{3.11}$$

Combining (3.5), (3.8), and (3.9), yields

$$\begin{aligned} \mathbf{I}_{i+1}^{j} &= \mathbf{I}_{i}^{j} + \sum_{n=1}^{N} \left(\mathbf{H}_{i+1}^{jn} \right)^{\mathrm{T}} \boldsymbol{\Sigma}_{n}^{-1} \mathbf{H}_{i+1}^{jn} \\ \hat{\boldsymbol{\lambda}}_{i+1}^{j} &= \hat{\boldsymbol{\lambda}}_{i}^{j} + \left(\mathbf{I}_{i+1}^{j} \right)^{-1} \sum_{n=1}^{N} \mathbf{i}_{i+1}^{jn} \end{aligned} \tag{3.12}$$

for *j* = 1,…, *J*, where

$$\begin{aligned}
&\mathbf{i}_{i+1}^{jn} = \begin{cases} \left(\mathbf{H}_{i+1}^{jn}\right)^{\mathrm{T}} \boldsymbol{\Sigma}_n^{-1} \mu_{i+1}^{jn}, & j \in J_{i+1}^n, \\ 0, & j \notin J_{i+1}^n, \end{cases} \\
&\mu_{i+1}^{jn} = \mathbf{z}_{i+1}^{jn} - h_n(\mathbf{x}_{i+1}^n, \hat{\boldsymbol{\lambda}}_i^j), \quad j \in J_{i+1}^n \\
&\mathbf{H}_{i+1}^{jn} = \begin{cases} \nabla_{\lambda^j} h_n \,|_{(\mathbf{x}_{i+1}^n, \hat{\boldsymbol{\lambda}}_i^j)}, & j \in J_{i+1}^n, \\ 0, & j \notin J_{i+1}^n, \end{cases} \\
&\mathbf{x}_{i+1}^n = f_n(\mathbf{x}_i^n, \mathbf{u}_i^n)
\end{aligned} \tag{3.13}$$

for $n = 1,\ldots, N, j = 1,\ldots, J.$

Equation (3.12) represents the EIF information and state updates for multi-robot geolocation. The EIF equations above are slightly different from the typical EIF equations (Maybeck 1979, Thrun et al. 2004) as the state estimate is obtained without computing the information vector. They are however equivalent.

3.5. Formulation of Trajectory Planning as an Optimal Control Problem

3.5.1. Problem Statement

In bearing-only target localisation, it is critical that the robots observe the targets from different viewpoints. The robot trajectories therefore play an important role in the effectiveness of information gathering. In this section, the trajectory planning problem is formulated on a finite time horizon [0, *T*]. The objective is to maximise the total information on the target locations at time step *T* by choosing suitable control actions for the *N* robots. In particular, the trajectory planning problem can be stated as follows.

Problem: *Suppose at time step 0, the pose of the n^{th} robot is $\mathbf{x}_0^n$, n=1…N. The initial location estimate of the j^{th} target is $\hat{\boldsymbol{\lambda}}_0^j$, j=1…J, and the information available on the target is $\mathbf{I}_0^j$, j=1…J. Compute the control actions, $\mathbf{u}_i$, for the N robots from time i=0 to time i=T–1,*

$$\mathbf{u}_0^n, \mathbf{u}_1^n, \ldots, \mathbf{u}_{T-1}^n \quad n = 1 \ldots N \tag{3.14}$$

such that

$$\min_{1 \le j \le J} \min(\mathrm{eig}(\mathbf{I}_T^j)) \tag{3.15}$$

is maximised, where $\mathbf{I}_T^j$ *is the information matrix of the* j^{th} *target at time T and is obtained by*

$$\mathbf{I}_T^j = \mathbf{I}_0^j + \sum_{i=0}^{T-1} \sum_{n=1}^{N} \mathbf{I}_{i+1}^{jn} \tag{3.16}$$

where $\mathbf{I}_{i+1}^{jn}$ is *given by* (3.5).

The inverse of the information matrix, i.e. the covariance matrix, describes the uncertainty. For a target location estimate, the 95% or 99% confidence bound can be represented as an ellipse with the longest axis representing the greatest uncertainty. In bearing-only localization, it is typical for the shape of the uncertainty ellipse of a target location to be long and narrow in the direction of the sensor reading due to the lack of range information. If the smallest eigenvalue of the information matrix is maximised, the largest axis of the uncertainty ellipse is minimised, which directly influences the direction of uncertainty. Therefore, the scalar from (3.15) is used as the objective function as it provides a good indication of the total information about the targets. Other scalar measures of information could also be used, such as the determinant or the trace of the information matrix. Note, however, that maximising the determinant of the information matrix would inherently minimise the area of the uncertainty ellipse of the target location estimate, which may still leave the target location highly uncertain in one direction.

3.5.2. Optimal Control Problem Formulation

The objective of the optimal control problem is to choose the control (3.14) in order to maximise the performance measure (3.15). At first glance, the trajectory planning problem is equivalent to a finite horizon optimal control problem for a nonlinear control system. The model of the “control system” is the compilation of equations (3.1), (3.5), (3.6), (3.8), (3.9), (3.10), and (3.11).

Note that in (3.5) and (3.8), $\mathbf{H}_i^{jn}$ is needed to calculate $\mathbf{I}_{i+1}^j$, and in (3.6) $\hat{\boldsymbol{\lambda}}_i^j$ is needed to calculate $\mathbf{H}_i^{jn}$, but to update $\hat{\boldsymbol{\lambda}}_{i+1}^j$ using (3.9), (3.10), and (3.11), the observation $\mathbf{z}_{i+1}^{jn}$ is required. However as the observation $\mathbf{z}_{i+1}^{jn}$ is not available until time $i+1$, a clear relationship between the control actions (3.14) and the performance measure (3.15) is not available at time 0. Because the value of $\mathbf{z}_{i+1}^{jn}$ is yet to be obtained until the next time step and the information from this observation contributes to the knowledge of the system model, the system model can only be updated when $\mathbf{z}_{i+1}^{jn}$ gradually becomes available at each iteration. Hence the above model is a *nonlinear control system with a gradually identified model.*

Seeing that the system is gradually identified, this problem cannot be solved by traditional control optimisation methods. In spite of this, the problem can be approximated as a traditional control problem by assuming:

Assumption I. The innovations at any time are zero. i.e.

$$\mu_{i+1}^{jn} = \mathbf{z}_{i+1}^{jn} - h_n(\mathbf{x}_{i+1}^n, \hat{\boldsymbol{\lambda}}_i^j) = 0 \tag{3.17}$$

for all $n = 1, \ldots, N, j = 1, \ldots, J, i = 0, \ldots, T-1$.

There is also an underlying assumption, as with any EIF implementation, that the innovation, $\mu_1^{jn} = \mathbf{z}_1^{jn} - h_n(\mathbf{x}_1^n, \hat{\boldsymbol{\lambda}}_0^j)$, is a random variable with zero mean, where the distribution of the true target location, $\boldsymbol{\lambda}^j$, is Gaussian with $\hat{\boldsymbol{\lambda}}_0^j$ as the mean. For this reason, the value of the innovations is assumed to be zero in Assumption I. It follows that the safe region R_{i+1} hence contains the safe region R_0.

Under Assumption I, the Jacobians present in (3.12) do not change and the state estimate remains the same, hence the system becomes deterministic and a solution can be computed before new information is obtained. The solution to this approximated problem, however, may be far from optimal. This is a key reason behind the motivation for using the MPC strategy, as it is a recursive strategy that continually incorporates new information into the planning process.

3.6. Nonlinear Model Predictive Control

In Model Predictive Control (MPC), given a planning horizon of *D* and information available at the current time, *D* control actions are obtained. Only the first control action is applied to the system, and at the end of the first time step, a new set of *D* control options are computed. This strategy is repeated at each time step *i* until $i = T$.

3.6.1. Prediction in D-step Optimisation

In the context of the problems considered in this thesis, an important step for MPC is to predict the maximal information gain after *D*-steps, for an arbitrary set of controls. At step 0, $\mathbf{x}_0^n$, $\hat{\boldsymbol{\lambda}}_0^j$ and $\mathbf{I}_0^j$ are known. Referring (3.1), (3.5), and (3.6), $\mathbf{x}_1^n$ are functions of the control input at time 0, $\mathbf{u}_0^n$, and hence $\mathbf{H}_0^{jn}$ are functions of $\mathbf{u}_0^n$. Therefore $\mathbf{I}_1^j$ is a function of $\mathbf{u}_0^n$. Note that $\hat{\boldsymbol{\lambda}}_1^j$ depends on the observation $\mathbf{z}_1^{jn}$ though $\mathbf{I}_1^j$ does not.

Information from the observation $\mathbf{z}_1^{jn}$ at time 0 is required in order to predict more than one step ahead. Although the real value of $\mathbf{z}_1^{jn}$ is unknown, Assumption I provides a mechanism for long term prediction.

3.6.2. *D*-step Optimisation Problem

Under Assumption I, the *D*-step optimisation problem becomes the following

***D*-step optimisation problem.** Given $\mathbf{x}_0^n$, $\hat{\boldsymbol{\lambda}}_0^j$ and $\mathbf{I}_0^j$, choose control

$$\mathbf{u}_1^n, \mathbf{u}_2^n, \ldots, \mathbf{u}_D^n, \ \ n = 1, \ldots, N \tag{3.18}$$

to maximise

$$\min_{1 \le j \le J} \min(\mathrm{eig}(\mathbf{I}_D^j)) \tag{3.19}$$

where $\mathbf{I}_D^j$ is given by

$$\begin{aligned}
\mathbf{I}_1^j &= \mathbf{I}_0^j + \sum_{n=1}^{N} \left(\mathbf{H}_1^{nj}\right)^{\mathbf{T}} \boldsymbol{\Sigma}_n^{-1} \mathbf{H}_1^{jn} \\
\mathbf{H}_1^{jn} &= \begin{cases} \nabla_{\boldsymbol{\lambda}^j} h_n \,|_{(\mathbf{x}_1^n, \boldsymbol{\lambda}_0^j)}, & j \in J_1^n \\ 0, & j \notin J_1^n \end{cases} \\
\mathbf{x}_1^n &= f_n(\mathbf{x}_0^n, \mathbf{u}_0^n) \\
&\vdots \\
\mathbf{I}_D^j &= \mathbf{I}_{D-1}^j + \sum_{n=1}^{N} \left(\mathbf{H}_D^{jn}\right)^{\mathbf{T}} \boldsymbol{\Sigma}_n^{-1} \mathbf{H}_D^{jn} \\
\mathbf{H}_D^{jn} &= \begin{cases} \nabla_{\boldsymbol{\lambda}^j} h_n \,|_{(\mathbf{x}_D^n, \boldsymbol{\lambda}_0^j)}, & j \in J_D^n \\ 0, & j \notin J_D^n \end{cases} \\
\mathbf{x}_D^n &= f_n(\mathbf{x}_{D-1}^n, \mathbf{u}_{D-1}^n)
\end{aligned} \tag{3.20}$$

for $n = 1,\ldots, N; j = 1,\ldots, J.$

Similarly, from time i to $i + D$, the D-step optimization problem is to choose control at time i,

$$\mathbf{u}_{i+1}^n, \mathbf{u}_{i+2}^n, \ldots, \mathbf{u}_{i+D}^n, \;\; n = 1, \ldots, N \tag{3.21}$$

to maximise

$$\min_{1 \le j \le J} \min(\mathrm{eig}(\mathbf{I}_{i+D}^j)) \tag{3.22}$$

where $\mathbf{I}_{i+D}^j$ is given by

$$\begin{aligned}
\mathbf{I}_{p+1}^j &= \mathbf{I}_p^j + \sum_{n=1}^{N} \mathbf{I}_{p+1}^{jn} \\
\mathbf{I}_{p+1}^{jn} &= \left(\mathbf{H}_p^{jn}\right)^{\mathrm{T}} \boldsymbol{\Sigma}_n^{-1} \mathbf{H}_p^{jn} \\
\mathbf{H}_p^{jn} &= \begin{cases} \nabla_{\boldsymbol{\lambda}} h_n \big|_{\left(\mathbf{x}_{p+1}^n, \boldsymbol{\lambda}_p^j\right)}, & j \in J_{p+1}^n \\ 0, & j \notin J_{p+1}^n \end{cases} \\
\mathbf{x}_{p+1}^n &= f_n(\mathbf{x}_p^n, \mathbf{u}_p^n)
\end{aligned} \tag{3.23}$$

for n=1,…, N, j=1,…, J, p=i,…, $i + D$.

Equations (3.21)-(3.23) describe a deterministic optimisation problem which can be solved by a number of optimisation techniques. Two optimization techniques implemented in this thesis are described below.

3.7. Solution to the *D*-step Optimisation Problem

Systematic techniques are required to solve the *D*-step optimal control problem and determine the sequence of control actions. There are many ways to search for the optimal control sequence. Two strategies, Exhaustive Expansion Tree Search (EETS) and Sequential Quadratic Programming (SQP), are described below.

3.7.1. Exhaustive Expansion Tree Search (EETS)

One of the solutions to this optimisation problem is to use an Exhaustive Expansion Tree Search (EETS) to conduct a coarse exhaustive search. EETS conducts a search among a limited number of control sequences. For each robot, let the number of possible control options it can take at each time step *i* be N_ω. Thus using (3.23), each robot *n* can move to N_ω different poses, $\mathbf{x}_{i+1}^n$, at time $i+1$ if they were to apply N_ω separate controls $\mathbf{u}_{i+1}^n$ over the period *i* to $i+1$. At the N_ω different poses $\mathbf{x}_{i+1}^n$, N_ω different $\mathbf{I}_{i+1}^{jn}$ matrices can be evaluated using (3.23).

For the *D*-step optimal control problem, different control options may be chosen from step *i* to step $i+D-1$. Thus the robot would have $(N_\omega)^D$ different control sequence options. The information gain for the $(N_\omega)^D$ options needs to be evaluated and compared, hence the computational cost of this strategy for a single robot is $O((N_\omega)^D)$. For multiple robots, each robot has $(N_\omega)^D$ different $\mathbf{I}_{i+D}^{jn}$ matrices at prediction step *D*. Using (3.23), $(N_\omega)^{DN}$ combinations of $\mathbf{I}_{i+D}^{jn}$ are available to evaluate $\mathbf{I}_{i+D}^{j}$. The computation cost for considering all combinations is exponential, $O(J\times(N_\omega)^{DN})$.

Only feasible and obstacle free control actions can be chosen to maximise measure (3.22). Constraints representing no-go-zones, depicted by the green circle in Figure 3-2, can be enforced by an explicit condition based on predicted target and robot locations. Branches containing infeasible control actions can be removed from the tree search accordingly, as illustrated in Figure 3-2.

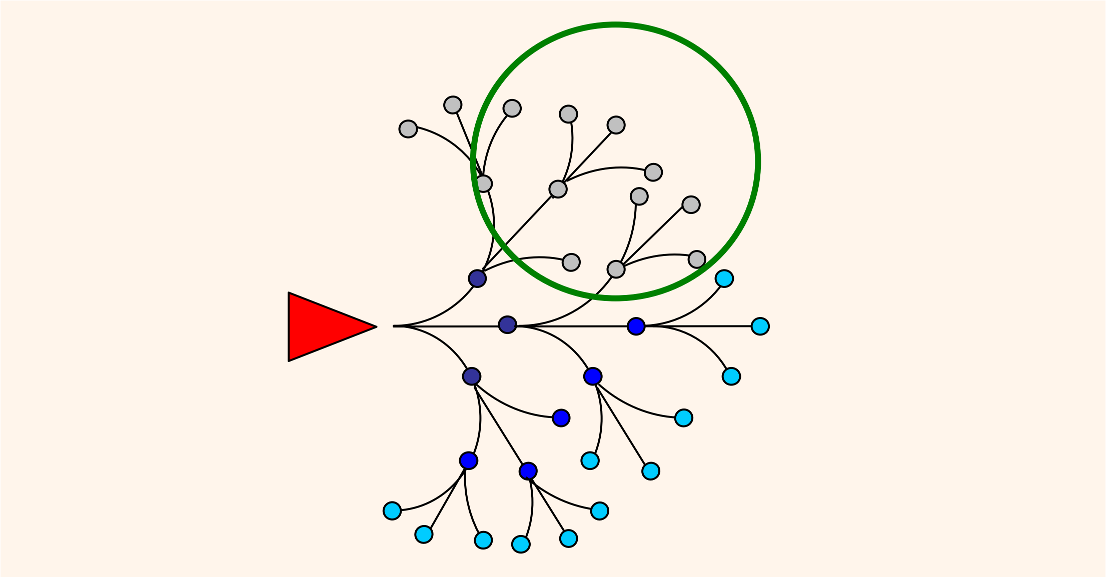
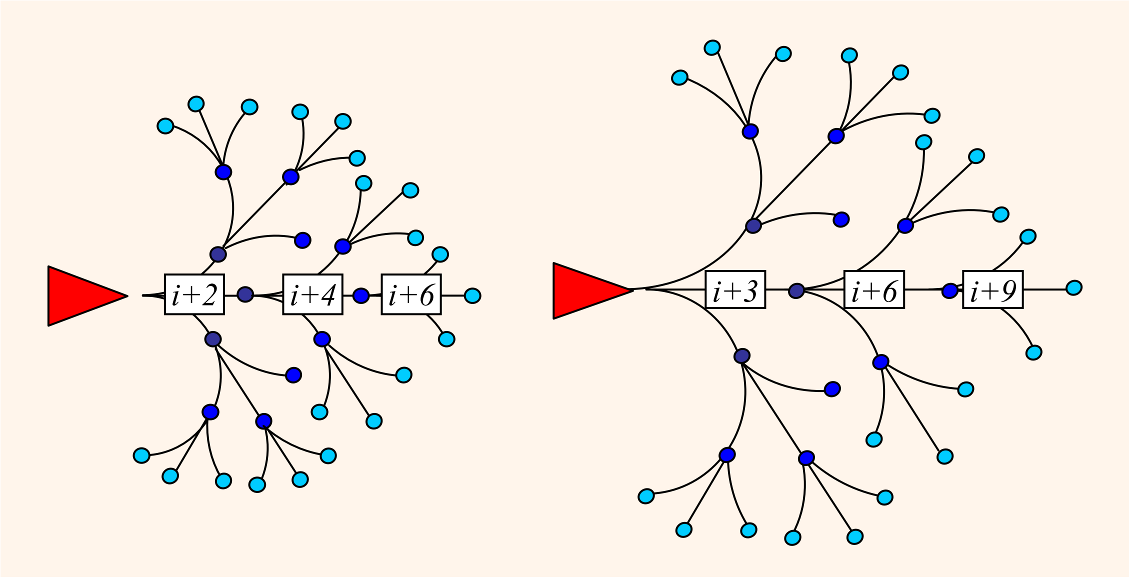


Figure 3-2 Branches through No-go-zone Removed from Search

Since no information is obtained when the targets are out of the robot's sensor range (as described in Section 3.4.2), it may be possible for a robot *n* to not gain any information even after looking at all possible trajectories in *D* steps. Therefore,

$$\sum_{j=1}^{J} \mathbf{I}_{i+D}^{jn} = zero\,matrix\,. \tag{3.24}$$

In this case, it is beneficial to predict further again if the computational capacity allows. However, if the plan is recomputed with an increased planning horizon, the computation load will increase exponentially. The following strategy may be applied in this situation. The planning horizon *D* is doubled but the number of control options is reduced by keeping a control option unchanged for two steps (e.g. [*i*; *i* + 2)) instead of one step (e.g. [*i*; *i* + 1)) as illustrated in Figure 3-3. Through this approach, the information gain is also evaluated every two steps and the computational cost remains the same (although the total planning time is doubled). As with the previous case, the control for only one step $i+1$ is applied before the above procedure is repeated.



(a) First repeat (b) Second repeat

Figure 3-3 Expanding the Tree Search

Although this is a coarser search, it allows the robot to look further ahead and provides it with an idea of the direction to head at the next step $i+1$, without severely burdening the processor. If no information can be obtained by doubling the planning horizon, the same strategy can be repeated with a larger step size. Alternatively, as all possible controls are equally optimal when there is no information gain, a feasible control may be randomly selected if the computation required does not allow for another replan.

3.7.2. Refining the EETS Solution using Sequential Quadratic Programming

The benefit of an exhaustive search is that it finds the global optimum among the finite control options. However, it can be argued that a coarse exhaustive search would not obtain the optimal solution because only a few discrete options are considered. An alternative method for the *D*-step optimisation is presented in this section. In this method, Sequential Quadratic Programming (SQP) is used to refine the results from EETS.

The SQP algorithm is a generalization of Newton's method for unconstrained optimization that finds a point near the current guess to the optimum point by minimizing a quadratic model of the problem. For each iteration, a positive definite quasi-Newton approximation of the Hessian of the Lagrangian function is calculated using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method and a quadratic programming (QP) problem is solved. The BFGS method is a method to solve an unconstrained nonlinear optimization problem and is derived from the Newton's method in optimization. Newton's method is a class of hill-climbing optimization techniques. It assumes that the objective function can be locally approximated as a quadratic in the region around the optimum and uses the first and second derivatives to find the stationary point, where the gradient is zero. The constraint functions are replaced by their linear approximations.

Unlike EETS, SQP is an optimization method based on hill climbing and may be trapped in a local optimum solution. Consequently, giving SQP a random point as the initial guess would not result in a good performance. Instead it is proposed for the system to be given the control sequence (3.21) for each robot that was obtained from EETS. This is such the SQP method is given an initial condition that is substantially close to the global optimum solution and the coarseness of EETS can be refined to solve the same objective function (3.22). In Figure 3-4, the optimum solution from SQP is obtained, depicted by the red circle, by passing it an initial guess from EETS. The initial guess is depicted by the light blue asterisk on the left.

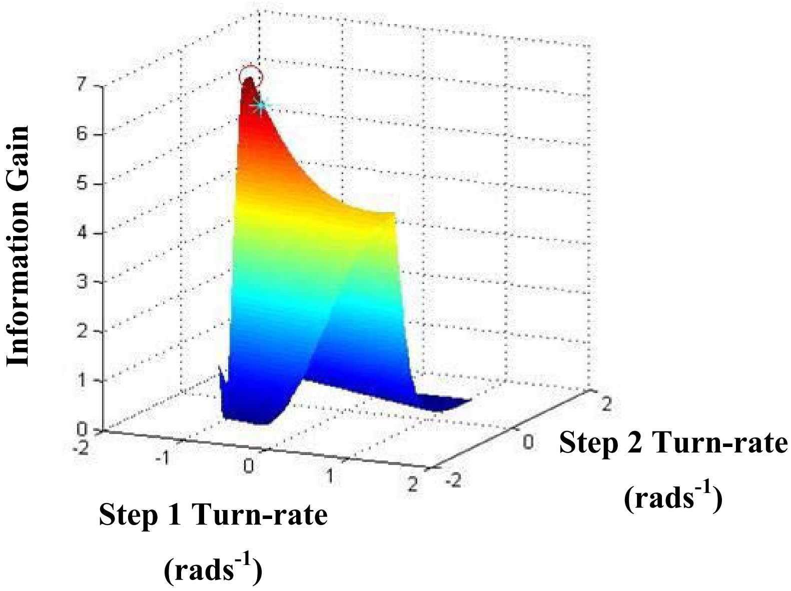


Figure 3-4 Optimum Solution Found by SQP with Initial Guess from EETS for a Single robot in a Two-dimensional Environment with Multiple Targets with *D*=2

3.7.3. A Decentralised Implementation

The robots are able to cooperate in maximising the objective function in the centralised approach presented above. However, cooperation is attained at the expense of greater computation time which increases exponentially with the number of robots. To demonstrate the value of cooperation, a simple decentralised framework for trajectory planning is implemented and compared (see Section 3.8.1.4). In this framework, the control actions of each robot are determined separately, ignoring the information gain through other robots. By using a decentralised framework, a lower computation time can be maintained by greatly sacrificing the optimality. In practice, a decentralised strategy should not be naively implemented. Strategies of cooperation such as computing additional negotiation strategies (Grocholsky 2006) are necessary for effective planning, however these are not explained further as improving upon decentralised approaches is not a focus of the research of this thesis.

3.8. Simulation Results for Target Localisation using MPC

Trajectory planning for target localisation is demonstrated through a computer simulation in both two-dimensional and three-dimensional scenarios. For the first case, two robots are to observe two targets in a two-dimensional environment. The initial locations of the targets have a large uncertainty representing approximate locations of the targets obtained from satellite surveillance. The robots are to localise these targets using bearing only sensors. To illustrate the ability of the proposed algorithm to enforce state constraints, a no-go-zone around the targets is enforced. Results obtained using EETS with three control options are compared to those obtained using EETS+SQP, EETS with nine control options and Decentralised EETS with three control options. The significance of using the full information matrix during the problem formulation is also demonstrated.

For the three-dimensional case, two UAVs are to localise three targets on the ground. These UAVs are equipped with a camera where the sensor field of view is dependent on the roll of the UAV. The no-go-zone constraint around the targets in this case is dynamic as it is dependent on the uncertainty of the target location. The more accurate the target location, the closer the UAVs may move towards the targets to make observations. The trajectory planning with MPC is compared to fixed control.

3.8.1. Trajectory Planning for Two Robots Observing Two Targets in a Two-dimensional Environment

This section provides a solution to the scenario depicted in Figure 3-1. Two robots are equipped with a bearing-only sensor with a field of view of ±π/2rad across the front of the robot, with a maximum range of 20m. The robots are set to move at a constant speed of 2ms^{-1} with a maximum turn-rate of π/6rads^{-1}. No-go-zone constraints of radius 2m around the targets are applied. Initial conditions $\mathbf{x}_0^n$, $\hat{\boldsymbol{\lambda}}_0^j$ and $\mathbf{I}_0^j$ are given manually and $\mathbf{u}_i^n$ is selected from a 3-step look-ahead ($D=3$) optimisation. The initial positions of the robots and features are place in an open environment of 900m^2 as seen in Figure 3-5, however, the motion of the robots are not restricted to the boundaries.

3.8.1.1. Process Model

The robot poses are predicted using the following model,

$$\mathbf{x}_i^n = [x_i; y_i; \phi_i] \tag{3.25}$$

$$\mathbf{x}_{t+\delta}^n = f_n\left(\mathbf{x}_t^n, \mathbf{u}_i^n\right) + w_{\mathbf{x}} \tag{3.26}$$

$$f_n = \begin{bmatrix} x_t^n + v_i^n \Delta t \cos(\phi_t^n + \omega_i^n \Delta t) \\ y_t^n + v_i^n \Delta t \sin(\phi_t^n + \omega_i^n \Delta t) \\ \phi_t^n + \omega_i^n \Delta t \end{bmatrix} \tag{3.27}$$

where $t = i, i+\delta, \ldots, i+1$ and Δt is the actual time difference between successive pose predictions. $\mathbf{x}_i^n$ is the pose of the nth robot at time *i* and $\mathbf{u}_i^n = [v_i^n, \omega_i^n]$ is the control input for the nth robot consisting of velocity and turn-rate and is constant from time *i* to time *i*+1. For the problem of geolocation, it is assumed that the robot locations are provided by an external source and hence is not estimated. Due to this assumption it is also assumed that the process noise $w_{\mathbf{x}}$ is zero for the multi-step predictions. This process model for the robot motion is iterated 10 times (δ=1/10) for every step *i*.

3.8.1.2. Observation Model

For the following bearing-only observation model, the state of the target is described by $\boldsymbol{\lambda}^j = [x^j; y^j]$ and the observation, $\mathbf{z}_i^{jn}$, is described by

$$\mathbf{z}_i^{jn} = \theta = \text{atan2}\left(\frac{y^j - y_i}{x^j - x_i}\right) + w_{\mathbf{z}}^n \tag{3.28}$$

where θ is the bearing from the robot to the target and $w_{\mathbf{z}}^n$ is the measurement noise with a standard deviation of $\sigma_{\mathbf{z}}^n = 0.03\,\text{rad}$.

3.8.1.3. Comparison of Different Strategies

Trials were conducted to assess computation times required and information gain achieved for the optimisation strategies proposed in Section 3.7. The first two columns of Table 3-1 compare EETS with three control options (N_ω=3) and the effect of refining this result using SQP. Computation times in brackets are for an implementation of the algorithm in Matlab on a 2GHz Pentium IV.

Table 3-1 demonstrates that the combination EETS+SQP often performs slightly better in terms of information gain than EETS alone. However, there are often instances where the information gains are equal (trials 1-4). There is a fundamental reason for this counter intuitive result. In the case of bearing-only localisation, often the highest information gain can be obtained by taking an observation at an angle that is perpendicular to the previous observation. To maximise the information gain, the robots would therefore often be driven at the maximum turn-rate to circumnavigate the targets. In fact, in the 200 trials undertaken, the resultant control actions were evaluated to be at the maximum turn-rate 70 percent of the time.

Although EETS+SQP was found to have slightly better information gain (on average 1.3% better than EETS N_ω=3), it has a significantly longer computation time as shown in the bracketed valued in the table. The additional computation time of SQP is dependent on how far the initial condition is from the optimal, as evident in Trials 5-10. Thus, the coarse EETS was found to produce an adequate solution without the additional computation of SQP.

Trial	**– Optimal Control Strategy –** Information Gained (Computation Time (s))			
	EETS N_ω=3	EETS (N_ω=3) + SQP	EETS N_ω=9	EETS (N_ω=3) Decentralised
1	**3.5947** (0.12)	**3.5947** (0.26)	**3.5947** (39.72)	2.0748 **(0.07)**
2	**2.2478** (0.13)	**2.2478** (0.27)	**2.2478** (39.76)	1.4044 **(0.08)**
3	**7.1881** (0.07)	**7.1881** (0.22)	**7.1881** (8.95)	6.8754 **(0.05)**
4	**5.2361** (0.09)	**5.2361** (0.24)	**5.2361** (25.16)	4.2059 **(0.06)**
5	0.3866 (0.12)	**0.3869** (3.06)	0.3868 (36.96)	0.2681 **(0.07)**
6	1.3465 (0.12)	**1.3696** (5.53)	1.3680 (36.88)	0.2652 **(0.07)**
7	1.4133 (0.11)	**1.4569** (4.50)	1.4530 (29.42)	0.9449 **(0.07)**
8	0.9559 (0.08)	**1.0025** (4.63)	0.9929 (11.89)	0.5803 **(0.05)**
9	1.3003 (0.20)	**1.3109** (4.95)	1.3082 (74.59)	0.7745 **(0.08)**
10	2.0831 (0.08)	**2.1418** (0.37)	2.1405 (21.10)	1.5059 **(0.06)**

Table 3-1 Information Gained and Computation Times of MPC

where Information Gained = $\min_{1\le j\le J} \min\left(\text{eig}\left(\mathbf{I}_{i+D}^{j}\right)\right)$

Trials were also conducted to compare EETS with nine control options (i.e. N_ω=9) instead of N_ω=3. The results of the finer search are shown in the third column of Table 3-1; the computation time has increased significantly (approx. 300 times greater) without achieving comparable improvements in information gain.

3.8.1.4. Results from a Naive Decentralised Implementation

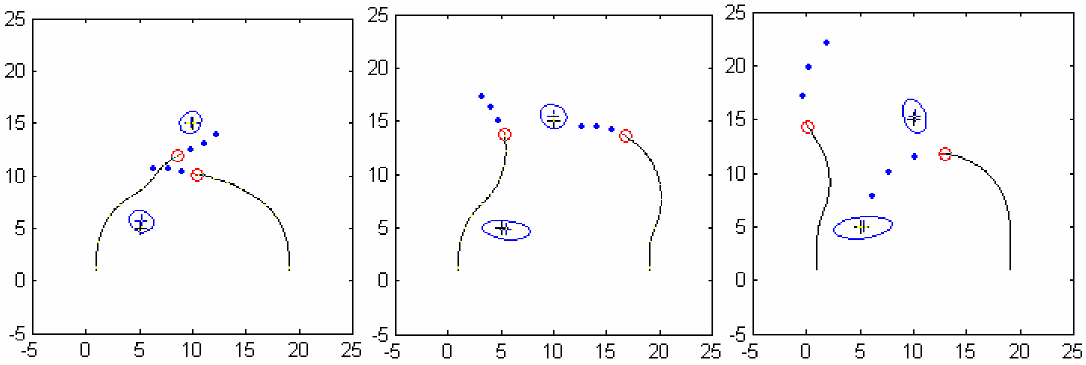
Cooperation among multiple agents plays a vital role. The last column of Table 3-1 contains the results from a decentralised strategy. The computation time for the decentralised approach is the shortest of all the approaches implemented, however, it comes at the cost of inferior global performance. The information gain is consistently the lowest, at a mean of 35% less information than centralised EETS. It is evident that cooperation is necessary for effective trajectory planning and therefore relevant strategies should be explored when implementing a decentralised approach.

3.8.1.5. Significance of Information Represented as a Matrix

An interesting observation can be made when EETS is simulated given three sets of initial information matrices $\mathbf{I}_0^j$ all with identical eigenvalues ($eig(\mathbf{I}_0^1)$=0.4384, 4.5616; $eig(\mathbf{I}_0^2)$=0.6972, 4.3028) and determinants ($det(\mathbf{I}_0^1)$=2, $det(\mathbf{I}_0^2)$=3). It is demonstrated that although the information gain is evaluated using the eigenvalues, the trajectory is actually governed by the whole information matrix.

Simulations were conducted with a terminal time of T = 10, 3-step look-ahead and three control options. Figure 3-5 shows the solution trajectories optimised for the different initial information matrices with identical scalar properties.

As the three environments have identical scalar measures of information, it is interesting to observe the performance of the trajectories optimised for a particular environment and applied the other environments. Table 3-2 displays the final information (i.e. the minimum eigenvalue of the information matrices) acquired from executing the three trajectories in Figure 3-5 with the three sets of initial information matrices. It is clear that the solution obtained is not as effective when the trajectories are applied to the other environments with identical scalar measures. As there is greater representation of information as a matrix, the solution for corresponding trajectories and environments are superior. This reinforces the importance of representing the information as a matrix.



(a) $\mathbf{I}_0^1$ = diag[0.4384, 4.5616] (b) $\mathbf{I}_0^1$ = [3 2; 2 2] (c) $\mathbf{I}_0^1$ = diag[4.5616, 0.4384]

$\mathbf{I}_0^2$ = diag[4.3028, 0.6972] $\mathbf{I}_0^2$ = [1 1; 1 4] $\mathbf{I}_0^2$ = diag[0.6972, 4.3028]

Figure 3-5 Trajectories of the Two Robots, with Different $\mathbf{I}_0^j$

	Path (a)	Path (b)	Path (c)
$\mathbf{I}_0^1$ (a)	**3.5363**	3.1927	2.0063
$\mathbf{I}_0^1$ (b)	1.7280	**3.0650**	1.6678
$\mathbf{I}_0^1$ (c)	0.8937	1.3412	**2.2307**

Table 3-2 Final Information for Different $\mathbf{I}_0^1$ and Associated Paths

3.8.1.6. Constraint Satisfaction

Figure 3-6 demonstrates that the proposed algorithm does not violate the constraints of the no-go-zones that are depicted by the circles around the targets. With a 3-step look-ahead, the robots are prevented from moving to a pose where violating a constraint is unavoidable in the next step. With a one-step look-ahead it is more likely that the robot would move close to the target to gain more information but may unknowingly encounter an obstacle in the following step that is within its minimum turn radius.

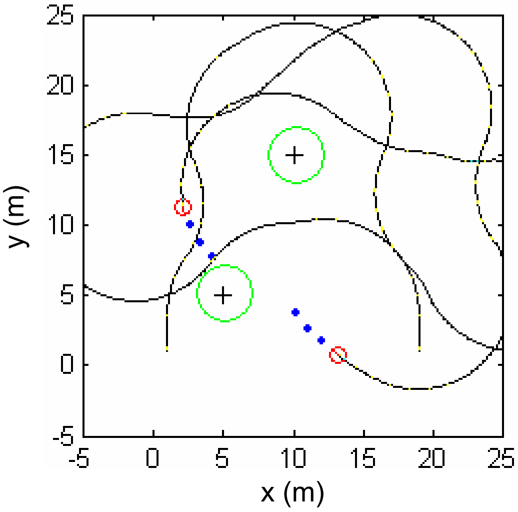


Figure 3-6 Trajectories Outside the No-Go-Zone after *i*=80 Time Steps

3.8.2. Trajectory Planning for Two UAVs Observing Three Ground Targets

Simulations were conducted in three-dimensional space with two UAVs and three targets on the ground. The two UAVs are set to fly at different set altitudes of 200m and 250m thus strategies for inter-robot collision avoidance are not required. The UAVs are to localise the targets on the ground within a prescribed terminal time. No-go-zones are prescribed around the targets and in this example the constraint is enforced such that the uncertainty of the target location is also taken into consideration. Each UAV has a maximum bank angle and minimum and maximum velocities limiting the roll and turn-rate. In the simulations, the UAVs have a maximum bank angle of π/6rad and fly with a constant velocity of 30ms^{-1}. Observations are made with a camera with ±π/12rad field of view and the sensor footprint depends on the current roll of the UAVs.

3.8.2.1. Three-dimensional UAV Process Model

The process model for the UAVs is obtained from Kim et al. (2004). This model is particular to the two UAVs (called Brumbies), deployed at the Centre of Excellence for Autonomous Systems, shown in Figure 3-7.

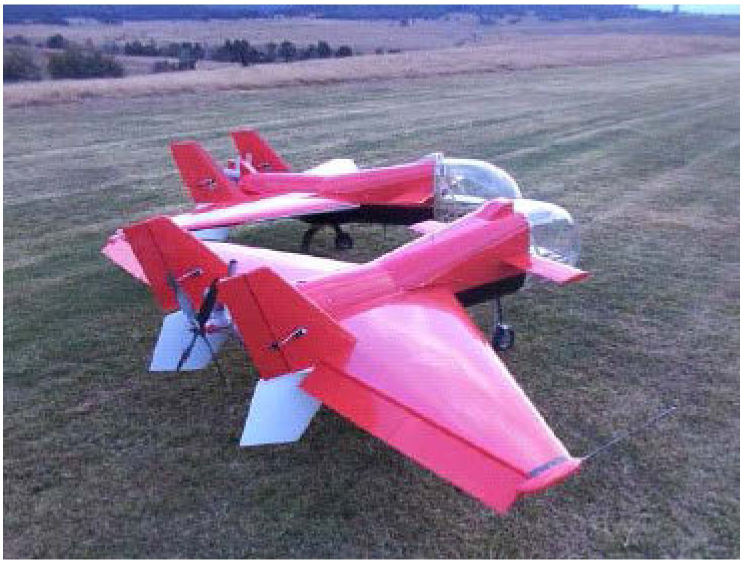


Figure 3-7 Brumbies in Field

The following describes a general model of the brumbies. The state of the UAVs are described by location $\mathbf{L}_i^n = [x; y; z]$, velocities $\mathbf{V}_i^n = [v_x; v_y; v_z]$ and the attitude $\mathbf{\Phi}_i^n = [\psi; \rho; \phi]$ (Euler angles roll, pitch and yaw),

$$\mathbf{x}_i^n = \begin{bmatrix} \mathbf{L}_i^n \\ \mathbf{V}_i^n \\ \mathbf{\Phi}_i^n \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{i-1}^n + \mathbf{V}_{i-1}^n \Delta t \\ \mathbf{V}_{i-1}^n + [\mathbf{C}_{i-1}^{nb}\, \mathrm{f}_i^b + g^n]\Delta t \\ \mathbf{\Phi}_{i-1}^n + \mathbf{E}_{i-1}^{nb} \omega_i^b \Delta t \end{bmatrix} \tag{3.29}$$

where f_i^b and ω_i^b are acceleration and rotation rates measured in the body frame. $\mathbf{C}_i^{nb}$ is the direction cosine matrix and $\mathbf{E}_i^{nb}$ is the matrix that transforms the rotation rates in the body frame to the Euler angle rates:

$$\mathbf{C}_i^{nb}=\begin{bmatrix} \mathrm{c}(\rho)\,\mathrm{c}(\phi) & -\mathrm{c}(\psi)\,\mathrm{s}(\phi)+\mathrm{s}(\psi)\,\mathrm{s}(\rho)\,\mathrm{c}(\phi) & \mathrm{s}(\psi)\,\mathrm{s}(\phi)+\mathrm{c}(\psi)\,\mathrm{s}(\rho)\,\mathrm{c}(\phi) \\ \mathrm{c}(\rho)\,\mathrm{s}(\phi) & \mathrm{c}(\psi)\,\mathrm{c}(\phi)+\mathrm{s}(\psi)\,\mathrm{s}(\rho)\,\mathrm{c}(\phi) & -\mathrm{s}(\psi)\,\mathrm{s}(\phi)+\mathrm{c}(\psi)\,\mathrm{s}(\rho)\,\mathrm{c}(\phi) \\ -\mathrm{s}(\rho) & \mathrm{s}(\psi)\,\mathrm{c}(\rho) & \mathrm{c}(\psi)\,\mathrm{c}(\rho) \end{bmatrix} \tag{3.30}$$

$$\mathbf{E}_i^{nb}=\begin{bmatrix} 1 & \mathrm{s}(\psi)\,\mathrm{s}(\rho)/\mathrm{c}(\rho) & \mathrm{c}(\psi)\,\mathrm{s}(\rho)/\mathrm{c}(\rho) \\ 0 & \mathrm{c}(\psi) & -\mathrm{s}(\psi) \\ 0 & \mathrm{s}(\psi)/\mathrm{c}(\rho) & \mathrm{c}(\psi)/\mathrm{c}(\rho) \end{bmatrix} \tag{3.31}$$

where s(.) and c(.) represent sin(.) and cos(.) respectively.

The following conditions are applied in this simulation: a) Altitude fixed, $\Delta\mathbf{V}_z=0$; b) Velocity constant $\mathbf{V}_c=C$; c) Roll, $\Delta\psi$, assigned as the control input, $\mathbf{u}_i^n$. The model (3.29) can now be simplified with

$$\begin{gathered} \Delta\phi=-9.8\tan(\psi+\Delta\psi)/\mathbf{V}_c \\ \omega=\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}=\begin{bmatrix}\Delta\psi\\ \Delta\phi\sin(\psi)\\ \Delta\phi\cos(\psi)\end{bmatrix} \end{gathered} \tag{3.32}$$

$$\Delta\mathbf{V}=\begin{bmatrix}\Delta V_x\\ \Delta V_y\\ \Delta V_z\end{bmatrix}=\begin{bmatrix}\mathbf{V}_c\cos(\phi+\Delta\phi)-V_x\\ \mathbf{V}_c\sin(\phi+\Delta\phi)-V_y\\ 0\end{bmatrix} \tag{3.33}$$

$$\mathrm{f}_i^b=\begin{bmatrix}\mathrm{f}_x\\ \mathrm{f}_y\\ \mathrm{f}_z\end{bmatrix}=\begin{bmatrix}\Delta V_x\cos(\phi)+\Delta V_y\sin(\phi)\\ 9.8\sin(\psi)-(\Delta V_x\sin(\phi)-\Delta V_y\cos(\phi))\cos(\psi)\\ 9.8\cos(\psi)+(\Delta V_x\sin(\phi)-\Delta V_y\cos(\phi))\cos(\psi)\end{bmatrix} \tag{3.34}$$

where $\Delta\psi$ is the change in the roll set by the trajectory planner.

3.8.2.2. Observation Model

Each observation $\mathbf{z}_{i+1}^{jn} = [\theta;\varphi]$ of target j from robot n comprises of both bearing, θ, and elevation, φ, measurements from the n-th robot to the j-th target at time step i+1, given by

$$\mathbf{z}_i^{jn} = h_n(\mathbf{x}^n, \boldsymbol{\lambda}^j) + \mathbf{w}_n$$
$$= \begin{bmatrix} \theta \\ \varphi \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}\left(\dfrac{y^j - y_i}{x^j - x_i} \right) + \mathbf{w}_n^\theta \\ \operatorname{atan2}\left(\dfrac{z^j - z_i}{\sqrt{(x^j - x_i + x_{offset})^2 + (y^j - y_i + y_{offset})^2}} \right) + \mathbf{w}_n^\varphi \end{bmatrix} \tag{3.35}$$

where $\boldsymbol{\lambda}^j = \left[x^j; y^j; z^j \right]$ denotes the state of the j-th target, h_n is a nonlinear function which depends on the model of the sensor equipped on the n-th robot and $\mathbf{w}_n$ is a zero-mean Gaussian measurement noise with standard deviation of $\sigma = \pi / 36$ rad is used for both the bearing and elevation measurements.

x_{offset} and y_{offset} are the offsets of the camera view due to the roll (ψ) and pitch (ρ) of the UAV, and are defined by

$$\begin{aligned} z_- &= z_i - z_i^j \\ x_\rho &= z_- \tan(\rho)\cos(\phi) \\ y_\rho &= z_- \tan(\rho)\sin(\phi) \\ x_\psi &= z_- \tan(\psi)\cos(\phi - \pi/2) \\ y_\psi &= z_- \tan(\psi)\sin(\phi - \pi/2) \\ x_{offset} &= x_\rho + x_\psi \\ y_{offset} &= y_\rho + y_\psi \quad . \end{aligned} \tag{3.36}$$

The information predicted to be observed depends largely on the roll of the UAV as the camera is mounted on the base of the UAVs, as seen in Figure 3-8.

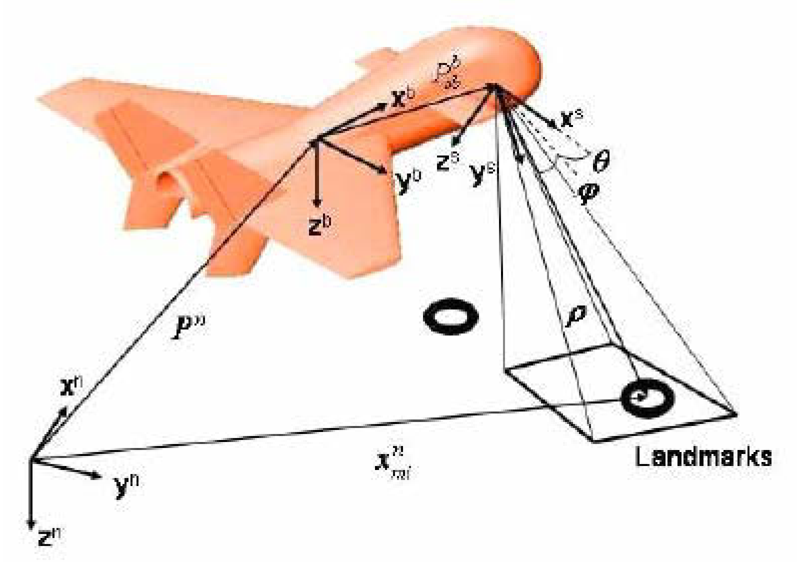


Figure 3-8 Camera View of the UAV (Kim et al. 2004)

3.8.2.3. Performance of MPC

Figure 3-9 illustrates the initial setup of the simulation with three targets on the ground and two UAVs at different altitudes. Ten trials were conducted with an increasing planning horizon. The UAVs are simulated using MPC from one-step (D = 1) to five-step (D = 5) planning horizon. Table 3-3 contains the average information gain from these trials and the single-step method (D = 1) is used as the benchmark.

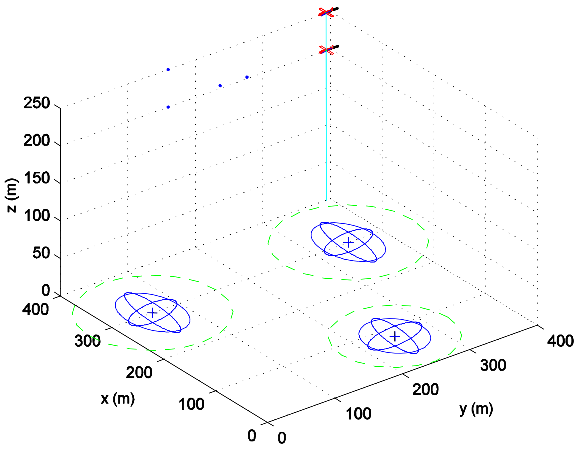


Figure 3-9 Initial Information and Starting Poses

System performance using an arbitrary fixed path, where the roll is a constant π/6rad (Figure 3-11), is also evaluated for the purpose of comparison. This path is selected such that each target is visible to at least one of the UAVs along the path and the no-go-zone constraints are not violated.

Results in Table 3-3 show that the information gain following the trajectory generated by MPC is much higher than that of the fixed path. Furthermore, it can be seen that higher information gain is obtained from the MPC strategy with the increase of the planning horizon *D*. After a terminal time of $T = 50$ steps, with a planning horizon of $D = 5$, MPC gained on average 426% of the information obtained using $D = 1$. The difference in information gain is not as large after 500 steps as the rate of information gain plateaus. This is a result of the planes flying at a fixed altitude high above the targets. The eigenvalues of the information matrices in the vertical direction eventually became dominant and will not increase significantly further due to the fixed altitude. Figure 3-10(a) shows the vertical offset of the planes and the uncertainties of the targets at the end of the mission. There is greater uncertainty in the vertical direction.

D	T	N_ω	$\mathbf{I}_{\max}$	**R** (%)	$\boldsymbol{D}$	T	N_ω	$\mathbf{I}_{\max}$	**R** (%)
0 (fixed ctrl)	50	NA	0.0080	84	0	500	NA	0.0212	53
1	50	3	0.0095	100	1	500	3	0.0403	100
2	50	3	0.0179	188	2	500	3	0.0535	133
3	50	3	0.0219	231	3	500	3	0.0561	139
4	50	3	0.0333	351	4	500	3	0.0584	145
5	50	3	0.0405	426	5	500	3	0.0596	148

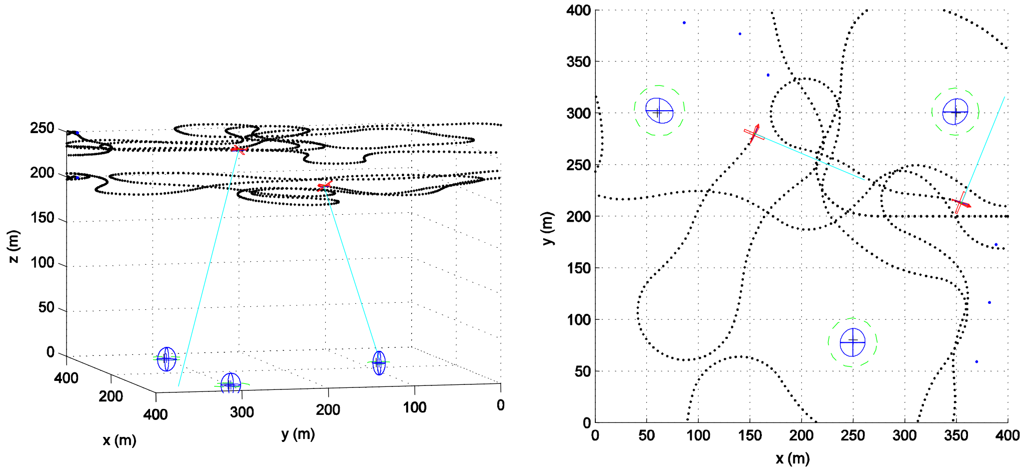
Table 3-3 Results from the Three Approaches

where $\mathbf{I}_{\max}$=min(eig($\mathbf{I}_T$)), R=$\mathbf{I}_{\max}$/$\mathbf{I}_{\max}$(*D*=1)*100

3.8.2.4. Incorporating Dynamic Constraints

A benefit of using MPC is that varying constraints can be continuously incorporated into the planning process. To illustrate this aspect, a no-go-zone constraint, depicted by the green circles in Figure 3-9 is enforced, i.e. the robots cannot fly within the maximal axis of the 95% confidence ellipse of any target in addition to a predefined safety distance. When the uncertainty of the target location is large, this no-go-zone constraint is larger. As the robots gain more information on the target's location, then the size of the no-go-zones around the targets reduce accordingly.

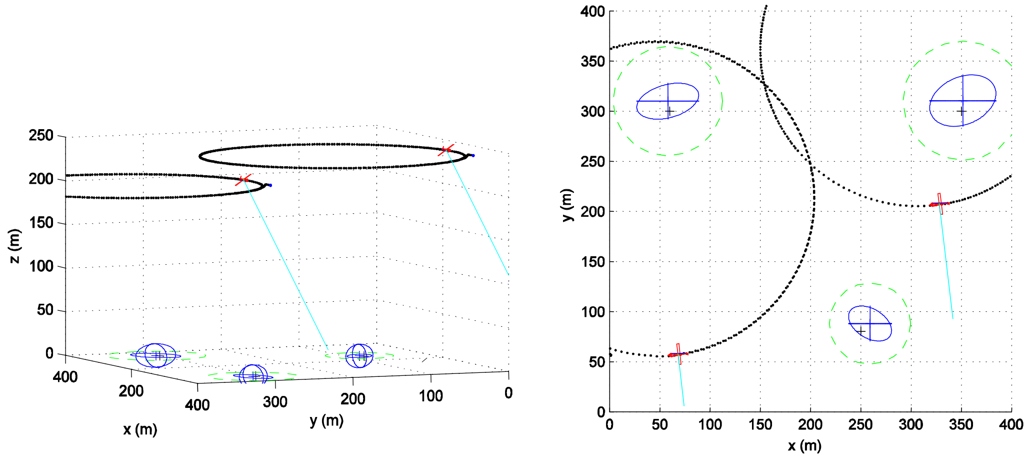
The result of using MPC is displayed in Figure 3-10. The UAVs gradually move closer to the targets as the sizes of the no-go-zones reduce so as higher information gain may be obtained from their observations. Figure 3-10(b) shows the plane-view depicting the paths outside the no-go-zones.



(a) MPC Path

(b) MPC Path within Constraints

Figure 3-10 MPC Path



(a) Fixed Path

(b) Fixed Path within Constraints

Figure 3-11 Fixed Control

3.9. Discussion

3.9.1. Computational Issues

3.9.1.1. Planning Horizon

By planning a number of steps ahead, a near optimal route can be chosen by assessing the maximum predicted future reward. Increasing D may provide a solution closer to the global optimum; however this results in an exponential increase in computational complexity. Size of D is clearly limited by the available computational capacity.

On the other hand, if the sensor noise is high, Assumption I in Section 3.5.2 is violated and the control action obtained by the D-step optimisation strategy may be far from optimal. Thus in many practical scenarios using a large D may not provide significant gains.

Additionally, in the proposed strategy, equal steps sizes in the planning horizon are used. It is possible to use varying step sizes, such as a geometric series, where there is a finer resolution at the start and an increasing step size for each time step in the future. Other variations to the structure of the tree search, not explored here, also represent an interesting subject for future work.

3.9.1.2. Optimisation Algorithm

Optimisation algorithms are necessary to systematically search through possible control actions that are within the constraints to obtain the solution that best meets the objective of the trajectory planning. Although the computational complexity of EETS grows exponentially with respect to the number of options N_ω, keeping N_ω small reduces the computational cost while having only a minor tradeoff in information gain, as seen from Table 3-1. This is due to the nature of the target localisation problem; if two observations are made at points very close to each other, then the information obtained from the observations will be similar.

Figure 3-5 shows that the information gain as a function of the control actions (the turn-rates) is relatively smooth, except for the steep drop due to the control actions moving the robots such that the targets are out of the sensor's field of view.

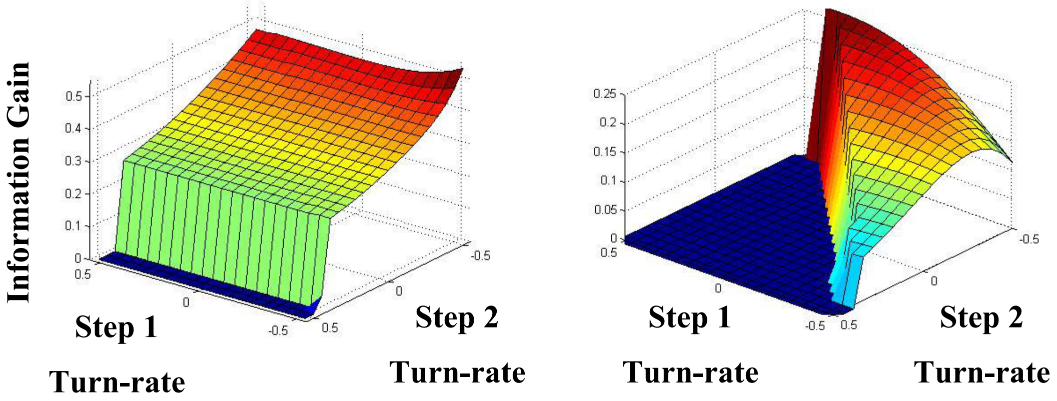


Figure 3-12 Graph Of Information Gain Vs. Turn-Rates

Taken from 2 Different Robot Locations

To further analyse the affect of considering a larger number of control options, 34 trials were conducted with one robot, one target in a two-dimensional scenario with the planning horizon set to D=2. Two cases are compared with N_ω=129 and N_ω=3. The results obtained are shown in Table 3-4. The study showed only 4% improvement in information gain with N_ω=129. It follows that, in the bearing-only trajectory planning problem, it is appropriate to select only a small number of possible control actions that result in significantly different observing points in the next time step.

	Information Gain						
N_ω	*Trial 1*	*Trial 2*	*Trial 3*	*Trial 4*	*Trial 5*	*Trial 6*	*Trial 7*
3	0.2346	1.3621	0.8192	1.5429	1.1056	1.7024	1.7184
129	0.2596	1.3621	0.8192	1.5435	1.1056	1.7178	1.7425

Table 3-4 EETS Increasing Control Options

3.9.1.3. Centralised Control

Centralised control suffers from the key problem of computational complexity increasing exponentially with the number of robots used. On the other hand, decentralised control without coordination may clearly perform poorly in comparison. An extreme case is when the two robots start from similar poses. Trajectories obtained from optimal decentralised control will intuitively result in similar trajectories for each robot, which is far from the global optimal. Better performance may be achieved by decentralised control combined with negotiation between robots (e.g. Grocholsky 2006) but negotiation processes are generally complex.

In the target localisation problem, the number of actively cooperating robots may be small in general. The cooperative control problem for a large number of robots can often be decoupled into two sub-problems; a task assignment problem and an actively cooperative control problem, in the case of a small group of robots. This presents a potentially fruitful area for future research.

3.9.2. Bearing-Only Target Initialisation

The strategies proposed in this chapter are effective only if an initial target location estimate, for example from satellite surveillance, is available. When the initial target locations are unknown, the fundamental task then becomes target search and exploration (Roy and Earnest 2006). Then in such case, target initialisation needs to be considered.

The initial target location estimate cannot be described by a single Gaussian distribution when the information is from only one observation using a bearing-only sensor. This prevents the use of the EIF from the outset due to the Gaussian assumption.

3.9.2.1. Strategies for Bearing-Only Target Initialisation

There are many established methods to represent non-Gaussian estimates. Ong et al. (2005) compared different ways of non-Gaussian representations including Gaussian mixture models, Parzen density estimates and particles. The results of the study led to the conclusion that particle representations are best used for initialisation. It is a popular strategy for bearing-only initialisation (Davison, 2003).

In Bryson (2007), a delayed strategy is used for the initialisation of targets. The robot poses and observations are stored until a sufficient baseline exists between two observations. The information from poses and observations are recovered in a batch update step. Due to a delayed initialisation, the trajectory planning does not consider the new target until it is initialised.

Kwok and Dissanayake (2004) used a multiple hypothesis filter where several hypotheses of the target location are created along the line of sight of the initial observation. Each hypothesis is integrated into the filter and treated as a separate target. As further observations are made, the false hypotheses are eliminated.

3.9.2.2. Incorporating new Targets in the Trajectory Planning during Initialisation

There may be cases where there are several targets being estimated and most of these targets have already been initialised. It then becomes difficult to weight the information gain from the targets described by the information matrix as well as the newly detected targets described by other forms such as particles.

For the delayed initialisation case, it is not possible to incorporate a new target in the planning before its initialisation. In the multi-hypothesis case, it is difficult to include the new target in the planning because there are multiple possibilities. Incorporating all the hypotheses would incorrectly alter the balance between the information gain of the new target and the information gain of existing targets.

These initialisation processes may be quite fast as several subsequent observations are often made while the UAV flies over the target. Given a reasonable sensor sample rate, it is unlikely that a target will be observed only once and then disappear for several observations. One possible option for the trajectory planning is to ignore the information gain from the new target, until the initialisation process is complete.

3.10. Summary

In this chapter, a trajectory planning problem for heterogeneous robots in bearing-only target localisation is investigated. The problem has been formulated as an optimal control problem for a nonlinear system with a gradually identified model and a nonlinear MPC strategy is proposed.

Simulations of two scenarios are presented comparing the information gain and computation times of different optimisation strategies. The results show that on average EETS ($N_{\omega} = 3$) is 20 times faster than EETS+SQP and achieves similar gains in information. It is demonstrated that EETS alone provides a near-optimal solution. The value of cooperation is also demonstrated with the centralised control, obtaining on average 35% higher information gain than a simple decentralised control solution. An interesting observation is presented where the trajectories obtained by the proposed method is governed by the information matrix and not the scalar measure of information. MPC is shown to perform better than an arbitrary fixed control and increasing the planning horizon improved results. It is also demonstrated that the MPC solution does not violate dynamic constraints.

According to the analysis and simulation results, the following conclusions are made for effective trajectory planning in multi-agent bearing-only target localisation: 1) Information matrices and EETS allow the planning of near optimal trajectories in terms of time and information gain. 2) A small number of control options are sufficient to obtain a near optimal result. 3) Maintaining a short planning horizon, *D*, is beneficial to reduce computation. 4) Centralised control is important for optimal robot coordination.

In this chapter it is assumed that the locations of the robots are known. However, it would be more realistic for the location estimates of the robots to inherently contain noise and errors, even if they are provided by an external source such as GPS. Often, the locations of the robots may not even be available nor observable externally, as would be the case in indoor environments. The process of estimating both the robot location and the target/feature locations is called Simultaneous Localisation and Mapping (SLAM). In the next chapter, the trajectory planning problem for SLAM is addressed. The objective is to plan the robot trajectory such that the quality of the SLAM result is maximised.

Chapter 4. Trajectory Planning for Point Feature based SLAM

4.1. Introduction

This chapter explores trajectory planning for point feature based Simultaneous Localisation and Mapping (SLAM). The main performance criteria in SLAM are to achieve good coverage of the unexplored area and to obtain a map with high accuracy within an available time. To this end, trajectory planning provides scope for the robot to exploit the latest knowledge obtained from sensor observations in actively mapping an environment. As the robot gradually updates its knowledge of the environment, the uncertainties of the robot pose and map features are continually taken into account. This process of trajectory planning for SLAM is referred to in the literature as Active SLAM (Zhang et al. 2006) or Simultaneous Planning, Localisation and Mapping (SPLAM) (Meger 2006).

For this problem, an Extended Kalman Filter (EKF) is considered as the underlying estimator. It is used to estimate the robot pose and the features of interest, where the uncertainty of the estimate is expressed by the covariance matrix. Figure 4-1 illustrates the SLAM problem where a relative map is created based on sensor observations.

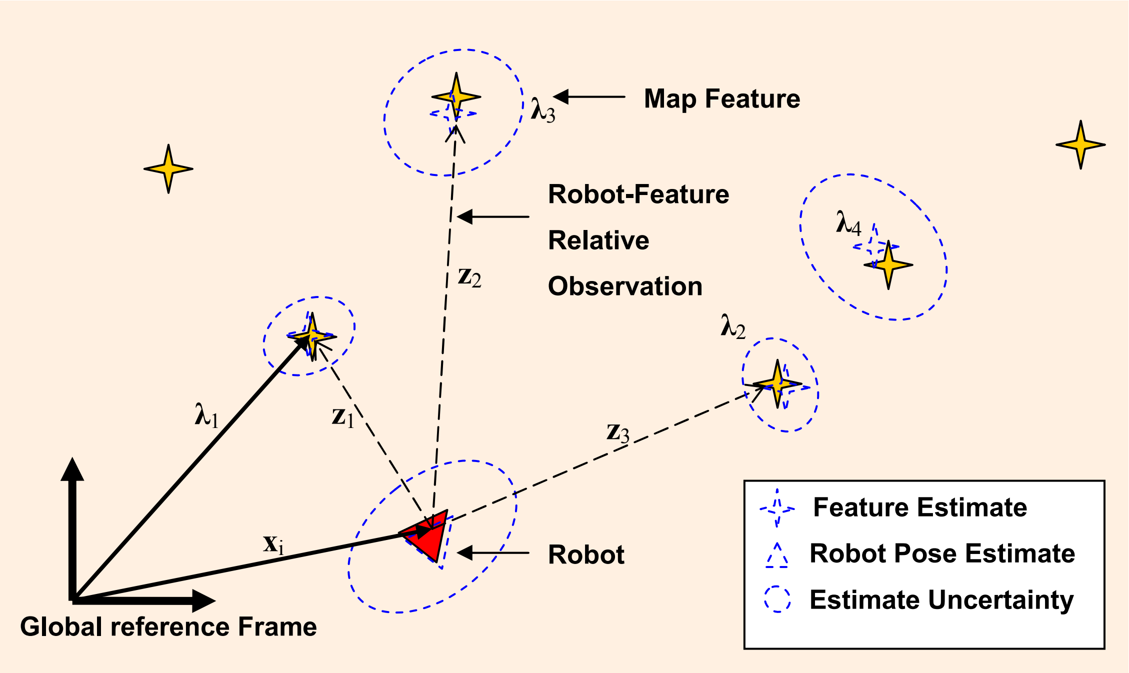


Figure 4-1 The SLAM Problem

This chapter demonstrates that Active SLAM can be formulated as an optimal control problem for a nonlinear control system with a gradually identified model. It is then shown that the MPC framework proposed in Chapter 3 can be used to obtain robot trajectories. The objective of the MPC optimisation is to maximize the information gathered from the environment within a finite time horizon. In MPC however, the planning horizon is limited by the computational capacity and only a local solution is possible. A novel technique is introduced that utilises an attractor combined with MPC to improve the solution. The attractor provides high level task intentions and incorporates global information about the environment for the local planner, thereby eliminating the need for costly global planning with longer horizons. It is shown that trajectory planning with MPC and an attractor can effectively deal with a range of objectives and results in improved performance over systems that employ local planning alone.

4.2. Background

In the SLAM problem, the feature locations are initially unknown. The robot builds a relative map based on observations of features and localises itself within this map. Both features locations and robot pose are estimated as illustrated in Figure 4-1. Trajectory planning for SLAM is more challenging than the problem addressed in Chapter 3 because the robot is required to revisit well known features to improve the accuracy of its pose estimate, explore unknown areas to ensure the map is complete, consider the detection and initialisation of new features, and account for new constraints imposed on the trajectory.

Several works present in the literature have addressed the Active SLAM problem. Localisability of the robot during the SLAM process is considered by Stachniss et al. (2004) through the use of active loop closing and Makarenko et al. (2002) by considering it as a utility at the destination. However, as the robot may require to localise itself before it reaches the destination, it may be beneficial to consider localisability along each step of the path and to determine appropriate instances to revisit well known features.

For exploration, both Stachniss et al. (2004) and Makarenko et al. (2002) performed frontier based exploration using an occupancy grid map. Sim (2005) on the other hand encouraged coverage by randomly placing virtual features in unexplored areas. Yet, this strategy is not particularly effective for systems with short planning horizons and limited sensing as the virtual features will not influence the robot's decision if they are not visible within the planning horizon. Overcoming this limitation by placing numerous features would increase the computational cost of the planner.

4.3. Trajectory Planning Problem in SLAM

Planning robot actions for SLAM requires fast algorithms that can adapt to changes in the current knowledge of the environment. Two main items not considered in the previous chapter, the addition of newly observed features and the update of the robot pose covariance, need to be considered here.

4.3.1. The Process and Observation Models

What follows is the formulation of a single robot SLAM problem. In SLAM, the state vector contains the robot's pose and the locations of the features of interest,

$$\boldsymbol{\zeta}_i \triangleq \{\mathbf{x}_i, \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_J\} . \tag{4.1}$$

Suppose the discrete-time process model of the robot is

$$\mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_\mathbf{x}) \tag{4.2}$$

where $f(.)$ is a nonlinear function, which depends on the type of robot and its dynamic model, $\mathbf{x}_i$ is the robot pose, $\mathbf{u}_i$ is the control input at time *i* and is constant during [*i*; *i* + 1), and $\mathbf{w}_\mathbf{x}$ is the zero-mean Gaussian process noise with covariance matrix $\mathbf{P}_w$.

Constraints on the robot's motion are to be incorporated into the planning process. The control constraints and state constraints can be expressed by

$$\mathbf{u}_i \in U_i,\ \mathbf{x}_t \in R_i,\ t \in [i, i+1), \tag{4.3}$$

where U_i is the set of admissible controls for the robot at time *i* and R_i describes the safe region for robots during time *i* to time *i* + 1.

The features are assumed to be stationary. Let J_{i+1} denote the set of the indices of the features that the robot can observe at time *i* + 1,

$$J_{i+1} = \{j_1, \ldots, j_{K_{i+1}}\} \tag{4.4}$$

where the integer K_{i+1} depends on the pose of the robot at time *i*+1, on the range of the sensor and the feature distribution in the environment. The K_{i+1} features may contain both previously observed features as well as new features. The observation of the robot at time *i* + 1 is then

$$\mathbf{z}_{i+1} = [\mathbf{z}_{i+1}^{j_1}, \ldots, \mathbf{z}_{i+1}^{j_{K_{i+1}}}] \tag{4.5}$$

such that for each feature, $j \in J_{i+1}$, the observation model is

$$\mathbf{z}_{i+1}^{j} = h_{i+1}^{j}(\mathbf{x}_{i+1}, \boldsymbol{\lambda}^{j}) + \mathbf{w}_{z}^{j} \tag{4.6}$$

where $\boldsymbol{\lambda}^{j}$ denotes the state of the *j*-th feature, $h_{i+1}^{j}(.)$ is a nonlinear function which depends on the model of the sensor, and $\mathbf{w}_{z}^{j}$ is a zero-mean Gaussian measurement noise with a covariance matrix $\boldsymbol{\Sigma}_{i+1}^{j}$.

4.3.2. Extended Kalman Filter based SLAM Algorithm

The Extended Kalman Filter (EKF) can be used to update the knowledge of the system. The complete process model, for a state vector containing the robot pose $\mathbf{x}_i$ and the locations of all the observed features, $\boldsymbol{\lambda}$, is

$$\boldsymbol{\zeta}_{i+1} = \mathbf{f}(\boldsymbol{\zeta}_i, \mathbf{u}_i, \mathbf{w}_{\mathbf{x}}) = \begin{bmatrix} f(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_{\mathbf{x}}) \\ \boldsymbol{\lambda} \end{bmatrix}. \tag{4.7}$$

The prediction step is

$$\begin{aligned} \hat{\boldsymbol{\zeta}}_{i+1}^{-} &= \mathbf{f}(\hat{\boldsymbol{\zeta}}_i, \mathbf{u}_i, 0) \\ \mathbf{P}_{i+1}^{-} &= \boldsymbol{\Gamma}_i \mathbf{P}_i \boldsymbol{\Gamma}_i^{\mathrm{T}} + \mathbf{Y}_i \mathbf{P}_w \mathbf{Y}_i^{\mathrm{T}} \end{aligned} \tag{4.8}$$

where $\hat{\boldsymbol{\zeta}}_i$ and $\mathbf{P}_i$ are the state estimate and covariance matrix at time *i* (after the update), $\hat{\boldsymbol{\zeta}}_{i+1}^{-}$ and $\mathbf{P}_{i+1}^{-}$ are the predicted state estimate and covariance matrix at time $i+1$ (before the update), $\boldsymbol{\Gamma}_i$ and $\mathbf{Y}_i$ are the Jacobians of $\mathbf{f}$ with respect to $\boldsymbol{\zeta}$ and $\mathbf{w}_{\mathbf{x}}$ which are evaluated at $(\hat{\boldsymbol{\zeta}}_i, \mathbf{u}_i, 0)$, respectively.

The update step in EKF SLAM is:

$$\begin{aligned} \hat{\boldsymbol{\zeta}}_{i+1} &= \hat{\boldsymbol{\zeta}}_{i+1}^{-} + \mathbf{K}_{i+1}\left(\mathbf{z}_{i+1} - \mathbf{h}_{i+1}\left(\hat{\boldsymbol{\zeta}}_{i+1}^{-}\right)\right) \\ \mathbf{P}_{i+1} &= \mathbf{P}_{i+1}^{-} - \mathbf{K}_{i+1}\mathbf{S}_{i+1}\mathbf{K}_{i+1}^{\mathrm{T}} \end{aligned} \tag{4.9}$$

where

$$\begin{aligned}\mathbf{K}_{i+1} &= \mathbf{P}_{i+1}^{-}\mathbf{H}_{i+1}^{\mathrm{T}}\mathbf{S}_{i+1}^{-1} \\ \mathbf{S}_{i+1} &= \mathbf{H}_{i+1}\mathbf{P}_{i+1}^{-}\mathbf{H}_{i+1}^{\mathrm{T}} + \mathbf{\Sigma}_{i+1}\end{aligned} \tag{4.10}$$

and $\mathbf{h}_{i+1} = \left[h_{i+1}^{1} \cdots h_{i+1}^{J_{i+1}}\right]^{\mathrm{T}}$ is the set predicted observations, $\mathbf{H}_{i+1}$ is the Jacobian of $\mathbf{h}_{i+1}$ evaluated at $\hat{\boldsymbol{\zeta}}_{i+1}^{-}$ and $\mathbf{\Sigma}_{i+1} = diag[\mathbf{\Sigma}_{i+1}^{j}]$. Note that $\mathbf{H}_{i+1}$ depends on J_{i+1} since $\mathbf{h}_{i+1}$ depends on J_{i+1}. Also, detection of new features increases the dimension of $\hat{\boldsymbol{\zeta}}_{i+1}$ and $\mathbf{P}_{i+1}$.

The EKF formulation requires the estimate of the state vector and the corresponding covariance matrix to be updated in a recursive manner.

Combining equations (4.8) and (4.9), the EKF formula can be summarized as

$$\begin{aligned}\hat{\boldsymbol{\zeta}}_{i+1} &= \hat{\mathbf{F}}_{P}(\hat{\boldsymbol{\zeta}}_{i}, \mathbf{P}_{i}, \mathbf{u}_{i}, J_{i+1}, \mathbf{z}_{i+1}) \\ \mathbf{P}_{i+1} &= \hat{\mathbf{G}}_{P}(\hat{\boldsymbol{\zeta}}_{i}, \mathbf{P}_{i}, \mathbf{u}_{i}, J_{i+1})\end{aligned} \tag{4.11}$$

where $\hat{\boldsymbol{\zeta}}_{i}$, $\mathbf{P}_{i}$ and $\hat{\boldsymbol{\zeta}}_{i+1}$, $\mathbf{P}_{i+1}$ denote the state vector estimate and covariance matrix at time i and time $i + 1$, respectively, $\mathbf{u}_i$ is the control applied at time i, J_{i+1} and $\mathbf{z}_{i+1}$ are defined in (4.4) and (4.5). The functions $\hat{\mathbf{F}}_{P}$ and $\hat{\mathbf{G}}_{P}$ are determined by the process and observation models along with the prediction and update formulas (4.8) and (4.9) respectively.

4.3.3. Problem Statement

The objective of the trajectory planning is to minimise the uncertainty of the system after a finite time T has elapsed. The covariance matrix at time T, $\mathbf{P}_T$, provides a complete representation of this uncertainty.

Problem Statement: Suppose the total time for the SLAM task is T. At time step 0, the robot pose is $\mathbf{x}_0$, the initial estimate of the J_0 features observed are $\boldsymbol{\lambda}_0^{j}, j = 1, \ldots, J_0$ and the covariance matrix is $\mathbf{P}_0$. The task is to choose suitable control actions for the robot during [0; T),

$$\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1} \tag{4.12}$$

such that a quantitative measure of the final covariance matrix $\mathbf{P}_T$ is minimised.

Note that different quantitative measures of the covariance matrix may be used depending on the application, such as the maximal/minimal eigenvalue, determinant, or the trace of the matrix. For this problem, the trace is selected as the quantifier, i.e.

$$\text{trace}(\mathbf{P}_T)\,. \tag{4.13}$$

It has been shown in Sim and Roy (2005) that in Active SLAM, optimisation using the trace of the covariance matrix from the EKF performs better than using the determinant. Computing the eigenvalue as used in Chapter 3 is computationally expensive due to the relatively large size of **P** in the case of SLAM. Furthermore, as a bearing-only sensor is not used in this chapter, the uncertainty ellipses of the features do not have the same characteristics as those present in the bearing-only target localisation problem.

4.3.4. An Optimal Control Problem for a Gradually Identified Model

The above planning problem can be regarded as a finite-time horizon optimal control problem for a nonlinear control system where the system dynamics are given by (4.11). The system state includes the estimate of the state vector $\hat{\zeta}_i$ and the independent elements of the covariance matrix $\mathbf{P}_i$. The objective of the control problem is to minimise the trace given by (4.13), which is a function of the system state.

It must be noted that the system given by (4.11) is not an ordinary nonlinear discrete-time control system. The reasons for this are:

(a) The dimension of the system state is not fixed. For example, as new features are detected in the environment, the dimensions of $\hat{\zeta}_i$ and the associated covariance matrix $\mathbf{P}_i$ are enlarged;

(b) The state constraints in (4.3) are not known a priori. For example, an obstacle may not be detected until it appears within the sensor field of view; and

(c) The dynamics of the system depend on J_{i+1} and $\mathbf{z}_{i+1}$. Neither of these are available until time step i + 1. For example, it is difficult to predict the set of features J_{i+1} that can be observed due to the inaccurate robot pose and feature position estimates, and the unknown locations of possible new features. In addition, the true observation $\mathbf{z}_{i+1}$ contains sensor noise.

Equation (4.11) shows that the uncertainty, $\mathbf{P}_{i+1}$, at time i+1, is not dependent on the true observation $\mathbf{z}_{i+1}$. It is dependent on the uncertainty at time i, the state vector estimate at time i, the control action taken at time i, and the set of features observed at time i + 1. However, as the estimate $\hat{\zeta}_i$ depends on the noisy observations, $\mathbf{z}_1,\ldots,\mathbf{z}_i$, when $\mathbf{P}_{i+1}$ is to be predicted at time 0, not only is it necessary to know the set of features that can be observed, $J_1,\ldots,J_{i+1}$, but also the real observations $\mathbf{z}_1,\ldots,\mathbf{z}_i$. In the special case when the feature set is not variable and the observation noises are very small (thus the innovations are small and $\hat{\zeta}_i \approx \hat{\zeta}_0, i=1,\ldots,M$), it is possible to predict $\mathbf{P}_{i+1}$ by only predicting $J_1,\ldots,J_{i+1}$; this is the method used in Sim and Roy (2005).

In general, as the information gathering task progresses, the knowledge about the environment accumulates. Firstly, there is increasing knowledge about the total number of features in the environment, thus the control system dimension and the state constraints become more defined. Secondly, the feature location estimates become increasingly accurate, thus J_{i+1} and $\mathbf{z}_{i+1}$ are more predictable and hence the uncertainties involved in the dynamics of the control systems become smaller. Therefore the control system (4.11) is a system with a gradually identified model.

4.4. Model Predictive Control for Active SLAM

As shown in the previous section, the model of the Active SLAM problem is gradually identified. It may be possible to use a general representation, in the EKF update equations (4.11), such as Markov Decision Process (MDP) or Partially Observable

Markov Decision Process (POMDP), to take into account all the possible dimensions of the state vector, all the possible J_{i+1} features and $\mathbf{z}_{i+1}$ observations based on knowledge about the feature distribution in the environment and the sensor noise. However, such a model would be excessively complex and would render the planning problem intractable. Following from Chapter 3, it is proposed to use Model Predictive Control (MPC) for Active SLAM. This provides for a simple but tractable strategy where new knowledge is exploited at every time step.

In MPC, recall that at each time step *i*, an optimal control problem of fixed planning horizon of *D* steps is solved and a sequence of *D* control actions

$$\mathbf{u}_i, \mathbf{u}_{i+1}, \ldots, \mathbf{u}_{i+D-1} \tag{4.14}$$

is obtained, but only the first control action, $\mathbf{u}_i$, is applied. This strategy requires a fixed control system model at each step to compute the *D*-step optimal control action. Therefore, a means to predict J_{i+1} and $\mathbf{z}_{i+1}$ for *D* steps is required.

4.4.1. Multi-Step Prediction for Active SLAM

In order to plan within a limited computational capacity using MPC, it is proposed to simplify the multi-step predictions at each time step. The mean value of the current state estimate may be used to predict J_{i+1} for *D* steps assuming no new features and perfect process models. The predicted J_{i+1} features will be used to predict $\mathbf{z}_{i+1}$ assuming zero observation innovations. Another assumption to be made in the *D*-step optimal control problem is that the control constraints and the state constraints are constant. Thus, to enable the planning of multiple steps in Active SLAM, the following assumption is formally stated.

Assumption I. For any possible control sequence $\mathbf{u}_0, \ldots, \mathbf{u}_{D-1}$, the group of features that are predicted to be observed at time $i+1$ $(0 \le i \le D-1)$ incorporate all the features that constitute the observation $\mathbf{z}_{i+1}$. Moreover, the safe region R_{i+1} contains the safe region R_0 as no new features are detected.

In Active SLAM, the real observations $\mathbf{z}_{i+1}$ are required to update the state. These observations for $i = 0,\ldots, D - 1$, are not available at time $i = 0$, and need to be predicted. In an EKF, there is an underlying assumption that the distribution of the true locations $\mathbf{x}_{i+1}$ is Gaussian with mean $\hat{\mathbf{x}}_{i+1}^{-}$. The observation noises are also assumed to be Gaussian with zero mean. Thus it can be stated that at time $i = 0$, the innovations $\mathbf{z}_{i+1} - \mathbf{h}_{i+1}\left(\hat{\zeta}_{i+1}^{-}\right)$ are all random variables with zero mean. This leads to the following assumption.

Assumption II. The innovations at any time i+1 are zero, i.e.

$$\mathbf{z}_{i+1} - \mathbf{h}_{i+1}\left(\hat{\zeta}_{i+1}^{-}\right) = 0 \tag{4.15}$$

for all $i = 0, \ldots, D - 1$.

At time $i = 0$, under Assumptions I and II, the D-step optimal control problem for the gradually identified system becomes an optimal control problem for an ordinary deterministic control system. This allows (4.14) to be computed in the following multi-step look-ahead control optimisation.

***D*-step optimal control problem**. Given $\hat{\zeta}_0$ and $\mathbf{P}_0$, find (4.14) such that trace($\mathbf{P}_D$) is minimized, where $\mathbf{P}_D$ is given by the following equations:

$$\begin{aligned}
&\mathbf{P}_1^- = \mathbf{\Gamma}_0\mathbf{P}_0\mathbf{\Gamma}_0^{\mathrm{T}} + \mathbf{Y}_0\mathbf{P}_{\mathbf{w}}\mathbf{Y}_0^{\mathrm{T}}\\
&\hat{\zeta}_1^- = \mathbf{f}\left(\hat{\zeta}_0, \mathbf{u}_0, 0\right)\\
&\mathbf{H} = \nabla_{\zeta}\mathbf{h}_1\big|_{\hat{\zeta}_1^-}\\
&\mathbf{S}_1 = \mathbf{H}\mathbf{P}_1^-\mathbf{H}^{\mathrm{T}} + \mathbf{\Sigma}_1\\
&\mathbf{K}_1 = \mathbf{P}_1^-\mathbf{H}\mathbf{S}_1^{-1}\\
&\mathbf{P}_1 = \mathbf{P}_1^- - \mathbf{K}_1\mathbf{S}_1\mathbf{K}_1^{\mathrm{T}}\\
&\hat{\zeta}_1 = \hat{\zeta}_1^-
\end{aligned} \tag{4.16}$$

$$\begin{aligned}
&\vdots \\
\mathbf{P}_D^- &= \mathbf{\Gamma}_{D-1}\mathbf{P}_{D-1}\mathbf{\Gamma}_{D-1}^{\mathrm{T}} + \mathbf{Y}_{D-1}\mathbf{P}_{\mathbf{w}}\mathbf{Y}_{D-1}^{\mathrm{T}} \\
\hat{\zeta}_D^- &= \mathbf{f}\left(\hat{\zeta}_{D-1}, \mathbf{u}_{D-1}, 0\right) \\
\mathbf{H} &= \nabla_{\zeta}\mathbf{h}_D\Big|_{\hat{\zeta}_D^-} \\
\mathbf{S}_D &= \mathbf{H}\mathbf{P}_D^-\mathbf{H}^{\mathrm{T}} + \mathbf{\Sigma}_D \\
\mathbf{K}_D &= \mathbf{P}_D^-\mathbf{H}\mathbf{S}_D^{-1} \\
\mathbf{P}_D &= \mathbf{P}_D^- - \mathbf{K}_D\mathbf{S}_D\mathbf{K}_D^{\mathrm{T}} \\
\hat{\zeta}_D &= \hat{\zeta}_D^-
\end{aligned} \tag{4.17}$$

Following this, at time *i*, Assumptions I and II need to be made from time *i* to *i* + *D* and the objective is to *minimise*

$$\mathrm{trace}(\mathbf{P}_{i+D}) \tag{4.18}$$

In the planning process, the possible future changes in the model are not considered, thus it is crucial to replan as soon as new state estimates are available.

4.4.2. Optimization Techniques

A range of optimisation techniques can be used to solve the *D*-step optimal control problem. An EETS was demonstrated in the previous chapter as a suitable technique in that it conducts a coarse exhaustive search. SQP was shown to increase the computational cost. SLAM itself is already computationally expensive and hence using SQP makes this problem computationally intractable. EETS is chosen in the following as the optimisation technique to perform the *D*-step optimisation.

4.4.3. Simulation Results for Active SLAM using MPC

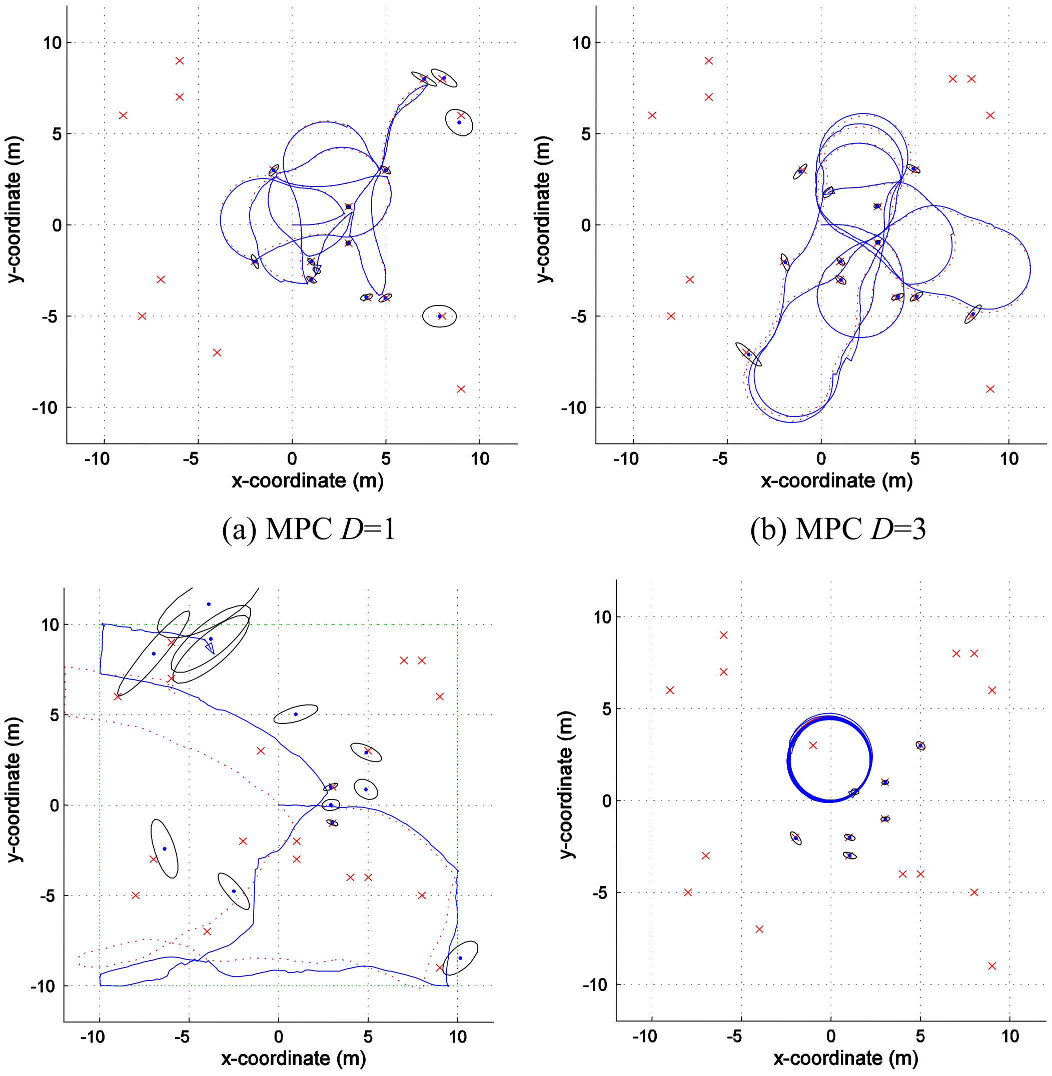
Simulations are conducted using a single robot with multiple features in a two-dimensional environment. Velocity and turn-rate are the available control inputs. Equipment specifications including the standard deviations, σ, for measurement and control noise are as follows: sensor field of view, ±π/8rad with σ=π/180rad for bearing noise; maximum sensor range, 5m with σ=0.2m; robot maximum velocity, 0.2ms^{-1} with $\sigma = 0.3\,\mathrm{ms}^{-1}$; robot maximum turn rate, π/180rads^{-1} with $\sigma = \pi/60$ rads^{-1}. As new features are detected, they are assigned an exclusion (no-go) zone of 0.3m around them.

4.4.3.1. Performance of MPC in Active SLAM

In the first set of simulations, an environment is considered where 20 features are scattered in a 400m^2 search space as seen in Figure 4-2. The tasks for the robot are to discover and localise these features while maintaining a good estimate of its pose. A terminal time of T=1000 discrete time steps is allotted to the robot. The length of each time step from i to i+1 is set to 0.4 seconds.

Examining now the single-step planning strategy, the robot may observe a new feature at pose $\mathbf{x}_i$ and in the subsequent steps, $\mathbf{x}_{i+1}$ and $\mathbf{x}_{i+2}$, the robot may continue to observe this feature. Suppose the robot moves past this feature at step i+3, such that at $\mathbf{x}_{i+3}$ the feature is no longer within the sensor field of view. Now the trajectory and kinematic constraints may be such that multiple steps are required for the robot to turn sufficiently for the feature to re-enter the field of view. With the single step planning strategy there will not be the necessary length in the planning horizon to actively revisit this feature, i.e. it falls out of the planning horizon and thus contributes no information to the algorithm. Figure 4-2(a) illustrates the problem, with trajectories generated for planning horizon D = 1 with N_ω= 5 control options. Clearly, the whole map is not uniformly targeted, also note the two features with larger uncertainties; the robot trajectory has been biased to nearby features that are observed in the next step.

Computing multiple steps in the planning horizon improves results. Now, the features detected may be revisited in an attempt to minimise the uncertainty of the map. Figure 4-2(b) illustrates where the trajectory is obtained with a planning horizon D=3 and N_ω=5 and for the sake of comparing the single-step and multi-step planning strategies, these two examples are selected for the similar number of features they identify. At solution end, i=T, the multi-step MPC strategy yields a smaller uncertainty, yet both strategies have a common failing in that features are only detected by chance as there is no explicit exploration strategy.



(a) MPC *D*=1

(b) MPC *D*=3

(c) Random Control

(d) Fixed Control

Figure 4-2 Robot Path and Feature Uncertainties

For the sake of some relevant discussion results are provided for random (Figure 4-2(c)) and fixed control (Figure 4-2(d)). Random control is a general approach when there is no information available for the planning of trajectories. The random control applies a random selection from a set of feasible control options and clearly provides a trajectory with a poor SLAM result. The robot heads predominantly in a straight line whilst swerving from left to right, under the influence of a constraint at the edge of the exploration space, it stops and turns randomly. This constraint is indicated by the green border. The results illustrate that although the exploration space is largely explored by the terminal time but the uncertainties of the robot and map features are quite high, thereby increasing the chance of an incorrect data association. Note the large ellipses of uncertainty around the features.

The fixed control (constant velocity and turn-rate) guarantees loop closure but not constraint satisfaction. The controls are arbitrarily selected. It gives the predictable result of Figure 4-2(d); the robot was unable to cover much of the area due to the restricted path. Although artificially simple, the simulation demonstrates how the robot localises correctly on its path as it continuously revisits these previously observed features. Note the small ellipses of uncertainty around the six observable features. A smaller turn-rate may be applied to enable the robot to move in a larger circle to obtain higher coverage, however this increases the chance that a new feature would be detected in its path and thus constraints would be violated.

4.4.3.2. Increasing Process Noise

The effect of process noise on the effectiveness of the trajectory planning is always of interest. Recall the underlying assumption in applying the multi-step prediction, Assumption II, where the innovations are assumed to be zero; it is valid if the process noise is small. Increasing the process noise would inherently increase the innovation. What follows are trials to examine the effect of the control noise on the effectiveness of the planning strategy. Table 4-1 gives a set of results for N_ω=3 control options, 20 features, 1000 loops and different process noise levels. In each case, the same initial conditions are set and the results are from an average of ten trials. Sensor limitations are removed to avoid large differences in the number of features detected between simulations. The table gives the average uncertainties and its ratio over the uncertainty for the single-step optimisation method. The trials show that the benefits of planning with a longer look-ahead decrease as the process noise increases. However the results in the table show that even with a high noise level in the last column, a multi-step planning still performs better than the single-step optimisation method. Obviously, increasing the observation noise would generate similar results as it would also increase the innovations. However this is not studied further as the true observation noise parameters would be used in practical implementations and cannot be varied through trajectory planning.

σ^2 ***Vel noise*** *(m/s)*	0.03		0.04		0.05		0.06	
σ^2 ***Turn noise*** *(deg/s)*	3		4		5		6	
D	$\mathbf{P}_{Trace}$	$\mathbf{R}_{Final}$	$\mathbf{P}_{Trace}$	$\mathbf{R}_{Final}$	$\mathbf{P}_{Trace}$	$\mathbf{R}_{Final}$	$\mathbf{P}_{Trace}$	$\mathbf{R}_{Final}$
0 (fixed ctrl)	0.0065	1.14	0.0084	1.17	0.0111	1.22	0.0153	1.31
1	0.0057	1.00	0.0072	1.00	0.0091	1.00	0.0117	1.00
3	0.0052	0.92	0.0067	0.93	0.0085	0.94	0.0112	0.96
5	0.0047	0.83	0.0061	0.85	0.0080	0.88	0.0106	0.91
7	0.0042	0.73	0.0055	0.77	0.0075	0.83	0.0107	0.92

Table 4-1 Results from Increasing Process Noise

where $\mathbf{P}_{Trace}$= *Trace*($\mathbf{P}_T$)/(number of rows in P) averaged over 10 trials

and $\mathbf{R}_{Final}$= $\mathbf{P}_{Trace}$ / $\mathbf{P}_{Trace}$ (*D*=1)

It should be noted that the benefits attainable by applying MPC over other methods also depend on many factors. These factors include the environmental conditions (e.g. feature density) and system constraints (such as field of view, maximum turn-rate), etc. For example, in a confined space with near complete sensor field of view, all methods may locate the features well and planning is not as critical. When the sensing is limited, the position of the sensor becomes imperative to the mapping of the environment and coverage also needs to be considered in the trajectory planning. It is apparent in Figure 4-2 (a) and (b) that the robot does not visit features that have not come into the sensor range as there is no explicit exploration strategy. This limitation is addressed in the next section.

4.5. Attractor aided MPC for Active SLAM

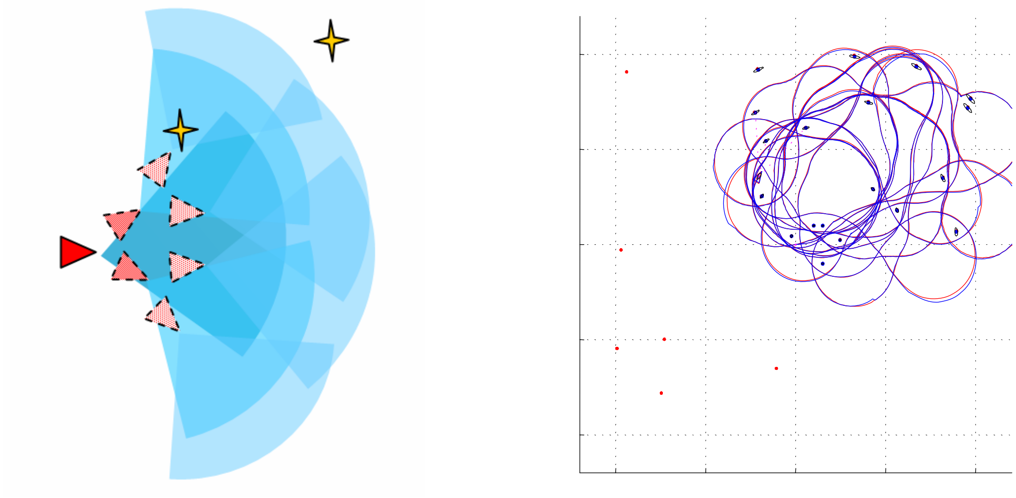
4.5.1. Limitations and Insight of MPC for Active SLAM

MPC with a few steps look-ahead is principally a local planning strategy and thus suffers from a fundamental shortcoming, along with other similar local planners. MPC with limited number of steps is unable to perceive beyond the set planning horizon. The planning algorithm will only optimise using features visible within the planning horizon, while ignoring distant features, even if they exist in the map. As illustrated in

Figure 4-3(a), map features outside the scope of the planning horizon are neglected from the optimisation.

MPC is also incapable of proactively exploring. The robot does not consider moving to unexplored areas of the environment. In an illustrative result, (see Figure 4-3(b)), the robot is given an area to explore. Selecting T=30000 in the simulation will provide coverage equivalent to the case of $T = \infty$ for the given finite space. The features detected by the robot are well localised due to the optimization of the map within the sensed area. Yet as there is no explicit exploration strategy in MPC, the features are detected by chance and thus six features remain undetected even after the robot has sufficiently localised the known features.

It is desirable to incorporate long term goals and exploration in the trajectory planning whilst maintaining simplicity and low computation. On the other hand, extending the planning horizon to enable the robot to incorporate long-term rewards is computationally expensive. Weighting utilities for multiple objectives for SLAM in an objective function, as in Frew (2005), may provide incentives for exploration and provide a means to incorporate long term rewards. It is however difficult to express the value of long-term rewards and tuning weights is cumbersome.



(a) Limited scope of MPC

(b) MPC path

Figure 4-3 Limitations of MPC

By observing simulations, some insights into the behaviour of the robot during the SLAM process are obtained. Key behaviours demonstrated in the implementation of MPC for point-feature SLAM are revealed below and although these are readily apparent from examining the algorithms, they are of interest to the concept introduced in the subsequent section:

a) When the robot is surrounded by features with large uncertainty, the robot itself will also have large uncertainty. An observation of a well-defined feature will improve the robot pose estimate and that of other features. Hence the information gain in observing the well-defined feature is large and the robot would be influenced to move in the direction of this feature so as to optimise the objective function in MPC.

b) When the robot pose has small uncertainty and the map includes features with large uncertainty then the robot will gain more information about the feature locations by making further observations. The large information gain in the direction of these features then influences the robot to move in this direction, so as to optimise the objective function in MPC.

c) Small uncertainty in the robot pose generally correlates with small uncertainty for location estimates of surrounding features. Little information gain will be made as time progresses. However if a feature with large uncertainty comes into the range of the planning horizon then robot will maximise information gain by heading in the direction of this feature, again optimising the objective function in MPC.

Furthermore, there are instinctive instances where certain tasks in SLAM are appropriate. These tasks include: a) covering unknown areas, b) observing well-defined features to localise the robot and c) localising poorly-defined features.

a) When the map features are well-defined and the robot location is also well-defined, then the robot should explore. The robot should seek opportunities to explore to minimise the time required to complete coverage of the environment. Exploration of unknown areas is important in Active SLAM to ensure good coverage.

b) If the estimate of the location of the robot has large uncertainty and the map feature estimates also have large uncertainty then there is not beneficial to move into unexplored areas. Doing so would result in an inaccurate map. There is a competing requirement in SLAM for the robot to regularly revisit known features to localise so as to not become lost. The larger the uncertainty of the robot pose estimate grows when exploring, the greater the chance the robot would make an incorrect data association. Revisiting well-known features is therefore critical for maintaining consistent estimates. However, if the robot revisits known features too frequently then there will be a corresponding increase in the time to complete the exploration.

c) If the estimates of the locations of map features contain large uncertainty and the uncertainty of the robot pose is low then it is necessary to localise these features to maximise the accuracy of the map. As new features are detected they will have large error in their estimated location. Subsequent observations of these features decrease this error. To be effective in the localisation of new features the robot should revisit well known features between subsequent observations of new features. This reduces the impact of odometry error in the robot location estimate and increases the correlation between the well-known features and the new features.

In order to conduct SLAM efficiently, high level planning is therefore required to instruct the robot when to focus on certain tasks. For each of these tasks it is desirable to change the trajectory of the robot such that it performs the particular task. Additionally it is advantageous to have a mechanism to switch between these tasks at appropriate instances during the SLAM process. From the insights gained by studying the behaviour of the robot, an attractor strategy is developed to aid MPC in the planning which allows for more effective task selection. It is proposed to use a feature as the attractor such that the properties inherent in the optimisation of the objective function can be exploited. That the attractor is included as a feature brings much adaptability to MPC. In Active SLAM, the objective function in the MPC strategy is predominately driven by the uncertainty of the features in the map. MPC still has one objective, i.e. to minimise uncertainty, and applying the insights discussed above, the properties of the attractor (as a feature) may be adjusted so as to achieve certain goals. The details of this strategy follow.

4.5.2. Concept of an Attractor and Reference Point

The attractor strategy is a novel heuristic that adds a virtual feature to which the robot is directed to move in order to facilitate the SLAM process. The attractor enables the incorporation of global information and high level goals into the planning process by converting the long-term goals to potential information gain from observations within the short planning horizon of MPC. This is also the mechanism that enables the robot to proactively explore. The attractor strategy allows the MPC framework previously established to be maintained. As the attractor is a virtual feature, it is only included in the estimation process during the trajectory planning phase. Once the control action for the robot is selected, the EKF proceeds as normal without the attractor.

The function of the attractor is to influence the gross motion of the robot towards desired locations by influencing the information gain of particular control actions. At each time step, *i*, the attractor is placed in the line of sight to an assigned reference point. Through tactical placement of the virtual feature, i.e. the attractor, the information gain for certain control actions can be increased and the motion of the robot is then directed towards a desired goal, which is the reference point. The reference point may be a specific point or on a feature of interest that needs to be observed based on whether the robot is exploring or mapping. To be able to influence the MPC objective function, the attractor must lie in a location visible to the robot within the planning horizon. The attractor should also be placed further than the distance the robot can move in the planning horizon such that the robot does not consider the attractor an obstacle during the planning horizon or possibly move past the attractor in the initial steps. It is proposed to place the attractor at a range equivalent to the robot’s current sensor range, which meets the above two requirements as illustrated in Figure 4-4. With this approach, the local MPC strategy is able to incorporate the high level decisions and goals through the incentive provided by the attractor. Placing the attractor closer to the robot results in higher information gain and renders the attractor's influence greater. However, the optimal distance from the robot for the attractor to be placed is yet unresolved and needs further investigation.

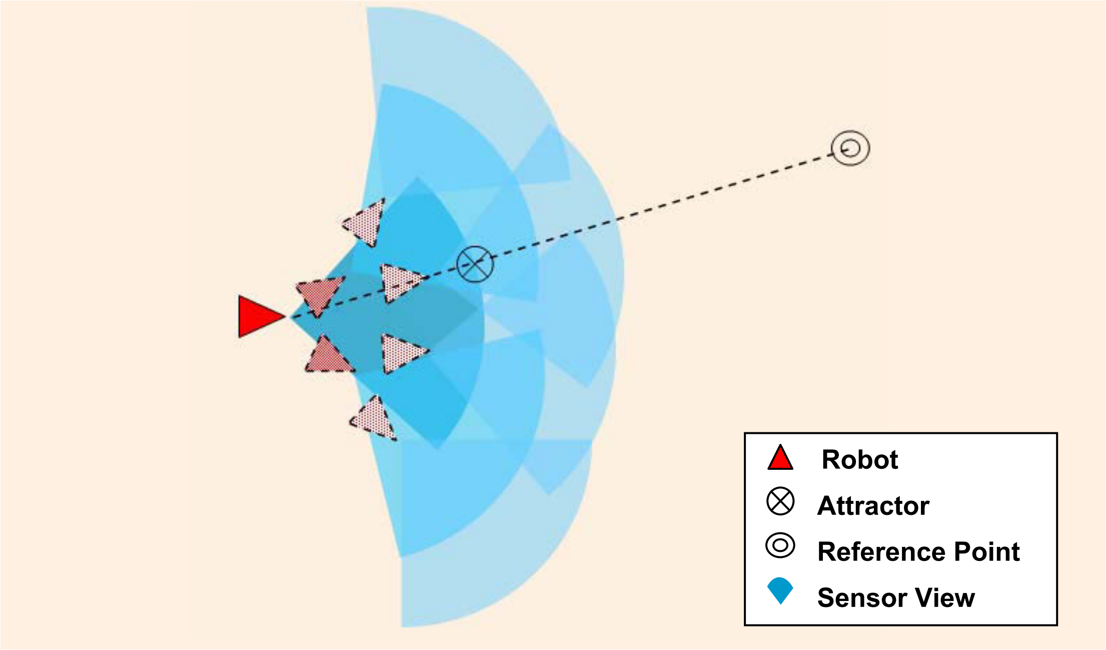


Figure 4-4 Placement of Attractor

4.5.3. State Machine for Active SLAM

4.5.3.1. Defining Modes in the State Machine

For efficient implementation of Active SLAM, a system is devised with three operating modes for the robot: *explore*, *improve localisation* and *improve map*, which corresponds to the three main tasks in SLAM. The aim of the trajectory planning is to maximise the accuracy of a built map and to maximise coverage of an unexplored area within a prescribed terminal time. This is achieved in two stages: (i) first maximise coverage while maintaining a map at a certain level of accuracy; (ii) then once the environment has been completely explored, minimise the uncertainty of the map. In Stage (i) all three modes are active to maximise coverage. Once the area has been explored, then it switches to Stage (ii) where only the *improve localisation* and *improve map* modes are active. Figure 4-5 illustrates these modes and their selection process which in turn are based on the uncertainty of the estimates.

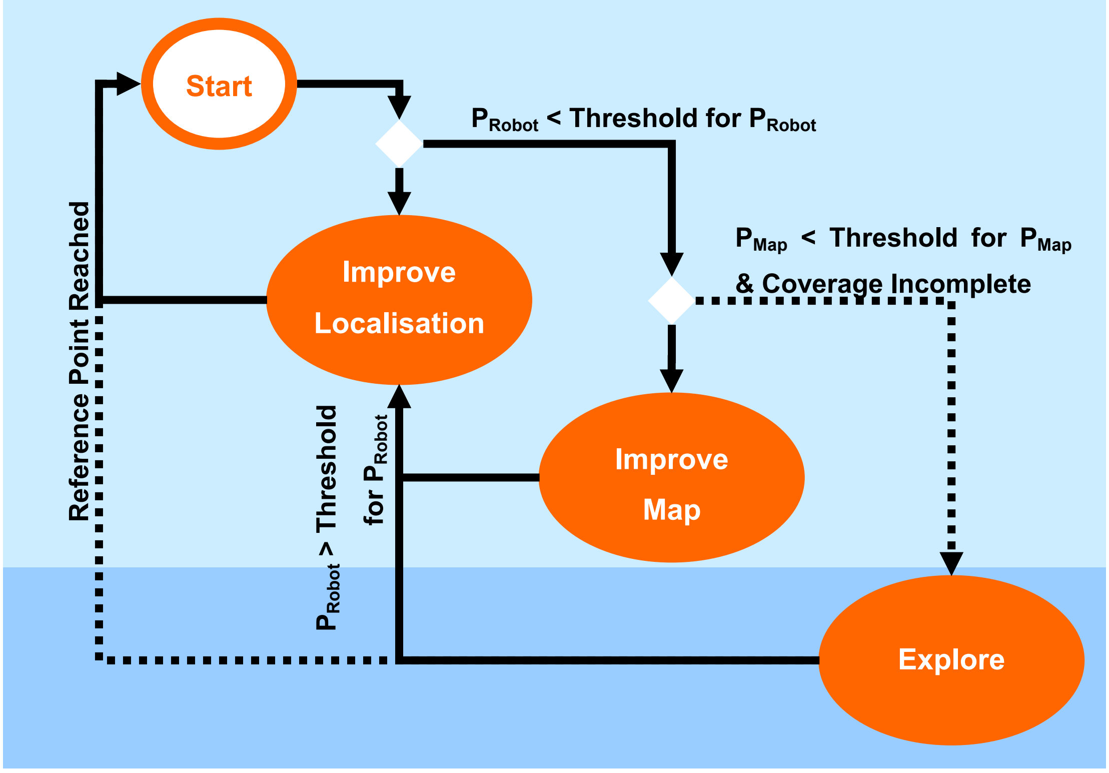

Start
P_{Robot} < Threshold for P_{Robot}
Improve Localisation
P_{Map} < Threshold for P_{Map} & Coverage Incomplete
Improve Map
Explore
Reference Point Reached
P_{Robot} > Threshold for P_{Robot}

Figure 4-5 State Machine for the Attractor Strategy, where Stage (i) incorporates both the light and dark blue zones; and Stage (ii) is defined by the light blue zone only

4.5.3.2. Mode Transitions

The robot determines its current mode in the state machine and appropriate instances to change modes based on the uncertainty of the state estimation. If the robot's location uncertainty exceeds a threshold, then the robot changes its mode to *improve localisation*. When both the uncertainty of the robot and map features is below a predefined threshold, the robot mode becomes *explore*. If the uncertainty of the map is high and the robot location estimate is acceptable, the mode would be to *improve map*. Upon visiting a poorly estimated feature under the *improve map* mode, the robot either repeatedly switches between *improve localisation* and *improve map* modes or chooses to *explore* depending on the uncertainty of the system state.

The robot remains in its current mode until either of the following two events occurs: a) the desired point or feature of interest falls into range of the sensor, or b) the robot's uncertainty exceeds a predefined threshold, triggering a mode transition to the *improve localisation* mode.

4.5.3.3. Implementation of the Attractor for the Three Modes

The modes in the state machine are facilitated by selecting suitable destination reference points to attract the robot. When the mode is *explore*, an exploration point is selected. When the mode is *improve localisation*, a nearby feature with a well-defined location estimate is selected and when the mode is *improve map*, a nearby feature with large uncertainty in the location estimate is selected. Selection of the particular exploration point or feature of interest is based on a heuristic of minimum distance such that it may be reached quickly.

For the selection of features of interest as the reference point, thresholds are set to determine if the uncertainty of the feature location is large or small. If no features are found to meet the threshold requirement, then the feature with the largest uncertainty or the smallest uncertainty is selected depending on the mode.

The thresholds for switching modes and selecting features as reference points are determined based on knowledge from previous experimental trials. For example, if the robot has a small maximum turn-rate it may need to turn a large circle to return to known features, in which case the threshold to switch to the improve localisation mode may be smaller. If the robot's sensor noise is large, then the feature estimates will generally have a larger uncertainty and converge at a slower rate, thus the thresholds for selecting a feature of interest may be larger.

In each mode, the attractor is placed in the direction of the reference point at a distance equivalent to the robot's sensor range. The position and covariance of the attractor also depends on the current mode of the robot and the current knowledge of the system, i.e. $\mathbf{P}_i$ and $\hat{\zeta}_i$.

***1) Explore*:** For this mode the closest exploration point is selected from an exploration point list. Initially the entire search space is covered uniformly with exploration points, each representing an unexplored area. Exploration points are removed once they fall within the range of the sensor. If the exploration points are distributed too sparsely then certain areas may be left unexplored, whereas the assignment of many exploration points will result in increased computations. Thus for this study, these points are distributed with a distance proportional to the robot's sensor range. Figure 4-6 displays an example initial environment; the exploration points are indicated by light blue dots, the feature distributions are depicted by the red dots and the initial robot pose is indicated by the blue triangle in the centre of the map.

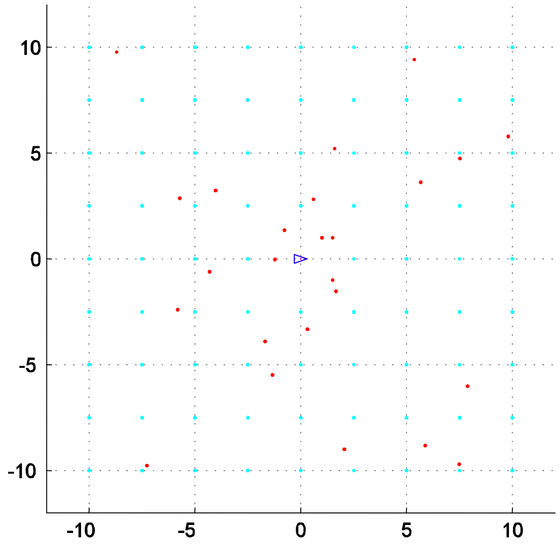


Figure 4-6 Initial Environment Setup (m)

An artificial state, $\zeta_{artificial}$, and an artificial covariance, $\mathbf{P}_{artificial}$ are created for the trajectory planning to include the state and covariance of the attractor. The attractor is added by initialising a new feature as if it was observed at the desired placement, i.e. the system state vector is now augmented with the location of the attractor. The new estimate becomes

$$\begin{aligned}\mathbf{P}_{artificial} &= diag\left(\mathbf{P}_i \quad \mathbf{P}_{attractor}\right)\\ \zeta_{artificial} &= \left[\hat{\zeta}_i^{\mathrm{T}} \quad \boldsymbol{\lambda}_{attractor}{}^{\mathrm{T}}\right]^{\mathrm{T}}\end{aligned} \tag{4.19}$$

where $\boldsymbol{\lambda}_{attractor}$ is the location of the attractor and $\mathbf{P}_{attractor}$ is the associated covariance generated from initialising a new point feature in the desired location.

Figure 4-7 shows a robot in the *explore* mode with the attractor having a large uncertainty. The robot is able to observe the attractor as a new feature and it moves in attempt to localise the attractor. This mode is no longer active once all the exploration points are covered.

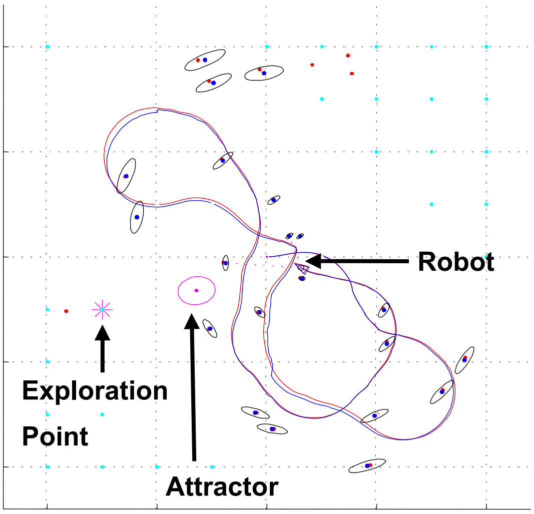


Figure 4-7 Attractor - Explore

2*) Improve Localisation*: For this mode, a well-defined feature is selected. Here, only the state vector, $\hat{\zeta}_i$, is changed. The state of the feature selected to be the reference point is altered to be at the desired placement of the attractor. The covariance $\mathbf{P}_i$ is left unchanged and thus the effect of the correlations from observing the well-defined feature is maintained, i.e.

$$\begin{aligned} &\mathbf{P}_{artificial} = \mathbf{P}_i \\ &\zeta_{artificial} = \hat{\zeta}_i \\ &\zeta_{artificial}\left(j_s\right) = \boldsymbol{\lambda}_{attractor} \end{aligned} \tag{4.20}$$

with j_s representing the index of the feature selected.

Figure 4-8 is a snapshot of the planning during the *improve localisation* mode, where the robot's uncertainty is quite high. The attractor in this case has a low uncertainty equivalent to the uncertainty of the feature selected. This feature is indicated by the magenta asterisk. The robot sees the attractor as a feature with a well-defined location and moves towards it to localise itself.

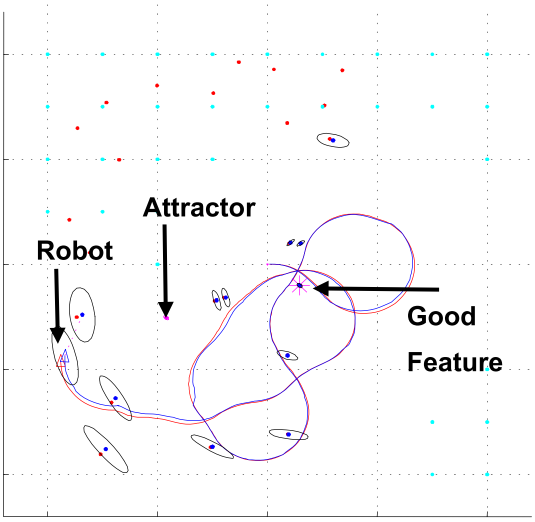


Figure 4-8 Attractor – Improve Localisation

3***) Improve Map*****:** For this mode, the artificial state and covariance is created as in (4.20), where only the state vector, $\hat{\zeta}_i$, is changed for the feature with large uncertainty selected. The covariance $\mathbf{P}_i$ is unchanged and thus the large uncertainty of the feature becomes that of the attractor.

Referring now to Figure 4-9, a snapshot of the *improve map* mode, note that the uncertainty of the attractor is equivalent to the uncertainty of the feature selected for the reference point. The robot sees the attractor as a feature with large uncertainty and moves towards it to localise it. Once all the exploration points have been visited, the robot may continue improving the accuracy of the map.

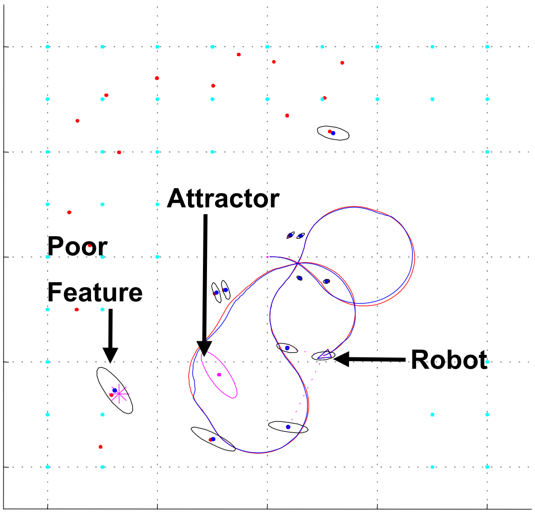


Figure 4-9 Attractor - Improve Map

4.5.4. Simulation Results for Attractor aided MPC

In the simulation, the robot is to explore and map an area of 400m^2 in a terminal time of T=3000. The environment consists of 22 randomly located features with three features visible from the starting position. The sensor range of the robot is set to 5m with field of view of $\pm\pi/4$rad. The thresholds for mode transitions and feature selection were based on the maximum eigenvalue of the feature or robot uncertainty and are listed as follows: well defined feature < 0.02, poorly defined feature > 0.2, improve localisation mode when robot uncertainty > 0.1, improve map mode when feature uncertainty > 0.2. Table 4-2 displays the results obtained using the proposed algorithm.

4.5.4.1. MPC+Attractor vs. MPC alone

The addition of the attractor improved coverage significantly. In comparing the coverage of MPC to MPC+Attractor, it can be observed in Table 4-2 that MPC+Attractor took 1606 time steps to cover 100% of the exploration space. After 3000 time steps MPC alone only managed to cover 88% of the exploration space.

Conversely, it is evident that there is a tradeoff between coverage and uncertainty reduction. The results from using the attractor obtained less accuracy. In the MPC+Attractor strategy, the robot is often led to the edges of the environment to ensure coverage. The exploration points may be far from a cluster of known features and there may be few or no features observed at these points. If a new feature is detected at these points then the uncertainty would consequently be large due to the robot traveling a long distance without making observations. When MPC is implemented without the attractor, the robot only traverses near known features and new features are only detected by chance. Hence, the new features detected would be reasonably close to known features. As a result the uncertainty does not grow excessively large. In Table 4-2, MPC alone achieved a slightly lower uncertainty of 0.0045 compared to 0.0048 from MPC+Attractor.

Comparison of Strategies for Active SLAM			
Method	**Area Covered (%)**	**Time Steps (*i*) to Complete Coverage**	**Trace($\mathbf{P}_T$)/(total rows in $\mathbf{P}_T$)**
MPC (3 steps) + Attractor	100	1777	0.0035
	100	1124	0.0035
	100	1643	0.0043
	100	1644	0.0082
	100	1760	0.0039
	100	1685	0.0056
Average	**100**	**1606**	**0.0048**
MPC (3 steps) without Attractor	86	N/A	0.0037
	91	N/A	0.0046
	84	N/A	0.0040
	80	N/A	0.0036
	95	N/A	0.0065
	92	N/A	0.0044
Average	**88**	N/A	**0.0045**
MPC (1 step) + Attractor	100	2760	0.0081
	100	2460	0.0092
	100	2195	0.0066
	100	2621	0.0043
	100	2594	0.0072
	100	2370	0.0088
Average	**100**	**2500**	**0.0074**

Table 4-2 Active SLAM Simulation Results

The performance of these strategies largely depends on the density of the features and the sensor range. The larger the sensor range, the higher the amount of features or the smaller the exploration space, the easier the exploration task would be. The differences in Table 4-2 are not influenced by these factors as they are kept constant. Variance in the results is principally due to the random placement of features and the sensor noise.

4.5.4.2. Multi-step Planning vs. Single-Step Planning

The attractor method allows local information to be incorporated into the planning along with the long term goals. Further simulations were conducted to observe the influence of multi-step MPC as compared to single-step look-ahead with the presence of an attractor. From the data in Table 4-2, the single-step planning method, on average, took 56% longer to cover the area and the final uncertainty of the system was 54% higher. Multi-step planning may perform better than a single step due to the robot predicting further ahead and thus considers features that may be slightly further away. Hence the trajectory is optimised based on more local knowledge. This demonstrates that even with the attractor providing global knowledge, the local optimisation of nearby features continues to largely influence the system performance.

4.6. Discussion

4.6.1. Number of Prediction Steps in MPC

The optimal length of the planning horizon in MPC is application specific. Intuitively, when the uncertainty involved is large, a short planning horizon with immediate rewards is desired. This is due to the high likelihood that the optimal plan will change significantly once new information is acquired. If the uncertainty is small, longer plans generally equate to obtaining higher benefits. However, the planning horizon D is limited by the computational capacity of the planner. The selection of D should be made depending on the resources available.

It is shown by the simulation results in Section 4.4.3.2 that there is a limit to the benefit attained when the uncertainty of the system is large.

4.6.2. Optimality of MPC

Theoretically, the performance of *D*-step optimization is better than the performance of the single-step optimisation. If a fixed number of control options are considered and an EETS is used to select the best option, then the set of control options considered in the *D*-step optimisation is larger than the set of control options considered single-step optimisation. Additionally, all the control options considered in the single step optimisation are also considered in the *D*-step optimisation. Similarly, performing single-step optimisation is better than random control and fixed control because the fixed and random control solutions are two options considered in the one-step optimization. Returning to Assumptions I and II, the original planning problem has been simplified, and by only looking *D*-steps ahead, there is no guarantee that the proposed MPC strategy is the best for the overall planning problem. However, the simulations do show that it outperforms the other strategies in most cases.

4.6.3. Coverage Improved

MPC has been demonstrated to be a good planning strategy in the optimisation of information gain; however, coverage relies on the chance of detection of features. Incorporating an attractor to the MPC strategy improves the coverage significantly as shown in Table 4-2. There is still nevertheless no guarantee of complete coverage. One example is where there are large distances between features; the uncertainty of the robot pose may grow too large before new features can be detected. This is a fundamental limitation of SLAM where features need to be frequently observed to maintain a good estimate.

4.6.4. Obstacle Avoidance

In some cases (subject to the available robot control options such as minimum velocity constraints and maximum turn-rate) there are no obstacle-free control options. This problem is encountered more frequently in the single-step planning method where the robot does not consider possible obstacles after one step and moves too close to no-go-zones. In the current implementation the robot is forced to stop and then turn randomly on the spot. However, other types of robots, such as UAVS, are not necessarily able to stop in mid-flight. This is one of the situations where it is worthwhile to plan more than a single step than to merely obtain more information. Planning obstacle free trajectories for fast robots is a challenging topic when uncertainties in the environment exist.

4.6.5. Localisability

Even with the incentive provided by the attractor to localise when necessary, there is no guarantee that the robot will not lose its location. A common reason for the robot losing its location is by making an incorrect data association when uncertain of its pose resulting in an inconsistent pose estimate. In some cases, depending on the robot's maximum turn-rate and control options available, the robot may move to the outskirts of the exploration space to observe new features and then require much time to turn and observe previously initialised features once again. If the robot is uncertain of its pose when returning to the exploration space, then the first observation may be incorrectly associated.

An approach to improve localisability would be to use more robust methods for data association. The nearest neighbour approach (Dissanayake et al. 2001) is currently implemented, but other approaches such as the joint compatibility test (Neira and Tardós, 2001) may reduce the occurrence of an incorrect association.

4.7. Summary

In this chapter, Model Predictive Control (MPC) is first proposed as a strategy for planning in a point-feature-based SLAM that uses an EKF in the estimation process. The effectiveness of the strategy is illustrated through simulated trials.

The MPC strategy is suitable for planning in SLAM, since:

(a) Changes in the environment including new features are managed by utilising updated models at each time step;

(b) Dynamic constraints which cannot be detected a priori are incorporated into the planning as new features are detected or the feature location estimates change;

(c) The prediction time horizon is flexible and can be set within the constraints of computational capacity;

(d) The strategy provides an improved control policy compared with single-step planning.

MPC alone has a number of limitations in that no explicit mechanisms for exploration or for high level planning are available. To introduce high level planning and exploration in Active SLAM, a novel technique of using an attractor together with MPC is developed. The attractor strategy is implemented using a state machine where three modes (a) improve localisation, (b) improve map and (c) explore are defined. These modes are based on the fundamental tasks inherent in SLAM. The attractor is implemented as a virtual feature as the optimisation in MPC is predominately governed by the features in the planning horizon. Simulation results show that combining MPC with an attractor further improves the SLAM result. Coverage is found to be significantly improved by the attractor while MPC is effective in optimising the information gain. Planning with MPC with the presence of the attractor using 3-step look-ahead is found to perform better than planning with a single step. Although the optimality of MPC or Attractor aided MPC cannot be guaranteed, using an attractor with MPC for Active SLAM is a good approach in terms of coverage, efficiency and accuracy.

The SLAM problem considered in this chapter uses points as features. Lines are natural features that exist in structured environments and SLAM using lines as features provides a more informative map. In the next chapter, a trajectory planning strategy is developed for SLAM with line features. This strategy is then implemented on a Pioneer2DX robot with a laser scanner to demonstrate Active SLAM in an indoor environment.

Chapter 5. Active SLAM with Line Features

5.1. Introduction

This chapter investigates trajectory planning for line-feature-based SLAM. The objective is to extend the MPC and attractor strategy developed in the previous chapter to actively explore and build maps in structured environments. Lines are a common feature in indoor environments and can be extracted from observations acquired by a laser range finder. Representing the environment with lines instead of point features result in a compact map with a smaller number of states. A map based on lines may also be more informative than point features due to the use of more measurements from the laser range finder. Since applying the EKF to line-feature SLAM has been found to generate inconsistent estimates in large scale environments (Rodriguez-Losada et al. 2006), unless geometric constraints are exploited, it was decided to develop another estimator for line-feature SLAM. A Smoothing and Mapping (SAM) technique was found to give consistent estimates and is used as the underlying estimation strategy. In the SAM state vector, the entire robot trajectory is retained such that linearization errors embedded into the robot pose are not accumulated. Based on the estimates obtained from this method, a planning strategy for efficient mapping and exploration in a structured environment is developed.

5.2. Background

There has been extensive research on the extraction of lines from a 2-D laser scan (Nguyen et al. 2005) and methods for increasing the accuracy and speed of the extraction process have been developed (Alempijevic 2004). Nguyen et al. (2005) compared six popular algorithms for line extraction from a two-dimensional laser range finder. It was found that algorithms that take advantage of the sequential property of laser scans had faster computation times. Algorithms such as the Hough Transform were slower but more accurate. Alempijevic (2004) developed an algorithm which pre-processed the laser scans to detect the slopes of lines, taking advantage of the sequential property of laser measurements. This allowed for a smaller region of the Hough Transform to be computed in extracting an accurate line.

SLAM using line features has been previously achieved (Garulli et al. 2005; Yuen and MacDonald 2003). Garulli (2005) used the EKF as the basis of the estimation and determined the covariance of the observation noise from the statistical properties of the laser range and bearing measurements. Yuen (2003) used Sequential Monte Carlo (SMC) SLAM to perform the estimation and demonstrated that performance is better for line features than with the EKF. Several variations for performing SLAM with line features have been developed. The SPMap (symmetries and perturbations map) (Castellanos et al. 1999) is one such method, where the robot pose and each feature are treated as geometric entities represented by a quadruple that includes a location vector, a perturbation vector, a covariance matrix and a self-binding matrix. A hierarchical multi-scale strategy (Pfister and Burdick 2006) is another approach in which features are grouped into blocks to reduce computation time.

Recently, the traditional implementation of the EKF has been shown to produce inconsistent estimations when used for SLAM in large-scale structured environments (Rodriguez-Losada et al. 2006). Rodriguez-Losada et al. (2006a), achieved real-time SLAM for large indoor environments using the EKF framework with ideas adapted from the SPMap. The computation cost involved in large scale EKF SLAM is reduced by using local sub-map fusion. The inconsistency from the EKF is overcome using shape constraints of collinearity, parallelism and perpendicularity. Shape constraints require prior knowledge of an environment but are a reasonable assumption for most structured environments.

In Dellaert and Kaess (2006), the SAM algorithm was presented as an accurate and fast method to perform SLAM. Compared against the EKF, results showed SAM to be faster for large-scale problems while providing consistent estimates. The method was only implemented for point-feature SLAM but the framework allows for line features to be estimated. It is viewed as a generic strategy that can be applied to mapping indoor environments without applying shape constraints.

There are many differences in optimising for information gain between SLAM with point features and SLAM with line features. In a point feature scenario, a robot mapping beacons would only see the beacon if it is near it, thus it would observe it for a small amount of time. On the other hand, when the robot is observing line features, the same feature would be observed for a long time. For this reason, the performance of MPC without an incentive for exploration for line-feature Active SLAM is likely to be significantly worse in terms of coverage as compared to that of point-feature based SLAM. Consider a scenario of a robot walking down a corridor, the line feature that the robot would observe at the start of the corridor would be the same feature that it observes at the end of the corridor. If it was to optimise for information gain then it would not move to the other end of the corridor. The robot would have a higher uncertainty due to accumulated errors from traversing and the robot would not gain more information from looking at the same feature with a higher uncertainty.

In addition, a robot mapping line features may observe the same feature at different places. Consider two rooms with common or aligned walls, which is common in most structured environments; a robot can observe the same line feature in the second room without returning to the first room.

Another difference between line-feature and point-feature SLAM is that in line-feature SLAM there are generally less features to observe but more features are required to localise the robot. With point-feature SLAM only two points are required for the robot to localise, however with line-feature SLAM the robot moving down a corridor with two parallel line-features is unable to localise itself along that corridor.

To the author's knowledge, Active SLAM using line feature estimation is yet to be studied. Exploration in structured environments is quite different to trajectory planning in open space. Corridors and doorways place constraints on the robot's trajectory. Additionally lines may obscure other lines such that the prediction of information from observations in the planning horizon requires knowledge of their visibility.

5.3. Trajectory Planning Problem for Line-feature SLAM

Suppose a mobile robot is placed in an unknown structured environment. The robot is required to explore an unknown area and produce a map. The trajectory of the robot needs to be optimised for map accuracy, area coverage and coverage time.

The following sections present the notations and formulate the trajectory planning problem.

5.3.1. Notations

The pose of the robot is denoted by the following symbols.

$X(1:i)\triangleq\{\mathbf{x}_m\}$	All the robot positions from which observations are made
$m=1,...,i$	Time steps where observations are made, where *i* is the index of the current pose
$\mathbf{x}_i=[x_i;y_i;\phi_i]$	Elements of robot pose

Observations are taken at each time step *i*, and are described by:

$Z(1:K)\triangleq\{\mathbf{z}_k\}$	Set of observation measurements with noise
$k=1,...,K$	Number of all observations from time *m*=1…*i*
$\mathbf{z}_k=[\alpha_k^{local};d_k^{local}]$	Observation to a line feature

The line features are detected and their properties are estimated during the mapping process are denoted by:

$L\triangleq\{\boldsymbol{\lambda}_j\}$	Set of line features
$j=1,...,J$	Number of line features detected
$\boldsymbol{\lambda}_j=[\alpha_j^{global};d_j^{global}]$	Line feature in global reference frame

5.3.2. Representation of a Line

A line may be represented in many different ways. For the work in this chapter, lines will be represented in polar coordinates, with the line segments represented by the corresponding endpoints in Cartesian coordinates. This is to maintain a compact representation and simplicity where only two terms are required to describe the state of each line to be estimated. The additional information of the endpoints of line segments can be stored separately to be used in the trajectory planning.

5.3.2.1. Parameters of a Line

The two parameters that represent a line are *α* and *d*, where *α* is the angle of the perpendicular to the line from the reference origin and *d* is the distance from the reference origin to the line (Figure 5-1). The equation of the line is described by

$$x\cos(\alpha)+y\sin(\alpha)=d\,. \tag{5.1}$$

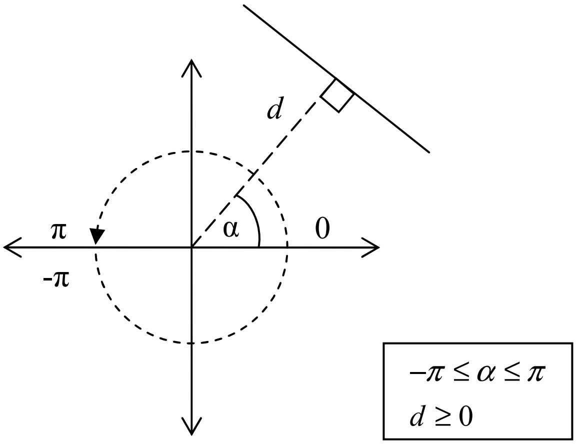


Figure 5-1 Line Parameters

5.3.2.2. Line Segments

Although a line described by (5.1) is infinite, in reality only segments of the line actually exist. As illustrated in Figure 5-2, there may be many line segments observed for a single line observation. A wall of a corridor broken up by doorways could be an example of a single line observation with multiple segments. Broken lines may also occur due to obstructions by obstacles. For the purposes of this chapter, all individual line segments associated with the observation are recorded using the coordinates of the endpoints in an array. This array is only required for the planning and will not be used for the estimation.

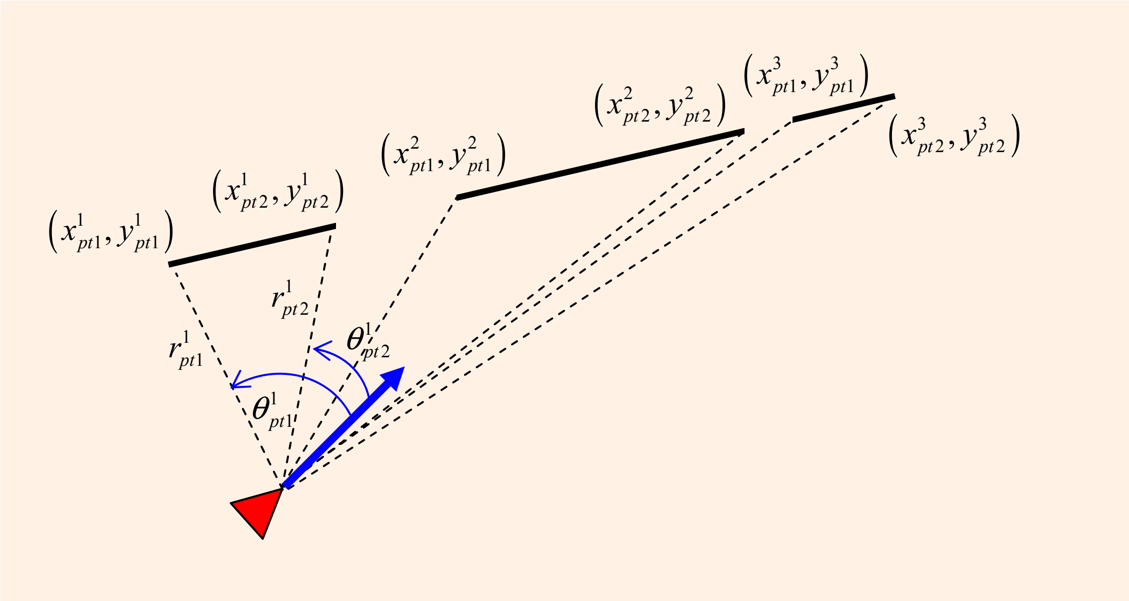

$\left(x_{pt1}^1, y_{pt1}^1\right)$
$\left(x_{pt2}^1, y_{pt2}^1\right)$
$\left(x_{pt1}^2, y_{pt1}^2\right)$
$\left(x_{pt2}^2, y_{pt2}^2\right)$
$\left(x_{pt1}^3, y_{pt1}^3\right)$
$\left(x_{pt2}^3, y_{pt2}^3\right)$
r_{pt2}^1
r_{pt1}^1
θ_{pt2}^1
θ_{pt1}^1

Figure 5-2 Multiple Line Segments for Single Line Observation

The coordinates of the endpoints, x_{pt}^{seg} and y_{pt}^{seg}, of a line segment are determined using the bearing and range measurements to each endpoint, i.e.

$$\begin{aligned} \varphi_{pt1}^{seg} &= \theta_{pt1}^{seg} + \phi_i \\ x_{pt1}^{seg} &= r_{pt1}^{seg} \cos\left(\varphi_{pt1}^{seg}\right) + x_i \\ y_{pt1}^{seg} &= r_{pt1}^{seg} \sin\left(\varphi_{pt1}^{seg}\right) + y_i \\ \varphi_{pt2}^{seg} &= \theta_{pt2}^{seg} + \phi_i \\ x_{pt2}^{seg} &= r_{pt2}^{seg} \cos\left(\varphi_{pt2}^{seg}\right) + x_i \\ y_{pt2}^{seg} &= r_{pt2}^{seg} \sin\left(\varphi_{pt2}^{seg}\right) + y_i \end{aligned} \tag{5.2}$$

where r_{pt1}^{seg} and r_{pt2}^{seg}, and θ_{pt1}^{seg} and θ_{pt2}^{seg} are the ranges and bearings to the endpoints of the line segments respectively, the superindex *seg* represents the line segment number from the current observation belonging to the infinite line described by *α* and *d*, the subindex *pt* represents the endpoint that is associated with the line segment *seg* and (x_i, y_i, ϕ_i) is the robot pose.

The properties of the line segments are stored in an array $\mathbf{\Omega}_q$ where q is the line segment index for each of the line segments in the map and j is the feature number of the associated infinite line, i.e.

$$\mathbf{\Omega}_q = \begin{bmatrix} j & x_{pt1}^{seg} & y_{pt1}^{seg} & x_{pt2}^{seg} & y_{pt2}^{seg} \end{bmatrix}. \tag{5.3}$$

The observed line segments are checked for overlap after data association is performed. If line segments belonging to the same feature overlap then they are merged to a single line segment. The newly observed line segment may overlap between more than one existing line segments, in which case all the overlapping line segments are merged with the new segment. Otherwise if there is no overlapping, the new line segment is then added to the array of line segments.

When α_j^{global} and d_j^{global} change during the update of the estimation process, the line segment parameters are adjusted to comply with the estimated parameters of the line by computing

$$\begin{aligned} x_{pt,i}^{q} &= x_{pt,i-1}^{q} + \left(d_j^{global} - x_{pt,i-1}^{q}\cos\left(\alpha_j^{global}\right) - y_{pt,i-1}^{q}\sin\left(\alpha_j^{global}\right)\right)\cos\left(\alpha_j^{global}\right) \\ y_{pt,i}^{q} &= y_{pt,i-1}^{q} + \left(d_j^{global} - x_{pt,i-1}^{q}\cos\left(\alpha_j^{global}\right) - y_{pt,i-1}^{q}\sin\left(\alpha_j^{global}\right)\right)\sin\left(\alpha_j^{global}\right) \end{aligned}. \tag{5.4}$$

5.3.3. Observation Model

Lines are extracted from each 2-D laser scan consisting of range and bearing measurements, using the line extraction strategy described in Alempijevic (2004). This algorithm provides parameters of a line given by α_k^{local} and d_k^{local}, where the reference origin is the robot position. These line parameters form observation $\mathbf{z}_k$. Let the general observation model be

$$\mathbf{z}_k = h_k(\mathbf{x}_i, \boldsymbol{\lambda}_j) + \nu_k, \tag{5.5}$$

where h is the relationship between α_k^{local} and d_k^{local} to α_k^{global} and d_k^{global}, and $v_k \sim N(0,\mathbf{\Sigma}_k)$ is the zero mean Gaussian noise with covariance $\mathbf{\Sigma}_k$. The detailed formula for h is given in the following.

5.3.3.1. Observation Function

Two sets of equations apply for the two different scenarios depending on the locations of the robot, line and origin. The following subsections describe the transformation of α_k^{local} and d_k^{local} with a reference for an arbitrary robot pose (x, y, ϕ) to α_k^{global} and d_k^{global} with a reference at the map origin for the two scenarios.

***Case* 1. Line is not between the robot and origin**

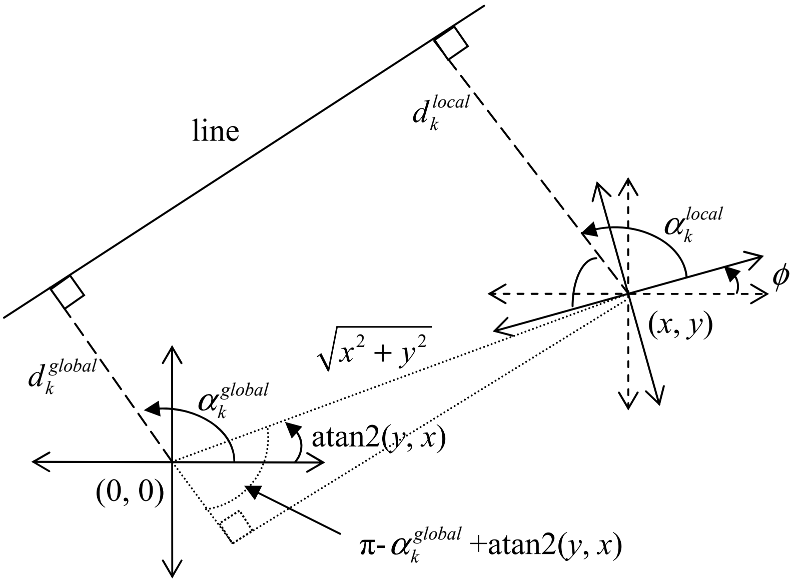


Figure 5-3 Parameters for a Line when the Origin and the Robot are both on One Side of the Line

If the line **<u>is not</u>** between the origin and the robot then the equation, derived from Figure 5-3, is:

$$\begin{aligned}\boldsymbol{\lambda} &= g_k^{case1}(\mathbf{x}, \mathbf{z}_k) \\ &= \begin{bmatrix}\alpha_k^{global} \\ d_k^{global}\end{bmatrix} = \begin{bmatrix}\alpha_k^{local} + \phi \\ d_k^{local} + \sqrt{x^2+y^2}\cos(\alpha^{local} + \phi \ - \tan^{-1}(y,x))\end{bmatrix}.\end{aligned} \quad (5.6)$$

This can alternatively be written in the following form, which can be found in (Garulli et al. 2005),

$$\begin{aligned}\boldsymbol{\lambda} &= g_k^{case1}(\mathbf{x}, \mathbf{z}_k) \\ &= \begin{bmatrix} \alpha_k^{global} \\ d_k^{global} \end{bmatrix} = \begin{bmatrix} \alpha_k^{local} + \phi \\ d_k^{local} + x\cos(\alpha_k^{local} + \phi\ ) + y\sin(\alpha_k^{local} + \phi\ ) \end{bmatrix}.\end{aligned} \tag{5.7}$$

The observation model for case 1 is the following.

$$\begin{aligned}\mathbf{z}_k &= \begin{bmatrix} \alpha_k^{local} \\ d_k^{local} \end{bmatrix} = h_k^{case1}(\mathbf{x}, \boldsymbol{\lambda}) + v_k \\ &= \begin{bmatrix} \alpha_k^{global} - \phi \\ d_k^{global} - x\cos(\alpha_k^{global}) - y\sin(\alpha_k^{global}) \end{bmatrix} + v_k.\end{aligned} \tag{5.8}$$

***Case* 2. Line is between the robot and origin**

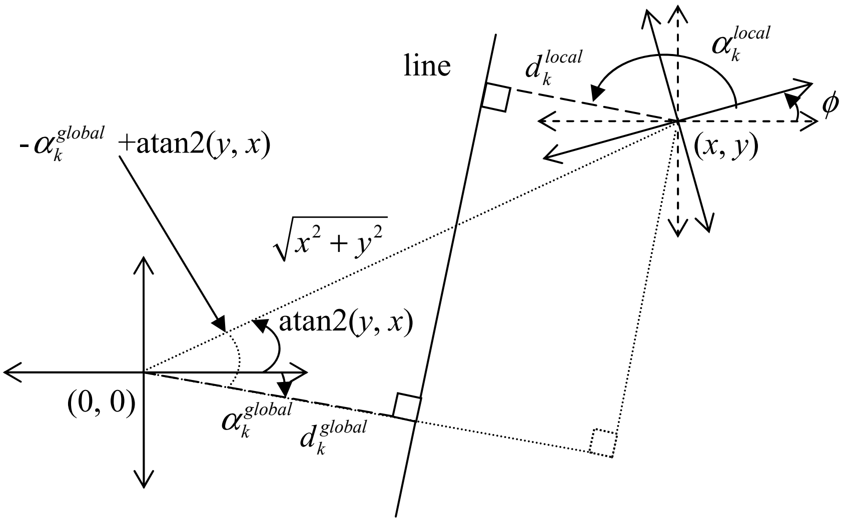


Figure 5-4 Parameters for a Line when the Line Lies Between the Origin and the Robot

If the line **is** between the origin and the robot then the equation, derived from Figure 5-4, is:

$$\begin{aligned}\boldsymbol{\lambda} &= g_k^{case2}(\mathbf{x}, \mathbf{z}_k) \\ &= \begin{bmatrix} \alpha_k^{global} \\ d_k^{global} \end{bmatrix} = \begin{bmatrix} \alpha_k^{local} + \phi\ + \pi \\ -d_k^{local} + \sqrt{x^2+y^2}\cos(\alpha_k^{local} + \phi\ + \pi - \tan^{-1}(y, x)) \end{bmatrix}.\end{aligned} \tag{5.9}$$

This can alternatively be written in the following form, which can also be found in (Garulli et al. 2005),

$$\begin{aligned}\boldsymbol{\lambda} &= g_k^{case2}(\mathbf{x},\mathbf{z}_k)\\ &= \begin{bmatrix}\alpha_k^{global}\\ d_k^{global}\end{bmatrix} = \begin{bmatrix}\alpha_k^{local}+\phi+\pi\\ -d_k^{local}+x\cos(\alpha_k^{local}+\phi+\pi)+y\sin(\alpha_k^{local}+\phi+\pi)\end{bmatrix}.\end{aligned} \tag{5.10}$$

The observation model for case 2 is then

$$\begin{aligned}\mathbf{z}_k &= \begin{bmatrix}\alpha_k^{local}\\ d_k^{local}\end{bmatrix} = h_k^{case2}(\mathbf{x},\boldsymbol{\lambda})+v_k\\ &= \begin{bmatrix}\alpha_k^{global}-\phi+\pi\\ -d_k^{global}+x\cos(\alpha_k^{global})+y\sin(\alpha_k^{global})\end{bmatrix}+v_k.\end{aligned} \tag{5.11}$$

With the two models, there may be the instance where α switches from $-\pi$ and π. In this event, the observation may not be associated correctly and a new line feature may be created. This issue however, may result in redundant features but does not cause any problems with the filter.

5.3.3.2. Covariance of Observation Noise

The covariance of the observation noise, $\boldsymbol{\Sigma}_k$, may be calculated using knowledge of the distribution of the sensor noise contained in each laser measurement that constitute the line observation.

The laser scan consists of range measurements up to 8m with bearings from $-\pi/2 \le \theta \le \pi/2$ rad at every 0.5 degrees which results in 361 measurements. The standard deviation of the range noise for these measurements is approximately $\sigma_r = 0.02$m. For each line detected, $\mathbf{z}_k$, the covariance, $\boldsymbol{\Sigma}_k$, of the observation is calculated from these range measurements. It is assumed that there is no noise in the bearing measurement. The set of range measurements for a single observation and their associated variances are $\mathbf{z}_{\text{ranges}}$ and $\mathbf{R}$ respectively and are defined as

$$\mathbf{z}_{ranges} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_\eta \end{bmatrix}, \quad \mathbf{R} = diag[\sigma_{r_1}^2 \quad \sigma_{r_2}^2 \quad \cdots \quad \sigma_{r_\eta}^2] \tag{5.12}$$

where η is the number of range measurements associated with the line measurement.

Using a least squares method, $\mathbf{\Sigma}_k$ is calculated in the following way:

The measured state is given in the form

$$X_{mea} = \begin{bmatrix} \alpha_k^{local} \\ d_k^{local} \end{bmatrix} = \mathbf{z}_k \, . \tag{5.13}$$

For a given bearing θ_ℓ measurement and the line equation (5.1), the observation model for the range measurement r_ℓ can be formulated as

$$r_\ell = h_{mea}(X_{mea}) = \frac{d_k^{local}}{\cos(\alpha_k^{local} - \theta_\ell)} + w_\ell \ , \tag{5.14}$$

where w_ℓ is the range noise with variance σ_r^2.

The whole observation model can be expressed as

$$\begin{aligned} \mathbf{z}_{ranges} &= \mathbf{H}_{ranges}(X_{mea}) + w_{ranges} \\ &= \begin{bmatrix} \dfrac{d_k^{local}}{\cos(\alpha_k^{local} - \theta_1)} \\ \dfrac{d_k^{local}}{\cos(\alpha_k^{local} - \theta_2)} \\ \vdots \\ \dfrac{d_k^{local}}{\cos(\alpha_k^{local} - \theta_\eta)} \end{bmatrix} + w_{ranges} \end{aligned} \tag{5.15}$$

where w_{ranges} is the measurement noise whose covariance matrix is **R**.

Using the equations from the information filter, the information matrix, $\mathbf{I}_{mea}$, can be calculated by

$$\mathbf{I}_{mea} = \nabla\mathbf{H}_{ranges}^{\mathrm{T}}\ \mathbf{R}^{-1}\nabla\mathbf{H}_{ranges}\,, \tag{5.16}$$

where $\nabla\mathbf{H}_{ranges}$ is the Jacobian of $\mathbf{H}_{ranges}$ w.r.t. state X_{mea}, i.e.

$$\nabla\mathbf{H}_{ranges} = \left.\frac{\partial\mathbf{H}_{ranges}(X_{mea})}{\partial X_{mea}}\right|_{(X_{mea})} \tag{5.17}$$

The final covariance of α and d can be obtained by the inverse of the information matrix, i.e.

$$\mathbf{\Sigma}_k = \mathbf{I}_{mea}^{-1}\,. \tag{5.18}$$

5.3.4. Single Step Process Model

For a single step, the discrete-time process model is defined by:

$$\begin{aligned}\mathbf{x}_i &= \begin{bmatrix} x_i \\ y_i \\ \phi_i \end{bmatrix} = f_i\left(\mathbf{x}_{i-1}, \mathbf{u}_i, w_i\right) = \begin{bmatrix} x_{i-1} + v_i \Delta t \cos\left(\phi_{i-1} + \omega_i \Delta t\right) \\ y_{i-1} + v_i \Delta t \sin\left(\phi_{i-1} + \omega_i \Delta t\right) \\ \phi_{i-1} + \omega_i \Delta t \end{bmatrix} \\ w_i &\sim N\left(0, \mathbf{P}_w\right)\end{aligned} \tag{5.19}$$

where the time elapsed between steps, measured in seconds, is Δt. The noise, w_i, come through the errors in velocity, v_i, and turn-rate, ω_i, i.e. $v_i = v_{\text{true}} + v_{\text{noise}}$ and $\omega_i = \omega_{\text{true}} + \omega_{\text{noise}}$. The covariance of the control noise is described by

$$\mathbf{P}_w = \begin{bmatrix} {\sigma_v}^2 & 0 \\ 0 & {\sigma_\omega}^2 \end{bmatrix} \tag{5.20}$$

where σ_v is the standard deviation of the velocity noise and σ_ω is the standard deviation of the turn-rate noise.

To accommodate process noise which has not been modelled, a small stabilising noise is included in the prediction with covariance as

$$\mathbf{Q} = \begin{bmatrix} 1e^{-6} & 0 & 0 \\ 0 & 1e^{-6} & 0 \\ 0 & 0 & 1e^{-6} \end{bmatrix} \Delta t\,. \tag{5.21}$$

This noise is added to the covariance of the robot pose at each step. However this noise should only be added when the robot is moving, because if the robot is stopped and no features are available, the covariance will grow without limits.

5.4. EKF-SLAM for Line Features

The robot pose and line features may be estimated using the EKF. However, for long experiments, the EKF has been shown to provide inconsistent estimates. As a result, the EKF is found to be unsuitable as the underlying estimator for performing SLAM unless additional shape constraints are enforced. On the other hand, an EKF can efficiently provide consistent estimates when the number of update steps is small. An EKF is therefore adequate for the purposes of multi-step prediction in the trajectory planning. This section describes the EKF algorithms as adapted for line-feature SLAM.

5.4.1. State vector

In an EKF, the state vector contains the current robot pose $\mathbf{x}_i$ and the array of line features L_i at update time step i, i.e.

$$\zeta_i = \begin{bmatrix} \mathbf{x}_i \\ L_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \phi_i \\ \alpha_1^{global} \\ d_1^{global} \\ \vdots \\ \alpha_J^{global} \\ d_J^{global} \end{bmatrix}. \tag{5.22}$$

5.4.2. Prediction

For the robot pose prediction the equations for a single step prediction (5.19) is used. As the features are stationary (i.e. $L_{i+1} = L_i$), state prediction is given by

$$\begin{aligned} \hat{\zeta}_i^- &= p_i\left(\hat{\zeta}_{i-1}, \mathbf{u}_i\right) \\ &= \begin{bmatrix} \hat{\mathbf{x}}_i \\ \hat{L}_i \end{bmatrix} = \begin{bmatrix} f\left(\hat{\mathbf{x}}_{i-1}, \mathbf{u}_i\right) \\ \hat{L}_{i-1} \end{bmatrix}. \end{aligned} \tag{5.23}$$

To update the covariance of the predicted state, first the Jacobian of p_i w.r.t. the state ζ_{i-1} is determined,

$$\mathbf{\Gamma}_\zeta = \left.\frac{\partial p_i(\zeta_{i-1}, \mathbf{u}_i)}{\partial \zeta_{i-1}}\right|_{(\hat{\zeta}_{i-1}, \mathbf{u}_i)}. \tag{5.24}$$

Next, the Jacobian of p_i w.r.t. the control $\mathbf{u}_i$ is computed as

$$\mathbf{Y}_\mathbf{u} = \left.\frac{\partial p_i(\zeta_{i-1}, \mathbf{u}_i)}{\partial \mathbf{u}_i}\right|_{(\hat{\zeta}_{i-1}, \hat{\mathbf{u}}_i)}. \tag{5.25}$$

Then from the standard EKF equations, the predicted covariance $\mathbf{P}_i^-$ is determined by

$$\begin{aligned}\widehat{\mathbf{P}}_i &= \mathbf{\Gamma}_\zeta \mathbf{P}_{i-1} \mathbf{\Gamma}_\zeta^{\mathrm{T}} + \mathbf{Y}_{\mathbf{u}} \mathbf{P}_w \mathbf{Y}_{\mathbf{u}}^{\mathrm{T}} \\ \mathbf{P}_i^- &= \widehat{\mathbf{P}}_i + \mathbf{Q}\end{aligned} \tag{5.26}$$

where **Q** is the covariance matrix of the stability noise defined in (5.21).

5.4.3. Data Association

The nearest neighbour method is used here for data association. Let $\boldsymbol{\zeta}_i^-$ and $\mathbf{P}_i^-$ be the current predicted state and covariance of the system, including the current robot pose and the locations of the line features.

All the line features in the state vector are converted to a local reference frame using (5.8) or (5.11), depending on the location of the line. Then for each measurement, *k*, acquired at time *i*, the innovation $\boldsymbol{\mu}_{jk}$ is calculated by

$$\boldsymbol{\mu}_{jk} = \begin{bmatrix} \alpha_k^{local} - \alpha_j^{local} \\ d_k^{local} - d_j^{local} \end{bmatrix} \tag{5.27}$$

where α_j^{local} and d_j^{local} are the line parameters of $\hat{\boldsymbol{\lambda}}^j$ converted to local coordinates.

Using $\boldsymbol{\Sigma}_k$ calculated from the process in Section 5.3.3.2, the observation covariance $\mathbf{S}_k$ can be obtained by

$$\mathbf{S}_k = \nabla h_k \mathbf{P}_i^- \nabla h_k^{\mathrm{T}} + \boldsymbol{\Sigma}_k \tag{5.28}$$

where the appropriate Jacobian, ∇h_k, can be calculated using Equations (5.8) or (5.11) depending on the location of the line.

The Mahanalobis distance, β_{jk}, is

$$\beta_{jk} = \boldsymbol{\mu}_{jk}{}^T \mathbf{S}_{jk}^{-1} \boldsymbol{\mu}_{jk}\,. \tag{5.29}$$

Many measurements are required for a line to be fitted, which minimises the possibility for false readings. Detected lines that cannot be associated are assumed to be new features. Accordingly, using the Mahanalobis distance, β_{jk}, and a chi-squared χ^2 gate of 99.99% confidence level with 2 degrees of freedom (DOF), the observations are categorised as either associated or as new features. The associations of observations k to features j are mapped in $\mathbf{\Psi}$ along with observations to new features, which are given an index of 0.

$$j_*^k = \arg\min_j \beta_{jk}$$
$$\mathbf{\Psi}_{ik} = \begin{cases} j_*^k & \beta_{j_*^k k} < \chi^2\left(99.99\%, 2\,\mathrm{DOF}\right) \\ 0 & \beta_{j_*^k k} \geq \chi^2\left(99.99\%, 2\,\mathrm{DOF}\right) \end{cases} \tag{5.30}$$

5.4.4. Initialising New Line Features

When a new feature is detected, i.e. $\mathbf{\Psi}_{ik}$ = 0, the measurements $\mathbf{z}_k$ and the predicted robot pose $\mathbf{x}_i$ are used to initialise a new feature in the state vector,

$$\boldsymbol{\lambda}_{J+1} = \begin{bmatrix} \alpha_k^{global} \\ d_k^{global} \end{bmatrix}, \tag{5.31}$$

where $\boldsymbol{\lambda}_{J+1}$ is calculated by either equation (5.7) or (5.10) depending on the location of the line. This feature is then added to the state vector, i.e.

$$\boldsymbol{\zeta}_i^- \leftarrow \begin{bmatrix} \boldsymbol{\zeta}_i^- \\ \boldsymbol{\lambda}_{J+1} \end{bmatrix}. \tag{5.32}$$

Then taking $\mathbf{\Sigma}_k$ from Sec. 5.3.3.2, the new covariance of estimation can be calculated by

$$\mathbf{P}_i^- \leftarrow \begin{bmatrix} I & 0 \\ 0 & \nabla g \end{bmatrix} \begin{bmatrix} \mathbf{P}_i^- & 0 \\ 0 & \mathbf{\Sigma}_k \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \nabla g^T \end{bmatrix}, \tag{5.33}$$

where ∇g is the Jacobian of equation (5.7) or (5.10) depending on the location of the new line feature. The above process is repeated for each new line feature detected.

5.4.5. Update of Associated Features

The features in the state vector are updated using the associated observations, i.e. $\mathbf{\Psi}_{ik} = j$. Using the innovations $\boldsymbol{\mu}_{jk}$ is calculated in (5.27), the innovations for the n associated measurements are then assembled forming

$$\boldsymbol{\mu} = \left[\boldsymbol{\mu}_{jk1} \cdots \boldsymbol{\mu}_{jkn}\right]^{\mathrm{T}}, \tag{5.34}$$

and depending on the location of the line, the appropriate Jacobian, ∇h, of the observation model can be calculated using either (5.8) or (5.11) for the n associated measurements thus yielding

$$\nabla h = \left[\nabla h_{jk1} \cdots \nabla h_{jkn}\right]^{\mathrm{T}}. \tag{5.35}$$

Then using the same $\mathbf{\Sigma}_k$ used in (5.28), the observation noise covariance for each associated measurement can be assembled and expressed as

$$\mathbf{\Sigma} = diag\left(\left[\mathbf{\Sigma}_{k1} \cdots \mathbf{\Sigma}_{kn}\right]\right). \tag{5.36}$$

Following the EKF equations, the Kalman gain is calculated and the predicted $\boldsymbol{\zeta}_i^-$ and $\mathbf{P}_i^-$ are updated accordingly, i.e.

$$\begin{aligned}
\mathbf{S} &= \nabla h \mathbf{P}_i^- \nabla h^{\mathrm{T}} + \mathbf{\Sigma} \\
\mathbf{K} &= \mathbf{P}_i^- \nabla h^{\mathrm{T}} \mathbf{S}^{-1} \\
\boldsymbol{\zeta}_i &= \boldsymbol{\zeta}_i^- + \mathbf{K}\boldsymbol{\mu} \\
\mathbf{P}_i &= \mathbf{P}_i^- - \mathbf{K}\mathbf{S}\mathbf{K}^{\mathrm{T}}
\end{aligned} \tag{5.37}$$

5.4.6. Estimation Result of EKF

In attempting to apply the EKF to line feature SLAM, it was found that the estimates soon became inconsistent in many simulations. An example simulation illustrates the problem in Figures 5-5 to 5-7. Here, the robot is set to run for 1000 steps with observations taken every 50 steps and robot pose predictions computed at every step. The robot is kept stationary for the first two updates and then moves continuously thereafter. A total of 20 updates are computed and data association is assumed. Figure 5-5 shows the robot pose estimate at every step and it is clear that the 2σ bound has been exceeded in the x and ϕ estimates. A clearer picture can be seen when only the pose estimates at the update steps are shown in Figure 5-6. The 2σ bound of the innovation covariance is also exceeded by the observation innovations (see Figure 5-7). This is a typical case in that the EKF fails after a short time.

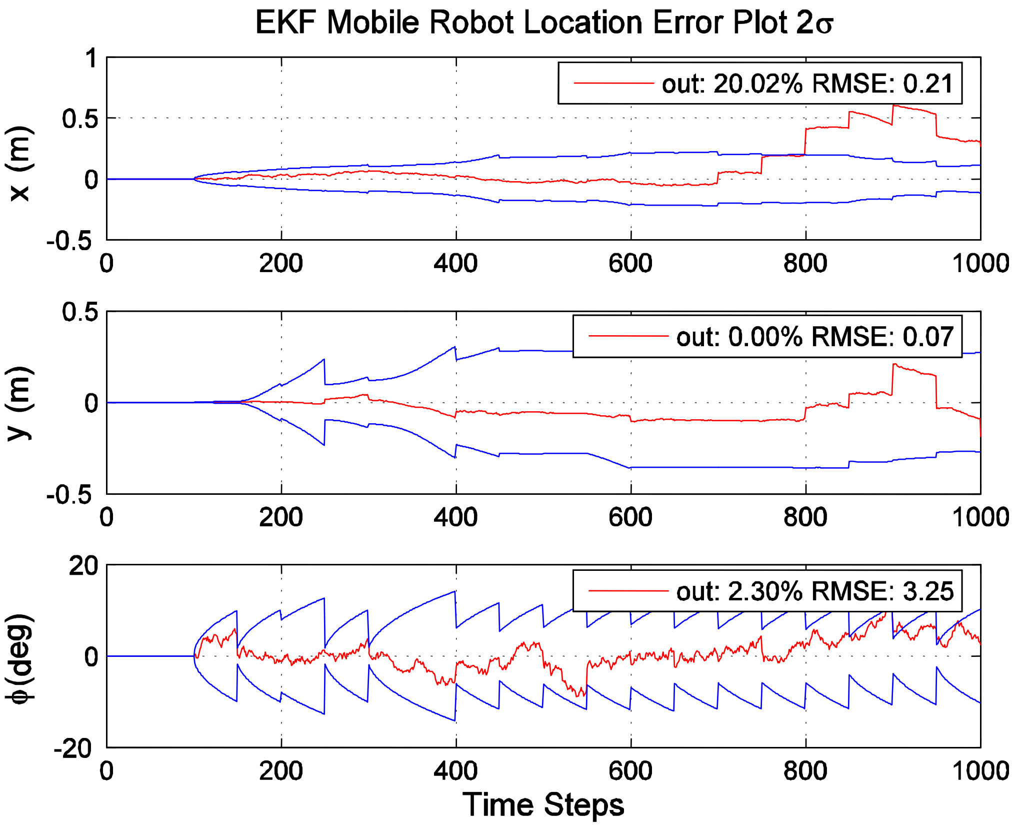


Figure 5-5 EKF Pose Estimate at Every Time Step

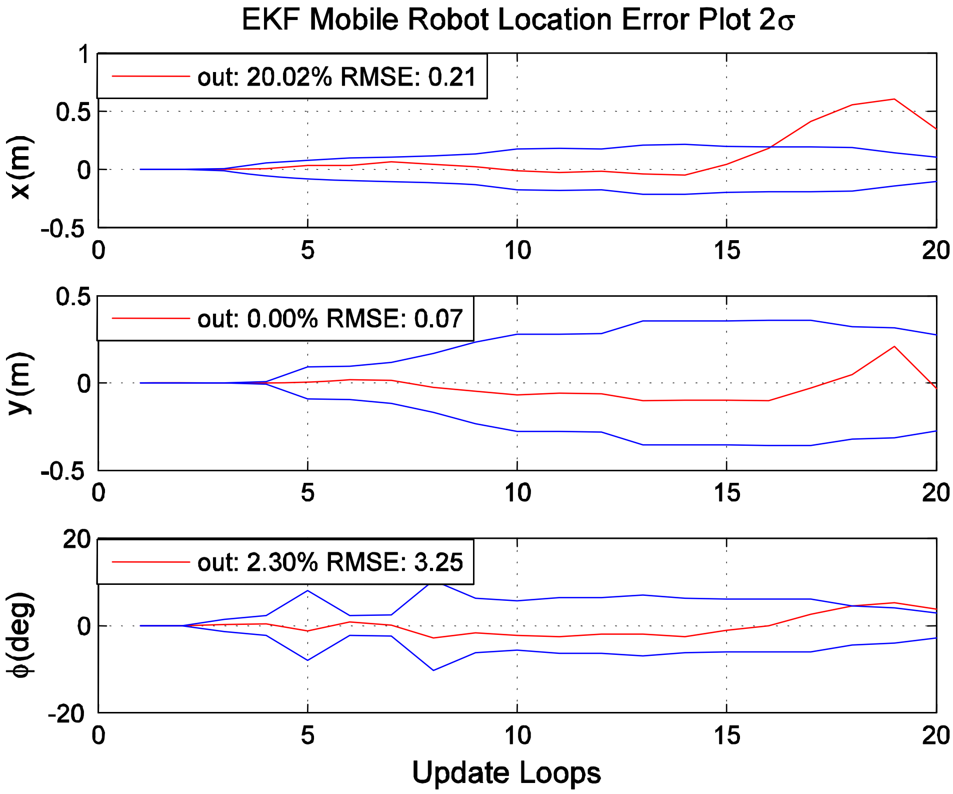


Figure 5-6 EKF Pose Estimate at Observation Update Steps

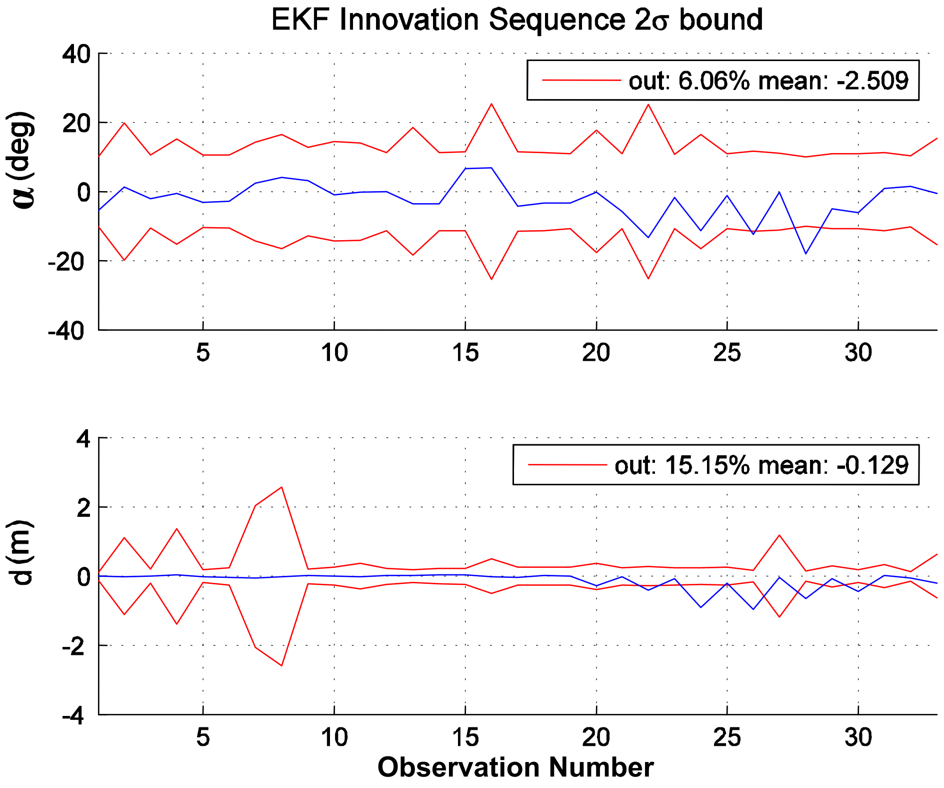


Figure 5-7 EKF Observation Innovation

Given that the EKF approach is not succeeding with line features in the case, an alternate algorithm suitable for mapping an arbitrary environment with line features is required to be explored. It is possible to apply constraints based on knowledge of the environment to improve the consistency of EKF (Rodriguez-Losada et al. 2006a), however it is decided that another algorithm is warranted. Heuristics used in the trajectory planning (Section 5.6.2.4) requires that previous robot poses are stored along with the associations of observations made from these poses to line features in the map. Incremental SAM (iSAM) provides a mechanism to maintain an estimate of all the robot poses, thus providing an accurate history of robot poses of which may be used in the trajectory planning strategy.

5.5. Incremental SAM (iSAM)

Using the algorithm from Dellaert and Kaess (2006), incremental Smoothing and Mapping (iSAM) is implemented and proves to perform much better in terms of maintaining consistent estimates. This approach allows the line representation already established for the EKF implementation to be utilised. The following subsections describe the iSAM algorithm as adapted to line features, including the modifications made to improve the efficiency and minimise errors in the initial pose estimate.

5.5.1. Relative Pose between Two Consecutive Observation Points

One key difference between EKF-SLAM and iSAM lies with the maintenance of all robot poses in the state vector. To minimise computation and to avoid recording all the positions between updates, it is necessary to compute the robot pose between consecutive observation points, such that only the poses where observations are made are included in the state vector. This is achieved by computing the relative pose, $\mathbf{x}_{rel}$, between the observation steps using the function γ, i.e.

$$\mathbf{x}_{rel} = \gamma(\mathbf{x}_{i-1}, \mathbf{x}_i) + q_i \tag{5.38}$$

where the noise term, q_i, is defined as a Gaussian with zero mean and covariance $\mathbf{\Lambda}_i$, i.e.

$$q_i \sim N(0, \mathbf{\Lambda}_i) \tag{5.39}$$

and the covariance matrix, $\mathbf{\Lambda}_i$, of control noise for multiple steps is calculated using the following process:

$$\begin{aligned}
&\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}} && (5.40)\\
&\mathbf{P}_{temp}^{0} = diag\left(\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}\right)\\
&\mathbf{x}_n = f_n(\mathbf{x}_{n-1}, \mathbf{u}_n)\\
&\mathbf{\Gamma} = \mathbf{\Gamma}_n^{n-1} = \left.\frac{\partial f_n\left(\mathbf{x}_{n-1}, \mathbf{u}_n\right)}{\partial \mathbf{x}_{n-1}}\right|_{\left(\hat{\mathbf{x}}_{n-1}, \mathbf{u}_n\right)}\\
&\mathbf{Y} = \mathbf{Y}_n^{n-1} = \left.\frac{\partial f_n\left(\mathbf{x}_{n-1}, \mathbf{u}_n\right)}{\partial \mathbf{u}_n}\right|_{\left(\hat{\mathbf{x}}_{n-1}, \mathbf{u}_n\right)}\\
&\mathbf{P}_{temp}^{-} = \mathbf{\Gamma}\mathbf{P}_{temp}^{n-1}\mathbf{\Gamma}^{\mathrm{T}} + \mathbf{Y}\mathbf{P}_w\mathbf{Y}^{\mathrm{T}}\\
&\mathbf{P}_{temp}^{n} = \mathbf{P}_{temp}^{-} + \mathbf{Q}\\
&\mathbf{x}_{rel} = \mathbf{x}_{N_{steps}}\\
&\mathbf{\Lambda}_i = \mathbf{P}_{temp}^{N_{steps}}
\end{aligned}$$

for $n = 1 \ldots N_{steps}$, where $\mathbf{Q}$ is the stabilising noise defined in (5.21).

Following this, the predicted state can be computed as

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ \phi_i \end{bmatrix} = \begin{bmatrix} x_{i-1} + x_{rel}\cos(\phi_{i-1}) - y_{rel}\sin(\phi_{i-1}) \\ y_{i-1} + x_{rel}\sin(\phi_{i-1}) + y_{rel}\cos(\phi_{i-1}) \\ \phi_{i-1} + \phi_{rel} \end{bmatrix} \qquad (5.41)$$

such that the relative pose between two consecutive observation points is given by

$$\mathbf{x}_{rel} = \gamma(\mathbf{x}_{i-1}, \mathbf{x}_i) = \begin{bmatrix} x_{rel} \\ y_{rel} \\ \phi_{rel} \end{bmatrix} = \begin{bmatrix} (x_i - x_{i-1})\cos(\phi_{i-1}) + (y_i - y_{i-1})\sin(\phi_{i-1}) \\ -(x_i - x_{i-1})\sin(\phi_{i-1}) + (y_i - y_{i-1})\cos(\phi_{i-1}) \\ \phi_i - \phi_{i-1} \end{bmatrix}. \qquad (5.42)$$

5.5.2. SLAM as a Least Squares Problem

The entire state to be estimated in iSAM includes all robot poses from time *m*=1…*i*, combined with all the landmarks from *j*=1…*J* and is defined by

$$\mathbf{\Theta} \triangleq (X, L) \rightarrow \hat{\mathbf{\Theta}} = \begin{bmatrix} \hat{\mathbf{x}}_1 & \dots & \hat{\mathbf{x}}_i & \hat{\boldsymbol{\lambda}}_1 & \dots & \hat{\boldsymbol{\lambda}}_J \end{bmatrix}^{\mathrm{T}}. \quad (5.43)$$

The aim is to estimate the position of all the poses and features. This is achieved by solving the least squares problem as follows

$$\mathbf{\Theta}^* \triangleq \arg\min_{\mathbf{\Theta}} \left\{ \sum_{m=1}^{i} \left\| \gamma_m(\mathbf{x}_{m-1}, \mathbf{x}_m) - \mathbf{x}_m^{rel} \right\|_{\mathbf{\Lambda}_m}^2 + \sum_{k=1}^{K} \left\| h_k(\mathbf{x}_{mk}, \boldsymbol{\lambda}_{jk}) - \mathbf{z}_k \right\|_{\mathbf{\Sigma}_k}^2 \right\}. \quad (5.44)$$

The first term in the equation above is the process innovation, which is expressed as

$$\gamma_i(\mathbf{x}_{i-1}, \mathbf{x}_i) - \mathbf{x}_i^{rel} \approx \left\{ \gamma_i(\mathbf{x}_{i-1}, \mathbf{x}_i) + \mathbf{F}_i^{i-1} \delta\mathbf{x}_{i-1} + \mathbf{G}_i^i \delta\mathbf{x}_i \right\} - \mathbf{x}_i^{rel} = \left\{ \mathbf{F}_i^{i-1} \delta\mathbf{x}_{i-1} + \mathbf{G}_i^i \delta\mathbf{x}_i \right\} - \boldsymbol{\varepsilon}_i \quad (5.45)$$

where $\partial\mathbf{x}$ is the difference between the true and estimated states; $\mathbf{F}_i^{i-1}$ and $\mathbf{G}_i^{i-1}$ are the Jacobians of (5.42), such that

$$\mathbf{F}_i^{i-1} = \left. \frac{\partial \gamma_i(\mathbf{x}_{i-1}, \mathbf{x}_i)}{\partial \mathbf{x}_{i-1}} \right|_{(\hat{\mathbf{x}}_{i-1}, \hat{\mathbf{x}}_i)} \quad \text{and} \quad \mathbf{G}_i^{i-1} = \left. \frac{\partial \gamma_i(\mathbf{x}_{i-1}, \mathbf{x}_i)}{\partial \mathbf{x}_i} \right|_{(\hat{\mathbf{x}}_{i-1}, \hat{\mathbf{x}}_i)} \quad (5.46)$$

and the $\boldsymbol{\varepsilon}_i$ is the odometry prediction error defined by

$$\begin{aligned} \boldsymbol{\varepsilon}_i &= \mathbf{x}_i^0 - \gamma_i(\mathbf{x}_{i-1}^0, \mathbf{x}_i) \\ &= \mathbf{x}_i^{rel} - \hat{\mathbf{x}}_i^{rel} \end{aligned} \quad (5.47)$$

where $\mathbf{x}_i^0$ and $\mathbf{x}_{i-1}^0$ are the initial guesses of the robot poses obtained either from the state vector or through prediction.

The second term in (5.44) is the observation innovation and is expressed as

$$h_k(\mathbf{x}_{ik},\boldsymbol{\lambda}_{jk})-\mathbf{z}_k \approx \left\{h_k(\mathbf{x}_{ik}^0,\boldsymbol{\lambda}_{jk}^0)+\mathbf{H}_k^{i_k}\delta\mathbf{x}_{ik}+\mathbf{J}_k^{j_k}\delta\boldsymbol{\lambda}_{ik}\right\}-\mathbf{z}_k=\left\{\mathbf{H}_k^{i_k}\delta\mathbf{x}_{ik}+\mathbf{J}_k^{j_k}\delta\boldsymbol{\lambda}_{ik}\right\}-\boldsymbol{\mu}_k \quad (5.48)$$

where $\boldsymbol{\lambda}_{jk}^0$ is the initial guess of the feature observed and $\mathbf{H}_k^{i_k}$ and $\mathbf{J}_k^{j_k}$ are Jacobians of (5.5), such that

$$\mathbf{H}_k^{i_k}=\left.\frac{\partial h_k(\mathbf{x}_{i_k},\boldsymbol{\lambda}_{j_k})}{\partial \mathbf{x}_{i_k}}\right|_{(\hat{\mathbf{x}}_{ik},\hat{\boldsymbol{\lambda}}_{ik})} \quad \text{and} \quad \mathbf{J}_k^{j_k}=\left.\frac{\partial h_k(\mathbf{x}_{i_k},\boldsymbol{\lambda}_{j_k})}{\partial \boldsymbol{\lambda}_{i_k}}\right|_{(\hat{\mathbf{x}}_{ik},\hat{\boldsymbol{\lambda}}_{ik})} \quad (5.49)$$

and $\boldsymbol{\mu}_k$ is the measurement prediction error defined by

$$\begin{aligned}\boldsymbol{\mu}_k &= \mathbf{z}_k - h_k(\mathbf{x}_{ik}^0,\boldsymbol{\lambda}_{jk}^0)\\ &= \mathbf{z}_k - \boldsymbol{\lambda}_j^{local}\end{aligned} \quad (5.50)$$

The least squares iSAM problem (5.44) is linearised to make it solvable. To update the state $\boldsymbol{\Theta}$, the term δ is to be solved in

$$\delta^* = \arg\min_{\delta}\left\{\sum_{m=1}^{i}\left\|\mathbf{F}_m^{m-1}\delta\mathbf{x}_{m-1}+\mathbf{G}_m^m\delta\mathbf{x}_m-\boldsymbol{\varepsilon}_m\right\|_{\boldsymbol{\Lambda}_m}^2+\sum_{k=1}^{K}\left\|\mathbf{H}_k^i\delta\mathbf{x}_{ik}+\mathbf{J}_k^j\delta\boldsymbol{\lambda}_{jk}-\boldsymbol{\mu}_k\right\|_{\boldsymbol{\Sigma}_k}^2\right\} \quad (5.51)$$

and the noise covariances $\boldsymbol{\Lambda}_m$ and $\boldsymbol{\Sigma}_k$ can be incorporated into the norm using the following property:

$$\|e\|_{\Sigma}^2 \triangleq e^{\mathrm{T}}\Sigma^{-1}e=(\Sigma^{-\mathrm{T}/2}e)^{\mathrm{T}}(\Sigma^{-\mathrm{T}/2}e)=\left\|\Sigma^{-\mathrm{T}/2}e\right\|_2^2. \quad (5.52)$$

By collecting the Jacobian matrices into **A** and the innovations into **b**, the standard least-squares problem can be obtained and solved given the following expression:

$$\delta^* = \arg\min_{\delta}\|\mathbf{A}\delta-\mathbf{b}\|_2^2. \quad (5.53)$$

This process is repeated for each update until the solution has converged, i.e. δ is smaller than a threshold. Appendix A gives further details on the implementation.

5.5.3. SAM vs. Incremental SAM

The key difference in the implementation of SAM and iSAM lies with the process of acquiring the initial guess for the least squares problem. In SAM, all the robot poses and features detected are given an initial estimate through prediction, thus data association needs to be assumed and the error in the robot pose predictions are accumulated. Alternatively, in iSAM the initial estimates for the robot pose are given by the current state and a prediction of only the current pose. This allows the robot pose estimate to be refined at every step giving a more accurate initial estimate for the subsequent poses. Data association is also possible in iSAM as a map is available, i.e. generated from the observations in the previous steps. Clearly, an incremental approach is required to perform Active SLAM. The algorithms described up to this point are applicable to both SAM and iSAM however the data association in the following section can only be used with iSAM.

5.5.4. Data Association

The data association in iSAM is the same as that implemented for the EKF-SLAM. First, the covariances and states in iSAM are modified to that form used in EKF-SLAM, where all the robot poses except the current pose are omitted. The remainder of the algorithm follows that described in Section 5.4.3.

The details for using iSAM estimates for data association are as follows. The current state estimate and associated covariance need to be extracted from the iSAM matrices $\mathbf{\Theta}$ and $\mathbf{A}$. To generate the terms necessary for the data association, the information matrix is first computed by

$$\mathbf{I} = \mathbf{A}^T \mathbf{A}, \tag{5.54}$$

then the covariance $\mathbf{P}$ is simply the inverse of the information matrix, i.e.

$$\mathbf{P} = \mathbf{I}^{-1} \tag{5.55}$$

However, the direct inverse is computationally demanding in Matlab. Cholesky Factorisation for sparse linear equations (Golub and Van Loan, 1996) is used instead where only relevant columns are selected to compute the sub-matrix of the inverse. This method reduces computation.

Once the inverse of the information matrix is obtained, the current state and associated covariance can be extracted as

$$\hat{\zeta}_i = \begin{bmatrix} \hat{\mathbf{x}}_i \\ \hat{L} \end{bmatrix} \leftarrow \begin{bmatrix} \blacksquare \\ \blacksquare \\ \hat{\mathbf{x}}_i \\ \hat{L} \end{bmatrix} \qquad \hat{\mathbf{P}}_i = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_i} & \mathbf{P}_{\mathbf{x}_i L} \\ \mathbf{P}_{L\mathbf{x}_i} & \mathbf{P}_L \end{bmatrix} \leftarrow \begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \mathbf{P}_{\mathbf{x}_i} & \mathbf{P}_{\mathbf{x}_i L} \\ \blacksquare & \blacksquare & \mathbf{P}_{L\mathbf{x}_i} & \mathbf{P}_L \end{bmatrix} \tag{5.56}$$

Using the current state and covariance, the nearest neighbour method described in Section 5.4.3 can be applied for data association.

5.5.5. Estimation Result of iSAM

Line feature SLAM using iSAM remains consistent for long simulation runs. To illustrate the performance iSAM, an example result from the simulation scenario of that described in Section 5.4.6 is shown in Figure 5-8 and Figure 5-9. It can be seen that the pose estimate remains consistent throughout the simulation. In a number of other simulation trials, consisting up to 100000 steps with an update every 50 steps and iSAM still maintains a good estimate of the robot pose and features.

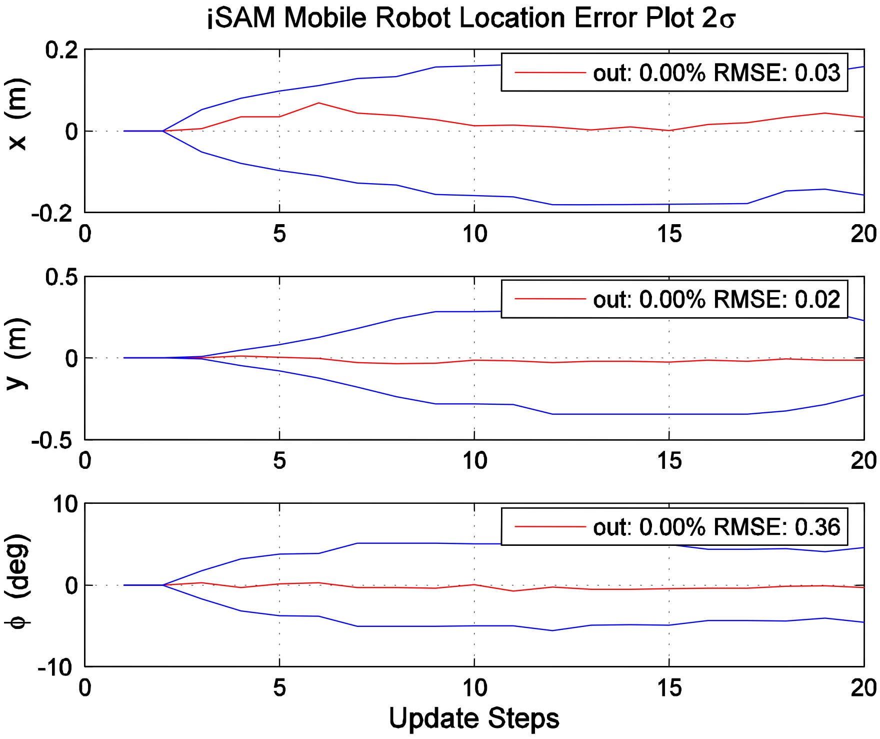


Figure 5-8 iSAM Robot Pose Estimate

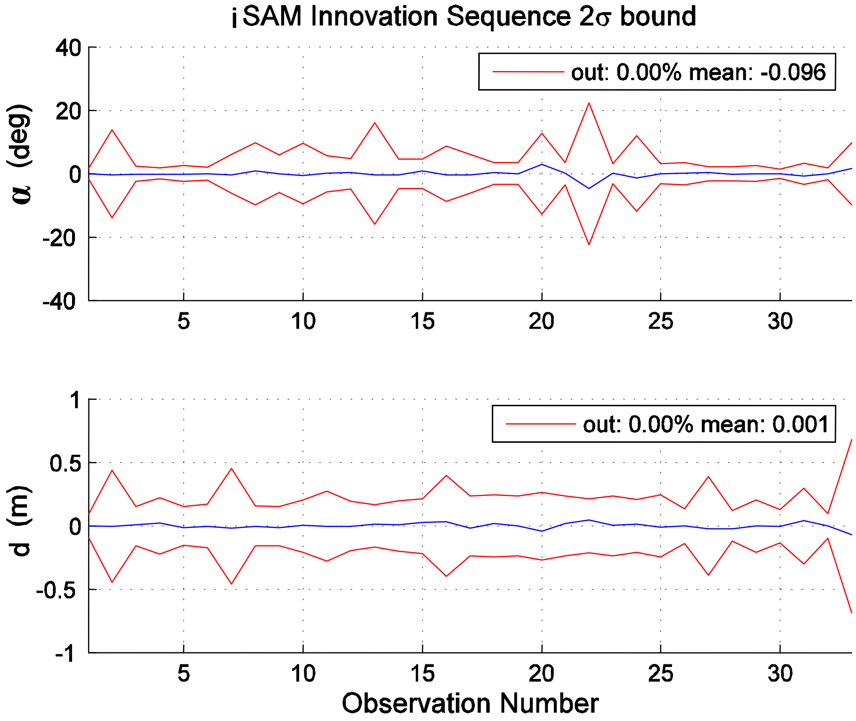


Figure 5-9 iSAM Observation Innovation

Given this demonstrated consistency in the iSAM estimation, it is deemed suitable to be used as the underlying estimator for the trajectory planning algorithms. Other consistent estimation algorithms may be used, however they are not studied further as the focus of the thesis is on trajectory planning.

5.6. Trajectory Planning

The MPC+Attractor strategy, demonstrated to encourage localisation, mapping and exploration (Section 4.5.4), is once again used as a basis for trajectory planning. A number of modifications must be applied to adapt the trajectory planning to structured environments and are described as follows.

5.6.1. Model Predictive Control

The implementation of MPC for line-feature SLAM requires not only the current state estimate to be considered, $\hat{\mathbf{x}}_i$ and L, but also the array of line segments, $\mathbf{\Omega}_q$, for determining the robot trajectory. Although the underlying estimator for SLAM is iSAM, the EKF continues to be used in the MPC strategy as it is more efficient and the estimates remain consistent for short periods required for MPC. Thus the state estimates from iSAM are converted to the form used in EKF-SLAM prior to the planning following the conversion process, as described in Section 5.5.4. The objective function is maintained to be the trace of the covariance matrix as the parameters of a line are measured in meters and radians which are similar in scale.

5.6.1.1. Obstacle Avoidance

Modifications need to be made to the obstacle avoidance developed in Chapter 4. Previously, the predicted pose of the robot was checked to determine if it lies within a certain radius of a point feature, however this is too simplistic for a structured environment. Not all objects present in an environment can be classified as line features and searching through an occupancy grid map requires significant computation. The sensor range is much greater than the planning horizon of the robot, which allows for the current laser scan of the robot to be used for determining whether a path is obstacle-free, which is similar to the idea used in González-Baños (2002).

At each step, the robot's direction of travel is calculated (i.e. bearing of the predicted robot pose to the current robot pose) and the laser measurement with the closest bearing is identified. If the predicted pose for the next step puts the robot beyond a boundary distance then the path is deemed infeasible. The boundary distance is taken as the laser range measurement less the width of the robot. If the predicted robot pose is feasible, then the information gain based on the predicted sensor readings and line observations are computed.

5.6.1.2. Prediction of Information from Line Observations

The line observations need to be predicted to determine the information gain for each control option. By use of the predicted robot pose and the line feature estimates, a prediction of sensor measurements may be made. If the robot is predicted to observe an adequate number of sensor measurements to an estimated line feature then that line is deemed to be observed. Observation noises of range measurements are simulated and all the measurements associated to the line feature estimate are then used to determine the covariance of the predicted line observation.

The feasible control option with the highest information gain based on predicted observations is selected to be executed. The strategy shows a similar characteristic to trajectory planning with point features, with the robot tending to move close to features in an attempt to obtain more information. In a structured environment this results in the robot approaching features and then having to stop and turn to avoid crashing, as seen in Figure 5-10.

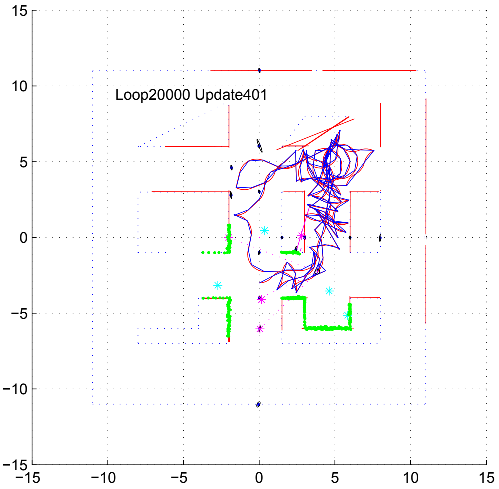


Figure 5-10 MPC and iSAM with Line Features

(red path = true path, blue path = estimated path)

Furthermore, with the robot performing Active SLAM with MPC alone, the resulting path is only around known features as there is no incentive to explore as evident in Figure 5-10. The attractor concept will again be helpful in providing this incentive.

5.6.2. Attractor Strategy for Active SLAM with Line Features

The attractor strategy, developed in Chapter 4 for point feature SLAM for an open environment, is based on heuristics that may not be suitable for line feature SLAM in structured environments. Thus an adaptation of this strategy together with a new set of heuristics is required.

5.6.2.1. Addition of Occupancy Grid Map

In Section 4.5.3.3, the reference point for the attractor was determined based on minimum distance and the attractor was placed at the maximum sensor range in the line of sight to the reference point. In a structured environment there may not be direct line of sight to the attractor when it is placed at the maximum sensor range and the closest exploration point may be behind a wall. To facilitate exploration, localisation and mapping in structured environment, an occupancy grid map is built in conjunction with the SAM line-feature map. The occupancy grid map records information for explored and unexplored areas as well as occupied and obstacle-free cells. It is used to determine frontiers for exploration and traversable areas. This map replaces the uniformly distributed exploration points used in Chapter 4. The heuristics used to select the reference point and the placement of the attractor are also modified to exploit the knowledge contained in the occupancy grid map.

The occupancy grid map is built based on the estimated poses from the iSAM algorithm. Creating a new map using all the accumulated poses in the iSAM state vector at each time step allows production of a map accurate to the current estimate, but is computationally expensive. To save computation, only the final pose in the iSAM estimate is used to create the occupancy grid map. It should be noted that the cells marked from previous time steps are not updated when the pose estimate is updated. This results in an occupancy grid map that is not statistically accurate. However for the trajectory planning strategy, the occupancy map is predominately used to find unexplored frontier regions and the information contained in this map is not used during estimation. Therefore it was found to be adequate for the intention of trajectory planning.

5.6.2.2. Attractor as a Virtual Point Feature

As the attractor has been demonstrated to be effective as a virtual point feature (as described in Section 4.5.2), it is maintained in the same form for line-feature Active SLAM. Defining the attractor to be a line feature requires many heuristics to define parameters such as the angle and distance of the line, the length of the line segment and the covariance of the observation of the line. On the whole, it is not a trivial task and requires more processing. The aim here is to maintain simplicity and minimise computation as the planning needs to be computed between iSAM updates. Thus the attractor is always initialised as a new point feature and placed at the end of the state vector. By maintaining the attractor as a point feature, it is clear which direction the robot is encouraged to move and this direction is determined by the reference point.

5.6.2.3. Reference Point for Exploration

In order to obtain a reference point for the *explore* mode in the state machine (see Section 4.5.3), the frontier points are extracted from the occupancy grid map. Frontier points are cells on the grid map that are unexplored and adjacent to an empty cell. These points are grouped together into regions and the centres of each region are used as potential reference points. A selection amongst these points is made based on the minimum absolute bearing to the robot. This encourages the robot to continue exploring in its current direction and minimises turning.

5.6.2.4. Reference Point for Localisation and Mapping

For the *improve localisation* mode and the *improve map* mode, a feature of interest is selected based on the covariance of the line features. A set of candidate well-defined or poorly-defined features are selected for the respective modes based on a threshold. Then a reference point for the attractor needs to be determined from the candidate set of features. Determining the reference point is not as simple as in point feature SLAM where the coordinates of a well-defined or poorly-defined feature is extracted from the state vector based on minimum distance. A line may have many segments and may not be continuous. In addition, lines may occlude other lines. Therefore, instead of using the position of the feature as the reference point, the robot poses that have previously observed the line feature are used as the potential reference points. These poses can easily be retrieved as all previous data associations are recorded for the iSAM process. Thus a set of robot locations derived from the candidate set of features is used as the group of potential reference points.

Once a group of potential reference points are obtained, the distance transform (see Appendix B for details) is computed for the occupancy grid map. Each cell is given a value based on its traversable distance from the robot. This information is then treated as a lookup table. Using the result of the distance transform, the reference point is selected from the group obtained based on minimum traversable distance. However, it is important to note that this reference point may still be occluded or out of the sensor range.

5.6.2.5. Placement of Attractor

The attractor is placed within the sensor range so as to lead the robot to the selected reference point. Previously (Chapter 4), the environment was not structured and direct line of sight to the attractor was always possible. In structured environments the attractor may be obscured by walls, thus its placement needs to be adapted to the environment. To determine this placement, a cell path is planned using the result of the distance transform and then starting from the reference point, the location of the first cell along the path that is visible to the robot is set as the location of the attractor. Figure 5-11 provides an illustration of this idea.

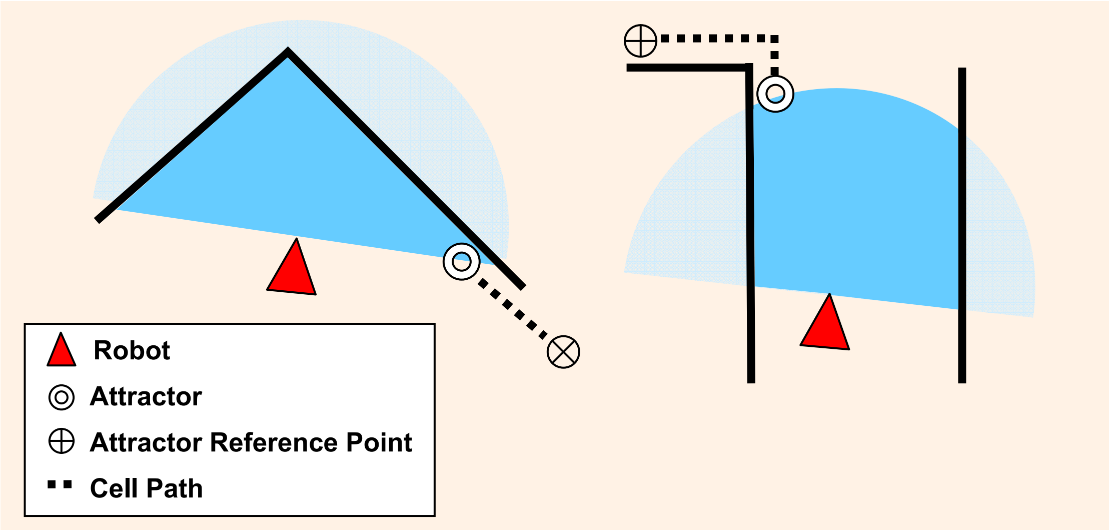


Figure 5-11 Placement of Attractor

5.7. Simulation Results

The proposed planning strategy using Attractor aided MPC is demonstrated for line-feature SLAM using iSAM in this section. The simulation results for MPC are also presented so as to provide a framework for discussion and to highlight its distinguishing features. The task of the robot is to map an enclosed indoor environment within a prescribed number of time steps. Three control options, N_ω=3, at each time step are available and a planning horizon of D=3 is used.

To illustrate the result where no planning is conducted, a "random select" method is implemented. The robot also has three available control options and a random selection is made from the options which do not result in collision with obstacles.

Two different indoor environments are given for the robot to map. The first is the floor plan of an office area similar to that of the office space occupied by the ARC Centre of Excellence for Autonomous Systems (CAS) at UTS. The second is an arbitrary environment, twice as large.

5.7.1. Simulation Results for a Section of the CAS Office Area

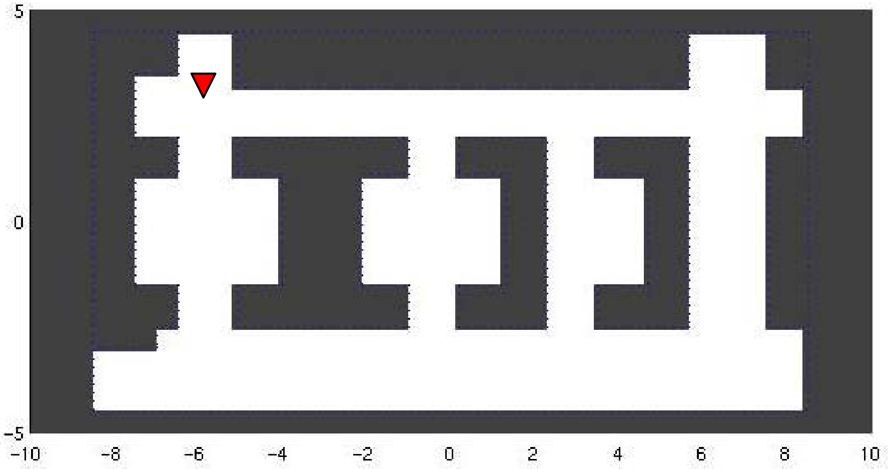


Figure 5-12 Section CAS UTS Floor Map (m)

The robot is initially placed at $\mathbf{x}_0 = \begin{bmatrix} -5.7 & 3.9 & -\pi/2 \end{bmatrix}^{\mathrm{T}}$ as denoted by the red triangle in Figure 5-12. The velocity and turn-rate are measured each step with $\Delta t = 0.055$ s and the laser makes a reading every 50 steps. The maximum velocity of the robot is set to be

$0.5ms^{-1}$ and the maximum turn-rate is $\pi/36rads^{-1}$. Simulations end after 3000 loops and updates are only computed when new observations are taken.

5.7.1.1. Map and Path

Mapping and planning using the MPC+Attractor strategy give good coverage and a well aligned map as seen in Figure 5-13. The robot pose estimate is accurate as evident by the current laser scan, depicted in green, matching the alignment of the walls. The magenta coloured star is the position of the attractor and the magenta coloured cross is the reference point for the attractor, which in this case is a group of frontier points. This particular path generated sensor coverage of 99.60% and the covariance of the line features is small.

It is apparent from the simulated mapping and planning using the MPC strategy (without the attractor) that there is no implicit strategy for the robot to explore. Figure 5-14 illustrates where the robot traverses many loops around its initial location to maximise information gain for the known features. For MPC, the uncertainty of line features is in general found to be small and the path of Figure 5-14 generated sensor coverage of 99.01%, which is a reasonable result.

For the random select method (Figure 5-15), there is no planning and the robot randomly moves from left to right while predominately walking in a straight line directed by the walls of the corridor. The robot does not return to localise unless, for example, where returning from a dead end. If the area was not enclosed, it could be expected that the robot would seldom cross a previously traversed area. Given the small enclosed map it is unremarkable that the random select method also achieved good coverage.

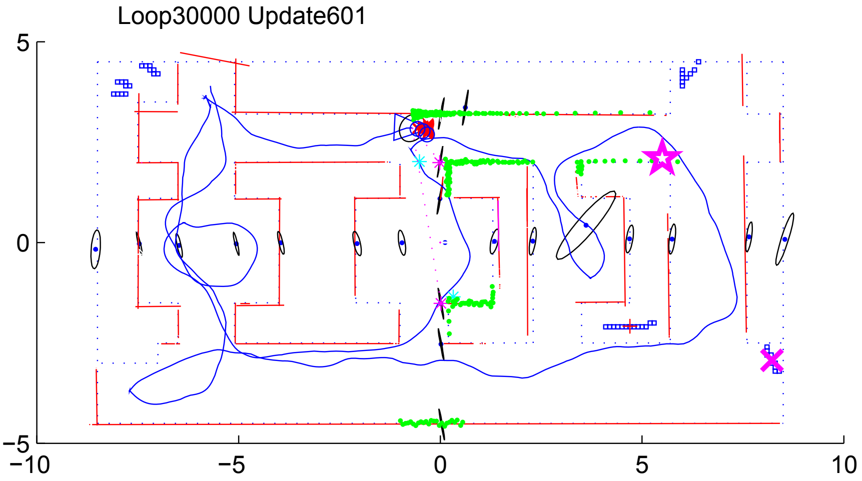


Figure 5-13 MPC+Attractor Path – CAS Map (m)

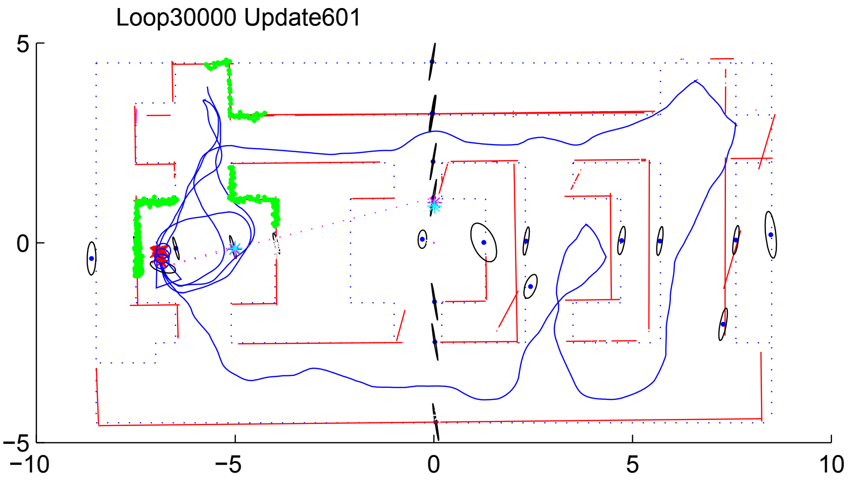


Figure 5-14 MPC Path – CAS Map (m)

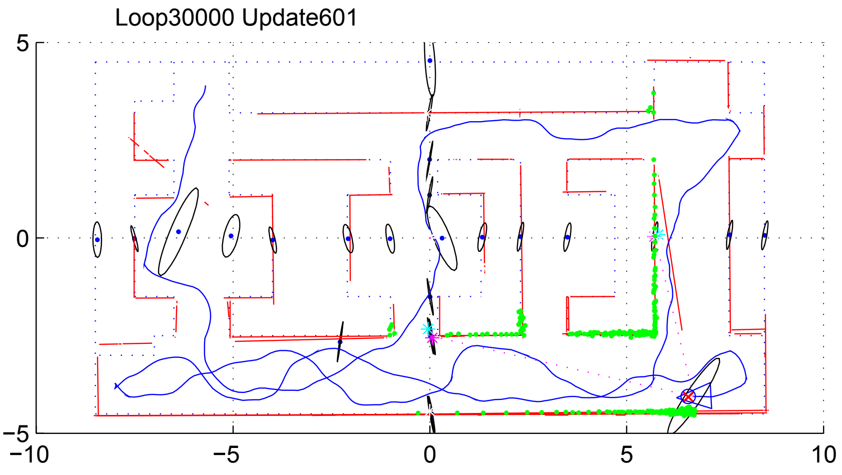
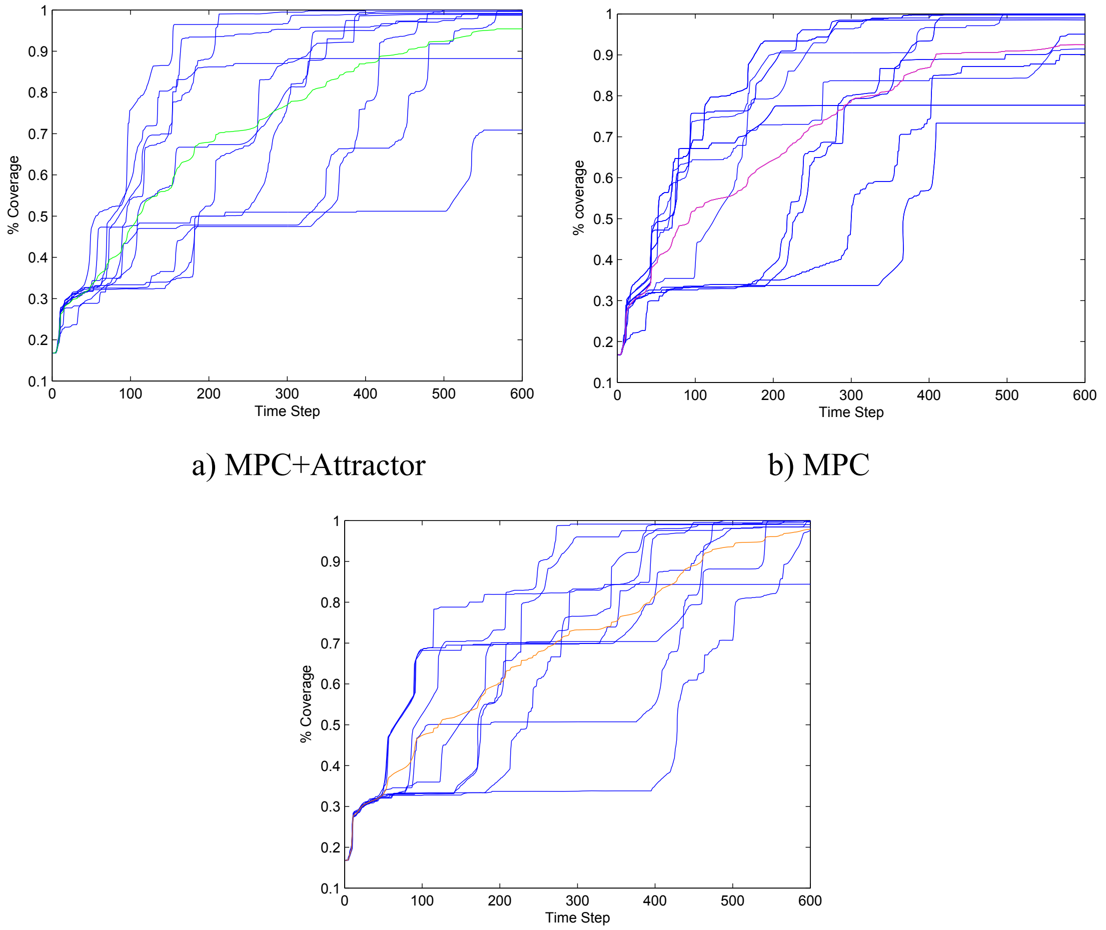


Figure 5-15 Random Select Path – CAS Map (m)

5.7.1.2. Rate of Coverage

The number of remaining unexplored cells is recorded at every time step and used to quantify the coverage rates. Ten trials were conducted and the rate of coverage achieved for the three methods is plotted in Figure 5-16 each showing a similar distribution, where the mean of these coverage rates are plotted in Figure 5-17. The MPC strategy and random select have the highest initial coverage increases and as time progresses the attractor strategy outperforms them by a small margin. For the latter, the knowledge of the map leads the robot to corners that have not been explored, which naturally would be a key advantage in many environments.



a) MPC+Attractor

b) MPC

c) Random Select

Figure 5-16 Rate of Coverage – CAS Map

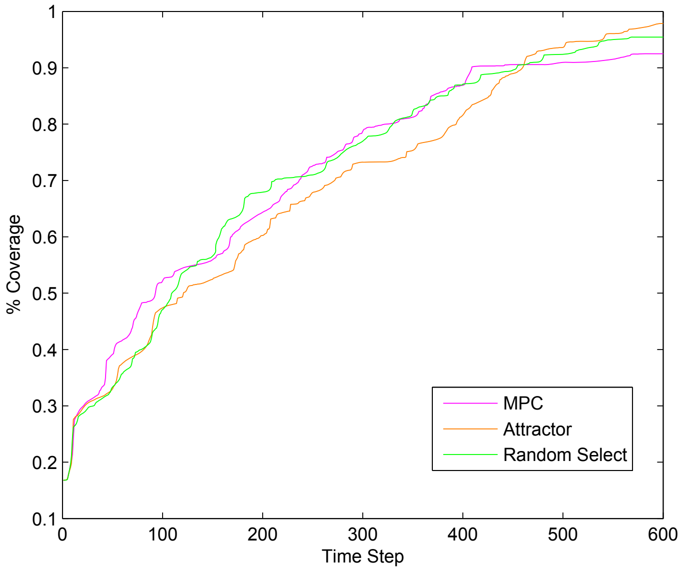


Figure 5-17 Rate of Coverage Mean for the 3 Strategies – CAS Map

5.7.1.3. Final Coverage and Uncertainty of State Estimate

The following tables show the resultant coverage and uncertainty of the state estimate from the ten trials. Table 5-1 displays the final percentage of coverage and the time step it has reached the maximum coverage. It can be seen in Table 5-1 that on average, MPC+Attractor obtains the highest coverage and is still covering new areas near the end of the simulation. For the MPC strategy, the robot on average fails to explore new areas after 75% of the time through the simulation and the random select strategy shows similar traits.

Table 5-2 displays the final uncertainty of the state estimates. It can be seen that the MPC strategy has the lowest uncertainty followed by MPC+Attractor and then the random select strategy. This result is consistent with the expected outcome as there is a trade-off between coverage and map accuracy in SLAM, such that the further the robot moves from its initial pose, the greater the uncertainty.

	MPC+Attractor		MPC		Random Select	
Trial	**C**	i_{cover}	**C**	i_{cover}	**C**	i_{cover}
1	99.77	593	100	521	99.14	527
2	99.86	539	99.77	395	99.62	428
3	88.43	549	99.48	458	99.16	410
4	98.44	527	77.57	204	98.74	545
5	97.50	599	87.23	427	99.78	327
6	99.80	546	73.31	409	98.85	432
7	99.66	595	95.04	579	70.88	551
8	100	491	35.90	554	99.97	500
9	99.03	552	99.01	414	88.20	304
10	100	527	98.46	284	99.73	568
Avg	**98.25**	**552**	**86.57**	**424**	**95.41**	**459**

Table 5-1 Final Coverage – CAS map

where C=final coverage percentage, i_{cover}= time step of obtaining maximum coverage

Trial	MPC+Attractor	MPC	Random Select
1	0.0414	0.0246	0.0095
2	0.0140	0.0114	0.0156
3	0.0084	0.0150	0.0274
4	0.0086	0.0082	0.0136
5	0.0110	0.0094	0.0188
6	0.0189	0.0095	0.0137
7	0.0148	0.0156	0.0382
8	0.0083	0.0168	0.0106
9	0.0149	0.0071	0.0181
10	0.0090	0.0104	0.0088
Avg	**0.0149**	**0.0128**	**0.0174**

Table 5-2 Trace of Covariance Matrix After 600 Updates – CAS map

showing *Trace*(P)/(number of rows in P)

Although these results show that the simulations with the Attractor produce a slightly better result than the other planning strategies, the result was not convincing. Mostly due to the fact that the robot sensor range relative to the size of the environment and the deviation of the values shown are quite large. A more complex environment is then created so as to purposely make it more difficult for a naive implementation, random select, to perform well. Such an environment would also provide a challenge the attractor aided MPC strategy.

5.7.2. Larger Environment with Divider

The size of the environment is increased such that the ratio of the sensor range to the map dimensions is smaller. The environment contains a wall dividing the area into two separate rooms, as in Figure 5-18. All the parameters from the previous simulation remain the same except the robot start pose is set to [0, -3, 0] denoted by the red triangle. The simulations are run for 100,000 steps with observations taken at every 50 steps.

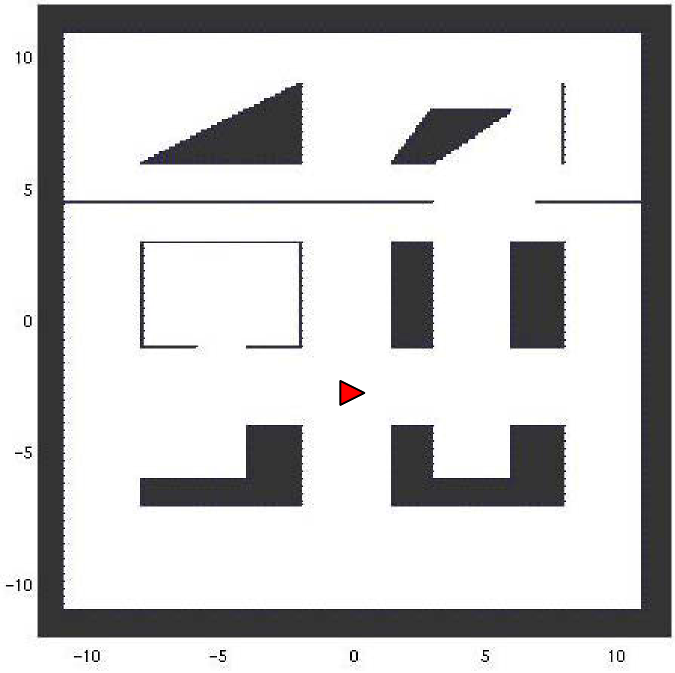


Figure 5-18 Larger Environment with Divider (m)

5.7.2.1. Map and Path

The resultant map and path for the three methods are displayed in Figure 5-19, Figure 5-20, and Figure 5-21. It can be seen in these maps that there are several false line segments. Occasionally, false line segments are initialised due to the noise and speed of turning or errors in the line extraction. There is nothing implemented at the moment to delete false lines. If an expected line segment is not observed then it may be deleted or segmented. A more suitable trajectory may be achieved if the map is more accurate but requires greater computation. An appropriate map management strategy could be used to delete the incorrect initialisations but this was not pursued further. Nevertheless, the robot is still able to maintain a consistent pose estimate as it is able to associate real observations to real lines and not to the false lines which are therefore not updated. The robot pose estimate is accurate as the current laser scan in green is aligned with the real line segments and not the false line segments.

The behaviour of the robot is quite different for the different strategies. In the trajectory generated using MPC, the robot predominately travelled around the centre of the map. There are more line features in the centre of the map and these features are close to the starting point of the robot. Observing features near the starting point minimises the robot uncertainty and staying in areas with more features leads to higher information gain. Hence the trajectory remains predominately in the centre and the robot rarely expands the path to the unknown areas of the map.

For the attractor aided MPC, the trajectory also remains near the starting point of the robot, however in this case, the robot does venture out further due to the attractor directing the robot out towards the frontiers. Thus the attractor once again improved the result of MPC, enabling the robot to explore. With knowledge of the line segments and the map, the robot was able to plan paths in and out of the rooms created by the divider.

In the random select strategy, it is found that the robot quickly covers the perimeter. The robot rarely stays in the centre of the map. Unlike the previous environment, this environment is open without as many dead-ends. Here, there are few areas that the robot cannot see by chance excluding the room created by the divider. The random select strategy does not perform as well in this environment in terms of coverage as there is a lower chance for this strategy to explore the entire area due to the divider. Analogous to other random methods such as Probabilistic Road Maps (PRMs), paths through narrow openings require many iterations of the algorithm to find.

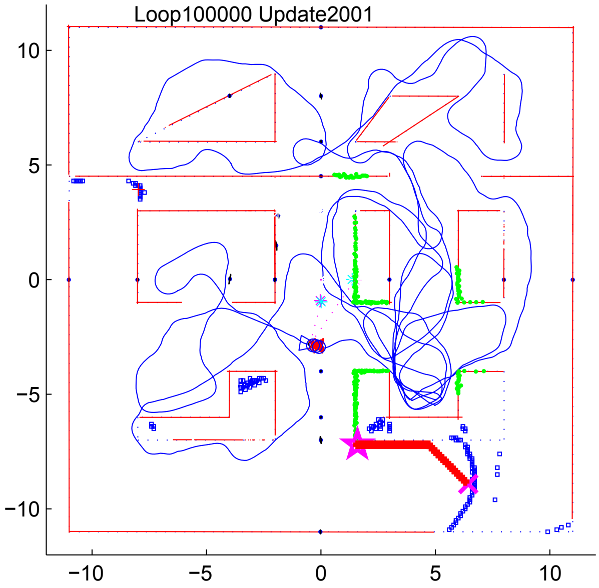


Figure 5-19 Attractor Path - Larger Environment

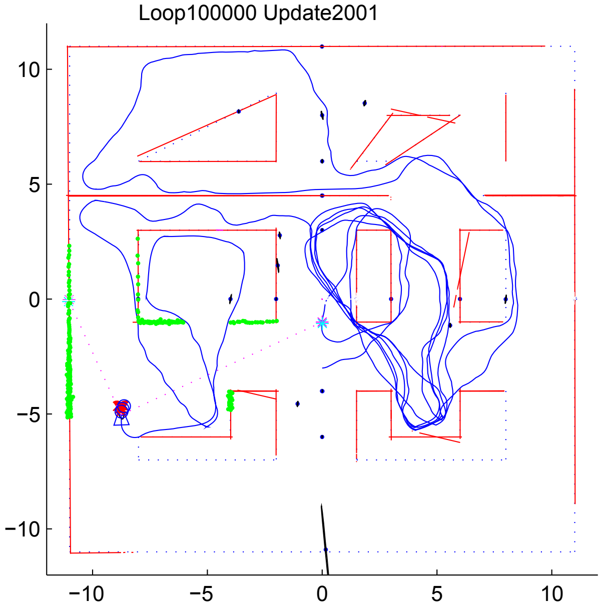


Figure 5-20 MPC Path - Larger Environment

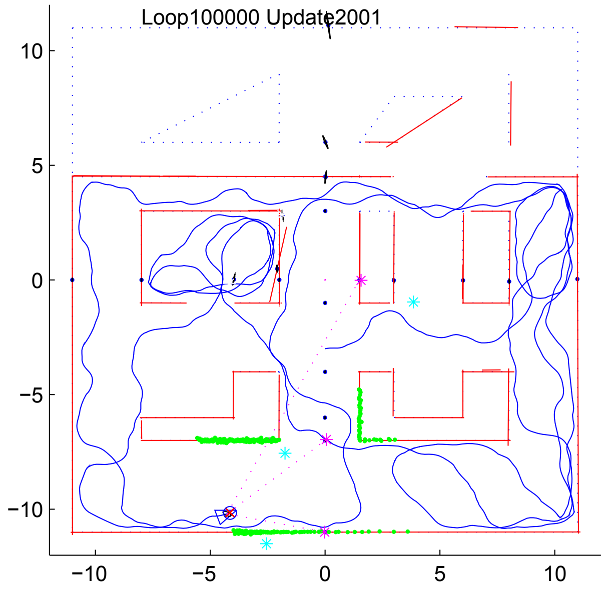
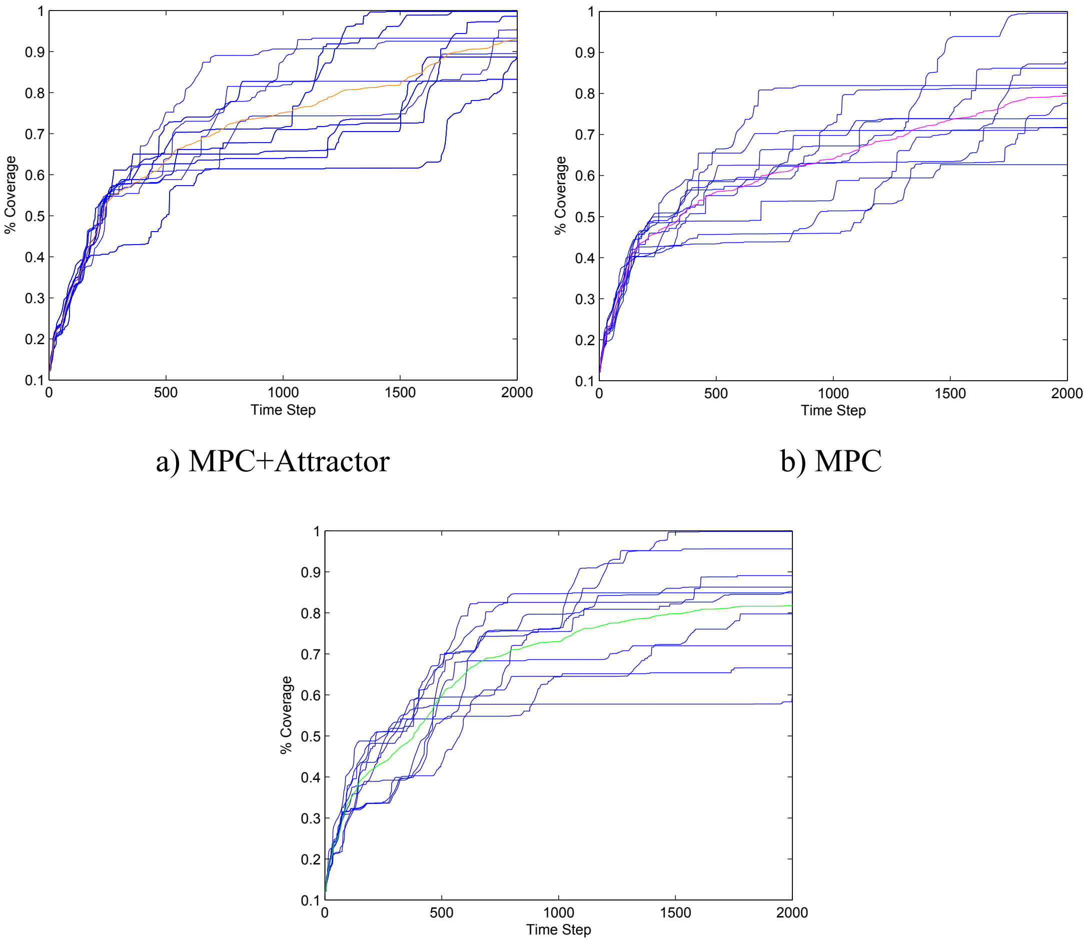


Figure 5-21 Random Select Path - Larger Environment

5.7.2.2. Rate of Coverage

The rates of coverage for the different strategies are shown in Figure 5-22 and Figure 5-23. It can be seen that the distribution of the coverage rates for the MPC and MPC+Attractor strategies are quite similar. The coverage of MPC+Attractor in the larger environment is much better than that of MPC alone. In the MPC strategy, as the robot remains near the good features and much of the space remains unexplored. The addition of an attractor has a significant influence on improving the coverage.

Random select method has the largest deviation of coverage rates depicting its unreliability as the variation in the result is large. Doorways or openings in the unknown environment reduce the probability of the robot to explore the entire space by random chance. Given that the environment to be mapped is unknown the using a random strategy is not a sensible choice.



a) MPC+Attractor

b) MPC

c) Random Select

Figure 5-22 Rate of Coverage for Larger Map - Larger Environment

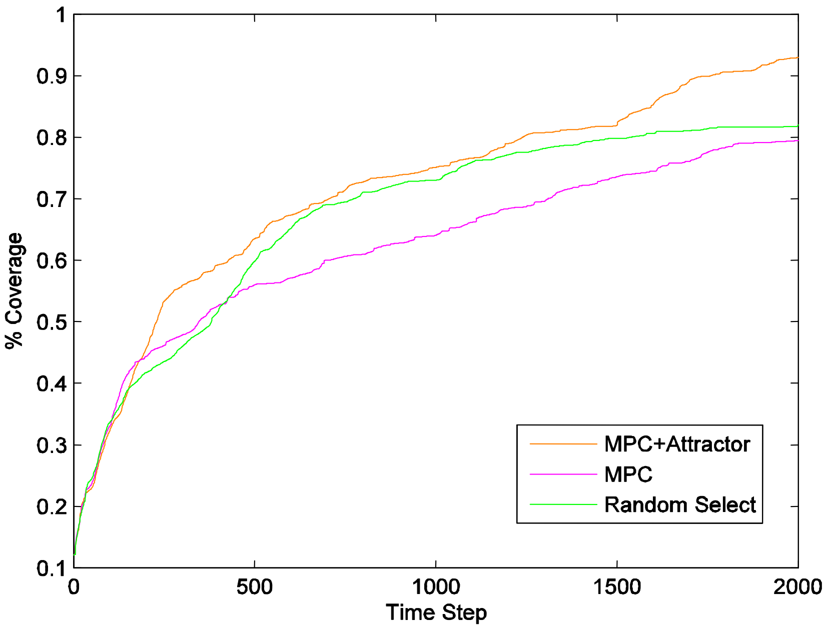


Figure 5-23 Mean of 3 Strategies - Larger Environment

5.7.2.3. Final Coverage and Uncertainty of State Estimate

The final values for coverage are displayed in Table 5-3. These values coincide with the rate of coverage figures. Clearly, there is a larger difference in the final percentage covered in that MPC+Attractor explored approximately 10% more of the environment than the random select method. This is largely the consequence of the divider placed in the environment.

Once again, from Table 5-4, it can be seen that the strategies that optimised for information gain had the lowest uncertainty of the state estimate and the strategies with higher coverage had larger uncertainty as expected.

	MPC+Attractor		**MPC**		**Random Select**	
Trial	**C**	i_{cover}	**C**	i_{cover}	**C**	i_{cover}
1	98.61	1945	81.47	1851	66.60	1787
2	99.91	1746	99.54	1832	99.85	1605
3	83.24	1872	73.87	1829	81.46	1854
4	88.54	2000	89.77	1945	86.28	1619
5	88.67	1646	86.14	1681	95.58	1531
6	99.79	1530	62.64	1459	60.08	2000
7	92.53	1439	71.69	1882	71.99	1353
8	93.27	1129	77.57	1980	89.10	1765
9	95.59	2000	64.47	1981	84.89	1076
10	89.50	1877	81.92	872	85.42	2000
Avg	**92.97**	**1718**	**78.91**	**1731**	**82.13**	**1659**

Table 5-3 Final Percent Coverage – Larger Environment

where C=final coverage percentage, i_{cover}= time step of obtaining maximum coverage

Trial	MPC+Attractor	MPC	Random Select
1	0.0005	0.0007	0.0010
2	0.0007	0.0010	0.0006
3	0.0008	0.0009	0.0018
4	0.0022	0.0007	0.0013
5	0.0005	0.0009	0.0050
6	0.0016	0.0009	0.0018
7	0.0013	0.0008	0.0010
8	0.0007	0.0018	0.0010
9	0.0009	0.0007	0.0011
10	0.0010	0.0005	0.0014
Avg	**0.0010**	**0.0009**	**0.0016**

Table 5-4 Trace of Covariance Matrix After 2000 Updates – Larger Environment showing Trace(P)/(number of rows in P)

5.8. Experimental Results

The proposed trajectory planning strategies are demonstrated in real-time in a practical experiment. Results using MPC as well as random select are also given. The details of the experiment and results obtained are described as follows.

5.8.1. Experiment Setup

The experimental setup is constructed from 4 sofas, several cardboard boxes, cushions and a table as seen in Figure 5-25. A Pioneer2DX robot, pictured in Figure 5-24 is used. This robot is equipped with a SICK laser range finder that allows scans to be taken up to 10Hz. The wheel encoders on the robot are used to obtain robot velocity and turn-rate. Each of the experiments is run for 150 observation steps. The robot is set to move at 0.15ms^{-1} with a maximum turn-rate of π/22.5rads^{-1}(8deg/s). The processor is 1.7GHz with 512MB of RAM.

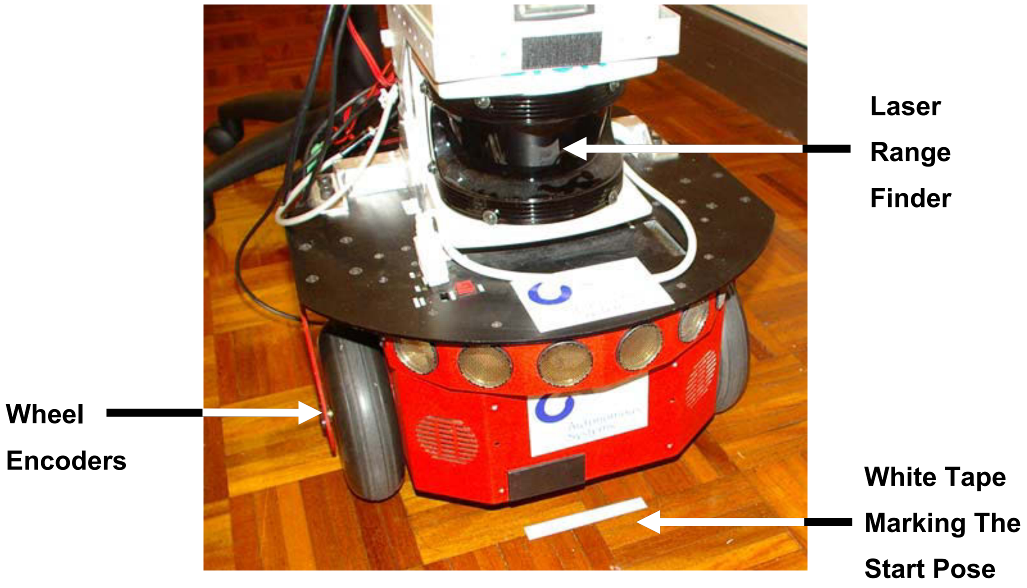


Figure 5-24 Pioneer2DX Robot

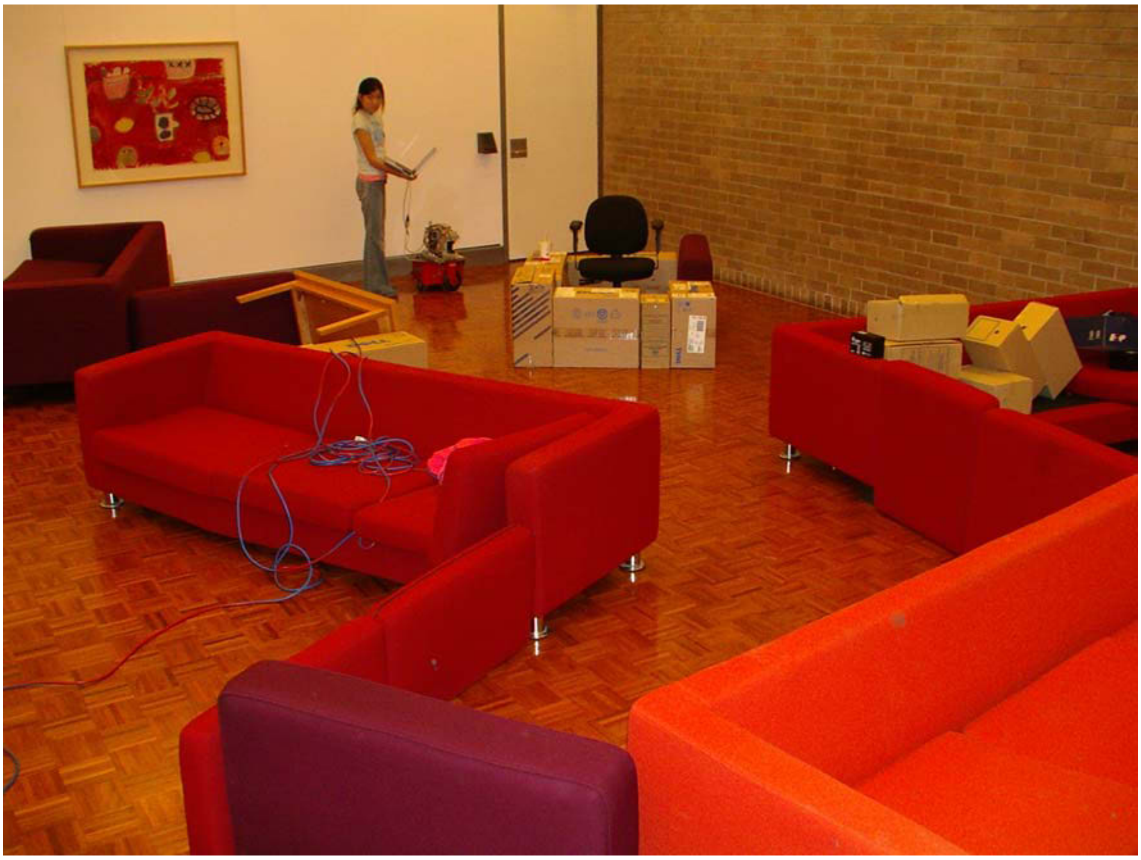


Figure 5-25 Experimental Setup

5.8.2. Extensions for Practical Implementation

The simulations applied the assumption that the computation time for the algorithms is instantaneous. In practice, this is not the case. The computation time required to process the observations, update the estimations and plan the trajectories is a major issue. It affects the frequency at which the robot's pose and map estimate is updated. The longer the computation time the longer the robot is unaware of its current situation and the less effective the planning. Several methods implemented to alleviate problems associated with this assumption are described below.

5.8.2.1. Robot Actions Recorded in Parallel to iSAM and Planning

It is required that the control measurements between observation time steps are recorded and used to obtain initial value for each *i*th robot pose for the iSAM. There are two main reasons: a) the computation time for iSAM and trajectory planning may take several seconds and the larger the time between updates of robot pose, the larger the error in the initial pose prediction by the linearised process model; and b) midway through the execution of a planned control action, it may be required that the robot change its control action to avoid an obstacle. Between the time steps *i*-1 to *i*, the control measurements are recorded by a thread parallel to the iSAM and trajectory planning computations in the array

$$\mathbf{C}_n^i = [v_n \quad \omega_n \quad \Delta t_n] \tag{5.66}$$

where Δt_n is the time difference between control measurements and $n = 1, \ldots, N_{steps}^i$, where N_{steps}^i is the number of control measurements between observation steps. Recall *v* and *ω* are velocity and turn-rate respectively.

5.8.2.2. Prediction of the Robot Pose at Time of Next Control Execution

It would be convenient to assume that the duration of a single time step (*i*-1 to *i*) for each control action is constant. However, in practice, the length of the control action increases with computation time. Planning for the next control action requires the system to know or predict the position of the robot after the computation time for planning has elapsed. If the trajectory planning uses the current pose estimate, then the trajectory is based on the robot pose at the time data is acquired and thus would be incorrect due to the computation time between data acquisition and plan execution as illustrated in Figure 5-26. First, data is acquired at the start of the first period (red) and then the robot pose is updated (end of yellow) to be at the time the data was collected. The planning then starts from the time where the system information is updated (start of blue bar) for the next control action. This plan is executed at the end of the planning process (green bar), i.e. based on the robot pose at the start of the time line. During this time the robot would have moved and is no longer at the position from which the data was acquired. Thus a prediction of the computation time between the times of data acquisition and the execution of new controls must be made.

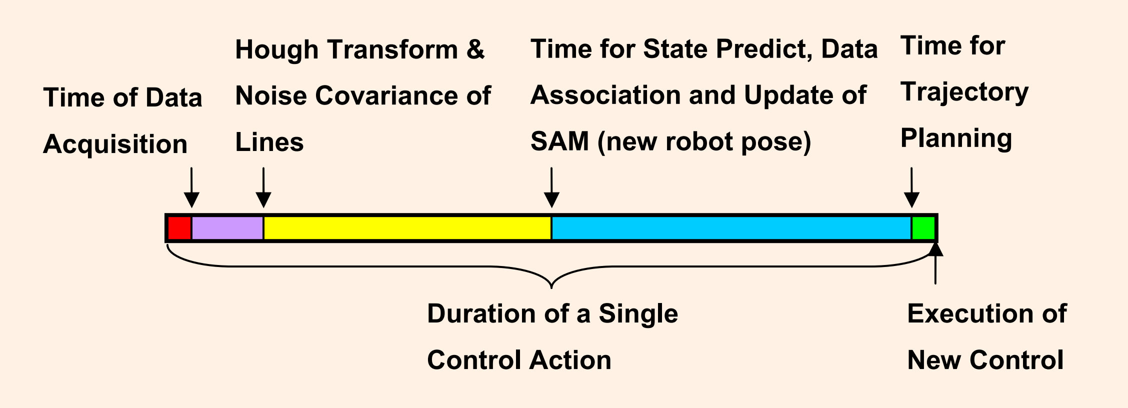

Time of Data Acquisition
Hough Transform & Noise Covariance of Lines
Time for State Predict, Data Association and Update of SAM (new robot pose)
Time for Trajectory Planning
Duration of a Single Control Action
Execution of New Control

Figure 5-26 Illustration of Delay of Plan Execution

The computation time for each control action varies with the number of features and robot poses, and the number of loops executed in the estimation and planning algorithms. This is the difficulty in predicting the duration of a control action. Generally, as the information and planning requirements evolve gradually, the duration of the previous steps are similar to that of the current. Hence the computation time of the previous step may be used to approximate the pose of the robot where the next planned control action is to be applied.

Prior to planning the trajectory, the robot pose from which the next control action is to be executed is approximated. Using the number of control measurements, N^i_{steps}, the mean of Δt_n for $n = 1, \ldots, N^i_{steps}$ recorded from the previous time step, the current control action $\mathbf{u}_n$ together with the process model (5.40) and (5.41), this pose is computed. Additionally, the robot poses in the tree search are determined using the duration of the previous step; i.e. during the planning phase, N^i_{steps} and the mean of Δt_n for

$n = 1, \ldots, N^{i}_{steps}$ are also used to predict the pose of the robot for the N_{ω} different control options in the planning horizon. Figure 5-27 illustrates this approach.

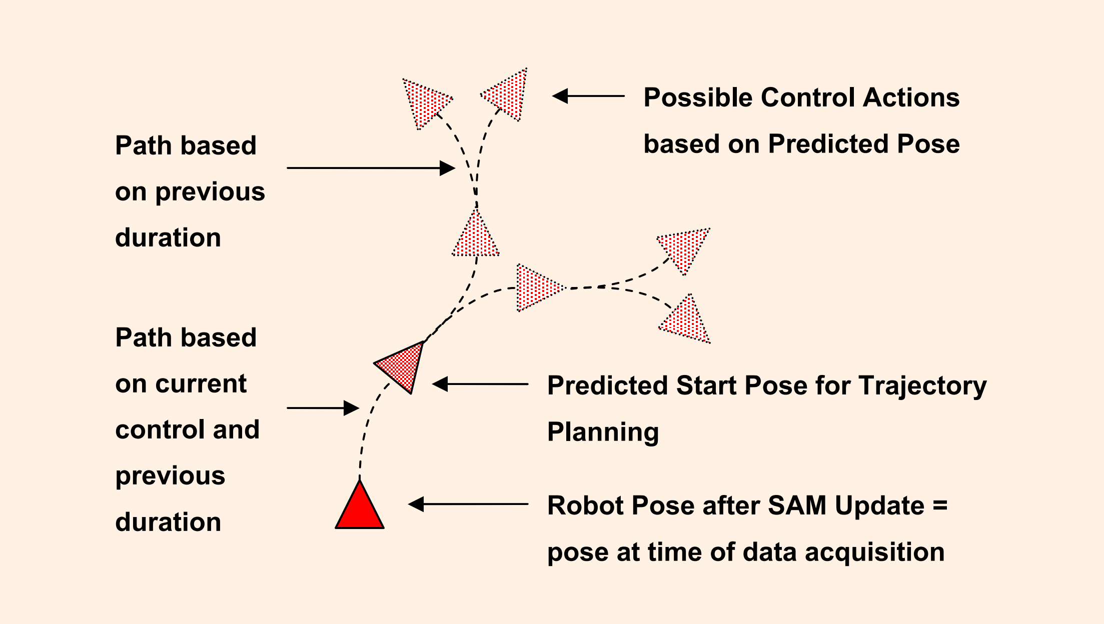

Possible Control Actions based on Predicted Pose
Path based on previous duration
Path based on current control and previous duration
Predicted Start Pose for Trajectory Planning
Robot Pose after SAM Update = pose at time of data acquisition

Figure 5-27 Predicted Start Pose

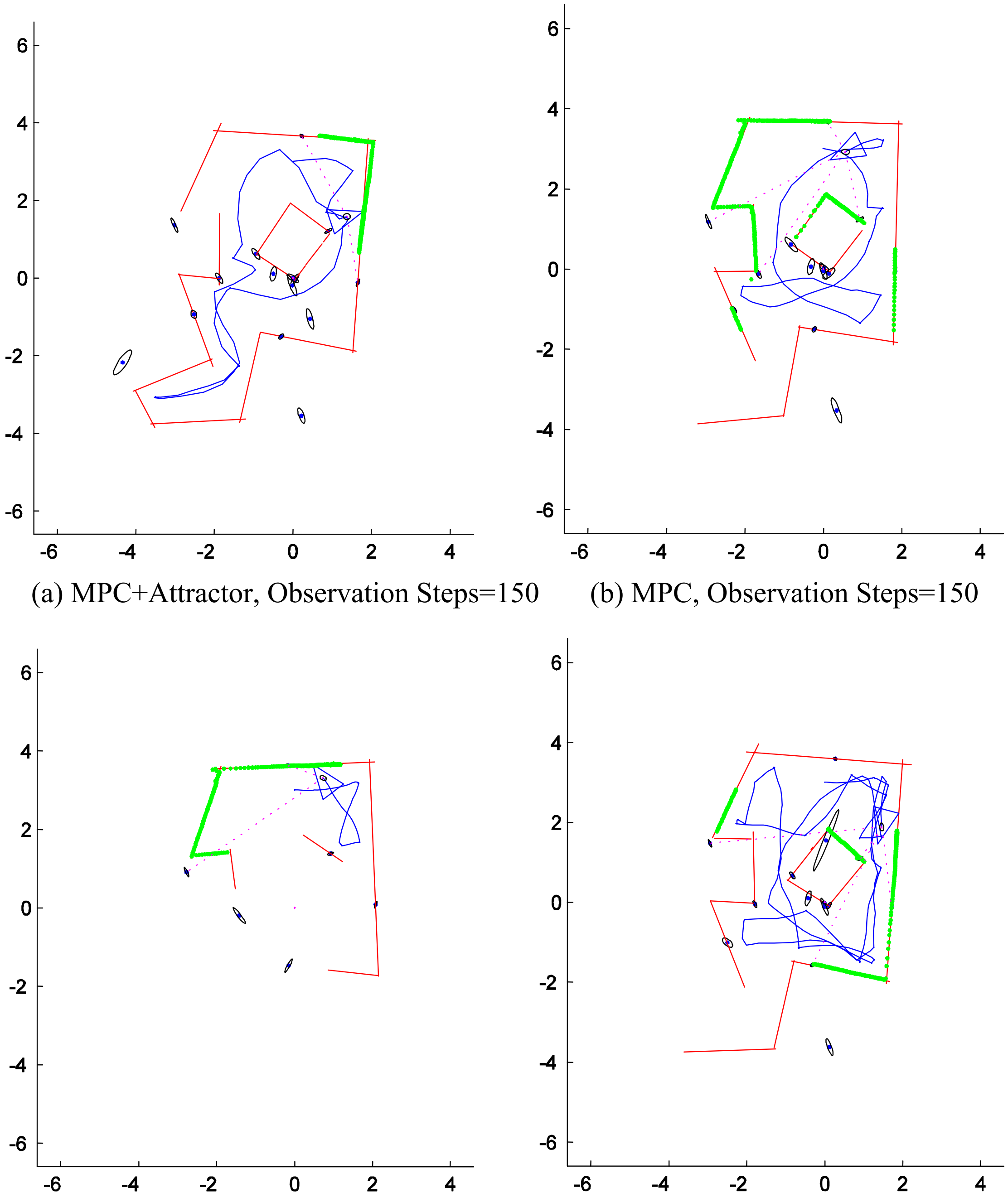
5.8.2.3. Emergency Obstacle Avoidance

Practical implementation of the planning algorithms requires the inclusion of overriding obstacle avoidance in case the robot encounters unexpected obstacles. The planning process may have errors, including but not limited to longer than expected computation times and significant noise in measured velocities. To act quickly (much faster than the planning algorithm) the obstacle avoidance relies only on laser measurements and if collision is imminent, the control action is to stop and turn away from the obstacle. This emergency obstacle avoidance process is also placed in the thread where control measurements are taken.

5.8.3. Map and Path

Figure 5-28 displays the resultant map and path for the three methods after 150 observation steps. The robot trajectory appears quite jagged in the diagrams. The reason for this is that only the robot poses at the update step are displayed. The poses between updates are not displayed as the relative pose is computed and intermediate poses are not included in the state vector (see Section 5.5.1).

It should also be noted that the maps displayed in Figure 5-28(a)-(d) are not all identically aligned. This is due to the variations in the start position of the robot between experimental runs. Although the start pose was marked with white tape as displayed in Figure 5-24, the robot cannot be positioned exactly the same for each experimental run as the placement relies on human judgement.



(a) MPC+Attractor, Observation Steps=150 (b) MPC, Observation Steps=150

(c) Random Select, Observation Steps=150 (d) Random Select, Observation Steps=450

Figure 5-28 Map and Path from Practical Experiment (m)

The map from the MPC+Attractor strategy in Figure 5-28(a) appears to be mostly complete. After 150 observation steps, most of the area is explored except for a single line-feature. The map from the MPC strategy is less complete as the robot in Figure 5-28(b) remained near the starting point at the top of the map. The map generated by the random select method is the least complete. In Figure 5-28(c) it can be seen that the robot has turned three times in the same vicinity due to poor decision making.

For the random select method, the path is relatively short. This corresponds to the computation time required for each loop. The robot, travelling at the same speed as the other strategies, travels for a shorter time due to the shorter computation time. It is difficult to observe the behaviour of this method for such a short experiment time. The Random Select strategy is run again for 450 observation steps and the map is displayed in Figure 5-28(d). For this experimental run, the total time is 312 seconds longer than the time for the MPC+Attractor. However the map from the Random Select is still less complete than the map from MPC+Attractor.

5.8.4. Loop Times

The computation time for each loop, previously illustrated in Figure 5-26, is realised for the three methods as displayed in Figure 5-29. It can be seen that the computation time for MPC+Attractor is significantly longer than that of Random Select. However despite the longer computation time, it is demonstrated that the algorithm can work in real-time to perform active SLAM. The primary reason why a loop time of 4 seconds or over is still manageable is by the strategy described in Section 5.8.2.1 where the measurements of the control actions are recorded independently.

Furthermore, it can be seen that the majority of the computation time is consumed by the MPC planning. The computation time for the iSAM update is predominately the computation of the Random Select strategy. It can be seen that after the initialisation phase the computation time for the Random Select strategy increases gradually. This is due to the increase in the number of features and the increase in the number of poses, even for a small environment such as this and the small amount of loops it is quite apparent.

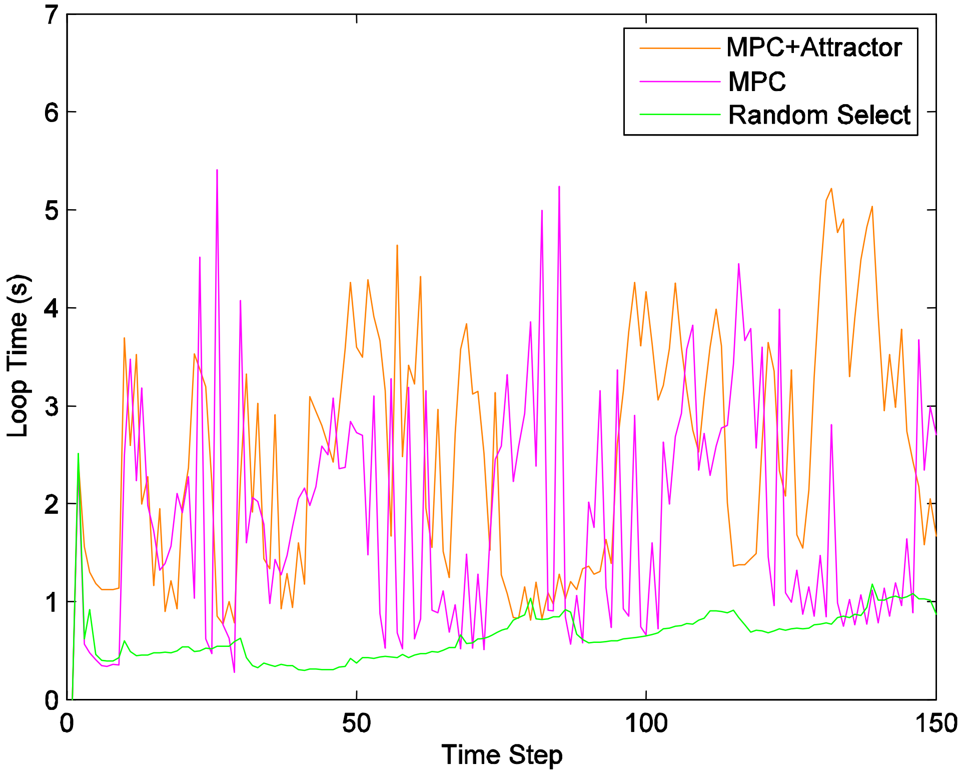


Figure 5-29 Loop Times vs. Observation Time Steps - Experiment

From Table 5-5 it can be seen that the length of the path (row 1) and duration of the experiment (row 3) is directly coupled with the computation time for each loop (row 4). It can be seen that the accuracy of the map (i.e the final trace of the covariance matrix **P** in row 6) for the MPC+Attractor strategy is not as good as the other two strategies. This is due to the lower frequency at which the estimation is updated.

The Random Select strategy enabled the robot to update its control, process information observations and update its estimate at a smaller time interval allowing a more accurate estimation. However despite the longer computation time required for the attractor, a good estimate of the map is still maintained.

		MPC + Attractor	**MPC**	**Random Select**	**Random Select x3**
1	Total Number of Control Measurement Steps	6483	4912	1544	1655
2	Total Number of Observation Steps	150	150	150	450
3	Total Experimentation Time (s)	374	281	98	686
4	Average Loop Time (s)	2.4784	1.8599	0.6477	1.524
5	Average Time between Control Measurements (s)	0.0656	0.0640	0.0708	0.0673
6	Trace($\mathbf{P}_T$)/number of rows in $\mathbf{P}_T$	0.0022	0.0018	0.0012	0.0034
7	Number of Features	15	13	6	14

Table 5-5 Experimental Results

In parallel to the main thread that computes the SLAM and trajectory planning, a control measurement thread is implemented for the practical experiment. To demonstrate the operation of the thread, Figure 5-30 and Figure 5-31 are displayed below. Figure 5-30 shows the number of velocity and turn-rate readings taken for each loop. This is directly correlated to the loop time (Figure 5-29). The longer the computation required to compute the SLAM and trajectory planning the greater the number of control measurements taken. As there is only one computer, the processor switches between the two threads regularly. In Figure 5-31 the time between taking velocity and turn-rate measurements is shown. It appears that it generally takes a measurement every 0.05 or 0.085 seconds. The difference in these times may be due to the obstacle avoidance being in the same thread or perhaps simply the operation of the processor. The more frequent the measurements, the less linearity error caused by the linearised process model as without using a separate thread (i.e. using only a single control measurement for each observation update), the error in the predicted robot pose for the initial guess for the iSAM was so large that the algorithm could not converge.

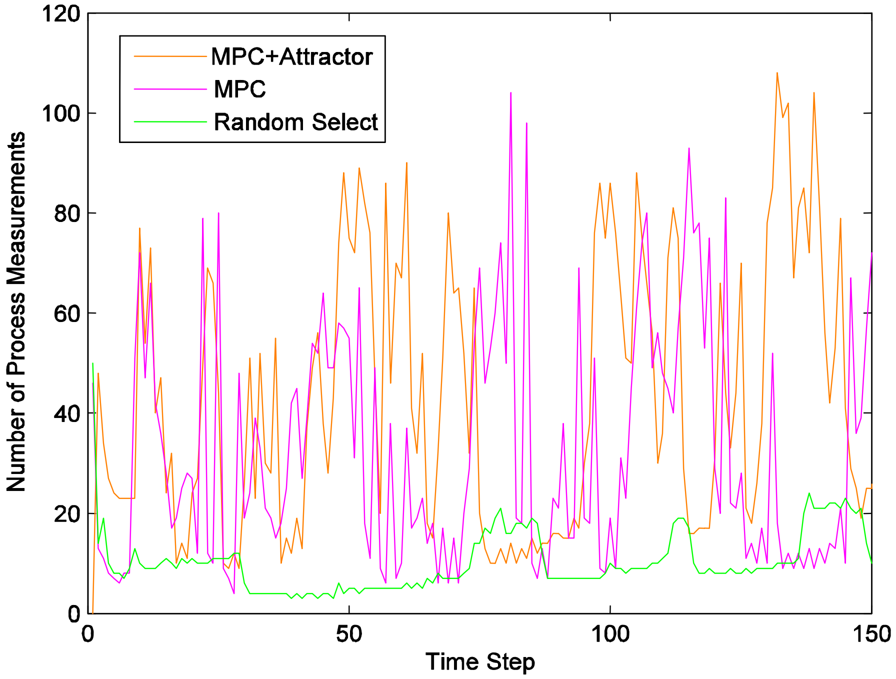


Figure 5-30 Number of Process Measurements per Loop

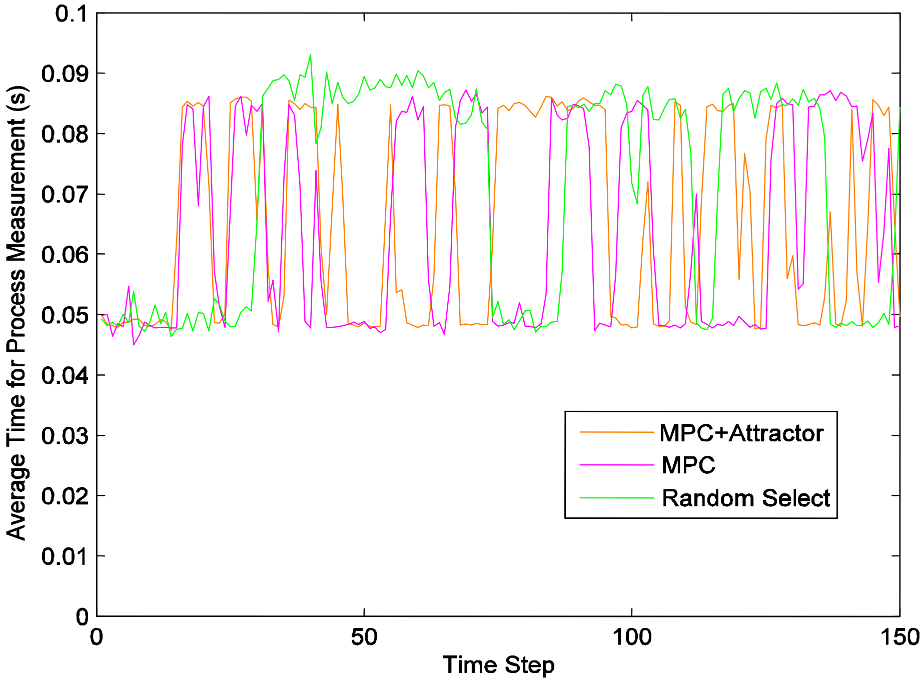


Figure 5-31 Time between Process Measurements

5.9. Discussion

5.9.1. Reduce Poses in State Vector

Although the iSAM algorithm provided consistent estimates, computation time remains an issue for long experiments. When the number of poses grows large in the iSAM state vector, a large amount of memory is required to store the corresponding information and covariance matrices. The computation time increases and the trajectory planning eventually becomes ineffective. The number of poses far exceeds the number of line features as there are generally only a few dominant lines in a structured environment, thus reducing the number of poses retained in the state vector would significantly reduce computation.

Previous robot poses can be removed by following the procedure in Section 5.5.4. A formulation of a new SAM algorithm was developed to compute the estimates using the observations obtained in the subsequent time steps after removing the poses. In this new algorithm, the state estimate immediately after the removal of poses is treated as an observation with the covariance of the estimates used as the covariance of the observation. An additional term is introduced into the least squares problem to include the state observation and is then solved. Further details are given in Appendix C. Removing the poses reduce the size of the information matrix but effectively make the information matrix denser.

To evaluate the effectiveness of this strategy of removing poses, a dataset of 1000 time steps was taken of the office area of the Centre of Excellence for Autonomous Systems (CAS) at UTS as seen in Figure 5-32. The laser scans, robot velocity, turn-rate and time were recorded at each time step. Each time step is approximately 0.3 seconds. This dataset was processed offline on 3.0GHz Pentium 4 processor with 2GB 400MHz DDR-RAM.

It was found through processing this dataset that the computation time (Figure 5-33) increased exponentially when all the poses were maintained in the state vector. When the poses were removed, the computation time remained short. However, the more frequently poses were removed, the more inconsistent the map estimates (Figure 5-34).

It should be noted that the EKF formulation is equivalent to removing poses at every step.

Ideally the covariance of the robot pose with the removal of poses should not differ from the case where all the poses are maintained. However it is observed that the covariance of the robot pose estimate (Figure 5-35) does change slightly after removing the poses. The main factor that may attribute to the differences in covariance is the fact that all the previous poses and associated observations could no longer be updated at every time step when the iSAM convergence loop is applied. The variation in the robot pose estimate may have attributed to incorrect data associations.

A strategy into which poses can be removed without causing inconsistencies in the estimation is necessary. There may be critical poses in the state vector that needs to be maintained for a consistent estimate. These could be the initial start pose or perhaps poses where loop closure occurs. A criterion for selectively removing poses is yet to be determined. Alternatively there may be critical instances where removal of poses is fatal for the estimation to remain consistent. In the experiment, the robot makes a loop around a set of cubicles. The robot covariance grows until it closes the loop. Removing the poses as the covariance grows may not be beneficial for the estimation; it may be best to wait until after a loop has been closed or when the covariance of the robot is low. These strategies require further investigation however as the focus of this thesis is on trajectory planning, they are not explored any further.

With the inconsistency found in the result obtained from removing the poses, it is decided that this algorithm developed to reduce robot poses would not be used in the actual experiment presented in this thesis.

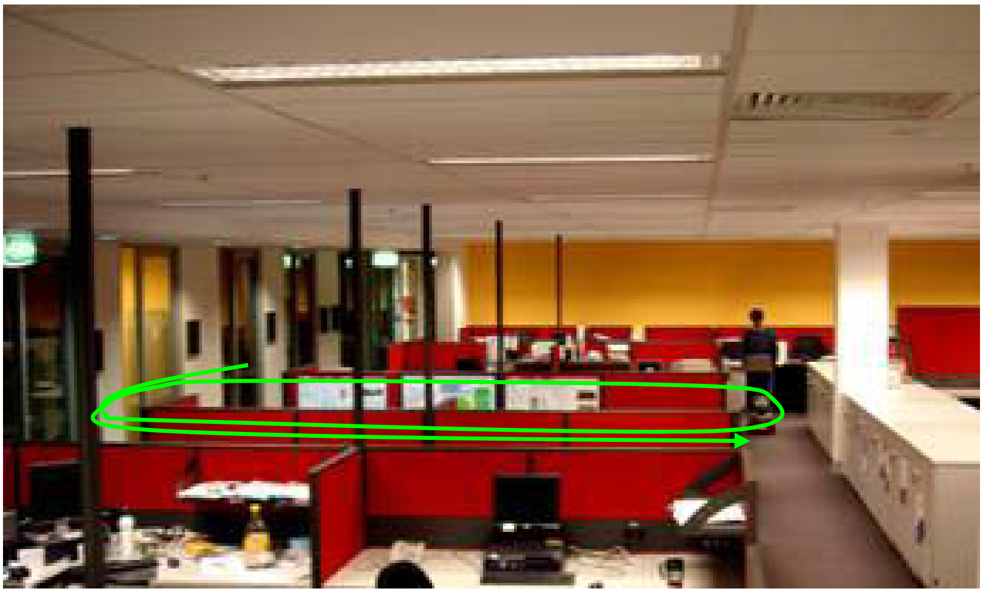


Figure 5-32 Environment of Dataset (CAS UTS office)

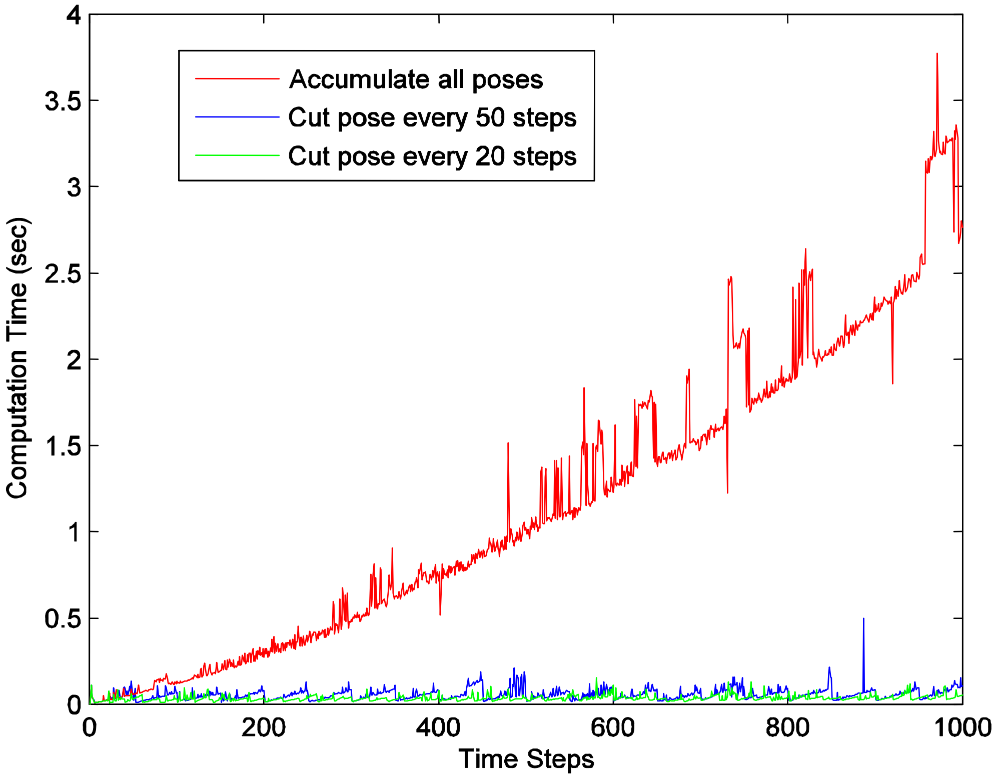
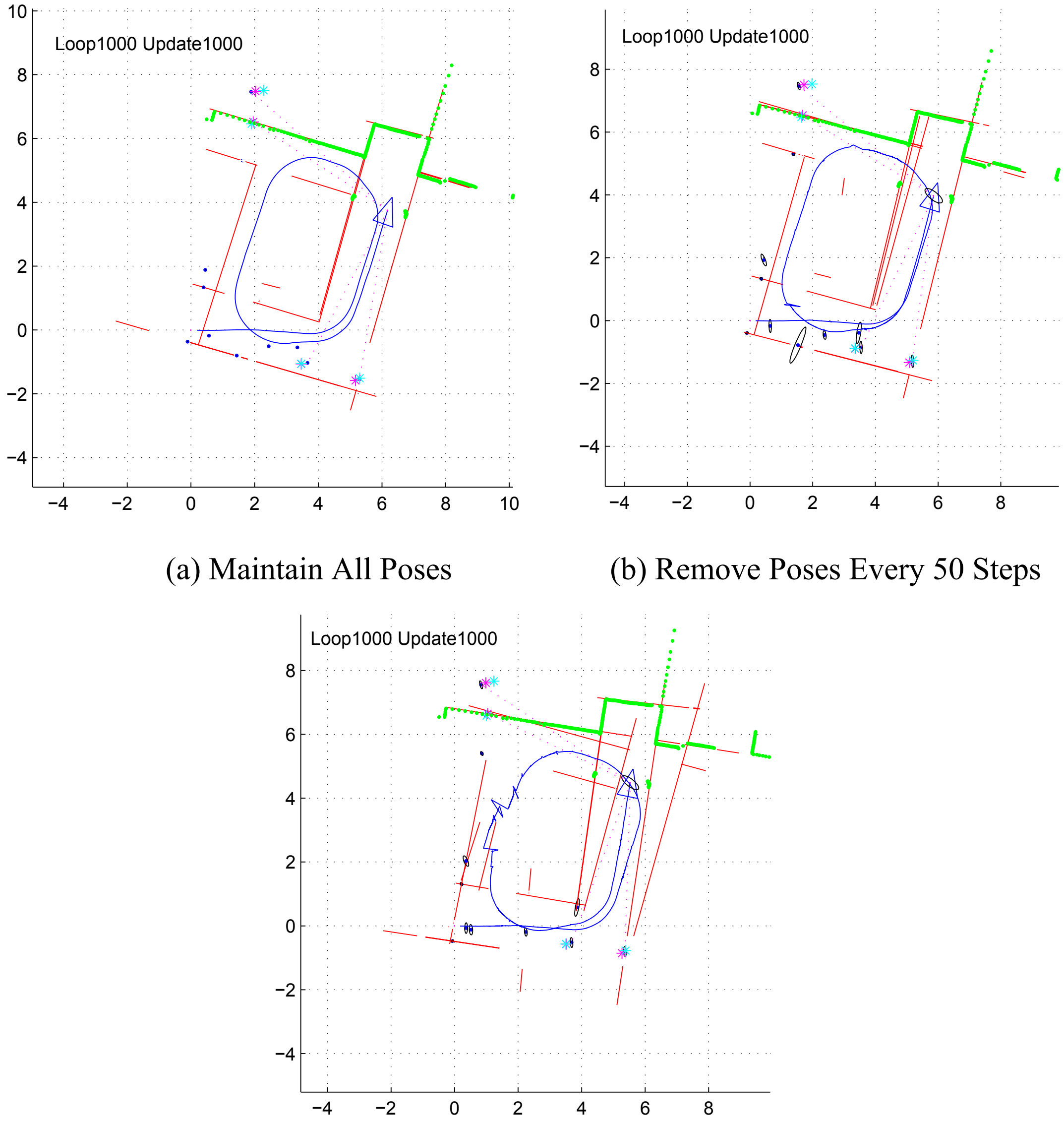


Figure 5-33 Computation Time for iSAM



(a) Maintain All Poses

(b) Remove Poses Every 50 Steps

(c) Remove Poses Every 20 Steps

Figure 5-34 Map from Removing Poses in SAM (m)

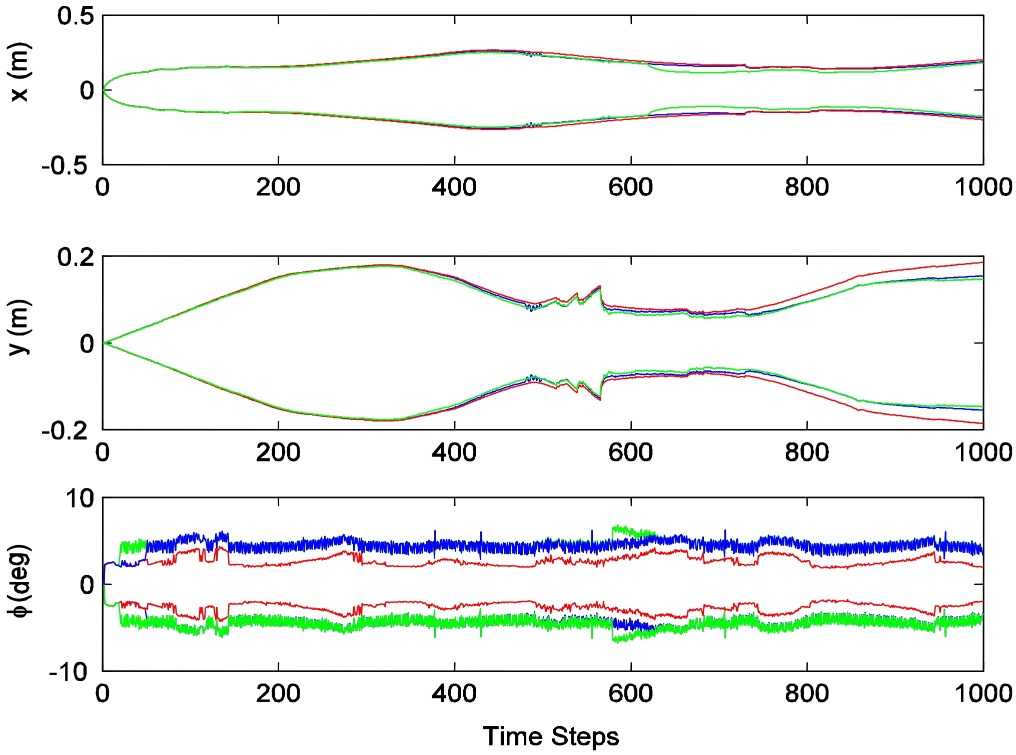


Figure 5-35 Covariance of Robot Pose for iSAM

where green=remove poses every 20 steps, blue=50 remove poses every steps, red=maintain all poses

5.9.2. General Practical Issues

It must be noted here that a faster processor will result in a higher frequency of updates and hence the results would be more accurate and the robot would be able to move faster. The experimental results were conducted on a computer that is 4 years old with a 1.7GHz processor and 512MB of RAM. Currently there are computers available that have dual processing power and with significantly more memory. Additionally, the current state of the system is displayed using a 16MB graphics card which is quite slow but was required for monitoring the current status during the experiment. A combination of Matlab and C was also used in the experiments. Although Matlab simplifies implementation, it is less efficient at processing code. These issues resulted in a lower frequency of updates especially for the more intelligent strategies.

During the experiment, the start position was marked with a tape. However the robot may not have been at exactly the same angle or position given human error. Hence the map may not always be perfectly aligned between experimental runs. The differences inherent in the initial robot start pose may also affect the decisions made in the planning.

One of the issues encountered during the experiment was the battery life of the robot and the laptop. It is discovered that they rarely lasted longer than an hour. Although this is not an issue of the planning algorithms, it should be noted that power cables for the laptop were dragged across the experiment which created dynamic obstacles or false readings when moved across the face of the laser scanner and would sometimes move boxes creating.

The lines in the environment were required to be at least a metre long for good detection. The robot was able to avoid smaller obstacles but the inclusion of smaller line segments in the estimation and map did not work as successfully.

The Pioneer Robot was not able to run smoothly at low speeds on carpet. Also, with the weight of the batteries, it also made it difficult to turn on the spot. Even on hard ground, there was significant slip of the robots wheel when it turned on the spot. This also prevented the laptop to be mounted on the Pioneer robot during the experiment. This resulted in the requirement of following the robot with the laptop and act as a human extension of the robot. This caused some dynamic obstacles or false readings when the robot turned unexpectedly and the laptop carrier could not get out of the way in time.

There were insufficient features for the robot to localise with in the CAS environment. Although the CAS environment has many long straight lines, they were mostly parallel. When the robot is moving down long corridors it is found that the robot has a tendency to loose its location along the corridor. It does not have sufficient features to localise with when there are only two parallel lines.

For short experimental trials, recording the robot velocity and turn-rate between updates is sufficient for real-time implementation. The computation time becomes a problem when it is larger than approximately 7 seconds depending on the robot speed. A method to reduce computation for long experimental runs requires further investigation.

5.10. Summary

This chapter studied trajectory planning for line-feature SLAM. SAM was used to solve the estimation problem since the EKF was found to give inconsistent estimates for large scale implementations (Rodriguez-Losada et al. 2006). The covariance of the noise of a line measurement was computed through a least squares method based on the laser measurements. The SAM algorithm was also modified to reduce the number of poses in the state vector and computation time by computing the relative pose of the robot between consecutive observation steps. This also reduces accumulated errors in the robot pose estimate.

Two planning strategies, using MPC and MPC+Attractor, are proposed. Modifications to these strategies are made to adapt them to structured environments. To determine frontiers for exploration, an occupancy grip map was incorporated to the trajectory planning. A demonstration of these strategies is given. Purely optimising for information gain with MPC is found not to be suitable for line-feature SLAM. Adding the attractor to MPC significantly improved the coverage and allowed it to handle more complex environments. Several modifications to the algorithm were also implemented to assist practical implementation. The proposed strategy, MPC+Attractor, is shown to perform well in all the simulations and real-time practical experiments conducted.

Chapter 6. Conclusion and Future Work

This thesis provides a solution to effectively map and localise through planning trajectories such that the information gain from selective observation points are maximised. Three problems were considered: 1) Multi-agent bearing-only target localisation 2) Point feature based Active SLAM, and 3) Line feature based Active SLAM. Certain objectives were set (see Section 1.3) to advance the trajectory planning strategies for feature-based localisation and mapping amid uncertainties. This chapter summarises the principle contributions made in meeting these objectives and provides suggestions for future research directions.

6.1. Summary of Contributions

The following lists the major contributions made in this thesis.

6.1.1. Formulation of the Trajectory Planning Problem as Optimal Control Problems with Gradually Identified Models

The trajectory planning problems were formulated as optimal control problems with gradually identified models as not all the information can be predicted before the observations are made. Assumptions are clearly stated for the unknown observations to convert the optimal control problem to a deterministic system. This is central to enabling multi-step predictions in the trajectory planning. In Chapter 3 bearing-only target localisation using the EIF was considered under the objective of maximising information gain. Chapter 4 considered minimisation of uncertainty for point-feature based SLAM using the EKF and Chapter 5 considered minimisation of uncertainty for line-feature based SLAM using iSAM.

6.1.2. Application of MPC for Trajectory Planning of Feature-based Localisation and Mapping

Model Predictive Control (MPC) is applied to the three tasks of feature-based localisation and mapping. The MPC strategy entails planning multiple steps ahead and only executing the first step, so as the updated information is exploited. Constraints such as robot maximum turn-rates, sensor field of view limitations and no-go-zones are incorporated into the trajectory planning. In Chapter 3, two optimisation strategies are introduced. The first strategy, EETS, conducts a coarse exhaustive tree search. This is then compared to EETS+SQP, where the solution from EETS is refined using SQP. Analysis of the results shows that EETS provides a near optimal solution with significantly less computation than EETS+SQP. Subsequent chapters, Chapter 4 and Chapter 5, also use EETS.

6.1.2.1. Multi-Agent Target Localisation

Simulation results in Chapter 3 showed that planning multiple steps performs better than the single-step planning method. On the other hand, increasing the number of control options at each step to a fine discretisation in the trajectory planning is found to increase the computational cost significantly with only minor benefits in information gain.

Cooperation among robots during the trajectory planning phase is shown to be valuable in the task of multi-agent bearing-only target localisation. A naive decentralised solution for using multiple robots for this task is implemented. As expected, the computation time for a decentralised implementation is reduced compared to a centralised approach at the cost of optimality in information gain.

The importance of representing the information in the form of an information matrix is also demonstrated. Paths generated for features with particular information content are applied to other features with identical determinants and eigenvalues but different information content. It is found that the information gained from a trajectory specifically optimised for particular information matrices of location estimates is superior to using trajectories optimised for other environments with identical scalar measures of location estimates.

6.1.2.2. Active SLAM

In the case of SLAM, it was found in Chapter 4 that for large process noise or high uncertainty in the system estimate, extending the planning horizon does not necessarily equate to higher information gain. Long term plans may not be realised when the uncertainty of the system is high. Thus the computation time may increase exponentially without achieving proportional gains in information.

Furthermore, MPC was found to be flexible and able to adapt to new constraints and information. In the SLAM case, when new features are detected, new no-go-zones and new information are easily incorporated into the MPC strategy. They are simply added in the following time steps. In addition, parameters for MPC, such as the planning horizon and the number of control options are flexible and can be made to meet the computational capacity.

6.1.3. Incorporating Long-term Goals

MPC alone lacks long term look-ahead in the planning and also lacks any implicit strategy for exploration. Thus a strategy based on an attractor is developed in Chapter 4 to improve the performance of MPC for point-feature based SLAM. Modifications to the attractor based strategy are made in Chapter 5 for line-feature based SLAM. The purpose of this attractor is to allow the incorporation of long term goals and provide an incentive for exploration for the MPC strategy. In this strategy, three modes a) explore, b) improve map and c) improve localisation, are devised for a state machine based on the fundamental tasks observed in SLAM. A feature or point of interest is then selected based on the current mode as the reference point. It is proposed to use the attractor in the form of a virtual feature and placed in the direction of the reference point. MPC is implemented with an attractor and demonstrated to give improved coverage.

6.1.4. Active SLAM for Line Features

A novel implementation of performing trajectory planning for line-feature SLAM was presented in Chapter 5. Since line features are natural features already present in structured environments, they obviate the need for pre-positioned laser beacons. Line features also provide a more informative map to the user and assists in trajectory planning in a structured environment. There is more information contained in a line-feature map than in a point feature map as more range measurements are utilised in each laser scan. For line feature observations, an algorithm is developed to determine the observation noise covariance using a least squares method.

6.1.4.1. Application of iSAM to Line-feature Mapping

In Chapter 5, implementations using EKF with line features were found to fail in a short time. iSAM is introduced as an alternative estimation algorithm to replace EKF. The iSAM algorithm is based on a least-squares formulation and is claimed to be more consistent in its estimation. Line-feature mapping is formulated for iSAM and the method is shown to maintain consistent estimates. Multiple robot poses between the steps where observations are made are predicted in a batch mode using the relative pose to reduce accumulated errors. This allows the robot velocity and turn-rate to be recorded at a higher frequency between iSAM updates and computed as a batch of pose predictions, which results in reduced error in the linearised process model and reduced number of poses to be computed in the state vector.

6.1.4.2. Trajectory Planning for Line-feature SLAM

The strategy for the attractor, presented in Chapter 4, is modified in Chapter 5 for line-feature trajectory planning. Localising in a line-feature map is more difficult than in a point-feature map since the robot requires more features to localise but now generally fewer of them are available. Visibility also becomes an issue in line-feature mapping, as lines may occlude other lines. For exploration, two enabling items are introduced; an occupancy grid map to determine frontiers in the map and the distance transform algorithm to determine shortest path to the robot. For localisation and mapping, the reference point for the attractor is set to previous robot poses that have observed the line feature instead of using the position of the line feature itself. The placement of the attractor also considers the actual visibility in the structured environment.

For line-feature SLAM, it is discovered that an attractor is imperative for performing exploration. A fundamental issue in line-feature SLAM lies with the fact that the robot observes a single line feature for a long time such that there may be no information gain from moving to observe the same feature. By incorporating the attractor, the MPC strategy perceives the information gain from the attractor and is lured to explore.

6.1.4.3. Practical Implementation of Line-feature Mapping with iSAM

Line-feature mapping with iSAM has been demonstrated on a Pioneer2DX robot in Chapter 5, where an environment constructed from available furnishings was actively mapped in real-time. Although the computation time for trajectory planning and estimation was assumed to be instantaneous in simulations, consideration of computation time is necessary in practice. A modification to the planning algorithm, in which the robot pose is predicted based on the computation time for the previous loop, is implemented in this practical experiment.

The computation time required for each time step effectively determines the frequency at which the estimation is updated. Consequently, trajectory planning algorithms requiring shorter computer times stand to lead to more accurate maps. This work demonstrates that trajectory planning with MPC+Attractor is sufficient for a robot moving at low speeds to successfully map the environment. It therefore represents an applicable strategy for real-time mapping and localisation.

6.2. Directions for Future Work

The research documented herein provided insight into possible future directions as well as unforseen challenges. These are put forward in the following brief discussion.

6.2.1. Improvements to the Trajectory Planning

Trajectory planning for feature-based mapping remains a challenging problem despite the development of a number of solutions. In particular, the need to consider multiple objectives and constraints such as coverage, localisation, map accuracy, minimising computations, robot dynamics, cooperation, and obstacle avoidance renders the problem difficult to solve.

To improve localisation, it may be possible to consider areas that are likely to result in incorrect data association or good data association can be incorporated into the planning to assist the estimation accuracy of the map and localisation. Analysing the feature distribution or similarities in features observed may also assist in active sensing.

Although incorporating an attractor into the trajectory planning has improved coverage, complete coverage can still not be guaranteed. Moreover, obstacle avoidance is an issue especially when the planning horizon is short. As the robots are assumed in this work to stop and turn on the spot when there are no feasible control options, further work is need to extend the presented techniques to other vehicles such as UAVs.

Computational cost remains an issue for large dimensions of control inputs or states. The optimisation strategy is the principal factor contributing to the large computational requirements for long planning horizons. The attractor based strategy developed to incorporate long-term goals is currently a heuristic and more formal methods are desirable. Using a geometric series in the tree search may reduce computation as it allows for a finer search in the immediate steps in the future and a coarser branching of the tree search for steps further in the future. A complete reformulation of the trajectory planning problem may be possible where the rewards are approximated, or alternatively a method may be developed to summarise the long-term rewards into the objective function.

6.2.2. Trajectory Planning for Multi-agent SLAM

As seen in the multi-agent work in Chapter 3, cooperation amongst robots to perform active SLAM provides a naturally interesting topic. Since heterogenous sensing and mutual information from different robots can provide higher quality maps, adopting appropriate strategies for coverage and information sharing may improve the time required to perform the localisation and mapping task. A centralised approach is optimal but computationally expensive. Decentralised techniques will be necessary due to the cost of centralised approaches. However, there would be an increasing need to consider the inference between the robots as more of them are deployed for the task.

6.2.3. Trajectory Planning for Vision-based Estimation

Planning with visual information can further extend the possible applications of the MPC+Attractor technique to environments that are non-structured. When cameras are used in the estimation, a large amount of information can be used such as texture, colour, and brightness. Hence the map will contain more information and the mapping and localisation is not constrained to specific features in the environment such as points and lines. Environments such as disaster zones rarely have solid line features or poles/corners to extract point features. Therefore other features should be explored and the trajectory planning inturn needs to be adapted.

6.2.4. Improvements in Mapping and Map Management

Strategies for reducing poses in the iSAM state vector without causing inconsistencies remains an outstanding issue. Such issues need to be resolved for large-scale mapping as the computational cost increases with the number of poses in the state vector. An algorithm is thus formulated in Appendix C to reduce the number of poses in the iSAM estimation. While the removal of poses from the state vector increases the density of the information matrix, the approach is still more computationally efficient due to the small number of line features. The experiments presented in Chapter 5 demonstrate that the computation time remains low when the poses are removed. The use of the approach requires balancing between estimation accuracy and computational time, since more frequent removals will reduce the accuracy. At the limit, removing the poses at every step would be equivalent to the EKF state estimate. Selectively removing poses or waiting for opportunistic times where the robot covariance is small may be a possible method to maintain consistency in the estimation but requires further investigation.

A comparative study between various line-feature based SLAM algorithms would be of interest. Using infinite lines as features rather than line segments provides a more compact map as there may be many line segments to an infinite line; however a loss of locality may occur. The use of SPMap and other algorithms may have other benefits such as a single observation model unlike two presented in Chapter 5 which may cause singularities and redundant features. Knowledge of strengths and weaknesses of different approaches would be of benefit to the research community.

Mapping for efficient trajectory planning may require the maps to be dynamic. Doors may open at different angles for example, or in the case of mining, walls may be receding and effective map management strategies need to take this into account. Labelling features as good or poor for localisation may enable data association to ignore poor features so as the robot does not try to localise with a dynamic feature. Dynamic maps may also include the removal of incorrectly initialised features.

6.2.5. Trajectory Planning for Highly Constrained Environments or Fast Moving Vehicles

The greatest pressure on a planning algorithm is where there is little room for error. This is the case when a robot is moving in a confined space such as an automated wheelchair in a house or when the robot moves at fast speeds such as UAVs or autonomous cars in traffic. The trajectory planning is required to account for all uncertainties in the map and potential noise and errors in the control whilst maintaining low computation. Setting safety bounds on the trajectory may be possible however in a confined space it may prohibit the robot from moving if it is too large. The processing of information from various sensors requires significant computation. For effective planning, the trajectory planning needs to incorporate the most current information. This is a growing area as there is less space available for living and the growing need for faster processes and higher productivity for industrial and domestic applications in robotics.

Appendix A - iSAM Convergence Loop

The following describes the SAM estimation process. This algorithm is repeated until the solution converges. It is assumed that the robot may start stationary but once it starts moving it would not become stationary until it has finished.

The initial robot pose and all the estimated poses, all the observations, and all the control inputs: $\hat{\mathbf{x}}$, **z**, and **u** respectively are inputs to iSAM.

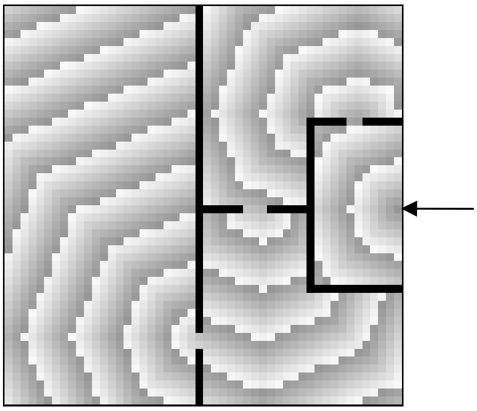
The predicted pose, relative pose, noise covariance of the relative pose $\hat{\mathbf{x}}_i$, $\hat{\mathbf{x}}_i^{rel}$ and $\mathbf{\Lambda}_i$,can be calculated following (5.42).

```
Set the convergence threshold $C_{Threshold}$=1$e$-3
Set convergence loop counter $\tau = 0$

Loop while $\|\delta\mathbf{x}\| > 1e-3$
    increment τ
    Loop $m = 1:i$ This loop calculates the Hessian for the poses and the odometry
                   prediction error up to time step $i$.
        If robot has started moving
            If first moving step Then
                Record the first step robot moves $s = m$
                If $\tau = 1$
                    Fill up the state vector with robot poses from the
                    first step
                    $\mathbf{\Theta}((m-s+1)\times 3-2:(m-s+1)\times 3,1) = \hat{\mathbf{x}}_m$
                End
                $\mathbf{x}_m^{rel}$ from process loop (5.40)
                $\hat{\mathbf{x}}_m^{rel} = \gamma(\hat{\mathbf{x}}_{m-1}, \hat{\mathbf{x}}_m)$ from (5.42)
                $\boldsymbol{\varepsilon}_m = \mathbf{x}_m^{rel} - \hat{\mathbf{x}}_m^{rel}$ The odometry prediction error
                $\boldsymbol{\varepsilon}_m^{new} = \left(\sqrt{\mathbf{\Lambda}_m}\right)^{-1} \times \boldsymbol{\varepsilon}_m$
            $\mathbf{G}_s^{s-1} = \left.\dfrac{\partial\gamma_m(\mathbf{x}_{m-1},\mathbf{x}_m)}{\partial\mathbf{x}_m}\right|_{(\hat{\mathbf{x}}_{s-1},\hat{\mathbf{x}}_s)}$
            $\mathbf{G}_s = \left(\sqrt{\mathbf{\Lambda}_m}\right)^{-1} \times \mathbf{G}_s^{s-1}$
            $\mathbf{A}(1:3,1:3) = \mathbf{G}_s$
        Else
            $\boldsymbol{\varepsilon}_m = \mathbf{x}_m^{relative} - \hat{\mathbf{x}}_m^{relative}$
            $\boldsymbol{\varepsilon}_m^{new} = \left(\sqrt{\mathbf{\Lambda}_m}\right)^{-1} \times \boldsymbol{\varepsilon}_m$
            $\mathbf{F}_{m-s}^{m-1-s} = \left.\dfrac{\partial\gamma_m(\mathbf{x}_{m-1},\mathbf{x}_m)}{\partial\mathbf{x}_{m-1}}\right|_{(\hat{\mathbf{x}}_{m-1-s},\hat{\mathbf{x}}_{m-s})}$
            $\mathbf{F}_{m-s} = \left(\sqrt{\mathbf{\Lambda}_m}\right)^{-1} \times \mathbf{F}_{m-s}^{m-s-1}$
            $\mathbf{G}_{m-s}^{m-1-s} = \left.\dfrac{\partial\gamma_m(\mathbf{x}_{m-1},\mathbf{x}_m)}{\partial\mathbf{x}_m}\right|_{(\hat{\mathbf{x}}_{m-1-s},\hat{\mathbf{x}}_{m-s})}$
            $\mathbf{G}_{m-s} = \left(\sqrt{\mathbf{\Lambda}_m}\right)^{-1} \times \mathbf{G}_{m-s}^{m-s-1}$
            $\mathbf{A} = \begin{bmatrix} \mathbf{G}_s & 0 & 0 & 0 \\ \mathbf{F}_{s+1} & \mathbf{G}_{s+1} & 0 & 0 \\ 0 & \mathbf{F}_{s+2} & \mathbf{G}_{s+2} & 0 \\ 0 & 0 & \mathbf{F}_{(m-s)} & \mathbf{G}_{(m-s)} \end{bmatrix}$
        End If
    End If
End Loop

Loop $m$ =1:$i$
    Loop $k$=1:K$_i$
        $\mathbf{\Psi}_{km}$ from Section 5.4.3
        If $\mathbf{\Psi}_{km}$>0 then associated
            $j = \mathbf{\Psi}_{mk}$
            $\boldsymbol{\lambda}_j^{local} = \begin{bmatrix} \alpha_j^{local} \\ d_j^{local} \end{bmatrix}$ from (5.8) or (5.11) depending on the
            location of the line
```

```
Else is a new feature
    $j = J + 1$
    $\boldsymbol{\lambda}_j = \begin{bmatrix} \alpha_k^{global} \\ d_k^{global} \end{bmatrix}$ from (5.7) or (5.10) depending on the location of the line
End If
$\boldsymbol{\mu}_k = \mathbf{z}_k - \boldsymbol{\lambda}_j^{local}$ The measurement prediction error.
$\boldsymbol{\mu}_k^{new} = \left(\boldsymbol{\Sigma}_k\right)^{-1} \times \boldsymbol{\mu}_k$
$\mathbf{H}_k^{m_k} = \left.\frac{\partial h_k(\mathbf{x}_{m_k}, \boldsymbol{\lambda}_{j_k})}{\partial \mathbf{x}_{m_k}}\right|_{(\hat{\mathbf{x}}_{m_k}, \hat{\boldsymbol{\lambda}}_{j_k})}$ using case 1 or 2 depending on the location of the line
$\mathbf{J}_k^{j_k} = \left.\frac{\partial h_k(\mathbf{x}_{m_k}, \boldsymbol{\lambda}_{j_k})}{\partial \boldsymbol{\lambda}_{m_k}}\right|_{(\hat{\mathbf{x}}_{m_k}, \hat{\boldsymbol{\lambda}}_{j_k})}$ using case 1 or 2 depending on the location of the line
$\mathbf{H}_m^k = \left(\sqrt{\boldsymbol{\Sigma}_k}\right)^{-1} \times \mathbf{H}_k^{m_k}$
$\mathbf{J}_{jk} = \left(\sqrt{\boldsymbol{\Sigma}_k}\right)^{-1} \times \mathbf{J}_k^{j_k}$
If at Stationary start position then don't need **H**
    $\mathbf{A} = \begin{bmatrix} \mathbf{G}_1 & & & & & \\ \mathbf{F}_2 & \mathbf{G}_2 & & & & \\ & \mathbf{F}_3 & \mathbf{G}_3 & & & \\ & & \vdots & & & \\ & & \mathbf{F}_M & \mathbf{G}_M & & \\ & & & & \mathbf{J}_{11} & \\ & & & & & \mathbf{J}_{22} \\ & & & & \mathbf{J}_{kj} & \end{bmatrix}$
Else
```

```
                $\mathbf{A} = \begin{bmatrix} \mathbf{G}_1 & & & & & \\ \mathbf{F}_2 & \mathbf{G}_2 & & & & \\ & \mathbf{F}_3 & \mathbf{G}_3 & & & \\ & & \vdots & & & \\ & & \mathbf{F}_M & \mathbf{G}_M & & \\ \mathbf{H}_1^1 & & & & \mathbf{J}_{11} & \\ & \mathbf{H}_2^2 & & & & \mathbf{J}_{22} \\ & & \mathbf{H}_M^j & & \mathbf{J}_{kj} & \end{bmatrix}$
            End If
        End Loop
    End Loop
    If   $\tau = 1$
        $\mathbf{\Theta} = \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\boldsymbol{\lambda}} \end{bmatrix}$
    End If

    $\mathbf{b} = \begin{bmatrix} \boldsymbol{\varepsilon}^{new} \\ \boldsymbol{\mu}^{new} \end{bmatrix}$
    $\mathbf{b} = \mathbf{A}^T \mathbf{b}$
    $\mathbf{R}_{\text{triangle}}$ = chol ($\mathbf{A}^T\mathbf{A}$) Cholesky factorisation
    $\delta\mathbf{x} = \mathbf{R}_{\text{triangle}}\backslash(\mathbf{R}_{\text{triangle}}^{\text{T}}\backslash\mathbf{b})$ ←Matlab backslash operator
    $\mathbf{\Theta} = \mathbf{\Theta} + \delta\mathbf{x}$
    If any distances are negative after update
        Add $\pi$ to the angle
        Multiply the distance by -1
    End If
    Update $\hat{\mathbf{x}}$ and $\hat{\boldsymbol{\lambda}}$ with new $\mathbf{\Theta}$
End Loop
```

Appendix B - Distance Transform



Source of Wavefronts

Figure B-6-1 Distance Transform

A popular method for path planning from a source to any destination in the map is the Distance Transform (Jarvis 1985, Lengyel 1990; Murray and Jennings 1997).

$\sqrt{2}$	1	$\sqrt{2}$
1	0	1
$\sqrt{2}$	1	$\sqrt{2}$

Figure B-6-2 Chamfer Distance Connectedness

The distance transform algorithm works by propagating wavefronts from the source which is the location of the robot. Each cell in the wavefront generated is designated a higher value than the previous. The Chamfer Distance Connectedness (Murray and Jennings 1997) is used as it is deemed to generate a more direct path. The shortest path can be determined by selecting any point on the map and then tracing the highest descent of wavefronts back to the source. Paths to multiple destinations can be planned from a single source without regeneration of the wavefronts. This is beneficial when computing paths to several destinations to compare costs.

Appendix C - Formulation of New SAM Algorithm for Computing the Estimates after Reducing Poses

The SAM state vector and its corresponding covariance matrix can be reduced to the EKF form as described in Section 5.5.4. After removing the unwanted poses, a new SAM algorithm is formulated, where the inputs include the current map, relative pose information and observations. This allows the continuation of the SAM estimation using the reduced state in subsequent time steps and by treating the map as an observation (instead of the real observations and control measurement as previously used) information is not lost.

To begin, prior to calculating $\hat{\mathbf{x}}_i$, the poses in the state vector can be reduced leaving the most current pose estimate $\hat{\mathbf{x}}_{i-1}$ before the update. First the unwanted poses and associated covariances are removed leaving $\hat{\boldsymbol{\Theta}}^{cut}$ and $\hat{\mathbf{P}}^{cut}$, i.e.

$$\hat{\boldsymbol{\Theta}}^{cut} = \begin{bmatrix} \hat{\mathbf{x}}_{i-1} \\ \hat{L} \end{bmatrix} \leftarrow \begin{bmatrix} \blacksquare \\ \blacksquare \\ \hat{\mathbf{x}}_{i-1} \\ \hat{L} \end{bmatrix} \quad \hat{\mathbf{P}}^{cut} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_{i-1}} & \mathbf{P}_{\mathbf{x}_{i-1}L} \\ \mathbf{P}_{L\mathbf{x}_{i-1}} & \mathbf{P}_L \end{bmatrix} \leftarrow \begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \mathbf{P}_{\mathbf{x}_{i-1}} & \mathbf{P}_{\mathbf{x}_{i-1}L} \\ \blacksquare & \blacksquare & \mathbf{P}_{L\mathbf{x}_{i-1}} & \mathbf{P}_L \end{bmatrix}. \tag{C.59}$$

To maintain the information from the removed poses, a new formulation for the least squares problem for SAM, similar to that described in Section 5.5, is required. This process is described as follows.

Step 1: Defining the Initial and Observed States

Let $\boldsymbol{\Theta}^{true}$ be the state

$$\boldsymbol{\Theta}^{true} = \begin{bmatrix} \mathbf{x}_{i-1} \\ L \end{bmatrix} \tag{C.60}$$

with the distribution $\boldsymbol{\Theta}^{true} \sim N(\hat{\boldsymbol{\Theta}}^{cut}, \hat{\mathbf{P}}^{cut})$ and the error of the estimate follows the distribution

$$\hat{\mathbf{\Theta}}^{cut} - \mathbf{\Theta}^{true} \sim N(0, \hat{\mathbf{P}}^{cut}) \tag{C.61}$$

Next, let $\mathbf{\Theta}^0$ be the initial state value and $\hat{\mathbf{\Theta}}^{cut}$ be the observed state and then $\hat{\mathbf{P}}^{cut}$ becomes the uncertainty of the observation.

The observation function is

$$h_{cut}(\mathbf{\Theta}^0) = \mathbf{\Theta}^0 \tag{C.62}$$

and the innovation **c** is then

$$\mathbf{c} = \hat{\mathbf{\Theta}}^{cut} - \mathbf{\Theta}^0 . \tag{C.63}$$

Step 2: Defining the new Least Squares Problem

Subsequently, as the robot moves and makes sensor observations, the new SAM problem now contains a new term, $\mathbf{\Delta}^*$, for the remaining state, $\hat{\mathbf{\Theta}}^{cut}$, after removing the poses, i.e.

$$\begin{aligned} \mathbf{\Theta}^* &\triangleq \arg\min_{\mathbf{\Theta}} \left\{ \sum_{m=\tau}^{i} \left\| \gamma_i(\mathbf{x}_{i-1}, \mathbf{x}_i) - \mathbf{x}_i^{rel} \right\|_{\Lambda_i}^2 + \sum_{k=k_\tau}^{K} \left\| h_k(\mathbf{x}_{mk}, \boldsymbol{\lambda}_{jk}) - \mathbf{z}_k \right\|_{\Sigma_k}^2 + \mathbf{\Delta}^* \right\} \\ \mathbf{\Delta}^* &= \sum_{M=1}^{W} \left\| h_{cut}(\mathbf{\Theta}_M^0) - \hat{\mathbf{\Theta}}_M^{cut} \right\|_{\hat{\mathbf{P}}_M^{cut}}^2 \end{aligned} \tag{C.64}$$

where W is the number of elements in $\hat{\mathbf{\Theta}}^{cut}$, τ is the time step where the poses were removed and k_τ is the index of the first observation observed at time τ. If the formulation for reducing poses is computed at the current time step prior to updating the state vector then $\tau = i - 1$. The new term $\mathbf{\Delta}^*$ is formulated such that it is similar to the existing terms in the least squares SAM problem and can thus be solved in the same manner by solving $\left\| \mathbf{A}\delta - \mathbf{b} \right\|$.

Step 3: Approximating the Error of the Estimation

Using the initial state value $\mathbf{\Theta}^0$, the error of the estimation is approximated

$$\mathbf{\Theta}^{true} - \hat{\mathbf{\Theta}}^{cut} \approx \mathbf{\Theta}^0 + (\mathbf{\Theta}^{true} - \mathbf{\Theta}^0) - \hat{\mathbf{\Theta}}^{cut} \tag{C.65}$$

where

$$(\mathbf{\Theta}^{true} - \mathbf{\Theta}^0) = \delta\mathbf{\Theta}^{true} = \begin{bmatrix} \delta\mathbf{x}_{i-1} \\ \delta L \end{bmatrix} \tag{C.66}$$

thus

$$\begin{aligned} \mathbf{\Theta}^{true} - \hat{\mathbf{\Theta}}^{cut} &= \mathbf{\Theta}^0 + \begin{bmatrix} \delta\mathbf{x}_{i-1} \\ \delta L \end{bmatrix} - \hat{\mathbf{\Theta}}^{cut} \\ &= \begin{bmatrix} \delta\mathbf{x}_{i-1} \\ \delta L \end{bmatrix} - (\hat{\mathbf{\Theta}}^{cut} - \mathbf{\Theta}^0) \end{aligned} \tag{C.67}$$

Step 4: Defining the new Linearised Least Squares Problem

The new linearised least squares problem now becomes

$$\delta^* = \arg\min_{\delta} \left\{ \sum_{m=1}^{i} \left\| \mathbf{F}_m^{m-1} \delta\mathbf{x}_{m-1} + \mathbf{G}_m^m \delta\mathbf{x}_m - \boldsymbol{\varepsilon}_m \right\|_{\mathbf{\Lambda}_m}^2 + \sum_{k=k_\tau}^{K} \left\| \mathbf{H}_k^m \delta\mathbf{x}_{mk} + \mathbf{J}_k^j \delta\boldsymbol{\lambda}_{jk} - \boldsymbol{\mu}_k \right\|_{\mathbf{\Sigma}_k}^2 + \mathbf{\Delta} \right\} \tag{C.68}$$

where $\mathbf{\Delta}$ is

$$
\begin{aligned}
\mathbf{\Delta} &= \left\| \begin{bmatrix} \delta\mathbf{x}_{i-1} \\ \delta L \end{bmatrix} - (\hat{\mathbf{\Theta}}^{cut} - \mathbf{\Theta}^0) \right\|^2_{\hat{\mathbf{P}}^{cut}} \\
&= \left\| \begin{bmatrix} \delta\mathbf{x}_{i-1} \\ \delta L \end{bmatrix} - \begin{bmatrix} \mathbf{c}_{\mathbf{x}} \\ \mathbf{c}_L \end{bmatrix} \right\|^2_{\hat{\mathbf{P}}^{cut}} \\
&= \left\| \begin{bmatrix} \delta\mathbf{x}_{i-1} - \mathbf{c}_{\mathbf{x}} \\ \delta L - \mathbf{c}_L \end{bmatrix} \right\|^2_{\hat{\mathbf{P}}^{cut}} \\
&= \begin{bmatrix} \delta\mathbf{x}_{i-1} - \mathbf{c}_{\mathbf{x}} \\ \delta L - \mathbf{c}_L \end{bmatrix}^{\mathrm{T}} \hat{\mathbf{P}}^{-1}_{cut} \begin{bmatrix} \delta\mathbf{x}_{i-1} - \mathbf{c}_{\mathbf{x}} \\ \delta L - \mathbf{c}_L \end{bmatrix} \\
&= \left\| \hat{\mathbf{P}}^{-\mathrm{T}/2}_{cut} \begin{bmatrix} \delta\mathbf{x}_{i-1} - \mathbf{c}_{\mathbf{x}} \\ \delta L - \mathbf{c}_L \end{bmatrix} \right\|^2_2 \\
&= \left\| \hat{\mathbf{P}}^{-\mathrm{T}/2}_{cut} \begin{bmatrix} \delta\mathbf{x}_{i-1} \\ \delta L \end{bmatrix} - \hat{\mathbf{P}}^{-\mathrm{T}/2}_{cut} \begin{bmatrix} \mathbf{c}_{\mathbf{x}} \\ \mathbf{c}_L \end{bmatrix} \right\|^2_2
\end{aligned}
\quad . \qquad \text{(C.69)}
$$

Step 5: Solving the new Linearised SAM Problem

To solve for the linearised problem $\|\mathbf{A}\delta - \mathbf{b}\|$, let $\mathbf{c}^{new}$ be defined as the innovations for the current state $\hat{\mathbf{\Theta}}^{cut}$ such that

$$
\mathbf{c}^{new} = \hat{\mathbf{P}}^{-\mathrm{T}/2}_{cut} \begin{bmatrix} \mathbf{c}_{\mathbf{x}} \\ \mathbf{c}_L \end{bmatrix} = \begin{bmatrix} \mathbf{c}^{new}_{\mathbf{x}} \\ \mathbf{c}^{new}_L \end{bmatrix}. \qquad \text{(C.70)}
$$

Using the new process innovation $\boldsymbol{\varepsilon}^{new}$ and observation innovation $\boldsymbol{\mu}^{new}$ from the next time step, $\mathbf{b}$ can be constructed as

$$
\mathbf{b} = \begin{bmatrix} \mathbf{c}^{new}_{\mathbf{x}} & \boldsymbol{\varepsilon}^{new} & \mathbf{c}^{new}_L & \boldsymbol{\mu}^{new} \end{bmatrix}^{\mathrm{T}}. \qquad \text{(C.71)}
$$

By separating the terms, $\mathbf{\Delta}$ can be expressed as

$$
\mathbf{\Delta} = \sum_{M=1}^{W} \left\| \mathbf{E}^i_M \delta\mathbf{x}_i + \mathbf{E}^j_M \delta\boldsymbol{\lambda}_1 \ldots + \mathbf{E}^j_M \delta\boldsymbol{\lambda}_J - \mathbf{c}_M \right\|^2_{\mathbf{P}}, \qquad \text{(C.72)}
$$

where **E** is an identity matrix and **P** can be dropped by the following property

$$\mathbf{E}_M^{new} = \left(\sqrt{\hat{\mathbf{P}}_M^{cut}}\right)^{-1} \times \mathbf{E}_M = \hat{\mathbf{P}}_{cut}^{-\mathrm{T}/2} \tag{C.73}$$

All the parameters required to construct $\mathbf{b}$ have now been defined for the equation $\|\mathbf{A}\delta - \mathbf{b}\|$. Next $\mathbf{A}$ will need to be calculated to solve for δ. Suppose the example state consists of one pose followed by two features, i.e.

$$\mathbf{\Theta} = \begin{bmatrix} \mathbf{x}_i \\ L \end{bmatrix} = \begin{bmatrix} \mathbf{x}_3 \\ \boldsymbol{\lambda}^1 \\ \boldsymbol{\lambda}^2 \end{bmatrix}. \tag{C.74}$$

Then the covariance would be

$$\hat{\mathbf{P}}^{-\mathrm{T}/2} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix}. \tag{C.75}$$

The least squares problem $\|\mathbf{A}\delta - \mathbf{b}\|$ can be expressed as

$$\begin{aligned} \mathbf{\Delta} &= \left\| \begin{bmatrix} \mathbf{P}_{11}\delta\mathbf{x}_{i-1} + \mathbf{P}_{12}\delta\boldsymbol{\lambda}_1 + \mathbf{P}_{13}\delta\boldsymbol{\lambda}_2 \\ \mathbf{P}_{21}\delta\mathbf{x}_{i-1} + \mathbf{P}_{22}\delta\boldsymbol{\lambda}_1 + \mathbf{P}_{23}\delta\boldsymbol{\lambda}_2 \\ \mathbf{P}_{31}\delta\mathbf{x}_{i-1} + \mathbf{P}_{32}\delta\boldsymbol{\lambda}_1 + \mathbf{P}_{33}\delta\boldsymbol{\lambda}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{c}_{\mathbf{x}_{i-1}} \\ \mathbf{c}_{\boldsymbol{\lambda}_1} \\ \mathbf{c}_{\boldsymbol{\lambda}_2} \end{bmatrix} \right\|_2^2 \\ &= \left\| \begin{bmatrix} \mathbf{P}_{11}\delta\mathbf{x}_{i-1} + \mathbf{P}_{12}\delta\boldsymbol{\lambda}_1 + \mathbf{P}_{13}\delta\boldsymbol{\lambda}_2 - \mathbf{c}_{\mathbf{x}_{i-1}} \\ \mathbf{P}_{21}\delta\mathbf{x}_{i-1} + \mathbf{P}_{22}\delta\boldsymbol{\lambda}_1 + \mathbf{P}_{23}\delta\boldsymbol{\lambda}_2 - \mathbf{c}_{\boldsymbol{\lambda}_1} \\ \mathbf{P}_{31}\delta\mathbf{x}_{i-1} + \mathbf{P}_{32}\delta\boldsymbol{\lambda}_1 + \mathbf{P}_{33}\delta\boldsymbol{\lambda}_2 - \mathbf{c}_{\boldsymbol{\lambda}_2} \end{bmatrix} \right\|_2^2 \end{aligned}. \tag{C.76}$$

Suppose now in the next time step there is one more pose $\mathbf{x}_4$ and one new and one old feature observed from this pose. The new state estimate would be

$$\mathbf{\Theta} = \begin{bmatrix} \mathbf{x}_3 & \mathbf{x}_4 & \boldsymbol{\lambda}^1 & \boldsymbol{\lambda}^2 & \boldsymbol{\lambda}^3 \end{bmatrix}^{\mathrm{T}} \tag{C.77}$$

and the new innovation vector would be

$$\mathbf{b} = \begin{bmatrix} \mathbf{c}_{\mathbf{x}_3}^{new} & \boldsymbol{\varepsilon}_4^{new} & \mathbf{c}_{\boldsymbol{\lambda}_1}^{new} & \mathbf{c}_{\boldsymbol{\lambda}_2}^{new} & \boldsymbol{\mu}_z^{new} \end{bmatrix}^{\mathrm{T}}. \tag{C.78}$$

Finally, the Hessian matrix **A** in the equation $\left\| \mathbf{A}\delta - \mathbf{b} \right\|$ can be constructed as follows

$$\mathbf{A} = \begin{bmatrix} \mathbf{P}_{11} & & \mathbf{P}_{12} & \mathbf{P}_{13} & \\ \mathbf{F}_4^3 & \mathbf{G}_4^4 & & & \\ \mathbf{P}_{21} & & \mathbf{P}_{22} & \mathbf{P}_{23} & \\ \mathbf{P}_{31} & & \mathbf{P}_{32} & \mathbf{P}_{33} & \\ & \mathbf{H}_4^z & & & \mathbf{J}_4^3 \\ & \mathbf{H}_4^z & & \mathbf{J}_4^2 & \end{bmatrix}. \tag{C.79}$$

Bibliography

Alempijevic A. 2004, "High-Speed Feature Extraction in Sensor Coordinates for Laser Rangefinders," *Australasian Conference on Robotics and Automation (ACRA)*, Canberra, Australia.

Allgower, F. and Zheng, A. 2000, *Nonlinear Model Predictive Control*, Boston: Birkhauser.

Bellingham, J., Richards, A. and How, J. P. 2002, "Receding Horizon Control of Autonomous Aerial Vehicles", *Proceedings of the American Control Conference*, Anchorage, AK, pp 3741-3746

Bitmead, R., Gevers, M. and Wetz, V. 1990, *Adaptive Optimal Control: The Thinking Man's GPC*, Prentice Hall

Bourgault, F., Furukawa, T. and Durrant-Whyte, H. F. 2004, "Decentralized Bayesian Negotiation for Cooperative Search," *presented at IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan.

Brooks, A., Makarenko, A., Williams, S. and Durrant-Whyte, H. 2005, "Planning in continuous state spaces with parametric POMDPs", *IJCAI Workshop Reasoning with Uncertainty in Robotics*, Edinburgh, Scotland, 30 July 2005.

Bryson, M., Kim, J. and Sukkarieh, S. 2005, "Information and Observability Metrics of Inertial SLAM for On-line Path-planning on an Aerial Vehicle," *IEEE International Conference on Robotics and Automation*, 18-22 April 2005, SLAM Workshop Barcelona, Spain.

Bryson, M. T. and Sukkarieh, S. 2007, "Building a Robust Implementation of Bearing-Only Inertial SLAM for a UAV" *In Journal of Field Robotics*, Special issue on SLAM in the field, vol. 24, no. 1-2, Feb, 2007, pp. 113-143

Castellanos, J. A., Montiel, J. M. M., Neira, J. and Tardos, J. D. 1999, "The SPmap: a probabilistic framework for simultaneous localization and map building," *Robotics and Automation, IEEE Transactions*, vol. 15, pp. 948-952.

Davison, A. 2003 “Real-time Simultaneous Localisation and Mapping with a Single Camera,” *in Proc. International Conference on Computer Vision*, Nice, October 2003.

Dellaert, F. and Kaess, M. 2006, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," *The International Journal of Robotics Research*, vol. 25, pp. 1181-1203.

Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H. F. and Csorba, M. 2001, “A solution to the simultaneous localization and map building (SLAM) problem” *IEEE Transactions on Robotics and Automation*, June 2001, vol. 17, no. 3, pp. 229-241

Feder, H., Leonard, J. and Smith, C. 1999, “Adaptive mobile robot navigation and mapping”, *International Journal of Robotics Research*, vol. 18, pp. 650-668.

Frew, E. W. 2005, "Receding Time Horizon Control Using Random Search for UAV Navigation with Passive, Non-cooperative Sensing," *presented at AIAA Guidance, Navigation, and Control Conference*, CA.

Foka, A. and Trahanias, P. 2005, “Real-time hierarchical POMDPs for autonomous robot navigation”, *IJCAI Workshop Reasoning with Undertainty in Robotics*, Edinburgh, Scotland, 30 July 2005.

Furukawa, T., Dissanayake, G. and Durrant-Whyte, H. F. 2003, "Time-optimal Cooperative Control of Multiple Robot Vehicles," *presented at International Conference on Robotics and Automation*, Taipei.

Garulli, A., Giannitrapani, A., Rossi, A. and Vicino, A. 2005, "Mobile robot SLAM for line-based environment representation," *44th IEEE Conference on Decision and Control, and 2005 European Control Conference*. CDC-ECC '05.

Golub, G. H. and Van Loan, C. F. 1996, *Matrix Computations*, Johns Hopkins University Press, Baltimore, third edition.

Gonzalez-Banos, H. H. and Latombe, J. C. 2002, "Navigation Strategies for Exploring Indoor Environments," *The International Journal of Robotics Research*, vol. 21, pp. 829-848.

Grocholsky, B. 2006, "Information-Theoretic Control of Multiple Sensor Platforms," The University of Sydney, PhD Thesis.

Gu, D. B. and Hu, H. S. 2005, "A stabilizing receding horizon regulator for nonholonomic mobile robots", *IEEE Trans. on Robotics*, vol. 21, no. 5, pp. 1022-1028

Helton, J. W. and James, M. R. 1999, "Extending H^∞ Control to Nonlinear Systems: Control of Systems to Achieve Performance Objectives", *Advances in Design and Control*, SIAM.

Huang, S. and James, M. R. 2003, "l^∞-bounded robustness for nonlinear systems: analysis and synthesis", *IEEE Trans. on Automatic Control*, vol. 48, no.11, pp. 1875-1891

Huang, S., Kwok, N. M., Dissanayake, G., Ha, Q. P. and Fang, G. 2005, "Multi-Step Look-Ahead Trajectory Planning in SLAM: Possibility and Necessity," *presented at IEEE International Conference on Robotics and Automation*, Barcelona, Spain.

Jarvis, R. A. 1985, "Collision-free trajectory planning using the distance transforms", *Mechanical Engineering Trans. of the Institution of Engineers*, ME10(3), September 1985.

Kaelbling, L. P., Littman, M. L. and Cassandra, A. R. 1998, "Planning and acting in partially observable stochastic domains", *Artificial Intelligence*, vol. 101, pp. 99-134

Kim, J., Ong, S., Nettleton, E. and Sukkarieh, S. 2004, "Decentralised approach to unmanned aerial vehicle navigation: without the use of GPS and preloaded map," *Journal of Aerospace Engineering*, vol. 218, pp. 399-416.

Kim, J. H. and Sukkarieh, S. 2004, “Improving the Real-time Efficiency of Inertial SLAM and Understanding its Observability” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai.

Kuwata, Y., Richards, A., Schouwenaars, T. and How, J. P. 2006, “Decentralized Robust Receding Horizon Control for Multi-vehicle Guidance” *Proceedings of the 2006 American Control Conference*, Minneapolis, Minnesota, USA, June 14-16, 2006, pp. 2047-2052

Kwok, N. M. and Dissanayake, G. 2004, “An efficient multiple hypothesis filter for bearing-only SLAM” *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004).* 28 Sept.-2 Oct. 2004, vol 1, pp. 736- 741

Lengyel, J., Reichert, M., Donald, B. and Greenberg, D. 1990, “Real--time robot motion planning using rasterizing computer graphics hardware”, *In Proc. of SIGGRAPH*, Dallas, Texas, August 1990, pp. 327-335

Li, X., Cheung, W. K. and Liu, J. 2005, “Towards solving large-scale POMDP problems via spatio-temporal belief state clustering”, *IJCAI Workshop Reasoning with Uncertainty in Robotics*, Edinburgh, Scotland, 30 July 2005.

Li, Y. F. and Liu, Z. G. 2005, “Information entropy-based viewpoint planning for 3-D object recontruction”, *IEEE Trans. on Robotics*, Vol.21, No.3, pp. 324-337, June 2005

Makarenko, A. A., Williams, S. B., Bourgault, F. and Durrant-Whyte, H. F. 2002, "An experiment in integrated exploration," *presented at IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Maybeck, P. 1979, *Stochastic Models, Estimation, and Control*, vol. 1: Academic, New York.

Meger, D., Rekleitis, I. and Dudek, G. 2006, “Simultaneous Planning, Localization, and Mapping in a Camera Sensor Network”, Book Chapter, *Distributed Autonomous Robotic Systems 7*, Ed. M. Gini and R. Voyles, Springer Japan, ISBN 978-4-431-35878-7, pp 155-164.

Mettler, B. and Toupet, O. 2005, “Receding Horizon Trajecotry Planning with an Environment-Based Cost-to-go Function”, *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, Seville, Spain, December 12-15 2005, pp. 4071-4076

Murray, D. and Jennings, C. 1997 “Stereo vision based mapping and navigation for mobile robots,” *Proceedings IEEE International Conference on Robotics and Automation*, 20-25 April 1997, vol. 2, pp. 1694 - 1699

Neira, J. and Tardós, J. D. 2001, “Data Association in Stochastic Mapping Using the Joint Compatibility Test” IEEE Transactions on Robotics and Automation, vol. 17, no. 6, Dec 2001, pp. 890 – 897

Nilsson, N. J. 1980. *Principles of Artificial Intelligence*, Tioga Publishing Company, pp. 72-88.

Nguyen, V., Martinelli, A., Tomatis, N. and Siegwart, R. 2005, "A comparison of line extraction algorithms using two-dimensional laser rangefinder for indoor mobile robotics," *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005).*

Ong, L. L., Ridley, M., Upcroft, B., Kumar, S., Bailey, T., Sukkarieh, S. and Durrant-Whyte, H. 2005, "A Comparison of Probabilistic Representations for Decentralised Data Fusion," *presented at 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference.*

Oshman, Y. and Davidson, P. 1999, "Optimization of Observer Trajectories for Bearings-only Target Localization," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 35, pp. 892-902.

Passerieux, J. M. and Cappel, D. V. 1998, "Optimal Observer Maneuver for Bearings-only Tracking," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 34, pp. 777-788.

Passino, K., Polycarpou, M., Jacques, D., Pachter, M., Liu, Y., Yang, Y., Flint, M. and Baum, M. 2002, "Cooperative Control for Autonomous Air Vehicles," *Cooperative Control and Optimization*, vol. Chapter 12. Netherlands.

Pfister, S. T. and Burdick, J. W. 2006, "Multi-scale point and line range data algorithms for mapping and localization," *Proceedings 2006 IEEE International Conference on Robotics and Automation*, 2006. ICRA 2006.

Ristic, B. and Arulampalam, M. S. 2003, "Tracking a Manoeuvring Target using Angle-only Measurements: Algorithms and Performance," *Signal Processing*, vol. 83, pp. 1223-1238.

Rodriguez-Losada, D., Matia, F., Jimenez, A. and Galan, R. 2006, “Consistency improvement for SLAM-EKF for indoor environments”. *Proceedings of the IEEE Conference on Robotics and Automation*, Orlando, Florida, May 2006, pp. 418-423.

Rodriguez-Losada, D., Matia, F., Jimenez, A. and Galan, R. 2006a, “Local map fusion for real-time indoor simultaneous localization and mapping” *Journal of Field Robotics*, pp. 291-309, Published Online: 24 Apr 2006

Rosenblatt, J. K. 1999, "Optimal selection of uncertain actions by maximizing expected utility," *presented at IEEE International Symposium on Computational Intelligence in Robotics and Automation*.

Roy, N. and Earnest, C. 2006, “Dynamic Action for Information Gain Maximization in Search and Exploration” *Proceedings of the American Control Conference (ACC 2006)*, Minneapolis, 14-16 June, 6 pages.

Seywald, H. 1994 “Trajectory optimization based on differential inclusion”, *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 3, May-June 1994, pp. 480-487.

Shim, D. H., Kim, H. J. and Sastry, S. 2003, "Decentralized nonlinear model predictive control of multiple flying robots," *presented at Proc. 42nd IEEE Conference on Decision and Control*, Hawaii USA.

Sim, R. 2005, "Stable Exploration for Bearings-only SLAM," *presented at IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain.

Sim, R. 2005a, "Stabilizing information-driven exploration for bearings-only SLAM using range gating," *presented at IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Sim, R. and Roy, P. N. 2005, "Global A-Optimal Robot Exploration in SLAM," *presented at IEEE International Conference on Robotics and Automation*, Barcelona, Spain.

Stachniss, C., Hahnel, D. and Burgard, W. 2004, "Exploration with active loop-closing for FastSLAM," *presented at IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Stentz, A. 1995, “The focussed D* algorithm for realtime replanning”, *In Proceedings of the Joint Conference on Artificial Intelligence.*

Stroupe, A. W. and Balch, T. 2005, “Value-based action selection for observation with robot teams using probabilistic techniques”, *Robotics and Autonomous Systems*, vol. 50, pp. 85-97.

Tang, Z. and Özgüner, Ü. 2005, “Motion Planning for Multitarget Surveillance with Mobile Sensor Agents”, *IEEE Transactions on Robotics*. vol. 21, no. 5, Oct 2005, pp 898-908

Thrun, S., Liu, Y., Ng, A. Y., Ghahramani, Z. and Durrant-Whyte, H. 2004, "Simultaneous Localization and Mapping with Sparse Extended Information Filters," *International Journal of Robotics Research*, vol. 23, pp. 693-716.

Tierno, J. E. 2001, "Distributed Autonomous Control of Concurrent Combat Tasks," *presented at American Control Conference*.

Wesselowski, K. and Fierro, R. 2003, "A Dual-mode Model Predictive Controller for Robot Formations," *42nd IEEE Conference on Decision and Control*, vol. 4, pp. 3615-3620

Yuen, D. C. K. and MacDonald, B. A. 2003, "Line-based SMC SLAM Method in Environment with Polygonal Obstacles," *presented at Australasian Conference on Robotics & Automation (ACRA)*, Brisbane, Australia.

Zhang, N., Li, M. and Hong, B. 2006, “Active Mobile Robot Simultaneous Localization and Mapping”, *IEEE International Conference on Robotics and Biomimetics*, Dec. 2006, pp. 1676-1681

Zheng, A. 1997, “A computationally efficient nonlinear model predictive control algorithm”, *In Proceedings of American Control Conference*, Albuquerque, NM