Ph.D. DISSERTATION

Robust Localization and Efficient Path Planning for Mobile Sensor Networks

모바일 센서네트워크에서의 강인한 위치 측정과 효율적 경로 탐색

BY

Junghun Suh

FEBRUARY 2016

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Robust Localization and Efficient Path Planning for Mobile Sensor Networks

모바일 센서네트워크에서의 강인한 위치 측정과 효율적 경로 탐색

지도교수 오 성 회

이 논문을 공학박사 학위논문으로 제출함

2015 년 11 월

서울대학교 대학원

전기컴퓨터 공학부

서 정 훈

서정훈의 공학박사 학위논문을 인준함

2015 년 11 월

위 원 장 이 범 희

부 위원장 오 성 회

위 원 심 형 보

위 원 김 현 진

위 원 박 재 병

Abstract

The area of wireless sensor networks has flourished over the past decade due to advances in micro-electro-mechanical sensors, low power communication and computing protocols, and embedded microprocessors. Recently, there has been a growing interest in mobile sensor networks, along with the development of robotics, and mobile sensor networks have enabled networked sensing system to solve the challenging issues of wireless sensor networks by adding mobility into many different applications of wireless sensor networks. Nonetheless, there are many challenges to be addressed in mobile sensor networks. Among these, the estimation for the exact location is perhaps the most important to obtain high fidelity of the sensory information. Moreover, planning should be required to send the mobile sensors to sensing location considering the region of interest, prior to sensor placements. These are the fundamental problems in realizing mobile sensor networks which is capable of performing monitoring mission in unstructured and dynamic environment.

In this dissertation, we take an advantage of mobility which mobile sensor networks possess and develop localization and path planning algorithms suitable for mobile sensor networks. We also design coverage control strategy using resource-constrained mobile sensors by taking advantages of the proposed path planning method.

The dissertation starts with the localization problem, one of the fundamental issue in mobile sensor networks. Although global positioning system (GPS) can perform relatively accurate localization, it is not feasible in many situations, especially indoor environment and costs a tremendous amount in deploying all robots equipped with GPS sensors. Thus we develop the indoor localization system suitable for mobile sensor networks using inexpensive robot platform. We

focus on the technique that relies primarily on the camera sensor. Since it costs less than other sensors, all mobile robots can be easily equipped with cameras. In this dissertation, we demonstrate that the proposed method is suitable for mobile sensor networks requiring an inexpensive off-the-shelf robotic platform, by showing that it provides consistently robust location information for low-cost noisy sensors.

We also focus on another fundamental issue of mobile sensor networks which is a path planning problem in order to deploy mobile sensors in specific locations. Unlike the traditional planning methods, we present an efficient cost-aware planning method suitable for mobile sensor networks by considering the given environment, where it has environmental parameters such as temperature, humidity, chemical concentration, stealthiness and elevation. A global stochastic optimization method is used to improve the efficiency of the sampling based planning algorithm. This dissertation presents the first approach of sampling based planning using global tree extension.

Based on the proposed planning method, we also presents a general framework for modeling a coverage control system consisting of multiple robots with resource constraints suitable for mobile sensor networks. We describe the optimal informative planning methods which deal with maximization problem with constraints using global stochastic optimization method. In addition, we describe how to find trajectories for multiple robots efficiently to estimate the environmental field using information obtained from all robots.

Keywords: Indoor localization system, Cost-aware path planning, coverage control for multi-robot

Contents

List of Figures

List of Tables

Abstract

The area of wireless sensor networks has flourished over the past decade due to advances in micro-electro-mechanical sensors, low power communication and computing protocols, and embedded microprocessors. Recently, there has been a growing interest in mobile sensor networks, along with the development of robotics, and mobile sensor networks have enabled networked sensing system to solve the challenging issues of wireless sensor networks by adding mobility into many different applications of wireless sensor networks. Nonetheless, there are many challenges to be addressed in mobile sensor networks. Among these, the estimation for the exact location is perhaps the most important to obtain high fidelity of the sensory information. Moreover, planning should be required to send the mobile sensors to sensing location considering the region of interest, prior to sensor placements. These are the fundamental problems in realizing mobile sensor networks which is capable of performing monitoring mission in unstructured and dynamic environment.

In this dissertation, we take an advantage of mobility which mobile sensor networks possess and develop localization and path planning algorithms suitable for mobile sensor networks. We also design coverage control strategy using resource-constrained mobile sensors by taking advantages of the proposed path planning method.

The dissertation starts with the localization problem, one of the fundamental issue in mobile sensor networks. Although global positioning system (GPS) can perform relatively accurate localization, it is not feasible in many situations, especially indoor environment and costs a tremendous amount in deploying all robots equipped with GPS sensors. Thus we develop the indoor localization system suitable for mobile sensor networks using inexpensive robot platform. We

focus on the technique that relies primarily on the camera sensor. Since it costs less than other sensors, all mobile robots can be easily equipped with cameras. In this dissertation, we demonstrate that the proposed method is suitable for mobile sensor networks requiring an inexpensive off-the-shelf robotic platform, by showing that it provides consistently robust location information for low-cost noisy sensors.

We also focus on another fundamental issue of mobile sensor networks which is a path planning problem in order to deploy mobile sensors in specific locations. Unlike the traditional planning methods, we present an efficient cost-aware planning method suitable for mobile sensor networks by considering the given environment, where it has environmental parameters such as temperature, humidity, chemical concentration, stealthiness and elevation. A global stochastic optimization method is used to improve the efficiency of the sampling based planning algorithm. This dissertation presents the first approach of sampling based planning using global tree extension.

Based on the proposed planning method, we also presents a general framework for modeling a coverage control system consisting of multiple robots with resource constraints suitable for mobile sensor networks. We describe the optimal informative planning methods which deal with maximization problem with constraints using global stochastic optimization method. In addition, we describe how to find trajectories for multiple robots efficiently to estimate the environmental field using information obtained from all robots.

Keywords: Indoor localization system, Cost-aware path planning, coverage control for multi-robot

Chapter 1

Introduction

1.1 Mobile Sensor networks

As the future of computing moves toward pervasive and ubiquitous platforms, we find ourselves in need of hardware, software, protocols and methodologies that promote their practical use. Wireless sensor networks (WSNs) are a perfect example of this developing technology. WSNs, which are new information technologies, collect information over the physical world via wireless sensing devices (sensor nodes or mote) which is deployed in the physical world and have already demonstrated their utility in a wide range of applications for monitoring, event detection, and control, including environment monitoring, building comfort control, traffic control, manufacturing and plant automation, and military surveillance applications (see [1] and references therein). As WSNs and other embedded computing technologies continue to evolve, it is imperative that we stay abreast of the challenges that arise in order to enable a seamless transfer to general public utilization. The greatest challenge we face when working with WSNs is their resource limitations. Memory, processor, communication rage, power supply, and

hardware quality have all been minimized in order to develop inexpensive, general-use sensor nodes with small form factors. The advantage of this approach is that hundreds of these devices can be deployed over a wide (possibly remote) area, at little cost, and handle tasks for which PC-class devices would not be well suited. Sensor network deployments are often determined by the application. Nodes can be placed in a grid, randomly, surrounding an object of interest, or in countless other arrangements. In many situations, an optimal deployment is unknown until the sensor nodes start collecting and processing data. For deployments in remote or wide areas, rearranging node positions is generally infeasible. Furthermore faced with the uncertain nature of the environment, stationary sensor networks are sometimes inadequate. However, when nodes are mobile, redeployment is possible. In fact, it has been shown that the integration of mobile entities into WSNs improves coverage, and hence, utility of the sensor network deployment and shows superior performance in terms of its adaptability and high-resolution sampling capability [2]. Mobility enables more versatile sensing applications as well.

Mobile sensor networks (MSNs) are a distributed collection of mobile robots each of which has sensing, computation, wireless communication, and mobility capabilities. This network of mobile robots with sensors is usually deployed in a large geographical area and collaborates among themselves to form a wireless sensor network in performing collaborative sensing to monitor and improve the quality of sensing of the environment. MSNs can efficiently acquire information by increasing sensing coverage both in space and time, thereby resulting in robust sensing under the dynamic and uncertain environments. While a mobile sensor network shares the same limitations of wireless sensor networks in terms of its short communication range, limited memory, and limited computational power, it can perform complex tasks, ranging from scouting and reconnaissance to en-

vironmental monitoring and surveillance by cooperating with other agents as a group. There is a growing interest in mobile sensor networks and it has received significant attention recently [3, 4, 5, 6, 7].

1.1.1 Challenges

Since WSNs consist of static sensors, several assumptions (e.g., known sensor position and static topology) are required to obtain accurate information from our environment. Thus it is important to understand how such assumptions change when mobility is integrated into the sensor network.

- Localization. One of the most significant challenges for MSNs is to know the exact locations of mobile sensors. The reliability of quality of information obtained from sensors depends on how exact the location of sensor is. In statically deployed sensor networks, Sensor position can be determined once when initially deployed since there is no change in movement. However in MSNs, location information for mobile sensors must be updated continuously as they frequently transfer to different locations to cover the sensing region. It requires real-time localization service, as well as time and energy.

- Path planning. Mobile sensors usually work in dynamic environment, so they should be moved from one location to another to obtain more information. Planning procedures should be performed prior to coordination of mobile sensors. Path planning has been studied extensively, however, many of the published techniques are inefficient for MSNs because they focus on the length or time of the path or time without considering the sensing region.

- Coverage control. MSNs can obtain the optimal information without requir-

ing dense placement of sensors. However since mobile sensors are resource-constrained, traversing to cover the whole region causes unnecessary energy consumption. Therefore, mobile sensors should be controlled to obtain as much information as possible with the minimum energy. There has been many researches for coverage control problem, but most of them is limited to the discretized space, so they cannot be easily applied to MSNs.

These are the goal of this dissertation. The main objectives of this dissertation are

- development of robust indoor localization using inexpensive robotic platform for mobile sensor networks;
- design cost-aware coordination system using mobile sensor nodes based on the environmental parameters; and
- implementation and evaluation of information gathering strategy based on the proposed cost-aware coordination.

There are other challenges in developing mobile sensor network system that are not addressed in this dissertation. On the hardware side, we need an inexpensive mobile sensor node which operates with low power consumption for a long-term deployment. On software side, we need reliable and robust communication and time synchronization protocols.

1.2 Overview of the Dissertation

Chapter 2 explains why existing methods for the challenging issues that are addressed in this dissertation are not suitable for mobile sensor networks and suggest

the proposed methods are the solutions for the challenging issues in mobile sensor networks.

In Chapter 3, we describe a vision-based coordinated localization algorithm for mobile sensor networks with camera sensors to operate under GPS denied areas or indoor environments. Mobile robots are partitioned into two groups. One group moves within the field of views of remaining stationary robots. The moving robots are tracked by stationary robots and their trajectories are used as spatio-temporal features. From these spatio-temporal features, relative poses of robots are computed using multi-view geometry and a group of robots is localized with respect to the reference coordinate based on the proposed multi-robot localization. Once poses of all robots are recovered, a group of robots moves from one location to another while maintaining the formation of robots for coordinated localization under the proposed multi-robot navigation strategy. By taking the advantage of a multi-agent system, we can reliably localize robots over time as they perform a group task. In experiment, we demonstrate that the proposed method consistently achieves a localization error rate of 0.37% or less for trajectories of length between 715 cm and 890 cm using an inexpensive off-the-shelf robotic platform. This chapter is based on [8].

Chapter 4 introduces the related works which evaluates the quality of the path considering the environmental field and the primary algorithms which form the basis of the proposed planning methods, which are sampling based path planning algorithms and stochastic optimization based path planing algorithm.

In Chapter 5, we develop a cost-effective motion planning method for robots operating in complex and realistic environments. While sampling-based path planning algorithms, such as rapidly-exploring random tree (RRT) and its variants, have been highly effective for general path planning problems, it is still difficult

to find the minimum cost path in a complex space efficiently since RRT-based algorithms extend a search tree locally, requiring a large number of samples before finding a good solution. This chapter presents an efficient nonmyopic path planning algorithm by combining RRT* and a stochastic optimization method, called cross entropy. The proposed method constructs two RRT trees: the first tree is a standard RRT* tree which is used to determine the nearest node in the tree to be extended to a randomly chosen point and the second tree contains the first tree with additional long extensions. By maintaining two separate trees, we can grow the search tree non-myopically to improve the efficiency of the algorithm while ensuring the asymptotic optimality of RRT*. From an extensive set of simulations and experiments using mobile and humanoid robots, we demonstrate that the proposed method consistently finds a path with the lowest cost faster than existing algorithms. This chapter is based on [9].

Chapter 6 presents a novel informative path planning algorithm using an active sensor for efficient environmental monitoring. While state-of-the-art algorithms find the optimal path in a continuous space using sampling-based planning method, such as rapidly-exploring random graphs (RRG), there are still some key limitations, such as computation complexity and scalability. We propose an efficient information gathering algorithm using RRG and a stochastic optimization method, cross entropy (CE), to estimate the reachable information gain of each node of the graph. The proposed algorithm maintains the asymptotic optimality of RRG and finds the most informative path satisfying the cost constraint. We demonstrate that the proposed algorithm finds a (near) optimal solution efficiently compared to the state-of-the-art algorithm and show the scalability of the proposed method.

Chapter 2

Background

We deal with challenging fundamental issues of mobile sensor networks such as localization, path planning, and coverage control in this dissertation. Many researchers have extensively studied to solve those problems. However, they have focused on solving problems without considering conditions which mobile sensor networks require. In this chapter, we introduce the existing methods for solving localization, path planning, and coverage control problems and their limitations, and suggest that the proposed methods are suitable for mobile sensor networks.

2.1 Localization in MSNs

In order to perform sensing or coordination using mobile sensor networks, localization of all sensor nodes is of paramount importance. A number of localization algorithms have been proposed for stationary sensor networks, e.g., [10, 11]. At present, the most widely used method for localization is NAVSTAR Global Positioning System (GPS) [12]. Approximately 24 satellites included in the system orbit the planet while transmitting the signal consistently. The location information is determined using signals from at least four satellites (one signal is used to

adjust the local clock uncertainty) based on trilateration method. Commercial use GPS has less than 10 meters and 0.1 microsecond accuracy in position and time synchronization, respectively. But they are applicable for outdoor environment and precise indoor localization is still a challenging problem [13, 14]. (For more information about various localization methods for wireless sensor networks, see references in [10, 11, 13, 14].) One promising approach to indoor localization is based on the ultra-wideband (UWB) radio technology [15]. But as stated in [15], the minimum achievable positioning error can be in the order of 10 cm's and it is not accurate enough to control and coordinate a group of robots. In addition, the method requires highly accurate time synchronization. In order to address these issues, UWB based localization is combined with infrared sensors using a team of mobile agents in [16]. However, it requires the deployment of UWB detectors in advance, which is not suitable for mobile sensor networks operating under uncertain or unstructured environments.

Localization using camera sensors has been widely studied in the computer vision community. Taylor et al. [17] used controllable light sources to localize sensor nodes in a stationary camera network. A distributed version of camera localization is proposed by Funiak et al. [18], in which relative positions of cameras are recovered by tracking a moving object. The sensor placement scheme is presented for the problem of minimizing the localization uncertainty in [19]. They proposed a triangulation-based state estimation method using bearing measurements obtained from two sensors. Meingast et al. [20] proposed a multi-target tracking based camera network localization algorithm. The critical concept applied in [20] is the use of spatio-temporal features, an approach taken in this dissertation. Tracks of moving objects are used as spatio-temporal features (tracks are detected by a multi-target tracking algorithm from [21]). In order to find matching

features over a pair of cameras, detection times of spatial features are used as well as spatial features such as Harris corners and scale-invariant feature transform (SIFT) keypoints [22]. Then the relative position and orientation between cameras are computed using multi-view geometry. Since an incorrect matching between spatio-temporal features is extremely rare compared to spatial features, the method provided outstanding performance under a wide baseline and varying lighting conditions.

But the aforementioned methods are designed for stationary camera networks and are not suitable for dynamic mobile sensor networks. In fact, in mobile sensor networks, we can take the advantage of mobility to improve the efficiency of localization. For instance, Zhang et al. [23] proposed a method to control the formation of robots for better localization. They estimated the quality of team localization depending on the sensing graph and the shape of formation. A multi-robot localization algorithm based on the particle filter method is presented in [24]. They proposed a reciprocal sampling method which selects a small number of particles when performing a localization process. Some authors have considered cooperative localization of multiple robots using bearing measurements. Giguere et al. [25] addressed the problem of reconstructing relative positions under the condition of mutual observations between robots. The constraint was later relaxed by adding landmarks in [26]. They used nonlinear observability analysis to derive the number of landmarks needed for full observability of the system and an extended information filter was applied to estimate the states of a team of robots using bearing-only measurements. Ahmad et al. [27] applied a cooperative localization approach to robot soccer games. They modeled the problem as a least squares minimization and solved the problem using a graph-based optimization method, given static landmarks at known positions. Tully et al. [28]

used a leap-frog method for a team of three robots performing cooperative localization, which is similar to the proposed method. In [28], two stationary robots localize the third moving robot from bearing measurements using an extended Kalman filter. After completing a single move, the role of each robot is switched and the process is repeated. In their experiments, robots covered a region of size $20\,m \times 30\,m$ and showed a localization error of $1.15\,m$ for a trajectory of length approximately $140\,m$. However, the experiments were conducted using an expensive hardware platform including three on-board computers, four stereo cameras, and a customized ground vehicle with many sensors. Hence, it is unclear if the approach is suitable for an inexpensive off-the-shelf robotic platform considered in this dissertation.

Therefore, we propose a coordinated localization algorithm for mobile sensor networks under GPS denied areas or indoor environments using an inexpensive off-the-shelf robotic platform. From experiments, we show that the proposed localization method provides consistently robust location information for low-cost noisy sensors. By implementing the leap-frog method [28] using the same robotic platform used in this dissertation, the proposed method achieves a localization error rate which is more than 15 times better than the leap-frog method for trajectories with a similar length.

2.2 Path planning in MSNs

Path planning has attracted much attention in the field of robotics due to its importance. The goal of a path planning algorithm is to find a continuous trajectory of a robot from an initial state to a goal state without colliding with obstacles while maintaining robot-specific constraints. A popular path planning algorithm is the rapidly-exploring random tree (RRT) [29] which is a sampling

based method. It is applied and extended by many researchers under static environments [30, 31] and dynamic environments [32, 33].

In contrast to RRT, which solves a single query problem, the probabilistic roadmap (PRM) [34] is another sampling based path planning algorithm which is applied to solve multiple query problems. However, since the performance is determined by the number of samples, importance sampling based PRM approaches have been proposed in order to select more samples near the region of interest (see [35] and references therein). Recently, cross entropy (CE) [36], a combination of importance sampling and optimization, has been applied to path planning problems [37].

However, the aforementioned methods are not suitable for mobile sensor networks which requires mobile sensors to be deployed in specific locations since they do not account for cost which may accumulate in the given environment as a robot moves. For instance, consider a nuclear power plant accident scenario, in which some regions are contaminated by radioactive materials. In this scenario, MSNs require robots to be deployed in disaster or hazardous environments while performing search-and-rescue operation. If a map of radioactive levels is available, it is desirable for robots to perform the search-and-rescue operation along the path with the minimum exposure to radioactive materials. Finding the minimum exposure path becomes a difficult problem if the radioactive map shows complex terrains of radioactive levels, in addition to obstacles present in the field. To handle such case a number of improvements have been made and applied to more complex cost maps in recent years [38, 39, 40, 41, 42, 43]. In order to increase the quality of a path, the nearest node of the tree to a random point is selected by computing the cost along the path when RRT extends a tree in [38, 41]. Ferguson et al. [39] modified the procedures of RRT such as

random sampling, node selection for extension, and node extension when generating RRT trees sequentially to improve the quality of the resulting path. Ettlin et al. [40] applied RRT to find the paths having low cost in rough terrain. They computed the cost of trajectories by biasing RRT towards low-cost areas. In [42], mobile robots are used to estimate environmental parameters of the field and coordinated towards the location with the most information. However, they did not consider the information gain along the trajectories of robots. In [43], Jaillet et al. presented the transition-based RRT (T-RRT) which finds low-cost paths with respect to a user given configuration space cost map based on a stochastic optimization method. However, since [43] extends the tree using a finite set of possible controls, the resulting paths can be suboptimal.

Motivated by this, we propose an efficient cost-aware planing strategy suitable for mobile sensor networks by considering the given environment, where it has environmental parameters such as temperature, humidity, chemical concentration, stealthiness and elevation.

2.3 Informative path planning in MSNs

In recent years, environmental monitoring has become increasingly important due to factors, such as global warming, ozone layer depletion, deforestation, ocean pollution, natural resource depletion, and population growth, to name a few. A number of different environmental monitoring areas have been studied extensively, including marine monitoring [44], ecological monitoring [45], aerial monitoring [46], and disaster monitoring [47, 48].

In order to monitor a large area, it is important to collect the most useful information with available sensing resources. Guestrin et al. [49] proposed a method for placing static sensors using the entropy while modeling the environmental

parameter using a Gaussian process. However, it requires a dense deployment of sensors to avoid sensing holes and static sensors are not suitable for time-varying processes we often find in nature. To overcome the limitations of static sensors, mobile sensor networks are introduced to increasing the sensing coverage both in space and time.

With mobile sensor networks, active sensing is possible by taking advantage of the mobility of mobile agents. We will call the problem of scheduling the trajectory of a mobile sensor for collecting the most useful information as an informative path planning (IPP) problem in this dissertation.

Singh et al. [47] presented an efficient informative planning strategy for maximizing the mutual information from a team of robots. In [50], an optimal IPP algorithm using branch and bound was proposed to maximally reduce the variance of the field of interest based on exhaustive search. However, aforementioned methods were applied to discretized search spaces, making it less scalable for large problems. A sampling based path planning method, a rapidly-exploring random tree (RRT) [29], has been applied to the IPP problem. Kwak et al. [51] applied a genetic algorithm to decide which node to extend an RRT tree for information gathering. A mobile target tracking controller was developed to maximize the information gain using a limited number of mobile sensors in [52], where the tracking accuracy along the path represents the information gain. While the recently proposed RRT* [53] seems like a good candidate for solving an IIP problem, Bry et al. [54] has shown that RRT* is not suitable as for solving an IPP problem with a cost constraint. The two key procedures of RRT*, selecting the parent of any newly inserted node and rewiring the node and any nodes of an RRT tree, are performed based on the consumed costs along the path to the node. However, those procedures in IPP does not work since those procedures

are affected by not only the information gain at a node of an RRT tree but also the cost constraint at the node.

Motivated by this fact, Hollinger et al. [55] proposed a rapidly-exploring information gathering (RIG) algorithm, which combines sampling-based motion planning with combinatorial optimization. Since it follows the overall structure of rapidly-exploring random graphs (RRG) [53], it can ensure that the optimal path will be found as the number of samples approaches infinity. It introduced a pruning strategy to improve the efficiency by reducing co-located nodes if they cannot lead to the optimal solution. The pruning strategy requires to know the upper bound on the possible reachable information gain using the given cost budget for each node. The reachable information gain is used as the upper bound and it is computed using a branch and bound algorithm [50]. However, while robots move in a continuous space, the reachable information gain is calculated by discretizing the state space. Hence, it requires heavy computation to perform the branch and bound algorithm to compute all reachable information gains for all nodes in a tree for a complex or large problem.

Therefore, we propose an efficient informative path planning suitable for mobile sensor networks by modeling a coverage control system consisting of multiple robots with resource constraints based on the proposed path planning method.

Chapter 3

Robust Indoor Localization

In this chapter, we present a vision-based coordinated localization system suitable for mobile sensor networks under GPS denied areas or indoor environments using inexpensive robot platform. We take the advantage of mobile sensor networks. In order to localize mobile robots, we first partition robots into two groups: stationary robots and moving robots. We assume each robot carries a camera and two markers[1]. The moving robots move within the field of views (FOVs) of stationary robots. The stationary robots observe the moving robots and record the positions of markers of moving robots. Based on the trajectories of markers, i.e., spatio-temporal features, we localize all the robots using multi-view geometry. Localization requires recovering relative positions, i.e., translation and orientation. While the translation between cameras can be recovered only up to a scaling factor in [20], we can recover the exact translation using the known distance between markers in the proposed algorithm.

A multi-robot navigation strategy is also developed using the rapidly-exploring random tree (RRT) [29], which moves a group of robots from one location to

[1]For robots moving on a flat surface, a single marker with a known height can be used.

another while maintaining the formation of robots for coordinated localization. Since the proposed localization algorithm requires a certain configuration of a robot team for good localization, the RRT algorithm is modified to guarantee the configuration condition.

We have implemented the proposed algorithm on a mobile robot platform made from an iRobot Create [56] mobile robot and conducted an extensive set of experiments. From experiments, we have discovered a set of configurations of robots, from which good localization is possible. We then applied these configurations in our coordinated multi-robot localization algorithm. Our experimental results show that the proposed method consistently achieves less than $1\,cm$ of localization error for trajectories of length less than $100\,cm$ and a localization error rate of 0.37% or less for longer trajectories with length between $715\,cm$ and $890\,cm$, making it a promising solution for multi-robot localization in GPS denied or unstructured environments.

In order to compare the performance of the proposed method, we have also implemented the leap-frog method [28] using the same robotic platform used in this dissertation. From experiments, the leap-frog method gives a localization error rate of 5.6% for trajectories with the average length of $820.6\,cm$. The proposed method achieves a localization error rate which is more than 15 times better than the leap-frog method for trajectories with a similar length.

3.1 An Overview of Coordinated Multi-Robot Localization

This section gives an overview of the proposed coordinated multi-robot localization method. Suppose there are N robots and we index each robot from 1 to

N. We assume that each robot's locational configuration is determined by its position and orientation in the reference coordinate system. Then the goal of the multi-robot localization problem is to estimate positions and orientations of all robots over time.

Let $X_i(k) = (P_i(k), R_i(k))$ be the locational configuration of robot i at time k with respect to the reference coordinate system, where $P_i(k) \in \mathbb{R}^n$ and $R_i(k) \in SO(3)$ are the position and rotation of robot i at time k, respectively.[2] Then the configuration of a multi-robot system at time k is

$$X(k) = (X_1(k), X_2(k), \ldots, X_N(k)).$$

The multi-robot localization problem is to estimate $X(k)$ for all k from sensor data.

Suppose that we have $X(k-1)$ with respect to the reference coordinate system and computed relative positions, $T_{ij}(k)$, and orientations, $R_{ij}(k)$, for a pair of robots i and j at time k. Then we can easily compute positions and orientations of all robots with respect to a single robot of choice. In order to map new positions of robots in the reference coordinate system, we require that there is at least one robot i such that $X_i(k) = X_i(k-1)$. Then taking positions with respect to this robot, we can recover the positions and orientations of all robots at time k with respect to the reference coordinate system.

Based on this idea, we develop a coordinated localization algorithm. At each time instance, we fix robot s and move other robots. Then we compute $T_{ij}(k)$ and $R_{ij}(k)$ for pairs of robots such that the pose of a robot can be computed with respect to robot s. Finally, we compute $X(k)$ based on $X_q(k-1)$. For $k+1$, we fix another robot r and move remaining robots and continue this process. By doing so, we can continuously estimate $X(k)$ for all times.

[2] $SO(3)$ is the special orthogonal group in $\mathbb{R}^3$ (the group of 3D rotations).

Now the remaining issue is how to estimate translations $T_{ij}(k)$ and orientations $R_{ij}(k)$ for pairs of robots. For this task, we make the following assumptions:

- Each robot carries a camera and markers.
- The internal parameters of cameras are known (e.g., focal lengths, principal points, and distortion coefficients).
- Each robot communicates with other robots via wireless communication.
- The clocks of all robots are synchronized.
- Either the distance between a pair of markers on a robot is known or the height of a single marker is known when a robot is moving on a flat surface.
- At least two robots which capture images are stationary.

Figure 3.1 illustrates an overview of our method. Robots carrying markers move within the FOVs of stationary robots. Each stationary robot captures an image, detects markers, and localizes positions of markers in its image frame at time k. The marker positions and image capture times are shared with other stationary robots. For a pair of stationary robots i and j, we can compute the relative translation $T_{ij}(k)$ and orientation $R_{ij}(k)$ from a pair of marker trajectories using multi-view geometry as discussed in Section 3.2. At time $k+1$, every stationary robot except at least one robot moves and repeats the same process. There is one remaining issue which is that we can only recover the relative positions up to a scaling factor when only images are used. To recover the absolute translation value, we need a known length. To resolve this issue, we assume that the markers on robots are placed at known heights.

Since the minimum number of robots required for the proposed coordinated localization algorithm is three, we will discuss our method using a mobile sensor

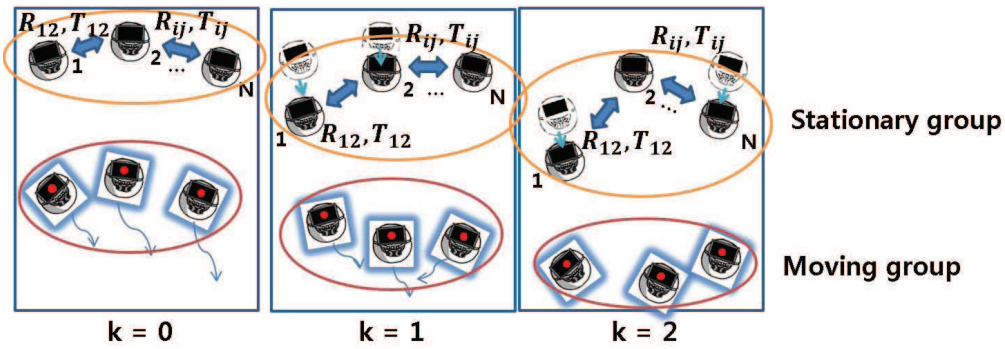


Figure 3.1: An overview of the proposed coordinated multi-robot localization algorithm. Robots in a moving group move within the field of view of stationary robots. Robots in a stationary group track markers of moving robots and exchange marker positions. The translations and orientations among stationary robots are computed from collected marker tracks using multi-view geometry. Finally, all robots are localized with respect to the reference coordinate system based on the position of at least one fixed robot since last update time.

network of three robots for the ease of exposition in this dissertation. However, the proposed method can be applied to a multi-robot system with a larger number of robots. Furthermore, while a single moving robot is used in our discussion and experiment, the method can be likewise applied to the case with multiple moving robots using the multi-target tracking method of [20].

3.2 Multi-Robot Localization using Multi-View Geometry

In this section, we focus on a single step of the coordinated multi-robot localization algorithm for localizing stationary robots by tracking a moving robot and

present how to control the moving robot using visual data from stationary robots.

3.2.1 Planar Homography for Robot Localization

When two cameras view the same 3D scene from different viewpoints, we can construct the geometric relation between two views using the homography if the scene is planar. The homography between two views can be expressed as follows:

$$H = K\left(R + \frac{1}{d}TN^T\right)K^{-1}, \tag{3.1}$$

where $K \in \mathbb{R}^{3\times3}$ is the intrinsic calibration matrix of the camera, $R \in SO(3)$ is the rotation matrix, $T \in \mathbb{R}^3$ is the translation vector, N is the unit normal vector of the plane with respect to the first camera frame, and d is a scale factor which represents the distance from the plane to the optical center of the first camera [57].

We can recover $\{R, T, N\}$ up to a scale factor from H using the singular value decomposition. From this derivation, we obtain two possible solutions [57]. In this application, corresponding points are located on the plane which is parallel to the ground and the tilted angle of each camera is fixed, so we can compute the normal vector of the plane. Among two solutions, we can find a unique solution since the normal vector which is perpendicular to the plane is available in our case. As explained in [57], from the singular value decomposition of H^TH, we obtain an orthogonal matrix $V \in SO(3)$, such that $H^TH = V\sum V^T$, where $V = [v_1, v_2, v_3]$. Let u be a unit-length vector such that $N = v_2 \times u$ and $v_2^Tu = 0$ and v_2 is orthogonal to N. Therefore, given v_2 and N, we can solve for u. Once we find u, we can form the new orthonormal basis $\{v_2, u, N\}$ and obtain R and T as follows:

$$R = WU^T \qquad \text{and} \qquad T = d(H - R)N, \tag{3.2}$$

where $U = [v_2, u, N]$ and $W = [Hv_2, Hu, \widehat{Hv_2}Hu]$, and $\hat{x} \in \mathbb{R}^{3\times3}$ is a skew-symmetric matrix. When we reconstruct positions of markers in the 3D space using data points from the image plane, we can find the exact scale factor using the distance between markers.

The homography can be computed from a number of pairs of feature correspondences. We use the spatio-temporal correspondence features by tracking a moving robot in the FOVs of scenes. We first segment a marker on the robot at each time instance from image sequences of each camera. It is performed by applying the maximally stable extremal regions (MSER) detector [58], a blob detection method. A centroid of the marker is used to build a marker image track. Even though the above detection algorithm for extracting marker positions of the moving robot is robust, it may contain outliers, which do not fit the 2D plane, such as debris on the ground or measurement errors. In order to robustly estimate the planar homography, we used the random sampling consensus (RANSAC) algorithm [59].

3.2.2 Image Based Robot Control

The trajectory of a moving robot can influence the quality of localization. Hence, it is desirable to move a robot such that the localization error can be minimized. Since a robot has to be controlled using data from cameras, it can be seen as a visual servo control problem. In visual servo control, namely the image-based visual servo (IBVS) approach, the control input to the moving robot is computed based on the error generated in the 2D image space [60]. However, in order to compute the control input, visual servoing requires the pseudo inverse of an interaction matrix which represents the relationship between the velocity of the moving robot and the time derivative of the error. Since the interaction matrix

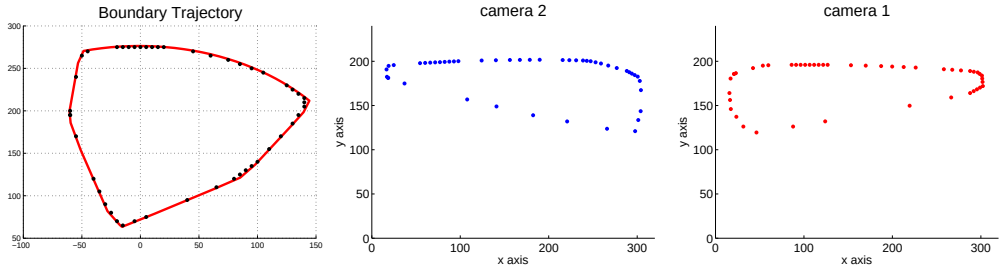
has six degrees of freedom, such a process requires at least three feature points at each time. Since we only consider a single marker in the present dissertation, the visual servoing approach is not applicable and an alternative robot controller is required.

Robot trajectory design for better localization

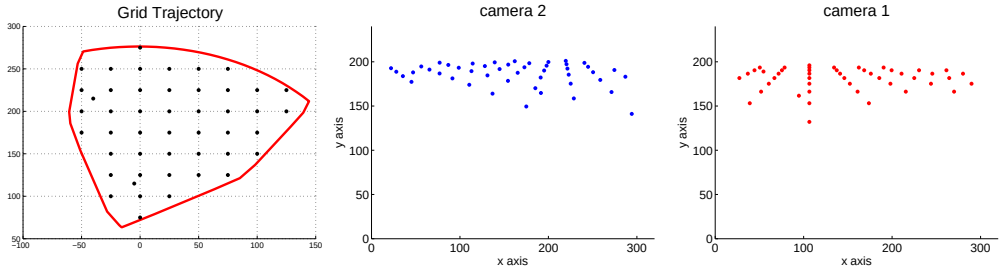
We have considered three scenarios in order to identify an ideal robot trajectory for minimizing the localization error and they are shown in Figure 3.2. The considered cases are (1) points along the boundary of the common FOV by two cameras (Figure 3.2(a)), (2) uniformly scattered points inside the FOV (Figure 3.2(b)), and (3) randomly placed points inside the FOV (Figure 3.2(c)). For each case, we randomly selected 50 point pairs and performed the proposed localization algorithm. We have repeated the process for 500 times. For each run, we computed the estimation error $\epsilon_i = |\hat{d}_i - d_{true}|$, for $i = 1, 2, \ldots, 500$, where $\hat{d}_i$ is the estimated distance between two robots for the i-th run using our localization method and d_{true} is the ground truth distance. Average localization errors are shown in Figure 3.3. It can be seen that points along the boundary of the FOV gives the lowest localization error. The simulation was conducted using MATLAB based on parameters of cameras used in experiments. A point is projected to the image plane with an additive zero-mean Gaussian noise with a standard deviation of one pixel.

Control

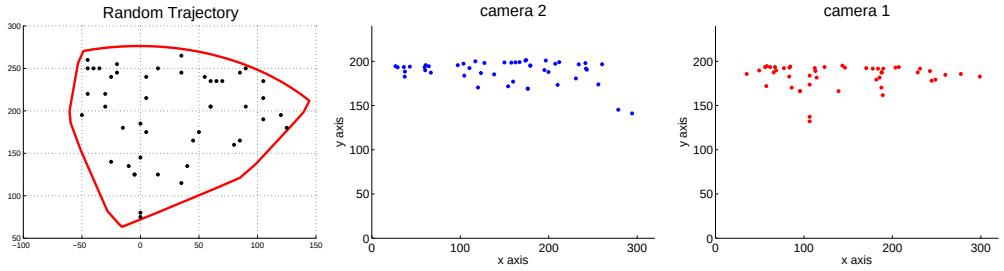
From the previous section, we have found that we can lower the localization error using feature points located at the boundary of the FOV. Based on this finding, we design an image-based robot controller which makes the moving robot follow



(a) Boundary points on the plane (top view) and image views.



(b) Uniform points on the plane (top view) and image views.



(c) Random points on the plane (top view) and image views.

Figure 3.2: Examples of three scenarios for an ideal trajectory: (a) boundary, (b) uniform, and (c) random. The left figure shows the top view and the middle and right figures show two separate camera views.

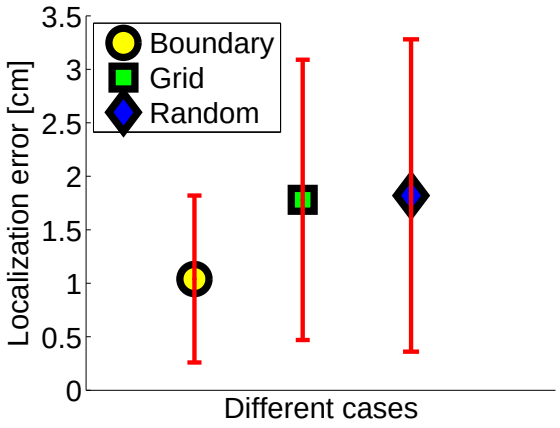
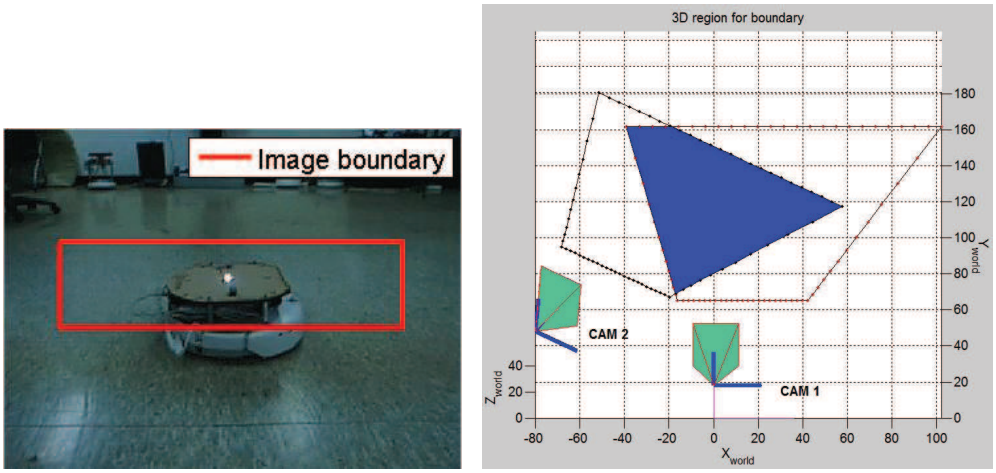


Figure 3.3: Average localization errors for three different scenarios considered in Figure 3.2. The average error is computed from 500 independent runs and the error bar shows one standard deviation from its mean value.

the boundary of the common FOV of stationary robots. One difficulty is that a moving robot can move beyond the FOV due to communication delay. In order to prevent this problem, we first set the boundary within the image frame as shown in Figure 3.4(a). The common FOV of two cameras on the ground plane is shown in Figure 3.4(b).

Since the actual heading of a moving robot is not available, it has to be estimated from image data. We estimate the heading direction of the moving robot using a batch least square filter over a finite window from measurements from both cameras. When the moving robot is near the boundary of a camera, the corresponding stationary robot sends a command to the moving robot to rotate by a predefined amount for a short duration. The direction of the rotation is determined by the normal vector of the boundary and the estimated heading direction. The moving robot combines commands from stationary robots and changes its



(a) (b)

Figure 3.4: (a) An example of a boundary within an image frame. (b) The common FOV of two cameras on the ground plane.

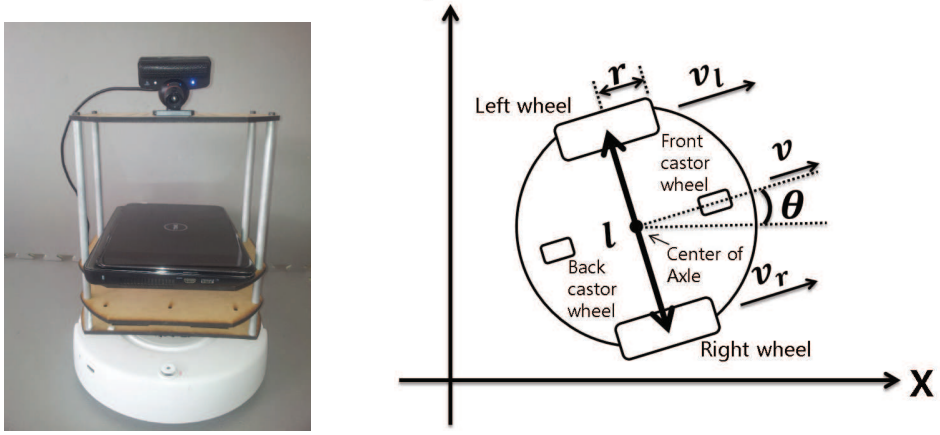
heading for a short duration. While this is an extremely simple controller, we have found it very effective for controlling the robot to move along the boundary of the FOV since we cannot reliably estimate the position and heading of the moving robot using a small number of noisy marker detection results.

3.3 Multi-Robot Navigation System

This section details how we implement the multi-robot navigation system including the robot platform used in the experiments and a multi-robot navigation method which moves a group of robots from one location to another while maintaining the formation of robots for coordinated localization.

3.3.1 Multi-Robot System

For our experiments, we used iRobot Create wheeled mobile robots [56] as mobile nodes in our mobile sensor network. The developed mobile platform is shown in Figure 3.5(a), which is equipped with a PS3 Eye camera, an ASUS notebook which runs Linux OS, and a white LED which works as a marker. WiFi (IEEE 802.11) is used for communication among robots. Each camera has a resolution of 320×240 pixels and runs at 40 frames per second (fps). As explained in [61], we used a one-server, two client model for communication. However, unlike [61], in this work, stationary robots consistently check the visibility of the moving robot to prevent the moving robot from going beyond the common FOV as explained in the previous section. The time synchronization operation is implemented as follows. Since each robot can have a different local time, all clocks are synchronized using the clock of the server at each time a stationary robot moves.



(a) (b)

Figure 3.5: (a) An iRobot Create based mobile robot platform. (b) A two-wheeled differential drive robot.

We used the two-wheeled differential drive robot dynamics in simulation and experiments. The parameters of the mobile robot are shown in Figure 3.5(b). Let l be the distance between the two wheels. Recall that q_x and q_y denote the position of a robot with respect to x and y axis, respectively, and q_θ denotes its heading. The dynamics of a two-wheeled differential drive robot can be expressed as follows, where v_r and v_l are the right and left wheel velocities, respectively.

$$
\begin{aligned}
&q_x(t+\Delta t) = \\
&\quad \begin{cases} q_x(t) + \frac{2v(t)}{w(t)} \sin\left(\tilde{w}(t)\right) \cos\left(\theta(t) + \tilde{w}(t)\right) & \text{if } w(t) \neq 0 \\ q_x(t) + v(t)\Delta t \cos\big(\theta(t)\big) & \text{otherwise} \end{cases} \\
&q_y(t+\Delta t) = \\
&\quad \begin{cases} q_y(t) + \frac{2v(t)}{w(t)} \sin\left(\tilde{w}(t)\right) \sin\left(\theta(t) + \tilde{w}(t)\right) & \text{if } w(t) \neq 0 \\ q_y(t) + v(t)\Delta t \sin\big(\theta(t)\big) & \text{otherwise} \end{cases} \\
&q_\theta(t+\Delta t) = q_\theta(t) + w(t)\Delta t, \qquad (3.3)
\end{aligned}
$$

where $\tilde{w}(t) = \frac{w(t)\Delta t}{2}$, $v(t) = \frac{v_r(t)+v_l(t)}{2}$ is the translational velocity, and $w(t) = \frac{v_r(t)-v_l(t)}{l}$ is the angular velocity [62].

We can not directly specify v and w to reach the specific position, because we can not compute v_r and v_l from (3.3). Thus, we consider only two motions by a robot (going-forward and turning) to reduce the modeling error and simplify its control. In order to correctly model the physical mobile platform used in the experiment, we have conducted a number of experiments to measure odometry errors. For the going-forward motion, we made a robot move forward at four different distances from $5\,cm$ to $20\,cm$ at a speed of $20\,cm/s$, 15 times each. The odometry error is shown in Figure 3.6(a). It is interesting to note that there is

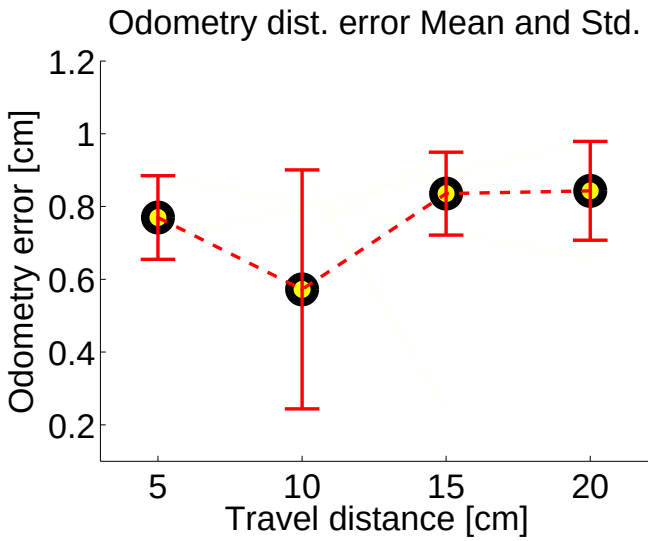
a bias term, i.e., the going-forward motion shows a bias of $0.8\,cm$ at $20\,cm/s$. For the turning motion, we have rotated a robot at different angular velocities 15 times each and the average angular error is shown in Figure 3.6(b). Since we obtained the similar results for the negative angular velocities, the results for the negative are not illustrated in this figure. The average angular error and its variance tend to fluctuate depending on the angular velocity. Especially for angular velocities less than $2\,^\circ/s$, the variance of the angular error is relatively large with respect to the magnitude of the velocity (see the inset in Figure 3.6(b)). But the mean increases as the angular velocity gets bigger, except $10\,^\circ/s$. Based on the experiments, we have obtained a more precise dynamic model of the mobile platform using (3.3). When a robot moves forward, its heading does not change, hence, the dynamics for the going-forward motion is as follows:

$$\begin{aligned} q_x(t+\Delta t) &= q_x(t) + v\cos(\theta(t))\Delta t + \alpha_1 \\ q_y(t+\Delta t) &= q_y(t) + v\sin(\theta(t))\Delta t + \alpha_1 \\ q_\theta(t+\Delta t) &= q_\theta(t) + \alpha_2, \end{aligned} \tag{3.4}$$

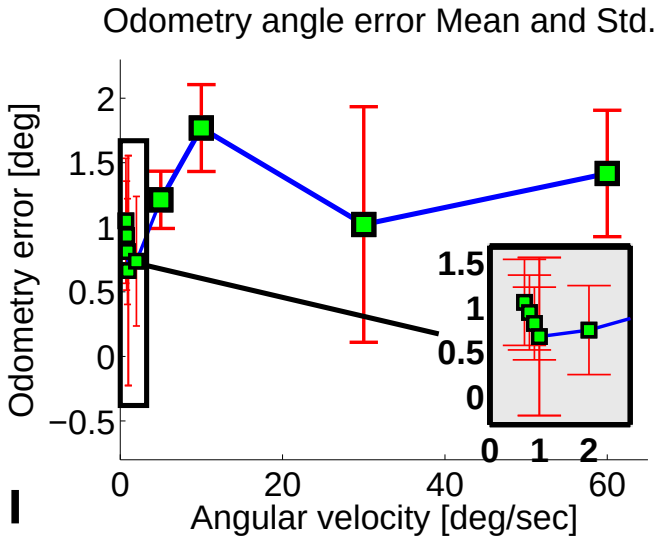
since $v_r = v_l$ and $w = 0$. The dynamics for turning becomes:

$$\begin{aligned} q_x(t+\Delta t) &= q_x(t) + \alpha_3 \\ q_y(t+\Delta t) &= q_y(t) + \alpha_3 \\ q_\theta(t+\Delta t) &= q_\theta(t) + \left(\frac{2v}{l} + \alpha_4\right)\Delta t, \end{aligned} \tag{3.5}$$

since $v_r = -v_l$ and $v = 0$ (i.e., the position of the robot is stationary). Here, $\alpha_1, \alpha_2, \alpha_3$, and α_4 are random variables representing noises including bias terms found from the experiments.



(a)



(b)

Figure 3.6: Average drift error for different movements. The average drift error is computed from 15 movements and one standard deviation is shown as an error bar (3.6(a)-3.6(b)). The inset in (b) is a magnified odometry error (deg) graph for angular velocity less than $2\,^{\circ}/s$.

3.3.2 Multi-Robot Navigation

We now consider moving a group of robots from one location to another location while localizing all robots based on the proposed multi-robot localization algorithm. We develop a multi-robot navigation algorithm based on the rapidly-exploring random tree (RRT) [29] which is a sampling-based path planning method. It quickly searches over a nonconvex configuration space by sampling a random point and incrementally builds a navigation tree by extending the tree towards the random point. While an RRT can be readily applied to a single robot, it is not straightforward to apply to a group of robots with constraints. For our multi-robot localization method, robots must satisfy a requirement about the configuration of robots, namely the distance between stationary robots and angles between them for better localization (see Section 3.4.1 for more information about the constraints).

Let $X_{team}(k)$ be the locational configuration of two stationary robots, i.e. $X_{team}(k) = [X_A(k), X_B(k), \theta(k)]$, where $X_A(k) \in \mathcal{X}$ and $X_B(k) \in \mathcal{X}$ are locational configurations of robots A and B, respectively, at time k, and $\theta(k)$ is the angle between two stationary robots, which is graphically illustrated in Figure 3.7. Recall that $X_i(k)$ consists of the position $P_i(k)$ and the rotation $R_i(k)$. In order for a team of three robots to correctly localize, the following conditions must be satisfied.

$$\begin{aligned} d_1 \leq \|P_A(k) - P_B(k)\| \leq d_2 \\ \theta_1 \leq \theta(k) \leq \theta_2, \end{aligned} \tag{3.6}$$

where $\theta(k)$ is computed using $R_A(k)$ and $R_B(k)$ and the parameters d_1, d_2, θ_1, and θ_2 are experimentally determined as discussed in Section 3.4 and fixed for the team. In order for a group of robots move, one of the stationary robots has to

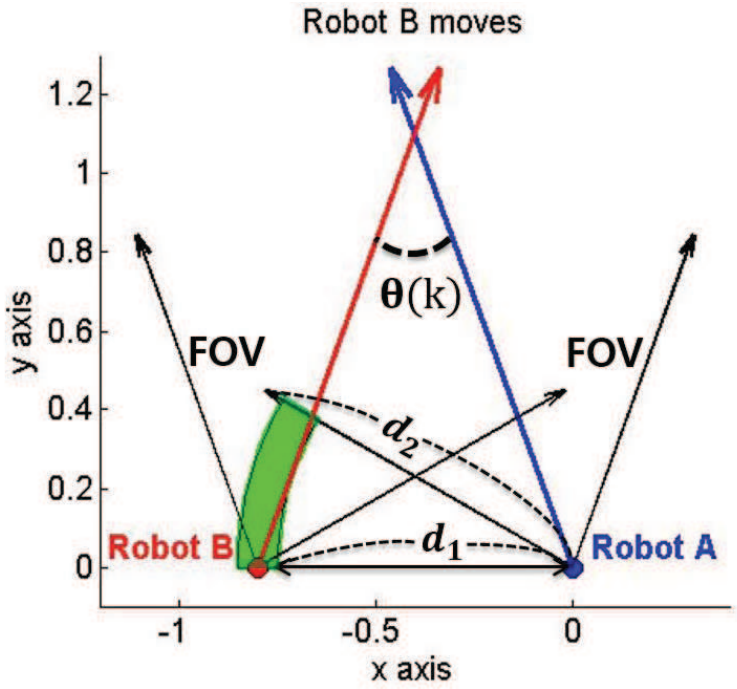


Figure 3.7: Illustration of the region $\mathbb{X}_{\text{move}}(k)$. Blue and red circle represent initial positions of robot A and B and blue and red arrow represent the headings of robots, respectively. Two black arrows with a gray region for each robot represent the field of view of each robot. The region with green color represents $\mathbb{X}_{\text{move}}(k)$.

move and there is a chance that the condition (3.6) can be violated. For a group of robots to navigate while localizing, the condition (3.6) has to be satisfied at all times.

Suppose that robot A is stationary and robot B moves forward. Then the region satisfying the first condition of (3.6) can be expressed as $\mathbb{X}_{\text{move}}(k)$, the green region in Figure 3.7. The second condition of (3.6) can be easily satisfied by rotating robot B with respect to the heading of robot A. Path planning of a team of robots satisfying the condition (3.6) is implemented using the RRT.

However, since one robot moves and the other robot is stationary, we have to alternatively move one robot at a time while satisfying (3.6) at each step. The procedure is similar to scheduling steps of a humanoid robot to move from one location to a target location while avoiding obstacles. Results of RRT based path planning for a robot team are shown in Section 3.4.

3.4 Experimental Results

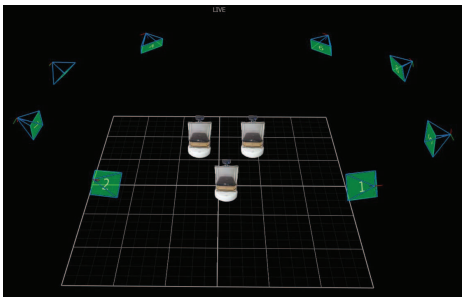
3.4.1 Coordinated Multi-Robot Localization: Single-Step

We first performed experiments for the single-step of the coordinated multi-robot localization algorithm in order to find a multi-robot configuration which results in good localization.

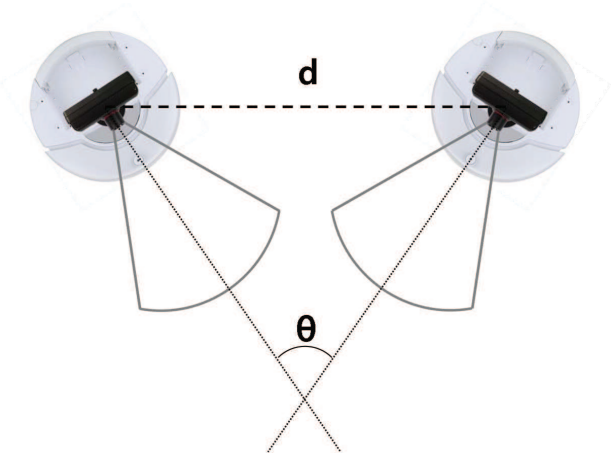
Figure 3.8 shows our experiment setup. We also used the Vicon motion capture system to collect the ground truth data in order to measure the performance of our algorithm. We conducted our experiments at four different baselines, d, between two stationary robots ($d = 60, 80, 100, 120\,cm$). For each baseline, we tested five different angles, θ, between robots ($\theta = 0\,^\circ, 10\,^\circ, 20\,^\circ, 30\,^\circ, 40\,^\circ$). See Figure 3.8(c) for how d and θ are defined. Hence, there is a total of 20 cases. For each case, we collected about 250 marker positions of a moving robot and ran 500 times using RANSAC. Then we localized robots using the proposed algorithm. The estimation error was computed using the ground truth locations obtained from the Vicon motion capture system.

Figure 3.9(a)-3.9(d) show the results of all 20 cases. The distribution of localization error is shown as a histogram for each case. The bin size of a histogram is $0.5\,cm$ and the color of a bin represents the number of runs with localization errors belonging to the bin. When this number is large, the bin color is red and

(a)

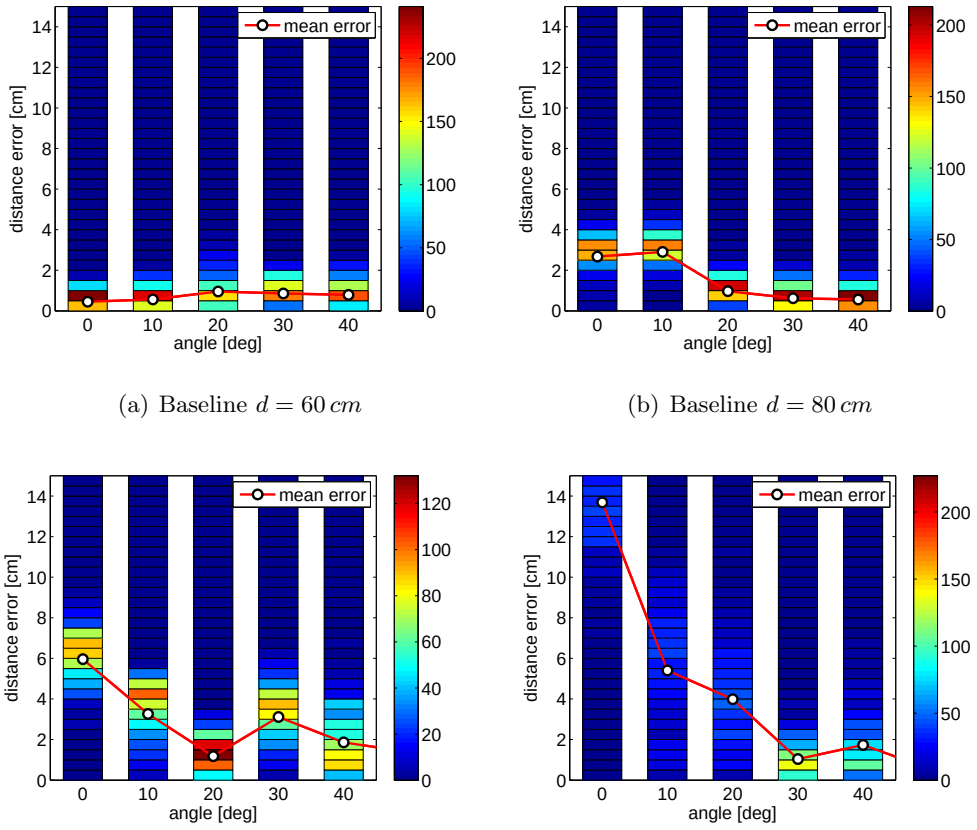


(b)



(c)

Figure 3.8: (a) Photos of the Vicon motion capture system installed in our lab. The Vicon motion capture system is used for providing the ground truth values. (b) An image obtained from the Vicon with robots placed on the reference coordinate system. (c) The experimental setup parameters. d is the distance between two stationary robots and θ is the angle between them.



(a) Baseline $d = 60\,cm$

(b) Baseline $d = 80\,cm$

(c) Baseline $d = 100\,cm$

(d) Baseline $d = 120\,cm$

Figure 3.9: Localization error distributions of 20 cases at different baselines ($d = 60, 80, 100, 120\,cm$) and angles ($\theta = 0\,^\circ, 10\,^\circ, 20\,^\circ, 30\,^\circ, 40\,^\circ$) between robots. Each case has 500 runs. An interval of each bin is $0.5\,cm$ and the color of each bin represents the number of runs with localization error belonging to the interval. A white circle represents the mean error of 500 runs.

the bin color is dark blue when this number is low. For instance, when $d = 60$ and $\theta = 0$, more than 200 runs resulted in error between $0.5\,cm$ and $1.0\,cm$. A white circle represents the mean error from 500 runs for each case. For $d = 60$ and $\theta = 0$,

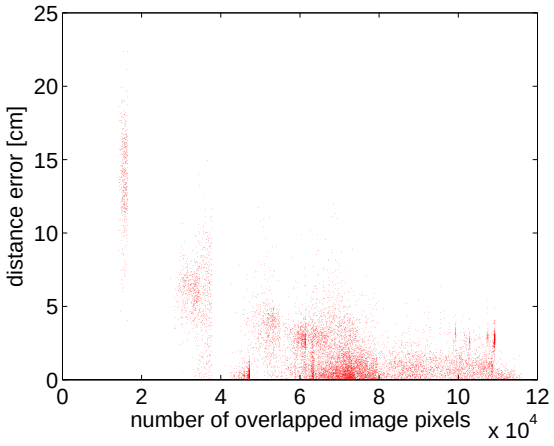


Figure 3.10: Scatter plot of the localization error as a function of the number of overlapping pixels between two cameras.

the mean error is $0.5\,cm$. As shown in Figure 3.9(a) and 3.9(b), when the baseline is $60\,cm$ or $80\,cm$, the mean error is within $1\,cm$, except when $(d = 80, \theta = 0)$ and $(d = 80, \theta = 10)$. On the other hand, as shown in Figure 3.9(c) and 3.9(d), the mean errors are relatively high for $d = 100\,cm$ and $d = 120\,cm$, especially at small angles. This is due to the fact that the overlapping area between two cameras is small for those cases. We plotted the localization error as a function of the number of overlapping pixels in Figure 3.10. Clearly, the size of the overlapping area determines the localization performance and we must account this when designing a multi-robot localization algorithm. Since the baseline distance and the angle between robots can be configured in our multi-robot localization algorithm for the best performance, the experimental results show how we should configure robots in our coordinated multi-robot localization algorithm as we do in the next experiment.

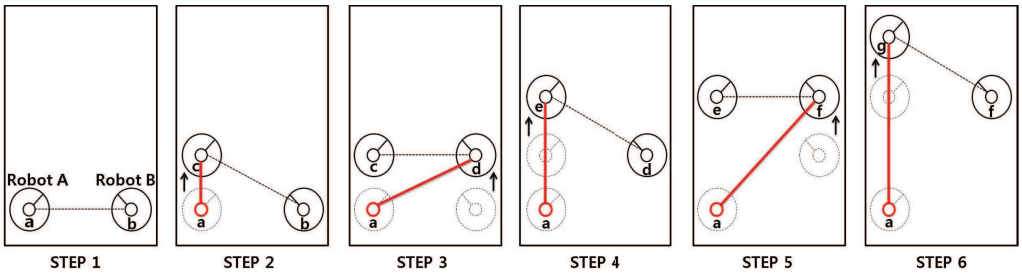


Figure 3.11: Multi-step experiment. The length of each red segment is the distance from the original position of robot A in step 1 to the new position of the robot with motion. Dashed lines show the relative poses that are computed at each step of the algorithm.

3.4.2 Coordinated Multi-Robot Localization: Multi-Step

In this experiment, we localize a group of robots as they move from one place to another as described in Section 3.3. Based on the previous experiment, we found that a baseline between 60 cm and 80 cm and an angle between 30 ° and 40 ° were ideal and this configuration was used in this multi-step experiment.

Figure 3.11 shows the movements of two robots going forward at different steps of the algorithm. A marker on Robot C is used for localization but Robot C is not illustrated in this figure. Because the space covered by the Vicon motion capture system was limited, we were able to perform six steps of the algorithm. Table 3.1 shows localization errors from the experiments. In the table, "seg" represents the line segment shown in Figure 3.11, "true" is the length of the segment computed by Vicon, and "est" is the length computed by our algorithm. At step 1, the difference between ground truth value and estimation value is 0.02 cm. After step 1, robot A goes forward for about 40 cm and turns to the left and robot B does not move. Since we know the angle between robot A and B from rotation matrix

seg	true	est	seg	true	est
a-b	77.14 cm	77.12 cm	a-b	77.14 cm	77.12 cm
b-c	85.76 cm	85.72 cm	a-c	36.22 cm	36.23 cm
c-d	78.29 cm	78.42 cm	a-d	85.73 cm	85.21 cm
d-e	88.16 cm	87.82 cm	a-e	74.01 cm	73.98 cm
e-f	80.47 cm	80.05 cm	a-f	107.82 cm	107.4 cm
f-g	89.82 cm	89.53 cm	a-g	109.73 cm	109.36 cm

Table 3.1: Results from the multi-step experiment. (See Figure 3.11 for segment labels)

R computed in step 1, when robot A rotates, we can maintain the pre-defined angle θ.

At step 2, the coordinate system with respect to robot B is the reference coordinate system and the localization error for the segment $a-c$ is only 0.01 cm. After step 2, robot A is stationed and robot B moves forward for about 40 cm. Again, we can maintain the pre-defined distance d by computing the position of robot B in step 2 with respect to the coordinate of robot A in step 1. And this process is repeated as shown in Figure 3.11. For all steps, the localization error was kept within 1 cm and the localization error of the longest segment $a-g$ was only 0.37 cm.

We also conducted the going forward experiments in the hallway to demonstrate its performance over a long distance (see Figure 3.12). A robot with a white LED plays the role of the moving group and two robots with a camera forms the stationary group. Table 3.2 shows localization results from the experiments in the hallway. For trajectories with length from 715 cm to 890 cm, the

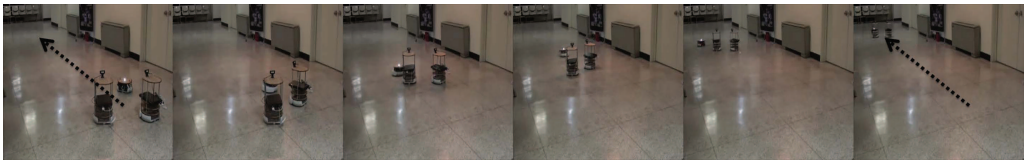


Figure 3.12: Photos from the hallway experiment. A group of robots moves along the straight line (black dotted line). At each step, a robot with an LED marker moves while the remaining two robots localize based on the movement of the robot with LED using the proposed algorithm.

Case	**Robot**	**True**	**Est.**	**Error**	**Err. Rate**
1	A	730 cm	728.1 cm	1.9 cm	0.26%
2	A	732 cm	733.5 cm	1.5 cm	0.20%
3	A	715 cm	716.7 cm	1.7 cm	0.23%
4	A	890 cm	886.7 cm	3.3 cm	0.37%
5	A	857 cm	854.5 cm	2.5 cm	0.29%

Table 3.2: Results from the multi-step experiment in the hallway.

achieved localization error is between 1.5 cm and 3.3 cm, making the localization error rate less than 0.37% of the length of the trajectory of the robot. Furthermore, the group of robots can follow the straight line without deviating from the desired path. See Figure 3.12 for photos from the experiments.

Next, we tested if a group of robots can make turns to avoid obstacles. Figure 3.13 shows snapshots of a multi-robot system making left and right turns. We first generated a desired path with right or left turns for the multi-robot system, satisfying the condition (3.6) in Section 3.3.2. Then a multi-robot system follows the given trajectory while localizing all robots. For each turn, four independent trials were performed and the results are shown in Figure 3.14(a) and Figure 3.14(b). Figure 3.14(c) shows localization errors of each robot as a function of time.

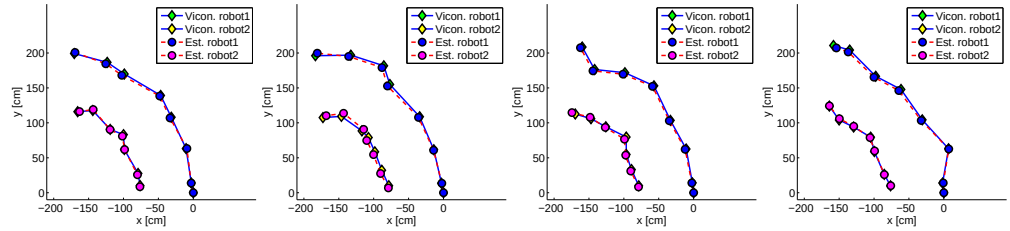
Lastly, we show the results from multi-robot navigation. Snapshots from the experiment are shown in Figure 3.15. Given a path found by the RRT-based multi-robot navigation algorithm, a multi-robot system moves cooperatively while performing localization. Figure 3.15(a) shows the planned path of the stationary robot found by the RRT-based multi-robot path planning algorithm to navigate to the goal location. Purple and green circles represent the planned positions of robot A and B, respectively. Blue and red arrows represent their respective headings. Figure 3.15(b) shows snapshots from the experiment showing the turn made by the multi-robot system. Black lines are the actual trajectories of robots following the planned path. Figure 3.16 shows results from obstacle avoidance experiments using the RRT-based multi-robot navigation algorithm and they are taken from different overhead cameras. As shown in the figure, robots can safely navigate to reach the goal location while avoiding obstacles.

Localization under the GPS denied or unstructured indoor environment is a

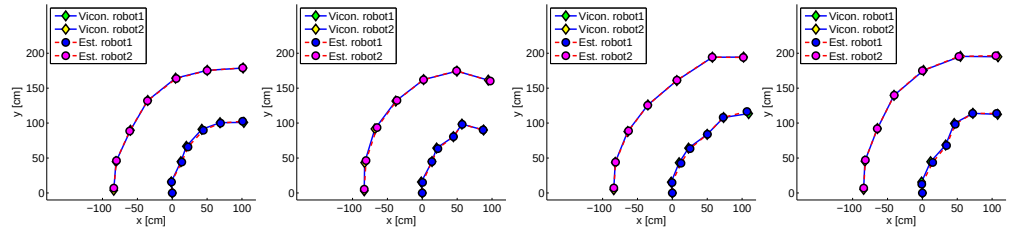
(a) Right turn

(b) Left turn

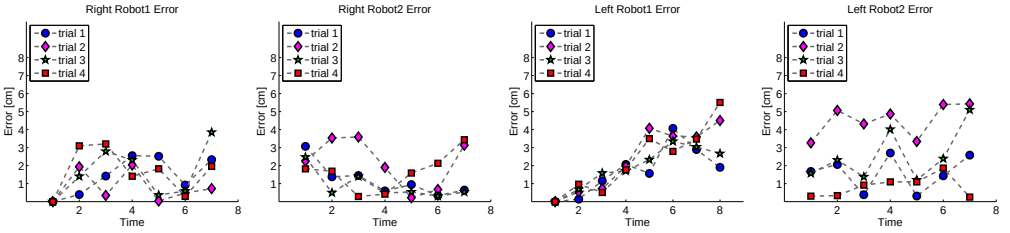
Figure 3.13: Snapshots from the multi-robot turning experiment. A team of robots are making left and right turns while maintaining its configuration. Black dotted arrows are the planned paths of the multi-robot system, satisfying the condition (3.6) in Section 3.3.2.



(a)



(b)



(c)

Figure 3.14: Estimated locations from turning experiments. Blue and magenta circles represent estimated positions of robot A and B, respectively. Green and yellow diamonds represent ground truth positions of robot A and B, respectively. (a) Left turn. (b) Right turn. (c) Localization errors as a function of time.

challenging problem. But the experimental results show that our algorithm can provide a promising solution to this challenging localization problem.

3.5 Discussions and Comparison to Leap-Frog

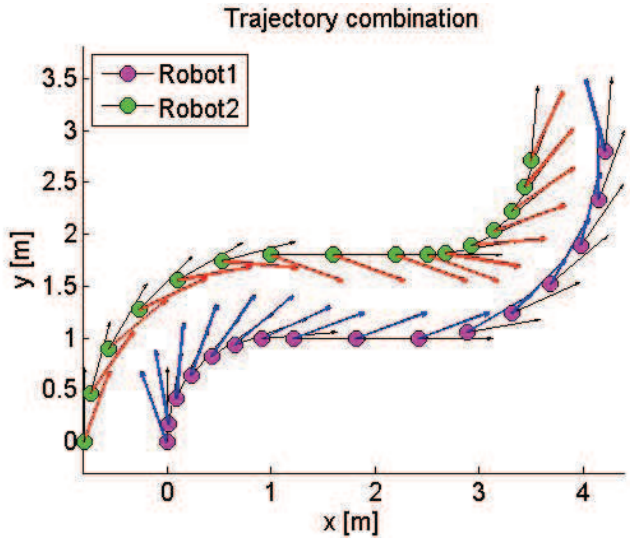
This section provides a short discussion on some practical aspects of the proposed localization algorithm and a comparison to the leap-frog method [28].

3.5.1 Discussions

Our experimental results in Section 3.4.2 suggests that precise localization is possible using an inexpensive robotic platform using camera sensors in an indoor environment with obstacles. Even if the environment is cluttered with obstacles, the proposed multi-robot navigation algorithm can find a path for the team of robots if a feasible path exists, following the probabilistic completeness of RRT [29].

The proposed coordinated localization algorithm requires regular communication between moving robots and stationary robots to make sure that the moving robots are within the field of view of stationary robots. Hence, a poor communication condition may affect the performance of the system since it will be difficult to control moving robots reliably. However, since the speed of the moving robot is known, we can predict when moving robots have to change their headings if communication delay can be estimated.

In order to match marker tracks, it is required to synchronize times of all robots. Mismatched timestamps can cause inaccurate localization. We have conducted a simple experiment to test the sensitivity of the proposed method against the time synchronization error. Based on collected data which has the time synchronization error less than 0.005 second, we have introduced time synchronization errors



(a)

(b)

Figure 3.15: (a) A trajectory found by the proposed multi-robot navigation algorithm. Purple circles and blue arrows represent the planned positions of Robot A and corresponding headings, respectively. Green circles and red arrows represent the planned positions of Robot B and corresponding headings, respectively. (b) Photos from the experiment following the trajectory. Black lines show the actual trajectories of robots.

(a) Photos from Camera 1

(b) Photos from Camera 2

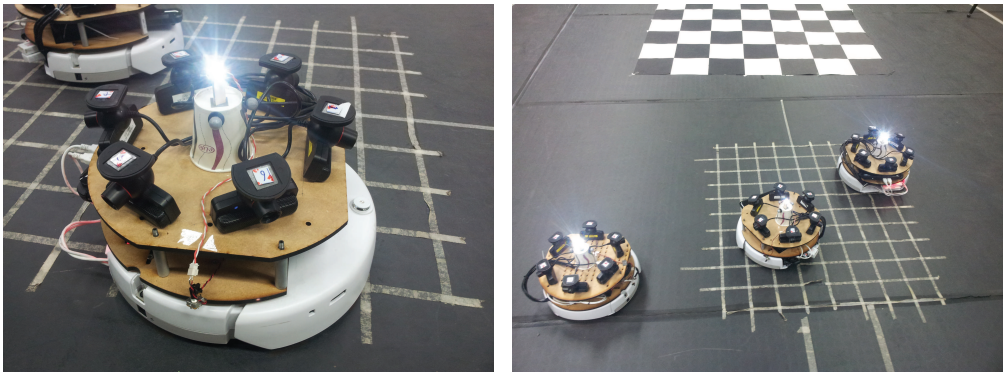
Figure 3.16: Photos from the experiment when the obstacle exists. Two sequences of photos are taken by two overhead cameras (Camera 1 and Camera 2). In order to avoid an obstacle, a team of robots is making necessary coordinated turns.

to the dataset. Our experiment shows that the localization error is kept under $1\,cm$ for a time synchronization error of 0.05 second, showing the robustness of the proposed method against the time synchronization error. However, we have observed an increase in the localization error when the time synchronization error is larger than 0.05 second. Hence, it is desirable to keep the maximum possible time synchronization error under 0.05 second or less. This condition can be satisfied in most cases. However, a wireless protocol with real-time guarantee, such as the guaranteed time slot (GTS) of IEEE 802.15.4 [63], can be utilized to avoid any unexpected time synchronization delays.

3.5.2 Comparison to Leap-Frog

We have also implemented the leap-frog localization method proposed in [28] for comparison. In its original implementation, the authors used the Learning Applied to Ground Vehicles (LAGR) platform equipped with three on-board computers, wheel encoders, and a set of four stereo cameras. However, since it is unclear if the approach is suitable for an inexpensive off-the-shelf robotic platform considered in this dissertation, we implemented the leap-frog algorithm using the robotic platform used in this dissertation, which includes PS3 Eye cameras, a white LED, and an iRobot Create platform as shown in Figure 3.17(a). A snapshot from the leap-frog localization experiment is shown in Figure 3.17(b). We placed six cameras on the robot as shown in Figure 3.18 to emulate an omnidirectional camera system. Note that a PS3 Eye camera has 75 degree field of view (FOV). An omnidirectional camera is required for the leap-flog system since measurements for its extended Kalman filter (EKF) algorithm is relative bearing angles.

In [28], a red ball is placed on each vehicle and a circle Hough transform is used to detect the position of other vehicle. In our implementation, we used an LED



(a) (b)

Figure 3.17: (a) A robot platform developed for leap-frog localization [28]. (b) A photo from the leap-frog localization experiment.

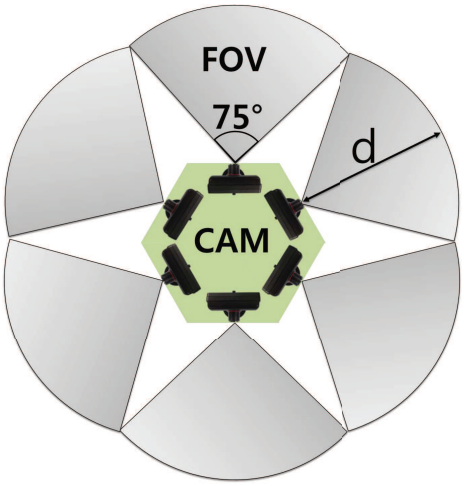


Figure 3.18: The configuration of six directional cameras for emulating an omni-directional camera. The gray region represents the FOV of each camera placed on iRobot Create.

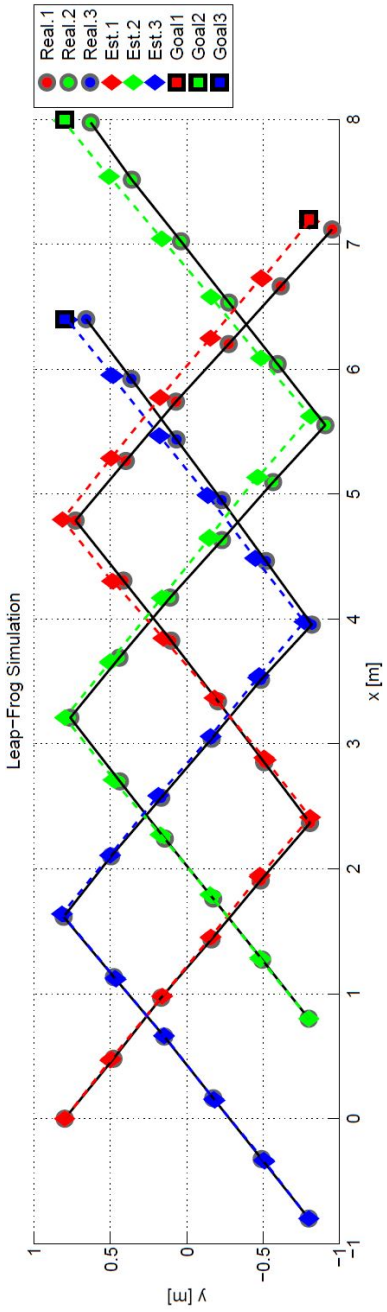


Figure 3.19: Simulation results of the leap-frog multi-robot localization method. The true positions of three robots are marked by red, green, and blue circles, respectively, while the estimated positions of three robots are shown in red, green, and blue diamonds. The red, green, and blue squares are the final goal positions.

as a marker for this purpose. In order to estimate the bearing angle using our omnidirectional camera system, we used the following second-order polynomial equation:

$$\theta_{LED} = aq^2 + bq + c, \tag{3.7}$$

where θ_{LED} is the relative angle of an LED with respect to the observing robot, q is the position of a detected LED in the horizontal coordinate, and a, b, and c are parameters of the polynomial. We collected 30 data pairs for each omnidirectional camera system and estimated values of a, b, and c based on the ground truth obtained from Vicon. Using estimated parameters, we found that the mean and standard deviation of the bearing angle error between the ground truth and the estimated bearings are $0.380\,^\circ$ and $0.286\,^\circ$, respectively. The error is small enough and (3.7) is used to estimate the bearing angle.

The key feature of the leap-frog localization is the extended Kalman filter (EKF) formulation using bearing-only measurements and multi-robot formation which maximizes the information gain [28]. We acquired the bearing angle by extracting the position of the LED using the MSER detector. Before applying the MSER detector, we made a differential image between an image with an LED on and an image with an LED off, to get rid of noise. We also implemented a leap-frog formation controller by alternating go-straight and turn motions. After each movement, we performed the EKF localization using bearing-only measurements acquired from robots. Since there are blind spots as shown in Figure 3.18, an additional routine for handling blind spots is required. This is implemented by adding additional go-straight motions inside the leap-frog algorithm.

We first conducted a simulation to verify the localization performance of the leap-frog algorithm based on the dynamical model of robots given in Section 3.3.1. The moving robot follows the leap-frog path by alternating go-straight and turn

motions. The initial distance between robots is 160 cm and the number of pairs of movements required to change the role of each robot is five (a movement pair consists of go-straight and turn motions). We added a Gaussian noise with variance of one to each bearing measurement. The localization result of the simulation is shown in Figure 3.19. The estimated positions of three robots are shown in red, green, and blue diamonds, respectively, and the true positions of three robots are shown in red, green, and blue circles, respectively. Red, green, and blue squares with black contour represent the goal positions of each robot. We performed a total of 100 trials. The average localization error of each robot is given in Table 3.3. We also checked the results when we changed the number of movement pairs. As shown in Table 3.3, the number of movement pairs does not have an effect on the results, so we applied five movement pairs in the physical experiment.

The localization results of physical experiments using the leap-frog method using three robots are shown in Figure 3.20. The localization error of robot 3 for the trajectory with length of 840 cm was 34 cm and the localization error of robot 1 and robot 2 are 46.3 cm and 58.6 cm, respectively. We can see that the localization error gets larger as each robot moves a longer distance. Note that the localization error of the actual experiment is relatively larger than that of simulation, showing the difficulty of controlling and localizing inexpensive robots. A comparison of the leap-frog method with the proposed algorithm is summarized in Table 3.4. The proposed algorithm shows an error rate which is 15 times smaller than the leap-frog method. The result indicates that the proposed algorithm is more suitable for inexpensive robotic platforms.

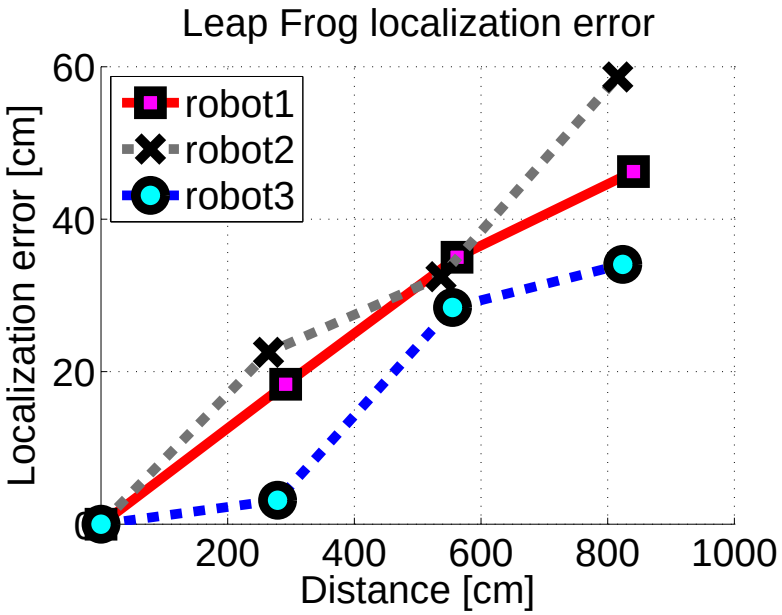


Figure 3.20: Localization errors from the leap-frog localization experiment.

No. Movement Pairs	5	10	15
Robot1	18.8 cm	17.2 cm	19 cm
Robot2	22.9 cm	21.4 cm	23 cm
Robot3	14.8 cm	14.1 cm	15.4 cm

Table 3.3: Average localization errors for different step sizes (method: leap-frog).

Algorithm	Mean Distance	Error	Error Rate
Leap-frog	826.9 cm	46.3 cm	5.6%
Proposed	715 $\sim$ 890 cm	1.5 $\sim$ 3.3 cm	0.20% $\sim$ 0.37%

Table 3.4: A comparison between the proposed method and the "Leap-Frog" method.

3.6 Summary

In this chapter, we have presented a coordinated localization algorithm for mobile sensor networks. The algorithm is designed to solve the challenging localization problem under the GPS denied or unstructured indoor environment by taking the advantage of the multi-agent system and mobility in mobile sensor networks. The proposed algorithm can solve the multi-robot navigation problem by considering the configuration constraint of a group of robots. Our experiment shows that there exists a configuration of robots for good localization and this configuration is applied to find a trajectory which makes a group of robots move from one location to another location. We also compared the performance of the proposed algorithm against the leap-frog method using an inexpensive off-the-shelf robotic platform. In experiments, the proposed method achieves a localization error of 0.37% or less for trajectories of length between $715\,cm$ and $890\,cm$. The experimental results show that the localization error increases as a robot travels a longer distance. This propagation of error can be reduced by detecting landmarks in the environment and this is our future research topic.

Chapter 4

Preliminaries to Cost-Aware Path Planning

As explained in Chapter 2, path planning suitable for mobile sensor networks should consider the environmental field when determining the path by improving the quality of the path based on the costmap which represents the environmental parameter. Thus, we assume an availability of a costmap of the field of interest, which represents environmental parameters, such as temperature, humidity, chemical concentration, wireless signal strength, stealthiness, and terrain elevation. Our objective is to design a path planning algorithm which guides a robot to follow a trajectory from the initial location to the destination with the minimal accumulated cost, along with the terminal cost and travel time.

We propose a cost-aware path planning algorithm for complex configuration spaces inspired by [64], which addresses the position-dependent path planning problem (PDPP). In a PDPP problem, the cost is solely dependent on the position of a robot. The algorithm proposed in [64] constructs an RRT tree by extending the tree using cross entropy path planning [37], which optimizes the trajectory

distribution using the stochastic optimization method called cross entropy (CE) [65]. In this dissertation, we consider a more general problem in which the cost is a function of the internal state of a robot. We refer this problem as a state dependent path planning (SDPP) problem.

This chapter first introduces the recent algorithms which considers the quality of the path and then studies the primary algorithms which form the basis of our proposed methods.

4.1 Related works

Recently, a number of cost-aware path planning algorithms have been proposed. Suh and Oh [64] presented a sampling-based path planning algorithm which finds a low-cost path with respect to a continuous costmap representing the environmental field. For planning a path, they used RRT and extended the search tree using a stochastic optimization method, called cross entropy (CE) [65]. In [66], a solar-intensity map is used as a costmap and a path which can charge the battery the most is found using dynamic programming. Murphy et al. [67] constructed a costmap representing traversability using aerial images and performed path planning using the A* search algorithm with heuristics to find a less riskier path. Above mentioned algorithms solve instances of PDPP since they consider an instantaneous cost at each location.

On the other hand, the accumulated cost in SDPP is determined depending on how a robot has reached the state. An energy efficient planning can be considered as an instance of SDPP, since the energy consumption at the current state depends on previous states. There are a number of studies on energy efficiency of motion planning for ground mobile robots [68, 69, 70, 71, 72, 73, 43] and humanoid robots [74, 75, 76, 77, 78]. In [68, 69, 70], energy-efficient path

planning algorithms were developed for a 2D plane with no changes in elevation. They determine the velocity schedule of a moving robot by minimizing energy consumption given a predetermined path [68] or finding a path using the A* algorithm [69] or dynamic programming [70]. On the other hand, [71] and [72] considered terrains with different elevations. Sun et al. [71] examined the energy consumed by a robot along the path in terms of friction and gravity, but they did not take into account the optimal velocity profile. In [72], a vehicle velocity profile was considered to minimize the energy consumption by an electric vehicle based on dynamic programming. However, they focused on the efficiency of in-wheel motors to maximize the travel distance on a predetermined road. Kwak et al. [73] minimized the consumed energy along a path when planning on a rough terrain based on particle-RRT (pRRT) proposed in [30] which extends a tree by using particles for estimation of the distribution of states at each node of the tree under the environment with uncertainty caused by input. They fused the energy function with the likelihood of successful tree extension in pRRT. Given a costmap representing the terrain with different elevations, Jaillet et al. [43] found a low-cost path using RRT but extending the RRT tree based on the Metropolis criterion.

Unlike the approaches using a ground mobile robot, Michieli et al. [74] performed the energy analysis of a humanoid robotic arm by representing the arm as a complex energy chain of mechanical and electrical components. Kulk et al. [75] proposed a low-stiffness walk of a humanoid robot by manually tuning the parameter on each motor in the robot, through a modification in the low-level controller. They used the electric current data measured with built-in current sensors to evaluate the performance of their work. The design of an energy-efficient and human-like gait for a humanoid was presented in [76] and [77], which focused

on a gait with low energy consumption. Kalakrishnan et al. [78] proposed a motion planning method based on a cost function which includes torque costs but without an experimental validation.

Recently, RRT*, which guarantees the asymptotic optimality, and its variants have been introduced [53, 79, 80, 81, 82]. Given a cost function, RRT* finds the optimal solution as the number of samples increases to infinity. In [83, 84, 85], modified RRT and RRT* were applied to manipulation tasks in high dimensional spaces. However, these algorithms focus on finding the shortest path without considering the energy consumed by a robot along the path. Hence, they are not suitable for energy-efficient motion planning.

The approach proposed in this dissertation improves efficiency in a complex terrain or a high dimensional space while ensuring the asymptotic optimality of RRT*. It has an anytime flavor in the sense that the proposed algorithm provides a near optimal solution and monotonically improves its solution towards the optimal one as more operations are allowed

4.2 Sampling based path planning

In general, the path planning problem is known to be PSPACE-hard problem in the computational point of view [86, 87] and the computation complexity increases exponentially as the dimension of the configuration space or the number of the obstacles increases [88].

In order to overcome such computational burden, many sampling based path planing approaches have been introduced during the last decades. They only require a collision detection algorithm without explicit construction of obstacles in the configuration space and incrementally search the configuration space. Thus, sampling based algorithms can be very efficient in complex and high dimensional

space. Although they are not complete, they have the property of the probabilistic completeness in the sense that the probability that they find a solution, if one exists, approaches one as the number of samples goes to infinity.

In this section, we introduce several sampling based algorithms which are based on the proposed methods. Before discussing those algorithms, we first describe the following common primitive procedures.

- A tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ or a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$.
- $Sample(\mathcal{X})$ function returns an independent, uniformly distributed sample from $\mathcal{X}$.
- $Nearest\_Neighbor(\mathcal{T}, x)$ or $Nearest\_Neighbor(\mathcal{G}, x)$ returns a vertex in $\mathcal{V}$ which is closest to x in terms of the Euclidean distance.
- $Steer(x, y)$ procedure returns a configuration z in a ball centered around x closest to y, i.e. $\arg\min_y \|z-y\|$ s.t. $\|x-z\| < \rho$, where $\rho > 0$ is a predefined steering parameter.
- $Parent(\mathcal{T}, x)$ returns a unique vertex $x' \in \mathcal{V}$ such that $(x', x) \in \mathcal{E}$.
- $Near(x, r)$ returns a set of all points in $\mathcal{V}$ that are within a ball of radius r [1] centered at x.
- $CollisionFree(x, y)$ checks whether the line connecting two points x and y is placed in $\mathcal{X}_{free}$.

One of the most representative sampling based path planning algorithm is the rapidly-exploring random tree (RRT) [29] which explores a high dimensional

[1] The radius is defined as $r \leq \gamma(\log(n)/n)^{\frac{1}{d}}$, where γ is a constant factor ensuring the optimality of the path, n is the number of all nodes in $\mathcal{V}$ and d is the state dimension.

configuration space using random sampling. An outline of RRT is shown in Algorithm 1. Once a starting point and the goal region are defined, the algorithm iteratively samples a random point x_{rand} over the configuration space and makes an attempt to expand the search tree $\mathcal{T}$, which is initialized with the starting point. The x_{rand} is sampled from a uniform distribution over the space but we can make the distribution biased towards the goal point for faster search of a path to the goal. The tree $\mathcal{T}$ is extended by connecting from x_{near}, which is the nearest point of the tree from x_{rand}, to a new point x_{new} in the direction of x_{rand}, provided that x_{near} can be found. x_{new} is computed by applying a control input $u \in U$, where U is a set of possible controls, to the vehicle dynamics for a fixed duration $\triangle t$. Once the tree $\mathcal{T}$ reaches the goal point, the path can be built by searching the tree $\mathcal{T}$ recursively from the goal point to the starting point.

Algorithm 1 RRT

$\mathcal{V} \leftarrow \{x_0\}, \mathcal{E} \leftarrow \emptyset, \mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E})$
while stopping_criterion is false **do**
  $x_{rand} \leftarrow$ Sample($\mathcal{X}_{free}$)
  $x_{nearest} \leftarrow$ Nearest_Neighbor($\mathcal{T}, x_{rand}$)
  $x_{new} \leftarrow$Steer($x_{nearest}, x_{rand}$)
  if CollisionFree($x_{nearest}, x_{new}$) **then**
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{new}\}, \mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{nearest}, x_{new})\}$
  end if
end while
return $\mathcal{T}$

Although RRT is known as an efficient approach for complex path planning problems, it focuses mainly on the feasibility of the path with less consideration on the cost of the path. So it is not suitable for optimal path planning problem.

In order to solve the limitation in the optimal point of view, the optimal rapidly exploring random graph (RRG) and optimal RRT (RRT*) have been proposed recently considering the importance of the quality of the path in [53].

RRG and RRT*, which are variants of RRT, are outlined in Algorithm 2 and 3, respectively. They are similar to RRT in that they iteratively explore the configuration space by incrementally increasing a graph or a tree towards x_{rand} through the connection from $x_{nearest}$ to x_{new}. However, RRG performs the additional connections to build a graph unlike RRT. Whenever x_{new} is added to $\mathcal{V}$, connections between x_{new} and all vertices included in $\mathcal{V}$ which are returned from the function $Near(x_{new}, r)$ are added if each connection is valid. RRT* is a sub-graph of RRG which prunes unnecessary edges contained in RRG which are not included in the best path found to each node. In order to perform such procedures, two following procedures are required. First, the parent of x_{new} is selected among all vertices in X_{near} based on connections from x_{near} to x_{new}. The selected parent gives the least cost from the starting point to x_{new} through itself. Once the parent is selected, the rewiring procedure is performed. For all vertices in X_{near}, if the cost of the unique path from the starting point to x_{near} is higher than the cost of the path to x_{near} through x_{new}, then x_{near} is disconnected from its old parent and x_{new} is assigned as the new parent of x_{near} and connected to x_{near}. By performing the above two procedures, RRT* maintains a directed tree structure.

4.3 Cross entropy method

Cross entropy (CE) is an adaptive stochastic optimization method designed to estimate the probability of rare events [36]. It has been also extended to solve combinatorial optimization problems, such as the traveling salesman problem [89].

Algorithm 2 RRG

$\mathcal{V} \leftarrow \{x_0\}, \mathcal{E} \leftarrow \emptyset, \mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$
while stopping_criterion is false **do**
  $x_{rand} \leftarrow \text{Sample}(\mathcal{X}_{free})$
  $x_{nearest} \leftarrow \text{Nearest\_Neighbor}(\mathcal{G}, x_{rand})$
  $x_{new} \leftarrow \text{Steer}(x_{nearest}, x_{rand})$
  if CollisionFree$(x_{nearest}, x_{new})$ **then**
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{new}\}$
    $X_{near} \leftarrow \text{Near}(\mathcal{G}, x_{new})$
    for $x_{near} \in X_{near}$ **do**
      if CollisionFree(x_{new}, x_{near}) **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{new}, x_{near})\}$
      end if
      if CollisionFree(x_{near}, x_{new}) **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{near}, x_{new})\}$
      end if
    end for
  end if
end while
return $\mathcal{G}$

Algorithm 3 RRT*

$\mathcal{V} \leftarrow \{x_0\}, \mathcal{E} \leftarrow \emptyset, \mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E})$
while stopping_criterion is false **do**
  $x_{rand} \leftarrow \text{Sample}(\mathcal{X}_{free})$
  $x_{nearest} \leftarrow \text{Nearest\_Neighbor}(\mathcal{T}, x_{rand})$
  $x_{new} \leftarrow \text{Steer}(x_{nearest}, x_{rand})$
  if $\text{CollisionFree}(x_{nearest}, x_{new})$ **then**
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{new}\}$
    $X_{near} \leftarrow \text{Near}(\mathcal{T}, x_{new})$
    $x_{min} \leftarrow x_{nearest}$, $c_{min} \leftarrow \text{Cost}(x_{nearest}) + \mathcal{J}(\text{Line}(x_{nearest}, x_{new}))$
    for $x_{near} \in X_{near} \setminus \{x_{nearest}\}$ **do**
      $c' \leftarrow \text{Cost}(x_{near}) + \mathcal{J}(\text{Line}(x_{near}, x_{new}))$
      if $c' < c_{min}$ *AND* $\text{CollisionFree}(x_{near}, x_{new})$ **then**
        $x_{min} \leftarrow x_{near}$, $c_{min} \leftarrow c'$
      end if
    end for
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{min}, x_{new})\}$
    for $x_{near} \in X_{near} \setminus \{x_{min}\}$ **do**
      $c' \leftarrow \text{Cost}(x_{new}) + \mathcal{J}(\text{Line}(x_{new}, x_{near})$
      if $c' < \text{Cost}(x_{near})$ *AND* $\text{CollisionFree}(x_{new}, x_{near})$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{new}, x_{near})\}$
      end if
    end for
  end if
end while
return $\mathcal{T}$

Consider the following optimization problem with a performance function S

$$\min_{x \in \mathcal{X}} S(x). \tag{4.1}$$

Let γ^* be the minimum of (4.1). By defining a family of auxiliary probability density functions (PDFs) $\{p(\cdot;\theta), \theta \in \Theta\}$ on $\mathcal{X}$, we can transform the deterministic problem into a stochastic problem. Then, (4.1) can be formulated as the following estimation problem:

$$l(\gamma) = \mathbb{P}(S(X) \leq \gamma) = \mathbb{E}_p[\mathbf{I}_{\{S(X) \leq \gamma\}}],$$

where X is a random vector with PDF $p(\cdot;\phi)$ for some $\phi \in \Theta$, γ is a real number, and $\mathbf{I}$ is the indicator function. The expectation is taken with respect to the distribution p. A simple (and crude) approach to estimate l is to use the following Monte Carlo method

$$\hat{l} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{I}_{\{S(X_i) \leq \gamma\}},$$

where $X_1, X_2, \ldots, X_N$ are samples drawn from p. However, for small γ, the probability l will be very small and a large number of samples are required to estimate l accurately. A better approach to estimate l using a smaller number of samples is the *importance sampling* method. Suppose that q is an other probability density function such that if $q(x) = 0$, then $\mathbf{I}_{\{S(X) \leq \gamma\}} p(x) = 0$. Then, l can be rewritten as

$$l = \int \mathbf{I}_{\{S(x) \leq \gamma\}} \frac{p(x)}{q(x)} q(x) dx = \mathbb{E}_q \mathbf{I}_{\{S(X) \leq \gamma\}} \frac{p(X)}{q(X)},$$

where the expectation is taken with respect to q, which is known as the importance sampling density. Hence, with N independent samples from q, we can estimate l using

$$\hat{l} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{I}_{\{S(X_i) \leq \gamma\}} W(X_i),$$

where $W(X_i) = p(X_i)/q(X_i)$ is known as the importance weight. If q can provide more samples for $\{S(X) \leq \gamma\}$, then we can estimate l more accurately than the simple Monte Carlo method.

While the best choice q^* for q is the density which minimizes the variance of the estimator $\hat{l}$, it cannot be computed in practice. Cross entropy attempts to find q which is closest to q^* by minimizing the Kullback-Leibler (KL) divergence between two densities. Suppose that the target distribution is $p(\cdot;\theta^*)$ from a family of distributions, where θ^* is the true parameter of the distribution p. The cross entropy algorithm shown below is used to find the optimal parameters θ^* which minimizes the KL divergence between q^* and $p(\cdot;\theta)$.

1. It generates $X_1, \cdots, X_N$ from $p(X;\hat{\theta}_{k-1})$, where $\hat{\theta}_{k-1} = \phi$ when $k = 1$. Compute the cost $S(X_i)$ and list them in the order i.e., $S(1) \leq \cdots \leq S(N)$.

2. Let $\hat{\gamma}_k$ be ϱ-th quantile of $S(X)$, i.e., $\hat{\gamma}_k = S_{\lceil \varrho N \rceil}$, where ϱ is a small number between 10^{-2} and 10^{-1}.

3. Compute the parameter $\hat{\theta}_k$ using the following equation

$$\begin{aligned} \hat{\theta}_k &= \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathbf{I}_{\{S(x) \leq \hat{\gamma}_k\}} W(X_i;\phi,\theta_{k-1}) \\ &\times \ln p(X_i;\theta), \end{aligned}$$

   where $W(x;\phi,\theta) = p(x,\phi)/p(x,\theta)$.

4. Increment k and iterate until $p(X;\hat{\theta}_k)$ converges to a delta function.

4.3.1 Cross entropy based path planning

In [37], the cross entropy method is applied to path planning, where the optimal control law with minimum time is sought for. Algorithm 4 shows the cross entropy path planning algorithm which finds a trajectory for a robot from the start

position x_{start} to the end position x_{end}. It samples a trajectory X from distribution $p(\cdot; v)$, where v is a set of controls. The algorithm first generates N random trajectories $X_1, \ldots, X_N$ from $p(X; v_i)$ considering constraints such as obstacles and vehicle dynamics over the configuration space. If $\varrho > 0$ is a small number, the algorithm selects ϱN elite trajectory samples among all trajectory samples that have less cost based on the cost function H, which is defined as:

$$H(X) = \int_0^T C(x(t))dt$$

where $C = 1 + \beta\|x - x_{end}\|^2$ with a positive constant β and T is the termination time of the trajectory [37]. Then, it updates the parameter v_i (i.e., a set of pairs of control input and its duration) using the elite set. The algorithm iterates until the sampling distribution $p(X; v_i)$ converges to a delta distribution.

Algorithm 4 Cross Entropy Path Planning

Require: 1. Start position x_{start} and end position x_{end}

2. Number of trajectory samples N

3. Coefficients ϱ and β

Ensure: Shortest time path from x_{start} to x_{end}

1: $i = 0$.

2: Draw N samples $X_1, \ldots, X_N$ from $p(X; v_i)$, where $p(X; v_i)$ is a uniform distribution when $i = 0$.

3: Select ϱN trajectory samples with lower costs among all trajectory samples to the cost function H.

4: Update the parameter v_i using the elite set.

5: $i = i + 1$.

6: Repeat steps 2–5 until $p(X; v_i)$ converges to a delta function.

Chapter 5

Fast Cost-Aware Path Planning using Stochastic Optimization

This chapter presents an efficient algorithm for solving the optimal motion planning problem in a complex configuration space which minimizes the accumulated cost of the path. The proposed algorithm can be applied to mobile sensor networks by considering the environmental field. In this dissertation, the complex configuration space is represented by the costmap of the field which is computed based on the environmental parameter using a Gaussian process. The Gaussian process (or Kriging in geostatistics) is a nonparametric regression method which has been successfully applied to estimate and predict complex physical phenomena [90].

The proposed method improves upon RRT* by introducing nonmyopic extensions using cross entropy. As explained in section 4.2, the RRT path planning algorithm, as well as RRT*, iteratively samples a random point x_{rand} over the

configuration space and makes an attempt to expand a search tree $\mathcal{T}$, which is initialized with a starting point. The tree $\mathcal{T}$ is extended by connecting from $x_{nearest}$, which is the nearest point of the tree from x_{rand}, to a new point x_{new} in the direction of x_{rand}. When RRT* is applied to a complex environment for path planning, it incrementally extends its tree from local search. So it requires a large number of samples to find an optimal solution and the problem gets worse for complex terrains or higher dimensional problems. We address this problem by constructing two RRT trees: a standard RRT* tree $\mathcal{T}$ and an extended tree $\mathcal{T}_e$. $\mathcal{T}$ is used to determine the $x_{nearest}$ of any x_{rand} for better exploration. $\mathcal{T}_e$, which includes $\mathcal{T}$, contains additional longer extensions for nonmyopic search over paths with less cost. When a new random state x_{rand} is chosen, the proposed algorithm searches for a path with the minimum cost from $x_{nearest} \in \mathcal{T}$ to x_{rand} using CE. The path with the minimum cost is inserted to $\mathcal{T}_e$ and x_{new} is selected from the path. The node x_{new} is also inserted to $\mathcal{T}$ and extra nodes in $\mathcal{T}_e$ are added to $\mathcal{T}$ if they allow a low-cost path to x_{new}. By utilizing two separate trees, we can ensure unbiased exploration over the space using $\mathcal{T}$ and, at the same time, improves the efficiency of search using nonmyopic extensions in $\mathcal{T}_e$.

We show that the proposed cost-aware path planning algorithm consistently finds low-cost paths against RRT [29] and RRT* [53] from a set of extensive simulations including physics-based simulations using the dynamic model of a two-wheel robot, Pioneer 3DX, on complex terrains and experiments using a humanoid robot, Nao, in a high dimensional configuration space.

5.1 Problem formulation

Let $\mathcal{Q}$ denote the operation region, in which a robot performs its tasks. Let $\mathcal{X} \subset \mathbb{R}^n$ be the state space of a robot, where $\mathcal{X} = \mathcal{X}_{free} \cup \mathcal{X}_{obs}$ and the state

$x \in \mathcal{X}$ includes the position $q \in \mathcal{Q}$ of a robot. $\mathcal{X}_{obs}$ represents the space where obstacles are placed or the robot may be in collision due to actuator bounds and $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$ is a free space. We assume that the state of the robot $x(t) \in \mathcal{X}$ is determined by

$$\dot{x}(t) = f(x(t), u(t)), \tag{5.1}$$

where $u(t) \in \mathcal{U} \subset \mathbb{R}^p$ is the control input applied at time t and f is a class C^∞ or smooth function.

Let $x_0 \in \mathcal{X}$ be the initial state of the robot and $x_{goal} \subset \mathcal{X}$ be the goal region. Let $\pi_u(t)$ be the trajectory solution to the differential equation (5.1) for given $u(t)$ from $t = 0$ to $t = T \in \mathbb{R}^+$, where T is the termination time. The trajectory $\pi_u(t)$ can be parameterized by a series of states and control inputs $(x(t), u(t))$ and it connects the initial state and the goal region such that $x(0) = x_0$ and $x(T) \in x_{goal}$. We assume in this dissertation that the trajectory is deterministic given the environment and control inputs. Then we want $\pi_u(t) \in \mathcal{X}_{free}$ for all times. Given the initial state $x_0 \in \mathcal{X}$ and the goal region $x_{goal} \subset \mathcal{X}$, such that $x(0) = x_0$ and $x(T) \in x_{goal}$, and a cost function $\mathcal{J}$, the minimum cost path planning problem can be expressed as:

$$\begin{aligned} \arg\min_{u(t):0\le t\le T} \quad & \mathcal{J}\Big(\pi_u(t)\Big) \\ & \text{subject to } \pi_u(t) \in \mathcal{X}_{free} \text{ for all } t \in [0, T] \\ & \text{and } \pi_u(T) \in x_{goal}. \end{aligned} \tag{5.2}$$

The classical path planning problem defines a cost function which only considers the length of a path, so the optimal solution is focused on how fast a robot can reach the goal region. However, the cost function for the minimum cost path planning problem is required to take into account the quality of a path while a robot is moving along the trajectory. This means the cost function $\mathcal{J}$ can depend

on a costmap $\mathcal{C}$ which depends on the position of a robot (i.e., $\mathcal{C} : \mathcal{Q} \to \mathbb{R}$) or an energy function E which depends on the state of a robot (i.e., $E : \mathcal{X} \to \mathbb{R}$).

In this dissertation, we consider a cost-aware path planning problem, which is to solve path planning problem while minimizing the cost along the path based on the defined cost function.

5.2 Issues with sampling-based path planning for complex terrains or high dimensional spaces

A sampling-based path planning algorithm, such as RRT and PRM, has some charming properties. First of all, the probabilistic completeness can be guaranteed, meaning that the probability that the algorithm finds a solution increases to one as the number of samples grows to infinity if a solution exists. Such algorithms can also find a solution rapidly for a complex high dimensional space since they incrementally grow a graph to randomly chosen point until they reach the goal region. Moreover, the implementation of a sampling-based method is relatively simple. But, both approaches are not suitable for the problem of cost-aware path planning in a complex terrain or a high dimensional space since they focus on finding a feasible path to the goal region, which is collision-free. A variant of RRT, called RRT*, can be a candidate approach for finding a path appropriate for the mission since it returns a minimum-cost path after an enough number of iterations due to the asymptotic optimality of RRT*. However, finding the minimum-cost path in a complex terrain or a high dimensional space can be considered as an extremely rare event as the dimension of the state space increases, so it requires an extremely large number of samples to find a good solution.

We demonstrate this using a toy example with a valley shown in Figure 5.1.

The goal region is located at a slightly lower altitude than the start position. We assume five angular velocities, $\omega \in \{-\frac{\pi}{2}, -\frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{\pi}{2}\}$, as a set of possible inputs with a constant translational velocity at an interval of 5 seconds. The duration of a trajectory is 60 seconds. The dynamic model and the energy function used in this example are described in Section 5.5.2. From 5^{12} randomly generated RRT trajectories, 6,038 trajectories reach the goal region (q_{rand}) and they are shown as black lines in Figure 5.1. The trajectory in magenta is the trajectory with the minimum cost among 6,038 trajectories. For a comparison, a path found by the tree extension step of the proposed algorithm (CAPP) is shown in white. The required energy for the path from CAPP is 0.3972 J (joule) and the path from RRT requires 6.9577 J, which is more than 17 times larger than a solution found by CAPP. When RRT* extends an RRT tree, it selects x_{new} by steering towards x_{rand} without considering the cost of the path to x_{rand} so it extends the tree based on a simple local search. Therefore, RRT* requires an extremely large number of samples to find the minimum cost path. The running time of RRT* for finding a trajectory with the same energy cost as CAPP was five times longer than that of CAPP.

We consider another toy example, in which a humanoid robot lowers one arm to place its hand at a specific position. Even if the motion is extremely easy, there is a clear difference in energy consumption depending on how to schedule the motion trajectory. Since robot arms has multiple joints and each joint represents a single state, the configuration space has high dimensions. We assume five joints for one arm and each joint can move within its limited angle range. When a robot takes an action, the consumed energy is determined according to the torque on each joint and the change of each joint angle. More detailed explanation about a humanoid robot and its energy function used in this work

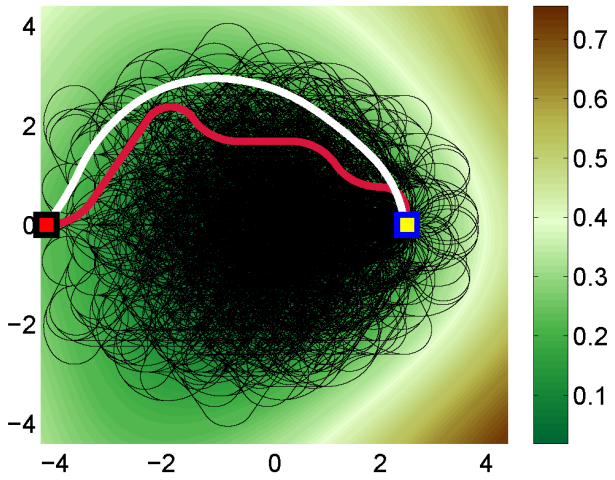


Figure 5.1: A simple terrain with a valley. Different colors represent different elevations (see the colorbar). The start position and goal region are marked by red and yellow squares, respectively. Random trajectories from RRT are shown in black. The magenta trajectory is the minimum cost path found by RRT and the white trajectory is a path found by the proposed algorithm.

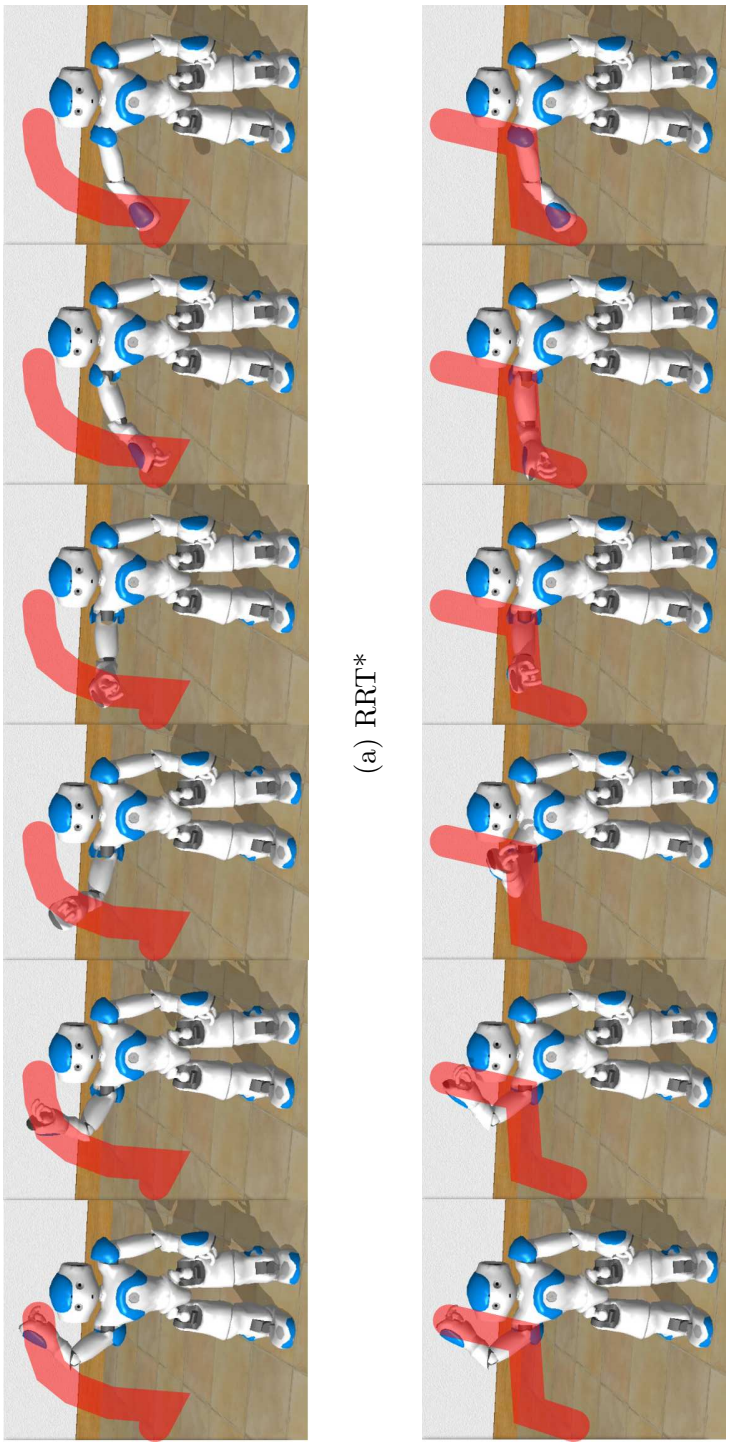
can be found in Section 3.4. Figure 5.1(a) and 5.1(b) show paths obtained from RRT* and the proposed algorithm, respectively, with a deadline of 500 seconds. The red thick lines shown in the snapshots represent the trajectory of an end effector. Since the torque on a single joint of the arm is affected by other joints of the arm, the consumed energy is highly dependent on the state of other joints. That is, there exist certain configurations of the arm which minimize the torque. Therefore, the robot should move while maintaining configurations with low energy consumption. The trajectory obtained from RRT* tends to go straight to the goal region while moving multiple joints simultaneously. On the other hand, the proposed algorithm finds a trajectory which moves each joint of the arm in a

more energy-efficient manner. It first lowers the arm by moving the shoulder pitch angle without changing the shoulder roll angle. When the pitch angle becomes zero, i.e., the direction of the arm becomes orthogonal to the body, it moves the shoulder roll joint since this configuration requires less torque on the shoulder roll joint. After moving the shoulder roll joint, the pitch joint moves again to the goal state. The robot following the trajectory obtained from RRT* consumes 420.86 J while the proposed algorithm provides a trajectory requiring only 296.61 J. Since RRT* requires dense sampling in order to find the optimal solution due to its myopic nature, it is computationally intractable in a high dimensional space. Hence, as demonstrated in this example, for solving an energy-efficient path planning problem in high dimensional spaces, we need an efficient nonmyopic approach.

5.3 Cost-Aware path planning (CAPP)

We now describe the proposed cost-aware path planning algorithm that generates a trajectory to minimize the accumulated cost along its path. The key idea is to sample from the configuration space and to grow a search tree (or RRT tree) by extending the tree using stochastic optimization. Unlike existing sampling based motion planning algorithms, in which the RRT tree is extended towards the random point based on the distance only, the proposed algorithm finds a path towards the random point with minimal cost in a nonmyopic manner. The proposed algorithm is based on RRT*, which guarantees the probabilistic completeness and the asymptotic optimality, and cross entropy to find cost-efficient controls from a parameterized continuous input space. Thus, the proposed algorithm is referred as cost-aware RRT* (CARRT*) in this dissertation.

The overall structure of CARRT* is given in Algorithm 5. The primitive procedures described in 4.2 are shared by CARRT*. Unlike RRT*, a specifically tailored



(a) RRT*

(b) The proposed

Figure 5.2: A simple scenario using a humanoid robot. A robot is assigned to lower its one arm to place its hand at the specific location. Two figures (a) and (b) show the process of following trajectories obtained from RRT* and the proposed algorithm, respectively. The red line represents the trajectory of an end effector of the arm.

extension procedure is added to CARRT*. The algorithm constructs two RRT trees: a standard RRT tree $\mathcal{T} := (\mathcal{V}, \mathcal{E})$ and an extended tree $\mathcal{T}_e := (\mathcal{V}_e, \mathcal{E}_e)$. The standard tree $\mathcal{T}$ is used to determine the nearest point to a random point x_{rand} (line 5) for better exploration. The extended tree $\mathcal{T}_e$ includes $\mathcal{T}$, such that $\mathcal{V} \subset \mathcal{V}_e$ and $\mathcal{E} \subset \mathcal{E}_e$. $\mathcal{T}_e$ contains additional branches with, which are not present in $\mathcal{T}$, and CARRT* uses $\mathcal{T}_e$ for finding low-cost long paths. The function CE_Extend described below is used for growing $\mathcal{T}_e$.

When performing an extension, if the distance between $x_{nearest}$ and x_{rand} is larger than a threshold $\eta > 0$, i.e. $\|x_{nearest} - x_{rand}\| > \eta$, CE_Extend function finds a cost-aware path $\mathcal{P}^*$ from $x_{nearest}$ towards x_{rand} using CE path planning (line 7), where $\eta > \rho$. A set of elite states, X_{CE}, is extracted from $\mathcal{P}^*$ based on the extension unit length ρ (line 8). Note that if we add all points in X_{CE} to $\mathcal{T}$, the tree will be biased towards x_{rand}, resulting poor exploration over the state space. This is the reason why CARRT* constructs two separate trees to tradeoff exploration and exploitation. The first point x_{ce_1} of X_{CE} is used to plan both trees as x_{new} in function $Plan\_DoubleTrees$ (lines 9–10). All other points in X_{CE} are inserted only to $\mathcal{T}_e$ using $Plan\_SingleTree$ with rewiring (lines 11–13). When the distance between $x_{nearest}$ and x_{rand} is smaller than η, the standard steering function of RRT* is applied (line 15). Since CE_Extend is not effective when x_{near} is close to x_{rand}, the parameter η is introduced.

The function $Plan\_DoubleTrees$, detailed in Algorithm 6, updates $\mathcal{T}$ and $\mathcal{T}_e$ using the newly selected x_{new} and it is illustrated in Figure 5.3. Solid circles and thick lines represent vertices and edges of $\mathcal{T}$, respectively. Hollow circles and dash lines represent vertices and edges of $\mathcal{T}_e$, respectively. In line 2 of Algorithm 6, nodes in $\mathcal{T}_e$, except children of x_{new}, which are close to x_{new} are returned as X_{near} using $Near$ (Figure 5.3(b)). X_{near} are within a ball of radius $r = \gamma_{RRT*}\left(\frac{\log(n)}{n}\right)^{\frac{1}{d}}$

Algorithm 5 Cost-Aware RRT*

$\mathcal{V} \leftarrow \{x_0\}, \mathcal{E} \leftarrow \emptyset, \mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E})$
$\mathcal{V}_e \leftarrow \{x_0\}, \mathcal{E}_e \leftarrow \emptyset, \mathcal{T}_e \leftarrow (\mathcal{V}_e, \mathcal{E}_e)$
while stopping_criterion is false **do**
  $x_{rand} \leftarrow$ Sample($\mathcal{X}_{free}$)
  $x_{nearest} \leftarrow$ Nearest_Neighbor($\mathcal{T}, x_{rand}$)
  if $\|x_{nearest}, x_{rand}\| > \eta$ **then**
    $\mathcal{P}^* \leftarrow$ CE_Extend($x_{nearest}, x_{rand}$)
    $X_{CE} \leftarrow$ Fragment($\mathcal{P}^*, \rho$)
    $x_{new} \leftarrow x_{ce_1} \in X_{CE}$
    $[\mathcal{T}, \mathcal{T}_e] \leftarrow$ Plan_DoubleTrees($\mathcal{T}, \mathcal{T}_e, x_{nearest}, x_{new}$)
    for $\{x_{ce_i}\}_{i=2,\cdots,n} \in X_{CE}$ **do**
      $\mathcal{T}_e \leftarrow$Plan_SingleTree($\mathcal{T}_e, x_{ce_{i-1}}, x_{ce_i}$)
    end for
  else
    $x_{new} \leftarrow$Steer($x_{nearest}, x_{rand}$)
    if CollisionFree($x_{nearest}, x_{new}$) **then**
      $[\mathcal{T}, \mathcal{T}_e] \leftarrow$ Plan_DoubleTrees($\mathcal{T}, \mathcal{T}_e, x_{nearest}, x_{new}$)
    end if
  end if
end while
return $\mathcal{T}, \mathcal{T}_e$

centered at x_{new} as explained in [53], where n is the number of vertices in $\mathcal{T}_e$. In line 3, the best parent node x_{min} of x_{new} is found based on the cost to x_{new} via x_{min} from $Choose\_Parent$ (Figure 5.3(c)). Here, $Cost(x)$ calculates the cost of the minimum-cost path from x_0 to x. If x_{min} is not included in $\mathcal{T}$, $Update\_Tree$ function (Algorithm 9) is called to insert all ancestor nodes of x_{min} from $\mathcal{T}_e$ to $\mathcal{T}$ (Figure 5.3(d)). By including nodes in $\mathcal{T}_e$ through $Update\_Tree$, CARRT* gains more chances for extending the tree in a nonmyopic manner. Lines 10–23 are for rewiring. If the rewiring condition (line 11) is satisfied for x_{near}, the parent of x_{near} is shifted to x_{new} like RRT*. However, CARRT* checks whether x_{near} is included in $\mathcal{T}$ for x_{new}. If it is not in $\mathcal{T}$, the node x_{near} and a new edge (x_{new}, x_{near}) are added to $\mathcal{T}$ (lines 13–14). On the other hand, if $x_{near} \in \mathcal{T}$, the rewiring procedure is the same as RRT* (lines 16–18). For x_{new}, $\mathcal{T}_e$ is updated by deleting an edge from the parent of x_{near} to x_{near} (line 20) and including an edge (x_{new}, x_{near}) (line 21). This rewiring case is shown in Figure 5.3(f). The stopping criterion of CARRT* can be the number of iterations after reaching the goal region x_{goal}, the maximum number of iterations, or a time deadline.

A critical function of CARRT*, CE_Extend, is described below.

5.3.1 CE_Extend

The objective of CE_Extend is to help extending a tree from $x_{nearest}$ with a low-cost path to x_{rand}. A low-cost path is found using cross entropy path planning described in Section 4.3.1. We propose two different approaches to generate a path to x_{rand} and they are:

1. Motion primitives: $\{(u_1, t_1), \cdots, (u_m, t_m)\}$, where control input $u_j \in \mathcal{U}$ is applied over time duration of t_j for $j \in \{1, \ldots, m\}$.

2. Waypoint primitives: $\{(x_1, \cdots, x_m)\}$, where $x_j \in \mathcal{X}_{free}$ for $j \in \{1, \ldots, m\}$.

Algorithm 6 Plan_DoubleTrees($\mathcal{T}, \mathcal{T}_e, x_{nearest}, x_{new}$)

$\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{new}\}, \mathcal{V}_e \leftarrow \mathcal{V}_e \cup \{x_{new}\}$
$X_{near} \leftarrow \text{Near}(\mathcal{T}_e, x_{new})$
$x_{min} \leftarrow$Choose_Parent$(X_{near}, x_{nearest}, x_{new})$
if $x_{min} \notin \mathcal{V}$ **then**
  $\mathcal{T} \leftarrow$Update_Tree$(\mathcal{T}, \mathcal{T}_e, x_{min})$
else
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{min}, x_{new})\}$
end if
$\mathcal{E}_e \leftarrow \mathcal{E}_e \cup \{(x_{min}, x_{new})\}$
for $x_{near} \in X_{near} \setminus \{x_{min}\}$ **do**
  $c' \leftarrow \text{Cost}(x_{new}) + \mathcal{J}(\text{Line}(x_{new}, x_{near})$
  if $c' < \text{Cost}(x_{near})$ *AND* CollisionFree(x_{new}, x_{near}) **then**
    if $x_{near} \notin \mathcal{V}$ **then**
      $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{near}\}$
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{new}, x_{near})\}$
    else
      $x_{parent} \leftarrow \text{Parent}(\mathcal{T}, x_{near})$
      $\mathcal{E} \leftarrow \mathcal{E} \setminus \{(x_{parent}, x_{near})\}$
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{new}, x_{near})\}$
    end if
    $\mathcal{E}_e \leftarrow \mathcal{E}_e \setminus \{(\text{Parent}(\mathcal{T}_e, x_{near}), x_{near})\}$
    $\mathcal{E}_e \leftarrow \mathcal{E}_e \cup \{(x_{new}, x_{near})\}$
  end if
end for
return $\mathcal{T}, \mathcal{T}_e$

Algorithm 7 Plan_SingleTree($\mathcal{T}_e, x_{nearest}, x_{new}$)

$\mathcal{V}_e \leftarrow \mathcal{V}_e \cup \{x_{new}\}$
$X_{near} \leftarrow \text{Near}(\mathcal{T}_e, x_{new})$
$x_{min} \leftarrow$Choose_Parent($X_{near}, x_{nearest}, x_{new}$)
$\mathcal{E}_e \leftarrow \mathcal{E}_e \cup \{(x_{min}, x_{new})\}$
for $x_{near} \in X_{near} \setminus \{x_{min}\}$ **do**
  $c' \leftarrow$Cost(x_{new}) + $\mathcal{J}$(Line(x_{new}, x_{near})
  if $c' <$ Cost(x_{near}) *AND* CollisionFree(x_{new}, x_{near}) **then**
    $\mathcal{E}_e \leftarrow \mathcal{E}_e \setminus \{(\text{Parent}(\mathcal{T}_e, x_{near}), x_{near})\}$
    $\mathcal{E}_e \leftarrow \mathcal{E}_e \cup \{(x_{new}, x_{near})\}$
  end if
end for
return $\mathcal{T}_e$

Algorithm 8 Choose_Parent($X_{near}, x_{nearest}, x_{new}$)

$x_{min} \leftarrow x_{nearest}$
$c_{min} \leftarrow$ Cost($x_{nearest}$) + $\mathcal{J}$(Line($x_{nearest}, x_{new}$))
for $x_{near} \in X_{near} \setminus \{x_{nearest}\}$ **do**
  $c' \leftarrow$Cost(x_{near}) + $\mathcal{J}$(Line(x_{near}, x_{new}))
  if $c' < c_{min}$ *AND* CollisionFree(x_{near}, x_{new}) **then**
    $x_{min} \leftarrow x_{near}$, $c_{min} \leftarrow c'$
  end if
end for
return x_{min}

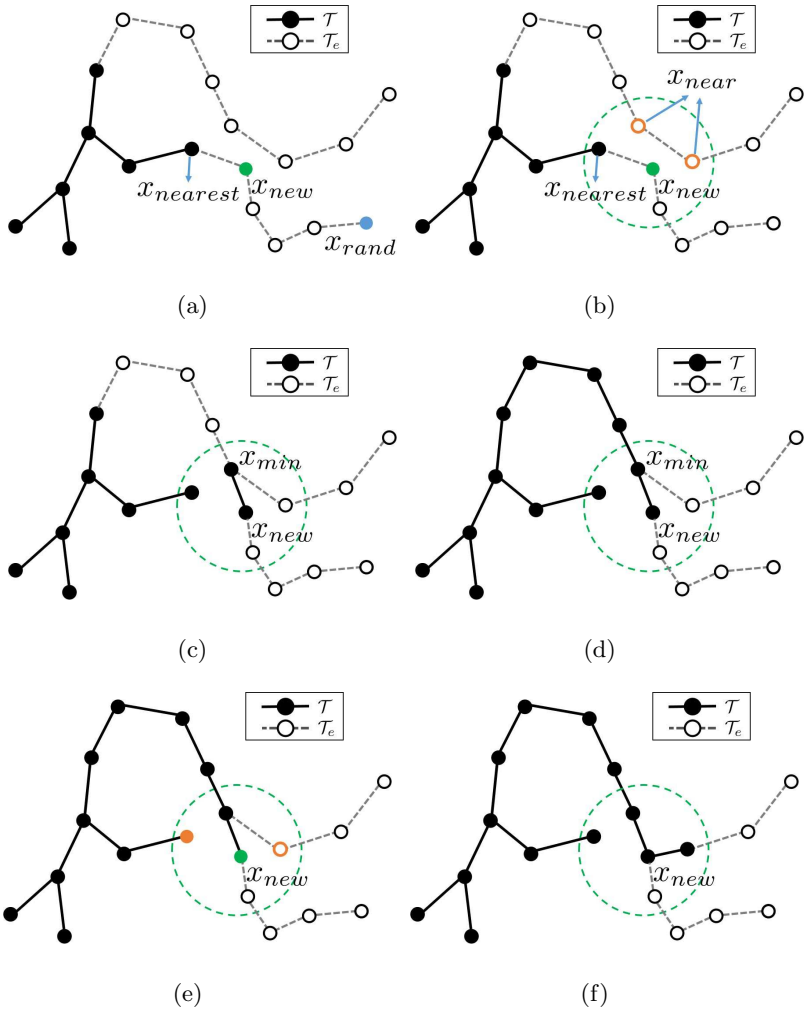


Figure 5.3: An illustration of Algorithm 6. The tree $\mathcal{T}$ is represented by solid circles and thick lines while the extended tree $\mathcal{T}_e$ is represented by solid circles and thick lines with additional branches represented by hollow circles and dash lines. (a) $x_{nearest}$ is selected from $\mathcal{T}$ for x_{rand}. (b) X_{near} of x_{new} are chosen from $\mathcal{T}_e$. (c) Among X_{near}, the node with the minimum cost is selected as x_{min}. (d) Once x_{min} is determined from $\mathcal{T}_e$, the ancestor nodes of x_{min} are added to $\mathcal{T}$ through $Update\_Tree$. (e,f) For x_{near}, the rewiring procedure is performed by deleting the edge from the parent of x_{near} to x_{near} and inserting an edge (x_{new}, x_{near}) to $\mathcal{T}_e$ if x_{near} is not included in $\mathcal{T}$.

Algorithm 9 Update_Tree($\mathcal{T}, \mathcal{T}_e, x_{min}$)

$x' \leftarrow x_{min}$
while $x' \notin \mathcal{V}$ **do**
  $x'' \leftarrow \text{Parent}(\mathcal{T}_e, x')$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{x'\}, \mathcal{E} \leftarrow \mathcal{E} \cup \{(x'', x')\}$
  $x' \leftarrow x''$
end while
return $\mathcal{T}$

The motion or the waypoint primitives are sampled from $p(\cdot; v)$, where $p(\cdot; v) = \mathcal{N}(\cdot|\mu, \Sigma)$ and $v = (\mu, \Sigma)$. The Gaussian distribution is used for the ease of computation but other distribution from a natural exponential family can be easily applied [91]. The basic idea behind cross entropy is to optimize v using the Kullback-Leibler divergence, such that the resulting $p(\cdot; v)$ is biased towards the optimal parameter distribution. In our case, we want $p(\cdot; v)$ to become a delta function and represent a minimum cost path.

Motion Primitives

The initial value $v_0 = (\mu_0, \Sigma_0)$ of v is set by selecting μ_0 and Σ_0 such that the motion primitive causes the shortest path from x_{near} to x_{rand} and each control is applied at an equal time interval. At the k-th iteration, we generate N_t trajectory samples $\mathcal{P}_1, \cdots, \mathcal{P}_{N_t}$ using motion primitives sampled from $p(\cdot; v_{k-1})$. In order to find a path to reach a region near x_{rand} while having the minimal accumulated cost as stated in the path planning problem (5.2), we define two parameters $\eta_a > 0$ and $\eta_b > 0$. We first select at most $\eta_a N_t$ trajectories whose terminal positions are close to the goal region and then select at most $\eta_b N_t$ elite trajectories with the lowest costs. After these two layers of filtering, we update $v_k = \{\mu_j^{mp}, \Sigma_j^{mp}\}_{j=1}^m$

from the selected elite samples $\pi_{el} = \{\pi_i\}_{i=1}^{\eta_b N_t}$, where

$$\begin{aligned} \mu_j^{mp} &= \frac{1}{\eta_b N_t}\Big[\sum_{l=1}^{\eta_b N_t} u_{lj}, \sum_{l=1}^{\eta_b N_t} t_{lj}\Big]^T \\ \Sigma_j^{mp} &= \frac{1}{\eta_b N_t}\sum_{l=1}^{\eta_b N_t}\big([u_{lj}, t_{lj}]^T - \mu_j^{mp}\big)\big([u_{lj}, t_{lj}]^T - \mu_j^{mp}\big)^T, \end{aligned} \tag{5.3}$$

where (u_{lj}, t_{lj}) is the j-th motion primitive parameterized by the l-th elite sample. In order to guarantee that the update procedure always reduces the cost, we remember the best sample in the previous step and reuse the sample in the current update procedure if it has a lower cost than current elite samples. With these two layers of filtering, we can find a solution to (5.2) which always arrives at the goal region. The maximum number of iterations in the CE_Extend function is fixed to N_{iter} in our algorithm. After iterations, the minimum cost path $\mathcal{P}^*$ from x_{near} to x_{rand} is selected. In our implementation, a discretized version of the optimal path planning problem is used as described in [64].

Waypoint primitives

The initial value v_0 of v is set such that a direct path from $x_{nearest}$ to x_{rand} is equally covered by each Gaussian component of v_0. At the k-th iteration, we sample N_t waypoints $\{X_i\}_{i=1}^{N_t}$ from $p(\cdot; v_{k-1})$, where $X_i = \{x_{ij}\}_{j=1}^{m}$. Assuming that there exists the parameterization function $g(\cdot)$ which gives the trajectory $\mathcal{P}(t)$ and control input $u(t)$ given two waypoints x_1 and x_2, i.e., $((\mathcal{P}(t), u(t)) = g(x_1, x_2))$, then for each X_i, we can construct a path $\mathcal{P}_i$ from x_{near} to x_{rand} by connecting x_{near} to x_{i1}, x_{ij} to $x_{i(j+1)}$ for all j, and x_{im} to x_{rand} using the function g. In order to connect between two waypoints, we used Dubins' curves [92] as a solution of the boundary condition problem. Dubins' curve can be characterized by six curves such as $\{LSL, RSR, LSR, RSL, RLR, LRL\}$ and each curve consists of three path segments such as turning right (R), turning left (L), and straight

line(S). For six curves, we derived the equations which represents the length of each path. Each solution for six length equations is formulated in Appendix E.

For the case of complex space with multiple obstacles, it is difficult to set the initial value v_0 since the connection from x_{ij} to $x_{i(j+1)}$ cannot be found easily due to obstacles. In such a case, a path obtained from RRT is used as v_0. For each $\mathcal{P}_i$, we compute the accumulated cost along the path. Unlike the path using motion primitives, all paths can reach x_{rand}, so we use a single-layer filtering without considering the terminal cost. We select $\eta_b N_t$ elite samples with the lowest accumulated costs. From the selected elite samples $X_{el} = \{X_i\}_{i=1}^{\eta_b N_t}$, we update $v_k = \{\mu_j^{wp}, \Sigma_j^{wp}\}_{j=1}^{m}$, where

$$\begin{aligned} \mu_j^{wp} &= \frac{1}{\eta_b N_t} \sum_{l=1}^{\eta_b N_t} x_{lj} \\ \Sigma_j^{wp} &= \frac{1}{\eta_b N_t} \sum_{l=1}^{\eta_b N_t} (x_{lj} - \mu_j^{wp})(x_{lj} - \mu_j^{wp})^T. \end{aligned} \tag{5.4}$$

5.4 Analysis of CAPP

In this section, the properties of the proposed algorithm are evaluated. We first describe and prove the probabilistic completeness of CARRT*. Then we analyze the optimality of CARRT*.

5.4.1 Probabilistic Completeness

As shown in [29], RRT is probabilistically complete and has an exponentially fast convergence rate for the probability of finding a solution if one exists as more vertices are added to the RRT tree. Furthermore, it is shown that its variants such as RRT* and the rapidly-exploring random graph (RRG) are also probabilistically complete [53]. Based on the analysis of those works, we extend the proof of the

probabilistic completeness of CARRT*. For the proof, we first need the following terms from [93].

Definition 1. *(Local controllability). A robot is defined to be local controllable if and only if,* $\forall x \in \mathcal{X}_{free}$*, the set of configurations that the robot can reach within a finite time contains a ball centred at* x*.*

In this work, we assume that a robot with holonomic dynamics is locally controllable as explained in the Definition 1. We denote the closed ball of radius ϵ centered at configuration x by $\mathcal{B}_\epsilon(x)$ and the set of all such balls by $\mathcal{B}_\epsilon$.

Definition 2. *(ϵ-reachable set). Let $\epsilon > 0$ and $x \in \mathcal{X}$ be given. The ϵ-reachable set of x, denoted by $R_\epsilon(x)$, is defined by*

$$R_\epsilon(x) = \{x' \in \mathcal{B}_\epsilon(x) |\ A \text{ path } \pi \text{ from } x \text{ to } x' \text{ is entirely contained in } \mathcal{B}_\epsilon(x)\}.$$

Definition 3. *(ϵ-free feasible path). A path π is said to be ϵ-free feasible, if the minimal distance between π and the obstacle region is ϵ, i.e., for all $x \in \pi$, $\mathcal{B}_\epsilon(x) \subset \mathcal{X}_{free}$.*

In terms of the notation used in this dissertation, the notion of probabilistic completeness can be stated as follows.

Definition 4. *(Probabilistic completeness). Suppose that there exists an ϵ-free feasible path from the starting point to the goal region. Then a sampling based motion planning algorithm is probabilistically complete if the probability that the algorithm finds an ϵ-free feasible path from the starting point to the goal region approaches one as the number of samples goes to infinity.*

Unlike RRT and its variants, such as RRG and RRT*, which select a discrete input from a finite set of inputs, CARRT* selects a random input from a continuous input space for tree extension. Therefore, we first show that even if RRT selects an input from a continuous input space, it has the probabilistic completeness property and then show that the proposed algorithm is probabilistically complete.

Theorem 1. *An RRT, which selects an input from a bounded continuous input set for tree extension, is probabilistically complete.*

Proof. See Appendix A. □

Theorem 2. *CARRT* is probabilistically complete.*

Proof. See Appendix B. □

5.4.2 Asymptotic optimality

In this section, we show that a path found by CARRT* converges to the optimal solution with probability one. Depending on the distance between $x_{nearest}$ and x_{rand}, CARRT* adopts two extension procedures: one is a long extension using *CE_Extend* when $\|x_{nearest} - x_{rand}\| > \eta$ and the other is a standard extension based on *Steer*. Considering the asymptotic optimality of RRT* [53], we prove the asymptotic optimality of CARRT* by showing the asymptotic optimality for the case when a tree is extended using long extensions. Let $c^* = c(\pi^*)$ be the cost of an optimal path, and c_n be the cost of the minimum-cost solution returned by CARRT* at the end of the n-th iteration. We denotes ζ_d the volume of a unit ball in the d-dimensional space.

Theorem 3. *If* $\gamma_{RRT^*} > (2(1 + \frac{1}{d}))^{\frac{1}{d}}(\frac{Vol(\mathcal{X}_{free})}{\zeta_d})^{\frac{1}{d}}$*, then CARRT* algorithm is asymptotically optimal, that is,* $\mathbb{P}(\{\lim_{n\to\infty} c_n = c^*\}) = 1$.

Proof. See Appendix C. □

5.5 Simulation and experimental results

In this section, we discuss results obtained from the proposed algorithm in simulation and experiments. The following planning problems are considered:

(P1) To find a trajectory which minimizes the accumulated position-dependent cost when a robot traverses over a field with environmental parameters;

(P2) To find a trajectory which minimizes the consumed energy when a robot traverses over a complex terrain; and

(P3) Motion planning for humanoid robots in a high dimensional space.

(P1) is an instance of PDPP while (P2) and (P3) are instances of SDPP. In order to evaluate the performance of the proposed algorithm, we compared the proposed method against the standard RRT [29], RRT* [53][1], and cross entropy path planning algorithms, SCERRT* and TCERRT*, from [79]. SCERRT* and TCERRT* are RRT* based path planning algorithms using cross entropy. While SCERRT* samples a random point from the state space, TCERRT* samples from a parameterized trajectory space [79]. Cross entropy based path planning algorithms including the proposed algorithm require several user defined parameters and they are set as follows: $N_t = 100, N_{iter} = 50, \eta_a = 0.2, \eta_b = 0.1, m = 8$. Additional parameters for SCERRT* and TCERRT* are set to the same values used in [79]. The proposed method has two versions depending on the method used to generate a path in CE_Extend and they will be referred as CARRT*(M)

[1]While there are a number of extensions to RRT*, including [80, 81, 82, 84, 85, 94], Some of them are focused on minimizing the path length. In addition, there is no publicly available implementation of their work for fair comparison, hence, we only compared to RRT and RRT*.

and CARRT*(W). CARRT*(M) uses motion primities (see Section 5.3.1) while CARRT*(W) uses waypoint primitives (see Section 5.3.1). For (P1) and (P3), only CARRT*(W) is applied since a simple dynamics model is assumed in both problems.

All simulations were run on a desktop with a 3.4 GHz Intel Core i7 processor with 16 GB of memory. The proposed method was implemented both in MATLAB and C/C++[2]. For problems (P1) and (P2), all algorithms were implemented in MATLAB. For (P3), the C/C++ implementation of the proposed method was compared to the C implementation of RRT* provided by the authors of [53][3].

5.5.1 (P1) Cost-Aware Navigation in 2D

Consider a surveillance region $\mathcal{Q} = [-15, 15]^2$ in which a robot operates. $\mathcal{Q}$ consists of a free space, $\mathcal{Q}_{free}$, and a collection of obstacles, $\mathcal{Q}_{obs}$. A simple linear motion model is assumed for the robot dynamics.

Position Dependent Cost Function

As a robot moves from one location to another, the robot is penalized by an instantaneous cost at its current location. The cost of the entire region $\mathcal{Q}$ can be represented as a costmap $\mathcal{C} : \mathcal{Q} \to \mathbb{R}^+$. Given a costmap, we can formulate a position-dependent cost function $\mathcal{J}_{\text{PDPP}}$ as follows:

$$\mathcal{J}_{\text{PDPP}}\Big(\pi_u(t)\Big) = \left(\frac{1}{T}\int_0^T \mathcal{C}(\mathcal{P}_u(t))dt + \epsilon \int_0^T dt\right), \tag{5.5}$$

where $\mathcal{P}_u(t) \subset \pi_u(t)$ is a path considering only the positions of $\pi_u(t)$, such that $\mathcal{P}_u(t) \in \mathcal{Q}$. The line integral of $\mathcal{C}$ along the path of a robot is the total accumulated

[2]The algorithm will be available at http://cpslab.snu.ac.kr/software

[3]http://sertac.scripts.mit.edu/web/Software.

cost until the robot arrives at the goal region. The last term with $\epsilon > 0$ is introduced to favor a shorter trajectory for trajectories with the same accumulated cost.

Costmap

We assume that a costmap is defined over $\mathcal{Q}$, representing environmental parameters. Since we cannot measure the environmental parameter at every location, we use a nonparametric regression method, namely Gaussian Process Regression (GPR), to predict the environmental parameter at a site where no measurement is made. GPR has been widely used as a nonparametric regression technique for modeling complex physical phenomena, including nonstationary geostatistical data analysis [95], nonlinear regression [96], wireless signal strength estimation [97], indoor temperature field modeling [98], and terrain mapping [99]. Compared to parametric regression methods, nonparametric regression methods are more expressive and yield better generalizability. Hence, methods such as Gaussian processes are well suited to model complex environments. We assume that the environmental parameter of interest, which defines the costmap $\mathcal{C}$, follows a Gaussian process. Suppose we have made n samples from the surveillance region $\mathcal{Q}$. Let $\mathbf{q} = \{q^{(1)}, q^{(2)}, \ldots, q^{(n)}\}$ be the set of locations at which samples are taken and $y^{(1)}, y^{(2)}, \ldots, y^{(n)}$ be the measurements. If $\mathcal{C}(q)$ is a GP, it can be fully described by its mean function $\mu(q) = \mathbb{E}(\mathcal{C}(q))$ and covariance function $K(q, q') = \mathbb{E}((\mathcal{C}(q) - \mu(q))(\mathcal{C}(q') - \mu(q')))$, i.e.,

$$\mathcal{C}(q) \sim GP(\mu(q), K(q, q')). \tag{5.6}$$

Let $Y = [y^{(1)} y^{(2)} \cdots y^{(n)}]^T$. Then, for any $q_* \in \mathcal{Q}$, the expected value of the cost function at q_* is

$$\mathcal{C}(q_*) = K_*^T (K + \sigma_w^2 I)^{-1} Y, \tag{5.7}$$

where $K = [K_{ij}]$ is the kernel matrix such that $K_{ij} = k(X_i, X_j)$, $K_* = [k(x_1, x_*) \cdots k(x_n, x_*)]^T$ [90].

In this dissertation, we used the squared exponential as a kernel function,

$$k(X, X') = \sigma_f^2 \exp\left(-\frac{1}{2\sigma_l^2} \sum_{m=1}^{n} (X_m - X'_m)\right), \tag{5.8}$$

where σ_f and σ_l are hyperparameters of the kernel.

The resulting costmap is defined over the continuous space and this is different from previous approaches where a cost function is defined over a discretized space. If cost values are known at discrete sites, the same method can be applied to smooth the costmap. Hence, by applying GPR, we obtain a costmap of infinite resolution. Note that the appropriate parameters for the kernel function and the variance of the observation model have to be learned from the data samples before the costmap is applied.

Evaluation

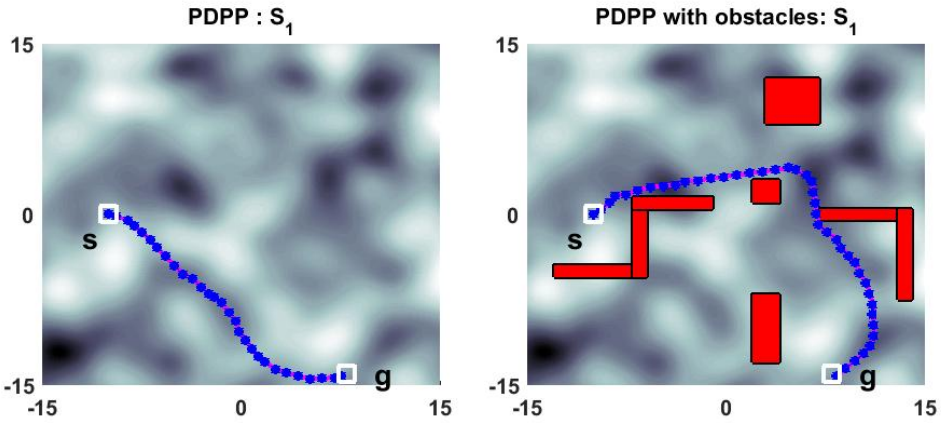
We generated four scenarios with different costmaps from a Gaussian process given in (5.6) with kernel function (5.8), where $\sigma_f^2 = 1.0$ and $\sigma_l^2 = 5.0$. Scenarios are indexed from S_1 to S_4. We used the Gaussian process MATLAB toolbox [90] for modeling the costmap. To consider obstacles, we added several obstacles placed at the same locations in all scenarios. The starting position x_0 and the goal region x_{goal} are randomly chosen for each scenario. Two examples of scenarios used in simulation are shown in Figure 5.4. The left figures in Figure 5.4(a) are scenarios without obstacles and the right figures are with obstacles. The costmap is shown as contours with different colors. The high cost region is represented in white and the low cost region is represented in black. The white squares represent x_0 and x_{goal} and they are marked by letter S and G, respectively. The obstacle region is represented by red rectangles.

For all scenarios, we performed a total of 10 simulations of each algorithm using different pre-specified random seeds and set the time deadline to terminate the algorithm after 3000 seconds. We first compared different algorithms for scenarios without obstacles. The left figures in Figure 5.4 show costmaps and the trajectory with the minimum cost found by the proposed algorithm over the costmap. Figure 5.5 shows the average trajectory cost found by each algorithm as a function of the running time for 10 trials for each scnerio. Since the cost of RRT was too high, so it is not included in Figure 5.5. The proposed algorithm shows excellent performance in all cases compared to other algorithms.

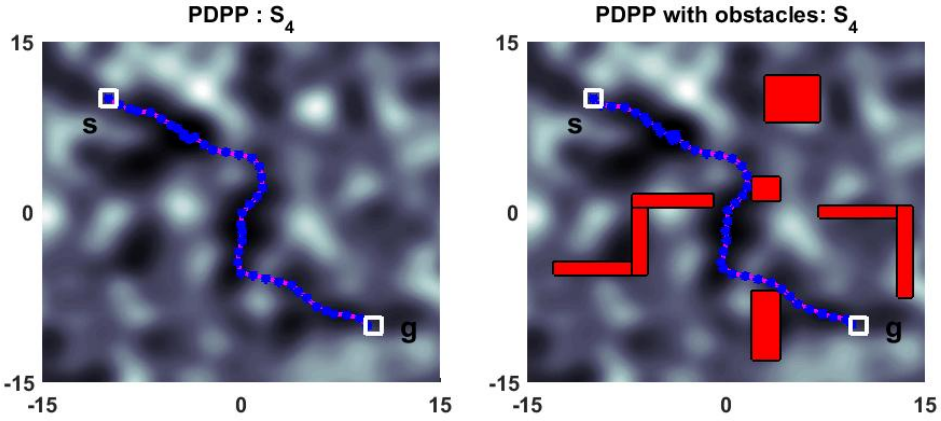
We also performed the same simulation but with obstacles. The trajectory with the minimum cost found by the proposed algorithm is shown in the right figures of Figure 5.4. The average trajectory costs from different algorithms are shown in Figure 5.6. For all cases, the proposed algorithm converges to the optimal solution faster than other algorithms. For all simulations, the proposed algorithm finds a trajectory with less cost in the beginning compared to other algorithms, thanks to longer extensions for nonmyopic search used in the proposed method. Note that the difference between CARRT* and RRT* is not large in all simulations since relatively simpler cases are considered.

5.5.2 (P2) Complex Terrain Navigation

In order to model a terrain with different elevations, we make a digital terrain model (DTM) or a height map using pairs of location and elevation values. We can display the terrain using the Gazebo simulator [100] based on the DTM.



(a) S_1



(b) S_4

Figure 5.4: Examples of scenarios used in simulation and the optimal trajectory with the minimum cost found by the proposed algorithm. The color over the field represent the costmap. The darker color represents low cost and the bright color represents high cost. Red rectangles are obstacles.

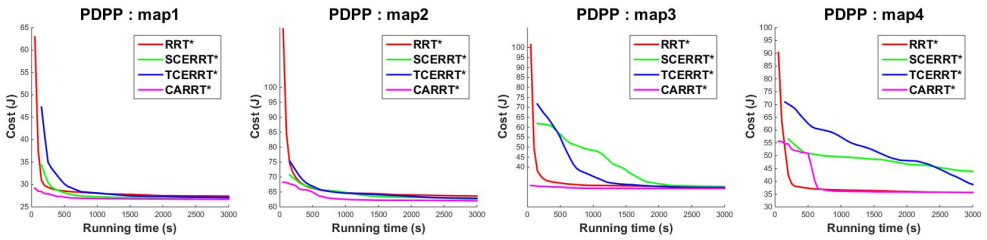


Figure 5.5: The average trajectory cost as a function of deadlines for each scenario (without obstacles). The average cost of each algorithm is computed from 10 independent trials. Legend: RRT* (red), SCERRT* (green), TCERRT* (blue), and CARRT* (magenta).

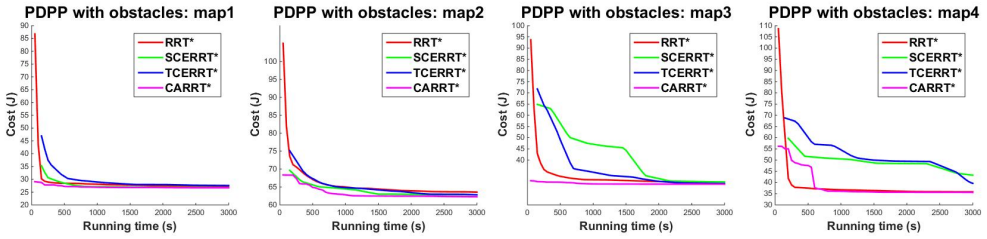


Figure 5.6: The average trajectory cost as a function of deadlines for each scenario (with obstacles). The average cost of each algorithm is computed from 10 independent trials. Legend: RRT* (red), SCERRT* (green), TCERRT* (blue), and CARRT* (magenta).

Dynamic Model

For this problem, we use a dynamical model of a two-wheeled robot (Dubins' car). However, the proposed method can be applied to other dynamic models with suitable energy functions. Let (q_x, q_y) be the position of a robot and θ be its heading. Suppose $v(t)$ is the translational velocity and $u(t)$ is the angular velocity, then the whole dynamics is as follows:

$$\dot{q}_x(t) = v(t)\cos(\theta(t)), \quad \dot{q}_y(t) = v(t)\sin(\theta(t)), \quad \dot{\theta}(t) = u(t).$$

In simulation, v is fixed at a constant value of $1\,m/s$. A discrete-time version of the above dynamical model is used in simulation where the sampling period is set to $0.1\,s$.

State Dependent Cost Function

Unlike the position dependent cost function used in (P1), we consider a state dependent cost function, $\mathcal{J}_{\text{SDPP}}$, which depends on the energy function $E : \mathcal{X} \to \mathbb{R}^+$. In order to measure the energy consumption by a robot along its path, we use the mechanical work (MW) criterion [43]. The MW criterion evaluates the quality of a path by regarding a nonnegative increment as a penalized cost. In other words, it is assumed that there is no cost loss for the negative slope of the time derivative of the energy function [43]. Using MW, we can define a cost function as follows:

$$\mathcal{J}_{\text{SDPP}}(\pi_u(t)) = \left(\int_0^T \mathbf{I}_{\left\{\frac{\partial E}{\partial t} > 0\right\}} \frac{\partial E(\pi_u(t))}{\partial t} dt + \epsilon \int_0^T dt \right), \tag{5.9}$$

where $\mathbf{I}_{\{.\}}$ is the indicator function. Hence, the cost is penalized when the time derivative of the energy function has a positive slope along the path of a robot

and the cost function $\mathcal{J}$ captures the penalized cost from $t = 0$ to $t = T$. The last term with ϵ plays the same role as the last term in (5.5).

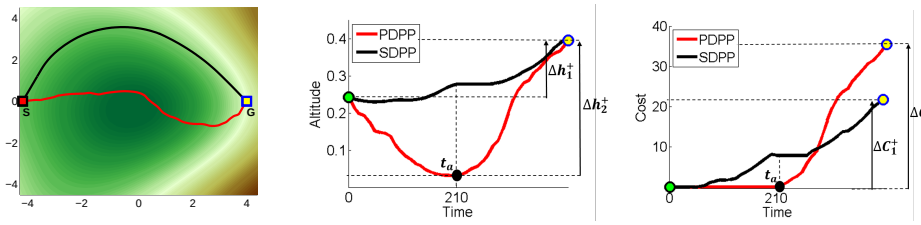
Consider a costmap represents terrain elevation. Then accumulated cost depends on the consumed energy along the path. Figure 5.7 shows a simple terrain with different elevations and paths found by PDPP and SDPP. There is a valley between the start position (red square) and the goal region (yellow square). The goal region is located at a slightly higher altitude than the start position. Since the cost function in PDPP considers the instantaneous cost at each location, it passes through the lowest altitude region. On the other hand, SDPP avoids the valley As shown in Figure 5.7(b) and 5.7(c), PDPP spends no energy until time t_a but spends more energy than SDPP to reach the goal region in terms of energy consumption. While the path is longer, SDPP produces a path which requires lower energy than PDPP. This example illustrates that when the energy consumption has to be minimized, the algorithm must be able to avoid local valleys and hills in the terrain.

Energy Function

Since we aim to minimize the energy consumption of a robot traversing a complex terrain, we can use the energy function $E : \mathcal{X} \to \mathbb{R}$ along the trajectory of the robot. Given the energy function E, the total energy consumption can be computed by integrating $\partial E/\partial t$ along the path of the robot as shown in (5.9). The energy at state x can be defined as follows:

$$E(x) = K(x) + P(x), \tag{5.10}$$

where K is the kinetic energy and P is the potential energy. Since we assume the translational velocity is constant, we ignored the kinetic energy in this dissertation. For the discussion below, the argument x is omitted.



(a) (b) (c)

Figure 5.7: (a) A simple scenario with trajectories found by PDPP (red) and SDPP black). The start position and goal region are marked by red and yellow squares, respectively. Different colors represent different elevations.(b) The altitude of a robot as a function of time. (c) The cumulative energy consumption as a function of time.

The potential energy P can be defined as:

$$P = P_g + P_f + P_a, \tag{5.11}$$

where $P_g = m_v g h$ is the potential energy due to gravity, $P_f = \mu m_v g \cos(\phi) \Delta s$ is the potential energy for friction, and $P_a = \frac{1}{2} \rho_f S_a C_d v^2 \Delta s$ is the potential energy for the aerodynamic force. Here, m_v is the mass of the vehicle, h is the height, g is the gravitational acceleration constant, ϕ is the heading angle with respect to the ground plane, μ_f is the coefficient of friction, Δs is the moving distance at velocity v, C_d is the drag coefficient, ρ_f represents the density of the fluid, and S_a represents the front area of the vehicle.

Evaluation

The cost function (5.9) based on the energy function defined in the previous section is applied to all algorithms. All parameters used in this simulation study

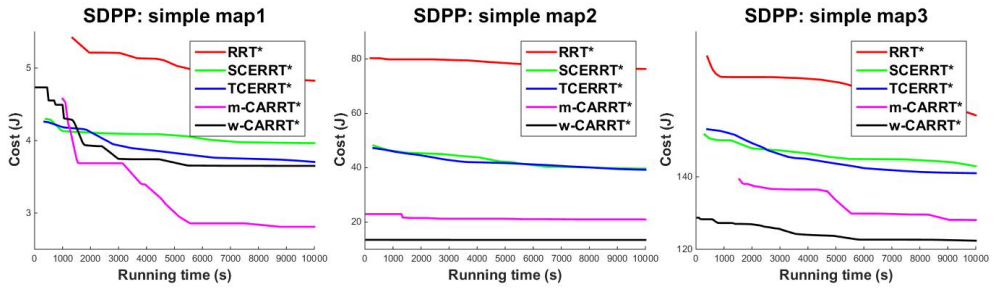


Figure 5.8: The average trajectory cost as a function of times for each scenario. The average cost of each algorithm is computed from 10 independent trials. Legend: RRT* (red), SCERRT* (green), TCERRT* (blue), CARRT*(M) (magenta) and CARRT*(W) (black).

are set based on a Pioneer 3DX robot as follows: $m_v = 9kg$, $\mu_f = 0.001$, $\rho_f = 1.22Ng^2m^{-4}$, $S_a = 0.087m^2$, $C_d = 0.05$. We assume that there is no obstacle in the terrain but it can be easily introduced.

We generated three simple scenarios to study behaviors of different algorithms. The first and second scenario has a valley and a hill, respectively, between x_0 and x_{goal}. The third scenario has a few valleys and hills. Each algorithm is run for 10 times to compute the average values. We extended the termination time to 10,000 s, compared to (P1), since it took more time to find the optimal path for these cases. Average energy consumptions for trajectories found by different algorithms are shown as a function of the running time in Figure 5.8. The optimal trajectory tends to go around local minimum or maximum regions, so it avoids the valley and hill. The average minimum energy consumed by paths generated by each algorithm is shown as a function of times in Figure 5.8.

We then tested algorithms with four complex scenarios. Each scenario has multiple valleys and hills, unlike the simple scenarios. We randomly set x_0 and x_{goal}

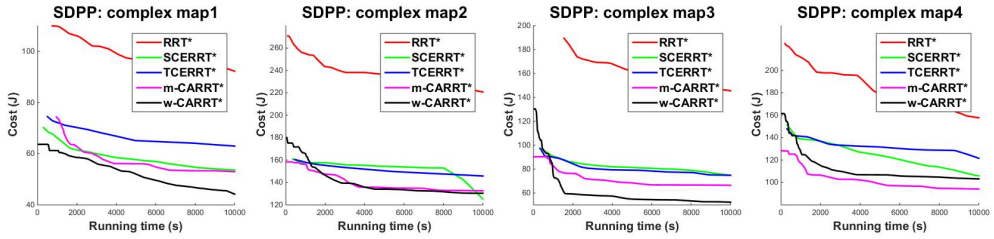


Figure 5.9: The average trajectory cost as a function of times for each scenario. The average cost of each algorithm is computed from 10 independent trials. Legend: RRT* (red), SCERRT* (green), TCERRT* (blue), CARRT*(M) (magenta) and CARRT*(W) (black).

for all scenarios like in P1. Figure 5.9 shows the average minimum energy consumption as a function of times for each algorithm. Once again, for all scenarios, the proposed algorithm shows the best result. Likewise, the proposed algorithm finds a trajectory with less cost at the beginning compared to other algorithms. Furthermore, the final cost of result from the proposed algorithm is much less than other algorithms, since the state space is more complex than the case of P1.

In order to validate trajectories obtained from simulation, we ran realistic physics-based simulations using the robot operation system (ROS) [101]. The ROS simulator provides a plugin for a differential-drive robot, such as Pioneer 3DX. The path found by a path planning algorithm is used to guide a robot in ROS and Gazebo is used to display how a robot moves in the terrain. Snapshots from the simulation are shown in Figure 5.10. The arrows represent a path found by an algorithm. As shown before, CARRT*(M) goes around the valley since it is more energy-efficient to move around a local valley.

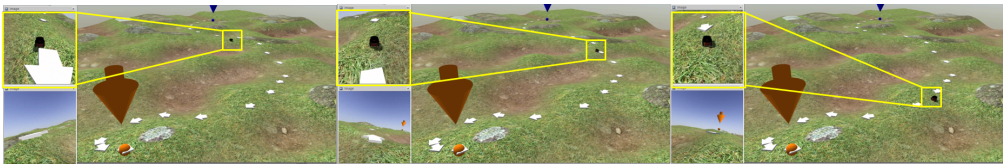


Figure 5.10: A sequence of snapshots of a Pioneer 3DX robot following a given trajectory for the proposed algorithm in a ROS+Gazebo simulator. The white arrows represent points from energy-efficient paths found by CARRT*(M).

5.5.3 (P3) Humanoid Motion Planning

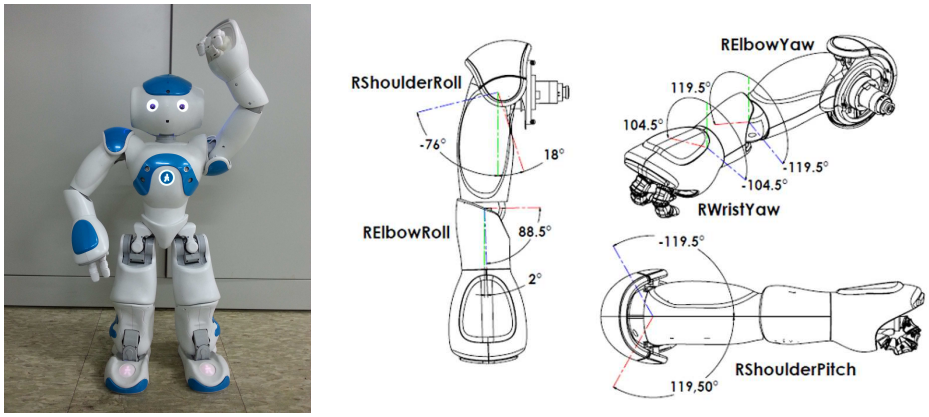
State Space

A humanoid robot Nao shown in Figure 5.11(a) is used for the experimental evaluation of CARRT*. It has a total of 25 degrees of freedom. Since we are interested in manipulating arms of a Nao and each arm has five joints, we limit the state space of Nao to five dimensional space, i.e., $\mathcal{X} \subset \mathbb{R}^5$. Figure 5.11(b) shows five joints. The configuration space $\mathcal{X}_{free}$ is defined within the joint angle constraints specified in Figure 5.11(b) and $\mathcal{X}_{obs}$ contains not only obstacles but also the torso and the head of a Nao. We also consider a ten-dimensional space for controlling both arms of a Nao.

Energy function

In this section, we define the energy function suitable for the motion of a humanoid robot. Unlike a ground vehicle, the consumed energy of a humanoid robot can be obtained by computing the joint torques. The time derivative of energy function $\partial E/\partial t$ shown in (5.9) can be defined as follows:

$$\frac{\partial E}{\partial t} = f(x) \cdot \frac{\partial x}{\partial t}, \tag{5.12}$$



(a) (b)

Figure 5.11: (a) A Nao humanoid robot from Aldebaran. (b) Joints of the right arm of Nao and joint angle constraints [102].

where $f : \mathcal{X} \to \mathbb{R}^d$ represents the torque function described in [103]. Joint torque values are obtained from the torque function using humanoid manipulator dynamics [103]. In this work, we ignore the velocity and acceleration of joints and approximate the torque function by considering only joint angles. When we measured the current flow on each joint from a robot in real experiments, the value was too noisy. So we moved arms slow enough along the given trajectory to obtain accurate measurements. Hence, the assumption barely affects the estimation of the consumed energy.

Evaluation

We compared the proposed algorithm against the standard RRT and RRT*. We generated two scenarios for single-arm and dual-arm manipulations: one is to position both hands to hold a box on a table by moving arms without colliding with the table and the another is to hand over an object from the right hand to

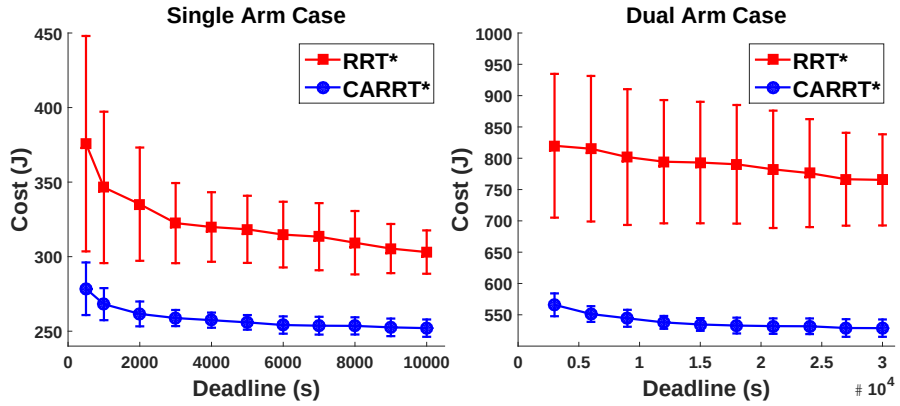
the left hand by avoiding obstacles. Both scenarios are shown in Figure 5.13 and 5.14. For a single arm case, the same tasks are applied to only one arm and the other arm is kept in the goal region.

For all scenarios, we performed a total of 10 simulations of each algorithm using different pre-specified random seeds and set the time deadline to terminate the algorithm as 10,000 seconds for a single-arm case and 30,000 seconds for a dual-arm case, respectively.

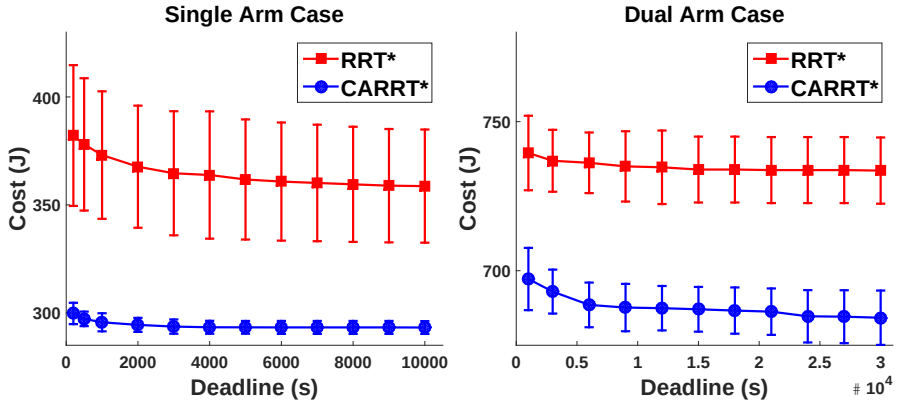
Figure 5.12 shows the average cost of 10 trials, along with one standard deviation error bars. The cost of the standard RRT was too high to be included in Figure 5.12. In Scenario 1, the average costs of RRT for single and dual arm cases were 896.6 J and 1882.4 J, respectively, and the average costs were 553.6 J and 1144.8 J, respectively, in Scenario 2. Even if RRT* converges to the optimal solution given enough time, the cost of its initial solution is relatively high and the converge rate is extremely slow. This is because RRT* requires dense sampling to improve the path by refining the tree. The proposed algorithm finds the near-optimal solution fast with fewer samples since it extends the tree using waypoints found by considering the quality of the path. Thus, the proposed algorithm can find a good solution faster. For both scenarios, the proposed algorithm shows the best results.

Experiments

The processes of following the trajectory using a real humanoid robot for two scenarios are shown in Figure 5.13 and Figure 5.14, respectively. Each figure shows results obtained from RRT* and CARRT*, respectively, for the dual-arm case. The red lines shown in Figure 5.14 represent the trajectories of end-effectors (i.e., both hands).

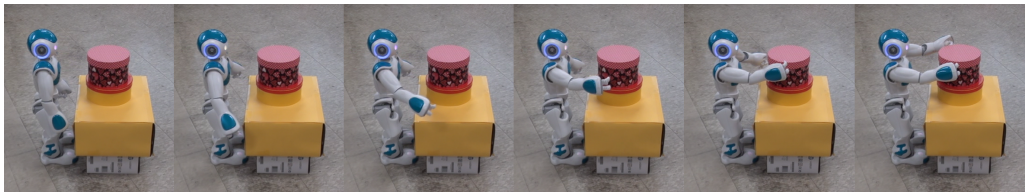


(a) Scenario 1

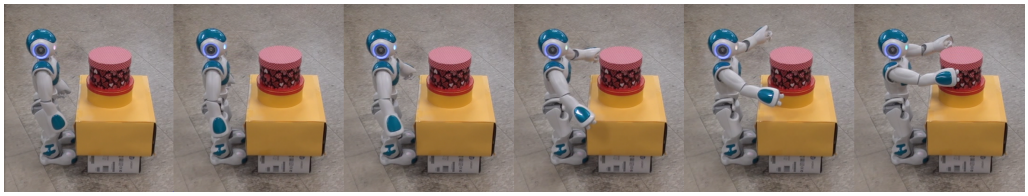


(b) Scenario 2

Figure 5.12: The average trajectory cost (in joules) as a function of deadlines for two manipulation scenarios. The average cost of each algorithm is computed from 10 different runs with random seeds and one standard deviation is shown as error bars.

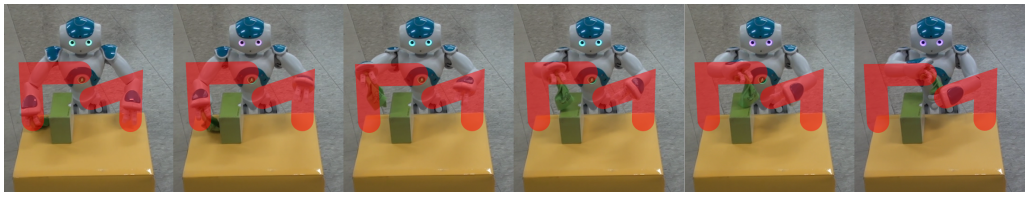


(a) RRT*

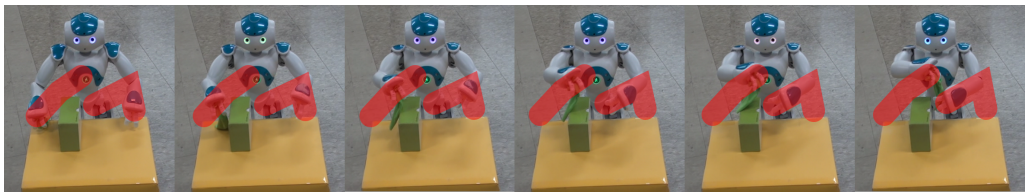


(b) CARRT*

Figure 5.13: Scenario 1: a robot moves hands to hold an object on a table. (a) and (b) show the process of following a trajectory obtained from RRT* and CARRT*, respectively from left (initial pose) to right (goal pose).



(a) RRT*

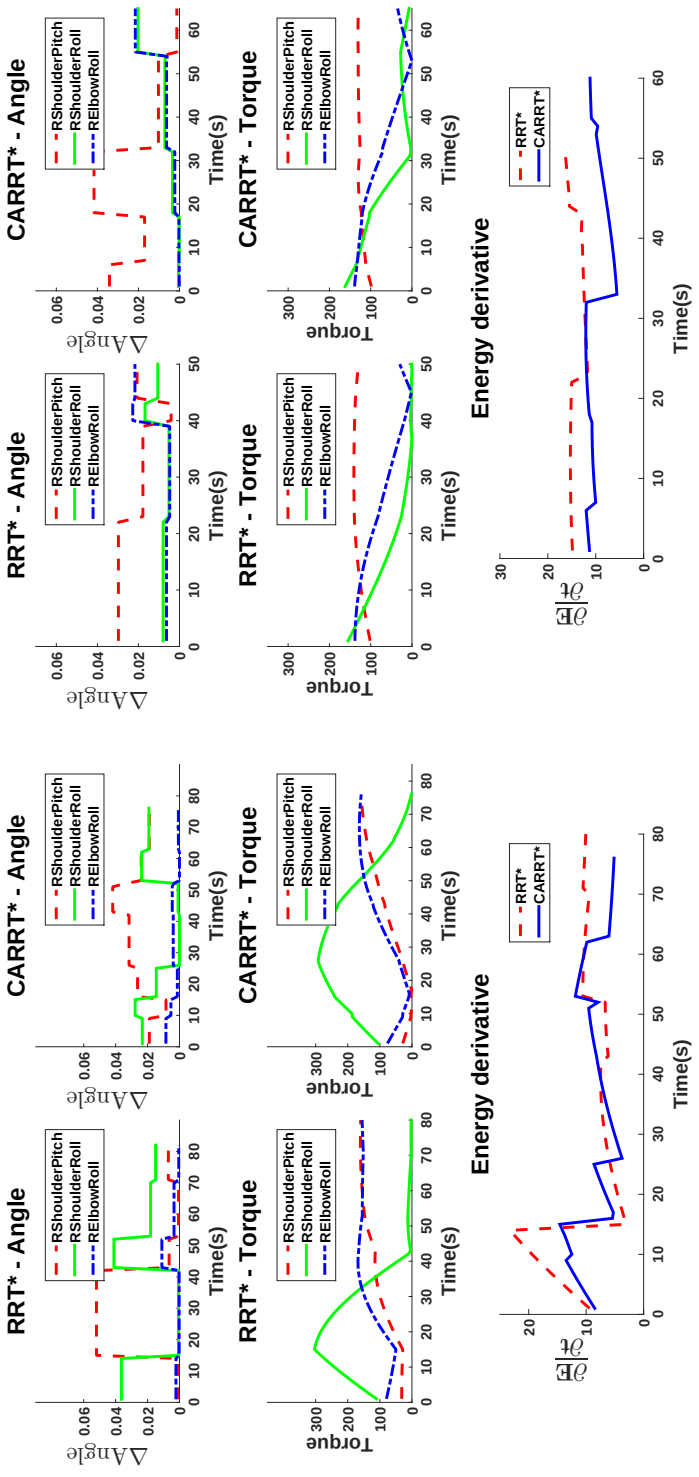


(b) CARRT*

Figure 5.14: Scenario 2: a robot hands over an object from the right hand to the left hand while avoiding an obstacle on a table. Red lines in both figures represent trajectories of hands.

We demonstrate the results of experiments through the torque value and angle derivative as a function of time shown in Figure 5.15. As explained in Section 5.2, the proposed algorithm tends to plan a motion while maintaining a specific configuration of the whole arm for energy efficiency. As shown in snapshots of the first column of Figure 5.13, two hands of the robot are located under the table. In order to place two hands in the position for holding an object, the robot should first raise two arms to its side by moving the RShoulderRoll joint. There is a big difference between two algorithms in terms of the energy consumption between $0\,s$ and $15\,s$. The energy consumption of RRT* is relatively high during this interval. RRT* immediately starts to move the RShoulderRoll joint while keeping the current joints stationary. However, the initial position of the first scenario requires more energy when it moves the RShoulderRoll joint. As shown in Figure 5.15(a), the torque value of RShoulderRoll is high at the beginning. Since a multiplication of the torque and the angle derivative represents the consumed energy derivative, it is more efficient to move other joints in the beginning, e.g., RShoulderPitch and RElbowRoll, to configurations with less torque values and then move the RShoulderRoll joint as less as possible, which is what CARRT* does.

The results are more distinguishable in the second scenario. While RRT* moves directly to the goal state with less consideration about energy-efficient configurations, the proposed algorithm first lifts both arms vertically and then move arms horizontally to hand over an object (see red lines shown in Figure 5.14 for the trajectory). It is confirmed from the torque and angle derivative values shown in Figure 5.15(b). Since all joints, especially RShoulderRoll, have large torque values between $0\,s$ and $18\,s$ as shown in Figure 5.15(b), the proposed algorithm changes only RShoulderPitch while raising arms. Then, it gradually increases an-



(a) Scenario 1

(b) Scenario 2

Figure 5.15: The angle change and torque value on each joint as a function of time for (a) first scenario and (b) second scenario. Top figures show how much each joints moves and bottom figures show the loaded torque on each joint. The red, green, and blue line represents the values of RShoulderPitch, RShoulderRoll and RElbowRoll, respectively.

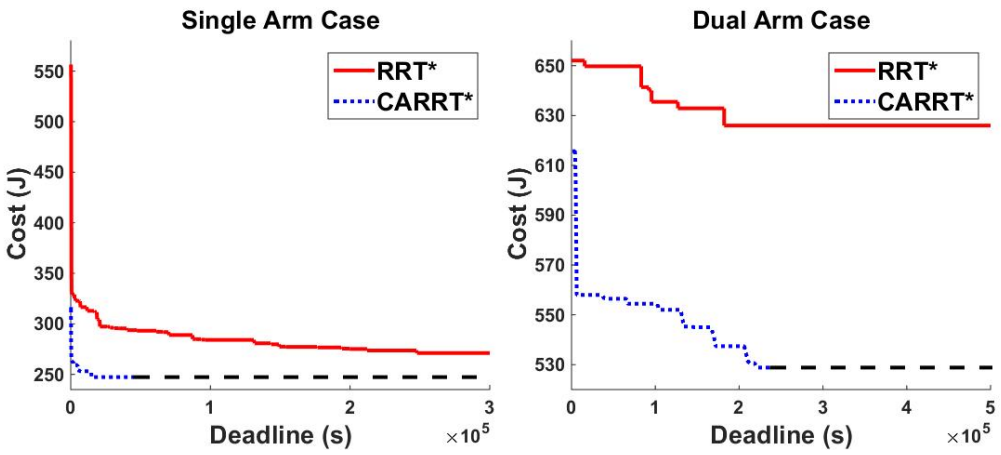


Figure 5.16: Costs of paths found by RRT* and the proposed algorithm over longer time deadlines (in seconds) for one trial.

gle derivatives of RSholderRoll and RElbowRoll after the torque value of those joints are reduced.

Even for longer deadlines, the proposed algorithm shows superior performance compared to RRT* as shown in Figure 5.16. Since dense sampling is required to find the optimal solution in RRT*, much more time is needed in a high dimensional space. Overall, the proposed algorithm tends to move joints for relatively less torque values while maintaining a specific configuration of other joints to reduce the consumed energy.

5.6 Summary

In this chapter, we have presented a cost-aware path planning algorithm which finds the minimum cost trajectory in a complex configuration space. When a costmap over a configuration space is available, the proposed algorithm finds a suitable trajectory of a robot with the minimum cost. Furthermore, it finds a

highly energy-efficient path for traversing a complex terrain with different elevation or performing a manipulation task in a high dimensional space. The proposed method takes advantages of a sampling-based RRT* for exploration and nonmyopic tree extension using a stochastic optimization method, cross entropy (CE). In simulation and experiments using a humanoid robot, the proposed algorithm finds the more cost-efficient path in a shorter time.

Chapter 6

Efficient Informative Path Planning

This chapter extends CAPP developed in chapter 5 to informative path planning for mobile sensor networks. Our information gathering algorithm is inspired by the information gathering algorithm based on RRG by Hollinger et al. [55]. In [55], the asymptotic optimal path is guaranteed for single agent and unnecessary nodes are efficiently pruned. We follow the similar approach but our algorithm is based on a more computationally efficient information gathering algorithm for multiple agents.

In this chapter, we propose a RRG-based planner, called CE-IPP, which extends the RIG planner [55]. CE-IPP uses cross entropy (CE) [36], a stochastic optimization method, to efficiently estimate the reachable information gain. The CE framework allows us to choose a (near) optimal informative trajectory among trajectory samples which satisfy the budget constraint. The proposed algorithm guarantees the asymptotic optimality like RRG and ensures to find a (near) optimal informative path to the goal region which satisfies the cost budget constraint,

if a such path exists. For each node in the graph, the proposed algorithm updates the lower bound of the reachable information gain of the most informative path to the goal region, which passes through the node. As more nodes are added to the graph, the lower bound of the reachable information gain of each node improves and converges to the maximum value in the limit.

6.1 Problem formulation

Let $\mathcal{Q}$ be a region, in which a robot (or a mobile sensor) performs its assigned sensing tasks. Let $\mathcal{X} \subset \mathbb{R}^n$ be the state space of the mobile sensor, where state $x \in \mathcal{X}$ includes the position $q \in \mathcal{Q}$. The motion model of the mobile sensor has the form:

$$x_{t+1} = f(x(t), u(t)), \tag{6.1}$$

where $u(t) \in \mathcal{U} \subset \mathbb{R}^p$ is the control input applied at time t.

Let $\mathcal{P}$ be the trajectory of a mobile sensor, which is obtained from the motion model (6.1) for given $u(t)$ from $t = 0$ to $t = T \in R^+$, where T is the termination time. We assume in this dissertation that the motion of the mobile sensor is deterministic. Let Σ be a set of all collision-free paths, such that, for $\mathcal{P} \in \Sigma$, $\mathcal{P}(t) \in \mathcal{X}_{free}$ for $t \in [0, T]$. Let $\Omega \subset \Sigma$ be a set of all feasible paths, such that, for $\mathcal{P} \in \Omega$, $\mathcal{P}(0) = x_{init}$ and $\mathcal{P}(T) \in x_{goal}$, where $x_{init} \in \mathcal{X}$ is the initial state and $x_{goal} \subset \mathcal{X}$ is the goal region. Let $c : \Sigma \to \mathbb{R}^+$ be a cost function, e.g., the path length or the energy consumption of a robot following the path. The goal of a path planning problem is to find the optimal control input and time parameterizing the feasible trajectory which minimizes the cost function. We assume that the cost function returns strictly positive value for any collision-free path and it is monotonic, additive, and bounded.

An IPP problem finds a path with the maximum information gain while considering the cost of the path. We can formulate an IPP problem as the following optimization problem by placing the cost of a path as a budget constraint:

$$\arg\max_{\mathcal{P}\in\Sigma} \quad \mathcal{I}\Big(\mathcal{P}\Big) \quad \text{subject to } c(\mathcal{P}) \quad \leq \quad \mathcal{B}, \tag{6.2}$$

where the function $\mathcal{I} : \Sigma \to \mathbb{R}^+$ returns the information collected along the path and $\mathcal{B} \in \mathbb{R}^+ < \infty$ is a budget for the maximum allowed cost. Generally, $\mathcal{I}$ is submodular, i.e., it has the diminishing return property. It is known that the problem of finding the optimal solution of a discrete version of (6.2) is NP-hard [104, 105]. Hence, a greedy algorithm is used in practice by discretizing the search space. If a path is defined in a continuous space, then solving (6.2) becomes more challenging.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are finite sets of vertices and edges, respectively. We denote $\mathcal{P}_a^b$ a trajectory initiated at a and arrived at b (i.e., $\mathcal{P}_a^b(0) = a$ and $\mathcal{P}_a^b(T) = b$). Consider two partial path $\mathcal{P}_{a1}^{b1}$ and $\mathcal{P}_{a2}^{b2}$ with different termination times T_1 and T_2, respectively. If $\mathcal{P}_{a1}^{b1}(T_1) = \mathcal{P}_{a2}^{b2}(0)$, then we denote $\mathcal{P}_{a1}^{b1}|\mathcal{P}_{a2}^{b2}$ as the concatenated trajectory, i.e.,

$$\mathcal{P}_{a1}^{b1}|\mathcal{P}_{a2}^{b2}(t) := \begin{cases} \mathcal{P}_{a1}^{b1}(t) \text{ for all } t \in [0, T_1], \\ \mathcal{P}_{a2}^{b2}(t - T_1) \text{ for all } t \in (T_1, T_1 + T_2]. \end{cases}$$

Given a vertex $v \in \mathcal{V}$, a path via v can be defined as follows.

Definition 5. *A v-trajectory $\mathcal{P}^v$ is a feasible concatenated trajectory $\mathcal{P}_{a1}^{b1}|\mathcal{P}_{a2}^{b2}$, such that $\mathcal{P}_{a1}^{b1}(T_1) = \mathcal{P}_{a2}^{b2}(0) = v$, $\mathcal{P}_{a1}^{b1}|\mathcal{P}_{a2}^{b2}(0) = x_{init}, \mathcal{P}_{a1}^{b1}|\mathcal{P}_{a2}^{b2}(T_1 + T_2) \in x_{goal}$, and $c(\mathcal{P}_{a1}^{b1}|\mathcal{P}_{a2}^{b2}) = c(\mathcal{P}_{a1}^{b1}) + c(\mathcal{P}_{a2}^{b2}) \leq \mathcal{B}$.*

With a slight abuse of notation, we will allow $v \in \mathcal{V}$ to be used in place of v's corresponding state $x \in \mathcal{X}$. Let $\mathcal{P}_{ALG,n}^v$ be the n-th v-trajectory via a fixed vertex

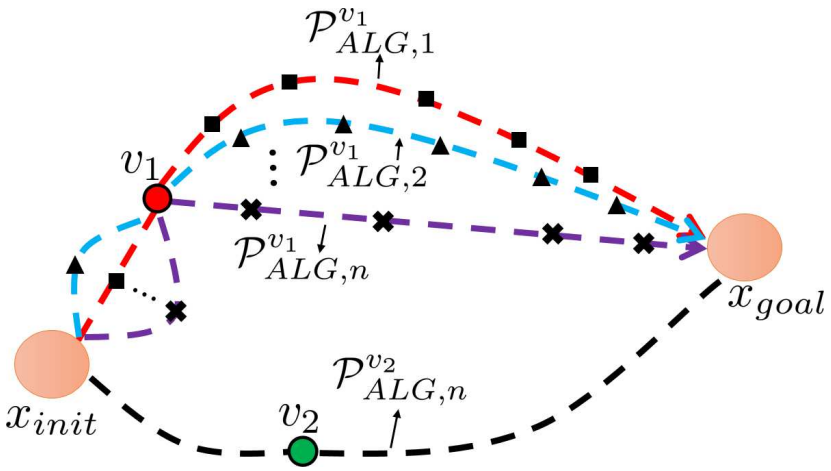


Figure 6.1: Illustration of $\mathcal{P}^{v}_{ALG,n}$ produced by an algorithm ALG. Given x_{init} and x_{goal}, the algorithm ALG finds trajectories $\mathcal{P}^{v}_{ALG,n}$ for all fixed vertices shown in red and green circles (e.g., v_1, v_2). A trajectory $\mathcal{P}^{v_1}_{ALG,n}$ is a n-th trajectory which passes through the fixed vertex v_1.

v produced by an algorithm ALG. Since $\mathcal{P}^{v}_{ALG,n}$ follows the Definition 5, it starts at x_{init}, reaches the goal region and has the cost less than the budget $\mathcal{B}$. For fixed vertices v_1 (red circle) and v_2 (green circle), $\mathcal{P}^{v_1}_{ALG,n}$ and $\mathcal{P}^{v_2}_{ALG,n}$ are shown in Figure 6.1. For any fixed vertex $v \in \mathcal{V}$, we use $g_{v,n} = \mathcal{I}(\mathcal{P}^{v}_{ALG,n})$ to denote the amount of information obtained along $\mathcal{P}^{v}_{ALG,n}$ and define $g_v = \max g_{v,j}$ for $1 \leq j \leq n$. Then the maximum of $g_{v,n}$, g^*_v, can be formulated as follows:

$$g^*_v = \limsup_{n \to \infty} \; g_{v,n}. \tag{6.3}$$

Since the algorithm constructs a graph $\mathcal{V}$, a path from x_{init} to v for $v \in \mathcal{V}$ can be obtained from the graph, so it estimates $\mathcal{P}^{x_{goal}}_{v}$, which is a partial path of $\mathcal{P}^{v}_{ALG}$. If there exists a vertex v which gives no $\mathcal{P}^{x_{goal}}_{v}$ due to the budget cost, g_v at v is set to 0. We can guarantee that $g^*_v < \infty$ for all $v \in \mathcal{V}$. Since the cost function returns monotonically positive value for any collision-free path, any v-trajectory

has finite length, so the amount of information is bounded. Let $\{\mathcal{V}_i^{ALG}\}_{i\in\mathbb{N}}$ be the set of vertices in the graph obtained from ALG at the end of iteration i. Assuming that we can update g_v at any fixed vertex $v \in \mathcal{V}$ as n increases, we can have g_v^* for all $v \in \{\mathcal{V}_i^{ALG}\}$ as $i \to \infty$ since $n \to \infty$ as $i \to \infty$. Then the maximum value among g_v^* for all $v \in \{\mathcal{V}_i^{ALG}\}$ is the maximum amount of information which can be obtained from the ALG, i.e.,

$$\max_{v\in\{\mathcal{V}_i^{ALG}\}} g_v^* \quad \text{as } i \to \infty. \tag{6.4}$$

The optimal informative path planning problem asks for finding a feasible path with maximum g_v^*, so it can be redefined as follows:

Problem 1. *Given g_v^* for all $v \in \mathcal{V}$, find a trajectory via v^* such that $v^* = \arg\max_{v\in\mathcal{V}} g_v^*$. If there exists no such path, then failure is reported.*

By solving Problem 1, we can find the asymptotically optimal path to the goal via a vertex $v \in \mathcal{V}$, which the vertex has the maximum g_v^* value among all vertices included in the graph $\mathcal{G}$. We propose an incremental graph structure based algorithm that utilizes a stochastic optimization method to generate informative trajectories satisfying constraints in terms of the cost budget. This stochastic optimization based method allows for the rapid generation of near optimal trajectories even for complex information quality objectives.

6.2 Cost-Aware informative path planning (CAIPP)

This section deals with the application of the CE method to informative path planning problem. We present the CE based informative path planning (CE-IPP) algorithm which finds a path to the goal region while maximizing information gain within the budget constraint. CE-IPP is based on RRG for optimal planning to

the goal region and the cross entropy method for solving maximization problem with inequality constraint. The key procedures can be outlined as:

1. Expand the graph to reach the goal region.

2. Generate a path to the goal within the budget at each node in the graph. This is done via the CE_Estimate function.

3. Update the lower bound of information at each node in the graph.

4. Repeat the above procedure until termination condition is satisfied.

To proceed the CE method, we need to define the probability density function (pdf) $p(\cdot;\theta)$ to update the CE parameters. It can be defined over a space of parameters θ used in generating trajectories in continuous space. The pdf $p(\cdot;\theta)$ can be also replaced by $M \times A$ probability matrix $\mathbf{P} = (P_{ma})$ for producing trajectories in discrete state space with M states, where A represents the number of actions taken at each state [106]. Thus P_{ma} denotes the probability of taking action a at state m and the summation of P_{ma} over a is 1 for all m.

6.2.1 Overall procedure

Algorithm 10 describes the main body of CE-IPP algorithm. The algorithm shares the overall structure as RRG. However, since the original RRG focuses on optimizing cost along the path without constraint, it has not been applied to information gathering problem which maximizes information with budget constraint. However, since the cost constraint is included additionally, it has not been able to apply the information gathering problem directly. In order to apply RRG to information gathering problem, we slightly modify RRG by adding the extra procedure when the graph grows. This procedure follows the approach presented in

[55]. Whenever the new node is added, edges are created from near nodes to the new node and vice versa. In order to determine whether to insert the edge to the graph or not, we check a cost integrated along the trajectory from x_{init} to the end node of the new edge (i.e., near node or new node). If the cost is over the budget, then the corresponding edge is deleted. Furthermore, in order to find a solution efficiently, we find a near-optimal path at each node using the cost-to-go value (i.e., the remained budget at the node) if there exists a feasible path. Then we can approximate possible information quantity to be able to collect at any node in the graph. Since CE-IPP follows RRG and each node has the achievable information quantity, it guarantees asymptotic optimality while satisfying the budget constraint and provide an efficient approach for a solution even if any node in the graph is not contained in the goal region.

The graph $\mathcal{V}$ initially contains only one point x_{init} as an initial node. The main control loop, lines 2-25, terminates once stopping criterion satisfies the condition. At each iteration, the graph grows by drawing a new sample x_{new} and then adding edges between x_{new} and its nearest vertices $x_{nearest}$ or between x_{new} and its near vertices x_{near} as in the RRG algorithm, where if the distance between $x_{nearest}$ and x_{new} is over δ, $Steer(x_{nearest}, x_{new})$ repositions x_{new} to be feasible and δ away from $x_{nearest}$ toward x_{new} (lines 5-7). A set of x_{near} in the graph that are close to x_{new} are returned as X_{near} through $Near$ function (line 8). X_{near} are within a ball of radius $r \propto (\frac{log(n)}{n})^{\frac{1}{d}}$ centered at x_{new} as explained in [53], where n is the number of vertices in $\mathcal{G}$ and d is the state dimension. The stopping criterion can be the number of iterations after reaching the goal region from any node in graph, the maximum number of iterations, or a time deadline. If the line segment connecting between the near nodes and the new node is not in collision with obstacles, then the edge is inserted into the graph

and Update_Bound function is called (lines 12-23). A function, Update_Bound, sets g_v at the node v, where g_v is the maximum value among the information quantities gathered along v-trajectories found so far. A v-trajectory is founded through CE_Estimate function.

Two critical functions of CE-IPP, Update_Bound and CE_Estimate, are described below.

6.2.2 Update_Bound

The Update_Bound procedure used in the CE-IPP algorithm is given in Algorithm 11 and shown in Figure 6.2. Whenever the new feasible point x_{new} is selected, since connections from the x_{new}'s all neighbor vertices to the x_{new} is tried, co-located nodes are generated at x_{new}. Figure 6.2(a) shows multiple co-located nodes of the vertex x_{new} (e.g., $x_{new\_1}, x_{new\_2}, x_{new\_3}$). These co-located nodes represent the vertices which have the same location but different paths $\mathcal{P}_{x_{init}}^{x_{new}}$. Because the cost of each $\mathcal{P}_{x_{init}}^{x_{new}}$ is different, each remaining budget at co-located nodes is determined according to cost of each $\mathcal{P}_{x_{init}}^{x_{new}}$. On the other hand, the co-located nodes of x_{near} arise when successful connections from the co-located nodes of x_{new} to x_{near} is done as shown in Figure 6.2(b). Given the co-located nodes at each vertex, Update_Bound function is performed. After updating the cost $\mathcal{C}_{x'\_i}$ using a co-located node x_i of x (line 3) and the remained budget $\mathcal{B}_{x'\_i}$ (line 4), CE_Estimate finds the near optimal trajectory $\mathcal{P}_{x'\_i}^{x_{goal}}$ initiated at each x'_i to the goal region within $\mathcal{B}_{x'\_i}$ and computes the information quantity $\mathcal{I}_{x'\_i}$ along $\mathcal{P}_{x_{init}}^{x'\_i} \cup \mathcal{P}_{x'\_i}^{x_{goal}}$ (line 5). If a path cannot be found at x'_i due to the budget .constraint, CE_Estimate function returns the empty $\mathcal{P}_{x'\_i}^{x_{goal}}$ and $\mathcal{I}_{x'\_i}$. Then we drop the co-located node x'_i, and repeat CE_Estimate for a different x'_i.

After setting a lower bound on possible information quantity along a path via

Algorithm 10 CE-IPP

Require: 1. Start position x_{init} and goal position x_{goal}

2. Budget $\mathcal{B}$

3. A ball of radius r for neighbors

Ensure: The most informative path from x_{init} to x_{goal} within $\mathcal{B}$

$\mathcal{V} \leftarrow \{x_{init}\}, \mathcal{E} \leftarrow \emptyset, \mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$ $\mathcal{C}_{x_{init}} \leftarrow 0$

while stopping_criterion is false **do**

  $x_{new} \leftarrow$ a random sample from $\mathcal{Q}$

  $x_{nearest} \leftarrow$ Nearest_Neighbor$(\mathcal{V}, x_{new})$;

  if $\|x_{nearest}, x_{new}\| > \delta$ **then**

    $x_{new} \leftarrow$ Steer$(x_{nearest}, x_{new})$

  end if

  $X_{near} \leftarrow$ Near(x_{new}, r)

  $g_{x_{new}} \leftarrow 0$

  $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{new}\}$

  $\mathcal{I}_{x_{new}} \leftarrow 0$

  for all $x_{near} \in X_{near}$ **do**

    if CollisionFree(x_{near}, x_{new}) **then**

      $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{near}, x_{new})\}$

      $g_{x_{new}} \leftarrow$ Update_Bound(x_{near}, x_{new})

    end if

  end for

for all $x_{near} \in X_{near}$ **do**
  if CollisionFree(x_{new}, x_{near}) **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{new}, x_{near})\}$
    $g_{x_{near}} \leftarrow$ Update_Bound(x_{new}, x_{near})
  end if
end for
$\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$
end while
Find v which satisfies (6.4)

Algorithm 11 Update_Bound(x, x')

Ensure: $g_{x'}$

$g_{x'} \leftarrow \max_{x'_i \in x'} \mathcal{I}_{x'_i}$
for all $x\_i \in x$ **do**
  $\mathcal{C}_{x'\_i} \leftarrow \mathcal{C}_{x\_i} + c(x\_i, x')$
  $\mathcal{B}_{x'\_i} \leftarrow \mathcal{B} - \mathcal{C}_{x'\_i}$
  $\langle \mathcal{P}^{x_{goal}}_{x'\_i}, \mathcal{I}_{x'\_i} \rangle \leftarrow$ CE_Estimate$(x', x_{goal}, \mathcal{P}^{x'_i}_{x_{init}}, \mathcal{B}_{x'_i})$
  if $\mathcal{I}_{x'\_i} > g_{x'}$ **then**
    $g_{x'} \leftarrow \mathcal{I}_{x'\_i}$
  end if
end for

x_{new}, a lower bound of each $\mathcal{I}_{x_{near}}$ along a path via the neighbors of x_{new} is set by choosing the best information quantity so far. Whenever any vertex x in the graph is selected as a neighbor of a newly steered vertex, the lower bound of information along a path via x is updated. Thus the information quantity along the path via x asymptotically increases to the optimal value.

6.2.3 CE_Estimate

The CE_Estimate function is the modified version of Algorithm 4 outlined in Section 5.3. The function returns a probabilistically near-optimal trajectory from any vertex v to the goal region using the remained budget $\mathcal{B}_v$ at v and the information quantity collected along the v-trajectory $\mathcal{P}^v$.

Unlike Algorithm 4 which focuses on samples minimizing a cost function, the function tries to generate samples which maximize an information function considering the budget constraint and it can be formulated as follows:

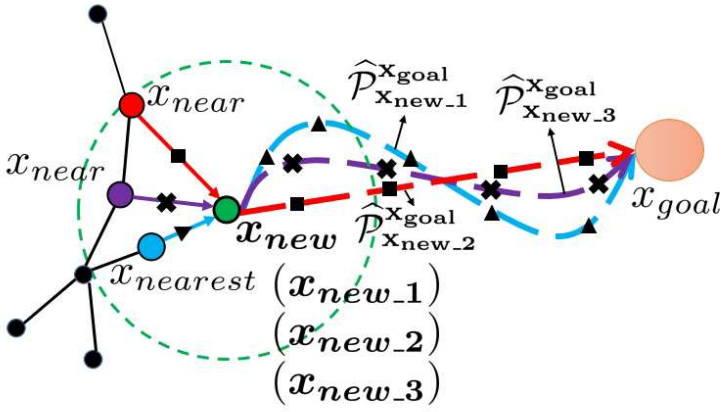
$$\max_{\mathcal{P}_v^{x_{goal}} \in \Omega} \mathcal{I}(\mathcal{P}^v_{x_{init}} \cup \mathcal{P}_v^{x_{goal}}) \text{ subject to } c(\mathcal{P}_v^{x_{goal}}) \leq \mathcal{B}_v, \tag{6.5}$$

where $\mathcal{P}^v_{x_{init}} \cup \mathcal{P}_v^{x_{goal}} = \mathcal{P}^v$ and $\mathcal{B}_v = \mathcal{B} - c(\mathcal{P}^v_{x_{init}})$. As shown in (6.5), the CE_Estimate function needs $\mathcal{P}^v_{x_0}$ as an input to find the near-optimal trajectory from v to the goal. Since the information function used in this dissertation is submodular, the prior trajectory to v is involved in determining $\mathcal{P}_v^{x_{goal}}$. By considering $\mathcal{P}^v_{x_0}$ as a part of the trajectory $\mathcal{P}^v$, the following property is satisfied.

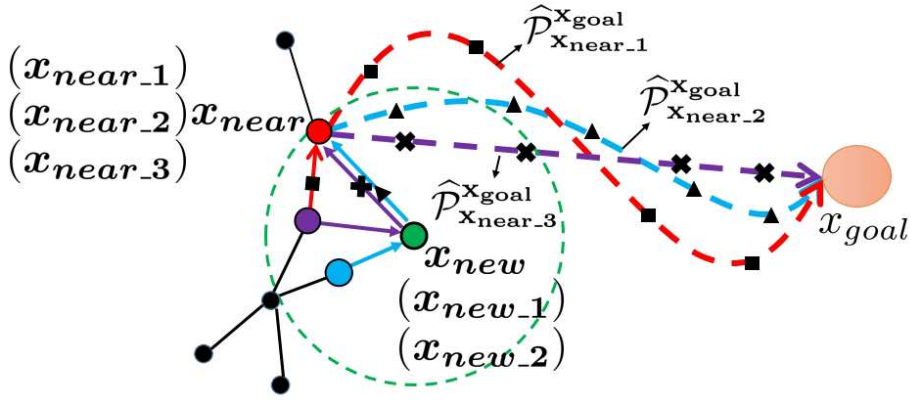
$$\mathcal{I}(\mathcal{P}^v) + \mathcal{I}(\mathcal{P}^v_{x_{init}} \cap \mathcal{P}_v^{x_{goal}}) \leq \mathcal{I}(\mathcal{P}^v_{x_{init}}) + \mathcal{I}(\mathcal{P}_v^{x_{goal}}), \tag{6.6}$$

where $\mathcal{P}^v_{x_{init}} \cap \mathcal{P}_v^{x_{goal}}$ represents the overlapped portion between two trajectories $\mathcal{P}^v_{x_{init}}$ and $\mathcal{P}_v^{x_{goal}}$.

It is important to generate trajectory samples from $p(\cdot;\theta_k)$ at $k = 0$ which cover the search space well. In this dissertation, it is assumed that $p(\cdot;\theta_k)$ follows



(a) Update procedure at x_{new}



(b) Update procedure at x_{near}

Figure 6.2: This figure shows the different paths initiated at multiple co-located nodes for each vertex in the graph. In (a), once x_{new} is inserted into the graph, since all edges from three neighbor vertices are added, there exists multiple co-located nodes of the vertex x_{new}. In contrast, in (b) edges from multiple co-located nodes of x_{new} to x_{near} generate co-located nodes at x_{near}. For each vertex in graph, Update_Bound function determines the best information quantity along each trajectory estimated at co-located nodes (e.g., $\widehat{\mathcal{P}}^{x_{goal}}_{x_{new\_i}}$ for $i = 1, 2, 3$ at x_{new}).

a normal distribution with the parameter $\theta_k = (\mu_k, \Sigma_k)$ like [79]. By adjusting the covariance Σ_0 to cover the region of interest, we can get those initial samples. However, there exists a limitation in initial sampling since a budget constraint is added in the procedure of generating samples. The coverage and the budget are in conflict each other (i.e., for good coverage, more budget is required and coverage can be poor for small budget). Thus we need to find the feasible region satisfying the budget for sampling. The parameter θ_k consists of m primitives. In this section, we assume that the primitives are waypoints and the budget represents the length of the path. This possible subset of states that satisfies the budget, can then be expressed in closed form in terms of the budget $\mathcal{B}$ as

$$\mathcal{X}_{sample\_free} = \{x \in \mathcal{X}_{free} | \|x - x'\|_2 + \|x_{goal} - x'\|_2 \leq \mathcal{B}\} \tag{6.7}$$

which is the general equation of an n-dimensional prolate hyperspheroid (i.e., a special hyperellipsoid). The focal points are x and x_{goal}, the transverse diameter is $\mathcal{B}$, and the conjugate diameters are $\sqrt{\mathcal{B}^2 - L^2}$ as shown in Figure 6.3. In other words, each waypoint should be sampled in $\mathcal{X}_{sample\_free}$ which is determined depending on the remained budget, i.e., $\mathcal{B}_v - c(x)$. Again we use the truncated normal distributions based on $\mathcal{X}_{sample\_free}$ by following (6.8).

$$f(x|\mu, \mu^-, \mu^+, \sigma) = \frac{\exp \frac{-(x-\mu)^2}{2\sigma^2}}{\sqrt{2\pi}\sigma[\Phi(\frac{\mu^+ - \mu}{\sigma}) - \Phi(\frac{\mu^- - \mu}{\sigma})]} \tag{6.8}$$

The lower and upper boundary of each sample for m-th primitive is computed based on the ellipse equation. In the setting of truncated normal distributions, all primitive samples are obtained by generating successively the components of

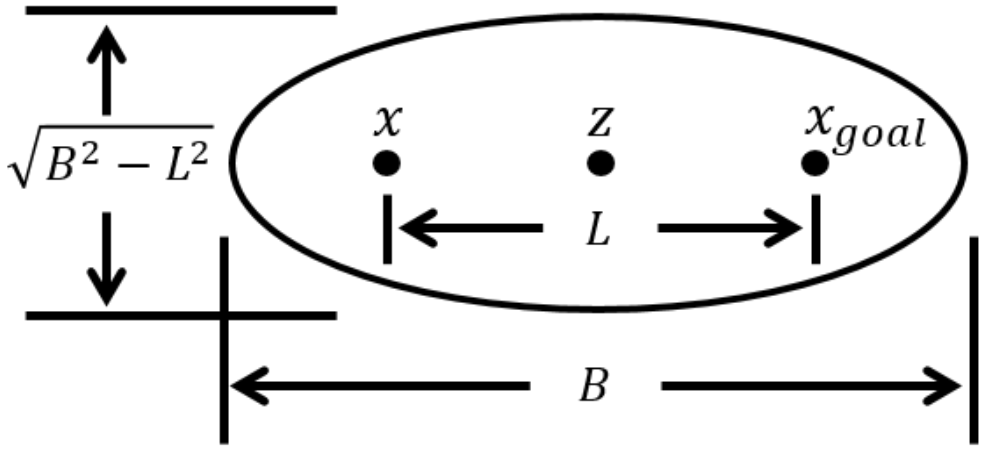


Figure 6.3: The sampling domain, $\mathcal{X}_{sample\_free}$, for a $\mathbb{R}^2$ problem seeking to satisfy the budget $\mathcal{B}$ is an ellipse with the initial state, x, and the goal state, x_{goal} as a focal points. A center of the ellipse is denoted as z. The shape of the ellipse depends on both initial and goal states and the budget $\mathcal{B}$.

$\mathcal{N}(\mu, \Sigma, \mathcal{X}_{sample\_free})$, i.e.,

$$\begin{aligned} x_1 &\sim \mathcal{N}(\mathbb{E}(\mu_1|\mu_2, \cdots, \mu_m), \mu_1, \mu_1^-, \mu_1^+, \sigma_1^2) \\ x_2 &\sim \mathcal{N}(\mathbb{E}(\mu_2|\mu_1, \mu_3, \cdots, \mu_m), \mu_2, \mu_2^-, \mu_2^+, \sigma_2^2) \\ &\vdots \\ x_m &\sim \mathcal{N}(\mathbb{E}(\mu_m|\mu_1, \cdots, \mu_{m-1}), \mu_m, \mu_m^-, \mu_m^+, \sigma_m^2) \end{aligned} \tag{6.9}$$

6.3 Analysis of CAIPP

In this section, we show that the trajectory obtained from CA-IPP algorithm converges to the optimal one as the number of samples goes to infinity, given some reasonable assumptions. We begin by stating following assumptions from [54, 53, 55].

Assumption 1. *For three different states x_1, x_2, and x_3, we assume that there exists trajectories which are $\mathcal{P}_a^b, \mathcal{P}_a^c$, and $\mathcal{P}_b^c$. If $x_2 \in \mathcal{P}_a^c$, then the concatenated trajectory $\mathcal{P}_a^b | \mathcal{P}_b^c$ must be equal to $\mathcal{P}_a^c$ and have same cost and information.*

This assumption states that the concatenated trajectory is consistent for intermediate points and it has consistent cost and information functions. This assumption is required since if infinite samples are added, samples will be infinitely close together.

Assumption 2. *There exists a ϵ-free feasible path π such that the minimal distance between π and the obstacle region is ϵ for any point $x \in \pi$, i.e., $\mathcal{B}_\epsilon(x) \in \mathcal{X}_{free}$.*

This assumption requires that some free space be available around any trajectory for convergence to the optimal trajectory. Final assumption is about the sample function.

Assumption 3. *The sample function returns an independent and identically distributed sample from $\mathcal{X}_{free}$, which are drawn from a uniform distribution.*

In order to show that CE-IPP can produce the optimal solution as the number of samples goes to infinity, we should obtain g_v^* at all $v \in \mathcal{V}_i^{ALG}$ as $i \to \infty$. It can be shown by the following lemma.

Theorem 4. *Let $\Omega_{v,\mathcal{B},i}$ denote the set of v-trajectories, where v is contained in the graph built by CE-IPP at iteration i for budget $\mathcal{B}$. We have that $\lim_{i\to\infty} \Omega_{v,\mathcal{B},i} = \Omega_{v,\mathcal{B}}$*

Proof. See Appendix D. □

6.4 Simulation and experimental results

6.4.1 Single robot informative path planning

This section is devoted to the effectiveness of the algorithm considered in the dissertation through the following simulations. The simulations were implemented in MATLAB and run on a computer with a 3.2GHz Intel i7 processor and 8 GB RAM. All simulations were performed five times for each algorithm using different pre-specified random seeds. A first set of simulations were run to illustrate the different performance of the proposed algorithm and RIG-graph algorithm presented in [55]. Since RIG-graph shows the asymptotic optimality, we examine how efficiently the algorithm finds the solution. First, we assume the following simple variance map which represents the uncertainty of the sensing field with a 10 km $\times$ 10 km environment. For fair comparison, the map is made similarly to the map used to apply RIG-graph to continuous space in [55] using Gaussian Process. We deployed 74 static sensors on equally spaced 11 by 11 grid points for the initial uncertainty map. By placing the static sensors, the initial map has a certain region which has high uncertainty than other regions as shown in Figure 6.4. High uncertainty is represented by red and low uncertainty is represented by blue. We assume that we can estimate the uncertainty map by measuring the static sensors at all times. The vehicle used in this dissertation can move 1 km distance in a continuous space based on the point-mass dynamics. Any other dynamics can be also applied to the algorithm. In order to capture the informativeness of the trajectory, we chose maximizing the reduction in variance of the field as the information function used in [50]. Since the selected information function is submodular, it requires discretized grid space as an input, so we convert the path found in continuous space to points in 11 by 11 grid space by

choosing the closest grid point. Furthermore, as [55] assume, we also assume that taking multiple measurements at the same location does not affect the collected information quantity. While [55] used the discrete environments to show the performance for finding the optimal solution, all simulations in this dissertation were run in the continuous environment with the 8 km budget constraint. Since there exists a fixed goal region, we limit the graph extension when the remaining budget of the node exceeds the shortest distance from the node to the goal region. Figure 6.4 shows a visual comparison of the RIG-graph and CE-IPP through sequential process of building a graph and obtaining the maximum informative trajectory from the graph. A white and black rectangle represent the start point and the goal region, respectively and cyan dots is the vertices in the graph. The maximum informative trajectory is represented by a black line. Since the proposed algorithm estimates the near-optimal path using CE at any node of the graph, more computation is required when the algorithm iterates for inserting a new node. However, CE-IPP returns an initial informative path more efficiently than RIG-graph since it can estimate the path using the remained budget at any node even if the graph does not reach the goal region. Moreover, the initially obtained path from CE-IPP is more informative than the results obtained from RIG-graph for much longer running time, thus convergence rate is much faster than RIG-graph. The results for five simulations are shown in Figure 6.5(a) for the maximum value of the informative path as a function of running time. For all trials, the proposed algorithm shows the superior performance.

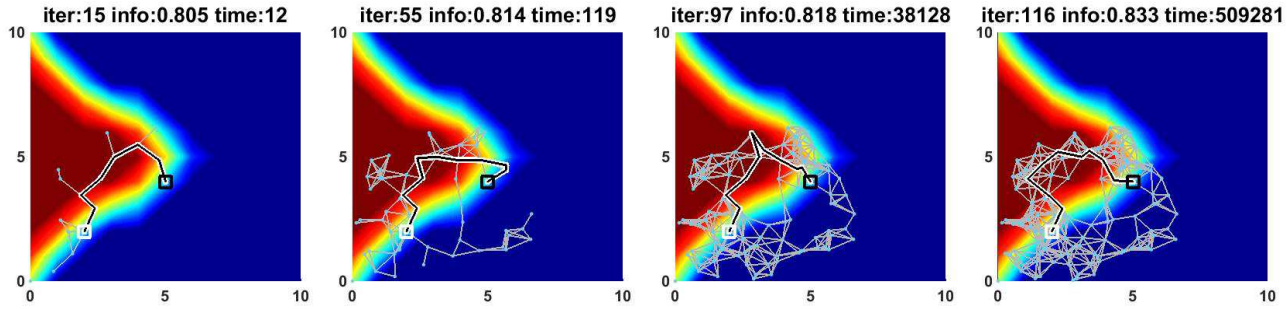
The performance is more distinguishable in a complex map. As shown in Figure 6.6, we placed 17 sensors randomly to make a complex map and set the budget to 20 to cover the whole region. The trajectory obtained from CE-IPP passes by covering high uncertainty regions over the whole search space. Figure 6.5(b)

shows five results of each algorithm. Again, the proposed algorithm finds the near-optimal solution efficiently compared to RIG-graph. As mentioned in the previous section, when RIG-graph computes the maximally reachable information to prune any co-located node which does not affect in terms of the optimality of the algorithm, a reachable region (i.e., a set of points which cover the region) is required. However, the reachable region is the same for many co-located nodes when a large initial budget is given, thereby causing pruning procedure meaningless. Furthermore, we scaled the grid size of the map to 21×21 with the same sensor placements and applied both algorithms. There is no big difference for CE-IPP even if the size of the map increases. On the other hand, RIG-graph requires more time for the bigger size map.

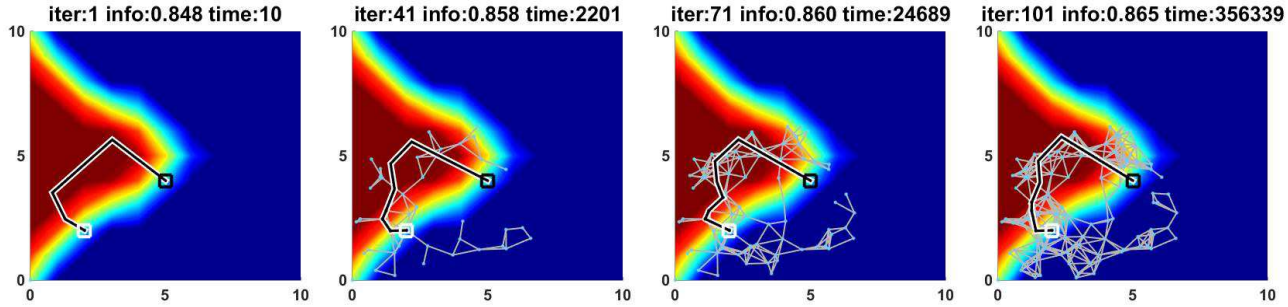
Hence, the proposed CE-IPP algorithm finds near optimal solution very quickly showing superior performance in computation efficiency and robustness for the size of the map compared to the RIG-graph algorithm.

6.4.2 Multi robot informative path planning

One approach for extending any single robot planning algorithm to plan simultaneous paths with multiple robots is to form a new graph where each node represents the vector of locations of all k robots, and then apply the single robot algorithm to this product graph. Unfortunately, the size of this product graph grows exponentially in k, which is infeasible for large teams of robots. However, in order to find the optimal path for each robot while performing a planning task simultaneously, such product graph is essential. One of the strength of the proposed algorithm is efficient in high dimensional space. Therefore, we applied the proposed algorithm in the product space. First we tested in the complex map explained in 6.4.1. In order to compare the results, we also applied the proposed

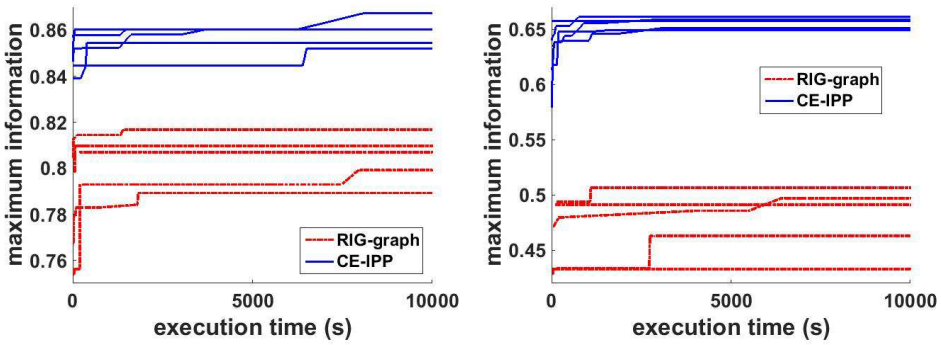


(a) RIG-Graph



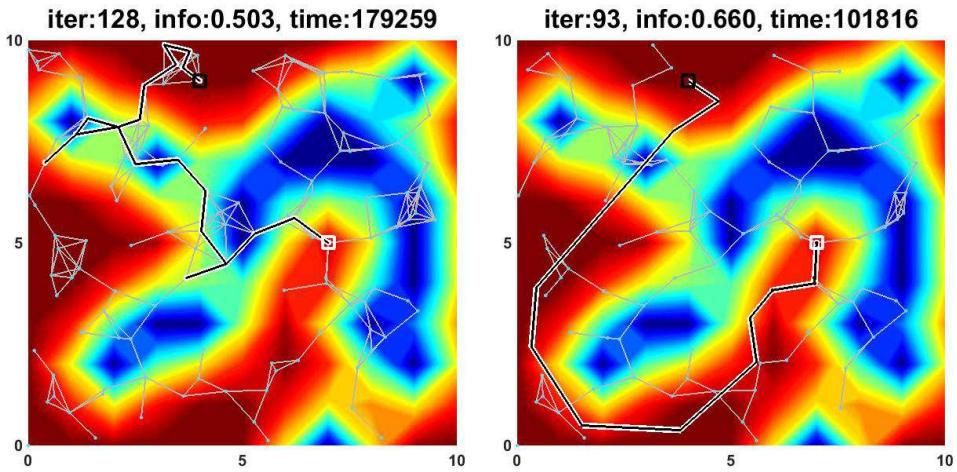
(b) CE-IPP

Figure 6.4: The sequential process of building a graph and obtaining the maximum informative trajectory from the graph. Each vertex is represented by cyan dot. The white and black rectangles represent a start point and goal region, respectively. The black line represents the maximum informative path at the current iteration.



(a) Simple map (b) Complex map

Figure 6.5: The maximally collected information quantity along the path as a function of running time in simple map(a) and complex map(b).



(a) RIG-Graph (b) CE-IPP

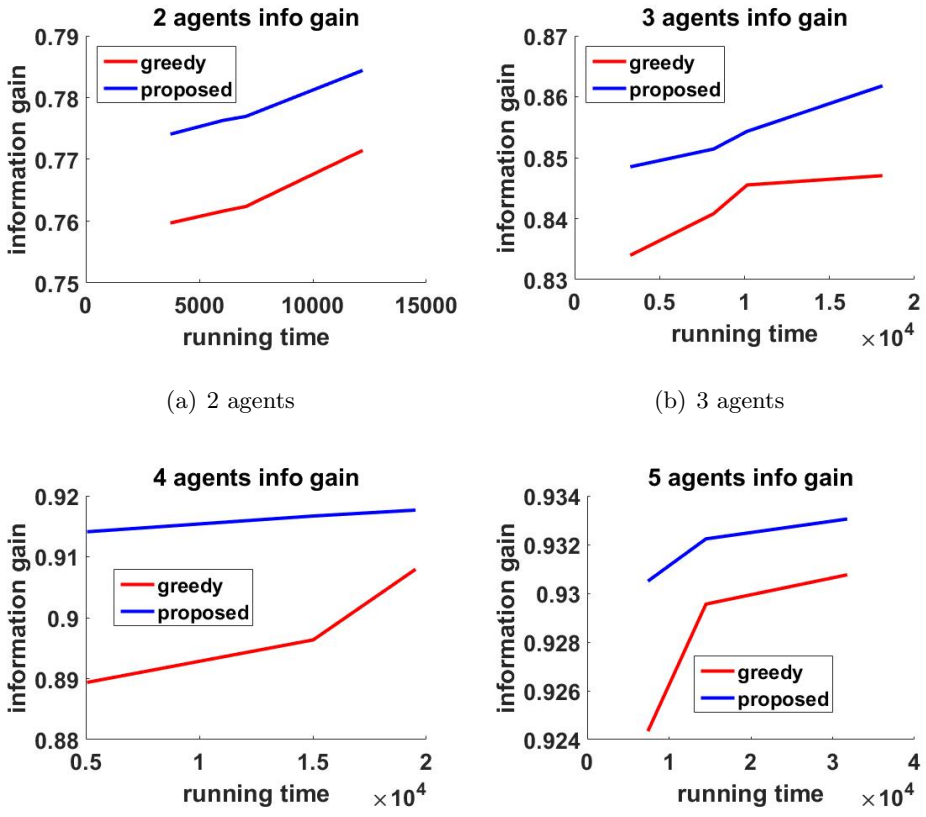
Figure 6.6: The results of two algorithms, RIG-graph (a) and CE-IPP (b), applied to the complex map generated using 17 static sensors.

algorithm in a greedy manner. In other words, after finding a path for a single robot within the time deadline, we found a path for another robot during the time deadline. We compared results for different running time using different number of robots. For fair comparison, we first set the total running time and distribute the time deadline to run single planning for each robot in greedy version. Figure 6.7 shows the obtained information from two approaches. For all cases, the results are superior to greedy version.

We also tested using another scenario which represents sea surface temperature field of Gulf of Mexico as shown in Figure 6.8. Red color represents high temperature and blue color represents low temperature. We placed no initial sensors and all robots have the same start point and goal region. Figure 6.9 and Figure 6.10 shows the trajectory results obtained from two different approaches, respectively. As shown in those figures, there exists the overlap region in greedy version. Since large overlap region means obtaining redundant information, the solutions are long way from the optimal solutions. However, the proposed multi robot planning approach finds much better solutions within the same running time.

6.5 Summary

We have presented a new approach which solves the informative path planning problem over continuous space for environmental monitoring. The proposed algorithm combines the sampling based planning method, RRG, and stochastic optimization method, CE. It guarantees asymptotic optimality, but, in addition, it can also return the near-optimal informative path at any node of the constructed spanning graph if there exists a path satisfying the constraint in terms of the cost function. We have shown that the proposed algorithm finds the optimal



(a) 2 agents

(b) 3 agents

(c) 4 agents

(d) 5 agents

Figure 6.7: The results of two approaches, greedy version and proposed method for different number of agents from 2 to 5 during the different running time.

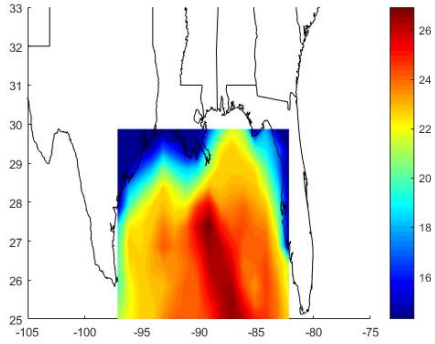
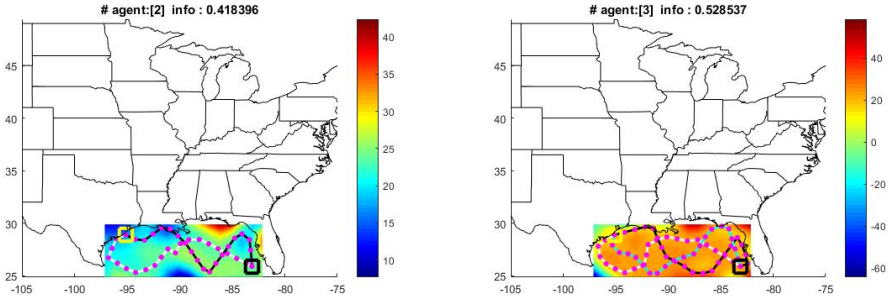


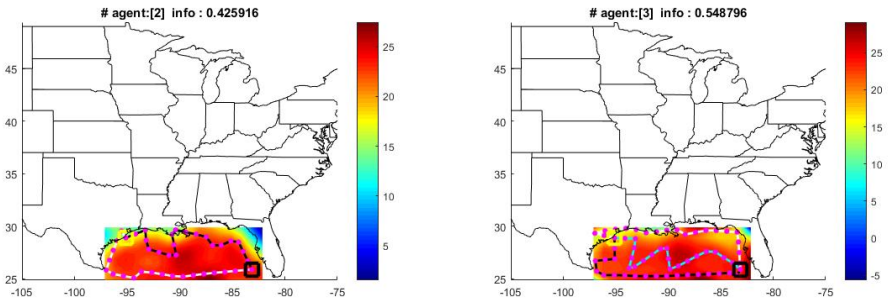
Figure 6.8: Sea surface temperature field of Gulf of Mexico. Red region represents high temperature and blue region represents low temperature.



(a) 2 agents

(b) 3 agents

Figure 6.9: Trajectory results using two and three agents in a greedy manner.



(a) 2 agents

(b) 3 agents

Figure 6.10: Trajectory results using two and three agents in a proposed multi-robot planning approach.

solution more efficiently by comparing against the state-of-the-art algorithm and show scalability of the proposed algorithm regardless of the size of grid resolution even when the search space is complex.

Chapter 7

Conclusion and Future Work

In this dissertation, we have solved two key fundamental problems in mobile sensor networks, which are localization and coordination problems of agents. For localization problem, we have presented a coordinated localization algorithm for mobile sensor networks. The global positioning system is able to provide synchronization and localization information, however in many situations, especially indoor environment, it cannot be relied on, and alternative methods are required. Therefore, we have designed the algorithm to solve the challenging localization problem under the GPS denied or unstructured indoor environment using an inexpensive off-the-shelf platform. By taking the advantage of a multi-agent system, we have shown that we can reliably localize robots over time as they perform a group task. In experiment, we have demonstrated that the proposed method consistently achieves a small localization error for long trajectories.

After developing the multi-robot localization system, we have presented a novel approach to handle the coordination of multiple agents for mobile sensor networks. Since robots perform tasks in a complex configuration space, where it has environmental parameters such as temperature, humidity, chemical concentra-

tion, stealthiness and elevation or has more than three dimensions, robots can be navigated efficiently through cost-aware planner for the given environment. Unlike the traditional methods, While sampling-based path planning algorithms, such as rapidly-exploring random tree (RRT) and its variants, have been highly effective for general path planning problems, it is still difficult or inefficient to find the minimum cost path in a complex space since RRT-based algorithms extend a search tree locally, requiring a large number of samples to find a good solution. The proposed algorithm uses global stochastic optimization method based on sampling based algorithm. By using global stochastic optimization method for tree extension, we have shown that the proposed method finds the minimum cost in the space with environmental parameters and a highly energy-efficient path for traversing a complex terrain with different elevation or performing a manipulation task in a high dimensional space. Furthermore, we have shown that the proposed algorithm has the probabilistic completeness property and asymptotic optimality when the number of samples goes to infinity.

We have also presented an efficient information gathering strategy suitable for mobile sensor networks. Resource constraints prevent us from using traditional coordination methods, because these typically require bulky, expensive, and sophisticated sensors, substantial memory and processor allocation, and a generous power supply. We describe the optimal planning method controlling mobile nodes in order to satisfy the resource constraints while collecting data.

For future works, more theoretical analysis about the convergence rate for the proposed algorithm and deep analysis for global stochastic optimization are required. Moreover, we will perform experiments using real robot to estimate the environmental field such as temperature of building and chemical concentration.

Appendices

Appendix A

Proof of Theorem 1

We first need the following lemma to prove Theorem 1 and Theorem 2.

Lemma 1. *Assume that there exists an input $u \in U = [u_{min}, u_{max}]$, which steers a robot from $x(t) = x \in \mathcal{X}_{free}$ to $x(t + \Delta t) = x' \in R_\epsilon(x)$ for some $\Delta t > 0$ and $\epsilon > 0$. Then, there exists $c > 0$, such that a randomly chosen input from U has the probability at least c of steering a robot from x to a point in $\mathcal{B}_\epsilon(x')$.*

Proof. If the state of the robot is at x at time t, the state at time $t + \Delta t$ is $x(t + \Delta t) = x(t) + \int_t^{t+\Delta t} f(x(t), u(t))dt$. Since

$$\dot{x} = f(x, u) = \frac{dx}{dt} \simeq \frac{x(t + \Delta t) - x(t)}{\Delta t}, \tag{A.1}$$

for small amount of time Δt, the solution can be written as

$$x(t + \Delta t) = x' = x + \Delta t f(x, u) + \xi_1, \tag{A.2}$$

where ξ_1 is the high order term. If a perturbed input, $u + \delta$, is applied to the system for small δ, then the state z at time $t + \Delta t$ is as follows:

$$z(t + \Delta t) = x + \Delta t f(x, u + \delta) + \xi_2. \tag{A.3}$$

Appendix A. Proof of Theorem 1

Provided that $|\xi_1 - \xi_2|$ is negligible for small Δt, we can find δ such that $\|x(t + \Delta t) - z(t + \Delta t)\| < \epsilon$ since f is a smooth function and Lipschitz continuous with respect to x and u in our operating domain. Hence, with probability at least $c = |\delta|/(u_{max} - u_{min}) > 0$, we can randomly select an input to steer the robot to $\mathcal{B}_\epsilon(x')$. □

We now prove Theorem 1. Suppose that there is an ϵ-free feasible path from the starting point x_{init} to the goal region x_{goal}. Then there exists a sequence of inputs $u_0, \ldots, u_k$, such that $x_0 = x_{init}, x_1, \ldots, x_{k+1}$ and $x_{k+1} \in x_{goal}$. We can then follow the proof of Theorem 3 in [29] to show the probabilistic completeness of RRT with inputs randomly selected from a bounded continuous set using Lemma 1.

Appendix B

Proof of Theorem 2

CARRT* attempts to perform a long extension towards x_{rand} if the distance between $x_{nearest}$ and x_{rand} is larger than η. Hence, if we can show that an ordinary RRT algorithm can construct a tree with a long extension constructed by CARRT*, Theorem 1 can be applied to show the probabilistic completeness of CARRT*. Suppose that there exists an ϵ-free feasible path $\pi(t)$ from $x_{nearest} = \pi(0)$ to $x_{rand} = \pi(T)$ towards x_{rand}. We first construct a set of balls of radius at least ϵ, $\mathcal{B} = \{\mathcal{B}_0, \ldots, \mathcal{B}_M\}$, covering π. The center of $\mathcal{B}_0$ is $x_{nearest}$ and the centers of two consecutive balls are exactly $\|\pi(t) - \pi(t + \Delta t)\|$ apart. Each ball $\mathcal{B}_m$ for $m \in \{1, \ldots, M\}$ has radius ϵ centered at each configuration $\pi(m\Delta t)$ for all $m \in \{1, 2, \ldots, M\}$, where $T = M\Delta t$. Without loss of generality, we assume that all vertices in the RRT tree, except $x_{nearest}$, are at least $(1 + i)\epsilon$ away from $\mathcal{B}_i$ for all i.

From Lemma 1, we know that $x_{nearest}$ can reach within $\mathcal{B}_1$ with a positive probability for a random input on some time interval $[0, \Delta t]$, Now, we extend the applied input time to $2\Delta t$. By the Lipschitz continuity of the system and the existence of an ϵ-free feasible path to the goal, there exists $\mathcal{R}_{2\epsilon}(x_{nearest})$ and

Appendix B. Proof of Theorem 2

$\mathcal{R}_{2\epsilon}(x_{nearest}) \cap \mathcal{B}_1$ is measurable. Hence, we can find a nonempty range of inputs to steer from a state in $\mathcal{R}_{2\epsilon}(x_{nearest}) \cap \mathcal{B}_1$ to a state in $\mathcal{B}_2$ on time interval $[\Delta t, 2\Delta t]$ using Lemma 1 again. Thus, we can sample inputs sequentially such that each new vertex added to the RRT tree falls in each ball $\mathcal{B}_m$ sequentially. Since M is finite, there is a positive probability that a long extension from $x_{nearest}$ to x_{rand} can be added by the RRT algorithm.

Appendix C

Proof of Theorem 3

The proof of the theorem is similar to that of Theorem 38 in [53] for holonomic dynamical system and Theorem 5 in [107] for nonholonomic dynamical system. We first review the asymptotic optimality condition of RRT* and show that CARRT* satisfies those conditions. In order to show the asymptotic optimality of RRT*, we begin with the construction of a sequence of paths $\{\pi_n\}$ of ϵ_n-free feasible paths from x_{init} to x_{goal} and show that π_n converges to the optimal path π^* as $n \to \infty$ by making a set of ball sequences $\{\mathcal{B}_n\}$, such that $\mathcal{B}_n$ covers π_n for all n [53]. Consider any sequence $\mathcal{B}_n = \{\mathcal{B}_{n,1}, \ldots, \mathcal{B}_{n,M}\}$ from $\{\mathcal{B}_n\}$. If each ball contains at least one vertex of RRT* with probability one, then a vertex x in $\mathcal{B}_{n,m}$ is connected to a vertex x' in the consecutive ball $\mathcal{B}_{n,m+1}$ for all m. This is possible due to the choice of γ_{RRT*} and the connection radius $r_n = \gamma_{RRT^*}(\frac{log(n)}{n})^{\frac{1}{d}}$. Let π'_n be a path obtained by RRT* after n iterations. It is shown that π'_n converges to π_n as n increases with probability one. Since π'_n converges to π_n and π_n converges to π^*, RRT* is asymptotically optimal. Hence, the critical condition for the asymptotic optimality is that vertices within radius r_n are connected in a cost-efficient manner.

Appendix C. Proof of Theorem 3

At the end of every iteration, CARRT* grows by generating the path $\mathcal{P}^*(t)$ from $x = \mathcal{P}^*(0)$ to $x' = \mathcal{P}^*(T)$ towards x_{rand} for any random point x_{rand} like RRT*. However, since CARRT* performs a long extension for $\|x - x_{rand}\| > \eta$, $\mathcal{P}^*(t)$ is obtained by applying sequentially sampled inputs with time interval Δt from x to x' unlike RRT*. In order to insert the whole path $\mathcal{P}^*(t)$ to the tree without losing information of $\mathcal{P}^*(t)$, we first discretize $\mathcal{P}^*(t)$ into a set of vertices $\{x_0, \ldots, x_K\}$ with time interval Δt, located at $\mathcal{P}^*(k\Delta t)$, where $x_0 = x, \ldots, x_K = x'$ and $T = K\Delta t$. Then, for a set of vertices, $\{x_1, \ldots, x_K\}$, except x_0, CARRT* regards each vertex as a new vertex added to the RRT tree and sequentially inserts x_k and edge to x_k to the tree. For any vertex x_k inserted to tree, CARRT* performs the rewiring procedure by attempting to create an edge from the vertex x_k to its neighbors within radius $r_n = \gamma_{RRT^*}(\frac{log(n)}{n})^{\frac{1}{d}}$. Note that since CARRT* adds several vertices to the tree in a single iteration unlike RRT*, n denotes the number of vertices in the tree. On this account, we can show the existence of the connection between balls with radius r_n. Using this fact and Lemma 71 in [53], we know that each ball in $\mathcal{B}_n$ contains vertices and every subsequent balls in $\mathcal{B}_n$ are connected via vertices with probability one as n increases. It follows that $\mathbb{P}(\lim_{n\to\infty} \pi'_n = \pi^*) = 1$ due to Lemma 72 in [53]. Hence, CARRT* returns an optimal solution asymptotically as desired.

Appendix D

Proof of Theorem 4

(Sketch) The proof of this lemma follows directly from Lemma 4.4 by Hollinger and Sukhatme [55]. They stated that all feasible trajectories for budget $\mathcal{B}$ can be generated if the algorithm builds an infinitely dense connected graph and the length of all feasible trajectories obtained from the algorithm is finite. If the number of samples goes to infinity within a ball around any $x \in \mathcal{X}_{free}$, the graph within the ball becomes the dense connected graph and this procedure can be extended to all x, thereby giving an infinitely dense connected graph within $\mathcal{X}_{free}$. Since the proposed algorithm follows the RRG structure like algorithms proposed in [53, 54, 55], it can also build such a graph. Furthermore, as explained in Section 6.1, any path included in $\Omega_{v,\mathcal{B},i}$ has finite length. Since $\Omega_{v,\mathcal{B}}$ is included in the set of all feasible trajectories for budget $\mathcal{B}$, the graph built by the proposed algorithm contains all trajectories via v.

Appendix D. Proof of Theorem 4

Appendix E

Dubins' curve

Dubins' curves represent the shortest path between any two configurations and each path in Dubins' curves can be generated using three primitive motions which are left (L), right(R) turn, and straight (S) motion. Such path has the minimum turning radius ρ (i.e.,$\rho = 1$) and moves at constant velocity [92]. Dubins' curves consist of possibly optimal six paths which are

$$\{LSL, RSR, RSL, LSR, RLR, LRL\}. \tag{E.1}$$

To be more precise, the duration of each primitive should be specified. Let any subscripts (t, p, q) denote total amount of rotation that accumulates during the application of the primitive for turning motion or the total distance traveled for straight motion. Using such subscripts, the Dubins' curves can be more precisely presented as

$$\{L_q S_p L_t, R_q S_p R_t, R_q S_p L_t, L_q S_p R_t, R_q L_p R_t, L_q R_p L_t\} \tag{E.2}$$

We derive the equations for the length of each path based on [108] given the initial and final configuration in this dissertation. We first define three motion

operators for any configuration as follows:

$$L_v(x,y,\theta) = (x + S(\theta+v) - S(\theta), y - C(\theta+v) + C(\theta), \theta+v) \quad \text{(E.3)}$$

$$R_v(x,y,\theta) = (x - S(\theta-v) + S(\theta), y + C(\theta-v) - C(\theta), \theta-v) \quad \text{(E.4)}$$

$$S_v(x,y,\theta) = (x + vC(\theta), y + vS(\theta), \theta), \quad \text{(E.5)}$$

where $L_v(x,y,\theta)$ is the result configuration when applying left turn motion operator at (x,y,θ) and v represents the moving distance along the motion segment.

Suppose that a path starts at $(0,0,\alpha)$ and ends at $(d,0,\beta)$. Then the length of each path is formulated as follows:

$$(1)\ L_q S_p L_t(0,0,\alpha) = (d,0,\beta). \quad \text{(E.6)}$$

Through the motion operators (E.3), this path can be represented by three scalar equations as follows:

$$p\cos(\alpha+t) - \sin(\alpha) + \sin(\beta) = d \quad \text{(E.7)}$$

$$p\sin(\alpha+t) + \cos(\alpha) - \cos(\beta) = 0 \quad \text{(E.8)}$$

$$\alpha + t + q = \beta\{\text{mod}2\pi\}. \quad \text{(E.9)}$$

The solution for t,p,q is found as

$$t = -\alpha + \arctan(\frac{dy - C(\alpha) + C(\beta)}{dx + S(\alpha) - S(\beta)}) \quad \text{(E.10)}$$

$$p = \sqrt{2 + A - 2C(\alpha-\beta) + 2dx(S(\alpha) - S(\beta)) - 2dy(C(\alpha) - C(\beta))}$$

$$q = \beta - \alpha - t$$

$$(2)\ R_q(S_p(R_t(0,0,\alpha))) = (d,0,\beta). \quad \text{(E.11)}$$

Three scalar equations:

$$p\cos(\alpha - t) + \sin(\alpha) - \sin(\beta) = d \tag{E.12}$$

$$p\sin(\alpha - t) - \cos(\alpha) + \cos(\beta) = 0 \tag{E.13}$$

$$\alpha - t - q = \beta\{\mathrm{mod}2\pi\}. \tag{E.14}$$

The solution is

$$\begin{aligned} t &= \alpha - \arctan(\frac{dy + C(\alpha) - C(\beta)}{dx - S(\alpha) + S(\beta)}) \\ p &= \sqrt{2 + A - 2C(\alpha - \beta) + 2dx(-S(\alpha) + S(\beta)) + 2dy(C(\alpha) - C(\beta))} \\ q &= \alpha - t - \beta \end{aligned} \tag{E.15}$$

$$(3)\ R_q(S_p(L_t(0,0,\alpha))) = (d,0,\beta). \tag{E.16}$$

Three scalar equations are

$$p\cos(\alpha + t) + 2\sin(\alpha + t) - \sin(\alpha) - \sin(\beta) = d \tag{E.17}$$

$$p\sin(\alpha + t) - 2\cos(\alpha + t) + \cos(\alpha) + \cos(\beta) = 0 \tag{E.18}$$

$$\alpha + t - q = \beta\{\mathrm{mod}2\pi\}. \tag{E.19}$$

The solution of this system is

$$\begin{aligned} t &= -\alpha + \arcsin(\frac{-p(-dy + C(\alpha) + C(\beta)) + 2(dx + S(\alpha) + S(\beta))}{p^2 + 4}) \\ p &= \sqrt{-2 + A + 2C(\alpha - \beta) + 2dx(S(\alpha) + S(\beta)) - 2dy(C(\alpha) + C(\beta))} \\ q &= \alpha + t - \beta \end{aligned} \tag{E.20}$$

$$(4)\ L_q(S_p(R_t(0,0,\alpha))) = (d,0,\beta). \tag{E.21}$$

The corresponding scalar equations are

$$p\cos(\alpha - t) - 2\sin(\alpha - t) + \sin(\alpha) + \sin(\beta) = d \tag{E.22}$$

$$p\sin(\alpha - t) + 2\cos(\alpha - t) - \cos(\alpha) - \cos(\beta) = 0 \tag{E.23}$$

$$\alpha - t + q = \beta\{\mathrm{mod}2\pi\}. \tag{E.24}$$

The corresponding solution is

$$\begin{aligned} t &= \alpha - \arcsin\left(\frac{p(dy + C(\alpha) + C(\beta)) + 2(-dx + S(\alpha) + S(\beta))}{p^2 + 4}\right) \\ p &= \sqrt{-2 + A + 2C(\alpha - \beta) - 2dx(S(\alpha) + S(\beta)) + 2dy(C(\alpha) + C(\beta))} \\ q &= \alpha + t - \beta \end{aligned} \tag{E.25}$$

$$(5)\ R_q(L_p(R_t(0,0,\alpha))) = (d,0,\beta). \tag{E.26}$$

Scalar equations:

$$2\sin(\alpha - t + p) - 2\sin(\alpha - t) + \sin(\alpha) - \sin(\beta) = d \tag{E.27}$$

$$-2\cos(\alpha - t + p) + 2\cos(\alpha - t) - \cos(\alpha) + \cos(\beta) = 0 \tag{E.28}$$

$$\alpha - t + p - q = \beta\{\mathrm{mod}2\pi\}. \tag{E.29}$$

The solution of this system is

$$\begin{aligned} t &= \alpha + \frac{p}{2} - \arctan\left(\frac{dy + C(\alpha) - C(\beta)}{dx - S(\alpha) + S(\beta)}\right) \\ p &= \arccos\left(\frac{1}{8}(6 - A + 2C(\alpha - \beta) + 2dx(S(\alpha) - S(\beta))) - 2dy(C(\alpha) - C(\beta))\right)\right) \\ q &= \alpha - \beta - t + p \end{aligned} \tag{E.30}$$

$$(6)\ L_q(R_p(L_t(0,0,\alpha))) = (d,0,\beta). \tag{E.31}$$

The corresponding scalar equations are

$$-2\sin(\alpha + t - p) + 2\sin(\alpha + t) - \sin(\alpha) + \sin(\beta) = d \tag{E.32}$$

$$2\cos(\alpha + t - p) - 2\cos(\alpha + t) + \cos(\alpha) - \cos(\beta) = 0 \tag{E.33}$$

$$\alpha + t - p + q = \beta\{\mathrm{mod}2\pi\}. \tag{E.34}$$

The corresponding solution is

$$\begin{aligned} t &= -\alpha + \frac{p}{2} + \arctan\left(\frac{dy - C(\alpha) + C(\beta)}{dx + S(\alpha) - S(\beta)}\right) \\ p &= \arccos\left(\frac{1}{8}(6 - A + 2C(\alpha - \beta) - 2dx(S(\alpha) - S(\beta))) + 2dy(C(\alpha) - C(\beta))\right)\right) \\ q &= -\alpha + \beta - t + p, \end{aligned} \tag{E.35}$$

where A represents $dx^2 + dy^2$ and $C(\cdot)$ and $S(\cdot)$ represent $\cos(\cdot)$ and $\sin(\cdot)$, respectively.

Appendix E. Dubins' curve

Bibliography

[1] S. Oh, L. Schenato, P. Chen, and S. Sastry, "Tracking and coordination of multiple agents using sensor networks: System design, algorithms and experiments," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 234–254, January 2007.

[2] A. Singh, M. Batalin, M. Stealey, V. Chen, Y. Lam, M. Hansen, T. Harmon, G. S. Sukhatme, and W. Kaiser, "Mobile robot sensing for environmental applications," in *Proceedings of the International Conference on Field and Service Robotics*, 2008.

[3] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[4] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stability of flocking motion," University of Pennsylvania, Technical Report, 2003.

[5] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[6] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[7] M. T. Pham and K. T. Seow, "Discrete-event coordination design for distributed agents," *IEEE Transanctions on Automatic Science and Engineering*, vol. 9, no. 1, pp. 70–82, January 2012.

[8] J. Suh, S. You, S. Choi, and S. Oh, "Vision-based coordinated localization for mobile sensor networks," *IEEE Transactions on Automation Science and Engineering (online)*, 2014.

[9] J. Suh, J. Gong, and S. Oh, "Energy-efficient high-dimensional motion planning for humanoids using stochastic optimization," in *Proceedings of the IEEE/RSJ International Conference on Humanoid Robots*, Seoul, Korea, November 2015.

[10] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler, "The effects of ranging noise on multihop localization: an empirical study," in *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks*, April 2005.

[11] M. Maróti, B. Kusý, G. Balogh, P. Völgyesi, A. Nádas, K. Molnár, S. Dóra, and A. Lédeczi, "Radio interferometric geolocation," in *Proceedings of the ACM SenSys*, Nov. 2005.

[12] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, Eds., *Globnal Positioning System: Theory and Practice*. Springer Verlag, 1997.

[13] A. Ledeczi, P. Volgyesi, J. Sallai, B. Kusy, X. Koutsoukos, and M. Maroti, "Towards precise indoor rf localization," in *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensors (HOTEMNETS)*, June 2008.

[14] J. Cho, Y. Ding, and J. Tang, "Robust calibration for localization in clustered wireless sensor networks," *IEEE Transactions on Automatic Science and Engineering*, vol. 7, no. 1, pp. 81–95, January 2010.

[15] S. Gecizi, Z. Tian, G. B. Giannakis, Z. Sahinoglu, H. Kobayashi, A. F. Molisch, and H. V. Poor, "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 70–84, 2005.

[16] A. Prorok and A. Martinoli, "Accurate indoor localization with ultra-wideband using spatial models and collaboration," *The international Journal of Robotics Research*, vol. 33, no. 4, pp. 547–568, 2014.

[17] C. Taylor and B. Shirmohammadi, "Self localizing smart camera networks and their applications to 3d modeling," in *Proceedings of the International Workshop on Distributed Smart Cameras*, 2006.

[18] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks*, April 2006.

[19] O. Tekdas and V. Isler, "Sensor placement for triagulation-based localization," *IEEE Transactions on Automatic Science and Engineering*, vol. 7, no. 3, pp. 681–685, July 2010.

[20] M. Meingast, S. Oh, and S. Sastry, "Automatic camera network localization using object image tracks," in *Proceedings of the IEEE International Conference on Computer Vision*, Octover 2007.

[21] S. Oh, S. Russell, and S. Sastry, "Markov chain Monte Carlo data association for multi-target tracking," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 481–497, March 2009.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] F. Zhang, B. Grocholsky, R. Kumar, and M. Mintz, "Cooperative control for localization of mobile sensor networks," in *GRASP Laboratory, University of Pennsylvania, internal paper*, 2003.

[24] A. Prorok and A. Martinoli, "A reciprocal sampling algorithm for lightweight distributed multi-robot localization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System*, 2011, pp. 3241–3247.

[25] P. Giguere, I. Rekleitis, and M. Latulippe, "I see you, you see me: Cooperative localization through bearing-only mutually observing robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System*, Octover 2012.

[26] R. Sharma, S. Quebe, R. W. Beard, and C. N. Taylor, "Bearing-only cooperative localization," *Journal of Intelligent and Robotic Systems*, vol. 72, no. 3-4, pp. 429–440, 2013.

[27] A. Ahmad, G. D. Tipaldi, P. Lima, and W. Burgard, "Cooperative robot localization and target tracking based on least squares minimization," in

Proceedings of the IEEE International Conference on Robotics and Automation, May 2013.

[28] S. Tully, G. Kantor, and H. Choset, "Leap-frog path design for multi-robot cooperative localization," in *Proceeding of the International Conference on Field and Service Robots, Cambridge, MA, USA*, 2009.

[29] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 3, pp. 378–400, May 2001.

[30] N. A. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Roma, April 2007.

[31] J. V. der Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," in *Proceedings of the Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[32] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and Gaussian processes," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, Sep. 2008.

[33] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using RRT," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, Nice, France, Sep. 2008.

[34] L. E. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, June 1994.

[35] A. Rodríguez, A. Pérez, J. Rosell, and L. B. nez, "Sampling-based path planning for geometrically-constrained objects," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[36] R. Y. Rubinstein, "Optimization of computer simulation models with rare events," *European Journal of Operational Research*, vol. 99, no. 1, pp. 89–112, 1997.

[37] M. Kobilarov, "Cross-entropy randomized motion planning," in *Proceedings of the Robotics: Science and Systems*, Los Angeles, USA, June 2011.

[38] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, Las Vegas, USA, Oct. 2003.

[39] D. Ferguson and A. Stentz, "Anytime rrts," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, Beijing, China, Oct. 2006.

[40] A. Ettlin and H. Bleuler, "Rough-terrain robot motion planning based on obstacleness," in *Proceedings of the International Conference on Control, Automation, Robotics and Vision*, Singapore, Dec. 2006.

[41] J. Lee, C. Pippin, and T. Balch, "Cost based planning with RRT in outdoor environment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, Nice, France, Sep. 2008.

[42] Y. Xu, J. Choi, and S. Oh, "Mobile sensor network navigation using Gaussian processes with truncated observations," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 1118–1131, Dec. 2011.

[43] L. Jaillét, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, Aug. 2010.

[44] N. E. Leonard, D. A. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. E. Davis, "Collective motion, sensor networks, and ocean sampling," *Proceedings on the IEEE*, vol. 95, 2007.

[45] N. Cao, K. H. Low, and J. M. Dolan, "Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms," in *Proceedings of the IEEE International Conference on Autonomous agents and multi-agent systems*, 2013.

[46] R. Marchant and F. Ramos, "Bayesian optimisation for informative continuous path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Hong Kong, China, May 2014.

[47] A. Singh, A. Krause, and W. J. Kaiser, "Nonmyopic adaptive informative path planning for multiple robots," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2009.

[48] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," in *Proceedings of the International Symposium on Robotics Research*, 2011.

[49] C. Guestrin, A. Krause, and A. P. Singh, "Near-optimal sensor placements in gaussian processes," in *Proceedings of the International Conference on Machine learning*, 2005.

[50] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Saint Paul, USA, May 2012.

[51] J. Kwak and P. Scerri, "Path planning for autonomous information collecting vehicles," in *Proceedings of the International Conference on Information Fusion*, 2008.

[52] D. Levine, B. Luders, and J. P. How, "Information-rich path planning with general constraints using rapidly-exploring random trees," in *Proceedings of AIAA Infotech@Aerospace Conference*, Atlanta, GA, April 2010.

[53] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[54] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, June 2011.

[55] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The international Jounal of Robotics Research*, vol. 32, no. 12, pp. 1271–1287, 2014.

[56] "iRobot." [Online]. Available: http://www.irobot.com/

[57] Y. Ma, J. Kosecka, S. Soatto, and S. Sastry, Eds., *An Invitation to 3-D Vision*. New York: Springer Verlag, 2003.

[58] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *Proceedings of 13th British Machine Vision Conference*, 2002, pp. 414–431.

[59] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[60] F. Chaumette and S. Hutchinson, Eds., *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008.

[61] J. Suh, S. You, and S. Oh, "A cooperative localization algorithm for mobile sensor networks," in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, August 2012.

[62] G. Dudek and M. Jenkin, Eds., *Computational Principles of Mobile Robotics*. New York, NY, USA: Cambridge University Press, 2000.

[63] A. Koubâa, M. Alves, and E. Tovar, "GTS allocation analysis in IEEE 802.15.4 for real-time wireless sensor networks," in *Proceedings of the IEEE International Conference on Parallel and Distributed Processing*, 2006.

[64] J. Suh and S. Oh, "A cost-aware path planning algorithm for mobile robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, Villamoura, Portugal, Oct. 2012.

[65] P. T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, 2005.

[66] P. Plonski, P. Tokekar, and V. Isler, "Energy-efficient path planning for solar-powered mobile robots," *Journal of Field Robotics*, 2013.

[67] L. Murphy and P. Newman, "Risky planning on probabilistic costmaps for path planning in outdoor environments," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 445–457, April 2013.

[68] Y. Mei, Y. Lu, Y. Hu, and C. Lee, "Energy-efficient motion planning for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, New Orleans, April 2004.

[69] S. Liu and D. Sun, "Optimal motion planning of a mobile robot with minimum energy consumption," in *Proceedings of the IEEE International Conference on Advanced Intelligent Mechatronics*, July 2011.

[70] G. Wang, M. J. Irwin, H. Fu, P. Berman, W. Zhang, and T. L. Porta, "Optimizing sensor movement planning for energy efficiency," *ACM Transactions on Sensor Networks*, vol. 7, no. 4, pp. 846–894, Feb. 2011.

[71] Z. Sun and J. H. Reif, "On finding energy-minimizing paths on terrains," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 102–114, Feb. 2005.

[72] X. Li, Y. Chen, and J. Wang, "In-wheel motor electric ground vehicle energy management strategy for maximizing the travel distance," in *Proceedings of American Control Conference (ACC)*, Montréal, Canada, 2012.

[73] J. Kwak, M. Pivtoraiko, and R. Simmons, "Combining cost and reliability for rough terrain navigation," in *Proceedings of the International Symposium on Artificial Intelligence Robotics and Automation in Space*, 2008.

[74] L. D. Michieli, F. Nori, A. A. Pini Prato, and G. Sandini, "Study on humanoid robot systems: An energy approach," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2008.

[75] J. A. Kulk and J. S. Welsh, "A low power walk for the nao robot," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2008.

[76] S. H. Collins, M. Wisse, and A. Ruina, "A three-dimensional passive-dynamic walking robot with two legs and knees," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 607–615, 2001.

[77] S. h. Collins and A. Ruina, "A bipedal walking robot with efficient and human-like gait," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[78] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011.

[79] M. Kobilarov, "cross entropy motion planning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, June 2012.

[80] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," CSAIL, MIT, Tech. Rep. MIT-CSAIL-TR-2013-021, Aug. 2013.

[81] J. D. Gammelland, S. S. Srinivasaand, and T. D. Barfoot, "Informed rrt*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic," *arXiv preprint arXiv:1404.2334*, 2014.

[82] D. Kim, j. Lee, and S. Yoon, "Cloud rrt*: Sampling cloud based rrt*," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014.

[83] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013.

[84] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, USA, Sept. 2011.

[85] A. Perez, S. Karaman, A. Shkolnik, E. Frazzoli, S. Teller, and M. R. Walter, "Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, USA, Sept. 2011.

[86] J. H. Reif, "Complexity of the mover's problem and generalizations extended abstract," in *Proceedings of the IEEE Conference on Foundations of Computer Science*, 1979.

[87] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, "On the complexity of motion planning for multiple independent objects; pspace-hardness of the "warehouseman's problem"," *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984.

[88] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[89] R. Y. Rubinstein, "The cross-entropy method for combinatorial and continuous optimzation," *Methodology and computing in applied probability*, vol. 1, no. 2, pp. 127–190, 1999.

[90] C. E. Rasmussen and C. K. Williams, Eds., *Gaussian Processes for Machine Learning*. Cambridge: The MIT Press, 2006.

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

[92] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[93] P. Švestka, "On probabilistic completeness and expected complexity of probabilistic path planning," *Technical Report UU-CS-96-20*, 1996.

[94] O. Arslan and P. Tsiotras, "Use of relaxation methods in samping-based algorithms for optimal motion planning," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2013.

[95] N. Cressie, "Kriging nonstationary data," *Journal of the American Statistical Association*, vol. 81, pp. 625–634, Nov. 1986.

[96] M. N. Gibbs and D. J. C. MacKay, "Efficient implementation of gaussian processes," Cambridge, Tech. Rep., 1997.

[97] B. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007.

[98] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, pp. 235–284, Feb. 2008.

[99] H. Durrant-Whyte, F. Ramos, E. Nettleton, A. Blair, and S. Vasudevan, "Gaussian process modeling of large scale terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.

[100] "Gazebo." [Online]. Available: http://gazebosim.org/

[101] "ROS-Introduction." [Online]. Available: http://wiki.ros.org/ROS/Introduction/

[102] "Nao Software Documentation 1.14." [Online]. Available: http://community.aldebaran-robotics.com/resources/documentation/

[103] J. Craig, Ed., *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, 2005.

[104] A. Krause, B. McMahan, A. Gupta, and J. Kleinberg, "Near-optimal sensor placements:maximizing information while minimizing communication cost," in *Proceedings of the International Symposium on Information Processing in Sensor Networks*, 2006.

[105] A. Krause and C. Guestrin, "Nonmyopic active learning of gaussian processes:an exploration-exploitation approach," in *Proceedings of the International Conference of Machine Learning*, 2007.

[106] S. Mannor, R. Y. Rubinstein, and Y. Gat, "The cross entropy method for fast policy search," in *Proceedings of the International Conference on Machine learning*, 2003.

[107] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proceedings of the IEEE Conference on Decision and Control*, Atlanta, GA, December 2010.

[108] A. M. Shkel and V. Lumelsky, "Classification of the dubins set," *Robotics and Autonomous Systems*, vol. 34, no. 4, pp. 179–202, 2001.

초 록

주위 환경변화에 대한 적응성이 약하다는 문제점이 있는 기존의 무선 센서네트워크와는 달리 무선 센서 네트워크에 로보틱스 기술을 적용한 모바일 센서네트워크는 이러한 문제점을 극복할 수 있다. 하지만 모바일 센서네트워크에서도 해결해야 할 문제들이 존재한다. 그 중 가장 먼저 해결해야하는 문제들이 바로 센서들의 위치정보를 알아내고 센서 배치를 위해 배치 장소까지 경로를 탐색하고 이동하면서 주위 환경에 대한 정보를 얻는 것이다. 본 논문에서는 모바일 센서네트워크의 이동성을 이용하여 위치 측정과 경로 탐색 알고리즘을 개발하고 제안한 경로 탐색 방법의 이점을 이용하여 에너지의 제한이 있는 모바일 센서로부터 주위 환경의 정보를 얻을 수 있는 coverage 제어 방법을 제안한다.

본 논문에서는 GPS가 동작하지 않는 환경, 특히 실내에서 저가의 로봇 플랫폼을 이용하여 모바일 센서 네트워크에 적합한 위치 측정 알고리즘을 개발한다. 먼저 로봇들을 2개의 그룹, 정지 로봇과 이동 로봇으로 나누고 이동 로봇들은 정지 로봇들의 시야 내에서 움직인다. 정지 로봇들은 카메라 센서를 통해 이동 로봇들을 추적하고 이동 로봇들의 이동 경로가 시공간적 특징점으로 사용된다. 이러한 시공간적 특징점으로부터 multi-view geometry를 이용하여 정지 로봇들의 상대적 위치들을 계산하고 전체 기준 좌표계에 따라 모든 로봇들의 위치를 계산한다. 모든 로봇들의 위치를 계산한 후에 정지 로봇들은 formation을 유지하면서 네비게이션 전략에 따라 이동하고 동시에 위치측정을 하면서 맡은 임무를 수행한다. 제안한 방법이 저비용의 노이즈가 심한 카메라 센서를 이용하여 지속적으로 강인한 위치 정보를 제공한다는 것을 실험을 통해 보여줌으로써 비용이 저렴한 로봇 플랫폼을 요구하는 모바일 센서네트워크에 적합하다는 것을 보여준다.

본 논문은 또한 특정위치로 모바일 센서들을 배치하기 위한 경로 탐색 알고리즘을 제안한다. 기존의 경로 탐색방법과 달리 주어진 환경을 고려하여 모바일 센서네트워크에 적합한 효율적인 비용인지 경로 탐색 방법을 개발한다. 본 논문은 샘플링 기반의 경로 탐색 방법의 효율성을 높이기 위해 전역적 확률 최적화 방법인 Cross

Entropy방법을 이용하여 트리를 확장하는 첫 번째 방법이다. 제안한 방법은 무작위로 선택된 노드 방향으로 트리를 확장하기 위한 트리내의 노드를 결정하기 위해서 사용되는 일반적인 RRT 트리인 첫 번째 트리와 트리를 확장할 때 전역적 확장을 수용하여 확장하는 두 번째 트리를 구성함으로써 알고리즘의 효율성을 개선하고 점근적 최적성을 보장한다.

모바일 센서네트워크에 적합하도록 에너지가 한정된 모바일 센서들을 이용하여 제안한 경로 탐색 방법을 기반으로 하는 coverage 제어 시스템을 개발한다. 일반적인 경로 탐색 방법과 달리 정보를 얻는 경로 탐색 방법은 제약이 있는 극대화 문제를 풀어야하기 때문에 훨씬 어렵고 discrete환경에서도 NP-hard 문제이다. 본 논문에서는 CE방법을 이용하여 효율적인 방법을 제안한다. 제안한 알고리즘은 최신 알고리즘보다 더 효율적이며 점근적 최적성을 보장한다. 또한 기존의 방법들과 달리 여러 로봇에 적용가능하기 때문에 모바일 센서네트워크에 적합하다고 볼 수 있다.

결론적으로, 제안한 방법들은 모바일 센서네트워크에서 가장 먼저 해결해야 하는 문제들인 센서들의 위치 측정, 경로 탐색, coverage 제어 문제들을 해결한다. 제안한 방법들은 기존의 방법들과 달리 가격이 저렴한 로봇 플랫폼을 이용하여 저렴한 센서에 강인한 위치 측정문제를 풀고 주위 환경을 고려하여 효율적으로 경로 탐색을 하며 여러 로봇에 적용할 수 있는 효율적인 coverage 제어 방법을 제안하였기 때문에 모바일 센서네트워크에 적합하다.

주요어: 실내 위치 측정 시스템, 비용인지 경로 탐색, 멀티 로봇을 이용한 coverage 제어

학 번: 2011-30962