Efficient Exploration to map unknown environments through autonomous robots

Submitted By:

Harsh Goel

Department of

Mechanical Engineering

In fulfilment of the

requirements for the Degree of

Bachelor of Engineering

National University of Singapore

Session 2019/2020

SUMMARY

For most search, surveillance and rescue applications in field robotics, robots are expected to autonomously explore an unknown environment. The process of exploration can be sped up if the robot knows where it's expected to gain new information. Thus, this project proposes a method to estimate the expected information gain at a given location in a partial map of the environment.

Given this distribution of information, this work further proposes a planning strategy that evaluates good informative paths for map building purposes. We implement two path sampling strategies which are optimised using the Cross-Entropy method. During path optimisation, we compare the time averaged statistics of the robot's sensor footprint along a path to the information distribution map to assess information gain. We demonstrate this algorithm for a single turtlebot on various partial maps and assess its merits.

With multiple robots sharing their perceptions to build a common map of the environment, exploration can be speeded up. In addition, decentralising the path planning process for map building makes the robot team robust to single agent failure. We identify that inferring of the path plan of neighbouring robots is crucial to decentralising multi agent trajectory planning for exploration and mapping and thus, propose a path estimation method based on particle filters.

ACKNOWLEDGEMENTS

The author would like to express his sincere gratitude to the Project Supervisors Dr. Marcelo H. Ang Jr and Dr. Guillaume Sartoretti for their guidance given throughout the duration of the dissertation. The author wishes to extend his gratitude towards the staff and lab assistants of the Controls and Mechatronics Laboratory for their assistance in carrying out the computer experiments for the dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1 Introduction

1.1 Background and Objective

Autonomous mobile robots have evolved over the past 2 decades and have the potential to change the way humans live and work. These robots can be made to work autonomously in a diverse set of environments ranging from offices and households to disaster sites and tough industrial environments. For autonomous robots to be fully functional, robots would have to maintain an internal description or a map of the environment to safely navigate through while carrying out the tasks they are designed for.

The process of map building of an environment to extract relevant features is an integral process for many robotics applications. Conventionally, these maps are represented as discrete occupancy grids composed of cells that represent the presence of an obstacle at a given location in the environment. In most of these applications, an autonomous robot would have no prior information of the environment it is placed within. Therefore, such a robot is initially tasked to explore the environment to map features such as obstacles, walls and empty space.

Thus, the primary objective of this project is to propose a new exploration strategy for building a map of an unknown environment. In the following section, we provide a summary and review some of the exploration algorithms in literature.

1.2 Literature Review

Over the past two decades various methods have been proposed for efficient exploration of the environment using a single robot to build such a map of the environment [1]. Most of these evaluate regions that transition from free space to unknown and unexplored space [1]. These regions are termed as frontiers and a robot is tasked to explore such frontiers to gain more information from the environment to further build the map. These methods employ a greedy search-based strategy where these frontiers are assigned a cost based on proximity [1]. Although quite simple and effective, the closest frontier approach does not account for new information that the robot is expected to gain at a frontier. Thus, exploration strategies that greedily explored frontiers that maximised information gain were proposed [2]. Information gain is defined as the decrease in uncertainty between successive measurements and the objective of this exploration strategy is to decrease the entropy of the map over time. However, this exploration strategy would result in repetitive and wasteful traversal of the robot to reach such frontiers with highest information gain [3].

Thus, for efficient exploration of the environment an autonomous robot is required to maximise information gain while minimising the travelling distance. Exploration strategies using Partially Observed Markov Decision Process (POMDP) [4] were develop for the robot exploration problem [5][6]. In this framework the selection of a frontier for exploration by the robot is modelled as an action, and the planning algorithm evaluates a set of actions that maximise a reward function. This reward function accounts for expected information gain, total

travelling length, total time, and total number of scans taken [7]. The robot learns a policy for exploration through reinforcement learning to select actions that maximises this reward function over a time horizon [8] [9]. Moreover, these methods also account for imperfections in sensor measurements and uncertainty in the robot's current observations. However, these approaches are computationally intractable, where for realistic exploration applications the problem of evaluating an optimal policy takes up to minutes or hours to solve [7]. It is crucial for a robot to evaluate a path to explore the environment in the least amount of time possible as we expect the robots to re-plan their paths whenever new information is acquired from a frontier which further expands the map. Thus, based on the literature review three crucial research questions are identified with respect to the robot exploration.

1) How do we represent the amount of expected new information gain in the map of the environment with respect to the robot's position?
2) What is a good path for a robot to take to maximise the amount of information it gains from the environment while minimising travelling distance?
3) How to compute this path in an acceptable amount of time?

Furthermore, multiple robots can speed up the exploration where these robots can share their perceptions and build a common map of the environment. This global map can be constructed in a centralized way or independently in each robot in a distributed and more robust manner. Frontier-based exploration strategies have been extended to coordinate multiple robots for exploration [10]. Assignment of regions for exploration by Voronoi partitioning of the known map [11] and optimal

frontier assignment using Hungarian methods [11] to robots in a team were two such initial centralised planning approaches. To increase system robustness decentralised approaches such as decision theoretic based task distribution frameworks using market-based bidding models [12] [13] were introduced. However, these approaches don't account for maximising information gain while minimising total travelling distance for the robots. Furthermore, these planning algorithms are typically restricted to short time horizons. In order to plan good paths for exploration using multiple robots, stochastic control methods using potential fields [14] and sampling-based motion planning approach using multi agent probabilistic roadmaps [15] were introduced. Furthermore, Singh et al [16] propose a method to evaluate optimal information gathering paths using multiple robots for sensor-based coverage purposes. However, these proposed algorithms rely on a centralised system to plan paths for individual robots which is not robust to failure. Thus, an open research question is identified with respect to the optimal robot exploration problem using multiple robots.

4) How to decentralise path planning to coordinate robots in a team to explore the environment?

We address these research questions in this project. The scope and organisation of the project is presented in the next section.

1.3 Scope and Organisation

The report is divided across 4 chapters. Chapter 2 explains the simulation environment and robot architecture used and can be skipped if the reader is familiar with designing turtlebot architectures and simulations on ROS and Gazebo. Chapter 3 and Chapter 4 present the bulk of the work done in this project. Chapter 3 describes a method to characterise the amount of new information that can be gained from the environment. Chapter 4 introduces the motion planning algorithm that maximises information gain.

In chapter 2 we begin to describe a simulation architecture developed on ROS and Gazebo for the robot used for the exploration problem. A simple differential drive turtlebot with a 3D camera that has a 120-degree horizontal field of view and range of 5 m is used for mapping. The chapter describes the robot's architecture developed in ROS to test the motion planning algorithms described in Chapter 4.

Chapter 3 describes a method used to evaluate information dense regions from the partial map of the environment. As a robot can retrieve certain amount of new information due the range of its sensors, we attempt to represent information gain that a robot can obtain with respect to its position on a known partial map of the environment. Naturally, frontiers to unexplored space are information dense as compared to other locations in the map. We use these frontiers to capture to obtain an information density map to evaluate good paths for a robot to execute.

We present a new sampling-based motion planning scheme in Chapter 4 that formulates a path for a single robot which maximises information gain from the surroundings. The planning algorithm accounts for the robot's sensor footprint

along a planned trajectory and incorporates the robot's physical footprint in the environment to avoid obstacles. Furthermore, a mathematical framework is also proposed that can be used to completely decentralise motion planning for exploration.

Chapter 5 presents a summary of the work conducted and outlines the scope for future work.

1.4 Contributions

The contributions of the project can be summarised as follows:

- A multi-robot simulator on ROS developed for easy deployment of algorithms to real physical robots
- Description of a method to represent maximum amount of new information that can be gained from the partial map of the environment as a function of the robot's position in the environment.
- Description of a motion planning algorithm that plans a good path to gain maximum amount of new information from the environment.
- A theoretical framework to decentralise the motion planning scheme to coordinate multiple robots for exploration and map building purposes.

Chapter 2 Overview of the Simulator and Software

2.1 Introduction

This chapter will briefly describe the simulation software and architecture developed on Gazebo using ROS as the middleware to test the developed motion planning algorithm in Chapter 4. A turtlebot is used as the test robot for the simulation study and an architecture for the robot was developed on ROS for the robot to perform navigation and mapping related tasks. This chapter would also briefly describe this architecture. Furthermore, we would give an overall picture on the workflow of the simulation software so that the reader could use it for future work. We show the results of testing the robot's architecture to develop a map of the environment and address a few limitations of the simulator.

-

2.2 Turtlebot Architecture Description

A turtlebot is a low cost and easy to develop robotic platform (shown in Fig 2.1) with open source software which allows easy deployment of algorithms developed for various autonomous mobile robotics tasks. The turtlebot operates on Robot Operating System (ROS) middleware. The turtlebot is simulated on the Gazebo environment by providing a URDF(Universal Robot Description File) format of the robot to the Gazebo package in ROS.

In this project, the turtlebot in the simulation uses a Kinect sensor with a field of view of 120 degree and a range of 5 m. The point cloud information is converted a

laser scan using ROS's depthimage_to_laserscan package for Simultaneous Localisation and Mapping (SLAM). Optionally, the simulation provides a URDF format file that simulates Hokuyo's laser scanner to provide laser scans directly for SLAM.

Fig 2.1 Turtlebot

Mapping of the environment is performed using Gmapping which produces an occupancy grid map whose values range from 0 – 100 to indicate the probability that a certain grid point is an obstacle. Thus, 0 indicates free space and 100 indicates an obstacle. Unknown and unexplored spaces are indicated as -1. Moreover, the Gmapping package was modified to provide the Shannon's entropy [17] of each cell of the occupancy grid map for evaluating information content each cell could provide. For unknown spaces which are assigned as -1, Shannon's entropy would be undefined and thus, for these regions we assume that probability of occupancy is 0.5. This assumption is valid as for unknown spaces, a grid cell on the map can either be an obstacle or free space with equal probability.

For global and local navigation, the Move Base ROS package is employed. Global navigation on Movebase employs A* search to plan a path given an end

destination.. The local navigation algorithm in Movebase is crucial for the project as it controls the robot to move along a trajectory that is planned by a global planner. Movebase employs the Dynamic Window Approach to control the robot along a planned trajectory. Furthermore, the robot trajectories from this global planner are smoothened by the local navigation planner for non-jerky control of the robot.

The exact robot architecture (graph of ROS nodes and topics) on ROS can be found in Appendix A.

2.3 Simulation Setup

The simulation is rendered using ROS middleware and Gazebo. The instruction set to deploy and run the simulation is provided in Appendix B.

The simulator was furthermore enhanced to support easy deployment of a team of turtlebots with similar capabilities that can be individually controlled. This can be easily achieved by declaring multiple robots within the launch file multi_robot_sim/launch/include/robots.xml. More details can be found in Appendix C.

The simulator supports merging of maps from many robots into a common map. This is achieved through the map_merge package in ROS. The simulator automatically detects the various robots used in the simulation (specified with a well-defined robot namespace) and merges their maps in real time.

2.4 Results

An environment to map using autonomous robots is built on Gazebo. Fig 2.2.a shows the setup of the environment.

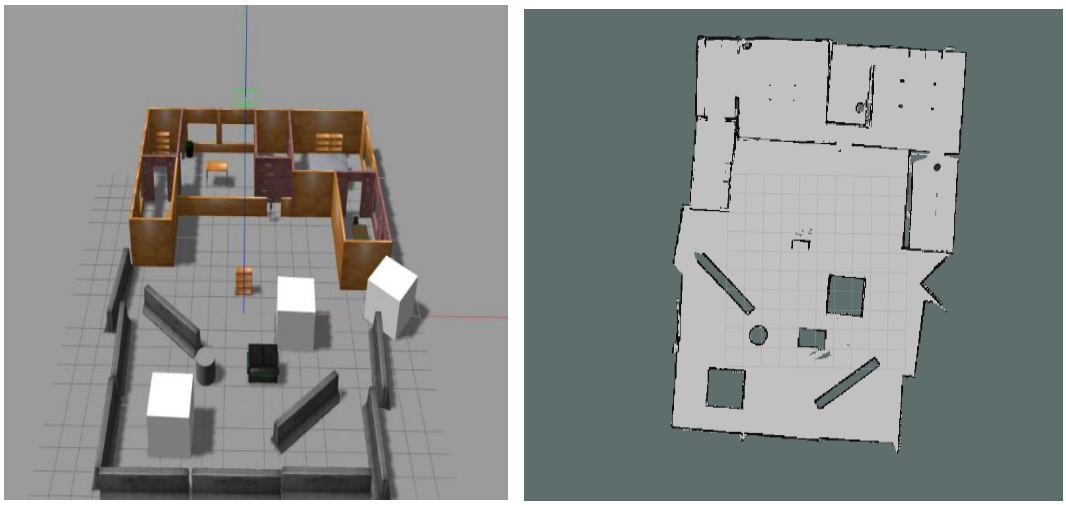


Fig 2.2 a) Environment with single turtlebot b) Map of the environment

2.5 Limitations

Some of the limitations of the simulator can be summarised as follows:

1. The simulator does not model communication within the robots for the creating a combined map of the environment. The design of the simulator is based on the assumption that there is a centralised system to merge maps from different robots and that all robots possess unrestricted communicate with this centralised system.

2. It is assumed that any robot in the simulator is aware of the exact location of the other robot. The simulator does not provide any means to a robot to evaluate the positions of other robots in its vicinity.

Addressing these limitations would improve the real-time fidelity of the simulator and would ease deployment and verification of future work on physical turtlebots.

2.6 Conclusions

This chapter has provided an overview of the simulation environment used to test the algorithms implemented in Chapter 3 and Chapter 4. Thus, significant engineering effort was invested into developing a reliable robot simulator in the first 8 to 10 weeks of the project. A ROS Architecture was developed for a turtlebot in the simulator for mapping and navigation functionalities.

Chapter 3 Characterising New Information

3.1 Introduction

This section briefly describes the algorithm used to evaluate the density of expected new information from the partial map of the environment known to the robot. Typically, robots would gain maximum amount of new information from frontiers to unknown space in this known partial map. Moreover, expected information content for map building would vary with properties of frontier's such as their size and no. of unknown cells in the vicinity of the frontier. Thus, this chapter proposes a method to capture such properties of a frontier and aims to provide a computationally tractable framework to build an information density map based on a known partial map. This information density map reflects the robot's expected information gain at a certain position in the known partial map.

3.2 Theory and Algorithm

To obtain this exact information map, ideally the robot would have to compute the number of unknown cell with in the range of the robot's sensors at each cell in the partial map. However, this method is computationally expensive especially when maps cover larger areas or have higher map resolutions. Thus, we attempt to provide a computationally tractable reliable estimate of the expected information gains based on the following assumptions.

- A partial map of the environment is known to the robot over which the information density map is computed.

- The map is ternary where -1 represents unknown space, 0 represents free space and 1 represents an obstacle.

- Robot possesses omnidirectional sensors.

The key idea employed is expected information gain from the frontiers in the given map reduces with distance between the robot and a frontier. For simplicity, we capture this distribution of information around the frontier using a Gaussian function centred at centroid of the frontier. Thus, the given information map would be a Gaussian mixture model of the form described below over the identified frontiers in the map.

$$I(x; v) = \sum_{k=1}^{K} \frac{w_k}{2\pi\sqrt{\epsilon_k}} e^{\frac{1}{2}(x-\mu_k)^T {\epsilon_k}^{-1}(x-\mu_k)} \tag{2.1}$$

$$\sum_{k=1}^{K} w_k = 1 \tag{2.2}$$

x represent a coordinate (x, y) on the map, and $v = (\mu_1, \epsilon_1, \mu_{2,}, \epsilon_2 \dots \mu_k, \epsilon_k)$ are parameters that correspond to the means μ_k and covariances ε_k. The means are the centroids of the computed frontiers and the covariances correspond to the spread of the gaussian function. This spread is required to capture useful properties of the frontiers such as it's size and shape. For instance, a larger frontier would have a larger spread. Thus, the first order moment of a frontier is calculated with respect to it's centroid as shown in Eq 3. This first order moment captures the spatial distribution of all cells in a frontier *i* with respect to it's centroid.

$$\mu_i = \sum_{k=1}^{K} (x_{ki} - x_{ci})\,(x_{ki} - x_{ci})^T \tag{2.3}$$

where $\mathbf{x_{ki}}$ is the coordinate of a frontier cell and $\mathbf{x_{ci}}$ is the coordinate of the centroid

Furthermore, each gaussian component computed for a frontier is weighted based on the expected new information content from the frontier. This is evaluated by computing the number of unknown cells in the range of the robot's sensor when the robot is placed at the centroid of the frontier. The weights are normalised to 1 prior to calculating the information map. The complete algorithm is summarised in Algorithm 3.1.

Algorithm 3.1: Computing Information Map

```
Result: Information Map
input  : A map M of size w × l
output: Information Map I of size w × l
Frontiers ← findFrontiers(M);
weights ← { };
variances ← { };
means ← { };
for each frontier f in Frontiers do
    v ← FirstOrderMoment(f);
    w ← InformationContent(f);
    m ← Centroid(f);
    add v to variances;
    add w to weights;
    add m to means;
end
I ← GausianMixture (means,variances,weights)
```

3.3 Results

The algorithm is tested over 2 maps to produce information density maps as shown in Fig 3.1 Fig 3.1 is the map obtained at the beginning of the exploration process where the robot performs an inplace rotation at the origin. Fig 3.1 b) is the map obtained at some unspecified time in the middle of the exploration process where there are fewer frontiers left to explore. In the information density map, darker regions correspond to higher expected information gain.

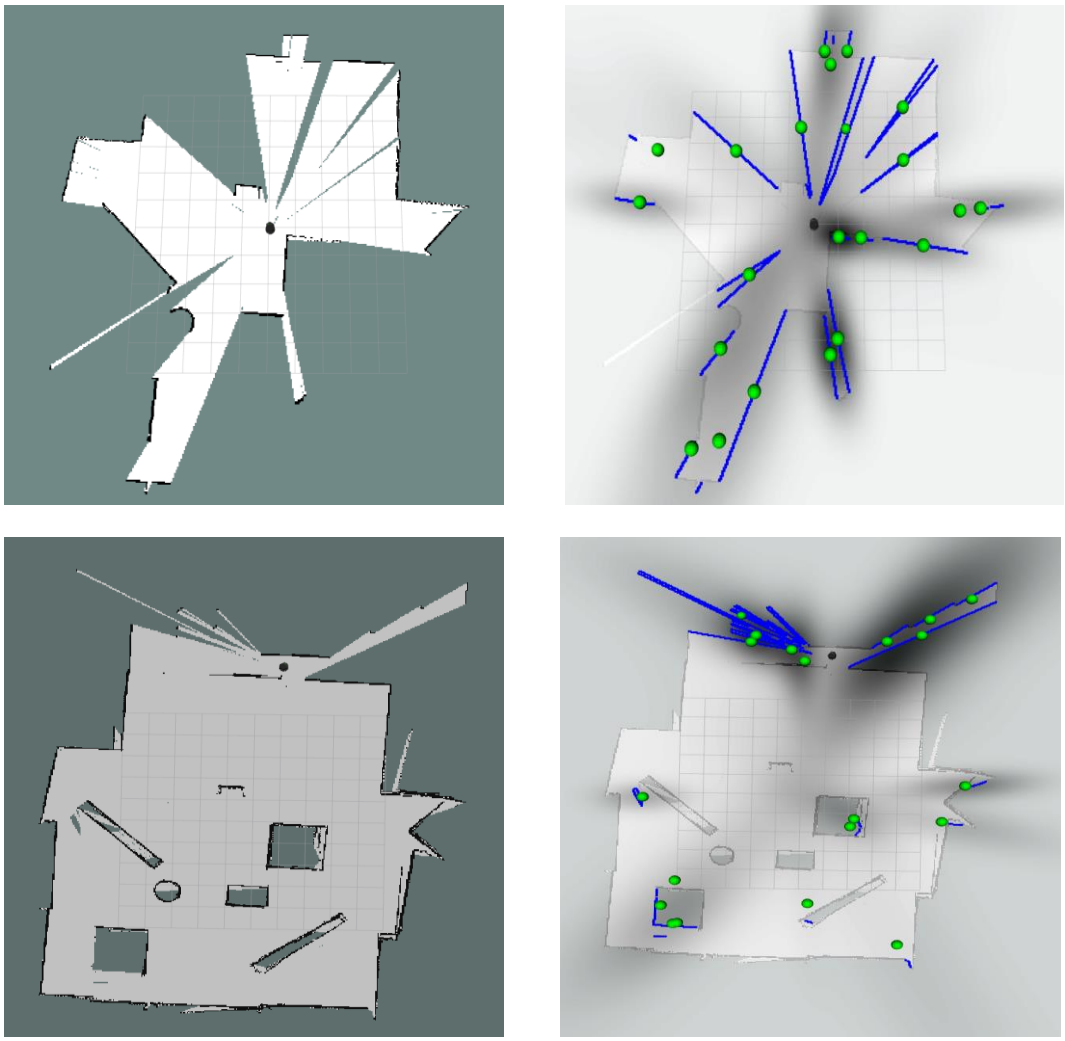


Fig 3.1 Information Distribution Map Results a,c) 2 partial maps of Environment b,d) Corresponding Information Distribution Maps

From Fig 3.1, it can be observed that the algorithm captures the information content over the partial map which is very intuitive. The following observations are made :

- Darker regions on the information map are closely concentrated around those frontiers to large unexplored spaces. Furthermore, these information maps accurately capture the relative importance of frontiers that are expected to provide more information as frontier are weighted based on their expected information gains.

- Larger frontiers may not necessarily provide new information. This is accurately reflected in Fig 3.1b. The large frontiers at the top left corner in Fig 3.1b have minimal information content compared to other regions.

3.4 Limitations

Some of the limitations of this approach can be summarises as follows:

1. The algorithm computes information gain at a frontier by calculating the number of all unknown cells in the range of the robot's sensor when placed at the centroid of the frontier. This computation includes unknown cells that are occluded by obstacles. Thus, this gives an inflated estimate of the information content at the frontier. As a result, in certain edge cases as shown in Fig 3.2, regions around frontiers close to large obstacles that encapsulate a large number of unknown cells appear to be darker than those regions around information dense frontiers at the boundaries of the map.

2. The computation of information gain at a frontier is based on the assumption that the robot's sensors are omnidirectional. For robot sensor's such as RGBD cameras which have horizontal field of views, information content would be maximum at certain orientations of the robot. The algorithm fails to capture this relationship.

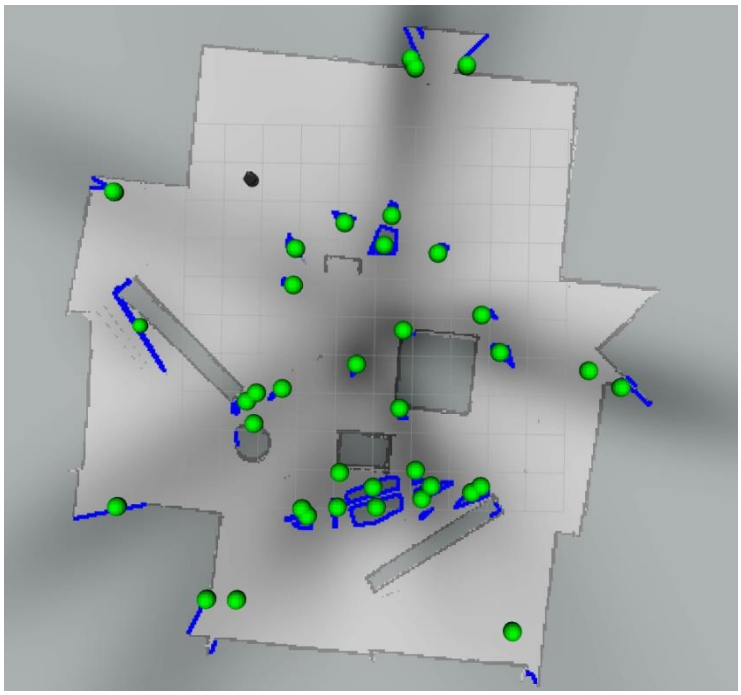


Fig 3.2 Edge Case where information map is inaccurate (Notice darker regions towards the centre as algorithm includes information from cells occluded by obstacle)

3.5 Future Work

There is scope for improving the information maps computed by the proposed method. A more reliable estimate of the expected information gain at the frontiers by eliminating unknown cells occluded by obstacles at frontiers. This can be achieved by incorporating ray-casting algorithms to identify observable unknown cells. Raycasting algorithms can also be used for evaluating the orientation of the robot that captures maximum amount of information from the frontier.

3.6 Conclusions

In summary, this chapter has provided a computationally tractable algorithm to evaluate the distribution of information gain across the partial map of the environment for map building. This information map is crucial for the evaluating a good path for a robot for map building. The chapter has addressed the limitations of the algorithm and has outlined the scope of future work to improve the performance of the map.

Chapter 4 Path Planning for Efficient Exploration

4.1 Introduction

In chapter 3 we proposed a method which allows the robot to infer and represent information content in a partial map. In this chapter, we address the problem of finding a good path for the robot to execute which maximises information gain to build the map of the environment. This problem is different from classical path planning algorithms in the sense that there is no goal for the robot to move to. Thus, the robot is supposed to select a good information rich trajectory from the set of all possible trajectories that the robot can execute over the given partial map to build a map of the environment. This is a combinatorial problem and is generally classified as a NP – Hard problem as no exact and efficient solutions are known. Therefore, good solutions to these classes of problems are stochastically sampled and optimised using techniques such as Bayesian Optimisation[19], and simulated annealing [20]. This chapter uses the Cross Entropy (CE) optimisation method to optimise good informative paths (see section 4.2.3).

Furthermore, it is intuitive that the robot spends more time in regions with high expected information gains to build good maps. In other words, the robot should spend more time sensing regions with high information gain while a robot moves along a path. More formally, the idea is to plan a trajectory over a sufficiently long-time horizon such that distribution of average time spent on gathering information from a region – while the robot moves along this trajectory – is similar to distribution of information in the partial map of the environment. Thus, we use the

principles of ergodic theory for dynamical systems[21] as shown in Section 4.2.1 to evaluate trajectories for robots to explore the environment.

Thus, this chapter proposes a motion planning algorithm that incorporates the above metric based on ergodicity in a sampling-based cross entropy trajectory optimisation framework for robot exploration. The key idea is that we try to model the search of a good path for the robot from a set of all possible paths as a low probability event. Thus, possible paths for a robot to take are sampled. A multi-level optimisation scheme improves these sampled paths at a given level by assessing the best paths. The best paths are used to sample paths for successive levels to produce better paths. The process is repeated until all samples drawn during a given iteration are identical. This would indicate that no further improvement can be made to the paths that are sampled.

We also present two different approaches for sampling feasible paths in sections 4.3 and 4.4 for a robot to safely execute in the environment based on two parametrisations of the trajectory. The cross-entropy optimisation framework is deployed over both the trajectory sampling schemes for evaluating informative paths. The merits of both sampling processes are analysed in section 4.5.

Since the process of mapping the environment is faster with multiple robots, we also attempt to propose a mathematical framework for a team of robots to plan a trajectory to build the map of the environment in a decentralised manner in section 4.6. This is important as decentralised systems are robust to failure of a single agent in the team.

4.2 Theory

This section provides a brief overview of the principles behind ergodic theory and cross entropy optimisation method.

4.2.1 Ergodic Theory

Ergodic theory is a statistical study of time dynamical systems averaged over time. Thus, for an agent with a sensor, the time average statistics quantifies the amount of time spent on gathering information through the robot's sensor while moving along a trajectory.

Following the developments made in [22], the ergodicity of robot's sensor footprints is evaluated as follows:

$$\Gamma(x) = \int_0^T f\big(x - \gamma(\tau)\big)d\tau \tag{4.1}$$

Where

$$f\big(x - \gamma(\tau)\big) = \begin{Bmatrix} 1\ if|x - \gamma(\tau)| \le r \wedge \theta_\tau - \beta \le arctan\left(\frac{(x-\gamma(\tau))_2}{(x-\gamma(\tau))_1}\right) \le \theta_\tau + \beta \\ 0\ otherwise \end{Bmatrix} \tag{4.2}$$

where $\gamma(\tau) = (q_x(\tau), q_y(\tau))$ and $\theta_\tau = q_\theta(\tau)$ denote the position and orientation of the robot at time τ respectively. r and β denote the range and horizontal field of view of the robot.

The robot's sensor footprint can be visualised in Fig 4.1.

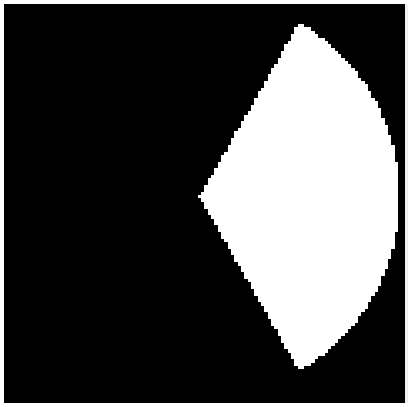


Fig 4.1 Sensor Footprint of Robot with Horizontal FOV of 120 and range 5m

A system exhibits ergodic behaviour with respect to a desired distribution over its state if its time averaged statistics is similar to the desired distribution. In this case, the distribution of the average time spent on gathering new information of a location should be similar to the information density map. ‘

Mathew and Mezic[21] proposes a metric based on fourier transformation for evaluating ergodicity for the trajectories of dynamic systems with respect to a distribution. Fourier coefficients are differentiable and thus useful for continuous systems. However, for sampling-based motion planning algorithms there is no need for this differentiability. There are many other measures available to compare two distributions. Here, we compare the similarity of the two distributions using KL Divergence – one is obtained from evaluating the time averaged statistics of the robot’s sensor along a trajectory and the other is the information map ***I(x)*** obtained from the developments made in Chapter 3. KL Divergence evaluates the relative entropy between two measures and is not symmetric.

$$D(\Gamma|I) = \int \Gamma(x) log \frac{\Gamma(x)}{I(x)} dx \tag{4.3}$$

4.2.2 Trajectory Parametrisations

A trajectory for a robot $\pi(t)$ represents the control and state tuple $\big(u(t), q(t)\big)$ of the robot at a time *t.* More formally a trajectory is a mapping $\pi: [0, t_f] \rightarrow U \times Q$ over a set of controls U and states Q.

By incorporating the evolving kinematics and dynamics of the robot, a trajectory can be parametrized in terms of independent vectors $z \in Z$, where $Z \subset R^{n_z}$is the parameter space. The parameter *z* can either be discrete controls $u(t)$ at a time t or discrete states $q(t)$. Based on this definition, there is a unique mapping from the parameter space R^{n_z} to the space of trajectories which is given by $\varphi: Z \rightarrow \pi$. In other words, for some trajectory in its respective domain, there is a vector in the parameter space such that

$$\pi = \varphi(z) \tag{4.4}$$

The control state tuples of a trajectory can be written as π(t) = φ(z,t). The cost *J(z)* for a given trajectory parameter z can be rewritten as :

$$J(z) = \int_0^T C\big(\varphi(z,t)\big)dt \tag{4.5}$$

Thus, our primary objective is to evaluate an optimal trajectory parameter $z*$ $t$hat minimised the above cost function.

$$z^* = arg_z min \int_0^T C\big(\varphi(z,t)\big)dt \tag{4.6}$$

4.2.3 Cross Entropy Optimisation

Cross-Entropy(CE) Optimisation [23] process is treated as an estimation of probabilities of rare events. The rare event in this problem is drawing a parameter z whose cost is very close to that of the optimal parameter z^* that minimises $J(z)$. This is achieved through the concept of importance sampling [24]. Importance sampling is a technique to estimate the properties of a specific distribution by drawing samples from another easily computable distribution such as a Gaussian.

For motion planning using CE Optimisation, feasible trajectory parameters are sampled from a probability prior such as a Gaussian over the domain R^{n_z} , that satisfy actuator and physical constraints (such as obstacles). The costs of these trajectory parameters are evaluated using the metric $J(z)$. However, as the cost of the optimal parameter is generally unknown, we employ a multi-level optimisation scheme where at each level, the trajectory parameter samples which have the lowest costs (top 1% to 10%) are used to compute a new refined distribution for sampling at subsequent levels. Eventually, over a few iterations the distribution from which trajectory parameters are sampled converge to a Dirac Delta distribution. A Dirac Delta distribution indicates that the samples drawn are identical and an optimal parameter has been found. The motion planning solution is thus the parameter which has the lowest cost from these identical samples.

More formally, we assume that trajectory parameter samples for the algorithm are drawn from a Gaussian Mixture Model(GMM) which is defined as follows:

$$p(z;v) = \sum_{k=1}^{K} \frac{w_k}{2\pi\sqrt{\in_k}} e^{\frac{1}{2}(z-\mu_k)^T {\in_k}^{-1}(z-\mu_k)} \tag{4.7}$$

Where and $v = (\mu_1, \epsilon_1, \mu_{2,}, \epsilon_2 \ldots \mu_k, \epsilon_k)$ are the gaussian parameters that define this Gaussian Mixture Model and z is a trajectory parameter. These parameter samples are used to generate a trajectory using kinematic and dynamic constraints. These trajectory samples are then optimised using the CE method. This optimisation is summarised in the following steps:

1) Initialise gaussian parameter v_0 that defines the sub space over which the trajectory parameters are sampled from.
2) Sample $z_1, z_2, .. z_n p(z; v_i)$ and evaluate costs $J(z_1), J(z_2), \ldots, , J(z_n)$
3) Compute trajectories whose costs are in top ρ^{th} quantile. ρ could be any value between 1% and 10%. We call these trajectories an elite set of trajectories
4) Compute new Gaussian parameters that v_{i+1} from the elite set of trajectories using expectation maximisation.
5) Repeat step 2 until the samples correspond to a Dirac distribution. Covariance matrices in the Gaussian Mixture Model parameters would have very small covariance values.
6) Assign trajectory parameter that has lowest cost as optimal solution

Do note that if the gaussian prior used for sampling does not provide a uniform coverage of the trajectory samples over the environment, the solution would converge to a local minimum. Ideally, with good coverage the optimisation algorithm would identify a global maxima.

We present two sampling strategies for trajectories in the following section. The generated trajectories are optimised by this framework.

4.3 Approach 1: Sampling Control Primitives

4.3.1 Trajectory Parametrisation and Generation

The algorithm assumes the dynamics of commonly available differential drive robots such as a turtle bot. The dynamics of the robot can be defined as follows.

$$\acute{x} = u_v cos(\theta)\ \acute{y} = u_v sin(\theta)\ \acute{\theta} = u_\theta \tag{4.8}$$

Where (u_v, u_θ) are control inputs for linear and angular velocity respectively.

For this algorithm, a trajectory is parameterised using a sequence of *n* control primitives $z = (u_{v_1}, u_{\theta_1}, u_{v_2}, u_{\theta_2}, \ldots u_{v_n}, u_{\theta_n})$ where each control primitive (u_{v_i}, u_{θ_i}) acts over a time interval Δt_i . For simplicity this time interval is assumed to be a constant τ. Given a robot's starting location $q_x(0), q_y(0), q_\theta(0)$, a sequence of control primitive generates a trajectory for the robot to take. Thus, for a given primitive (u_{v_i}, u_{θ_i})acting over a time interval $[t_{i-1}, t_i]$, where $t_i - t_{i-1} = \tau$, the trajectory spline $\varphi(z, t) = (u_v, u_\theta, q_x, q_y, q_\theta)$ for a time $t \in [t_{i-1}, t_i]$ is given by

$$u_v(t) = u_{v_i} \tag{4.9}$$

$$u_\theta(t) = u_{\theta_i} \tag{4.10}$$

$$q_\theta(t) = q_\theta(t_{i-1}) + u_{\theta_i}\Delta t_i \tag{4.11}$$

$$q_x(t) = \begin{cases} q_x(t_{i-1}) + \dfrac{u_{v_i}}{u_{\theta_i}}\left(sinq_\theta(t) - sinq_\theta(t_{i-1})\right) & if u_{\theta_i} \neq 0 \\ q_x(t_{i-1}) + u_{v_i}\Delta t_i cosq_\theta(t) & if u_{\theta_i} = 0 \end{cases} \tag{4.12}$$

$$q_y(t) = \begin{cases} q_y(t_{i-1}) + \dfrac{u_{v_i}}{u_{\theta_i}}\left(cosq_\theta(t) - cosq_\theta(t_{i-1})\right) & if\ u_{\theta_i} \neq 0 \\ q_y(t_{i-1}) + u_{v_i}\Delta t_i sinq_\theta(t) & if\ u_{\theta_i} = 0 \end{cases} \tag{4.13}$$

where $\Delta t_i = t - t_{i-1}$

Do note that the condition imposed for q_x, q_y based on u_{θ_i}arises from parametrising this trajectory spline as an arc.

4.3.2 Trajectory Costs

The following considerations were made to evaluate a trajectory generated using the sequence of control primitives.

1) Proximity of robot's position with respect to obstacles in the environment.
2) Ergodicity of the robot's sensor footprint with respect to the distribution of information in the environment
3) Length of the total trajectory

For consideration 1, we use the costmap $M: \gamma \rightarrow R$ generated by move_base where $\gamma(t) = (q_x(t), q_y(t))$ is the position of the robot. This costmap represents the difficulty of traversing a trajectory by checking its proximity to obstacles for collision avoidance. For consideration 2, we use the principles of ergodic theory which is introduced in Section 4.2.1.

Thus, the total cost of the trajectory encoded by the parameter z and starting position $q(0) = \big(q_x(0), q_y(0), q_\theta(0)\big)$

$$J(z) = w_1 \int_0^{t_n} M\big(\gamma(t)\big)dt + w_2 D\big(\Gamma(x) \mid I(x)\big) + w_3||q|| \qquad (4.14)$$

We apply the cross entropy optimisation framework as described in Section 4.2 for the computation of the optimal parameter z* given the initial robot conditions $q_x(0), q_y(0), q_\theta(0)$. Note that the length of the trajectory also includes it's angular component.

4.3.3 Numerical Results

The performance of the algorithm is demonstrated over two partial maps.

The map in Fig 4.2a) is obtained when the robot performs an in-place rotation during the beginning of exploration. The trajectory is composed from a sequence of 3 control primitives. u_{v_i} components of the control primitives which indicates the speed of the robot is set to a constant of 0.8 m/s. Each primitive acts for a constant time of 5 seconds and the robot initiates its trajectory from the origin. Thus, the planning process occurs over a time horizon of 15 seconds. The optimal trajectory is shown in Fig 4.2.a and the corresponding information map and time averaged statistics of the robots sensor footprint are shown in Fig 4.2.b and Fig 4.2.c.

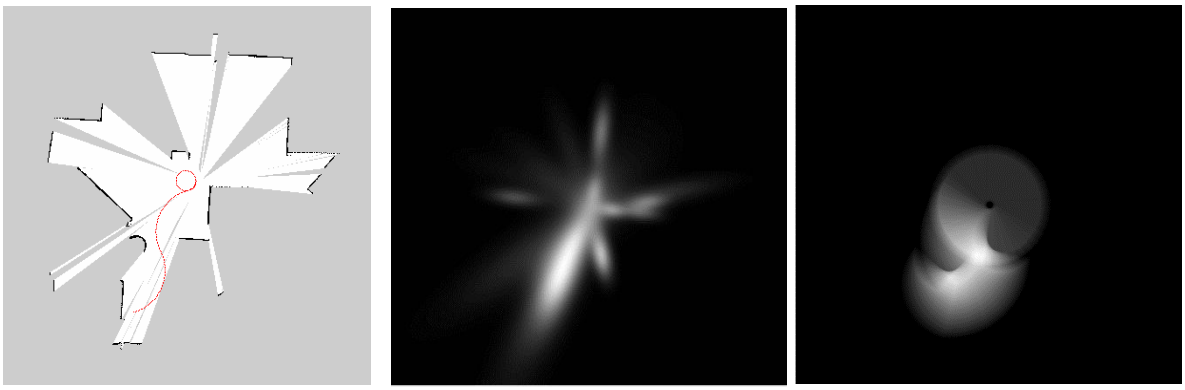


Fig 4.2 Motion Planning Results on Initial Map a) Map with Optimal Trajectory b) Information Distribution Map c) Ergodic Sensor Footprint of Robot

The map in Fig 4.3.a is obtained when the robot maps out a large section of the environment. The trajectory is composed from a sequence of 5 control primitives where the u_{v_i} components of the control primitives which indicate the speed of the robot is set to a constant of 1.0 m/s. Each primitive acts for a constant time of 5 seconds and the robot initiates its motion plan from the origin. Thus, the planning process occurs over a time horizon of 25 seconds. The optimal trajectory is shown in Fig 4.3.a and the corresponding information map and ergodicity of sensor footprint of the robot are shown in Fig 4.3.b and Fig 4.3.c. Notice how the robot moves towards the regions with highest information gain while covering those regions with intermediate information gain.

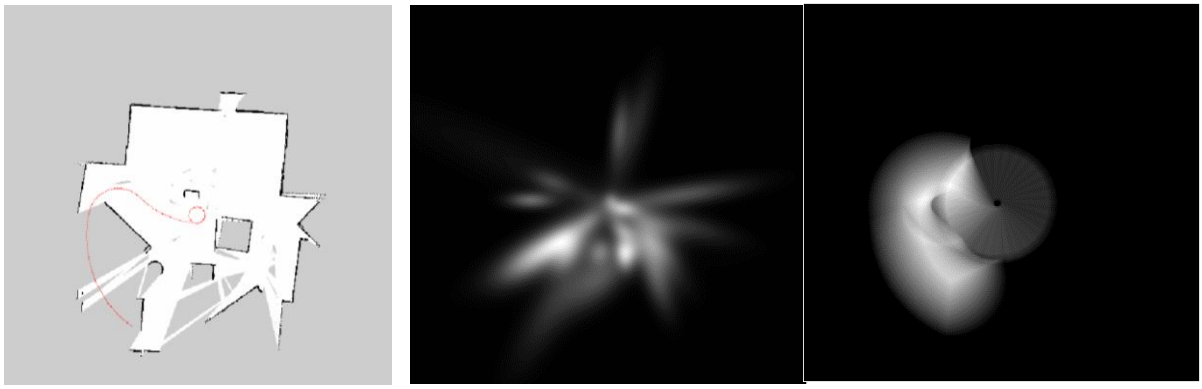


Fig 4.3 Motion Planning Results on a Larger Map a) Map with Optimal Trajectory b) Information Distribution Map c) Ergodic Sensor Footprint of Robot

The optimisation process stops when the sampled trajectories are identical. The convergence of the samples can be verified in Appendix D. Appendix D also plots the convergence plots of the costs.

4.3.4 Limitations

A primary issue with this approach is the computation time required for sampling. While sampling for control primitives, the algorithm rejects trajectories that do not satisfy obstacle constraints and the control primitives are resampled until a feasible trajectory is found. Due to this, the computation time for motion planning process increases.

Furthermore, as the size of the map increases (Fig 4.4), generated trajectories are not effectively distributed across the map of the environment especially with the presence of features such as narrow passages. Good coverage of the environment is required for generating an optimal path using the cross-entropy optimisation algorithm. Without good coverage, the algorithm would either converge to a local maximum or not converge at all. Thus, parameters crucial to the planning algorithm such as number of primitives, and planning horizon need to be tuned manually. In this case as shown in Fig 4.4, convergence is observed to a local optimal solution (see Appendix E).

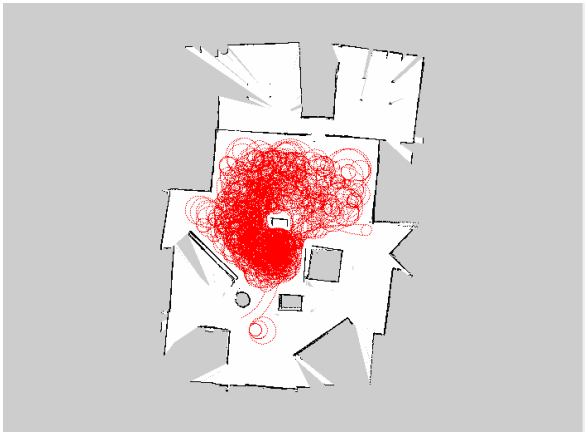


Fig 4.4 Sampled Trajectories over a larger map with narrow passage with 8 control primitives and planning horizon of 60 seconds

4.4 Approach 2: Sampling from Probabilistic Roadmaps

4.4.1 Roadmaps

A probabilistic roadmap (PRM) [25] is a motion planning strategy to determine a path between a robot's starting and goal configurations. The primary idea behind probabilistic roadmaps is to sample random collision free configurations or states of the robot from the known map of the environment and connect these configurations/states to neighbouring configurations/states. The result is a graph of connected configurations/states on which classical graphical search algorithms can be applied for robot path planning.

Since no goal configuration is known to us, we use these roadmaps to generate trajectories for a robot. Furthermore, using roadmaps is advantageous as collision free trajectories can be generates in lesser amount of time compared to generating trajectories from control primitives. Furthermore, roadmap algorithms sample free states uniformly and randomly. This is useful for finding optimal paths as seed trajectories for the first iteration of the Cross - Entropy Optimisation process will be uniformly distributed over the environment.

In summary, this approach employs a roadmap $R = \{V, E\}$ which is constructed using the PRM algorithm. Roadmap vertices V correspond to collision free vertices on the map or $V = \{ (q_x, q_y) : (q_x, q_y)\ is\ collision\ free\ on\ map\}$. Edges are connected between two sampled positions if their relative distance is below a threshold $D_r$and the path is collision free. D_r is determined using actuator constraints.

4.4.2 Trajectory Parametrisation and Generation

For this algorithm, a trajectory is parameterised using a sequence of positions $z = (q_{x_1}, q_{y_1}, q_{x_3}, q_{y_2}, \ldots q_{x_n}, q_{y_n})$ where each position primitive $\gamma_i = (q_{x_i}, q_{y_i})$ is adjacent to position primitive $\gamma_{i-1} = (q_{x_{i-1}}, q_{y_{i-1}})$ on the roadmap. Adjacency of two positions in the context of the roadmap implies that there is an edge between these two positions.

Thus, we produce position variants using a randomised variant of Depth First Search. The algorithm for generating these trajectories is summarised in Algorithm 4.1. Furthermore, we impose the condition that the robot travels between adjacent position primitives (γ_i, γ_{i-1}) from the time interval $[t_{i-1}, t_i]$, where $t_i - t_{i-1} = \tau$. Thus, the trajectory spline $\varphi(z, t) = (u_v, u_\theta, q_x, q_y, q_\theta)$ for a time $t \in [t_{i-1}, t_i]$ can be defined as:

$$u_v(t) = \frac{|\gamma_i - \gamma_{i-1}|}{\tau} \tag{4.15}$$

$$u_\theta(t) = 0 \tag{4.16}$$

$$q_\theta(t) = arctan\left(\frac{(\gamma_i - \gamma_{i-1})_y}{(\gamma_i - \gamma_{i-1})_x}\right) \tag{4.17}$$

$$q_x(t) = q_{x_i} + u_v(t)\, \Delta t_i cos q_\theta(t) \tag{4.18}$$

$$q_y(t) = q_{y_i} + u_v(t) \Delta t_i sin q_\theta(t) \tag{4.19}$$

Algorithm 4.1: Sampling Postion Primitives from Roadmap

Result: Information Map
input : $R = < V, E >$ - A roadmap where V is a set of sampled states and E is a set of collision free edges between any two adjacent states
N - num of sampled trajectories
start - starting location to roadMap
k - number of primitives for trajectory
output: A set of trajectories T of size N

```
T ← {};
R ← addVertex(start , R);
j ← 1;
while j ≤ N do
    i ← 1;
    traj ← {start} //stores a trajectory;
    isVisited ← {} //checks if node is visited;
    primitive ← start //stores current primitive ;
    prev ← {} //queue for parent primitives in path;
    while i ≤ k do
        neighbours ← getNeighbours(primitive,R);
        for each n in neighbours do
            if n in isVisited then
                remove n from neighbours;
            end
        end
        if neighbours is empty then
            node ← pop(prev);
            continue;
        else
            add primitive to prev;
            primitive ← random(neighbours);
            add primitive to traj;
        end
    end
    add traj to T;
end
```

4.4.3 Trajectory Costs

The cost of a trajectory is evaluated using equation 4.14 as described in Section 4.3.2. It has been reproduced as follows:

$$J(z) = w_1 \int_0^{t_n} M\big(\gamma(t)\big)dt + w_2 D\big(\Gamma(x) \vee I(x)\big) + w_3 ||q|| \qquad (4.14)$$

We apply the cross-entropy optimisation framework as described in Section 4.2 for the computation of the optimal parameter z*. Intuitively, the cross-entropy framework is loosely coupled with the probabilistic roadmap in this algorithm. At each stage of the optimisation process, the refinement of trajectory parameters corresponds to the assignment of more importance to certain regions on the map where the robot is expected to get most information. In other words, the cross-entropy optimisation framework narrows down the domain, from which the PRM algorithm samples collision free states, to smaller specific regions on the map which are important for generating good informative paths. Eventually by virtue of Algorithm 4.1, similar trajectories are generated once sampling of free configurations is confined to very small regions on the map.

4.4.4 Numerical Results

The motion planning results are obtained on the maps shown in Fig 4.2.a and 4.3.a and have been reproduced in Fig 4.5.b and 4.6.b.

For the first map, a trajectory is composed of a sequence of 5 position primitives sampled from the roadmap as shown in Fig 4.5.a using algorithm 4.1. The planning horizon is set to 15 seconds, where each primitive is expected to be completed by the robot in 3 seconds. The corresponding information map and time averaged statistics of sensor footprints are shown in Fig 4.5.c and Fig 4.5 d.

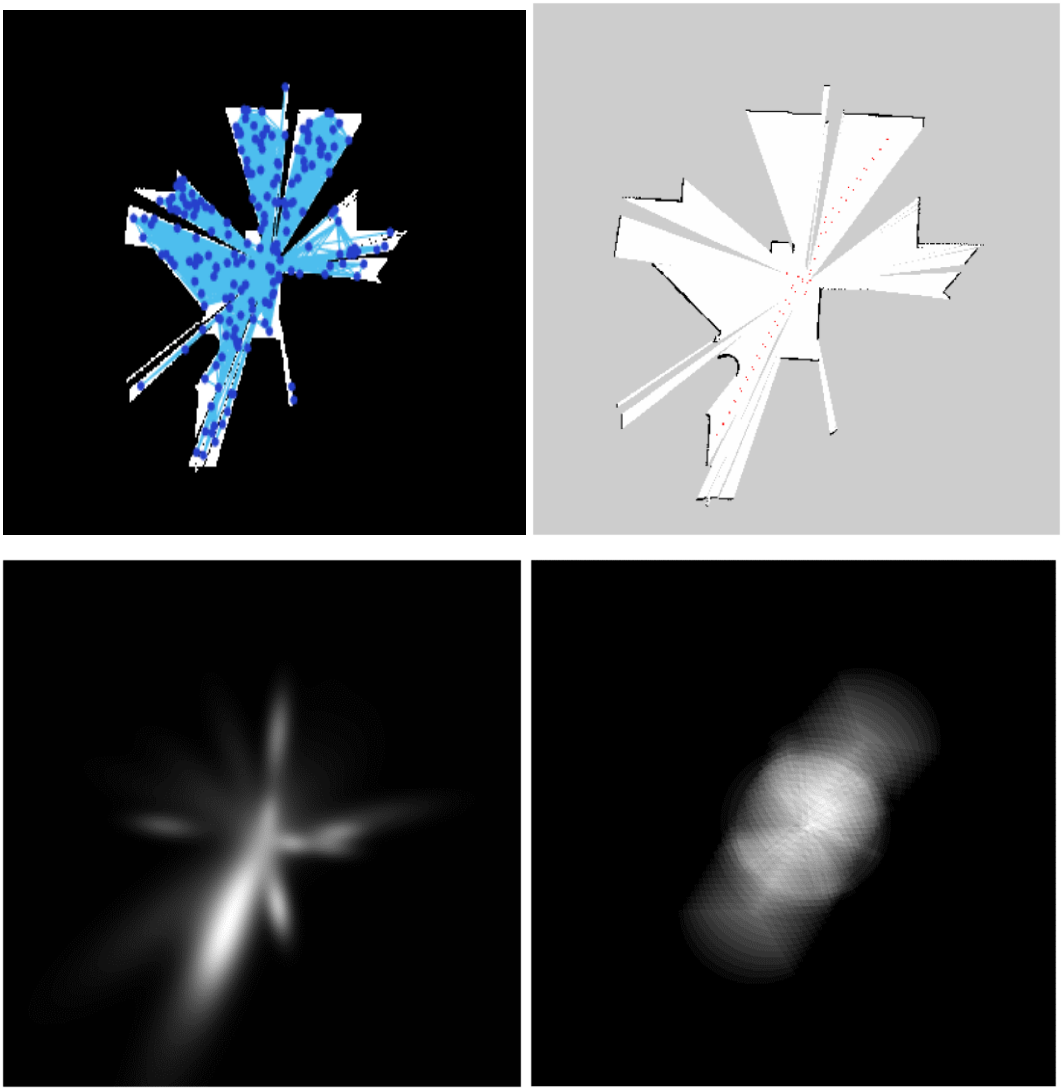


Fig 4.5 Motion Planning Results on Initial Map a) Roadmap for Sampling b) Map with Optimal Trajectory c) Information Distribution Map d) Ergodic Sensor Footprint of Robot

For the second map, a trajectory is composed of a sequence of 10 position primitives sampled from the roadmap as shown in Fig 4.6.a using algorithm 4.1. The planning horizon is set to 25 seconds, where each primitive is expected to be completed by the robot in 2.5 seconds. The corresponding information map and time averaged statistics of sensor footprints are shown in Fig 4.6.c and Fig 4.6 d.

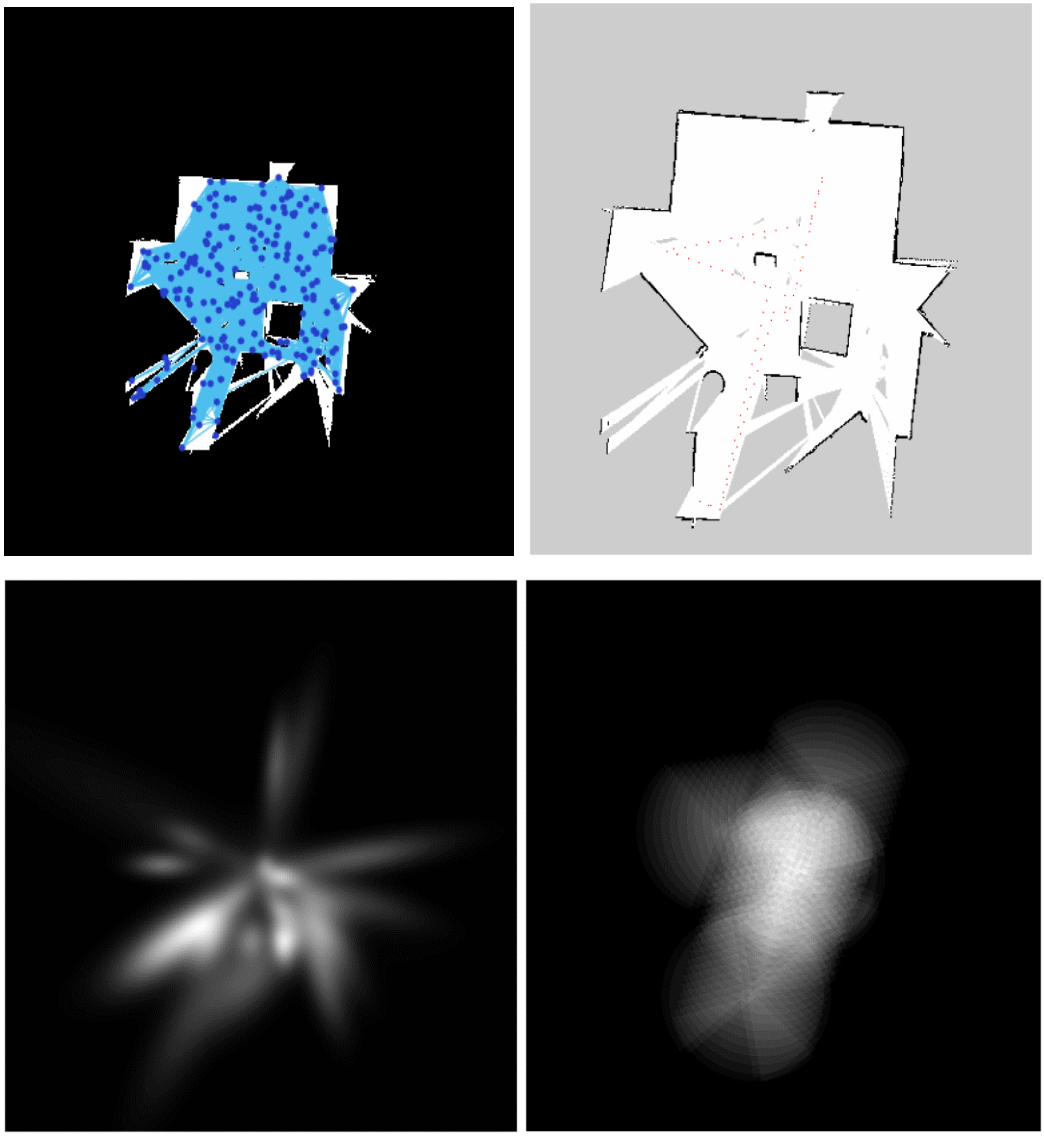


Fig 4.6 Motion Planning Results on a Larger Map a) Roadmap for Sampling b) Map with Optimal Trajectory c) Information Distribution Map d) Ergodic Sensor Footprint of Robot

The curious reader is referred to Appendix F to observe the trajectory convergence process. It can be observed that the sampled trajectories are uniformly distributed over the environment.

4.4.5 Discussions

As observed from Fig 4.5.a and Fig 4.6.a, the roadmaps sample free configurations uniformly over the environment. Thus, trajectories generated using algorithm 4.1 are more uniformly distributed compared to those generated using the approach in Section 4.3 (see Appendix F). Furthermore, as the size of the map increases as shown in Fig 4.7; we observe that the generated trajectories effectively cover the map of the environment especially in maps with features such as narrow passages. (see Appendix G).

One limitation of this approach is that after a few iterations, generally (3 iterations) the costs of the trajectory increases erratically (see Appendix F). As mentioned before, the cross-entropy optimisation framework narrows down the domain, from which the PRM algorithm samples collision free states, to smaller specific regions on the map which are important for generating good informative paths. After a few iterations, these specific regions narrow down to multiple points. If an obstacle exists between two points, the roadmap algorithm would fail to connect these points. This leads to a fragmented roadmap and the trajectories are sampled from a partition of the roadmap by virtue of Algorithm 4.1. This explains the increase in costs after a few iterations.

4.5 Performance Comparison

The merits of both sampling approaches are analysed in this section. Table 4.1 summarises the costs of the best path produced after the 2 approaches are applied.

	Sampling using Control Primitives				Sampling From RoadMaps			
Maps	Total Cost	Ergodicity Cost	Length	Time for planning (s)	Total Cost	Ergodicity Cost	Length	Time for planning
Map Fig 4.2.a	12.61	9.75	22.62	15.2	14.29	13.15	29.1	24.45
Map Fig 4.3.a	12.79	10.73	31.43	24.22	17.1	15.01	41.84	26.54
Map Fig 4.4	29.5	27.16	48.56	94.43	27.94	25.37	56.32	27.45

Table 4.1 Performance Comparisons between two approaches

It is observed that during the beginning of exploration where maps are smaller and features such as narrow corridors are less prominent, path sampling using a sequence of control primitives results in a better path than using roadmaps. As the map gets bigger, better paths are obtained by the latter approach. It is also observed that the resulting path from the second approach is much longer as compared to the first approach.

Thus, a hybrid approach to sampling trajectories is recommended where during the beginning of the exploration process, control primitives are sampled to generate trajectories. As more features are observed in the map, trajectories can be generated using roadmaps.

4.6 Towards Decentralised Exploration

4.6.1 Introduction

Based on the results of the motion planning approaches in Section 4.3 and Section 4.4, a robot is expected to move towards regions with high expected information gain. With multiple robots and no coordination scheme, robots would crowd towards such regions. This behaviour is undesired and thus, a robot should be able to adapt its motion plan to move towards another region of high information density when it observes another robot moving towards the same region it has initially planned for. Therefore, the key insight towards decentralising motion planning that would coordinate robots in a team for exploration is this notion of a robot predicting the motion plan of another robot in its neighbourhood based on the other robot's prior locations. We try to address this problem in this section.

The problem of path prediction is regarded here as inferring future positions of another robot based on a priori positions. We break the problem down into 2 sections. Firstly, as a robot's position evolves by virtue of a kinematics model such as (4.8), we first estimate hidden variables such as speed and angular velocity of the robot. This estimated velocity is then projected using the kinematics model to evaluate future positions of the robot.

4.6.2 Estimation

More formally, the estimated velocities ($\boldsymbol{v}^*_{1:k}$) of another robot are those velocities that maximise the belief of the robot being at its prior positions $\boldsymbol{q}_{1:k}$ and starting position $\boldsymbol{q_0}$. Note that $\boldsymbol{v}^*(i) = (v*_x(i), v*_y(i))$ and $\boldsymbol{q}(j) = (q_x(j), q_y(j))$. Thus,

$$\{\boldsymbol{v}^*_{1:k}\} = \underset{\overline{\boldsymbol{v}}_{1:k}}{\text{argmax}} \; p(\overline{\boldsymbol{v}}_{1:k} \mid \boldsymbol{q_0}, \boldsymbol{q}(1:k)) \tag{4.20}$$

This estimation problem (4.20) includes two sources of difficulty: long time horizon and continuous space. By exploiting the Markov property of the kinematics model where future observations and states to be dependent on the current observation and state, we can simplify the estimation problem by evaluating the current velocity $\boldsymbol{v}*_{\boldsymbol{i}}$ that produces observation $\boldsymbol{q}_i$ based on prior observation $\boldsymbol{q_{i-1}}$. In other words,

$$\{\boldsymbol{v}*_{\boldsymbol{i}}\} = \underset{\overline{\boldsymbol{v}}_i}{\text{argmax}} \prod_{j=1}^{i} p(\boldsymbol{q_j} \mid \overline{\boldsymbol{v}}_j) \; p(\overline{\boldsymbol{v}}_j \mid \boldsymbol{q_{j-1}}) \tag{4.21}$$

If the state space is discrete, dynamic programming methods such as Viterbi's algorithm [26] can be used to solve this problem. However, since the state space is continuous, we need to perform this inference over a continuous state. Thus, an approximate inference algorithm such as a particle filter [27] which discretises continuous space by sampling particles from valid regions of the state space. In other words, the procedure using a particle filter algorithm can be summarised as follows:

1) Initialise particles sampled uniformly from the domain of velocities.
2) Begin loop and set j = 1
3) Predict future state $\bar{\boldsymbol{q}}_j$ based on sampled velocity particles $\bar{\boldsymbol{v}}_{\boldsymbol{j}}$, using the kinematics model
4) Update likelihood of velocity particles based on proximity of predicted future state $\bar{\boldsymbol{q}}_j$ with respect to $\boldsymbol{q}_{\boldsymbol{j}}$
5) Resample particle velocities based on new obtained likelihood using importance sampling

The use of particle filters to estimate hidden parameters sets using current observations has been successfully demonstrated in [28].

4.6.3 Prediction

Therefore, after obtaining a reliable estimate of the posterior belief as described in (4.20) based on observations made on robot positions, we can now predict where the robot is expected to be in future time instances. This is a trivial problem as we first sample particle velocities based on the belief distribution using importance sampling. This belief distribution has been computed using the particle filter algorithm as described before. Future robot positions can be predicted by applying the kinematics model over the sampled velocities. Note that the future positions of the robot would also be distribution.

4.6.4 Discussions

After inferring such future positions, a robot can evaluate which regions is neighbouring robot moving towards. The problem now lies in incorporating this inference as a metric for planning a new path for a given robot. Future work can be

conducted on this problem to obtain a fully decentralised motion planning algorithm to coordinate exploration effort in robot teams.

4.7 Future Work

The scope of future work primarily lies in resolving some of the limitations observed in the above-mentioned approaches to planning good informative paths. Some of these include:

- Develop algorithms that tunes parameters for a good initial distribution trajectories that are parametrised by sequences of control primitives over the environment.
- Real time verification of map building using the motion planning algorithms developed in Section 4.3 and 4.4 on the simulator described in Chapter 2. This can be done by implementing a ROS plugin for the motion planning algorithm within the navigation stack.

Based on the concepts developed in Section 4.6, future work can also be scoped on developing a decentralised motion planning scheme for multi-robot map building applications. Thus, work could be conducted on:

- Real time verification of trajectory inference on simple frontier-based exploration
- Incorporating inferences on trajectories of neighbouring robots to adjust the motion plan for a given robot for decentralised multi-robot exploration.

4.8 Conclusions

In summary, this chapter has provided two sampling-based motion planning algorithms that implement cross entropy optimisation to evaluate a good exploration path for a robot to maximise information gain for map building. The chapter has addressed the limitations of the algorithms and has outlined the scope of future work to improve the performance of these algorithms. Furthermore, this chapter has introduced a mathematical framework for a robot to predict and infer the trajectories other robots in its vicinity. This framework is useful for coordinating exploration in a decentralised manner where a given robot can adjust its motion plan in accordance with the inference it makes on motion plans of other robots.

Chapter 5 Concluding Remarks

This Chapter presents a summary of the material covered in the preceding Chapters, reiterating the main contributions, and discusses some avenues of future work

5.1 Summary of Contributions

We began with a comprehensive literature review on the work done on exploration using mobile robots and presented some of the open research questions in Chapter 1.

Chapter 2 provided an overview of the simulation environment developed on ROS and Gazebo. A turtlebot is used for simulation purposes and an architecture was developed for the turtlebot for mapping and navigation purposes. We presented some of the limitations of the simulator and scope of future work to improve real time fidelity of the simulator for multi-robot mapping applications.

Chapter 3 describes a method used to evaluate the distribution of information across the partial map of the environment. This chapter presents a method to reliably estimate a distribution of expected information gain as a function of the robot's position at a given position.

We present a sampling based probabilistic motion planning scheme in Chapter 4 that formulates a path which maximises information gain from the surroundings using the Cross-Entropy Optimisation framework. The planning algorithm incorporates a metric to measure the ergodicity of a robot's sensor footprint along a planned trajectory with respect to the information distribution map in Chapter 3.

This chapter also presents a mathematical framework that can used to coordinate exploration using multiple robots in a decentralised manner.

5.2 Summary of Future Work

The following are some of the avenues for future work to be conducted:

1) Modelling communication between robots for improving real time fidelity of the simulator.
2) Modelling localisation of robots in the vicinity of another robot.
3) Reliably estimating expected information gain at a frontier using ray casting algorithms
4) Automated parameter tuning for good initial coverage of trajectory samples parametrised by a sequence of control primitives.
5) Real time verification of map building with developed simulator by integrating navigation stack with motion planning algorithm.
6) Real time verification of trajectory inference algorithm for decentralised motion planning.
7) Incorporating inferences of trajectories of neighbouring robots to adjust motion plan for a given robot in a decentralised manner.

References

[1] Stachniss, C. (2009). *Robotic mapping and exploration*. Berlin: Springer. doi:10.1007/978-3-642-01097-2

[2] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using raoblackwellized particle filters. In Proc. of robotics: science and systems (RSS), pages 65–72. Citeseer, 2005.

[3] F. Amigoni. Experimental evaluation of some exploration strategies for mobile robots. In IEEE International Conference on Robotics and Automation, pages 2818–2823. IEEE, 2008.

[4] Sondik, E. (1971). The Optimal Control of Partially Observable Markov Decision Processes. Ph.D. thesis, Stanford University, Stanford, California

[5] Y. Du, D. Hsu, H. Kurniawati, W. S. Lee, S. C. W. Ong, and S. W. Png. A POMDP Approach to Robot Motion Planning under Uncertainty. Int. Conf. on Automated Planning and Scheduling, 2010.

[6] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large POMDPs. Journal of Artificial Intelligence Research, 27(1):335–380, 2006.

[7] Shade, R. J. (2011). *Choosing where to go: Mobile robot exploration*

[8] Kollar, T., & Roy, N. (2008). Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research, 27*(2), 175-196. doi:10.1177/0278364907087426

[9] Jeong, H., Schlotfeldt, B., Hassani, H., Morari, M., Lee, D. D., & Pappas, G. J. (2019). Learning Q-network for Active Information Acquisition. *arXiv preprint arXiv:1910.10754*.

[10] Yan, Z., Jouandeau, N., & Cherif, A. A. (2013). A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems, 10*(12), 399. doi:10.5772/57313

[11] Wurm, K. M. (2012). *Techniques for Multi-Robot Coordination and Navigation* (Doctoral dissertation, Verlag nicht ermittelbar).

[12] Robert Zlot, Anthony (Tony) Stentz, M. Bernardine Dias, and Scott Thayer. Multi-robot exploration controlled by a market economy. In Proceedings of ICRA'02, pages 3016-2023, Washington, DC, USA, May 2002.

[13] M. Bernardine Dias and Anthony Stentz. A free market architecture for distributed control of a multirobot system. In Proceedings of IAS'00, pages 115-122, Venice, Italy, July 2000.

[14] Mac Schwager, Daniela Rus, and Jean-Jacques Slotine. Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. International Journal of Robotics Research, 30(3):371-383, March 2011

[15] Sandip Kumar and Suman Chakravorty. Multiagent Generalized Probabilistic RoadMaps: MAGPRM. In Proceedings of IROS'12, pages 37473753, Vilamoura, Portugal, September 2012

[16] Singh, A., Krause, A., Guestrin, C., Kaiser, W., and Batalin, M. (2007). Efficient planning of informative paths for multiple robots. In Proc. International Joint Conference on Artificial Intelligence (IJCAI).

[17] Shannon, Claude E. (July–October 1948). "A Mathematical Theory of Communication". *Bell System Technical Journal* (PDF). 27 (3): 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x. hdl:11858/00-001M-0000-002C-4314-2

[18] Fox, D.; Burgard, W.; Thrun, S. (1997). "The dynamic window approach to collision avoidance". *IEEE Robotics & Automation Magazine*. **4** (1): 23–33. doi:10.1109/100.580977.

[19] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in Advances in neural information processing systems, 2012, pp. 2951–2959

[20] P. J. Van Laarhoven and E. H. Aarts, “Simulated annealing,” in Simulated Annealing: Theory and Applications. Springer, 1987, pp.7–15

[21] G. Mathew and I. Mezic, “Metrics for ergodicity and design of ergodic dynamics for multi-agent systems,” Physica D: Nonlinear Phenomena, vol. 240, no. 4, pp. 432–442, 2011.

[22] Ayvali, E., Salman, H., & Choset, H. (2017, September). Ergodic coverage in constrained environments using stochastic trajectory optimization. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5204-5210). IEEE.

[23] M. Kobilarov, “Cross-entropy randomized motion planning,” in Proceedings of Robotics: Science and Systems, Los Angeles, CA, USA, June 2011.

[24] R. Y. Rubinstein and D. P. Kroese, The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning. Springer Science & Business Media, 2013.

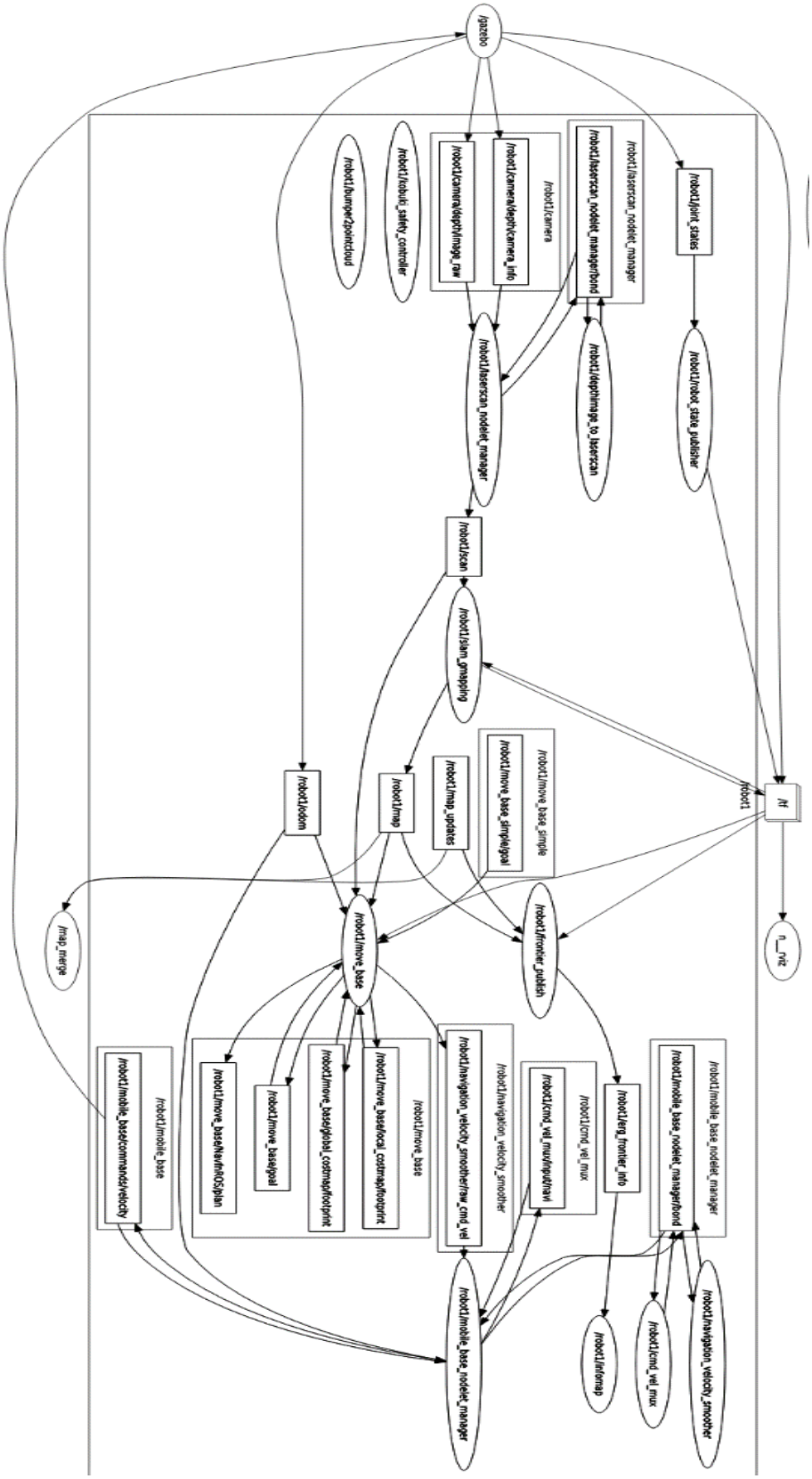
[25] *Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; Overmars, M. H. (1996), "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", IEEE Transactions on Robotics and Automation, 12 (4): 566–580, doi:10.1109/70.50843*

[26] G. D. Forney, "The viterbi algorithm," in *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268-278, March 1973.

[27] S. Godsill, A. Doucet, and M. West, “Maximum a posteriori sequence estimation using monte carlo particle filters,” Annals of the Institute of Statistical Mathematics, vol. 53, no. 1, pp. 82–96, 2001.

[28] Ha, Jung-Su & Chae, Hyeok-Joo & Choi, Han-Lim. (2018). Approximate Inference-Based Motion Planning by Learning and Exploiting Low-Dimensional Latent Variable Models. IEEE Robotics and Automation Letters. PP. 1-1. 10.1109/LRA.2018.2856915.

Appendix A ROS Architecture



Appendix B Command Set for running a simulation

1. The code for the simulator and project can be found at https://github.com/mercury070599/final_year_dissertation.

2. The following commands should be used to deploy the multi_robot simulator

```
roscore
```

Launches core functionalities of ROS

```
roslaunch multi_robot_sim robots_gazebo_rviz.launch
```

Launches the simulation with Gazebo for visualisation and Rviz for sensor logging and visualisation, and displaying maps

```
roslaunch multi_robot_sim robots_gazebo.launch
```

Launches the simulation with Gazebo for visualisation.

```
roslaunch multi_robot_sim keyboard_teleop_robot1.launch
```

Launches teleoperative controller for robot 1

```
roslaunch multi_robot_sim keyboard_teleop_robot2.launch
```

Launches teleoperative controller for robot 2

```
roslaunch multi_robot_sim keyboard_teleop_robot3.launch
```

Launches teleoperative controller for robot 3

Appendix C Deploying multiple robots

1. Multiple robots can be deployed by modifying the multi_robot_sim/launch/include/robots.launch.xml

2. Add the following code to deploy a robot

```
<!-- BEGIN ROBOT 1-->
  <group ns="robot1">
    <arg name="initial_pose_x" value="1" />
    <arg name="initial_pose_y" value="0" />
    <arg name="initial_pose_z" value="0" />
    <arg name="initial_pose_yaw" value="-1.5" />
    <param name="map_merge/init_pose_x" value="$(arg
initial_pose_x) "/>
    <param name="map_merge/init_pose_y" value="$(arg
initial_pose_y)"/>
    <param name="map_merge/init_pose_z" value="$(arg
initial_pose_z)"/>
    <param name="map_merge/init_pose_yaw" value="$(arg
initial_pose_yaw)"/>
    <!-- <param name="tf_prefix" value="robot1_tf" /> -->
    <include file="$(find
multi_robot_sim)/launch/include/robot.launch.xml" >
      <arg name="initial_pose_x" value="$(arg initial_pose_x)" />
      <arg name="initial_pose_y" value="$(arg initial_pose_y)" />
      <arg name="initial_pose_z" value="$(arg initial_pose_z)" />
      <arg name="initial_pose_yaw" value="$(arg
initial_pose_yaw)" />
      <arg name="robot_name"  value="robot1" />
    </include>
  </group>
```

3. Set a different namespace for each robot deployed. The namespace can be changed by modifying the <group ns> tag. Set different initial positions of the robot.

.

Appendix D Cross Entropy Motion Planning Optimisation

For map described in Fig 4.2.a

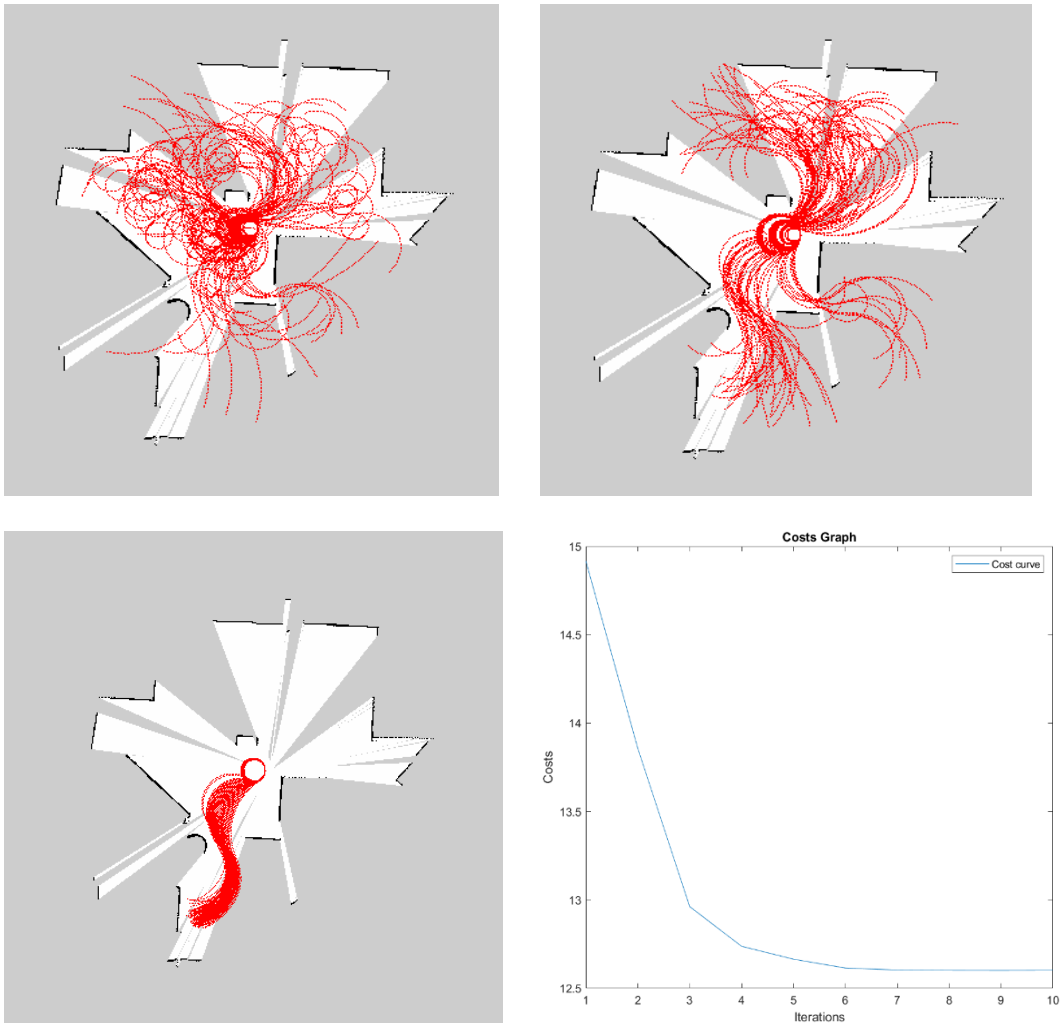


Fig E.1 Trajectory samples at a)1st iteration b)3rd iteration c) 6 th iteration of the cross entropy motion planning process for map in Fig 4.2.a d) cost of best trajectory at each iteration

For map described in Fig 4.3.a

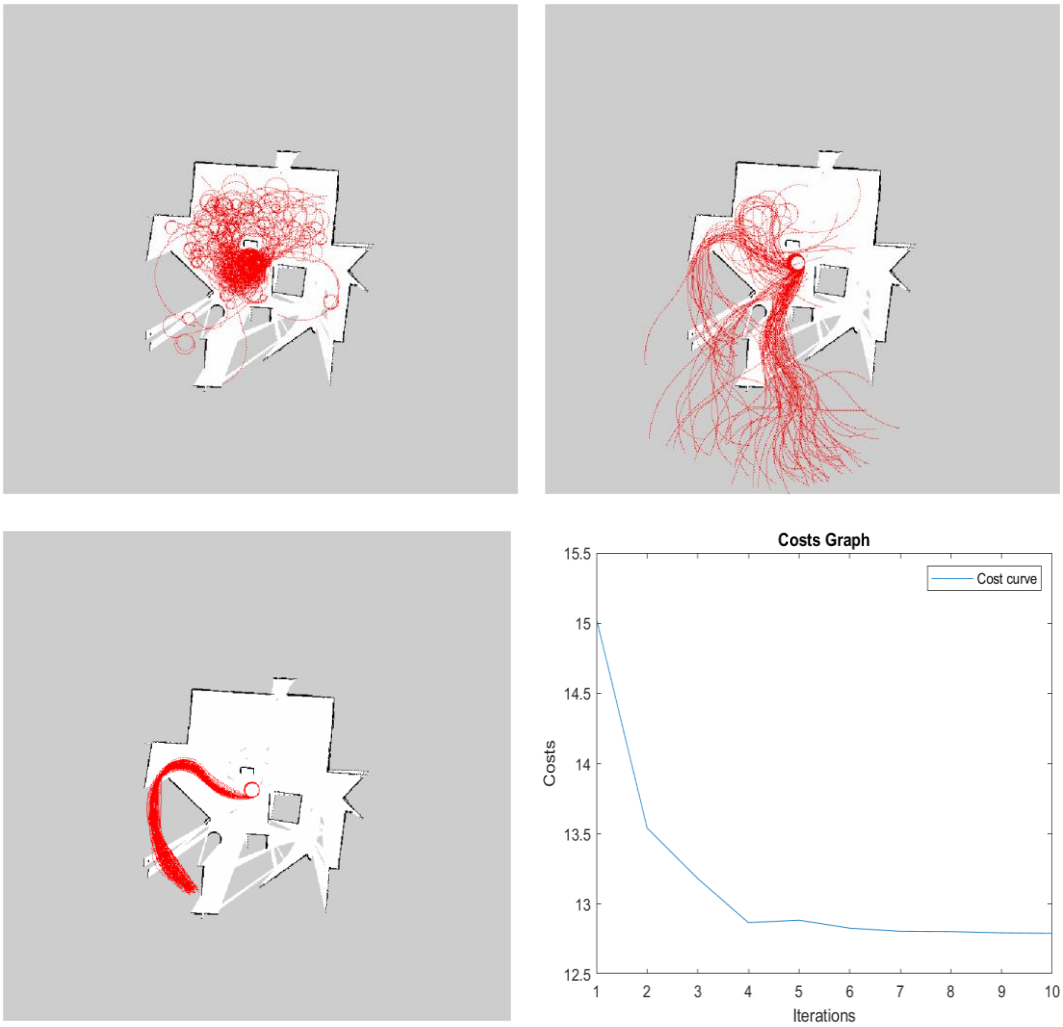
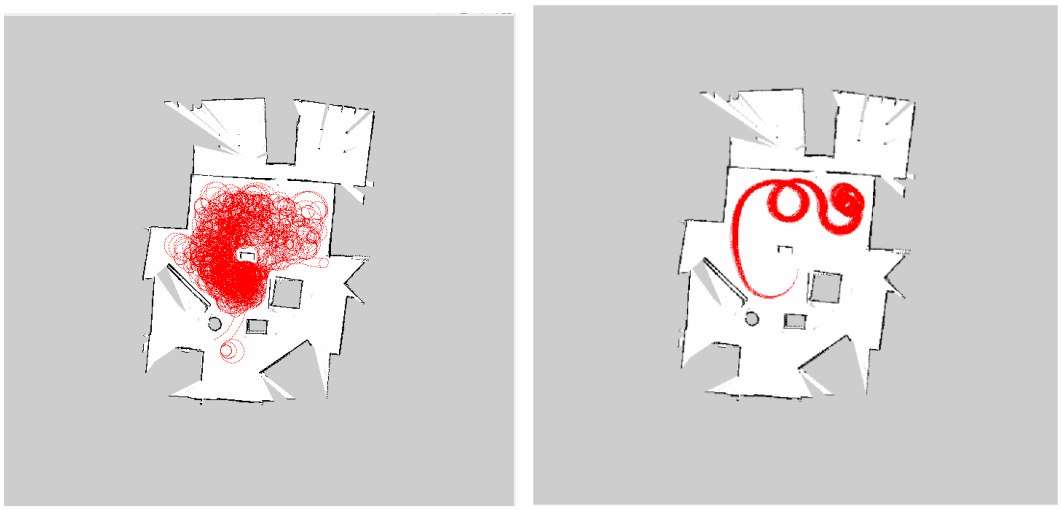


Fig E.2 Trajectory samples at a)1st iteration b)4th iteration c) 5th iteration of the cross entropy motion planning process for map in Fig 4.3.a d) cost of best trajectory at each iteration

Appendix E Convergence of Motion Planning using Control Primitives on a larger map

For map described in Fig 4.4



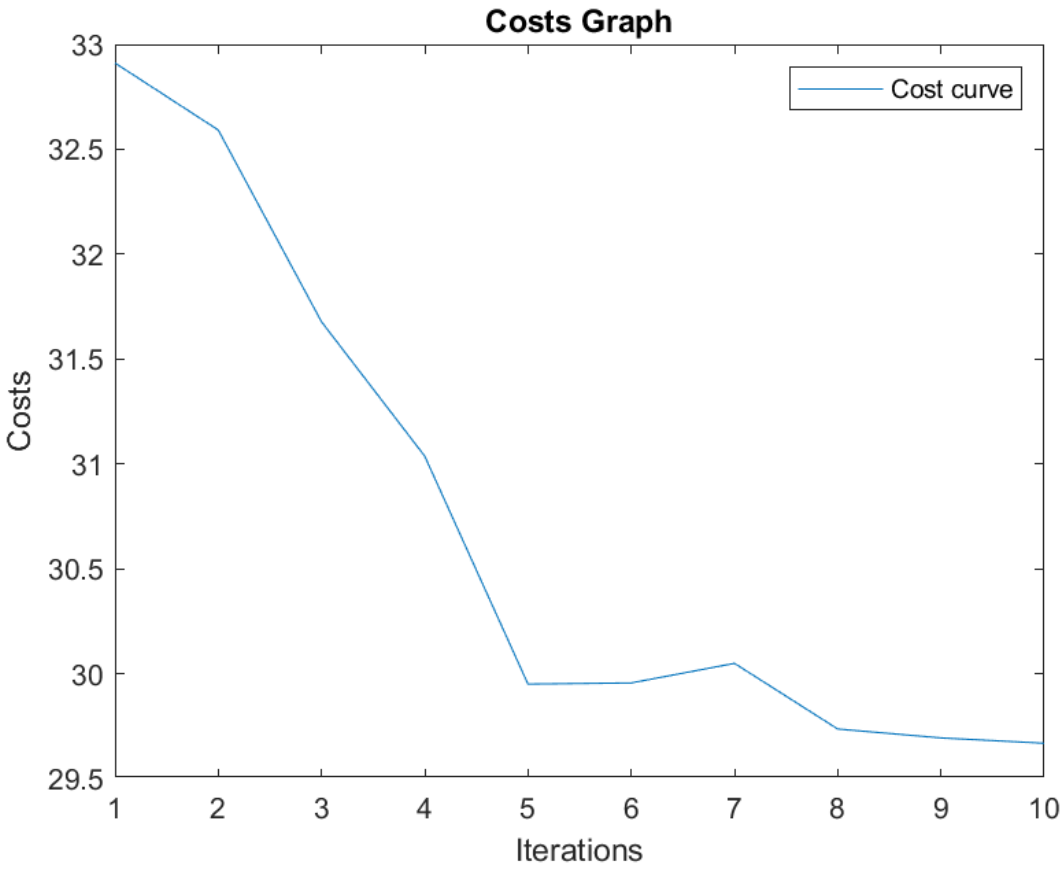


Fig F.2 Trajectory samples at a)1st iteration b)10th iteration during the cross entropy motion planning process for map in Fig 4.4 c) cost of best trajectory sample at each iteration

Appendix F Trajectory Sampling from RoadMaps of Motion Planning

For Map in Fig 4.2.a

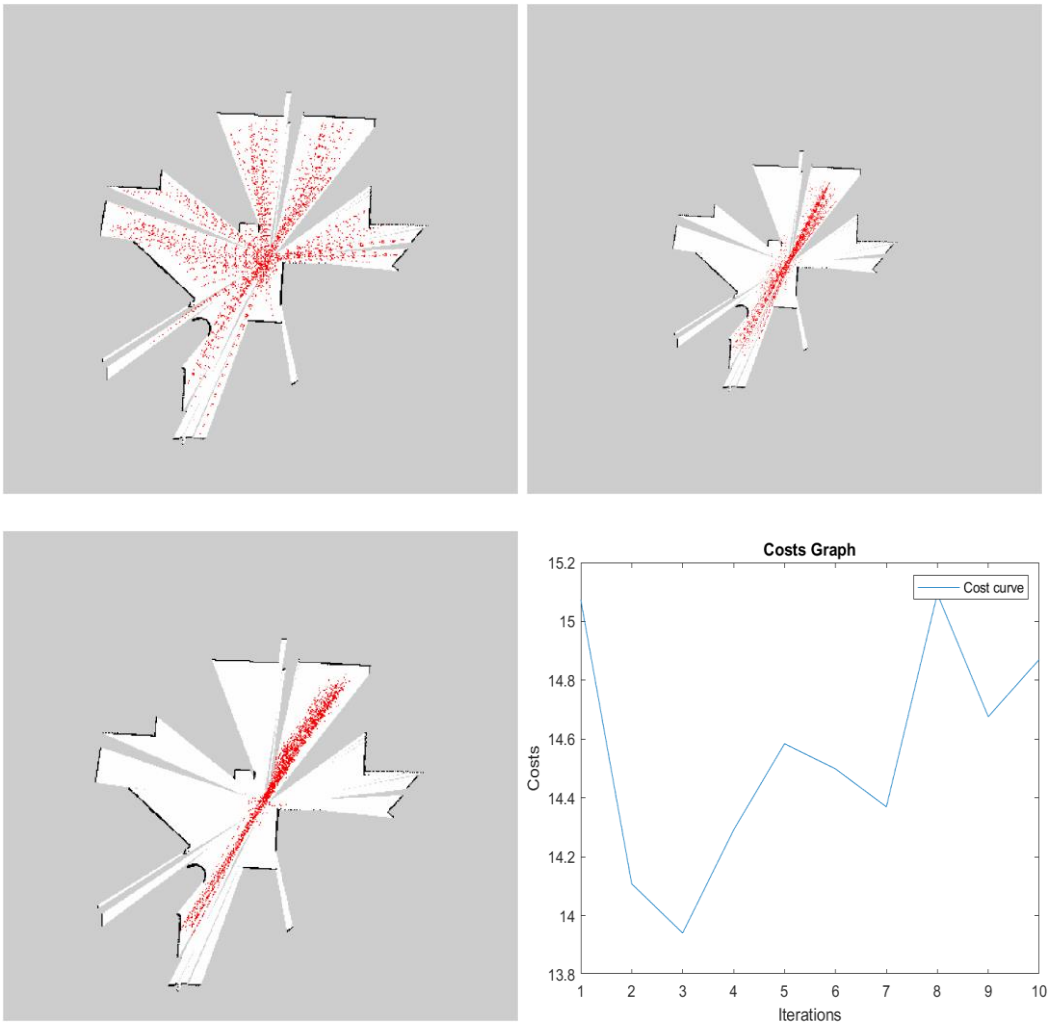


Fig G.1 Trajectory samples at a)1st iteration b)2nd iteration c) 3rd iteration of the cross entropy motion planning process for map in Fig 4.2.a d) cost of best trajectory at each iteration

For Map in Fig 4.3.a

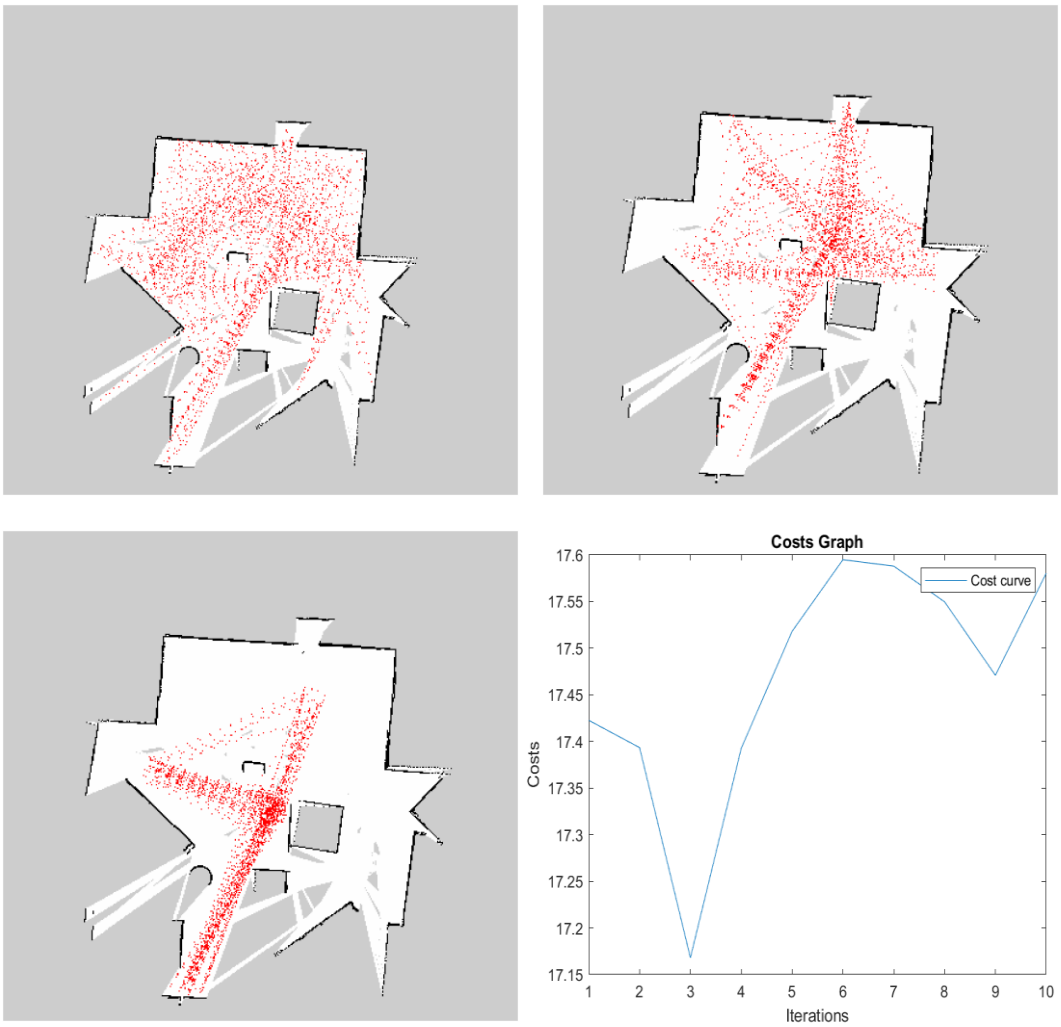


Fig G.2 Trajectory samples at a)1st iteration b)2nd iteration c) 3rd iteration of the cross entropy motion planning process for map in Fig 4.2.a d) cost of best trajectory at each iteration

Appendix G Convergence of Motion Planning on Larger Map using Roadmap Sampling

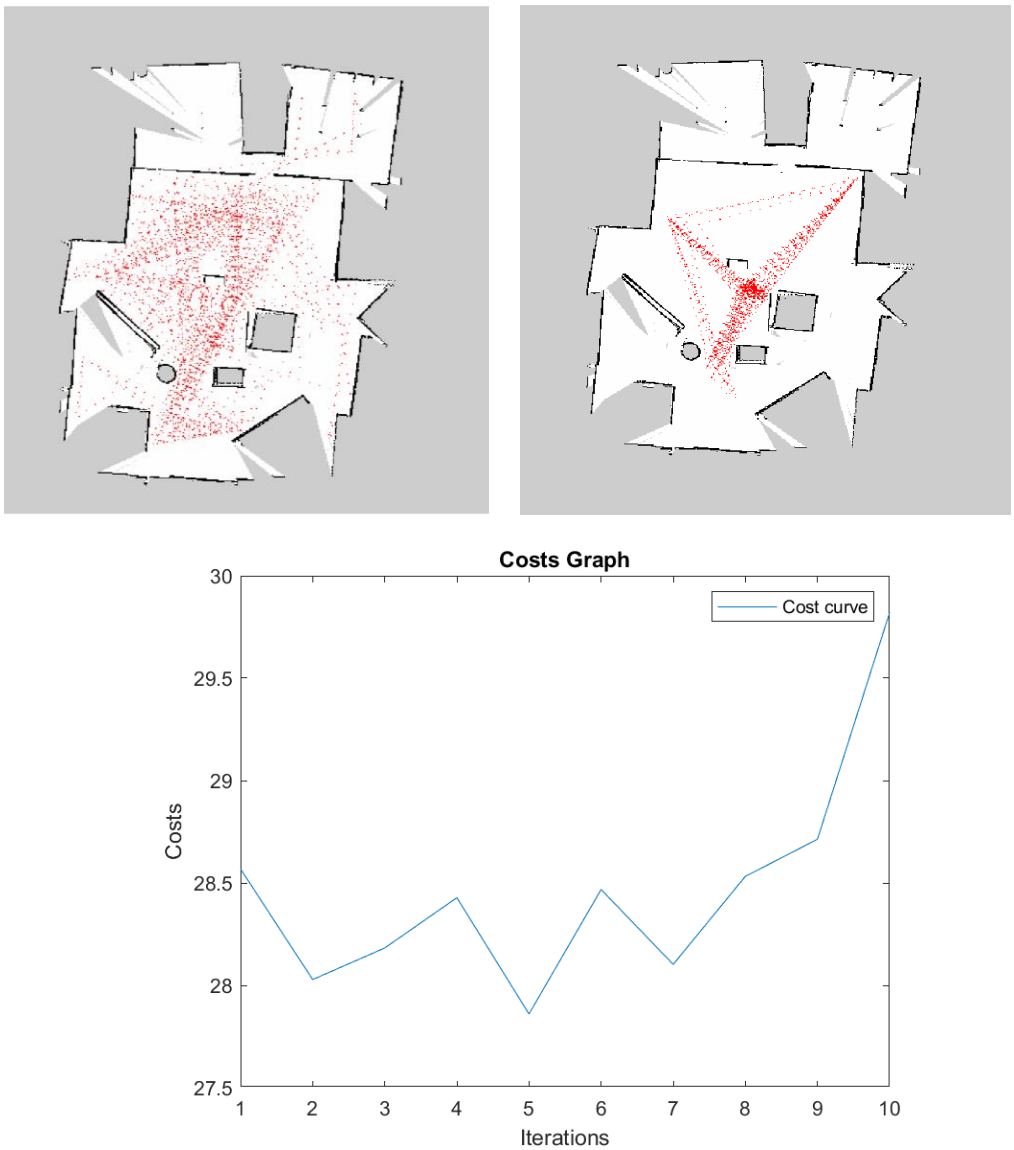


Fig H.1 Trajectory samples at a)1st iteration b)5th iteration during the cross entropy motion planning process for map in Fig 4.4 c) cost of best trajectory sample at each iteration