Learning Probabilistic Generative Models For Fast Sampling-Based Planning

Jinwook Huh
University of Pennsylvania, jinwookhuh@gmail.com

Follow this and additional works at: https://repository.upenn.edu/edissertations

Part of the Artificial Intelligence and Robotics Commons, and the Robotics Commons

Recommended Citation
https://repository.upenn.edu/edissertations/3480

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/edissertations/3480
For more information, please contact repository@pobox.upenn.edu.
Learning Probabilistic Generative Models For Fast Sampling-Based Planning

Abstract
Due to their simplicity and efficiency in high dimensional space, sampling-based motion planners have been gaining interest for robotic manipulation in recent years. We present several new learning approaches using probabilistic generative models for fast sampling-based planning. First, we propose fast collision detection in high dimensional configuration spaces based on Gaussian Mixture Models (GMMs) for Rapidly-exploring Random Trees (RRT). In addition, we introduce a new probabilistically safe local steering primitive based on the probabilistic model. Our local steering procedure is based on a new notion of a convex probabilistically safety corridor that is constructed around a configuration using tangent hyperplanes of confidence ellipsoids of GMMs learned from prior collision history. For efficient sampling, we suggest a sampling method with a learned Q-function with linear function approximation based on feature representations such as Radial Basis Functions. This sampling method chooses the optimal node from which to extend the search tree via the softmax function of learned state values. We also discuss a novel constrained sampling-based motion planning method for grasp and transport tasks with redundant robotic manipulators, which allows the best grasp configuration and approach direction to be automatically determined. Since these approaches with the learned probabilistic models require large size data and time for training, it is essential that they are able to be adapted to environmental change in an online manner. The suggested online learning approach with the Dirichlet Process Mixture Model (DPMM) can adapt the complexity to the data and learn new Gaussian clusters with streaming data in newly explored areas without batch learning. We have applied these approaches in a number of robot arm planning scenarios and have shown their utility and effectiveness in simulation and on a physical 7-DoF robot manipulator.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Electrical & Systems Engineering

First Advisor
Daniel D. Lee

Second Advisor
Manfred Morari

Keywords
Gaussian mixture model, Machine learning, Motion and path planning, Probabilistic generative model, Robotics, Sampling-based planning

Subject Categories
Artificial Intelligence and Robotics | Computer Sciences | Robotics

This dissertation is available at ScholarlyCommons: https://repository.upenn.edu/edissertations/3480
LEARNING PROBABILISTIC GENERATIVE MODELS
FOR FAST SAMPLING-BASED PLANNING

Jinwook Huh

A DISSERTATION
in
Electrical and Systems Engineering
Presented to the Faculties of the University of Pennsylvania
in
Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
2019

Supervisor of Dissertation

Daniel D. Lee, Professor of Electrical and Systems Engineering

Graduate Group Chairperson

Victor M. Preciado, Associate Professor of Electrical and Systems Engineering

Dissertation Committee
Manfred Morari, Professor of Electrical and Systems Engineering
University of Pennsylvania
Frank Chongwoo Park, Professor of Mechanical and Aerospace Engineering
Seoul National University
Maxim Likhachev, Associate Professor of Robotics Institute
Carnegie Mellon University
LEARNING PROBABILISTIC GENERATIVE MODELS
FOR FAST SAMPLING-BASED PLANNING

© COPYRIGHT

2019

Jinwook Huh
To my love, Yugen
I have been a graduate student at the University of Pennsylvania for the past five years. During this time, I have received a lot of support and help from many people, and I am grateful to those who have encouraged and assisted me during my studies. My advisor, Prof. Daniel Lee, has supported me in various ways and provided me with very insightful advice. He has given me challenging and dynamic opportunities that have helped me grow as a researcher and increase my confidence along with my capabilities. His invaluable advice will continue to guide me throughout my entire professional life. I would like to thank Prof. Manfred Morari for his encouragement and guidance for my dissertation. Prof. Morari was particularly gracious with his time and attention. I am also grateful to Prof. Frank Park for his warm and genuine support and keen advice for my dissertation. I am thankful as well to Prof. Maxim Likhachev for his invaluable and insightful advice and assistance. I truly appreciate all members of my committee for their advice which made my dissertation more fruitful and complete. Thank you very much.

I want to express my appreciation to Dr. Yong Woon Park for his considerable support and mentorship for my PhD program. Without the support from Dr. Park, I would not have had a chance to start this program. I would also like to thank the team members who worked with me at ADD. Although many of them expressed their aversion to seeing me go, they also showed considerable encouragement to me for my efforts. I am very sorry to leave them, but I thank them for their support.

During the five years, I also received a lot of help from good friends. I want to thank Prof. Yungkyun Noh for his important support and mentorship. He gives me a lot of advice about PhD life and suggestions for my research direction whenever
we meet. I really appreciate him. I also appreciate Prof. Kwang Jin Yang and Prof. Kee-Eung Kim for their mentorship for my dissertation. I would like to thank Ömür for our discussions every Wednesday. I have learned many things over the two years that I have known him. Thank you very much. I would like to express my gratitude to the Lee group members who helped me a lot in my research and school life. Thanks to Steve, Bhoram, Heejin, Clark, Marcell, Xiang, Yongbo, Daewon, SJ, Chris, Steven, Ty, and Arbaaz for the good relationships which I will always remember. I would also like to thank my cubemate friends at GRASP, especially Ximing, Min, and Konstantinos, with whom I often had lunch together and sometimes discussed our research intensively. Thanks to my Korean friends, Sangdon, Kuk, Sooyong, Seungwon, Elijah, and Jaehyung, who help me in my UPenn life. I would also like to thank my colleagues at the Samsung AI Center who helped me a lot during my internship. I also express my gratitude to Cindy and Dale who help my family adjust to American life and my appreciation to Lizz for her help with my writing.

In fact, the sacrifice of my family was considerable during these five years. I am grateful to my family in Jeju (my parents, Yunghong and Soonja, and my siblings, Jeongsim, Suan, Jinil and Sunghee) and my family in Busan (my parents-in-law, Hojong and Younghee, and siblings-in-law, Byungkook and Hyeonjin) for supporting me. Thanks especially to my three children, Jeongyoon (Jeremy), Jaeyong (Jacob), and Junwoo (Jayden) for growing up well in spite of their fathers absence. I dedicate this dissertation to my wife, Yugene, who is always the most supportive person in my life.
ABSTRACT

LEARNING PROBABILISTIC GENERATIVE MODELS
FOR FAST SAMPLING-BASED PLANNING

Jinwook Huh
Daniel D. Lee

Due to their simplicity and efficiency in high dimensional space, sampling-based motion planners have been gaining interest for robotic manipulation in recent years. We present several new learning approaches using probabilistic generative models for fast sampling-based planning. First, we propose fast collision detection in high dimensional configuration spaces based on Gaussian Mixture Models (GMMs) for Rapidly-exploring Random Trees (RRT). In addition, we introduce a new probabilistically safe local steering primitive based on the probabilistic model. Our local steering procedure is based on a new notion of a convex probabilistically safety corridor that is constructed around a configuration using tangent hyperplanes of confidence ellipsoids of GMMs learned from prior collision history. For efficient sampling, we suggest a sampling method with a learned Q-function with linear function approximation based on feature representations such as Radial Basis Functions. This sampling method chooses the optimal node from which to extend the search tree via the softmax function of learned state values. We also discuss a novel constrained sampling-based motion planning method for grasp and transport tasks with redundant robotic manipulators, which allows the best grasp configuration and approach direction to be automatically determined. Since these approaches with the learned probabilistic models require large size data and time for training, it is essential that they are able to be adapted to environmental change in an online manner. The suggested online learning approach with the Dirichlet Process Mixture Model (DPMM) can adapt the complexity to the data and learn new Gaussian clusters with streaming data in newly explored areas without batch learning. We have applied these approaches in a number of robot arm planning scenarios and have shown their utility and effectiveness in simulation and on a physical 7-DoF robot manipulator.
# Contents

Contents vii  
List of Tables x  
List of Figures xi  
1 Introduction 1  
2 Related Work 7  
  2.1 Fast collision checking 7  
  2.2 Efficient sampling 9  
  2.3 Effective tree extension 12  
  2.4 Constrained sampling-based planning 15  
  2.5 Adaptive to changed environment 17  
  2.6 Online learning 19  
3 Learning High-Dimensional Mixture Models for Fast Collision Detection 21  
  3.1 GMM-based RRT 23  
    3.1.1 Mapping to circular feature space for the GMM 24  
    3.1.2 Incremental EM clustering of the GMM 25  
    3.1.3 Collision check based on the GMMs 26  
    3.1.4 Online learning of the GMM 27  
    3.1.5 Random sampling using the GMM distribution 29  
  3.2 Results 30  
    3.2.1 The learning result of GMM parameters 31  
    3.2.2 Comparison with other RRT approaches 32  
    3.2.3 The result of online learning model 33  
    3.2.4 GMM-based RRT with Webots simulation 36  
  3.3 Discussion 37  
4 Efficient Node Selection and Sampling with Q-Learning 39  
  4.1 Q-Learning 42  
    4.1.1 Temporal difference learning 42  
    4.1.2 Optimal action selection 43  
    4.1.3 Feature extraction 44
List of Tables

8.1 GMM and PRM computation times ...................... 148
8.2 Comparison of GMM-RRT with several approaches .. 149
List of Figures

1.1 Manipulation applications in various areas .......................... 2
1.2 Learning approach to resolve issues of RRT .......................... 4

3.1 A manipulator moves to the shelf to grasp an object. Ellipsoids mean the 90% confidence level of the collision. The blue and magenta lines depict trees, and the black line is the final path. The red dots are collision exemplars for the GMM. (Best viewed in color) ............... 22
3.2 The overall workflow of the GMM-based RRT. Our main contributions are highlighted in red. ..................................................... 23
3.3 (a) The GMM after three RRT iterations (b) The GMM after 10 RRT iterations. The learned GMM is refined online with additional exemplars every RRT iteration. It depicts a 3-dimensional space projected from a 7-dimensional space. .......................................................... 24
3.4 The GMM clustering result: (a) the clustering with Euclidean distance (b) the clustering in 2n-dimensional circular feature space. For the visualization, 7-dimensional space is projected to 2-dimensional plane. (Best viewed in color) .......................................................... 26
3.5 Illustrating GMM learning with Bhattacharyya distance: additional exemplars (red dots) are included into other clusters and GMMs are updated by MLE, or they make a new cluster (Best viewed in color) . 28
3.6 Illustrating the biased random sampling using the learned GMM distribution .......................................................... 30
3.7 The Kinect sensor data with AABB ................................. 31
3.8 The execution time of the RRT with respect to the number of components on the example of Fig. 3.7(a) ................................. 32
3.9 Probability density distributions of collision and collision-free exemplars with respect to the relative Mahalanobis distance on the example of Fig. 3.7(a) .......................................................... 33
3.10 Comparison with the basic and bi-directional RRT. It shows that the time and the number of collision checks are less than for the basic and bi-directional RRTs. The path length of the end-effector is similar to that of the bi-directional RRT .......................................................... 34
3.11 The GMMs are learned with additional exemplars as the RRT proceeds. This figure shows that RRT execution time is converged after the 3rd or 4th RRT procedure. .......................................................... 34
3.12 GMM is updated under the changed environment.

3.13 The result of simulation in Webots: (c) and (d) show the portion of the GMM-based collision checks.

4.1 QS-RRT planning with learned Q-function: (a) learned RBF features, (b) the RRT planning result in the two-dimensional C-space with the learned state value, and (c) manipulation planning with a real robot in the shelf environment.

4.2 The overall workflow of the suggested learning approach with RRT planning.

4.3 RBF features in the 2DoF planar manipulator of Fig. 4.11: (a) collision region (blue), collision-free region (red), and classification criteria (green) by an RBF threshold value and (b) RBF feature of collision regions.

4.4 RBF networks for the feature of Q-function approximation.

4.5 Sampling method in C-space during RRT planning: The softmax node selection method chooses the node based on the learned state values (green dot) among the open nodes (yellow dot). When the random node (blue dot) is chosen, the standard RRT extension grows the tree. The size of dots represents the probability of selection.

4.6 Softmax node selection chooses a node with probability in the open set. The selected node is removed and the new explored node is added in the open set.

4.7 Comparison between RRT and QS-RRT with obstacles (yellow) and safety boundary (green).

4.8 Nonholonomic constraints.

4.9 RRT planning with a nonholonomic constraint (Maximum curvature).

4.10 The learned state-value (left column) and RRT planning results with the learned sampling policy in the navigation problem (right column).

4.11 Comparison results with the standard RRT in the 2DoF planar manipulator: (a) manipulation results in the workspace, (b) QS-RRT planning results in the C-space with the learned state values, and (c) standard RRT planning results with collision samples (Magenta). (Best viewed in color)

4.12 Comparison results with the traditional approaches in the 2DoF planar manipulator as shown in Fig. 4.11.

4.13 The performance comparison of QS-RRT in the changed environment: (a) the original environment, (b) the perturbed environment with obstacle changes (blue circles), (c) and (e) show planning results in the original C-space, (d) and (f) show planning results in the perturbed C-space.

4.14 Performance with respect to the change level (lower is better).

4.15 The performance comparison: (a) RRT, (b) QS-RRT in the greatly changed environment.
4.16 Comparison results among RRT (the first column), QS-RRT (the second column), and nonholonomic QS-RRT (the third column) ...
4.17 Simulation and experiment in the physical system: (a) simulation conditions (b) experiment with a physical robot. The start and goal configurations are randomly assigned for planning in the shelf environment.
4.18 Comparison results with bidirectional RRT, QS-RRT, and bidirectional QS-RRT: (a) simulation results with 100 randomly assigned start and goal configurations and (b) test results in the physical robot system with eight sequential configurations.
4.19 The performance comparison: (a) RRT, (b) QS-RRT, (c) QS-RRT with the goal feature (d) QS-RRT with the RBF feature. The green dots are collision nodes.
4.20 $\beta(q)$ using RBF for navigation functions.
4.21 Comparison results: odd rows show navigation function results with RBF feature representation and soft node selection approach, and even rows show QS-RRT results.
4.22 Learning curve.

5.1 (left) Probabilistically safe corridor in 3D space constructed around a sample configuration (red) by using tangent hyperplanes (gray) of confidence ellipsoids of a learned GMM of C-space obstacles. (right) Local steering via probabilistically safe corridor in 2D space: An RRT is extended along the safe direction (red dotted line) towards the projection of a sample goal (red) onto the associated probabilistically safe corridor (red polygon), instead of the standard straight-line extension (blue dotted line) towards the sample goal.
5.2 Examples of learned GMMs. Ellipsoids show the confidence regions associated with the confidence level of $\kappa = 0.9$. (left) Gaussian mixtures in the 3D workspace shown in Fig. 5.9, (right) Gaussian mixtures in the C-space of a 2DoF planar manipulator.
5.3 GMM confidence regions. (a) Super level sets of individual Gaussians at confidence level $\kappa = \kappa_k$. (b) Super level sets of Gaussians at the confidence levels corresponding to a shared probability level. (c) An example C-space (collisions are in blue and free space is in red) and (d) the associated confidence ellipsoids of learned GMM distributions from collision samples (black in (c)).
5.4 Local steering via probabilistically safe corridors. (left) Example tree extension using a probabilistically safe corridor in 2D space, (right) Probabilistically safe corridor in 3D space.
5.5 Examples of task-space steering of a robotic manipulator. Here, the new configuration (magenta), suggested by the straight line planner from the nearest configuration (black), is adjusted to a better configuration (green) based on the associated probabilistically safe corridor.
5.6 RRT planning performance for a 2DoF planar manipulator.
7.3 Online learning with the rejection sampling ........................................ 120
7.4 DPMM results (a) collision clustering (b) collision-free clustering .... 120
7.5 Illustration of the probabilistic collision map ................................. 121
7.6 Online update of probabilistic collision map ................................. 122
7.7 RRT planning with probabilistically safe corridor based on online learning 123
7.8 Planned arm motion trajectories can vary significantly due to relatively small changes in pose between the robot base and objects in its environment: (a) original arm motion trajectory and (b) arm motion after pose change ................................................................. 125
7.9 Previous GMMs are predicted with IK according to the environmental change and updated with new exemplars as the GMM-based RRT proceeds (Best viewed in color) ................................................................. 126
7.10 The scheme of the GMMs prediction using IK in the changed environment. The transformed collision and collision-free distribution in the new C-space can be estimated by the change of the virtual end-effector in the Euclidean space ................................................................. 130
7.11 Visualization of predicted GMMs from previous GMMs by applying IK: the lower level joints (5th and 6th joints) are noticeably transformed compared to the upper level joints (1st and 3rd). 2-dimensional space is projected from 7-dimensional space for visualization purposes. .... 131
7.12 The training and test scenarios: (a) in order to train the model, the collision and collision-free exemplars are obtained from the ordinary RRT of this trajectory (b) the test is conducted with the learned GMMs in this scenario ................................................................. 135
7.13 The comparison results with the bi-directional RRT: the GMM-based RRT is relatively faster in the complicated path planning .......... 136
7.14 Execution time of the GMM-based RRT as the model update proceeds 137
7.15 Changed environment cases used in our experimental results: (a) translation (b) translation and rotation ................................. 138
7.16 Comparison results with the bidirectional RRT: error bars represent one standard deviation of 10 iterations ........................................... 139
7.17 Performance results with the previous GMMs and the predicted GMMs: (a) scenario (b) comparison results ........................................... 140
7.18 The novel test scenario is composed of different configurations from the training and previous test configurations ................................. 141
7.19 The test result on the example of Fig. 7.18: error bars represent one standard deviation of 10 iterations ........................................... 142

8.1 Planning results of PRM and RRT in the configuration space of a 2DoF planar manipulator ........................................... 144
8.2 PRM result: The vertices are randomly generated for constructing the roadmap. When the number of vertices is less than 300, it fails to find a path between the start (yellow) and goal (green) configurations .... 146
8.3 Performance curve of Lazy PRM: (a) execute time (b) collision checks 149
Chapter 1

Introduction

Planning for robotic manipulation with a high dimensional manipulator has received much attention in recent years due to its application in the service and industrial areas (Fig. 1.1). The sampling-based motion planners have become popular especially because of their simplicity and efficiency in high dimensional space [70, 83, 69, 11]. However, sampling-based approaches have several critical problems. For example, they spend most of their computational time on sampling and collision checking in complicated high-dimensional spaces [117], and they have difficulty in narrow passages; as a result, sampling-based planners might be inefficient with an increasing number of samples in a complicated environment. In order to increase the efficiency of their algorithms, we consider several issues of sampling-based planning.

First, the sampling-based planning needs more than tens of thousands of kinematic-based collision detection routines in order to generate a final path; therefore, the collision check is a serious impediment to the efficiency of sampling-based planning.

Second, it requires an efficient and effective sampling method since the performance of sampling-based planning depends on its sampling method. The effective sampling reduces the number of samples and collision checks; thus improving the performance of sampling-based planning. However, although Rapidly-exploring Random Tree (RRT) repeats its planning in the same environment, there is no systematic approach to improving its sampling method and tree extension. The RRT constructs the tree structure from scratch and finds a reliable path. Since it can obtain experi-
Figure 1.1: Manipulation applications in various areas

ence from many samples and collision check routines for each planning, exploitation of previous experience provides essential and valuable information for a fast and efficient RRT. The Probabilistic Roadmap (PRM), which is another sampling-based planning approach, also has a dependency on the sampling method for its performance.

In addition, the performance of sampling-based planning approaches could be degraded in a complicated environment, such as narrow passages, since the RRT and the PRM require many samples to find paths through those narrow passages. Since the biased search toward unexplored Voronoi regions does not consider collision space, the RRT has difficulty growing in the cluttered environment. In order to resolve this problem, the RRT should have a good sampling or tree extension algorithm to narrow passages efficiently.

Next, traditional sampling-based planning methods still face challenges for grasp and transport tasks with external constraints. These challenging issues arise from the fact that the constraint manifolds defined in the task space are not analytically mapped into the configuration space (C-space). For example, it is not obvious how constraints defined in the workspace (e.g., avoiding collision) can be combined with constraints defined in the task space (e.g., keeping an object upright) for planning.
in the C-space. In addition, sampling-based planning methods typically require setting a specific goal configuration, but it can be difficult to determine a single target configuration when there are many redundant grasping solutions.

Finally, it is difficult for traditional sampling-based motion planning algorithms to handle environmental changes in an efficient and accurate manner. A slight change in positioning between the robot base and its workspace can render previously found solutions for motion trajectories worthless. Naively, such a change necessitates re-planning a new motion trajectory from scratch, leading to heavy computational loads and slow and inefficient robot behaviors. For online learning, the approach can adapt the complexity to the data and learn new Gaussian clusters with streaming data in newly explored areas.

We suggest a systematic approach for sampling-based planning to achieve several planning missions efficiently using learning of probabilistic generative models in complicated high-dimensional environments. In many practical applications, multiple path trajectories are required for various tasks within the same or slightly changed environment, such as in the Amazon robotics challenge [2]. In these cases, the planner needs to be continually improved as the planning algorithm proceeds. The single-query planner, such as RRT, can find a path quickly by growing a tree from the start to the goal point without pre-processing [83, 69]. However, RRT has no systematic way to take advantage of information from previous experience. It requires computing samples and collision checks from scratch in building trees whenever a new start or goal configurations is specified, even when a similar solution has already been computed on a previous query in the workspace. In contrast, the multiple-query planner, such as PRM, can reuse prior information regardless of changes in the start and goal configurations [70]. However, if there is a small change in the environment, PRM needs to find the invalid part of the graph and then repair the roadmap before it generates a new path. Lazy techniques for PRM attempt to circumvent some of these issues by storing information updates and delaying validity checking during the graph construction process [19, 39]. Even though Lazy PRM may also be applied in perturbed environments, we seek a viable and systematic approach to efficiently
generate multiple trajectories using previous experience regardless of small changes to the environment.

In order to achieve efficiency with sampling-based planning, we present several new learning approaches using probabilistic generative models. First, we present a new approach for fast collision detection in high dimensional configuration spaces. The proposed method is based upon Gaussian Mixture Models (GMMs) that are learned using an incremental Expectation Maximization clustering algorithm trained online using exemplars provided by a slow, conventional kinematic-based collision detection routine. Moreover, the number of collision checks needed can be drastically reduced using biased random sampling from the learned GMM distribution.

We also suggest a learning approach for efficient sampling of RRTs based upon a learned state-action value function (Q-function). Our novel sampling method chooses the optimal node to extend the tree via the learned state value computed from the node feature representation. Our softmax node selection procedure avoids becoming stuck at the local minima and maintains the completeness property of RRTs. We employ several features in learning the Q-function, including the Radial Basis Function (RBF) scores of collision and collision-free regions in the C-space. Since this approach allows the RRT to explore efficiently while avoiding obstacles via the Q-function, the RRT planner is continually adapted to the environment in an online manner.

For effective tree extension, we propose an RRT method using the safety margin of a probabilistic model. This approach enhances the RRT performance near challenging
regions, especially in narrow passages, by adjusting the direction of the new node toward collision-free space. In order to guide the direction of the tree extension, we first construct a probabilistically safe corridor using the margin of the collision space of GMMs. The probabilistically safe corridor is a polyhedron composed of tangent half spaces on the surfaces of ellipsoids representing a given probabilistic confidence level. With a polyhedron of constructed safety corridor, we suggest an optimization-based tree extension algorithm to increase connectivity with less likelihood of collision. Since the tree extends within the probabilistically safe corridor, this approach can dramatically improve efficiency by reducing the number of collisions.

For grasp and transport tasks with a redundant robotic manipulator, we present a novel constrained, sampling-based motion planning method. We utilize a planning margin for grasping with constraints that allows the best grasp configuration and approach direction to be determined automatically. For manipulators with many degrees of freedom, our method efficiently chooses the optimal grasp pose when there are many redundant solutions. The method also introduces a parameterized intermediate pose that is optimized to determine the approach direction, increasing robustness under sensor uncertainty and execution errors. Our method also considers transporting the grasped object to the desired target position using the RRT algorithm that incorporates soft constraints via appropriate cost penalties.

Finally, we propose approaches to handling changes in the environment and for online learning to refine and update the model. In practical applications for robotic manipulation, the representation of collision and collision-free regions in C-space can change due to relative motion between the robot base and workspace. We show how to rapidly adapt to such changes by transformation of the parameters of the Gaussian mixture model to new configurations. The transformed model is initially used as a prior and then continually updated and refined as the RRT planning algorithm proceeds in real-time. In addition, we propose an online probabilistic generative model based on a nonparametric Bayesian model for efficient sampling-based motion planning. Although sampling-based planning approaches can exploit probabilistic models for significant improvement of performance, they require training for the learning
model. They also need to update the model whenever new samples are obtained in unexplored areas. We propose Dirichlet Process Mixture Modeling (DPMM) to learn probabilistic models with an online method. Since this approach is a nonparametric Bayesian model, it can adapt the complexity to the data and generate new Gaussian clusters with new samples in newly explored areas. Based on this online modeling, we generate probabilistic collision maps for fast collision checking or probabilistically safe corridors for local steering.

We demonstrate the effectiveness and efficiency of our algorithms on a number of simulated and experimental applications. Our experimental results show a marked improvement in computational efficiency in comparison to previously studied approaches.
Chapter 2

Related Work

There has been growing interest in sampling-based motion planners in recent years [83, 69, 70, 11], and we now discuss approaches in the relevant literature. First, we describe methods for reducing the computational time on collision checking and then explain the learning approach for sampling methods in sampling-based planning. Third and fourth, we will explain approaches to improving tree extension and constrained sampling-based planning algorithms. Finally, we will discuss adaptive and online learning approaches for the changed environment.

2.1 Fast collision checking

As we described, sampling-based approaches have a critical problem in that they spend most of their computational time on checking for collisions. It requires an efficient collision detection algorithm to maintain efficiency. Several algorithms have been investigated to reduce the number of collision checks by changing the sampling method [20, 57] or by delaying collision checks [117, 19]. Sanchez and Latombe [117] present a plausible approach that delays collision tests until they are required to check a candidate’s path, and the authors successfully demonstrate that this approach spends less time than previous approaches. However, since the delayed collision check retries the tree expansion whenever a collision is detected in the candidate’s path, the performance might be degraded in a complicated environment.
Other machine learning approaches have been conducted to make RRT more efficient [103, 4]. Pan and Manocha [103] suggest a probabilistic collision detection algorithm that considers the uncertainty of the environment with a Support Vector Machine (SVM). Kernel perceptron [35] represents collision and collision-free subspaces of a configuration space for fast collision checking. The recently proposed RR-GP algorithm [4] based on a learned motion pattern model that combines the flexibility of Gaussian Processes (GP) with the RRT is able to find a safe path in a dynamic and constrained environment. While research has successfully shown that machine learning can efficiently enhance the performance of the RRT, no approach has been able to reduce the computational time effectively.

In terms of collision detection, precise and fast collision detection algorithms have been developed in the robotics and computer graphics areas in order to accelerate computation [51, 86, 111, 75]. In the Robot Operating System (ROS), the motion planner uses the OctoMap, which is a kind of Octree-based representation of the environment for collision detection [54].

Unlike previous approaches, we demonstrate that the learned GMMs can detect a collision rapidly in the high dimensional configuration space, and the model is refined and updated online by new exemplars. Moreover, the number of collision checks is remarkably reduced by the biased random sampling from the learned GMM. Another key feature of our approach is that we introduce the GMMs for fast collision detection in the high dimensional configuration spaces using the incremental EM clustering, whereas other approaches deal with the GMMs for the imitation of demonstration trajectories [30] or exploit the GMMs to model the distribution of feasible positions and orientations for grasping [72]. In addition, it exploits collision-free GMMs for an efficient biased sampling compared to other approaches that handle only the hyperplanes of collision and collision-free regions.
2.2 Efficient sampling

There has been much prior research on robotic manipulation by sampling-based motion planning because of its simplicity and efficiency in high dimensional spaces. [83, 69, 70, 11]. However, sampling-based motion planners typically require a considerable number of node expansions and collision checks. Therefore, there are several biased sampling methods [57, 131] for improving the efficiency of the algorithm. Urmonson and Simmons [131] present heuristic quality functions based on the cost of guiding the biased RRT growth. Although their approach can improve the quality of paths efficiently in some environments, it can also result in high computational cost in more complicated environments. Lazy collision checking approaches have also been applied for reducing the number of collision checks [19, 61], but they may not perform well in cluttered environments, resulting in considerable backtracking and recomputation of invalid regions in a cluttered environment.

There are several sampling-based planning approaches for improving the quality of paths by incorporating cost values [66, 1, 46]. However, they assume predefined explicit cost functions, or they simplify cost functions with empirical weights of features, and they focus on minimizing the cost such as terrain cost or motions. Jaillet et al. [66] suggest a cost-map based RRT that minimizes the cost of the trajectory, but this approach increases planning time in favor of finding a more cost effective path. In addition, cost-based search methods have the problem of converging to a local minimum of the cost function. Barraquand and Latombe suggest a Monte-Carlo algorithm to escape from the local minima of the potential field [9]. However, this method requires adjusting many parameters and takes a long time with many local minima. In addition, it requires expensive computation to determine the cost of actions at every step in continuous action spaces.

Some works use previous tree structures to reduce computation time [61, 108] when the environment is unchanged. Phillips et al. [108] also try to reuse the constructed high-level tree structure for fast planning, but the constructed tree structure could be modified considerably even when the environment is only slightly changed.
Berenson et al. [15] and Coleman et al. [31] suggest approaches to storing and exploiting stored trajectories for new planning problems. The Lightning framework by Berenson et al. [15] saves previous planned trajectories in a path library and repairs the retrieved trajectories which have similar endpoints of the path to the start and goal for a new task. For repairing the infeasible path, it finds infeasible parts and applies the multiple RRTs to revise those trajectories. Since the Lightning framework saves trajectories, it requires a large size database for the path library. To reduce the size of the database, Coleman et al. [31] suggest the Thunder framework, which stores the experience in a sparse roadmap spanner. These approaches are very useful for invariant constraints such as joint limits and self-collisions. However, the stored trajectories sometimes need to update significantly under slight changes in the workspace. In addition, it is a time consuming process to find impaired parts in the previous trajectories and the roadmap. In addition, it is difficult to decide the size of the path library or the roadmap and remove unnecessary or impaired trajectories in the path library or from paths in the roadmap according to the change environment. In addition, it is a challenge to remove an inefficient trajectory once it is registered in the library, although it may find a better trajectory later. In contrast, since our approach learns the state value of the configuration space, it is robust to a slightly changed environment, and it is unnecessary to save a large size database for the previous experience.

Many Reinforcement Learning (RL) approaches have been eagerly applied in robotics [7, 106, 76] to use previous experience in planning and control. These RL approaches focus on learning parameterized controllers for high performance in control tasks, while our learning approach is better indicated for efficient sampling in RRT. There has been research which combines PRM with reinforcement learning [147, 104], but it is not applicable to continuous space in high dimensional space since those methods use a Q-table with discrete states for the Q-value. Ekenna et al. [41] use the local learning approach to increase spatial and temporal adaptability in a hybrid environment. However, their method uses the scaled Euclidean distance metric, which is not the exact distance metric between nodes. We use noble RBF
features to predict proximity to the collision boundary directly in the C-space.

Regarding the Temporal Difference (TD) learning approach with linear function approximations, there has been much research, such as Least Squares Temporal Difference (LSTD) and Least Squares Policy Iteration (LSPI) [21, 123], ever since Sutton et al. [127] proved that the policy search methods with function approximations guarantee convergence. These days, deep neural networks have been actively applied to Deep Reinforcement Learning (DRL) after Mnih et al. showed that the deep learning model can apply to the control policy in Atari games [94]. In addition, many policy search methods are able to handle the high dimensional space problems of robotic planning [90, 118, 95]. These approaches show that the deep learning model learns control policies directly from high-dimensional input data, and many DRL approaches have been applied to the real robot systems beyond simulations [85]. However, the deep networks suffer from high sample complexity with high dimensional features, such as images, to avoid local optimal solutions. Moreover, it takes a long time for training, and they require retraining procedures even when environments or conditions are only slightly changed.

There are several learning sampling policies based on deep neural networks [63, 110]. Conditional variational autoencoder (CVAE) [63] is suggested for learning complex manifold and valid regions for sampling based planning. This approach learns the CVAE from demonstrations of successful motion planning results and experience. It can generate random samples with the learned CVAE. The conditioning variable includes the initial state, the goal state, and an occupancy grid of the workspace. Motion Planning Network [110] learns an encoder network and a planning network; the encoder network encodes a point cloud into a latent feature, and the planning network predicts the next robot configuration given the current configuration, the goal configuration, and the latent feature of the point cloud. It trains the encoder network by minimizing the reconstruction error of the point cloud and trains the planning network to minimize errors between the estimated configuration and the configuration from the demonstration. In addition, to guarantee the feasibility of the trajectory based on the networks, it also proposes a hybrid algorithm by combin-
ing the network with a traditional planning algorithm. However, these approaches require many successful trajectories for training. In order to obtain many trajectories, it needs a lot of collision checks and training time. In addition, these approaches require a large amount of occupancy grid maps or point clouds for training. In contrast, since QS-RRT is a feature-based learning approach without workspace information, it has a smaller number of parameters and requires less training time and information compared to approaches with deep neural networks. Moreover, since we can add more features in the Q-function, QS-RRT is extendable to various problems, such as planning with nonholonomic constraints.

There are some approaches to leveraging features in the workspace for efficient sampling. Zucker et al. suggest an adaptive sampling strategy based on a policy gradient method for improving sampling-based planning [149]. Since this approach requires workspace features and discretization of the workspace, it is difficult to generalize to all problem domains. Learning from demonstrations for robot motion planning [141] improves the performance of planning with task constraints by teaching from human demonstrations. However, it requires time-consuming demonstrations whenever new task and constraints are assigned.

The key features of our approach are efficient sampling and tree extension based on the Q-learning approach, and RRT is continually improved as the planning algorithm proceeds. While previous learning approaches for sampling strategy are specialized for each problem case, our approach can be generalized to various problems since it is a more flexible structure based on the Q-function with linear function approximations. While other approaches leverage information from the workspace, our approach learns the configuration space directly; thus it is appropriate for manipulation problems which have no direct mapping between the C-space and the workspace.

### 2.3 Effective tree extension

Sampling-based planning approaches suffer from heavy computational time in complex environments because they typically require a considerable number of sample
configurations and their collision checks. Therefore, several biased sampling methods [20, 57] and rejection sampling methods [120, 121, 143] are proposed to reduce the number of sample nodes and so to improve computational efficiency. However, these approaches have many heuristic parameters and require explicit configuration space information, such as visibility or collision boundaries, which usually limit their application to low dimensional settings. Another alternative approach to increasing the computation efficiency is to reduce the number of collision checks, using either by lazy collision checking [61, 117, 19] or fast probabilistic collision checks [58, 60, 103, 4]. Prior planning experience and collision history is also leveraged in biased sampling in order to perform informed configuration space exploration [58, 23, 22]. Exact safety certificates are also utilized for minimizing the computational cost of collision checks [17]. However, these methods remain unable to fully address the challenges of sampling-based planning in complex configuration spaces.

To increase the connectivity around complicated regions, Zhang and Manocha present a steering approach that retracts sample configurations to become more likely to be connected to nearby nodes [146]. However, it requires a significant number of iterations to find a new collision-free configuration that is around the collision boundary along with an appropriate distance-to-collision measure. In practice, since measuring the exact distance-to-collision in high dimensional configuration spaces is very challenging, its applicability is also limited to low dimensional motion planning problems. Instead of explicit modeling of collision-free space, Principle Component Analysis (PCA) is used to adjust the tree extension heading to collision-free space based on local random collision-free samples [128] or neighborhood nodes [34] around the nearest node. Moreover, workspace topology is utilized in biasing configuration space exploration for planning around difficult regions [109, 40], but the topology of high-dimensional configuration space (e.g., robot manipulators) is significantly different and more complex than the corresponding workspace topology.

To leverage previous experience, utility functions for connectivity between vertices [22] are suggested for the sampling algorithm of PRM, and a task-guided planner [112] exploits the connectivity in the workspace. In addition, volumetric information
of collision-free space [81] is applied for an efficient search.

There are several sampling strategies which leverage the surface of collision configurations to find nearby free configurations [96]. Obstacle-Based PRM (OBPRM) [96] increases the connectivity of PRM in cluttered environments by sampling near the C-obstacle surface. Obstacle-Based RRT (OBRRT) [114] extends a tree using an obstacle vector based on information from the workspace obstacle surface or approximate C-obstacle surface. These approaches are novel in that they place samples close to obstacle boundaries or exploit information about C-obstacle regions to adapt the sampling strategy. However, since OBRRT [114] computes an obstacle vector based on collision configurations that lie in one plane of the C-space for computing the obstacle vector, unlike our approach, it is unable to determine the optimal direction considering all C-space regions existing in various directions. In addition, since OBRRT chooses a random parallel direction to the obstacle vector that lies on the plane, it is effective on the C-obstacle which is composed of planes; however, the C-obstacle of the manipulator has various surface shapes. In contrast, since our approach considers all adjacent C-obstacle regions, and it addresses various shapes of C-obstacles based on GMMs, it finds more optimal directions compared to previous approaches.

Local safe corridors [138, 47, 26, 92] have recently found significant applications in collision-free motion planning by using sequential composition of simple local planners [32]. Such safe corridors are usually constructed based on a convex decomposition of the environment, which requires an explicit representation. In [6], a sensory steering algorithm is proposed for sampling-based motion planning that increases the connectivity of randomized motion planning graphs, especially around narrow passages, by exploiting local geometry of configuration spaces via convex local safe corridors. This construction is further extended to integrate local system dynamics and local workspace geometry in kinodynamic motion planning [102]. However, the original construction of sensory steering requires an explicit representation of configuration space obstacles or an explicit distance-to-collision metric, and so its direct application to high dimensional motion planning is limited. We enhance this sensory steering al-
algorithm to adapt it to high dimensional settings, such as robotic manipulation, by defining probabilistically safe corridors that are constructed using a learned approximate probabilistic model of a configuration space.

2.4 Constrained sampling-based planning

For the grasping pose selection, several papers investigate manipulability and dexterity \cite{134, 145, 52} to evaluate the grasp configuration. However, they focus only on the evaluation of arm configuration without considering collisions during arm motion planning. Several algorithms have attempted to enhance grasping selection by using an integrative method for grasping and planning \cite{77, 50, 122}, but they require pre-learned information for the tasks.

In general, redundant manipulators with more than six degrees of freedom (DoF) have no analytic solution to the Inverse Kinematics (IK) problem. There are several sampling-based planning approaches for grasping and motion planning without explicit IK solutions or specific goal configurations \cite{137, 132, 28, 16, 133}. Ciocarlie et al. \cite{28} suggest collision-aware inverse kinematics for grasp selection by searching a feasible grasp configuration corresponding to the desired end-effector pose over the parameterized redundant joints. While their method finds a feasible solution efficiently, it is unable to choose an optimal solution for grasp planning. In order to perform sampling-based planning approaches without a specific grasp configuration, Weghe et al. use the Jacobian Transpose (JT) between the target pose and the current pose \cite{137}, and Bertram et al. \cite{16} suggest the IK-RRT, which generates random target configurations from plausible grasp configurations in order to exploit the advantages of the bi-directional RRT \cite{132, 133}. However, these approaches have the limitation that they are unable to impose orientation constraints.

Regarding sampling-based planning for transport with constraints, Berenson et al. \cite{13} suggest the projection of random configurations to the region defined by the manifold of constraints (CBiRRT). Many approaches suggest different iterative projection methods based on a similar framework \cite{124, 140, 100, 79}. However, these
approaches have the disadvantage that infeasible samples are projected on the boundary as opposed to the center of the nearest constraint manifold. In another approach, Stilman [124] suggests tangent-space (TS) and first-order retraction (FR) sampling schemes that are able to explore the entire C-space of constrained motion. However, their method still considers only hard constraints, and it could be inefficient in the cluttered environment since it finds the optimal solution iteratively and then checks for collisions at the end.

There are several methods for sampling-based motion planning with constraints [73] for real applications. Atlas-based RRT [65] projects a random point to a tangent space of the chart of the manifold. To improve the tree extension within the constraint manifold, it keeps the extension in the chart until it reaches the random point or the new node is unable to satisfy the constraint. However, Atlas-based RRT is time consuming for finding the neighboring charts when adding a chart to the atlas, and the mapping onto the atlas requires heavy computation. Tangent bundle RRT [71] projects the new node onto the constraint manifold and generates a new tangent space only when the RRT tree reaches a boundary of the constraint manifold or when the distance from the new node is larger than a certain threshold. This approach can significantly reduce the number of projections to the manifold. Kang and Park [68] suggest a Gaussian Process approach for a point-to-manifold distance function based on GMMs of a given data set in the constrained manifold. Since the representation is differentiated and it can obtain the gradient of the point-to-manifold distance function analytically, it applies CBiRRT without an iterative projection process for several constrained planning problems. While these approaches handle hard constraint, such as a closed chain constraint problem, our approach focuses on soft-constraints. We define cost functions including constraints and find a sample to minimize this cost function. This framework is appropriate for problems with soft constraints, such as maintaining the end-effector pose.

Whereas most constrained sampling-based planning approaches consider only hard constraints, Kunz and Stilman [79] handle soft constraints, such as the pose of the grasped object. The overall procedure is similar to that of CBiRRT [14] except for
the distance metric of joint space error. It also uses an iterative Jacobian pseudo-inverse to find the configuration that satisfies the constraints. Although they consider soft constraints, there is still the limitation of slowdown in the cluttered environment since the iterative Jacobian pseudo-inverse is applied for convergence to constraint manifolds.

Regarding sampling in the task space, Shkolnik et al. [119] suggest using a sampling approach directly in the task space considering collisions. The Jacobian pseudo-inverse method is used for the growth of the tree. It is able to reduce the computational time dramatically since sampling and searches for nearest neighbors in the task space are able to reduce the dimensionality and guide the RRT growth efficiently to the goal. However, in order to perform the RRT in the task space, there should be an efficient method for finding a configuration corresponding to a node in the task space. The suggested Jacobian pseudo-inverse method is effective if the task space is an open space with few obstacles, but it is difficult to use when the environment is cluttered with obstacles as explained in [135]. Furthermore, this approach considers only the position state in the task space, and it is limited, being unable to consider soft constraints.

The key feature of our approach in this thesis is that we exploit the planning margin and the parameterized intermediate configuration for grasping with constraints. Our approach also proposes sampling-based planning of transport manipulation tasks that incorporates soft constraints via appropriate cost penalties.

2.5 Adaptive to changed environment

Several learning approaches are suggested in order to use previous experience or to adapt to environmental changes. First, there have been proposals to accelerate the planning process using an Experience Graph [61, 108] when the environment is not changed over a long period of time. It generates an E-Graph from prior experience that consists of feasible end effector locations. However, since the 7DoF manipulator has redundant solutions corresponding to the end-effector, it might be possible that
other parts of the manipulator will collide when the environment is changed, although the end-effector does not collide in the same location as the prior experience. Furthermore, in order to consider dynamic conditions, the Experience Graph may be modified according to the environmental change.

Osa [101] suggests a learning approach with a demonstration under various environment states and statistical models of environmental conditions. Although it has the advantage of reducing computation time in various environment conditions, it requires knowledge of the conditions of the environment in advance. Moreover, it also works in only pre-defined environment conditions and could have trouble adapting to unexpected environmental changes.

Lien et al. [87] proposed a motion planner that generates roadmaps based on the geometric models of obstacles and exploits them when similar obstacles exist in the new environments. However, this approach is not flexible for unstructured obstacles. In order to solve dynamic planning problems, Multipartite-RRT is also presented for mobile robots and manipulators [148]. This approach suggests that the tree is expanded towards previous planning waypoints and exploits subtrees of previous tree structures. In addition, Rodriguez et al. [115] suggest a framework of a two-stage roadmap that is dynamically changed by moving obstacles.

In order to adapt to the changing environment, Lehner et al. [84] suggested incremental elastic roadmaps in order to handle dynamic environments. Whereas their work is conducted in a low dimensional workspace, our approach suggests a generative model in the high dimensional configuration space. Kallmann and Mataric [67] suggest a dynamic roadmap with cells and their connectivities in the workspace. It can easily adapt to environmental changes, but it is unable to represent all possible configurations due to the maximum number of the configurations corresponding to each cell. Burns et al. [24] also proposed a utility-guided algorithm, which uses the nearest neighbor model. The nearest neighbor model needs much more exemplars in a high dimensional space compared to our GMM-based model.

The key feature of our approach is that we exploit previously learned GMMs in order to adapt to the environmental change. Unlike the original GMM-based
RRT [58], it can predict and update the GMMs without the whole EM procedure. Moreover, the computational time of the online learning is remarkably reduced by the prediction and update of previous GMMs. In conclusion, with the initial GMMs by the EM procedure, the model can be adapted to a change of the environment if it is not significant. In contrast, other approaches exploit the old tree structures generated for past path plans [148] or construct graph structures for the feasible end-effector positions [61]. For large changes in robot pose, however, it is difficult to directly warp previously planned trajectories to the new configuration space, and these methods will only be successful under restricted environmental conditions.

### 2.6 Online learning

Several approaches use probabilistic models for representation of the configuration space, since the exact computation of distance-to-collision in complex high-dimensional configuration spaces is difficult [38]. Gaussian mixture learning in [58], support vector machine (SVM) [103] or kernel perceptron [35] are applied to represent collision and collision-free subspaces of a configuration space for fast collision checking and biased sampling over free space. In addition, an efficient sampling method based upon learning a state-action value function (Q-function) [59] and a probabilistically safe local steering primitive are presented to enhance the performance of sampling-based motion planning in complex high-dimensional configuration spaces. Although these approaches show that the performance is significantly improved with the probabilistic model, they require training for the learning model. In addition, they need to update the model whenever new samples are obtained in unexplored areas.

The Expectation-Maximization (EM) algorithm is a popular parametric modeling approach [36]. Although probabilistic models are used for modeling of observed data, traditional parametric models have difficulty in choosing the proper model and model complexity [129]. Figueiredo et al. [43] suggest an unsupervised learning approach for a finite mixture model. This approach automatically selects the number of components, and it is less sensitive to initialization. Since this approach updates a set of
parameters which depends on the number of mixtures using an iterative method, it requires heavy computation. This approach also uses variants of the EM algorithm for the initialization problem. While a parametric model fixes the number of parameters according to data size, a nonparametric Bayesian model assumes infinite dimensional parameters and adapts its complexity to the data [49]. The Dirichlet process (DP) is a nonparametric model which is suggested by Ferguson [42], and Neal proposes Gibbs sampling for the Dirichlet Process Mixture Model (DPMM) [97]. Blei and Jordan address a variational inference approach for DPMM [18]. In our work, we show that DPMM can be used to continually improve an RRT for trajectory planning by online learning.
In recent years, there has been growing interest in sampling-based motion planners [70, 83, 69], but sampling-based approaches have a critical problem in that they spend most of their computational time on checking for collisions. According to [117], this collision checking uses more than 90% of the total computational time; as a result, sampling-based planners might be inefficient with an increasing number of samples. If obstacles are composed of geometric primitive shapes, the collision can be easily checked by the distance between the manipulator and the surfaces of the obstacles. However, it is difficult to construct meshes of the surface from point clouds in real sensor data. Furthermore, the RRT needs more than tens of thousands of kinematic-based collision detection routines in order to generate a final path; therefore, the collision check is a serious impediment to the efficiency of an RRT.

In this chapter, GMMs are learned for fast collision detection in the high dimensional configuration space as shown in Fig. 3.1. With a learned mixture of Gaussians, the tree can expand by avoiding this space, and the collision is easily checked by the probabilistic distance from the mixture. It has the advantage of reducing the search space of the RRT; as a result, the computational time is remarkably reduced. First, the GMMs are learned using an incremental Expectation and Maximization (EM) clustering algorithm with exemplars provided by the kinematic-based collision
detection routine. Thereafter, the collision of the new node is determined by Mahalanobis distances from GMMs. If the decision based on the GMMs is ambiguous, the kinematic-based collision detection routine is conducted instead of the probabilistic distance. These exemplars obtained during the RRT planning are used to update the model again. Finally, the collision check of the generated candidate path is conducted with the kinematic-based collision detection in order to confirm its safety. If a collision is detected in the final path, random samples are distributed once again around the collision point, and the model is updated by the new exemplars.

Our approach differs from the traditional approaches using GMMs. We introduce GMMs for fast collision detection in high dimensional configuration spaces using the incremental EM clustering, whereas other approaches deal with the GMMs for the imitation of demonstration trajectories [30] or exploit them to model the distribution of feasible positions and orientations for grasping [72].

In summary, the core contributions of the chapter include 1) an efficient collision check algorithm based on a discriminative classifier with an ensemble of collision and collision-free GMMs; 2) a novel formulation for online GMM parameter learning; 3)
Figure 3.2: The overall workflow of the GMM-based RRT. Our main contributions are highlighted in red.

an adaptive sampling method using the learned GMM that considerably reduces the number of collision checks. This allows us to apply our method for motion planning of manipulators with high degrees of freedom in a complex geometric environment.

### 3.1 GMM-based RRT

In order to classify collision and collision-free samples using the GMM, the parameters of the GMM are learned by the incremental EM clustering algorithm. First, exemplars in \( n \)-dimensional configuration space are mapped into a \( 2n \)-dimensional circular feature space. The incremental EM clustering algorithm is then applied to learn an ensemble of GMMs in \( 2n \)-dimensional space: one is the collision GMM from collision exemplars, and the other is the collision-free GMM from collision-free exemplars. Using these learned GMMs, the collision of a new configuration sample is determined by the Mahalanobis distance from the GMMs. Next, the learned model is also refined online with every RRT procedure as shown in Fig. 3.3. Finally, for efficient sampling of tree expansion, we use the biased random sampling from the learned GMM distribution. Fig. 3.2 shows the overall workflow from learning to RRT planning, and Algorithm 1 presents pseudocode for the GMM-based RRT algorithm.
Figure 3.3: (a) The GMM after three RRT iterations (b) The GMM after 10 RRT iterations. The learned GMM is refined online with additional exemplars every RRT iteration. It depicts a 3-dimensional space projected from a 7-dimensional space.

3.1.1 Mapping to circular feature space for the GMM

For efficient EM clustering, the mapping method of \( n \)-dimensional joint configuration is necessary since joint angles have a singularity between \(-\pi\) and \(\pi\). Each \( n \)-dimensional input variable \( x \) is mapped into a \( 2n \)-dimensional circular feature space \( u(x) = (\cos(x), \sin(x)) \). Since joints of the manipulator are in an angle domain ranging from \(-\pi\) to \(\pi\), the Euclidean metric of the joint vector is unsuitable for clustering in the joint configuration space.

Fig. 3.4(a) shows the result of the clustering in \( n \)-dimensional space using Euclidean distance. Although the data around \(-\pi\) is near the points around \(\pi\), it is supposed that these points are far from each other. On the other hand, we considered that the exemplars are distributed on the \( 2n \)-dimensional sphere surface, thus near points come within the same cluster as can be seen in 3.4(b). From this result, we can conclude that the suggested \( 2n \)-dimensional circular feature space is more suitable for the angle joint features.
Algorithm 1 GMM-based RRT

Require: \( X \_{\text{col}}, X \_{\text{free}} \) exemplars

1: \( G_{\text{col}}, G_{\text{free}} \leftarrow \text{BUILD\_GMM}(X \_{\text{col}}, X \_{\text{free}}) \)

2: \( \mathcal{T} \).init\((q_{\text{init}})\)

3: while Distance\((q_{\text{goal}}, q_{\text{new}}) > d_{\text{min}} \) do

4: \( q_{\text{rand}} \leftarrow \text{StateSampling}(G_{\text{col}}-\text{free}) \)

5: \( q_{\text{near}} \leftarrow \text{NodeSelection} (\mathcal{T}, q_{\text{rand}}) \)

6: \( q_{\text{new}} \leftarrow \text{NodeExpansion}(\mathcal{T}, q_{\text{rand}}, q_{\text{near}}) \)

7: \( d_{\text{col}}, d_{\text{free}} \leftarrow \text{MahalanobisDist}(q_{\text{new}}, G_{\text{col}}, G_{\text{free}}) \)

8: if \((d_{\text{free}} - d_{\text{col}}) > \delta_{\text{col}}\) then

9: \( q_{\text{new}} \) is collision

10: else if \((d_{\text{col}} - d_{\text{free}}) > \delta_{\text{col-free}}\) then

11: \( q_{\text{new}} \) is collision-free

12: else

13: \( \text{KinematicCollisionCheck}(q_{\text{new}}) \)

14: end if

15: if \( q_{\text{new}} \) is then

16: \( \mathcal{T} \).addTree\((q_{\text{new}})\)

17: end if

18: end while

19: \( \text{CollisionFreeCheck}(X_{\text{pathset}}) \)

20: \( G_{\text{col}}, G_{\text{free}} \leftarrow \text{UPDATE\_GMM}(G_{\text{col}}, G_{\text{free}}, X_{\text{new}}) \)

### 3.1.2 Incremental EM clustering of the GMM

In \(2n\)-dimensional circular feature space, we apply incremental EM clustering in order to learn GMM parameters. The cluster merging is also determined by the Bhattacharyya distance during the EM clustering procedure. For the initial parameters of clusters in the EM procedure, we apply the idea of the incremental k-means algorithm suggested by Likas et al. [88]. They show that the solution is optimal and consistent experimentally, although they could not prove it theoretically. This approach is modified and implemented as follows. First, the clustering is initialized with one cluster, and the initial center point of the next cluster is found depending on the equation as below.

\[
b_n = \sum_{j=1}^{N} \max (d_{k-1}^j - \| x_n - x_j \|^2, 0), \quad i = \arg \max_n b_n,
\]

where \(d_{k-1}^j\) is the squared distance between \(x_j\) and the center of the cluster to which
it belongs in the prior $k - 1$ clusters [88].

Thereafter, EM is conducted to learn the GMM parameters with this new initial center $x_i$ and the previous GMM’s centers, $\mu_{1,...,k-1}$. After the EM clustering, we use the Bhattacharyya distance$^1$ for the measure of similarity to determine whether there is a cluster included in other clusters. If the value of Bhattacharyya distance is high, it means that probability distributions $p$ and $q$ overlap each other considerably and the cluster is merged into another cluster. This incremental EM clustering is repeated by increasing clusters until the number of clusters arrives at a given number $K$ or it is converged since clusters are merged repeatedly.

### 3.1.3 Collision check based on the GMMs

Given the learned GMMs, the collision check requires less time. It can determine the collision with the minimum Mahalanobis distances from the ensemble of GMMs. If the decision based on the GMMs is ambiguous, i.e., it is between two thresholds, the kinematic-based collision detection is conducted. The $\delta_{col}$ and $\delta_{col-free}$ are determined

---

$^1$Bhattacharyya distance is as follows:

- Bhattacharyya coefficient: $BC(p, q) = \sum_{x \in X} \sqrt{p(x)q(x)}$
- Bhattacharyya distance: $D_B(p, q) = -\ln(BC(p, q))$
by the distributions of the relative Mahalanobis distances between the collision and collision-free GMMs.

- Mahalanobis distance from $k$th Gaussian of GMM

$$D_{M_k}(x) = \sqrt{(x - \mu_k)^T S_k^{-1}(x - \mu_k)}$$

- Collision

$$\left( \min_{col\text{-}free} D_{M_j}(x) - \min_{col} D_{M_i}(x) \right) > \delta_{col},$$

- Collision-free

$$\left( \min_{col} D_{M_i}(x) - \min_{col\text{-}free} D_{M_j}(x) \right) > \delta_{col\text{-}free},$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, M$

where $N$ is the number of collision GMM components, and $M$ is the number of collision-free GMM components.

### 3.1.4 Online learning of the GMM

Now that we have explained the GMM with the incremental EM procedure, we will introduce the idea of refining and improving the learned model online as the RRT planning algorithm proceeds. For the update of the GMM parameters, there are two kinds of new exemplars; one is obtained from the kinematic-based collision check during the RRT procedure, and the other is intentionally generated when a collision is detected in the final candidate path.

For the update with new exemplars gathered during the RRT, each exemplar is assigned to a cluster by comparing the probability of each Gaussian model, then the Gaussian components are updated by maximum likelihood estimation (MLE) as shown in Fig. 3.5.

Second, when the collision is detected in the final candidate path, it is caused by the inaccuracy of the collision detection or a change in the environment. Therefore
the new random exemplars are distributed around the collision points to update the GMM. In order to determine whether these exemplars are included in other clusters, the Bayes classification error is considered. Unfortunately, however, minimizing the Bayes classification error is intractable due to the non-analytical behavior of the minimum function; therefore, the Bhattacharyya distance is applied to update the GMM instead of the Bayes error [99]. If the variation of the distribution is small including the new exemplars, the distribution is updated with them; on the other hand, if the variation of the distribution is considerable, a new Gaussian distribution is generated for the new exemplars. Fig. 3.5 illustrates that the GMM is learned online when new random exemplars are generated. One case is that, as shown in figure 3.5, new random exemplars become new Gaussian clusters. The other case is where the exemplars are included in the nearest Gaussian distribution. Since the learned model is efficiently refined as the RRT planning algorithm proceeds, the accuracy and the confidence can be increased through the online learning procedure.
3.1.5 Random sampling using the GMM distribution

The RRT has a shortcoming which causes it to explore slowly when the sampling domain is not well adapted to the problem [143]. For the efficient Probabilistic Roadmap (PRM) planner, Boor et al. [20] suggest Gaussian sampling around collision points. It is more essential to choose samples around a collision region for the roadmap. Although this approach is suitable for the PRM, it is not adequate for the RRT since samples from the region around the collision are less feasible than those from other regions.

We apply the distribution from the GMM to reduce the number of collision checks. For the tree extension in the RRT, a random node $x_{\text{rand}}$ is chosen from the $K$ multivariate Gaussian mixture components of the collision-free GMM as below.

$$x_{\text{rand}} \sim \sum_{k=1}^{K} \pi_k \mathcal{N}_{\text{col-free}}(\mu_k, \Sigma_k) + \lambda \cdot U_{2n},$$

where $\pi_k$ is the probability that a new random node is drawn from the $k$th mixture component, and $\lambda \cdot U_{2n}$ is a regularization term to prevent overfitting. $U_{2n}$ means the uniform distribution over $2n$-dimensional circular feature space. We can determine the hidden variable $\pi_k$ through the EM procedure, but we assume that the mixture weights are equal, $\pi_k = 1/K$.

The selection of the random node $x_{\text{rand}}$ depending on the collision-free GMM does not mean that the new node $x_{\text{new}}$ is always the collision-free node, but it means that the tree has the tendency to extend toward collision-free distributions as shown in Fig. 3.6. This approach can reduce the number of collision checks since it increases the probability of selecting the collision-free node for the new node. Therefore, it can easily find a path to the goal state following the collision-free distribution. The regularization exists in order to prevent eliminating the possibility of finding a more efficient path existing outside the space of the GMMs. If the direction of the new node is selected based only on the GMMs, this approach might overfit, and it is therefore inefficient with different initial and goal states. Moreover, these uniformly chosen nodes are utilized to refine the GMMs by online learning. Following this approach,
the tree can be naturally extended while avoiding obstacles.

3.2 Results

This section provides a series of experimental results which validate the proposed approaches. First, we evaluate each step with data using the Microsoft Kinect sensor; one situation involves a shelf and the other is a microwave. Next, simulations are performed on arm motion planning for the THOR robot [144] with the Webots simulator of Cyberbotics Ltd. All experiments are conducted with a 2.7GHz PC using Matlab. In order to verify the performance of the GMM-based RRT, it is compared with the basic RRT and the bidirectional RRT (RRT-Connect) [78] which incorporate the kinematic-based collision detection algorithm with the Octree structure representation of the Axis-Aligned Bounding Box (AABB) as shown in Fig. 3.7. GMM-based RRT conducts the same kinematic-based collision detection routine when the GMM-based collision check is ambiguous.
3.2.1 The learning result of GMM parameters

For the GMM, the number of components $K$ and the probabilistic distance thresholds are important parameters. The execution time of the GMM-based RRT with respect to the number of Gaussian components is plotted in Fig. 3.8, which reveals that the execution time decreases sharply at first but slows its rate of decrease once the number of components reaches seven. Therefore, we choose seven for the number of components for the GMM. In addition, Fig. 3.9 shows the probability density function of the relative distance between two Mahalanobis distances of the collision and collision-free GMMs. We determine the $\delta_{col}$ and $\delta_{col-free}$ using the distribution in Fig. 3.9. On the basis of $\delta_{col}$ and $\delta_{col-free}$, the false positive rate is about 5%, and the false negative rate is about 23%. The false positive rate is more critical to the performance of the RRT, because false positive exemplars lead to failure of the RRT. The final path could be infeasible when false negative samples are included in the final path. On the contrary, most false negative exemplars with relatively small distances usually exist near the collision exemplars, meaning that they might have a high probability of collision. Therefore, the false negative rate only slightly affects the performance of the RRT. However, the false negative samples could also degrade the performance of the RRT when the rate is too high, although these samples have
3.2.2 Comparison with other RRT approaches

In order to evaluate the performance of the GMM-based RRT, we compare it with other approaches: the basic RRT and the bidirectional RRT. We assume 10 cases which have different initial and goal configurations of the 7 DoF manipulator with the Kinect sensor data, as shown in Fig. 3.7(a). To learn the GMM, we utilize the exemplars obtained from 10 RRT planning iterations of Case 1, and then 50 RRT iterations are repeated for the 10 cases for accurate evaluation without the randomness property of the RRT. As can be seen in Fig. 3.10, the GMM-based RRT is five times faster than the bi-directional RRT, and the number of collision checks of the GMM-based RRT is also less than that of the bidirectional RRT. In addition, both the GMM-based RRT and the bi-directional RRT have similar lengths of trajectory of the end-effector. Summing up the three results, the GMM-based RRT is better than the other approaches.
than the other approaches in that its running time is shorter, the number of collision checks is smaller, and the path length does not get any longer when viewing these results with 10 cases.

### 3.2.3 The result of online learning model

In order to accurately evaluate the online learning, we show the resulting convergence of the model and the update of the model under the changed environment. The number of exemplars for the learning increases as the RRT planning algorithm proceeds. In Fig. 3.11, after every RRT procedure, the GMM model is updated with new exemplars, and the RRT proceeds again. Less time is consumed for the RRT with the more accurate model. The execution time of the RRT is quickly converged after the 3rd or 4th RRT procedure. In this result, the first RRT procedure is conducted with no prior exemplars, i.e., the RRT is conducted with only the kinematic-based collision detection. It shows that the model could converge and still be more accurate with more exemplars, leading to a decrease in computation time and number of collision checks.
Figure 3.10: Comparison with the basic and bi-directional RRT. It shows that the time and the number of collision checks are less than for the basic and bi-directional RRTs. The path length of the end-effector is similar to that of the bi-directional RRT.

Figure 3.11: The GMMs are learned with additional exemplars as the RRT proceeds. This figure shows that RRT execution time is converged after the 3rd or 4th RRT procedure.
Figure 3.12: GMM is updated under the changed environment.
Fig. 3.12 shows the result of updating the initial model under the changed environment. At first, the microwave is closed, and the initial GMM is learned under this closed condition. However, the microwave is opened later, and the final path generated by the RRT collides, as shown in Fig. 3.12(b). When the collision is detected in the final path, the new random exemplars are distributed around the collision points. The GMM is then updated with these new exemplars and finally, the RRT generates a new final path as shown in Fig. 3.12(c) and 3.12(d). Fig. 3.12(e) is the collision GMM in the configuration space when the microwave is closed, and Fig. 3.12(f) shows the new exemplars around the collision point. Fig. 3.12(g) is the updated collision GMM including the new exemplars. We can see from Figs. 3.12(c) and 3.12(d) that the GMMs are updated successfully and the RRT works well with the updated GMMs without the collision. These results verify that this algorithm can be applied in online learning.

### 3.2.4 GMM-based RRT with Webots simulation

In order to demonstrate reliability in the THOR robot, we conducted two scenarios in the Webots simulation. In the first scenario, as shown in Fig. 3.13(a), the robot moves its arm to the lower shelves and then moves to the upper shelf, and in the second scenario, the robot moves its arm and gets ready to rotate a valve as shown in Fig. 3.13(b). The first scenario has a more complicated condition because it is difficult for the robot to insert its arm between the shelves and to move its arm from the lower shelf to the upper one. The robot has two Hokuyo Lidar sensors in the chest and the head for perception in the Webots simulator. Figs. 3.13(a) and 3.13(b) show the simulation environments. Each motion is repeated for 10 iterations with the same GMMs, and the result is similar to the result shown in Fig. 3.10. The average time of the GMM-based RRT is 8.76 seconds in the shelf case and 3.00 seconds in the valve case, whereas the average time of the bi-directional RRT is 91.65 seconds for the shelf and 10.93 seconds for the valve. The computational time of the GMM-based RRT is less than that of the bidirectional RRT. In addition, the GMM-based RRT is about 10 times faster than the bi-directional RRT for the shelf, and the GMM-based
RRT is about 3 times faster than the bidirectional RRT for the valve. It shows that the GMM-based RRT works more effectively in the complicated environment. Figs. 3.13(c) and 3.13(d) compare the number of kinematic-based collision checks and the number of GMM-based collision checks in each iteration. As the portion of GMM-based collision checks is increased, the computational time of the GMM-based RRT is reduced.

### 3.3 Discussion

This chapter has presented fast collision detection in high dimensional configuration spaces for the RRT. The proposed method is based upon Gaussian Mixture Models (GMMs) that are learned using an incremental Expectation Maximization clustering
algorithm, and the models are continually refined and improved as the RRT planning algorithm proceeds. Additionally, the number of collision checks can be remarkably reduced using the biased random sampling from the learned GMM. Following this approach, the overall results verify that the GMM-based RRT is five times faster than previous approaches. We can enhance the approach with an idea to classify samples by the tree structure of the GMMs, which can improve the accuracy and confidence of the collision check based upon the GMM.
Chapter 4

Efficient Node Selection and Sampling with Q-Learning

This chapter suggests a learning approach for efficient sampling of Rapidly-exploring Random Trees (RRTs) based upon a learned state-action value function (Q-function). In a number of applications, such as for the Amazon Robotics Challenge [2], multiple realizations of path trajectories need to be generated for tasks in the same or slightly perturbed environments. In these cases, the planner needs to be continually improved as the planning algorithm proceeds. A single-query planner, such as the RRT, can find a path quickly by growing a tree from the start to the goal point without pre-processing [83, 69]. However, the RRT does not have any systematic way to take advantage of information from previous experiences. It requires computing samples and collision checks from scratch in building trees whenever a new start or goal configuration is specified, even when a similar solution has been computed on a previous query in the workspace. On the other hand, a multiple-query planner, such as the Probabilistic RoadMap (PRM), can reuse prior information regardless of changes in the start or goal configurations [70]. However, if there are small changes in the environment, the PRM needs to find the invalid part of the graph and then repair the roadmap before it generates a new path. Lazy check techniques for PRM attempt to circumvent some of these issues by storing information updates and by delaying validation during the graph construction process [19, 39]. Even though Lazy
Figure 4.1: QS-RRT planning with learned Q-function: (a) learned RBF features, (b) the RRT planning result in the two-dimensional C-space with the learned state value, and (c) manipulation planning with a real robot in the shelf environment.

PRM may also be applied in perturbed environments, we seek a viable and systematic approach to efficiently generating multiple trajectories using previous experience regardless of small changes to the environment.

We suggest a novel approach to learning an efficient sampling strategy using prior information for the RRT. Our approach, Q-function sampling RRT (QS-RRT), minimizes the number of tree extensions and escapes local minima efficiently by using a probabilistic node selection procedure based upon the learned value of the node in the configuration space (C-space) as shown in Fig. 4.1(b). The ideal RRT planner must grow the tree towards the goal configuration while avoiding obstacles and minimizing the expansion to irrelevant areas. QS-RRT chooses a node for tree extension by an optimal sample selection method based on the learned state value and prevents becoming stuck at local minima using efficient sampling. In addition, the suggested approach can find the optimal extension given a node in the tree based on the learned state-action value function (Q-function) in the C-space. QS-RRT is a novel planning framework that combines the fast random exploration of RRT with tree extensions based on Q-learning. Therefore, it leads to fast RRT planning and efficient robot behaviors. In addition, since QS-RRT is robust to the slowly or slightly changing environment, it can overcome the limitation of the RRT and PRM planners.

The proposed learning approach is based on Temporal Difference (TD) learning with a generalized linear Q-function approximation. For the feature-based repre-
sentations of the Q-function, we suggest using the score values of the Radial Basis Function (RBF) features of collision and collision-free regions in the C-space along with other feature functions. Since this approach learns the weight coefficients of the RBF to predict proximity to the collision region in the C-space, it differs from other RL approaches that train RBF coefficients for state value function approximation [126]. In our approach, therefore, it is unnecessary to re-learn the RBF coefficients when either the start or goal positions are changed. In addition, we propose a softmax node selection method to sample a node for tree extensions based upon the softmax probability distribution of state values. Since the softmax node selection includes a finite probability for a random RRT tree extension, our suggested planner with this softmax node selection has the probabilistic completeness guarantee of the RRT as well as computational efficiency in high-dimensional spaces.

In summary, the core contributions of this paper include: 1) a learning approach and softmax node selection method for efficient RRT sampling; 2) an integrated RRT planner that takes advantage of the optimal node extension based on Q-function and the traditional RRT extension; and 3) a novel RBF feature representation in the C-space for effective Q-function approximations. Since our approach also satisfies constraints upon joint limits and collision avoidance with performance improvement, it allows for motion planning to be much faster in various applications, such as manip-
ulation planning, compared to previous planning algorithms, as illustrated in Fig. 4.1. The key features of our approach are efficient sampling and tree extension based on the Q-learning approach, and RRT is continually improved as the planning algorithm proceeds, as shown in Fig. 4.2.

4.1 Q-Learning

In our approach, the states are given by a continuous C-space, and the action is to choose a tree extension given a particular state. To describe the learning approach, we first define the following Markov Decision Process \((\mathcal{S}, \mathcal{A}, T, R, \gamma)\), where \(\mathcal{S} \subseteq \mathbb{R}^n\) is the state space, \(\mathcal{A} \subseteq \mathbb{R}^n\) is the action space, \(R: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}\) is the reward function, \(T: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}\) is the transition function, and \(\gamma \in [0, 1]\) is the discount factor. The transition function is defined as follows: \(T(s, a) = s',\) where \(s' = s + a\) denotes the next state from a state \(s\) under action \(a\). The reward function \(R\) is positive on goal states and negative on collision states. Since we learn the Q-function to maximize the sum of the rewards, the tree can have growth biased toward the goal through collision-free space by the learned Q-function.

4.1.1 Temporal difference learning

Due to the continuous space and action spaces, the action-state value cannot be tabulated for all states and actions. Function approximation is needed to reduce the number of training data and generalize the experience for similar state-action values. We approximate the Q-function with features \(\phi(s, a)\), described in Section 4.1.3, and apply the TD approach for learning of the Q-function. Let the weights of feature functions be \(w = [w_1, \ldots, w_n]; s, s' \in \mathcal{S};\) and \(a \in \mathcal{A}\). Then,

\[
Q(s, a) = w_1\phi_1(s, a) + w_2\phi_2(s, a) + \cdots + w_n\phi_n(s, a), \quad (4.1)
\]

\[
J(w) = \frac{1}{2} [R + \gamma \max_{a'} Q(s', a') - Q(s, a)]^2. \quad (4.2)
\]
To find a minimum of $J(w)$, we can update the $w$ by the gradient descent method as follows:

$$w \leftarrow w + \alpha \Delta w$$ \hspace{1cm} \text{(4.3)}

$$\Delta w = -\nabla_w J(w) = (R + \gamma_{a'} Q(s', a') - Q(s, a)) \nabla_w Q(s, a),$$ \hspace{1cm} \text{(4.4)}

where $\nabla_w Q(s, a) = [\phi_1(s, a), \phi_2(s, a), \ldots, \phi_n(s, a)]^T$, and $\alpha$ is a step size.

### 4.1.2 Optimal action selection

In the previous section, we describe the learning approach for the Q-function. Even though we have a good Q-function approximation, we need a method for finding an action which corresponds to the maximum state-action value in a given state. In this chapter, the given state is a node in the tree, and the action is an extension of the tree from the node. We need the optimal tree extension with a given state which can maximize the state value of the next state. We suggest using Nelder-Mead (NM) optimization to find the extension from the node by maximizing the Q-function. Since tree extensions have a fixed step size, the next state $s'$ is on the surface of a hypersphere. Therefore, we project the transformation of vertices of the NM procedure, such as reflection or expansion, onto the surface of the hypersphere by normalization. The approach is appropriate since the transformation of the vertices does not change significantly in any iteration. To overcome the local minima problem of the NM method, we check the state values in the principal axes and then generate the initial vertices as $n + 1$ random vertices with Gaussian random samples around the principal axis having the maximum state value. Furthermore, to compute an average of the joint angles, we apply the mean of circular quantities, introduced in [136]. With this optimization approach, we can find the optimal action $a(s)$ and state value $v(s)$ to maximize the state-action value given a state as follows:

$$a(s) = \arg \max_{a'} Q(s, a'), \quad v(s) = \max_{a'} Q(s, a').$$
4.1.3 Feature extraction

We have described the form of the Q-function in Equation 4.1. Henceforth in this section, we will describe how to construct features explicitly representing characteristics of states in the C-space. We recommend several features that are effective for Q-function approximation.

First, we suggest using RBF feature representations in the C-space. It is intractable to measure the distance to collision boundaries explicitly in the C-space [38]. We predict proximity to the collision region using RBF features instead of the distance. We learn the weight coefficients of activation values in the RBF, and the weighted sum of the activation values (i.e., score values) can measure the proximity to the collision region. The RBF features can represent proximities from collision and collision-free regions, as shown in Fig. 4.3(b). The green line in Fig. 4.3(a) shows the boundary of the collision determined by the score values. In this chapter, we use these score values as features for the Q-function approximation. Samples for RBF networks can be obtained during the RRT procedures since the RRT generates a lot of samples and then checks whether the samples satisfy collision avoidance and constraints of self-collision and joint limits. RBF features include the information of constraints since samples already have the information. In addition, it is unnecessary to re-learn the RBF coefficients when either the start or goal positions are changed.
Figure 4.4: RBF networks for the feature of Q-function approximation

since the RBF features are unrelated to them.

As shown in Fig. 4.4, we first make clusters by k-means clustering. We let \( k \) denote the number of clusters and \( m_j \) denote the number of samples in the \( j \)-th cluster. Then, \( m = \sum_{j=1}^k m_j \), \( x_i \in \mathbb{R}^n \), \( X \in \mathbb{R}^{m \times k} \), \( Y \in \mathbb{R}^{m \times 2} \), and the \( j \)-th radial basis function \( \psi_j(x) = \exp(-\beta_j \| x - \mu_j \|^2) \). Then, we compute the variance of the samples \( \sigma_j^2 \) of each cluster as follows:

\[
\sigma_j^2 = \frac{1}{m_j} \sum_{i=1}^{m_j} \| x_{ij} - \mu_j \|^2, \quad \beta_j = \frac{1}{2\sigma_j^2} \quad \text{for} \quad j \in 1, \ldots, k,
\]

where \( x_{ij} \) is a sample in the \( j \)-th cluster, and \( \mu_j = \frac{\sum_{i=1}^{m_j} x_{ij}}{m_j} \).

Once we obtain the \( \beta_j \) for each of the clusters, we can compute the weight coefficients \( \Theta = [\theta^c \quad \theta^{nc}] \) of radial basis functions by the method of least squares as follows:

\[
\Theta = (X^T X)^{-1} X^T Y, \quad [X]_{ij} = \psi_j(x_i), \quad Y = [y^c \quad y^{nc}],
\]

\[
y^c_i = \begin{cases} 
1 & \text{if } x_i \text{ collision} \\
0 & \text{otherwise}
\end{cases}, \quad y^{nc}_i = \begin{cases} 
1 & \text{if } x_i \text{ collision-free} \\
0 & \text{otherwise}
\end{cases}
\]

With these coefficients and radial basis functions, we compute score values of a configuration state by taking the weighted sum of the activation values. We use the
following score values for the features of the Q-function:

\[ \phi_1(s, a) = \sum_{j=1}^{k} \theta^c_j \psi_j(s+a), \quad \phi_2(s, a) = \sum_{j=1}^{k} \theta^{nc}_j \psi_j(s+a), \]

where \( \theta^c_i \) is the weight associated with collision configurations, \( \theta^{nc}_i \) is the weight associated collision-free configurations, and \( \psi_i \) is \( i \)-th radial basis function with \( i \in \{1, \ldots, k\} \).

Fig. 4.3(b) shows score values of states for the collision in the C-space of the 2 Degree of Freedom (DoF) planar manipulator.

Next, we define the distance from the state to the goal in the C-space as another feature function. For the 7DoF manipulator, since we consider a periodic property of joint angles, we define it as follows: \(^2\)

\[ \phi_3(s, a) = \| 1 - \cos(s + a - q_{goal}) \|, \quad s, a, s_{goal} \in \mathbb{R}^n. \]

We define an additional feature that is the distance between the end-effector position of the state and the end-effector position of the goal configuration,

\[ \phi_4(s, a) = \| t(s + a) - t(q_{goal}) \|, \]

where \( t(q) \) denotes the translation components of the forward kinematics of configuration \( q \).

### 4.2 Sampling for tree extension

In this section, we describe a novel sampling approach for the RRT including a probabilistic node selection method based upon the learned Q-function as shown in Fig. 4.5. The strategy samples a node in the tree with the probability based on the learned state values and finds the optimal extension based on the Q-function given the node. This approach combines with the random sampling extension of RRT since the tree extension based on the learned Q-function can be trapped in local minima. On the

\(^2\)If \( Z \in \mathbb{R}^n \), then \( \cos(Z) \) is a vector whose \( i \)-th entry is \( \cos(Z_i) \).
other hand, the random sampling of RRT has good exploration capability toward the larger Voronoi regions, but it could expand to irrelevant areas. To minimize the tree extension and escape the local minimum efficiently, our approach combines the RRT with the tree extension using the probabilistic choice of the node based upon the learned state values. In addition, our suggested approach differs from other previous sampling approaches for tree extension, such as Expansive-Spaces Tree (EST) planner [56, 107] or the adapting RRT growth method [38]. While the EST planner chooses a node based on the number of neighbor nodes, and the adapting growth method changes based on environmental conditions, our probabilistic sampling method is based on the learned Q-function and updates with experience in the environment without adjusting any parameters manually.

### 4.2.1 Openset

First, we define the *openset* and describe how it is used for sampling and extending the search tree. The *openset* $S$ is a list of open nodes which have been discovered by exploration of the tree but not yet selected by *softmax node selection*. Once an open node is selected, it is removed from the *openset*, as illustrated in Fig. 4.6. Initially, the *openset* has only the start node and adds new collision-free nodes as they are explored.
Figure 4.6: Softmax node selection chooses a node with probability in the openset. The selected node is removed and the new explored node is added in the openset.

by tree extension. As shown in Fig. 4.6, an open node in the openset is selected by softmax node selection, described in Section 4.2.2, using the expected state value, and that selected node is extended with the Q-function.

The openset has a random node to combine with the exploration property of the RRT, and this random node has the same value as the state value of the start node. If the random node is chosen, the standard RRT extension scheme extends the tree from the nearest node to the given random node.

4.2.2 Softmax node selection

Softmax node selection is a method of choosing an open node based upon the learned value of nodes in the openset for the RRT extension. For the node selection, one of the simplest approaches is $\epsilon$-greedy selection. However, since the number of choices is changed according to the number of open nodes in the openset, $\epsilon$-greedy selection is inappropriate for node selection in this framework. In this chapter, we make a probability distribution using the softmax function based on the expected state values of open nodes in the openset. Since the output of the softmax function represents a categorical distribution over open nodes, as shown in Fig. 4.6, the softmax node selection method chooses a node with probability
Algorithm 2 QS-RRT with learning sampling policy

1: \textit{w}.	exttt{init}()
2: \For{episode =1, max\_episode}{
3: \textit{T}.	exttt{init}(\textit{q}_{init}), \textit{v}_{rand} = \text{ComputeStateValues}(\textit{q}_{init})
4: \While{Distance(\textit{q}_{goal}, \textit{q}_{new}) > d_{min}}{
5: \For{i = 1, m}{
6: \textit{v}_{i} = \text{ComputeStateValues}(\textit{q}_{i})
7: }
8: \textit{k} \gets \text{SoftmaxNodeSelection}(\textit{v}_{1...m}, \textit{v}_{rand})
9: \If{\textit{k} == m + 1}{
10: \textit{q}_{sample} \gets \text{RandomSampling()}
11: }
12: \Else{
13: \textit{s} \gets \text{PopOpenNode}(k)
14: \textit{q}_{sample} \gets \text{ActionOptimization}(s, \textit{w})
15: }
16: \textit{q}_{near} \gets \text{NodeSelection}(\textit{T}, \textit{q}_{sample})
17: \textit{q}_{new} \gets \text{NodeExtension}(\textit{T}, \textit{q}_{sample}, \textit{q}_{near})
18: \textit{b}_{chk} = \text{CollisionCheck}(\textit{q}_{new})
19: \If{\textit{b}_{chk} \text{ is FALSE}}{
20: \textit{T}.\texttt{addTree}(\textit{q}_{new})
21: }
22: \textit{r} \gets \text{ComputeReward}(\textit{q}_{new}, \textit{b}_{chk})
23: \textit{w} \gets \text{UpdateWeights}(\textit{q}_{new}, \textit{w}, \textit{r})
24: }
25: \textit{w} \gets \text{UpdateWeights}(\textit{q}_{new}, \textit{w}, \textit{r}_{final})

\[ P(s) = \frac{\exp(v(s)/\tau)}{\sum_{i=1}^{m+1} \exp(v(s_i)/\tau)} \quad s_i \in S, \]

where \( \tau \) is a temperature parameter which is usually annealed over time for learning, and it is a constant value in the test. The symbol \( m \) is the number of open nodes in the \textit{openset}, \( v(s) \) corresponds to the expected state value in the state \( s \), \( v(s_{m+1}) \) is the state value of the random node, and \( S \) is the \textit{openset}. Note that the state value of the random node is set to the same value as the state value of the start node.

Once one open node is chosen with the probability distribution, we find the action maximizing the Q-function. The maximum state value node still has the highest selection probability, but all other nodes, including the random node, also have selection probability according to their state values. The \textit{softmax node selection} approach has
two advantages as follows. First, the randomness increases in complicated conditions since the number of nodes in the openset decreases due to the high probability of colliding with obstacles. It means that the probability of choosing the random node in the openset (i.e., standard RRT random extension) increases in the softmax distribution. It helps the tree to escape easier from the complicated environment or local minima. For example, when there is no open node except the random node, the RRT can be extended with exploration toward the larger Voronoi regions. Next, when the tree is close to the goal node, its randomness decreases since the state values of the open nodes increase, while the state value of the random node is a constant equal to the state value of the start node. Therefore, the probability of choosing the open node, which has a higher state value, increases, and the probability of choosing the random node decreases relatively.

4.3 QS-RRT

We combine the suggested sampling approach based upon Q-learning with the RRT, which is called “QS-RRT” throughout this chapter. QS-RRT has the same connection strategy from the tree to new nodes as traditional RRT approaches. In addition, since QS-RRT has a probability of choosing random nodes, it will ultimately find a path as the number of sample points reaches infinity even though the learned Q-function becomes worse. Thus, QS-RRT is probabilistically complete [80]. Therefore, the random exploration method helps QS-RRT to find an existing solution and also to compensate for the disadvantage of biased exploration. In addition, QS-RRT can take advantage of the efficient tree extension based on the Q-function. Algorithm 2 presents pseudocode for QS-RRT with the suggested sampling and tree extension methods. When the RRT arrives at the goal state, the reward is 100, and when it collides with an obstacle, the reward is -30 (Line 21). Weights are updated by Equation (4.4), and the step size $\alpha$ is set to 0.13 (Line 22). We analyze the convergence and performance according to the step size in Section 4.5.9.
4.4 Nonholonomic QS-RRT

In the previous section, we suggest an approach based on Q-function approximation with feature-based representation, which incorporates the tree extension based on the Q-function with the random exploration of the RRT. This work handles a holonomic system and assumes that all transitions between nodes are feasible. In the robotics area, there are many nonholonomic systems, such as wheeled mobile robots and unmanned aerial vehicles, which have maneuverability limitations. Since the history of states is needed in order to determine the current state, the original QS-RRT is not directly applied in such nonholonomic systems. Environmental constraints such as collision regions can be implicitly represented in the configuration space, but the differential constraints such as nonholonomic constraints are not easily represented in the configuration space. In addition, because it is difficult to impose differential constraints onto undirected edges between vertices, sampling-based planners are hard to apply to problems with nonholonomic constraints [125]. Although a tree-based planner, such as an RRT, is relatively easier to apply to problems with nonholonomic constraints compared to a roadmap-based planner, it requires a lot of samples and edges to connect between nodes since control with nonholonomic constraints should be imposed at each edge of the tree. Therefore, sampling-based planning in a cluttered environment having obstacles with nonholonomic constraints is difficult since it must plan a trajectory which satisfies nonholonomic constraints without obstacle collisions.

The traditional approaches generate a collision-free path and then uses a path smoothing technique to satisfy nonholonomic constraints. This hierarchical planning is divided into two steps: a global planning for a collision-free path from the start to the goal state without considering differential constraints and a local path planning with differential constraints between two states from the global path planned in the first step, such as path smoothing [64, 44, 3], Clothoids [45, 74], and splines [33]. However, this approach is inefficient, and the local planner might not find a satisfactory path since the differential constraints are not considered in the global path
Another approach for nonholonomic motion planning is a deterministic search method which discretizes control and state space. The control space is discretized into a finite subset, and the control space sampling applies a control input from this set during fixed times [10]. Since this approach samples a control which satisfies the differential constraints, all control inputs are satisfied with feasibility. State space is also discretized into a set of states, and a graph structure represents configurations and connections between these states, which represent feasible paths between states. This state lattice representation can formulate motion planning problems into graph search problems [89]. The weakness of these approaches lies in their resolution lost due to discretization [55] or heavy computation with high discretized resolution. Therefore, it is difficult to define an efficient state space discretization.

In this section, we extend the QS-RRT with nonholonomic constraints and show how it can handle nonholonomic systems by updating the Q-function with a feature of nonholonomic constraints and a new reward function for violation of constraints.
Figure 4.8: Nonholonomic constraints

Figure 4.9: RRT planning with a nonholonomic constraint (Maximum curvature)
4.4.1 Nonholonomic constraint for QS-RRT

RRT dealing with differential constraints tackles the constraints of the system by choosing the allowable input and applying forward simulation using this input. Traditional RRT algorithms satisfy the differential constraints by sampling the space of action trajectories [82]. However, it is difficult to generate a good sequence of control inputs by applying constant control inputs with fixed time. In the nonholonomic QS-RRT, we utilize splines as a local planner which connect two states and parameterizes splines instead of time and input discretization. The local planner connects two points guaranteeing the continuity of curvature and satisfying the upper-bounded curvature constraints. Fig. 4.8 shows the allowable curvature of random points which satisfies the feasibility constraint. The feasibility of constraint defines an allowable region within the maximum curvature angles. The curvature is decided by the angle $\angle ABC$ in Fig. 4.8, and the maximum curvature ($\kappa_{\text{max}}$) is specified based on the kinematic constraints of the vehicle. If the random point is sampled with the allowable boundary of constraints, there is no difference between the RRT and the nonholonomic RRT. However, if the sample exists outside of the feasible region, it will be projected into the allowable region as shown in Fig. 4.8. This approach circumvents the costly computation of feasible region calculation while preserving the exploration property of the RRT. Fig. 4.9 shows the comparison between the original RRT and the nonholonomic RRT with a maximum curvature.

We suggest the nonholonomic QS-RRT satisfying maximum curvature constraints. QS-RRT also has the maximum curvature constraint and penalizes the reward whenever a new sample violates the maximum curvature constraint. Since we can define a spline based on the Bazier curve with heading vectors [139], we additionally define the feature for the nonholonomic constraint as follows:

$$\phi_4(s, a) = \acos\left(\frac{\overrightarrow{s_p s} \cdot \overrightarrow{ss_c}}{\| \overrightarrow{s_p s} \| \| \overrightarrow{ss_c} \|}\right)$$

where $s_c = s + a$, $s_p$ is the parent node of $s$. In addition, when the new sample violates the maximum curvature, we give a negative reward (-10) to penalize for violation.
4.5 Results

We evaluate QS-RRT under various experimental settings using both a simulator and a physical robot system. To validate the performance of QS-RRT, we compare it with several sampling-based planning approaches in 2D navigation and two kinds of manipulations (2DoF planar manipulator and 7DoF manipulator). We learn RBF features by using the samples generated during the standard RRT planning. It takes 3.17 seconds to learn the RBF features with 61,910 samples (0.97 seconds with 10,000 samples) of 7DoF manipulators, and it takes 0.47 seconds with 10,000 samples (0.08 seconds with 3,000 samples) of 2DoF manipulators. Since we accumulate samples during the RRT planning, it is unnecessary to include sampling time. However, if we learn the RBF with samples from scratch, it takes 0.75 seconds to collect 10,000 samples of the 2DoF manipulator, and it takes 28.82 seconds to collect 10,000 samples of the 7DoF manipulator. Note that the time depends on the collision check algorithm and complexity of the environment. Since the RBF learning is independent on start or goal configurations, it is unnecessary to relearn the RBF features whenever the planning repeats with different start and goal configurations. Therefore, we don’t include learning time of RBF in the experiment result. In addition, we demonstrate QS-RRT on a real humanoid robot and provide results under real settings. All experiments are performed on a 2.7GHz PC, and all programs are written in Matlab.

4.5.1 2D Navigation

In a 2D navigation scenario, we randomly choose the start and goal position and train the Q-function with 500 random start and goal points. Fig. 4.10 shows our results. In the 2D scenario, there are several obstacles and two points randomly chosen as the start and goal points. The left column holds the learned state values, and the right column shows the RRT planning result. The blue point is a start point and the red point is goal point. The black lines show the generated tree. We can see that the RRT trees are properly extended to the gradient of the state value. Figs. 4.10(c) and 4.10(d) are more challenging cases for the QS-RRT since the tree must escape local...
minima and avoid obstacles to the goal point. The tree escapes the local minima by the tree extension policy based on softmax node selection. Fig. 4.10(d) shows that the tree has more randomness at first, and the randomness decreases according to how close the tree is to the goal position, since the probability of the random node choice decreases as the tree approaches the goal point.

4.5.2 2DoF planar manipulator

Fig. 4.11 shows QS-RRT results in the 2DoF planar manipulator. We obtain the RBF features as shown in Fig. 4.3(b) by training the weights of RBF networks with 10,000 collision and collision-free exemplars obtained during the standard RRT procedures. Once we train the RBF features, we use them for learning with the Q-function approximation. Since QS-RRT considers collision and collision-free RBF score values as features of the Q-function, the tree is efficiently extended by avoiding obstacles with the softmax node selection method based on the learned state values. Fig. 4.11 shows the planned 2DoF planar manipulator motion in the workspace (Fig. 4.11(a)) and the tree with the learned Q-function in the C-space (Fig. 4.11(b)). These
Figure 4.11: Comparison results with the standard RRT in the 2DoF planar manipulator: (a) manipulation results in the workspace, (b) QS-RRT planning results in the C-space with the learned state values, and (c) standard RRT planning results with collision samples (Magenta). (Best viewed in color)
results are more efficient than those of the standard RRT, as shown in Fig. 4.11(c).

In order to validate the performance of QS-RRT quantitatively, we compare it with several sampling-based approaches in planning tasks with 100 random start and goal configurations. Fig. 4.12 shows quantitative results comparing several approaches \[83, 57, 131\] in terms of the time, the number of collision checks, and the path length efficiency. The number of nearest neighbors \( k = 5 \) and 10% goal bias are given to \( B_k \)RRT, and 10% goal bias is given to Biased-RRT. As in Fig. 4.12, QS-RRT is significantly more efficient than other approaches. \( B_k \)RRT has a slightly shorter length than QS-RRT, but it has heavy computational cost compared to other approaches. Biased-RRT performs well when there is no obstacle between the start and the goal configurations. However, the computational time increases exponentially as the environment gets more complicated. Another notable result is that the number of collision checks in QS-RRT is remarkably decreased. The reason is that the tree expands by following the gradient of the state value. Since the sampling-based planners spend most of the time for collision checks, it is a strong advantage of QS-RRT.

### 4.5.3 Resilience to environment changes

The purpose of this simulation is to show that the learned weights can work when the environmental conditions are perturbed. Figs. 4.13(a) and 4.13(b) show the trajectories and execution times in the original and perturbed environments. Although
Figure 4.13: The performance comparison of QS-RRT in the changed environment: (a) the original environment, (b) the perturbed environment with obstacle changes (blue circles), (c) and (e) show planning results in the original C-space, (d) and (f) show planning results in the perturbed C-space.
the RBF and the weights of the Q-function are learned in the original environment, we can see that QS-RRT (Fig. 4.13(f)) is still very efficient compared to the standard RRT (Fig. 4.13(d)) in the perturbed environment. Note that both standard RRT and QS-RRT have no memory from the initial planning. As shown in Figs. 4.13(d) and 4.13(f), since a narrow passage has disappeared in the perturbed environment, the trajectories in Figs. 4.13(c) and 4.13(e) are infeasible there. Therefore, the planner needs to find a completely different route even when the obstacles are only slightly changed, as shown in Fig. 4.13(b). Since the planning in the perturbed environment is more difficult in this example, compared to the original environment, the planning time increases. However, QS-RRT does not degrade when compared with the planning time of RRT. QS-RRT extends in the same direction as the tree in the unchanged conditions. If the free space is changed and the tree extension is blocked by the obstacle, QS-RRT flexibly extends into another free space, since it has the exploration property with the suggested softmax node selection method. Therefore, we can see that QS-RRT can effectively utilize the biased property based on the Q-value even though the Q-value is not learned perfectly.

For quantitative comparison, we define the slightly changed environment as cases where the positions of obstacles in the workspace are randomly changed, as shown in Figure 4.13(a), and the upper bound of the random change of the positions increases as the change level increases. In this simulation, the random change is the variation of the obstacle position with respect to the maximum size of the workspace. For example,
a 6% random change means that the position of the obstacles is randomly changed within 6% of the maximum workspace size. We define six change levels, and the upper bound of random changes of the obstacles varies from 6% (Level 1) to 36% (Level 6). For accurate evaluation without randomness, we generate ten random workspaces for each level and perform planning in each workspace with 100 different start and goal points, and we average the planning time and the number of collision checks of 1,000 planning executions (10 workspace × 100 planning). Note that start and goal configurations should be different since the feasible start and goal configurations shift according to the environmental change. In addition, since the difficulty level also changes according to environment conditions, we evaluate the performance of QS-RRT with RRT baseline performance in the same workspace and with the same start and goal configurations. Figure 4.14(a) illustrates the average planning time for QS-RRT with respect to the change level, normalized by the planning for the baseline planner (RRT) in the same environments with the same start and goal configurations. Figure 4.14(b) illustrates the average number of collision checks with respect to the change level. QS-RRT in changed environments exploits the trained Q-function and RBF features in the original environment. As the change of environment increases, the performance is slightly degraded; but we can see that QS-RRT has a good performance within 0-18% change. In addition, although the change level is more than 30%, QS-RRT is still faster than RRT since the Q-function has other features, such as goal configurations, and QS-RRT includes the softmax node selection algorithm.

Moreover, since we can also update the RBF with new exemplars, we can use QS-RRT in both the slightly and the continuously changing environments. Next, when the RBF values are totally changed in the greatly changed environment, we need to update RBF features. Fig. 4.15 shows the result in the greatly changed environment, in which case, we update only the RBF features. With the updated RBF features, we can see that the performance is good without updating weights in the Q-function. Since it takes 3.17 seconds to learn the RBF features with 61,910 samples of 7DoF manipulators, we can obtain good performance for greatly changed environments without heavy computation.
4.5.4 Result with nonholonomic constraint

Fig. 4.16 shows the comparison results among RRT, QS-RRT, and nonholonomic QS-RRT. Since QS-RRT extends the tree to avoid obstacles without considering the constraint, we can see that it avoids obstacle with large curvature (second column). The trajectory of QS-RRT follows the middle between obstacles. However, since nonholonomic QS-RRT considers the constraint in the Q-function and the reward function, the trajectory has small curvature while avoiding obstacles (third column) as shown in Fig. 4.16. From this result, we can see that QS-RRT can consider nonholonomic constraints properly.

4.5.5 7DoF manipulator

To evaluate 7DoF manipulation planning for the humanoid robot, we must consider self-collision, joint constraints, and obstacles in the workspace, which presents more complicated conditions. We use 61,910 collision and collision-free exemplars for RBF features, the training method of RBF coefficients is trained by the same method described in Section 4.1.3. For learning the weights of the Q-function, two collision-free configurations are randomly assigned to the start and goal configurations in the environment, as shown in Fig. 4.17(a). Since we reflect the trained Q-function for planning, the planning time decreases as the learning proceeds, as shown in the learning curve in Fig. 4.22. Once the weights of the Q-function are trained, they
Figure 4.16: Comparison results among RRT (the first column), QS-RRT (the second column), and nonholonomic QS-RRT (the third column)
do not require retraining even if the environment is slightly perturbed, since they are robust to slight changes. If the environment is greatly changed, we just need to update the RBF features, as described in Section 4.5.3.

For the quantitative evaluation, we conduct QS-RRT and the bidirectional RRT (RRT-Connect) [78] with 100 randomly assigned start and goal configurations, and we use the average execution time, the number of collision checks, and path length as the evaluation criteria. Since QS-RRT is able to apply a bidirectional method, we evaluate its performance as well. Fig. 4.18(a) shows results comparing standard bidirectional RRT, QS-RRT, and bidirectional QS-RRT. Our approaches are significantly more efficient than the standard bidirectional RRT. The number of the collision checks in QS-RRT is dramatically reduced in comparison with traditional RRT.

### 4.5.6 Experimental result in the real environment

As shown in Fig. 4.18(b), we conduct a series of manipulation tasks with a humanoid robot whose task is to insert its arms inside a shelf placed atop a table. In these experiments, we evaluate the performance of the proposed algorithm. The robot performs eight sequential target configurations around the shelf on the table. The robot is initially positioned 30 cm from the table, allowing it little space to insert its
(a) The comparison results in the simulation of Fig. 4.17(a)

(b) The comparison results in the real robot of Fig. 4.17(b)

Figure 4.18: Comparison results with bidirectional RRT, QS-RRT, and bidirectional QS-RRT: (a) simulation results with 100 randomly assigned start and goal configurations and (b) test results in the physical robot system with eight sequential configurations.
arms (length: 85cm) into the shelf (35cm × 37cm) while avoiding collisions with the table. We apply the bidirectional approaches in this experiment. For QS-RRT, we use the learned Q-function in the simulation. Fig. 4.18(b) shows that QS-RRT is six times faster and has ten times fewer nodes in comparison with the bidirectional RRT.

4.5.7 Analysis of features

If we use only the distance from the state to the goal or RBF features alone in the Q-value estimation, we would be unable to obtain the desired result, but since we obtain the feature weights by Q-learning, we can achieve the desired outcome if we use all features for the Q-function. Fig. 4.19(a) shows the results of the original RRT, and Figure 4.19(b) shows the results of QS-RRT. QS-RRT has less collision and tree extension compared to the original RRT. When we only use the distance from the
state to the goal as the feature in the Q-value, we obtain the result as shown in Fig. 4.19(c). It has many collision points since it is biased to the goal and tries to extend in the direction of the shortest distance without considering obstacles. In contrast, when we only use the RBF feature in the Q-value, we obtain the result as shown in Fig. 4.19(d). The tree has less collision compared to the original RRT, but it has a tendency to extend far from the obstacles, which makes it arrive at the goal point slowly.

4.5.8 Comparison with navigation functions

A potential function is a combined function of attraction to the goal with repulsion from obstacles. Using the potential function, the trajectory is produced without heavy computation. However, it can become trapped in local minima and fail to find a path. Barraquand and Latombe suggest a Monte-Carlo algorithm to escape from the local minima of the potential field [9]. However, their method requires adjusting many parameters and takes a long time with many local minima. In addition, it requires expensive computation to determine the cost of actions at every step in continuous action spaces.

In order to resolve local minima, Rimon and Koditschek [113] suggest a navigation function which has a unique minimum at the goal point. It assumes that obstacles are disjointed from each other, and the repulsive function is a multiplication of $\beta_i(q)$ which is positive in free space and negative inside an obstacle. The attractive function is defined as $\gamma_\kappa(q) = (d(q,q_{goal}))^{2\kappa}$. Note that the navigation function can have local minima if $\kappa$ is small and the quality of the generated trajectory depends on $\kappa$. In the navigation function, we can find a trajectory to the goal by following the gradient of $\frac{\gamma_\kappa}{\beta}(q)$. When $\beta$ is close to zero, $\frac{\gamma_\kappa}{\beta}(q)$ could be very large. Therefore, [113] introduces analytic switches, $\sigma_\lambda(x) = \frac{x}{\lambda + x}$ and defines $s(q,\lambda)$ as follows:

$$s(q,\lambda) = (\sigma_\lambda \circ \frac{\gamma_\kappa}{\beta})(q).$$

However, $s(q,\lambda)$ could not become a Morse function with this composition. [113]
introduce a sharpening function $\xi_{\kappa}(x) = x^{\frac{1}{\kappa}}$, and the final navigation function is as follows:

$$\psi(q) = (\xi_{\kappa} \circ \sigma_{\lambda} \circ \frac{\gamma_{\kappa}}{\beta})(q) = \frac{d(q, q_{\text{goal}})}{[d(q, q_{\text{goal}})^{2\kappa} + \beta(q)]^{\frac{1}{\kappa}}}.$$ 

Basically, both the potential and the navigation functions require measurement of the distances to obstacles. Therefore, it is difficult to apply them in the C-space since the distance to the collision regions in the C-space cannot be measured directly.

Moreover, the navigation function has undesirable local minima in an arbitrary environment although it has a nice formulation with spherical obstacles. In order to generalize the navigation function, [113] shows transformation from star shaped obstacles to spherical obstacles using diffeomorphism, but it remains difficult to apply to any arbitrarily shaped obstacles. Paternain et al. [105] upgrade navigation functions to resolve this issue with a curvature condition for obstacles. However, this approach requires the assumptions of strong convexity for objective function and no intersection between obstacles. Therefore, the application areas are still limited in practice.

In contrast to navigation functions, we apply novel RBF features to predict proximity to the collision boundary directly in the C-space. We can replace $\beta(q)$ with the weighted sum of the activation values (i.e., score values) to measure the proximity to the collision region as shown in Fig. 4.20(b). Since RBF score values have similar properties as $\beta(q)$, we can easily establish the navigation function without the heavy computation.

In addition, QS-RRT minimizes the tree extension and escapes local minima efficiently by using a probabilistic node selection procedure based upon the learned value of the node in the C-space. QS-RRT is a novel planning framework that combines the fast random exploration of RRT with tree extensions based on Q-learning. In order to escape undesirable local minima of navigation functions, we can apply the softmax node selection method which is suggested with QS-RRT.

Fig. 4.21 shows the comparison results between navigation function and QS-RRT.
The navigation function with RBF representation has good representation in the C-space, as shown in Fig. 4.20(c). In addition, our softmax node selection and extension approach can combine with the navigation function for efficient RRT planning with good performance. However, the navigation function has a steep gradient when the goal state is around the collision boundary, and it requires tuning of $\kappa$ while QS-RRT learns parameters. QS-RRT has better performance than the planning with navigation functions and our softmax node selection method.

4.5.9 Learning curve

We evaluate the performance of QS-RRT with the learned weights of the Q-function in every episode. QS-RRT is performed with 30 different start and goal configurations of the 7DoF manipulator for evaluation. Fig. 4.22 shows the learning curve based on the average time of QS-RRT compared to the average time of RRT. The convergence of the training depends on the learning step size. As the step size increases, the convergence is faster, but if it is too large, such as 0.7, the performance has some unstable fluctuation. If the step size is 0.13, it converges stably after 400 episodes. Compared to the RRT, QS-RRT has better performance even at the beginning when the learning is not yet finished.
Figure 4.21: Comparison results: odd rows show navigation function results with RBF feature representation and soft node selection approach, and even rows show QS-RRT results.
4.6 Discussion

This chapter has presented a learning approach for a fast sampling-based motion planning algorithm in a high-dimensional C-space. The proposed approach is based on Q-function approximation with a feature-based representation, which incorporates the tree extension based on the Q-function with the random exploration of an RRT. RBF features are used to represent collision and collision-free regions in the C-space, and a softmax node selection approach preserves the asymptotic completeness guarantee of the RRT. In addition, we show how QS-RRT can handle nonholonomic systems by combining the Q-function with a feature of nonholonomic constraint and a reward function for violation of constraints. We demonstrated that the proposed QS-RRT planner displays significant performance improvement over conventional approaches in several example scenarios, including planning with a 7DoF manipulator.
Chapter 5

Robust Tree Extension with Safety Margin of Probabilistic Model

In this chapter, we introduce a new probabilistically safe local steering primitive for sampling-based motion planning in complex high-dimensional configuration spaces (C-spaces). Due to its simplicity and flexibility in handling a diverse set of C-spaces without requiring an explicit representation, sampling-based motion planning is the mainstream approach to global motion planning for high-dimensional, highly nonlinear robotic systems, such as robot manipulators [70, 83, 56, 69]. However, the performance of such randomized motion planners strongly depends on the choice of distance measure, sampling method, and local steering; and it is known to degrade significantly around complicated regions of C-spaces, such as narrow passages [57, 91].

This performance degradation is usually considered as a sampling issue, because uniform sampling has a Voronoi bias towards larger regions of yet unexplored C-spaces. Accordingly, many heuristic rejection sampling approaches and retraction methods are suggested to mitigate this issue, but retraction methods often require a distance-to-collision measure [116, 146]. In contrast, assuming that this performance decay is due to the lack of effective local steering, in [6] a geometric local steering policy that can “feel” the local geometry of C-spaces is proposed for efficient planning around narrow passages; however, its computation also requires a distance-to-collision measure. Since the exact computation of distance-to-collision in complex
Figure 5.1: (left) Probabilistically safe corridor in 3D space constructed around a sample configuration (red) by using tangent hyperplanes (gray) of confidence ellipsoids of a learned GMM of C-space obstacles. (right) Local steering via probabilistically safe corridor in 2D space: An RRT is extended along the safe direction (red dotted line) towards the projection of a sample goal (red) onto the associated probabilistically safe corridor (red polygon), instead of the standard straight-line extension (blue dotted line) towards the sample goal.

In high-dimensional C-spaces is difficult [38], Gaussian mixture learning [58] and locally weighted regression [23] are applied to construct approximate probabilistic models of collision and collision-free subspaces of C-spaces for fast collision checking and biased sampling over free space and difficult regions of C-spaces. In particular, simultaneous modeling of collision and free subspaces is shown to be critical for local planning around narrow passages [37]. In this chapter, by combining the strengths of [6] and [58], we introduce a new notion of probabilistically safe corridors for probabilistically safe guided local steering for sampling-based planning without requiring an explicit computation of distance-to-collision.

More precisely, we construct a probabilistically safe corridor around a configuration using tangent hyperplanes of confidence regions of learned Gaussian mixtures that separate the input configuration from the confidence ellipsoids, as illustrated in Fig. 5.1 (left). Accordingly, we propose a probabilistically safe local steering primitive towards a sample goal configuration via its projection onto the probabilistically safe corridor, as shown in Fig. 5.1 (right). Since the proposed steering method exploits the local geometry of C-spaces via learned Gaussian mixture models (GMMs) and generates steering motion within probabilistically safe corridors, in our numerical
simulation and experiments, we observe that it yields a better exploration of C-spaces while minimizing collision likelihood.

In summary, the main contributions of the chapter include:

i) A novel geometric approximation of C-space obstacles by confidence ellipsoids of learned GMMs,

ii) A new construction of probabilistically safe corridors using tangent hyperplanes of confidence ellipsoids,

iii) An effective probabilistically safe local steering primitive that can minimize collision likelihood.

Using numerical simulations and real experiments, we demonstrate that the proposed probabilistically safe local steering approach can dramatically improve the performance of randomized motion planners around narrow passages and significantly outperform the straight-line local planner in high dimensional C-spaces by decreasing the number of collisions.

5.1 Probabilistically Safe Corridors

In this section, we first present a brief overview of how learning of Gaussian mixtures\(^3\) can be used for approximate probabilistic modeling of C-spaces and then introduce the new notion of a probabilistically safe corridor around a configuration that identifies a safe neighborhood of the configuration with minimal collision risk. Accordingly, we propose a practical extension\(^4\) of the standard RRT planner, called Safety-Guided RRT (SG-RRT), where tree extension is guided to ensure safety constraints defined by probabilistically safe corridors.

\(^3\)Although other probabilistic (mixture) models can be used for approximating \(\mathcal{F}\) and \(\mathcal{O}\), we find it convenient to use Gaussian mixtures since their confidence regions can be accurately and efficiently approximated using confidence regions of individual Gaussians which have an ellipsoidal form.

\(^4\)Safety guided steering via probabilistically safe corridors can be integrated with any (sampling-based) motion planning algorithm (e.g., probabilistic roadmaps–PRMs) as a local steering primitive, especially for uncertainty-aware belief-space planning.
5.1.1 Gaussian Mixture Modeling of C-spaces

Let $C$ denote the C-space of a robotic system embedded in an $n$-dimensional Euclidean space $\mathbb{R}^n$ and denote by $F \subset C$ and $O \subset C$, respectively, the free subspace and the collision subspace (i.e., obstacles) of the C-space $C$, which, by definition, satisfy $F = C \setminus O$. In general, an explicit representation of the free space $F$ or the collision space $O$ in terms of simple geometric shapes is known to be very difficult to obtain, especially for high-dimensional complex systems, such as robotic manipulators. Hence, as in [58], we consider approximate probabilistic representations of the free space $F$ and the collision space $O$ in terms of Gaussian mixtures models, respectively denoted by $\mathcal{G}M(\mu_F, \Sigma_F, \omega_F)$ and $\mathcal{G}M(\mu_O, \Sigma_O, \omega_O)$, that are constructed using collision and collision-free sample configurations as described below. Here, a Gaussian mixture distribution $\mathcal{G}M(\mu, \Sigma, \omega)$, consisting of $K \in \mathbb{N}$ mixture components, is parametrized by a list of mixture means $\mu := (\mu_1, \mu_2, \ldots, \mu_K) \in (\mathbb{R}^n)^K$, a list of positive-definite covariance matrices $\Sigma := (\Sigma_1, \Sigma_2, \ldots, \Sigma_K) \in (\mathbb{R}^{n \times n})^K$, and a list of normalized mixture weights $\omega := (\omega_1, \omega_2, \ldots, \omega_K) \in (\mathbb{R}_{\geq 0})^K$, satisfying $\sum_{k=1}^K \omega_k = 1$, and its value at a point $x \in \mathbb{R}^n$ is given by

$$\mathcal{G}M(x; \mu, \Sigma, \omega) := \sum_{k=1}^K \omega_k N(x; \mu_k, \Sigma_k),$$  \hspace{1cm} (5.1)

where $N(x; \mu, \Sigma)$ is the multivariate Gaussian distribution with mean $\mu$ and covariance matrix $\Sigma$,

$$N(x; \mu, \Sigma) := \frac{1}{\det(2\pi \Sigma)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right).$$  \hspace{1cm} (5.2)

Note that the numbers of mixtures, $K_F$ and $K_O$, used for modeling the free space $F$ and the collision space $O$ can be different. In particular, the Meanshift clustering algorithm used in this chapter automatically determines the number of mixture components using sample configurations based on a geometric bandwidth parameter as described below. It is also important to highlight that one can simply use $\mathcal{G}M(x, \mu_F, \Sigma_F, \omega_F)$ and $\mathcal{G}M(x, \mu_O, \Sigma_O, \omega_O)$ to estimate how likely a configuration
Figure 5.2: Examples of learned GMMs. Ellipsoids show the confidence regions associated with the confidence level of $\kappa = 0.9$. (left) Gaussian mixtures in the 3D workspace shown in Fig. 5.9, (right) Gaussian mixtures in the C-space of a 2DoF planar manipulator.

is in collision, which is leveraged in [58] for fast collision checking and biased sampling. In addition to such demonstrated potential improvements, we shall show below that confidence regions of these GMMs can be utilized for understanding the local geometry of the C-space $C$ and for increasing the quality of the local steering heuristic (which is the Euclidean distance in our case) to better approximate the true geodesic (cost-to-go) metric of the C-space $C$.

**Learning Gaussian Mixtures**

One can use a number of Expectation-Maximization (EM) variant methods for Gaussian mixture learning for modeling the free space $F$ and the collision space $O$ using collision and collision-free sample configurations in an offline or online manner, as in Chapter 3. In this chapter, we apply the Meanshift clustering method [27] with a Gaussian kernel for learning Gaussian mixtures using collision information of sample configurations obtained during previous attempts of a randomized motion planner, which is a convenient way of learning from past experiences and exploiting the collision history. In addition, this approach resolves the problem that general mixture modeling approaches have no explicit way of determining the required number of mix-
tures, because the Meanshift clustering requires a kernel bandwidth $B$ instead of the number of clusters $K$. The kernel bandwidth $B$ can be set based on the desired level of spatial resolution. With the bandwidth $B$, we initialize the clusters and then perform a single step EM update to estimate cluster statistics. We set the membership weight value as $z^i_k = 1$ if the $i$th point in $N$ samples is included in the $k$th cluster, and $z^i_k = 0$ otherwise. Then, the cluster statistics (mass $m_k$, mean $\mu_k$, covariance matrix $\Sigma_k$, and weight $\omega_k$) for the $k$th cluster are given by

$$m_k = \sum_{i=1}^{N} z^i_k, \quad \mu_k = \frac{1}{m_k} \sum_{i=1}^{N} z^i_k x_i, \quad \omega_k = \frac{m_k}{\sum_{j=1}^{K} m_j},$$

$$\Sigma_k = \frac{1}{m_k} \sum_{i=1}^{N} z^i_k (x_i - \mu_k)(x_i - \mu_k)^T, \quad \text{for } k \in \{1, \ldots, K\}.$$

In Fig. 5.2, we present some examples of probabilistic models of different C-spaces and workspaces constructed by the suggested approach. Fig. 5.2 (left) shows a probabilistic model to define the collision space from 3D point clouds obtained by a depth sensor. Fig. 5.2 (right) shows the generated probabilistic models using collision information of samples in the C-space of a 2DoF planar manipulator. Such probabilistic representations of C-spaces can be utilized for collision likelihood estimation as a computationally efficient alternative to the exact distance-to-collision measurement [58].

Confidence regions of Gaussian Mixtures

While a Gaussian mixture model $\mathcal{GM}(\mu_\mathcal{F}, \Sigma_\mathcal{F}, \omega_\mathcal{F})$ of the free space $\mathcal{F}$ can be used to bias sampling over the free space, in addition to its use in fast collision checking [58], we propose a new novel use for the confidence regions of a Gaussian mixture model $\mathcal{GM}(\mu_\mathcal{O}, \Sigma_\mathcal{O}, \omega_\mathcal{O})$ of the collision space $\mathcal{O}$ for understanding the local geometry of the C-space $\mathcal{C}$, which is the main contribution of the present chapter.

Definition 1. The confidence region $\mathcal{C}_p(\kappa)$ of a continuous probability distribution $p : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ associated with a confidence level $\kappa \in [0, 1]$ is defined to be the super level set $\mathcal{L}_p(\tau) := \{ x \in \mathbb{R}^n | p(x) \geq \tau \}$ of $p$, for some $\tau \in \mathbb{R}_{\geq 0}$, over which the cumulative
mass distribution of \( p \) is \( \kappa \), i.e.,

\[
C_p(\kappa) = \mathcal{L}_p(\tau) \quad \text{such that} \quad \int_{\mathcal{L}_p(\tau)} p(x) \, dx = \kappa. \tag{5.3}
\]

Hence, it is convenient to have \( L_p(\kappa) \) denote the level function of \( p \) that returns the corresponding level of \( p \) defining the confidence region \( C_p(\kappa) \), i.e.,

\[
C_p(\kappa) = \mathcal{L}_p(L_p(\kappa)). \tag{5.4}
\]

Although confidence regions of an arbitrary probability distribution cannot be expressed explicitly in terms of simple geometric shapes and so are needed to be computed numerically \[62\], confidence regions of Gaussian distributions have an analytical ellipsoidal form.

**Remark 1.** For any confidence level \( \kappa \in [0, 1] \), the ellipsoidal confidence region \( C_{N(\mu, \Sigma)}(\kappa) \) and the level function \( L_{N(\mu, \Sigma)}(\kappa) \) of the Gaussian distribution \( N(x; \mu, \Sigma) \) are, respectively, given by

\[
C_{N(\mu, \Sigma)}(\kappa) = \left\{ x \in \mathbb{R}^n \mid (x - \mu)^T \Sigma^{-1} (x - \mu) \leq F^{-1}_{\chi^2_n}(\kappa) \right\}, \tag{5.5}
\]

\[
L_{N(\mu, \Sigma)}(\kappa) = \frac{1}{\det(2\pi \Sigma)^{\frac{1}{2}}} \exp \left( -\frac{1}{2} F^{-1}_{\chi^2_n}(\kappa) \right), \tag{5.6}
\]

where \( F_{\chi^2_n} : \mathbb{R}_{\geq 0} \rightarrow [0, 1] \) denotes the cumulative probability distribution of \( \chi^2_n \) distribution with \( n \) degrees of freedom. Hence, for any \( \tau \in \mathbb{R}_{\geq 0} \), the confidence level \( \kappa \) of the super level set \( \mathcal{L}_{N(\mu, \Sigma)}(\tau) \) of the Gaussian distribution \( N(\mu, \Sigma) \) is explicitly given by

\[
\kappa = L_{N(\mu, \Sigma)}^{-1}(\tau) = F_{\chi^2_n} \left( -\log(\tau^2 \det(2\pi \Sigma)) \right). \tag{5.7}
\]

Accordingly, since it lacks an exact closed-form expression, we suggest approximating the confidence region of a Gaussian mixture distribution \( \mathcal{G}M(\mu, \Sigma, \omega) \) associated with a confidence level \( \kappa \in [0, 1] \) as a union of ellipsoidal confidence regions of individual Gaussians, associated with confidence levels \( \kappa := (\kappa_1, \kappa_2, \ldots, \kappa_K) \) that satisfy
Figure 5.3: GMM confidence regions. (a) Super level sets of individual Gaussians at confidence level $\kappa_k = \kappa$. (b) Super level sets of Gaussians at the confidence levels corresponding to a shared probability level. (c) An example C-space (collisions are in blue and free space is in red) and (d) the associated confidence ellipsoids of learned GMM distributions from collision samples (black in (c)).

$$\sum_{k=1}^{K} \omega_k \kappa_k = \kappa,$$ as

$$\overline{C_{G,M}(\mu, \Sigma, \omega)}(\kappa) := \bigcup_{k=1}^{K} C_{N(\mu_k, \Sigma_k)}(\kappa_k),$$

$$= \bigcup_{k=1}^{K} \left\{ x \in \mathbb{R}^n \mid (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \leq F^{-1}_{\chi^2_n}(\kappa) \right\},$$

Observe that, by construction, we have

$$\int_{\overline{C_{G,M}(\mu, \Sigma, \omega)}(\kappa)} G_{M}(x; \mu, \Sigma, \omega) \, dx \geq \kappa.$$  \hfill (5.10)

A standard choice of the confidence levels of individual Gaussians is $\kappa_k = \kappa$ for all
positive-definite covariance matrix \( \Sigma \) as shown in Fig. 5.3 (a); however, this usually yields a poor approximation of the actual confidence region of the mixture model because less accurate Gaussians with high variances become more influential in determining the confidence region. A more accurate analytical choice for the individual confidence levels is \( \kappa_k = L_{\mathcal{N}(\mu_k, \Sigma)}^{-1}(\frac{\tau}{\omega_k}) \) based on a shared probability level \( \tau = \sum_{k=1}^{K} \omega_k^2 L_{\mathcal{N}(\mu_k, \Sigma)}(\kappa) \) \([5]\). Alternatively, in this chapter, we use an iterative search algorithm to find a more accurate shared probability level \( \tau \) as described in \([5]\) and thus set \( \kappa_k = L_{\mathcal{N}(\mu_k, \Sigma)}^{-1}(\frac{\tau}{\omega_k}) \) for all \( k \), as shown in Fig. 5.3 (b). With this approach, we obtain confidence regions of GMMs that approximately represent C-space obstacles, as illustrated in Fig. 5.3 (c)-(d).

### 5.1.2 Probabilistically safe corridors

Suppose \( \mathcal{G}\mathcal{M}(\mu_\mathcal{O}, \Sigma_\mathcal{O}, \omega_\mathcal{O}) \) to be a GMM constructed as described above for modeling the collision subspace \( \mathcal{O} \) of a C-space in \( \mathbb{R}^n \) and let \( \mathcal{C}_{\mathcal{G}\mathcal{M}(\mu_\mathcal{O}, \Sigma_\mathcal{O}, \omega_\mathcal{O})}(\kappa_\mathcal{O}) \) be the corresponding approximate confidence region associated with a desired confidence level \( \kappa = \sum_{k=1}^{K} \omega_\mathcal{O}_k \kappa_\mathcal{O}_k \). Accordingly, we define the probabilistically safe corridor around a configuration \( p \in \mathbb{R}^n \) to be

\[
\mathcal{S}\mathcal{C}_\mathcal{O}(p) := \left\{ x \in \mathbb{R}^n \mid \frac{1}{\sum_{k=1}^{K} \omega_\mathcal{O}_k} \sum_{k=1}^{K} \omega_\mathcal{O}_k (x-p) \geq \min \left( \frac{F_{\chi^2_n}^{-1}(\kappa_{\mathcal{O}_k})}{\left\| \frac{1}{\sum_{k=1}^{K} \omega_\mathcal{O}_k} \right\|}, 1 - \epsilon \right), \forall k \right\},
\]

(5.11)

\[
\mathcal{S}\mathcal{C}_\mathcal{O}(p) := \left\{ x \in \mathbb{R}^n \mid \frac{1}{\sum_{k=1}^{K} \omega_\mathcal{O}_k} \sum_{k=1}^{K} \omega_\mathcal{O}_k (x-p) \leq \max \left( \frac{F_{\chi^2_n}^{-1}(\kappa_{\mathcal{O}_k})}{\left\| \frac{1}{\sum_{k=1}^{K} \omega_\mathcal{O}_k} \right\|}, \epsilon \right), \forall k \right\},
\]

(5.12)

which is constructed using tangent hyperplanes of confidence ellipsoids of Gaussians and is a closed convex polytope, as depicted in Fig. 5.4. Here, \( \epsilon \in \mathbb{R} \) is a scalar safety tolerance parameter, and \( \| \cdot \| \) denotes the standard Euclidean norm, and for any positive-definite covariance matrix \( \Sigma \in \mathbb{R}^{n \times n} \), a positive-definite choice of \( \Sigma^{-\frac{1}{2}} = V \left( \frac{1}{\sqrt{\sigma_1}}, \frac{1}{\sqrt{\sigma_2}}, \ldots, \frac{1}{\sqrt{\sigma_n}} \right) V^T \) where \( \Sigma = V \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n) V^T \) is the singular-value decomposition of \( \Sigma \). It is also useful to observe from (5.5) that \( F_{\chi^2_n}^{-1}(\kappa_{\mathcal{O}_k}) = \left\| \Sigma_{\mathcal{O}_k}(\mu_{\mathcal{O}_k}-p) \right\|^2 \) for any confidence region boundary point \( p \in \partial \mathcal{C}_\mathcal{N}(\mu_{\mathcal{O}_k}, \Sigma_{\mathcal{O}_k})(\kappa_{\mathcal{O}_k}) \). Hence, the safety constraints encoded by \( \mathcal{S}\mathcal{C}_\mathcal{O} \) are relaxed with increasing \( \epsilon \).

**Proposition 1.** For \( \epsilon \geq 0 \), the probabilistically safe corridor \( \mathcal{S}\mathcal{C}_\mathcal{O}(p) \) of a configuration
Figure 5.4: Local steering via probabilistically safe corridors. (left) Example tree extension using a probabilistically safe corridor in 2D space, (right) Probabilistically safe corridor in 3D space.

\( p \in \mathbb{R}^n \) is a nonempty convex neighborhood of \( p \); and for \( \epsilon > 0 \), \( SC_O(p) \) strictly contains \( p \) in its interior \( SC_O(p) \), i.e., for any \( p \in \mathbb{R}^n \)

\[
  p \in SC_O(p) \quad \forall \epsilon \geq 0, \text{ and } p \in \tilde{SC}_O(p) \quad \forall \epsilon > 0.
\]  

(5.13)

**Proof.** By definition (5.12), the probabilistically safe corridor \( SC_O(p) \) is constructed as an intersection of half-spaces and so is a convex polytope. Moreover, for any \( \epsilon \geq 0 \) (resp. \( \epsilon > 0 \)), these half-spaces are guaranteed to contain \( p \) (resp. strictly in their interiors). Thus, the result follows.

**Proposition 2.** For \( \epsilon \leq 0 \), the probabilistically safe corridor \( SC_O(p) \) of a probabilistically safe state \( p \in \mathbb{R}^n \setminus \tilde{C}_{\gamma,M} (\mu_O, \Sigma_O, \omega_O) (\kappa_O) \) contains \( p \) in its interior \( \tilde{SC}_O(p) \) and is also probabilistically safe, i.e.,

\[
  p \in \mathbb{R}^n \setminus \tilde{C}_{\gamma,M} (\mu_O, \Sigma_O, \omega_O) (\kappa_O) \quad \implies \quad p \in \tilde{SC}_O(p) \subset \mathbb{R}^n \setminus \tilde{C}_{\gamma,M} (\mu_O, \Sigma_O, \omega_O) (\kappa_O).
\]  

(5.14)

**Proof.** For any \( p \in \mathbb{R}^n \setminus \tilde{C}_{\gamma,M} (\mu_O, \Sigma_O, \omega_O) (\kappa_O) \), we have from (5.5) that

\[
  \frac{\sqrt{F_{\gamma,k}^{-1}(\kappa_O)}}{\tilde{F}_{\gamma}^2(p-\mu_O)} < 1 \leq 1-\epsilon \quad \text{for all } k.
\]

Hence, the result directly follows from (5.12) and the fact that
for any safe configuration \( p \in \mathbb{R}^n \setminus \overline{\mathcal{C}_{G,M}(\mu_O, \Sigma_O, \omega_O)}(\kappa_O) \), the probabilistically safe corridor \( \mathcal{SC}(p; \mu_O, \Sigma_O, \omega_O) \) is bounded by tangent hyperplanes of confidence regions of individual Gaussians that strictly separates the point \( p \) from the Gaussian confidence ellipsoids.

Note that the safe corridor \( \mathcal{SC}_O(p) \) around a probabilistically unsafe configuration \( p \in \overline{\mathcal{C}_{G,M}(\mu_O, \Sigma_O, \omega_O)}(\kappa_O) \) can be empty for \( \epsilon < 0 \), especially for GMMs with significant overlap. Fortunately, many Gaussian mixture learning algorithms yield proper mixture models with minimal overlap. Moreover, in order to resolve this issue, one can consider using a nonnegative \( \epsilon \), which adaptively relaxes the safety constraints of \( \mathcal{SC}_O(p) \) depending on the safety level of the configuration \( p \) and yields a nonempty relatively safe corridor \( \mathcal{SC}_O(p) \). Thus, an optimal selection of \( \epsilon \) is \( \epsilon = 0 \), which ensures nonempty safe corridors for all configurations (Proposition 1) and exact probabilistically safe corridors for probabilistically safe configurations (Proposition 2).

### 5.1.3 Guided steering via safe corridors

We now describe a novel use of probabilistically safe corridors for guided local steering of sampling-based planning, in particular, RRTs. In the original RRTs, a sample configuration \( q_{\text{rand}} \) is randomly drawn in the C-space, and then its nearest node \( q_{\text{near}} \) in the tree is found based on a distance measure, which is set to be the standard Euclidean distance in this chapter. Then, a new configuration \( q_{\text{new}} \) is slightly extended from \( q_{\text{near}} \) towards \( q_{\text{rand}} \), say using the standard straight-line steering. If \( q_{\text{new}} \) is collision-free, it is added to the tree as a new node, which is connected to the nearest node. If \( q_{\text{new}} \) collides with an obstacle, then tree construction repeats with another \( q_{\text{rand}} \).

In this chapter, we propose a new approach for tree expansion where \( q_{\text{new}} \) is adjusted to head towards collision-free space using probabilistically safe corridors \( \mathcal{SC}_O \), as shown in Fig. 5.4, by projecting \( q_{\text{rand}} \) onto \( \mathcal{SC}_O(q_{\text{near}}) \) as follows:

\[
q_{\text{proj}} = \Pi_{\mathcal{SC}_O(q_{\text{near}})}(q_{\text{rand}})
\]  

(5.15)
Algorithm 3 Tree Extension in C-Space

Require: \( \mu_O, \Sigma_O \)

1: \( \mathcal{T}.\text{init}(q_{init}) \);
2: \textbf{while} Distance\( (q_{goal}, q_{new}) > d_{min} \) \textbf{do}
3: \( q_{rand} \leftarrow \text{GetRandomSampling()}, \ iter = 0; \)
4: \textbf{while} \( \text{iter} < \text{max_iter} \) \textbf{do}
5: \( q_{near} \leftarrow \text{GetNearestNeighbor}(\mathcal{T}, q_{rand}); \)
6: \( q_{proj} \leftarrow \text{SteeringGuide}(\mu_O, \Sigma_O, q_{near}, q_{rand}); \)
7: \( q_{adj} \leftarrow \text{StraightLineSteering}(q_{near}, q_{proj}, \delta); \)
8: \textbf{if} StraightLine\( (q_{near}, q_{adj}) \) is Collision-Free \textbf{then}
9: \( \mathcal{T}.\text{addTree}(q_{adj}), \ \text{iter} = \text{iter} + 1; \)
10: \textbf{else}
11: \( \text{break;} \)
12: \textbf{end if}
13: \textbf{end while}
14: \textbf{end while}

where \( \Pi_A(x) := \arg \min_{a \in A} \|x - a\| \) is the metric projection of a point \( x \in \mathbb{R}^n \) onto a closed convex set \( A \subseteq \mathbb{R}^n \); that is, \( \Pi_A(x) \) returns the closest point of set \( A \) to the input point \( x \). Hence, the tree is extended towards \( q_{proj} \) instead of \( q_{rand} \), as shown in Fig. 5.4.

Proposition 3. If a sampling-based motion planning algorithm is probabilistically complete for the standard straight-line steering, then the straight-line steering towards the projected goal onto probabilistically safe corridors, as described in (5.15), preserves its probabilistic completeness for \( \epsilon > 0 \).

Proof. The result simply follows from Proposition 1 because the probabilistically safe corridor \( SC_O(p) \) of a configuration \( p \in \mathbb{R}^n \) strictly contains \( p \) in its interior for \( \epsilon > 0 \), and the metric projection onto a probabilistically safe corridor locally behaves as the identity map. In other words, for \( \epsilon > 0 \), the straight-line steering towards the projected goal onto probabilistically safe corridors is locally equivalent to the standard unconstrained straight-line steering.

\[ \square \]

One computational challenge of our guided steering approach is that it requires recomputation of the metric projection of \( q_{rand} \) onto \( SC_O(q_{near}) \) for each new selection of \( q_{rand} \) and so \( q_{near} \). Metric projection onto a convex polytope can be solved using any
state-of-the-art quadratic optimization solver. For efficiency, we apply the active-set method for quadratic optimization, which is an iterative solver that ensures a feasible solution and a decrement on the objective function at each iteration. This enables us to inherit some useful information from prior computation and stop its computation after the desired number of iterations. In order to reduce computational cost, we keep \( q_{\text{rand}} \) the same until a maximum number of iterations \( \text{max}_\text{iter} \) is reached. This enables us to warm-start the active set method with the active constraints of the previous computation. If active constraints at the optimal solution are given, then a quadratic optimization problem with inequality constraints can be converted into a quadratic problem with equality constraints, which requires significantly less computational time to solve. For example, previous active constraints could be still active for slightly changed \( q_{\text{near}} \) if the sample goal \( q_{\text{rand}} \) is kept the same. Therefore, to increase computational efficiency, we always check first if the quadratic optimization is feasible with previously active hyperplane constraints of probabilistically safe corridors.

**Tree extension in the C-space**

Algorithm 3 presents the pseudocode for the proposed tree extension methods in the C-space. Here, the nearest node \( q_{\text{near}} \) of a random goal \( q_{\text{rand}} \) in tree \( T \) is extended by a new node \( q_{\text{adj}} \) towards the projected goal \( q_{\text{proj}} \) through the probabilistically safe corridor \( SC_O \) of \( q_{\text{near}} \). If the random goal \( q_{\text{rand}} \) satisfies the safety corridor constraints, then the tree is directly extended to the random goal, just like the standard straight-line extension method. In our implementation, we set the maximum number of iterations, \( \text{max}_\text{iter} \) (Line 4), for using the same random goal \( q_{\text{rand}} \) to be 3, and we select the maximum stepsize of the straight-line planner, \( \delta \) (Line 7), manually depending on the desired accuracy level of collision checks.

**Tree extension in the task space**

For task space planning, we also use probabilistically safe corridors for guiding the end-effector of a manipulator as described in Algorithm 4. Using forward kinematics,
Algorithm 4 Tree Extension in Task Space

Require: \( \mu, \Sigma \)

1: \( T.init(e_{init}, q_{init}) \);
2: while Distance\( (q_{goal}, q_{new}) > d_{min} \) do
3:   \( q_{rand} \leftarrow \) GetRandomSampling();
4:   \( q_{near} \leftarrow \) GetNearestNeighbor\( (T, q_{rand}) \);
5:   \( q_{new} \leftarrow \) StraightLineSteering\( (q_{near}, q_{rand}, \delta) \);
6:   \( X_{rand}, X_{near}, X_{new} \leftarrow \) FwdKin\( (q_{rand}, q_{near}, q_{new}) \);
7:   \( X_{proj} \leftarrow \) SteeringGuide\( (\mu, \Sigma, X_{near}, X_{rand}) \);
8:   \( \Delta X_{adj} \leftarrow \frac{X_{proj} - X_{near}}{||X_{proj} - X_{near}||} \cdot ||X_{new} - X_{near}||; \)
9:   \( q_{adj} \leftarrow q_{near} + J^\dagger(q_{near}) \Delta X_{adj} ; \)
10: if StraightLine\( (q_{near}, q_{adj}) \) is Collision-Free then
11:   \( T.addTree(q_{adj}) \);
12: end if
13: end while

we define \( X_{rand} \) to be the end-effector position of the random goal \( q_{rand} \) and \( X_{near} \) to be the end-effector position of the nearest node \( q_{near} \) of \( q_{rand} \) in tree \( T \). Here, our objective is to steer the end-effector position \( X_{near} \) towards \( X_{rand} \) via the projection \( X_{proj} \) of \( X_{rand} \) onto the \( \mathcal{SC}_\mathcal{O}(X_{near}) \) along the safe corridor \( \mathcal{SC}_\mathcal{O}(X_{near}) \) in 3D space, as shown in Fig. 5.4. Accordingly, we select a steering step that is proportional with the stepsize of the standard straight-line steering of the end-effector as

\[
\Delta X_{adj} = \frac{X_{proj} - X_{near}}{||X_{proj} - X_{near}||} \cdot ||X_{new} - X_{near}||, \tag{5.16}
\]

and determine the corresponding configuration as:

\[
q_{adj} = q_{near} + J^\dagger(q_{near}) \Delta X_{adj}, \tag{5.17}
\]

where \( J^\dagger \) is the pseudoinverse of manipulator Jacobian \( J \), satisfying \( J^\dagger = J^T(JJ^T)^{-1} \).

In Fig. 5.5, we illustrate the guided steering of a manipulator using probabilistically safe corridors in the task space. The new configuration (magenta), suggested by the standard straight line planner, collides with obstacles, whereas the adjusted configuration (green), consistent with probabilistically safe corridors, moves in the tangent direction of obstacles.
GMM-based biased sampling

In our experiments, we also compute the mixtures of Gaussian $\mathcal{G}M(x, \mu_F, \Sigma_F, \omega_F)$ for modeling the free space, which is used for biased sampling over the free space as described in [58]. For the settings where biased sampling is used, instead of uniform sampling in Line 3 in Algorithms 3 and 4, we randomly sample a configuration from the collision-free Gaussian mixture distribution $\mathcal{G}M(x, \mu_F, \Sigma_F, \omega_F)$. This sampling method increases the likelihood of a new sample being collision-free and can thus increase the computational efficiency of planning as discussed below.

5.2 Results

We evaluate SG-RRT in various environments using both a simulator and a real robot. We analyze the performance of SG-RRT by comparing it with several existing RRT approaches. In addition, we demonstrate SG-RRT on a real humanoid robot and provide results under real settings. All experiments are performed on a 2.7GHz PC, and all planners are implemented in Matlab.
5.2.1 Learning Gaussian Mixture Models

In all our experiments, we learn GMMs offline by using the samples generated during the standard RRT planning (which was rich enough for accurate modeling, see Fig. 5.6(b)) and by manually selecting the kernel bandwidth for the Meanshift clustering, so that the desired level of representation resolution is guaranteed. In particular, we select the Gaussian kernel sizes for the Meanshift clustering as 10 degrees for 2DoF manipulator planning, 20 degrees for 7DoF manipulator planning, and 5 cm for task space planning. GMM learning takes 1.61 seconds for 191 clusters from 10,000 collision samples for the 2DoF manipulator, 58.97 seconds for 1,096 clusters from 19,456 collision samples for the 7DoF manipulator, and 3.64 seconds for 189 clusters from a 3D point cloud (including 18,413 data points) for task space planning. For probabilistically safe corridors, we set the desired confidence level $\kappa = 0.9$ and the safety tolerance $\epsilon = 0.01$ for all cases. In future work, we plan to consider online GMM learning for adaptive motion planning in dynamic environments.
Figure 5.7: Safety-guided RRT planning performance with respect to the number of collision samples used for GMM learning

5.2.2 2DoF planar manipulator

For ease of visual presentation, we first consider motion planning of a 2DoF planar manipulator whose first link is 0.4 units long and second link is 1.6 units long, as illustrated in Fig. 5.6(a). In Fig. 5.6, we compare the computational performance of several variants of RRT planners (the standard RRT, the biased-RRT with 10% goal bias, and the bidirectional RRT) with and without our proposed safety guided steering. Here, GMMs are learned offline along the collision space boundary (as shown in Fig. 5.3(d)) using collision samples obtained during the standard RRT planning (green points in Fig. 5.6(b)), and they are used online for constructing probabilistically safe corridors. In our quantitative evaluation, we consider the total execution time and the total number of collision checks as a performance measure, and we obtain the statistics (average and standard deviation) of these performance measures by running each planning algorithm 50 times for 20 different start and goal pairs. Overall, we observe that our safety guided steering increases computation performance significantly over the standard straight-line steering by dramatically reducing the required number of planning iterations (i.e., collision checks) to find a path between any given start and goal pair, as shown in Fig. 5.6(e). Because safety guided
steering via probabilistically safe corridors minimizes collision risk by adaptively adjusting steering direction and stepsize, our safety guided local planner yields steering actions that are significantly less likely to be in a collision, whereas the standard straight-line planner has more than a 50% chance of colliding, as seen in Fig. 5.6(e). Finally, we find it useful to emphasize that the construction of and the projection onto a probabilistically safe corridor takes around 0.2 msec on average for each new sample (denoted by “CorridorTime” in Figure 5.6(d)), which is in the same order of magnitude as the computation cost of a collision check that takes around 0.3 msec.

In Fig. 5.7, we demonstrate how the average number of RRT iterations (i.e., collision checks) required for finding a path between any given start and goal pair changes with the number of sample collision configurations (i.e., training data) used for Gaussian mixture learning. As expected, the performance of RRT planning with safety guided steering increases with the increasing size of training data as a result of the increasing accuracy of the GMM.

In Fig. 5.8, we present an application of our safety guided steering to the probabilistic roadmap (PRM) planning of the 2DoF planar manipulator. As seen in Fig. 5.8, our safety guided steering noticeably increases the connectivity of PRM as compared to the standard straight-line planner. Here, two vertices of PRM are said to be connected if safety guided steering can join them in at most 100 steps.
Figure 5.9: RRT planning performance for a 7DoF manipulator: (top) Sequential planning tasks, (middle) Average execution time, (bottom) Average number of collision checks

5.2.3 7DoF manipulator in 3D space

In order to validate the performance of SG-RRT quantitatively in high dimensional space, we compare it with traditional approaches with a 7DoF manipulator in 3D space using the Webots simulator of the Cyberbotics Ltd. company. Fig. 5.9 (top) shows the simulation scenario that is composed of seven sequential planning tasks. This scenario includes a difficult task, where the robot must remove its arm from the lower shelf and then insert it into the upper shelf. We repeat the simulation trials 50 times for accurate evaluation and use the average execution time and the number of collision checks as the evaluation criteria.

For comparison, we evaluate the standard RRT, safe-guided RRT (SG-RRT), and safe-guided RRT in the task space (WSSG-RRT). In addition, since we can apply GMM-based sampling as described in Section 5.1.3, we also evaluate GMM-based RRT (Gmm-RRT), GMM-based safe-guided RRT (GmmSG-RRT), and GMM-based safe-guided RRT in the task space (GmmWSSG-RRT). Note that we apply a bidirectional method (RRT-Connect) [78] in all approaches. The Gmm-RRT can be faster than the standard RRT, and the GmmSG-RRT is the fastest among all approaches.
The WSSG-RRT and the GmmWSSG-RRT are faster than the RRT and Gmm-RRT. This demonstrates that the end-effector of the manipulator is effectively guided by the safety corridor in the high dimensional space, and it can reduce the computational time and the number of collision checks compared to traditional approaches. We also observe in Fig. 5.9 that SGRRT planning is faster and requires less collision checks in C-spaces than in task spaces, because probabilistically safe corridors are geometrically more informative when constructed in C-spaces than in task spaces. Therefore, the tree extension with the safety corridor is significantly more efficient than the traditional methods.

5.2.4 Physical robot experiments

We demonstrate the performance of SG-RRT on a 7DoF manipulator (length: 85cm) of an actual humanoid robot and an RGBD camera (ASUS Xtion Live Pro) with the scenario shown in Fig. 5.10 (top). The robot is positioned 35cm from the shelf (35cm × 37cm) on the table. Figure 5.10 presents the comparison results of GmmSG-RRT and the standard RRT in terms of the execution time and the number of collision checks. Note that we apply a bidirectional method (RRT-Connect) and give 10% goal biased samples. Since the GmmSG-RRT adjusts a new node in the direction that avoids obstacles using probabilistically safe corridors and also utilizes biased sampling over collision-free space, the sample connectivity increases around narrow spaces, and tree expansion efficiently avoids obstacles. GmmSG-RRT is significantly efficient even when the robot needs to insert its arm onto the shelf. On the other hand, the computational time and the number of collision checks for the standard RRT planner dramatically increases in such complicated tasks.

5.3 Discussion

In this chapter, we present an effective local steering approach for sampling-based motion planning using probabilistically safe corridors of learned GMMs of C-spaces. We construct a probabilistically safe corridor around a configuration using tangent
Figure 5.10: RRT planning performance with an actual physical robot: (top) Experiment with a physical robot, (middle) Average execution time, (bottom) Average number of collision checks.

Hyperplanes of confidence ellipsoids of GMMs that are learned using collision history to approximate C-space obstacles. Accordingly, we propose a probabilistically safe local steering primitive that extends a random motion planning graph towards a sample goal using its projection onto the associated probabilistically safe corridor, which heuristically minimizes collision likelihood. We observe that the proposed local steering approach improves the performance of sampling-based planning in challenging regions, especially narrow passages, by adjusting steering direction and stepsize. In our simulations and experiments with a real robot manipulator, we demonstrate that our proposed safety guided local planner shows significant performance improvement over the standard straight-line planner for randomized motion planning of 2DoF and 7DoF manipulators. In Chapter 7, we plan to extend our work using online GMM learning for online adaptive planning.
Chapter 6

Constrained Sampling-Based Planning for Grasping and Manipulation

This chapter presents a novel constrained, sampling-based motion planning method for grasp and transport tasks with a redundant robotic manipulator. Recently, there has been growing interest in sampling-based planners for grasping and manipulation due to their simplicity and efficiency [70, 83, 69]. However, traditional sampling-based planning methods still face challenges in complex environments with clutter and with external constraints. First, these planning methods typically require setting a specific goal configuration, but it can be difficult to determine a single target configuration when there are many redundant grasping solutions. Another challenging issue arises from the fact that the constraint manifolds defined in the task space are not analytically mapped into the configuration space (C-space). For example, it is not obvious how constraints defined in the workspace (e.g., avoiding collision) can be combined with constraints defined in the task space (e.g., keeping an object upright) for planning in the C-space.

This chapter proposes an integrated sampling-based planning approach for grasping and manipulation subject to such constraints. We first address efficiency and robustness in optimal grasp planning by introducing the definition of the planning
Figure 6.1: Constrained sampling-based planning: (a) Grasping a slanted object; (b) Transporting the object with obstacle constraints

*margin.* Our proposed planning method, Maximum Planning Margin RRT (MPM-RRT), selects a grasp pose and an approach direction by maximizing an augmented *planning margin* cost in addition to a conventional grasp score. A parameterized intermediate pose is determined along with the optimal approach direction by maximizing the approach distance and penalizing deviations from joint limit constraints. Defining this intermediate approach pose increases robustness under execution errors and sensor uncertainties.

Compared to previous work that finds a reachable grasp within a ranked set based on the grasp-score function [12], or that employs Jacobian Transpose (JT) [137] or Inverse Kinematics (IK) [132] for feasible grasp planning, our method provides a principled framework that integrates manipulator motion planning with grasp planning. Instead of using pre-defined relative distances to the target for pre-grasp shapes [29, 93], we suggest using a parameterized intermediate pose.

We also address manipulation planning for object transport with soft constraints imposed on the end-effector pose. Previous studies have approached the problem by projecting samples onto the constraint manifolds [13], iteratively applying the Jacobian pseudo-inverse method to find the corresponding configuration [119] of a random sample in the task space. These approaches are limited in that the solutions tend to barely satisfy the given hard constraints. On the other hand, we suggest
a formulation of transport planning that finds collision-free paths while penalizing deviations from the task constraints of the end-effector.

In summary, the core contributions of the chapter include: 1) an efficient grasp planning algorithm which selects a robust grasp configuration by considering the proposed planning margin; 2) a stable and efficient transport using the constrained sampling-based planning, combined with an optimization strategy with imposed soft constraints in the C-space; and 3) an integrated path planning framework for both grasp and transport that considerably reduces computational time. Our methods allow us to efficiently use redundant manipulators for a variety of grasping and transport tasks (Fig. 6.2).

### 6.1 Maximum planning margin function

We define planning margin and other concepts in this section and in Section 6.2, we describe how they are incorporated to select a grasping pose. Infinitely many grasping poses for an object are possible as well as infinitely many solutions of the arm
configuration for each grasping pose. In order to select a good grasping pose, many previous methods choose the highest quality grasping pose based on the object shape among reachable solutions satisfying constraints. On the other hand, we suggest an integrative method for selecting the grasping pose considering manipulator planning.

The underlying idea behind the *planning margin* is that it is desirable to have a spatial margin large enough for a manipulator to be fully dexterous within the joint limits. Thus, we are interested in not only grasp quality itself but also finding a path that guarantees an optimal approach direction for the entire manipulation planning. This can also be useful when the target has a uniform grasp score over its shape or when an object-specific grasp configuration cannot be reached due to environmental or kinematic constraints.

Let us use the notation $T^\alpha_\beta$ to denote the homogeneous transformation of frame $\beta$ relative to frame $\alpha$, which is defined as

$$T^\alpha_\beta = \begin{bmatrix} R^\alpha_\beta & t^\alpha_\beta \\ 0 & 1 \end{bmatrix}.$$  

In order to describe the transformations, we define the reference frame $O$, the object body frame $w$, the gripper frame at the grasping pose $g$, and the gripper frame at
the intermediate pose \( a \) (Fig. 6.3). Now, the concept of *planning margin*, the key element of this study, refers to the maximum Euclidean distance of the end-effector from an intermediate pose \( a \) to a grasping pose \( g \),

\[
\mathcal{D} = \max \| t_g - t_a \|.
\]

Note that configuration between \( t_g \) and \( t_a \) satisfies constraints such as joint limits and collision avoidance.

Let us describe the *approach vector* to discuss the intermediate pose \( a \). A set of grasping poses \( S \) of the target object can be written as follows:

\[
S = \{ T_{w_{g1}}, T_{w_{g2}}, T_{w_{g3}}, \ldots \},
\]

where \( g_i \) is the \( i \)-th gripper frame, and \( T_{w_{gi}} \) is the relative end-effector pose to the object. With the RPY representation of the orientation \( R \), we can write a \( 6 \times 1 \) displacement vector \( e_{g_k} \):

\[
e_{g_k} = \begin{bmatrix} t_{g_k} \\ \Omega_{g_k} \end{bmatrix}, \quad \Omega_{g_k} = \begin{bmatrix} \arctan2(R_{g_k32}^o, R_{g_k33}^o) \\ -\arcsin(R_{g_k31}^o) \\ \arctan2(R_{g_k21}^o, R_{g_k11}^o) \end{bmatrix},
\]

to denote \( T_{o_{g_k}} \) [14].

Let \( \vec{v}_{a,k} \) be a unit vector directed towards a grasping pose \( e_{g_k} \) from an intermediate pose, which we will call *approach vector*. The *approach vector* is defined as perpendicular to the major axis of the target object to increase robustness in grasping, so the approach motion is a rectilinear motion along the approach direction. It is known that orthogonal grasping has a higher success rate than non-orthogonal grasping [8]. The approach vector can be written as,

\[
\vec{v}_{a,k} = [R_{g_k}^o t_{g_k}^o] \cdot [-1 \ 0 \ 0 \ 0]^T.
\]

Then an intermediate pose \( a \) is found along the corresponding approach vector that satisfies constraints such as joint limits and collision avoidance (Fig. 6.3 and 97).
Now we describe how to evaluate the maximum planning margin for a given grasp pose $e_g$, which will be needed for the entire planning. First, let $q$ denote a point in the C-space $Q$ of the manipulator. In particular, let us write $Q_c$ to represent the C-space in which the manipulator satisfies constraints of joint limits and collision avoidance. The manifold of end-effector poses $\mathcal{R} \subseteq SE(3)$ is defined by the forward kinematics function of the manipulator $f$ as follows.

$$f : Q_c \rightarrow \mathcal{R},$$
$$f^{-1}(e) = \{q \in Q_c | f(q) = e\}, \ e \in \mathcal{R}.$$ 

Then, in order to deal with the redundancy of the system, we define a $(n_q - n_w)$-dimensional parameter space ($n_q$ and $n_w$ are dimensions of configuration and work spaces) $\Theta$ which can span the whole initial configuration of an end-effector’s pose. In the case of a 7DoF manipulator, $\dim(\Theta) = 1$. We denote a configuration state with a fixed value of $\theta \in \Theta$ by $q_\theta$. With $\theta$, the mapping from $\mathcal{R}$ to $Q_c$ can be computed efficiently.

Now let $\tau^i : [0, 1] \rightarrow Q_c$ be a path from the grasping pose $e_g$ to an intermediate pose in the C-space ($s \subseteq [0, 1]$). Starting from the initial configuration $q_{\theta, 0}^i = \tau(0)^i$.
which corresponds to the grasping pose, we may compute all configurations along the approach vector \( \vec{v}_{a} \) iteratively using the Jacobian pseudo-inverse method. Letting \( t(q) \) be the translation components of \( f(q) \in SE(3) \), \( q_{g,f}^i = \tau(1)^i \). We can write the maximum planning margin of the end-effector \( e_g \) as,

\[
D_i = \max_{\theta \in \Theta} \| t(q_{\theta,f}^i) - t(q_{\theta,0}^i) \| \quad (6.1)
\]

subject to

\[
q_{\theta,0}^i = f^{-1}(e_g, \theta) \\
q_{\theta,k+1}^i = q_{\theta,k}^i + \eta J^\dagger(q_{\theta,k}^i) \Delta e \\
q_{\theta,k+1}^i \in \tau^i(s) \subset Q_c,
\]

where \( \Delta e = [\vec{v}_{a,k}^T, 0 0 0]^T \), and \( J^\dagger \) is the pseudo inverse of Jacobian \( J \) (i.e., \( J^\dagger = J^T(JJ^T)^{-1} \)).

Thus, we can compute the maximum planning margin with respect to the parameter \( \theta \) given \( e_g \) via Eq. (6.1). Our strategy for determining the grasping pose, which will be fully described in the following section, incorporates searching in this parameter space for a maximum planning margin.

### 6.2 Optimization planning for grasping

As mentioned, we may consider the quality of grasping pose together with the spatial planning margin when determining the grasping pose. Let \( G(e_{g_i}) \) be the grasp success rate function of the grasping pose \( e_{g_i} \) based on the object model. Then a general form of the optimization problem for the grasping pose can be written as follows:

\[
e^* = \arg \max_{e_{g_i}, \Theta} \{ \alpha D_i(e_{g_i}, \Theta) + (1 - \alpha) G_i(e_{g_i}) \}, \quad (6.2)
\]
where $\alpha$ is a weight value between the grasp success rate and planning margin. With this formulation, we can now describe the best grasping pose in terms of grasp quality and spatial margin. It is still non-trivial to find an optimal solution in this non-convex optimization problem in multidimensional space: The derivatives of the objective function are unknown, and the planning margin computation involves solving the pseudo-inverse of the Jacobian. The weights will vary from object to object since some shapes or materials require more delicate handling than others. Although learning the object-specific weights will be an interesting further study, we focus on the newly introduced term, planning margin, in this chapter.

For the purpose of evaluation of our proposed planning margin, an axis-symmetric object is an appropriate target. Since the symmetry renders the grasp score uniform around the symmetric axis, the maximum planning margin alone determines the optimization solution of Eq. (6.2). Then the grasping pose $e_{gi}$ can be described using the relative yaw angle $\psi$ between the object and the gripper. The cost can be visualized in two-dimensional space of the parameter ($\theta$) and the relative orientation between the object and gripper ($\psi$), as shown in Fig. 6.5.

We apply the Nelder-Mead (NM) method [98] with multiple initial points to find an optimal solution via search. Even though it is generally hard to achieve global optimality in a non-convex problem, we found that this algorithm gives reasonable local solutions rather quickly and could be further sped up by parallelization.

Figure 6.5: Visualization of the non-convex planning margin functions with respect to end-effector’s yaw angle and the parameter
After determining the best grasping pose ($e^*$), we compute an intermediate pose $q_{\text{interm}}$ that constitutes a preshaping state for the approaching action. Inspired by human grasping motion [25], the need of this preshaping state naturally arises when aligning the gripper pose toward the object to increase robustness of grasping. The intermediate pose $q_{\text{interm}}$ can be computed from the optimal solution as follows:

$$q_{\text{interm}} = q_0^* + l^* \cdot J^\dagger(q_0^*) \Delta e^*,$$

where $l^*$ is an approach distance from the object to the intermediate point. The distance is decided by joint limit margins since the longer the approach distance, the closer the joint angles are to the joint limits. That is,

$$l^* = \arg \max_l (l + \log(\min_j |q_i - q_{i,\text{limit}}|)).$$

This objective function with a logarithmic barrier penalty for joint limits is visualized in Fig. 6.10(b). During the execution of the whole motion, the manipulator first reaches a desired intermediate pose by the standard RRT, then the gripper opens at the preshaping state and finally, the manipulator approaches the target grasping pose using the Jacobian pseudo-inverse method with no collision, satisfying joint limits (Fig. 6.6).
Algorithm 5 MPM-RRT \((q_{\text{init}}, t_{\text{goal}})\)

1. \(e_{\text{init}} \leftarrow \text{ForwardKinematics}(q_{\text{init}})\)
2. \(T.\text{init}(e_{\text{init}}, q_{\text{init}})\)
3. while Distance\((t_{\text{goal}}, t_{\text{new}}) > \delta_{\text{min}}\) do
4. \(t_{\text{rand}} \leftarrow \text{GetRandomSampling()}\)
5. \(e_{\text{near}}, q_{\text{near}} \leftarrow \text{GetNearestNeighbor}(T, t_{\text{rand}})\)
6. \(e_{\text{new}}, q_{\text{new}} \leftarrow \text{FindOptimalConfig}(t_{\text{rand}}, e_{\text{near}}, q_{\text{near}})\)
7. if \(q_{\text{new}} \neq \text{NULL}\) then
8. \(T.\text{addTree}(e_{\text{new}}, q_{\text{new}})\)
9. end if
10. end while
return \(\text{GetFinalPath}(T)\)

6.3 Transport planning with soft constraints

This section describes our transport planning with soft constraints. Planners sometimes need to conserve the end-effector’s orientation, such as keeping an object upright while moving it to the target position. Moreover, the target end-effector position \(t \in \mathbb{R}^3\) is usually given without specific arm configurations. The standard RRT algorithm grows trees in the C-space without consideration of the end-effector’s pose. Thus, the sampling should be restricted within the region where the constraints are satisfied, and the target arm configuration should be resolved together with the transport planning. (It is computationally intractable to restrict random samples by a rejection approach since the orientation of the end-effector is the result of all arm joint configurations).

In order to address these issues, we suggest a strategy that samples in the task space (Algorithm 5) and then searches for an optimal configuration that satisfies imposed soft constraints (Algorithm 6). Shkolnik et al. [119] have suggested sampling directly in the task space and applying the Jacobian pseudo-inverse to find a corresponding configuration to the sample in the task space, but this approach is effective only in an open space with few obstacles as described in [135].

First, a node \(N_i\) in the tree is defined as the pair of a configuration \(q_i\) and a pose of end-effector \(e_i\) in the task space.

\[
N_i = \{q_i, e_i\}
\]
Algorithm 6 Optimization with Nelder-Mead method

1: function FindOptimalConfig(t\text{rand}, e_{\text{near}}, q_{\text{near}})  
2: \quad q_1, \ldots, q_{n+1} \leftarrow \text{RandomVertices}(q_{\text{near}})  
3: \quad c_1, \ldots, c_{n+1} \leftarrow \text{ComputeCost}(q_1, \ldots, q_{n+1}, q_{\text{near}})  
4: while difference($c_1, c_{n+1}$) > $\delta$ do  
5: \quad q_1, \ldots, q_{n+1} \leftarrow \text{UpdateVertices}(q_1, \ldots, q_{n+1})  
6: \quad c_1, \ldots, c_{n+1} \leftarrow \text{ComputeCost}(q_1, \ldots, q_{n+1}, q_{\text{near}})  
7: end while  
8: if CheckConstraints($q_1$) = TRUE then  
9: \quad e_1 \leftarrow \text{ForwardKinematics}($q_1$) return $e_1, q_1$  
10: end if  
11: return $e_1 = NULL, q_1 = NULL$  
12: end function  
13: function Compute\_Cost(q_1, \ldots, q_{n+1}, q_{\text{near}})  
14: \quad e_j \leftarrow \text{ForwardKinematics}(q_{\text{near}})  
15: for $i \in \{1, \ldots, n+1\}$ do  
16: \quad e_i \leftarrow \text{ForwardKinematics}(q_i)  
17: \quad c_i = \alpha_1\|q_i - q_{\text{near}}\| + \alpha_2c_d(q_i) + \alpha_3|\phi_i - \phi_j| +  
18: \quad \alpha_4|\theta_i - \theta_j| + \alpha_5|\psi_i - \psi_j|  
19: end for  
20: c_1, \ldots, c_{n+1} \leftarrow \text{Ordering}(c_1, \ldots, c_{n+1}) return $c_1, \ldots, c_{n+1}$  
21: end function

$q_i \in Q_c$, $e_i \in \mathcal{R}$, $f(q_i) = e_i$

where $Q_c$ is the C-space at which the manipulator satisfies constraints such as joint limits and collision-free. A new node $N_i$ is connected with an existing node $N_j$ in the tree, for which

$$j = \arg \min_k \|t_i - t_k\|, \quad k \in \{1, \ldots, N_t\}$$

where $t_i$ and $t_k$ are the positions of $e_i$ and $e_k$, and $N_t$ is the number of nodes in the tree.

Now, we define an obstacle cost function which penalizes the manipulator for being close to obstacles in the workspace, similar to [150],

$$c_d(q) = \begin{cases} 
(d_{\text{min}} - \mathcal{D}(q))^2, & \text{if } \mathcal{D}(q) < d_{\text{min}} \\
0, & \text{otherwise}
\end{cases}$$

where $\mathcal{D}(q)$ is the distance to the obstacle from the manipulator. Then, we can find
the $q_{\text{new}}$ of the new node $N_{\text{new}}$ by solving the optimization problem as follows:

$$
\min \omega = \alpha_1 \| q_{\text{new}} - q_j \| + \alpha_2 c_d(q_{\text{new}}) + \alpha_3 |\phi_{\text{new}} - \phi_{\text{target}}| + \\
\alpha_4 |\theta_{\text{new}} - \theta_{\text{target}}| + \alpha_5 |\psi_{\text{new}} - \psi_j| 
$$

(6.3)

s.t.

$$
\arg \min_k \| t_{\text{new}} - t_k \|, \quad k \in \{1, \cdots, N_t\}
$$

where $R_c \in SE(3)$ is a collision free space in the task space and $\phi$, $\theta$, and $\psi$ define the orientation of the end-effector with respect to the task frame. Eq. (6.3) is a constrained optimization problem with variables $\phi$, $\theta$, $\psi$, and the parameter space $\Theta$. This formulation optimizes the path by penalizing deviated motion from desirable constraints and avoiding collisions. For instance, collision-free configurations will be favored by penalizing the collision-related cost, and configurations that are not satisfied with the constraints will be avoided by the preservation cost. The regularization parameters were empirically obtained to minimize the computational time of transport planning in our preliminary test with Scenario 3 in Fig. 6.8(b).

The Nelder-Mead (NM) method is applied to obtain the numerical solution of the
optimal configuration in each node that minimizes the cost function with constraints imposed in the C-space. The initial value in the NM method is the configuration of the nearest node that is connected to the new node; thus, it finds a solution near the configuration of the nearest node. Since the difference between the nearest node and a new node is not significant, the NM method converges very quickly when it initiates at the nearest node. It converges within 10 iterations in our problem. If the new node has no configuration that satisfies the constraints, the RRT rejects the new node and generates another random node in the task space. Algorithm 5 and Algorithm 6 present a pseudocode. In Algorithm 6, the UpdateVertices function follows the procedure of the NM algorithm, such as reflection and expansion. In summary, our approach has the following key advantages:

- Since this approach rejects a node that violates constraints, only existing nodes that are connected to collision-free nodes are extended; consequently, the tree grows biased toward the collision-free space and reduces the search space.

- It converges quickly since it searches an optimal solution near the configuration of the nearest node.

- It has a structure for finding an optimized path with several soft constraints.

We combine all the discussed ideas for grasp and transport tasks in a single planning framework and denote it as MPM-RRT. MPM-RRT can efficiently generate paths for grasp and transport tasks under constraints without heavy computation.

### 6.4 Results

We evaluate our method, MPM-RRT, under various experimental settings using both a simulator and a real system. In order to validate the performance of MPM-RRT quantitatively, we have four scenarios for grasping (Fig. 6.8(a)) and four scenarios for transport (Fig. 6.8(b)), and we compare MPM-RRT with two previous approaches on the Webots simulator of Cyberbotics Ltd. Note that we implement JT-RRT and
IK-RRT with parameters and algorithms as described in [137, 132]. Note that the reachability space is not considered in IK-RRT. To apply JT-RRT for grasping, we consider the center point of the target object as the target position in the workspace $x_{goal}$ in [137]. Post-processing, such as smoothing, is not included for evaluation. In addition, we demonstrate the integrated performance of MPM-RRT on a real humanoid robot with 3D perception, and we provide qualitative observations under real settings.

All experiments are performed using a 7DoF manipulator with a three-finger gripper, and all programs are written in Matlab on a 2.7GHz PC. All simulation trials are repeated 50 times for each scenario for accurate evaluation, and we use the average execution time and path length of each scenario as evaluation criteria.

### 6.4.1 Grasping

In Scenarios 1 and 2 in Fig. 6.8(a), the object is located under or on top of a shelf, which acts as an obstacle together with the desk, and the manipulator starts from different initial configurations. In Scenario 3, the robot is initially positioned very close to the desk, allowing it little space to approach the object while avoiding collisions to the desk. Scenario 4 was intended to be more challenging since the
Figure 6.9: The grasping results in scenarios of Fig. 6.8(a): MPM-RRT is relatively faster in the complicated scenarios.

A manipulator must be inserted onto one shelf to grasp the target object from beneath the upper shelf.

Fig. 6.9 shows our results of all scenarios compared with JT-RRT and IK-RRT in terms of time and path length efficiency. As can be seen in Fig. 6.9, MPM-RRT for grasping is significantly more efficient than the other approaches. JT-RRT performs well when there is no obstacle between the initial pose and the target grasping pose, as in the first scenario. However, the computational time exponentially increases as the environment gets more complicated. IK-RRT is faster than JT-RRT, but it has longer average paths and is slower than MPM-RRT. Another noticeable result is that the computation time of MPM-RRT does not increase much even when the environmental condition becomes more complicated. Fig. 6.10 shows the convergence of the grasping selection and the computation of the approach distance, as explained in Sec 6.2.
Figure 6.10: (a) Fast convergence of the grasping pose selection. (b) Determining the approach distance by maximizing the cost with logarithmic barrier penalty. (Best viewed in color)

6.4.2 Transport

Transport scenarios are shown in Fig. 6.8(b). Scenario 1 involves placing the target object beneath the shelf, and Scenario 2 involves placing the slanted object on the top of the shelf while avoiding obstacles. The third scenario is an arrangement of multiple target objects, and the last scenario is to transport the slanted object to beneath the shelf. For a strict comparison in the transport task, only the target position of the end-effector is given without a specific arm configuration. Unlike JT-RRT and IK-RRT, MPM-RRT has constraints to preserve the roll and pitch angles of the end-effectors for stability of the grasped object, although we can apply different constraints, such as preserving different angles. In spite of considering constraints, MPM-RRT has shorter trajectories while completing the task faster (Fig. 6.11). Moreover, as shown in Fig. 6.12, the variance of the joint angles with MPM-RRT is smaller. This implies that MPM-RRT produces smoother and more stable angular changes while preserving the roll/pitch angle of the target during transport. (Due to space limitations, the figure does not include JT-RRT, which has similar variances to IK-RRT.) In summary, it is clear that MPM-RRT for transport is more efficient and effective than previous approaches.
Figure 6.11: The transport result in scenarios of Fig. 6.8(b): It shows that time and path length are less than for JT-RRT and IK-RRT.

Figure 6.12: The transport planning result of the third scenario in Fig. 6.8(b) : (a) MPM-RRT with soft constraints, (b) IK-RRT.
6.4.3 **Experimental result in the real environment**

We have conducted a series of grasping and transport tasks with a 7DoF manipulator and a three-finger gripper of a humanoid robot. We implemented a 3D perception module to obtain the obstacle map and to estimate the target 3D pose using an RGBD camera (ASUS Xtion Live Pro). The perception module first segments out large planar objects, then clusters point clouds, and finally searches for the target among the clusters given its 3D model, as shown in Fig. 6.13.

We conducted three tasks for the evaluation of the suggested grasping and transport approach. All tasks require the robot to grasp the object with *maximum planning margin* while maintaining the roll and pitch angles of the end-effector during transport. In the first task, the robot must move objects to varying elevations to evaluate transporting items to different heights. The second task involves moving an object to a location that is already occupied. The robot will need to grasp a slanted object and move it out of the way and then move the other object to the correct location. In the third task, the robot moves an object from the top of a shelf to place it on the shelf below while preserving end-effector orientation. While this task is difficult because there is very little space between the robot and the shelf, our suggested approach successfully executed the tasks (Fig. 6.14).
6.5 Discussion

This chapter has presented an efficient constrained sampling-based planning algorithm for grasp and transport tasks. We suggested grasping selection and an optimal approach direction based on the planning margin, which is applicable for grasp planning with constraints. A parameterized intermediate pose has also been proposed to increase robustness to execution errors and sensor uncertainties. Additionally, this chapter has proposed sampling-based planning with soft constraints for the transport, which includes an efficient method for finding a configuration corresponding to the samples in the task space, satisfying constraints such as joint limits and obstacle avoidance. Our proposed method has been demonstrated on several example applications in both simulation and real-world environments, verifying that it produces a marked improvement in computational efficiency over previous approaches.
Chapter 7

Online/Adaptive Learning for Sampling-Based Planning

This chapter addresses online learning of probabilistic models and a method that is adaptive to a changed environment. In previous chapters, probabilistic generative models were discussed for fast sampling-based planning. However, even though the performance is dramatically improved, the weakness of these approaches is that they require learning time and samples for training the models. Therefore, we need an online approach for modeling to minimize the burden of the learning phase as shown in Fig. 7.1. In this chapter, we suggest an online learning approach based on the Dirichlet Process (DP), which is a non-parametric Bayesian approach for representation of the configuration space (C-space).

In addition, since the robot usually moves while performing a task, the relative pose between the robot and the environment can be changed. In this case, since the collision regions in the configuration space transform significantly when the pose changes and it has a complicated relationship to the workspace movement, we need new GMMs under pose changes. The naive approach is learning the new model again from scratch, but it is not applicable since it needs new exemplars for the training, and it is time-consuming. Therefore, we introduce an approach to transform the previously learned models in order to adapt to the pose changes.
7.1 Online learning with Dirichlet Process

In previous chapters, several sampling-based planning approaches for probabilistic models with efficient sampling, local steering, or fast collision checks were suggested. Although these approaches show that the performance is significantly improved with the probabilistic model, they require training for the learning model. In addition, they need to update the model whenever new samples are obtained in unexplored areas.

We introduce an online learning approach with a non-parametric Bayesian approach for representation of the configuration space based on the Dirichlet Process (DP). This representation is exploited for efficient sampling-based motion planning in configuration spaces. Compared to previous sampling-based planning approaches based on probabilistic models which require a parameter of the model, such as the number of clusters, and batch learning with a large volume of data, this approach can adapt the complexity to the data and learn new Gaussian clusters with stream data in newly explored areas. Since this approach is a non-parametric Bayesian model, it can overcome the problem of parametric models which suffer from model selection and right complexity, and the learned representation can be applied for efficient sampling, local steering, or a probabilistic collision map. In addition, the learned probabilistic model is continually updated and enhanced with samples obtained as the Rapidly-exploring Random Tree (RRT) planner proceeds.

We propose Dirichlet Process Mixture Models (DPMM) to learn probabilistic
models for efficient sampling. Based on this model, we generate probabilistic collision maps for fast collision checks or probabilistically safe corridors for local steering (Fig. 7.2). In addition, we utilize the probabilistic model for biased sampling to decrease the number of samples for sampling-based planning algorithms.

Our approach differs from the previous approaches using the Gaussian Mixture Models (GMMs) with large-size static examplars for sampling-based planning. We introduce DPMM for continual updating of Gaussian mixtures with newly obtained samples, whereas previous approaches simply use rich enough samples to learn the GMM model [58].

In summary, the main contributions include:

1. a novel online approximation of collision and collision-free configuration subspaces,

2. a probabilistic collision map using the generated probabilistic model,

3. an effective RRT planning approach with the probabilistic collision map, biased sampling, and a probabilistically safe local steering primitive.

We will demonstrate that the proposed approach can dramatically improve the performance of randomized motion planners with online updates.
7.1.1 Dirichlet Process Mixture Model

We will briefly explain the DPMM in this section. Traditional parametric modeling suffers from model selection and right complexity [129, 49]. While the parametric model needs to decide important factors, such as the number of mixture models, non-parametric Bayesian modeling assumes infinite dimensional parameters and adapts its complexity to the data. Since we update the probabilistic model with new data obtained during planning, the non-parametric Bayesian model is appropriate for our approach. When we get new samples in unexplored regions, new Gaussian clusters are generated for modeling with these new samples. We apply the Dirichlet Process (DP) as a prior for the infinite mixture model, which is defined as follows:

\[ G \sim DP(\alpha, G_0) \]
\[ \eta_k \sim G \]
\[ X_k \sim F(\eta_k) \]

where DP has two parameters, a base distribution \( G_0 \) and a concentration parameter \( \alpha \). \( \eta_k \) is a random variable for the parameter of the distribution of the \( k \)th observation, and \( F \) is multivariate Gaussian. In practice, drawing a concrete distribution \( G \) is impossible, since it requires an infinite amount of information [18]. Since another representation of DP, Chinese Restaurant Process (CRP), is applicable, we sample for the component indicator \( c_1, \ldots, c_N \) as well as the component parameters \( \eta_c \) for all \( c \) in \( \{c_1, \ldots, c_N\} \) based on CRP formulation instead of sampling for \( \eta_1, \ldots, \eta_N \) directly. In addition, since the mean and variance of multivariate Gaussian \( F \) are unknown, we choose the Normal Inverse Wishart distribution (NIW) for the base distribution \( G_0 \), which is a conjugate prior to multivariate Gaussian with unknown
mean and variance. The infinite mixture model is defined as follows:

\[
\eta_{1,2,\ldots,\infty} \sim NIW(\mu_0, \kappa_0, \nu_0, S_0) \\
c_{1,2,\ldots,N} \sim CRP(\alpha) \\
X_i \sim F(\eta_{c_i})
\]

where \(\alpha\) is a hyperparameter of the Dirichlet Process, \(F\) is a multivariate normal distribution, \(G_0\) is a conjugate prior to \(F\), and \(\eta_{c_i}\) is a parameter set \((\mu, \Sigma)_{c_i}\) for multivariate normal distributions. For the clustering given exemplars, we want to infer \(P(c_{1:N}|X_{1:N})\). We sample indicators of data points by Gibbs sampling based on probability \(P(c_i = k|X_{1:N}, c_{-i}, \theta, \alpha) \propto P(c_i = k|c_{-i}, \alpha)P(X_i|X_{-i}, c_{-i}, c_i = k, \theta)\), where \(\theta\) is a set of hyperparameters \(\mu_0, \kappa_0, \nu_0, S_0\) of the conjugate base distribution \(G_0\). The first term of the probability is given by the CRP scheme as follows:

\[
P(c_i = k|c_{-i}, \alpha) = \begin{cases} 
\frac{N_{-i,k}}{N-1+\alpha} & \text{for existing clusters} \\
\frac{\alpha}{N-1+\alpha} & \text{for any new cluster}
\end{cases}
\]

where \(N_{-i,k}\) is the number of elements in the \(k\)th cluster excluding \(i\)th data, and \(N\) is the total number of elements. The second term is the probability of the data given previous data points, and it is a ratio of normalizing constants of the posterior distribution of the \(k\)th parameter, one including and one excluding the \(i\)th data point \[18\] as follows:

\[
P(X_i|X_{-i}, c_{-i}, c_i = k, \theta) = \frac{P(D_k|\mu_0, \kappa_0, \nu_0, S_0)}{P(D_{k,-i}|\mu_0, \kappa_0, \nu_0, S_0)}
\]

where \(D_k\) is a data set in the \(k\)th cluster including \(i\)th data, and \(D_{k,-i}\) is a data set in the \(k\)th cluster excluding \(i\)th data. When \(G_0\) is not conjugate, the second term has no simple closed form. Algorithm 7 shows the pseudocode for DPMM.
Algorithm 7 Dirichlet Process Mixture Model

Require: $X_{\text{data}}$
1. $n_{\text{samples}} \leftarrow \text{size}(X_{\text{data}})$
2. $X_{\text{data}} \leftarrow \text{RandomSort}(X_{\text{data}})$
3. $I_{\text{set}} = \text{ones(size}(X_{\text{data}}))$
4. $S_1 \leftarrow \text{init}\_S$
5. $\mu_1 = \text{mean}(X_{\text{data}})$
6. for $i = 1, n_{\text{samples}}$ do
7. \hspace{1em} $\text{update}\_\text{parameters}(X_{\text{data}}(i), \lambda_1, \nu_1, S_1)$
8. end for
9. for $\text{iter} = 1, T$ do
10. \hspace{1em} for $i = 1, \text{randperm}(n_{\text{samples}})$ do
11. \hspace{2em} $O_c = I_i$
12. \hspace{2em} if $\text{cardinal}(O_c) == 1$ then
13. \hspace{3em} remove\_\text{cluster}(O_c)$
14. \hspace{2em} else
15. \hspace{3em} down\_\text{date}\_\text{param}(X_{\text{data}}(i), \lambda_{O_c}, \nu_{O_c}, S_{O_c})$
16. \hspace{2em} end if
17. \hspace{2em} for $k = 1, K$ do
18. \hspace{3em} $\text{crp}(k) \leftarrow (\text{cardinal}(k)) / (N + \alpha - 1)$
19. \hspace{3em} $\text{llhood}(k) \leftarrow \text{compute}\_\text{llh}(X_i, \lambda_k, \nu_k, S_k)$
20. \hspace{3em} $\text{lnprob}(k) \leftarrow \text{log}(\text{crp}(k) + \text{llhood}(k))$
21. \hspace{2em} end for
22. \hspace{2em} $\text{crp}(K + 1) \leftarrow \alpha / (N + \alpha - 1)$
23. \hspace{2em} $\text{llhood}(K + 1) \leftarrow \text{compute}\_\text{llh}(X_i, \lambda_{\text{init}}, \nu_{\text{init}}, S_{\text{init}})$
24. \hspace{2em} $\text{lnprob}(K + 1) = \text{log}(\text{crp}(K + 1) + \text{llhood}(K + 1))$
25. \hspace{2em} prob = $\exp(\text{lnprob}\_\text{logsumexp}(\text{lnprob}))$
26. \hspace{2em} $I_i \leftarrow \text{randsample}(\text{prob})$
27. \hspace{2em} $\text{update}\_\text{param}(X_i, \lambda_{I_i}, \nu_{I_i}, S_{I_i})$
28. \hspace{1em} end for
29. end for

7.1.2 Online learning of GMMs

In this section, we address the update of the probabilistic model with newly obtained samples. The clusters are generated with the first obtained data using the approach DPMM, described in the previous section. With this initial model, the newly obtained data is assigned to a cluster based on the probability $P(c_i = k|X_{1:N}, c_{-i}, \theta, \alpha)$, and the parameters of the cluster are updated. In this initial assignment, the data is assigned into existing clusters only (Algorithm 8). After the initial assignment of new data, the clustering is optimized by Gibbs sampling based on probability...
Algorithm 8 DPMM with new data

Require: : $X_{\text{new}}, X_{\text{set}}, \mu_{\text{set}}, \nu_{\text{set}}, S_{\text{set}}, I_{\text{set}}$

1: $\text{n\_new\_samples} \leftarrow \text{size}(X_{\text{new}})$
2: $\text{n\_samples} \leftarrow \text{size}(X_{\text{set}})$
3: for $i = 1$, randperm(n\_new\_samples) do
4:   $\text{n\_samples} = \text{n\_samples} + 1$
5:   $N = \text{n\_samples}$
6:   for $k = 1, K$ do
7:      $\text{crp}(k) \leftarrow (\text{cardinal}(k)) / (N + \alpha - 1)$
8:      $\text{lhhood}(k) \leftarrow \text{compute\_llh}(X_i, \lambda_k, \nu_k, S_k)$
9:      $\text{lnprob}(k) \leftarrow \log(\text{crp}(k) + \text{lhhood}(k))$
10: end for
11: $\text{crp}(K + 1) \leftarrow \alpha / (N + \alpha - 1)$
12: $\text{lhhood}(K + 1) \leftarrow \text{compute\_llh}(X_i, \lambda_{\text{init}}, \nu_{\text{init}}, S_{\text{init}})$
13: $\text{lnprob}(K + 1) = \log(\text{crp}(K + 1) + \text{lhhood}(K + 1))$
14: $\text{prob} = \exp(\text{lnprob\_logsumexp}(\text{lnprob}))$
15: $I_i \leftarrow \text{randsample}(\text{prob})$
16: $\text{update\_param}(X_i, \lambda_{I_i}, \nu_{I_i}, S_{I_i})$
17: $X_{\text{set}} \leftarrow [X_{\text{set}}; X_i]$
18: $I_{\text{set}} \leftarrow [I_{\text{set}}; I_i]$
19: end for

\[ P(c_i = k | X_{1:N}, c_{-i}, \theta, \alpha) \propto P(c_i = k | c_{-i}, \alpha) P(X_{i} | X_{-i}, c_{-i}, c_i = k, \theta) \]  (Algorithm 9). In this optimization step, the new cluster can be generated if it is essential. Since the model is updated with new sample data every RRT planning iteration, it is more accurate.

Interestingly, the obtained collision samples are located in the boundaries of the collision region since the extension of the tree is stopped after it collides with obstacles. Therefore, the Gaussian mixtures with collision samples have small variance in the perpendicular direction at the boundary, and the insides of the collision region remain unexplored by the RRT. To resolve these issues and refine the model, we suggest a sampling method which draws additional random samples according to the inverse of the probability of the sample distribution. We apply the rejection sampling approach, which rejects a sample with high probability of sample distribution (Fig. 7.3). This rejection sampling also prevents the samples from drawing within the same region. Fig. 7.4 shows the clustering results of collision and collision-free exemplars obtained during 10 RRT planning iterations.
Algorithm 9 DPMM Optimization

Require: \(X_{set}, \mu_{set}, \lambda_{set}, \nu_{set}, S_{set}, I_{set}\)

1. \(n_{samples} \leftarrow \text{size}(X_{set})\)
2. for \(\text{iter} = 1, T\) do
   3. for \(i = 1, \text{randperm}(n_{samples})\) do
      4. \(O_c = I_i\)
      5. if \(\text{cardinal}(O_c) == 1\) then
         6. remove_cluster\((O_c)\)
      7. else
         8. downdate_param\((X_{data}(i), \lambda_{O_c}, \nu_{O_c}, S_{O_c})\)
      9. end if
      10. for \(k = 1, K\) do
          11. \(\text{crp}(k) \leftarrow (\text{cardinal}(k)) / (N + \alpha - 1)\)
          12. \(\text{llhood}(k) \leftarrow \text{compute_llh}(X_i, \lambda_k, \nu_k, S_k)\)
          13. \(\text{lnprob}(k) \leftarrow \log\left(\text{crp}(k) + \text{llhood}(k)\right)\)
      14. end for
      15. \(\text{crp}(K + 1) \leftarrow \alpha / (N + \alpha - 1)\)
      16. \(\text{llhood}(K + 1) \leftarrow \text{compute_llh}(X_i, \lambda_{init}, \nu_{init}, S_{init})\)
      17. \(\text{lnprob}(K + 1) = \log(\text{crp}(K + 1) + \text{llhood}(K + 1))\)
      18. \(\text{prob} = \exp(\text{lnprob}\_\text{logsumexp}(\text{lnprob}))\)
      19. \(I_i \leftarrow \text{randsample}(\text{prob})\)
      20. update_param\((X_i, \lambda_{I_i}, \nu_{I_i}, S_{I_i})\)
   21. end for
   22. end for

7.1.3 Probabilistic collision map

In this section, we explain an approach for exploiting the learned model by the online learning approach for sampling-based planning. First, we generate the probabilistic collision map based on the learned Gaussian mixtures by DPMM as shown in Fig. 7.5. We let \(k\) denote the number of Gaussian mixtures obtained by DPMM. Each \(j\)th Gaussian has mean \(\mu_j\), and Covariance \(\Sigma_j\). We can compute the weight coefficients \(\Theta\) of Gaussian mixtures by the method of least squares as follows:

\[
\Theta = (X^T X)^{-1} X^T y, \quad [X]_{ij} = \psi_j(x_i),
\]
Figure 7.3: Online learning with the rejection sampling

Figure 7.4: DPMM results (a) collision clustering (b) collision-free clustering

\[ \psi_j(x_i) = \mathcal{N}(x_i; \mu_j, \Sigma_j) \] for all samples \( x_i, i = 1 \ldots N \)

\[ y_i = \begin{cases} 
1 & \text{if } x_i \text{ is a collision configuration} \\
0 & \text{otherwise} 
\end{cases} \]

With these coefficients and Gaussian mixtures, we compute the probability collision map as follows:

\[ p(x) = \sum_{j=1}^{k} \theta_j \mathcal{N}(x; \mu_j, \Sigma_j) \]  \hspace{1cm} (7.1)

This map is updated according to the new mixtures which are updated with new
samples by DPMM. Fig. 7.6 shows the update of the probabilistic collision map in the C-space of the 2DoF planar manipulator according to new samples. In the first step, since the obtained samples are distributed over small areas, there is a model in the small areas. With increasing samples, the model becomes more accurate as shown in Fig. 7.6 (Step 1-Step 20). This probabilistic collision map can be used for fast collision checks [58] and biased sampling of sampling-based planning.

7.1.4 Local steering with a probabilistic model

We also apply the learned model by the online learning method for local steering, such as the probabilistically safe corridor. Fig. 7.7 shows the planning result with the probabilistically safe corridor based on the probabilistic distribution. Note that the suggested approach is conducted like the basic RRT planner when it has no information as shown in the first step (Fig. 7.7(a)). After updating the probabilistic model with samples from RRT planners with the online learning approach, we can generate a probabilistic model and use it for efficient local steering of the RRT. Since the probabilistic corridor becomes more accurate according to the accuracy of the model, and the planning becomes more efficient, as discussed in Chapter 5.

7.1.5 GMM-based sampling

For efficient sampling of the RRT, we exploit collision-free distribution to reduce the number of collision checks. For the tree extension in the RRT, a random node $x_{rand}$
Figure 7.6: Online update of probabilistic collision map

is chosen from the $K$ multivariate Gaussian mixture components of the collision-free GMM as follows.

$$x_{rand} \sim \sum_{k=1}^{K} \theta_k \mathcal{N}_{\text{free}}(\mu_k, \Sigma_k) + \lambda \cdot \mathcal{U}_n,$$

where $x_{rand}$ is the probability that a new random node is drawn from the $k$th mixture component, and $\lambda \cdot \mathcal{U}_n$ is a regularization term to prevent overfitting. $\mathcal{U}_n$ means the uniform distribution over $n$-dimensional space. We can determine the hidden variable $\theta_k$ by Equation (7.1).

The tree has a tendency to extend toward collision-free distributions, and it can reduce the number of collision checks since it increases the probability of selecting a collision-free node for the new node. The regularization exists in order to prevent elimination of the possibility of finding a more efficient path existing outside the space of the GMMs.
Figure 7.7: RRT planning with probabilistically safe corridor based on online learning
7.1.6 Discussion

This chapter has presented an online learning approach based on a Dirichlet Process Mixture Model (DPMM). For the future work, we will focus on high-dimensional space and the case of a changed environment, which is one of the more difficult tasks when we apply learning approaches in planning. We will apply an evolutionary clustering algorithm to adapt to the changed environment. In addition, since Gibbs sampling is time-consuming, we will suggest an efficient method to handle the complexity of computation and the number of samples.

7.2 Adaptive motion planning with high-dimensional mixture models

Recently, there has been much interest in better methods for robotic manipulation in dynamic and uncertain environments as robots work more closely with people [2]. However, it is difficult for traditional motion planning algorithms to handle environmental changes in an efficient and accurate manner. A slight change in pose between the robot base and its workspace can render previously found solutions for motion trajectories worthless. Naively, such a change necessitates replanning a new motion trajectory from scratch leading to heavy computational loads and slow and inefficient robot behaviors.

We propose a method for adaptive motion planning using high-dimensional Gaussian mixture models (GMMs). Previously in Chapter 3, we presented a GMM-based fast collision detection method for the RRT, which has been shown to significantly reduce computational times over other sampling-based planning methods. Now we address the problem of relearning GMMs that represent the collision and collision-free regions in configuration space (C-space) under environmental changes. A naive implementation would require relearning the mixture models from scratch via the Expectation-Maximization (EM) algorithm by repeatedly calling a slow, inefficient kinematic based collision check routine on a large number of training examples. In
this work, we suggest a novel GMM-based RRT which rapidly adapts to scenarios where the robot repeatedly moves relative to the desired workspace.

In order to update the learned collision prediction models, we incorporate knowledge about the relative motion between the previous C-space map and the current C-space map. Under this shift, GMM parameters such as the Gaussian mean parameters, are remapped as follows. First, the position of a virtual end-effector, corresponding to the configuration of one of the Gaussian means is computed by Forward Kinematics (FK). Then, the position of the end-effector is transformed by the relative motion, and the resulting configuration is computed by Inverse Kinematics (IK). Hence, the changed configuration results in a transformed mean parameter for a GMM in the new C-space. The transformed model is then updated with new exemplars as the GMM-based RRT plans in the changed environment. Thus, in our proposed method, the current GMM models are initialized using information from the prior models. As a result, this approach can dramatically improve the efficiency of the GMM-based RRT.

Our approach differs from other traditional methods for adaptive motion planning. We exploit a compact representation model via the GMMs for fast collision detection in the high dimensional C-spaces. These models are rapidly initialized from prior experience using kinematic transformations and then updated with new exemplars.
during the planning iterations. In contrast, other approaches exploit the old tree structures generated for past path plans [148] or construct graph structures for the feasible end-effector positions [61]. For large changes in robot pose, however, it will be difficult to directly warp previously planned trajectories to the new C-space (see Fig.7.8), and these methods will only be successful under restricted environmental conditions.

In summary, the core contributions of this section include 1) an adaptive motion planning routine incorporating an efficient online prediction of high-dimensional mixture model parameters for changing environments; 2) an algorithm to update the predicted GMMs with new exemplars; and 3) utilization of the learned GMMs to efficiently bias the RRT in the new environment. With this approach, the initial GMMs can be rapidly adapted to environmental changes using a small number of training examples, allowing for fast motion planning of manipulators in various applications.

7.2.1 Problem formulation

We aim to develop an adaptive motion planning with learned high-dimensional Gaussian models in the changed environment. Whenever the robot moves or the environ-
ment is changed, the topological representation of collision and collision-free regions in C-space are changed; thus, the Gaussian models are accordingly transformed in the C-space. However, it is not straightforward to estimate the transformation in the C-space from the information in 3D Euclidean space. However, we can exploit the compact representation of the GMMs, which is determined only by the mean vector and covariance matrix, in order to estimate its transformation.

We assume that the relative motion between the robot base and the workspace can be estimated by the change of previous and current world models, and the relative motion is used to predict the collision and collision-free regions in the new C-space. In addition, the robot repeatedly conducts various planning tasks in the same workspace and sometimes moves slightly in order to work more efficiently. When the robot has moved, the relative distance between the robot and the workspace is changed. The problem is that the transformation of relative motion is not directly applied to estimate the change of GMMs in the new C-space. If we can predict the changed parameters from the GMMs in the previous C-space, we can use them to update the new GMMs. However, it is difficult to predict the representation of the new GMMs in the C-space analytically by using the relative motion between the robot and the workspace. Therefore, we suggest an approach to predict the transformation of the GMMs via IK of the manipulator. In addition, since the predicted model has uncertainty, we also propose an online update approach where the newly predicted model is converged to the accurate model in the new C-space.

7.2.2 Adaptive GMM-based RRT

In this section, we suggest an adaptive GMM-based RRT for fast motion planning in the changed environment. As can be seen in Fig. 7.9, since the representation of collision regions in the C-space is transformed to the new regions due to relative motion, the previous GMMs are transformed to the new representation. In addition, the change of topological structure of the collision region in the C-space can be represented by the transformation of the GMMs through the prediction and update procedure. Since transformations are non-isometric in the C-space, the number of
Algorithm 10 Adaptive GMM-based RRT

Require: $G_{pos}, G_{neg}, M_{camera}$

1. $R, t \leftarrow \text{ESTIMATE MOTION}(M_{camera}, M_{new})$
2. $G'_{pos} \leftarrow \text{PREDICT GMM}(G_{pos}, R, t)$
3. $G'_{neg} \leftarrow \text{PREDICT GMM}(G_{neg}, R, t)$
4. $G_{pos} \leftarrow G'_{pos}, G_{neg} \leftarrow G'_{neg}$
5. $\mathcal{T}.init(q_{init})$
6. while Distance($q_{goal}, q_{new}$) > $d_{min}$ do
7.   $q_{rand} \leftarrow \text{StateSampling}(G_{neg})$
8.   $q_{near} \leftarrow \text{NodeSelection}(\mathcal{T}, q_{rand})$
9.   $q_{new} \leftarrow \text{NodeExpansion}(\mathcal{T}, q_{rand}, q_{near})$
10. $d_{pos}, d_{neg} \leftarrow \text{MahalanobisDist}(q_{new}, G_{pos}, G_{neg})$
11. if $(d_{pos} - d_{neg}) < \delta_{pos}$ then
12.    $q_{new}$ is collision
13. else if $(d_{pos} - d_{neg}) > \delta_{neg}$ then
14.    $q_{new}$ is collision-free
15. else
16.    $X_{new} \leftarrow \text{KinematicCollisionCheck}(q_{new})$
17. end if
18. if $q_{new}$ is collision-free then
19.    $\mathcal{T}.addTree(q_{new})$
20. end if
21. end while
22. CollisionFreeCheck($X_{pathset}$)
23. $G_{pos}, G_{neg} \leftarrow \text{UPDATE GMM}(G_{pos}, G_{neg}, X_{new})$

Gaussian mixtures might need to be changed by merging or adding. We use the Bhattacharyya distance that measures the similarity of the probabilistic distributions for this procedure [58].

Prediction of GMMs

In order to estimate the change of the collision and collision-free regions, we need to predict the transformation of the Gaussian distribution, i.e., the linear transformation of the Gaussian models in the C-space by the homogeneous transformation in 3D Euclidean space. Our proposed method predicts the transformed collision and collision-free distributions in C-space based on the change of the end-effector in the Euclidean space. As illustrated in Fig. 7.10, the transformed collision configuration can be obtained indirectly from the configuration of the manipulator.
The relative distance and position between the robot and the workspace are also changed after the robot has moved. This relative motion is obtained by the homogeneous transformation in 3D Euclidean space obtained from the changed world model. Once the relative motion between the robot and the workspace is observed, we can exploit this for computing the new desired arm configuration and thus for prediction of the transformed Gaussian parameters. The collision C-space and non-collision C-space are decided by the collision of arm configurations, and the collision is decided by the relative distance between the robot and obstacles. Therefore, we can exploit the relative motion for prediction of the transformed Gaussian parameters. In other words, when the robot approaches obstacles, the position of the end-effector should be closer to the obstacles. Therefore, the transformed Gaussian mean and covariance are computed by the shifted arm configuration.

In order to predict the new mean, our method first solves the FK to compute the desired arm configuration corresponding to one of the Gaussian means. Thereafter, the changed position of the end-effector is estimated by the 3D homogeneous transformation and then, the changed configuration is computed by IK. The resulting configuration can be the transformed Gaussian mean. Finding the configuration corresponding to the end-effectors at the desired position is the IK problem.

However, the IK problem for a 7DoF manipulator has an infinite number of solutions and thus requires an optimal approach suitable for an objective function. In
order to solve the IK, there have been many approaches [132, 130]. Among them, we need to find a solution that has the lowest accumulated joint variance. We find the solution that minimizes the variance of the configuration from the original configuration by the numerical approach [144]. Since the GMMs represent collision and collision-free probabilistic distributions of the manipulator, not only the end-effector part, the GMMs preserve information of collision and collision-free distributions of the manipulator, although they are transformed by the suggested IK approach.

Furthermore, the variance is propagated in the prediction step by taking the uncertainty of the prediction. We assume that the uncertainty is zero mean Gaussian with covariance that is dependent on the change of the mean vector. Therefore, it is proportional to the change of the mean vector, and it is additive to the covariance matrix. Fig. 7.11 shows the predicted GMMs from previous GMMs by the prediction step. As can be seen in Fig. 7.11, the lower level joints (the 5th and 6th) are noticeably transformed compared to the upper level joints (the 1st and 3rd). This is a reasonable result because the lower level joints are more affected by the environmental change.
Figure 7.11: Visualization of predicted GMMs from previous GMMs by applying IK: the lower level joints (5th and 6th joints) are noticeably transformed compared to the upper level joints (1st and 3nd). 2-dimensional space is projected from 7-dimensional space for visualization purposes.
Online Update

After prediction, the GMM parameters are updated with the new exemplars obtained from the kinematic-based collision check during the GMM-based RRT procedure online. Since the new exemplars obtained from every iteration are not plentiful enough to generate Gaussian models, the GMMs are updated with the new exemplars by the Maximum Likelihood Estimation (MLE) until sufficient exemplars are gathered. Each exemplar is assigned to a GMM cluster in the predicted GMMs by comparing the probability of each Gaussian model, after which, the Gaussian parameters are updated by the MLE with new exemplars.

After gathering sufficient exemplars, observation models, which are assumed to be Gaussians, are generated by the new exemplars. Here, the clustering is conducted based on Mahalanobis distances of the new exemplars from the predicted GMMs. The GMMs can be updated by the Bayes rule, given

\[ X_{\text{pred}} \sim \mathcal{N}(\mu_p, \Sigma_p), \quad X_{\text{obs}} \sim \mathcal{N}(\mu_o, \Sigma_o), \]

as

\[ X_{\text{update}} \sim \mathcal{N}(\Sigma_p + \Sigma_o \mu_p + \Sigma_p + \Sigma_o \mu_o, \frac{1}{\Sigma_p^{-1} + \Sigma_o^{-1}}). \]

Hence, our prediction and update can be conducted in an online manner adapting to changing environments every iteration.

Relative motion from camera

Let us review the perception stage of this work to obtain the dynamic motion of the robot relative to the world. It is assumed that a prior world map is prepared, which we built by cleaning 3D point clouds captured from an RGBD camera. We used the depth channel of the camera and the Iterative Closest Point (ICP) algorithm to estimate the rotation and translation of the robot. To make this problem tractable, we assume that the initial guess of the pose at the very first frame is reasonably good, and that the motion changes gradually afterward.
Let $\mathbf{R}_c^t_k$ be the rotation matrix representing the target orientation with respect to the camera frame, observed at time $k$, and $\mathbf{t}_c^t_k$ be the position of the same. The relative motion observed can be written as,

$$
\Delta \mathbf{R}_c^t_k = \mathbf{R}_c^t_k \mathbf{R}_c^{t_k-1}^T
$$

$$
\Delta \mathbf{t}_c^t_k = \mathbf{t}_c^t_k - \mathbf{t}_c^{t_k-1}
$$

Note that, by changing the superscript from $c$ to $b$, we may express the same quantities with respect to the body frame as $\mathbf{R}_b^t_k$ and $\mathbf{t}_b^t_k$. Given $\mathbf{R}_b^c$, the transformation from the body frame to the camera frame, we can transform the observed rotation and translation from ICP as,

$$
\Delta \mathbf{R}_b^t_k = \mathbf{R}_b^c \Delta \mathbf{R}_c^t_k \mathbf{R}_b^c
$$

$$
\Delta \mathbf{t}_b^t_k = \mathbf{R}_c^b \Delta \mathbf{t}_c^t_k
$$

Thus, once a new motion increment is estimated from the point cloud measurement, the change is interpreted in the body frame and then used for the prediction of the GMMs by means of IK.

**Path smoothing**

Another advantage of the GMM-based RRT is that it requires less computational time for the collision detection. Therefore, it can be used for path smoothing after planning by the RRT. Sample-based planners are fast, but they sometimes generate jerky and unnatural paths. Thus, the RRT needs smoothing methods such as the shortcut or spline algorithms [53, 48]. The shortcut method is widely applied for path smoothing since it can be implemented effectively. However, it requires expensive computation time since the feasibility of a local path between two random points must be checked at each step. In addition, since the computational cost depends on the level of smoothing of the path, a large number of collision checks are required for the high quality smoothing path. Since the collision check based on GMMs is very fast compared to the ordinary collision check based on the kinematics, it can be
effectively applied to the shortcut smoothing method.

7.2.3 Experiments

This section provides a series of experimental results which validate the proposed approach. First, we evaluate the GMM-based RRT and compare it with traditional approaches in a real experimental environment. Experiments are performed using a 7DoF manipulator and Matlab on a PC with an Intel 2.7GHz. An ASUS Xtion Live Pro camera was used for 3D perception of the environment. For comparison, we implemented two traditional approaches incorporating the kinematic-based collision detection algorithm with the Octree structure representation of Axis-Aligned Bounding Box (AABB). Moreover, we evaluate the performance of the adaptive planning with updated mixture models in the changed environment.

Learning GMMs and test of the GMM-based RRT

First, the initial GMMs are learned via an incremental EM with the exemplars gathered from 10 iterations of the ordinary RRT. The initial and goal configuration for training is shown in Fig. 7.12(a). In order to evaluate the performance of the GMM-based RRT, we set a test scenario that consists of consecutive paths as shown in Fig. 7.12(b). Note that the last two steps (the 8th and 9th) are longer and complicated than the others, and that the learned model has been built upon a simpler case compared to the test scenario. Starting from the initial model, the GMMs are continuously updated online with new exemplars obtained from the GMM-based RRT.

Fig. 7.13 shows the result of the comparison between the GMM-based RRT and traditional approaches. In order to accurately evaluate the performance, we repeat the experiments 10 times and take the average of execution time, number of collision checks, and path lengths. As can be seen in Fig. 7.13, the GMM-based RRT has a better performance compared with the basic RRT and the bidirectional RRT. The GMM-based RRT is two times faster than the bidirectional RRT, and the number of collision checks of the GMM-based RRT is also less than that of the bidirectional
Figure 7.12: The training and test scenarios: (a) in order to train the model, the collision and collision-free exemplars are obtained from the ordinary RRT of this trajectory (b) the test is conducted with the learned GMMs in this scenario.
Figure 7.13: The comparison results with the bi-directional RRT: the GMM-based RRT is relatively faster in the complicated path planning.
Figure 7.14: Execution time of the GMM-based RRT as the model update proceeds

RRT. In addition, the GMM-based RRT has a similar length of trajectory of the end-effector.

It is noticeable that as the path becomes complicated, as in the 8th and 9th cases, the execution time of the bi-directional RRT increases drastically, as shown in Fig. 7.13(a). On the other hand, the execution times of the GMM-based RRT for the same cases increases comparatively less than with other approaches. This shows the robustness and efficiency of our method under the increased complexity of the desired trajectory. Summing up these results, the GMM-based RRT is more efficient and more effective than the bidirectional RRT.

Result of the online update

The GMMs are continually updated by MLE with new exemplars every RRT iteration. It is an advantage of the GMM-based RRT, since the model can be more accurate with new exemplars obtained from every iteration. Ten iterations of the GMM-based RRT are conducted with the updated models in order to statistically evaluate the accuracy and the performance of the online learning. As shown in Fig. 7.14, the execution time decreases as the model is updated every step. The first update is the result of the initial GMMs learned from the training scenario shown in Fig. 7.12(a), and the others show the results of the GMMs after each online update. We can see that the GMMs quickly converge after the 3rd or 4th online update procedure. This
In order to evaluate the performance of adaptation to the changed environment, we tested two cases as shown in Fig. 7.15. In the first case, the shelf is moved 0.1m in the $x$ direction and 0.1m in the $y$ direction, as shown in Fig. 7.15(a), and 0.1m in the $x$ direction and 0.1m in the $y$ direction and then rotated 30 degrees, as shown in Fig. 7.15(b). In this testbed, the robot approaches the shelf, which reduces the workspace of the manipulator; thus, the planned trajectories can vary significantly even by small changes, as shown in Figure 7.8. Therefore, our experiment setup is appropriate for performance evaluation under a changed environment.

Fig. 7.16 shows the comparison result of the GMM-based RRT with the bidirectional RRT in the changed environment. The test scenario is similar to Fig. 7.12(b) except that the given configuration is slightly changed to avoid collision. The experiment is repeated 10 times to perform more accurate evaluation. As can be seen in Fig. 7.16, the GMM-based RRT in the changed environment still maintains a
Figure 7.16: Comparison results with the bidirectional RRT: error bars represent one standard deviation of 10 iterations.
Figure 7.17: Performance results with the previous GMMs and the predicted GMMs: (a) scenario (b) comparison results

performance that is faster than the bidirectional RRT. Moreover, the variance of the execution time and the number of collision checks are remarkably less. Thus, being able to adapt online, our method works effectively under changes in the environment.

In order to evaluate the performance of incorporating the prediction step, we show the comparison result of the RRT with the bias sampling based on the previous GMMs and the predicted GMMs in Fig. 7.17. In this scenario, the robot approaches 10cm closer to the obstacles after generating the previous GMMs and compares execution times of the RRT planning for the task where the manipulator is inserted into the shelf. The simulation is repeated 50 times to perform more accurate evaluation. As shown in Fig. 7.17(b), the performance is degraded with the previous GMMs, and transformation of GMMs based on the prediction step results in a more efficient RRT.

**Test result with novel path scenario**

Now, the given sequence of the trajectories for updating GMMs is the same as that for the evaluation. In this section, we evaluate the performance of the GMM-based RRT for a novel path sequence from the training paths. The sequential configurations for the test are shown in Fig. 7.18, and the test procedure is (a) → (b) → (c) → (d)
Figure 7.18: The novel test scenario is composed of different configurations from the training and previous test configurations.

→ (e) → (f) → (a). The first purpose of this test is to verify that the updated model can be exploited in path planning for any initial and goal configurations in the same environments and second, to show that the GMM-based RRT with the updated model works effectively for different paths from the training.

As can be seen in Fig. 7.19, the performance of the GMM-based RRT is better than the bidirectional RRT. The average computation time for the GMM-based RRT is 40.01s, and it is 80.36s for the bidirectional RRT on the example of Fig. 7.18. Moreover, it also has a small variance in the time and the number of collision checks. Since the test sequence was not given before, this test result means that the GMM-based RRT can conduct path planning fast and efficiently for arbitrarily given initial and goal configurations. In addition, the model is predicted and updated according
to the environment change; therefore, it shows that the GMM-based RRT is applied to the changed environment.

Finally, with the path smoothing approach, the total path length is reduced by about 6.2%, taking 3.51 seconds to plan, and the paths have considerably less jerky motion.

### 7.2.4 Discussion

This section has presented a fast adaptive sampling based motion planning algorithm with high-dimensional mixture models to adapt to environmental changes. The proposed approach is based on the GMM-based RRT, which incorporates fast probabilistic collision detection in high-dimensional C-spaces. The learned GMMs are adapted via inverse kinematics and are continually updated and refined as the RRT planning algorithm proceeds. Additionally, since our method is refined in an online manner after the initial learning of GMMs, it can be applied effectively and efficiently to time-varying scenarios. We show that the adaptive GMM-based RRT displays performance gains up to five times faster than traditional approaches in some common arm planning scenarios. Thus, we believe the proposed method shows much promise in making robots more responsive and agile in a variety of tasks.
Chapter 8

Comparison and Conclusion

8.1 Comparison

Sampling-based planning approaches can find a feasible solution quickly and work well in high-dimensional space. However, to pass through narrow passages, a lot of samples are required by increasing the number of nodes; thus, it could be very slow in a complicated environment even though it finds solutions very fast in uncluttered space. Although it guarantees probabilistic completeness, it could fail nevertheless to find a solution within a given time. This performance degrade is usually considered to be an issue of sampling, because uniform sampling has a Voronoi bias towards yet unexplored larger regions of configuration spaces. Therefore, it is hard for sampling-based planners to find a solution in complicated environments. GMM-RRT is able to perform better compared to traditional sampling-based planning, since it uses the learned representation model of configuration space for sampling and local steering of sampling-based planning. PRM and RRT are popular sampling-based approaches each with its own advantages and shortcomings. Compared to PRM and RRT, GMM-RRT is a novel approach to resolving their shortcomings; PRM requires a long learning step, and RRT plans a new path from scratch although it plans within the same environment. In this section, we briefly explain two sampling-based approaches: Single-Query approach, e.g. RRT, and Multi-Query approach, e.g. PRM, and then compare them with GMM-RRT.
8.1.1 PRM

Compared to RRT, Probabilistic Roadmap (PRM) requires a learning phase to construct a probabilistic roadmap. It generates random vertices, checks the connectivity between the two vertices (local planner), and stores them as a graph structure. The performance depends on a good sampling method for vertices. PRM requires a lot of collision checks for evaluating the validity of vertices and connectivity of edges. Since the collision vertices and invalid edges are removed after collision checks, it requires a lot of samples and collision checks for construction in a complicated environment, as shown in Fig. 8.1(a). In addition, PRM, like RRT, has the problem in that the nearest neighbor search could be slower when increasing the number of vertices [142]. The critical problem of PRM is that the constructed graph could be disconnected into several parts and thus with some start and goal configurations, be unable to find a solution. To prevent this disconnectivity, the PRM approach suggests an expansion step; the higher the weight, the higher the chances the node will get selected for expansion. In the expansion step, it selects a node to expand based on the weight which is decided based on the number of nearby nodes within a given distance or failure rate of connectivity. However, this expansion step can’t guarantee the connectivity
of the roadmap. The heuristic approach is to increase the number of vertices sufficiently. However, this has an weakness in that it requires heavy construction time and many collision checks. As shown in Table 8.1, the number of collision checks and the computation time increase exponentially as the number of vertices increases. After constructing the roadmap, the query phase finds a path using a sequence of edges in the roadmap by a graph search algorithm, such as A*. If it fails to find a feasible sequence of edges due to disconnectivity, it continues to increase random vertices or the connectivity distance between vertices until it can find a path. For a complete roadmap, it spends heavy computation time in the learning phase or requires another learning phase whenever it fails to find a solution, but it could be very time consuming, as shown in Fig. 8.2.

8.1.2 RRT and PRM

Compared to RRT, PRM requires a learning phase which generates random vertices and checks the connectivity between nodes. In the complicated environment, it is slow due to a lot of collision checks for connectivity. In addition, since it requires memory to maintain this constructed graph structure, it requires a large memory in high dimensional space. After constructing the roadmap, it can find a solution in the graph relatively quickly compared to RRT. However, it could have no solution due to disconnectivity in the graph. Compared to PRM, RRT requires no learning phase, but the planning is relatively slower, since it plans a path from scratch for every planning task. However, it can find a solution if one exists, and it is unnecessary to use memory to save the tree structure. Both RRT and PRM have the difficulty of being exponentially slow as the number of vertices or nodes increases due to nearest neighbor search. Therefore, the number of nodes could be another bottle neck in high dimensional and complicated environments; thus it would be highly beneficial if we can reduce the number of nodes.
Figure 8.2: PRM result: The vertices are randomly generated for constructing the roadmap. When the number of vertices is less than 300, it fails to find a path between the start (yellow) and goal (green) configurations.
8.1.3 GMM-RRT and PRM

While PRM requires many collision checks for graph structure and requires a large memory for a graph structure in high dimensional space, GMMs for GMM-RRT requires relatively smaller samples for modeling compared to PRM. Since it uses samples obtained from RRT planning, it has an advantage in that it can update the model as the RRT planning proceeds. In addition, it requires little memory for mixture of Gaussians in high dimensional space compared to the graph structure of PRM.

To briefly compare the computation cost of the learning phases of the GMM-RRT and PRM methods, we provide in Table 8.1 the average computation time for the GMM and PRM constructions for 2DoF planar manipulator planning (Fig. 8.1). As expected, for the same number of samples, GMM learning is around two orders of magnitude faster than the PRM construction, because the connectivity test of the PRM approach is significantly costlier computationally than the statistics computation of GMM. In this test, PRM has the distance threshold to connect between vertices within 0.25 of the maximum length of the space. PRM can construct a complete roadmap of more than 300 vertices which can find a solution of any given start and goal configuration. However, since PRM has a weakness in that it is unable to decide the number of vertices for a complete roadmap, we usually generate sufficient random vertices. As a result, the learning of PRM is slower, because it requires more time and collision checks as the number of vertices increases. As seen from Table 8.1, PRM requires more than 80,000 collision checks for a complete roadmap with 300 vertices. Although we cannot compare the suggested approach with PRM directly, we can see that it requires fewer samples (4,000) for GMMs.

8.1.4 GMM-RRT and Lazy PRM

While the original PRM is unable to decide the number of vertices for constructing the roadmap, Lazy PRM does not need to decide the number of vertices, since it updates the roadmap whenever it is unable to find a solution. Lazy PRM is suggested
Table 8.1: GMM and PRM computation times

<table>
<thead>
<tr>
<th>Num. of Samples</th>
<th>Sampling Time</th>
<th>GMM Time</th>
<th>Total Time</th>
<th>Num. of Vertices</th>
<th>PRM Time</th>
<th>Collision Checks</th>
<th>Connected PRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>0.1665</td>
<td>0.0489</td>
<td>0.2154</td>
<td>100</td>
<td>2.4750</td>
<td>7.983</td>
<td>No</td>
</tr>
<tr>
<td>500</td>
<td>0.2023</td>
<td>0.1220</td>
<td>0.3244</td>
<td>150</td>
<td>5.4377</td>
<td>18.361</td>
<td>No</td>
</tr>
<tr>
<td>1,000</td>
<td>0.3855</td>
<td>0.2632</td>
<td>0.6487</td>
<td>200</td>
<td>10.3674</td>
<td>35.306</td>
<td>No</td>
</tr>
<tr>
<td>2,000</td>
<td>0.7640</td>
<td>0.4236</td>
<td>1.1876</td>
<td>250</td>
<td>16.0856</td>
<td>55.517</td>
<td>No</td>
</tr>
<tr>
<td>4,000</td>
<td>1.5159</td>
<td>0.8545</td>
<td>2.3704</td>
<td>300</td>
<td>23.0590</td>
<td>81.274</td>
<td>Yes</td>
</tr>
<tr>
<td>6,000</td>
<td>2.2659</td>
<td>1.2205</td>
<td>3.4904</td>
<td>350</td>
<td>30.2114</td>
<td>107.841</td>
<td>Yes</td>
</tr>
<tr>
<td>8,000</td>
<td>2.8811</td>
<td>1.4230</td>
<td>4.3011</td>
<td>400</td>
<td>38.6380</td>
<td>134.851</td>
<td>Yes</td>
</tr>
<tr>
<td>10,000</td>
<td>3.6009</td>
<td>1.6100</td>
<td>5.2109</td>
<td>450</td>
<td>49.1138</td>
<td>171.122</td>
<td>Yes</td>
</tr>
</tbody>
</table>

to minimize the number of collision checks and reduce learning time. It handles applications in high-dimensional but relatively uncluttered configuration space. Lazy PRM reduces the number of collision checks by avoiding local planning. It checks and explores only essential parts of the configuration space. The advantage of Lazy PRM is that it can reduce the number of collision checks in an uncluttered environment. Since a roadmap of paths is assumed to be feasible, Lazy PRM updates the roadmap by removing infeasible nodes and by checking the edges of the essential parts of planning given start and goal points. Although it can apply for a relatively uncluttered configuration space and minimize the number of collision checks, it could be worse in cases of cluttered configuration space and narrow passages. It spends a lot of time updating the graph structure, if it repeats many update steps such as in cluttered and narrow passages. In addition, the critical problem is that removing an edge could divide the graph into two components which are unconnected to each other, so it is very expensive to check for graph connectivity. Fig. 8.3 shows the Lazy PRM planning result in a 2DoF environment, as shown in Fig. 8.1. We average the time after 50 iterations for 20 planning cases in a 2D environment. The planning time of Lazy PRM is very slow at the beginning, since it repeatedly removes invalid edges and updates the roadmap in a complicated environment. The average time of the first planning takes more than 50 seconds due to reconstructing the roadmap; it is very slow compared to our learning and planning approach which takes less than a second. Therefore, it is clear that the Lazy PRM approach is inefficient in the complicated environment compared to GMM-RRT.
Figure 8.3: Performance curve of Lazy PRM: (a) execute time (b) collision checks

Table 8.2: Comparison of GMM-RRT with several approaches

<table>
<thead>
<tr>
<th>Method</th>
<th>Learning Phase</th>
<th>Learning Time</th>
<th># samples for learning</th>
<th>Success of planning</th>
<th>Planning time</th>
<th># collision checks for planning</th>
<th>Memory for model</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM-RRT</td>
<td>Require</td>
<td>Short</td>
<td>Small</td>
<td>Always</td>
<td>Relatively fast</td>
<td>Small</td>
<td>Small</td>
<td>Gaussian Mixtures</td>
</tr>
<tr>
<td>RRT</td>
<td>No</td>
<td>-</td>
<td>-</td>
<td>Always</td>
<td>Relatively slow</td>
<td>Large</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PRM</td>
<td>Require</td>
<td>Long</td>
<td>Large</td>
<td>Sometimes fail</td>
<td>Fast**</td>
<td>-</td>
<td>Large</td>
<td>Graph structure</td>
</tr>
<tr>
<td>Lazy PRM</td>
<td>Require</td>
<td>Relatively short*</td>
<td>Relatively low*</td>
<td>Always</td>
<td>Fast**</td>
<td>Small*</td>
<td>Large</td>
<td>Graph structure</td>
</tr>
</tbody>
</table>

* The Lazy PRM requires heavy computation and collision checks in complicated environments.
** When the roadmap is not connected, it requires the update of roadmap.

8.1.5 GMM-RRT and RRT

The advantage of the original RRT compared to PRM or GMM-RRT is that it does not require learning time or memory for the graph structure. However, the traditional RRTs are slower than GMM-RRT and require many collision checks. The performance is always the same although it repeats planning trials within the same environment since it plans a path from scratch every time. On the other hand, GMM-RRT learns the probabilistic model with samples obtained from RRT planning and plans faster with fewer collision checks. Therefore, the performance significantly increases as RRT planning repeats as the model updates, since the learned model can be used for fast collision checks, biased sampling, and guidance of local steering, as shown in previous chapters. We have already shown that Gaussian mixture learning has been applied to construct approximate probabilistic models of collision and collision-free subspaces of a C-space for fast collision checking and biased sampling over free space. In addition, we have introduced a new probabilistically safe local steering primitive for sampling-based motion planning in complex high-dimensional C-spaces.
Our local steering procedure is based on the novel notion of a convex probabilistically safe corridor that is constructed around a configuration using tangent hyperplanes of confidence ellipsoids of Gaussian mixture models learned from prior collision history. Through these approaches, we show that GMM-RRT is superior to traditional RRT approaches.

8.1.6 Discussion

Table 8.2 shows the comparison of GMM-RRT with other sampling-based approaches. Although GMM-RRT requires learning of GMM, it requires less time and fewer samples compared to PRM, and it is faster than RRT approaches. In addition, it is more powerful in a complicated environment and in narrow passages. While PRM can not find a solution if the vertices are insufficient to fully connect a graph, GMM-RRT can always find solutions. In addition, Lazy PRM is very slow in a complicated environment and although it takes a very similar approach to GMM-RRT by updating the roadmap, it is clear that GMM-RRT is more efficient than Lazy PRM. In conclusion, we recommend GMM-RRT for the conditions which require a complicated environment and repeated planning.

8.2 Conclusion

This thesis presented a framework for resolving several issues of sampling-based planning based on learning of a probabilistic model and representation of configuration space with that probabilistic model.

First, we presented a new approach for fast collision detection in high dimensional configuration spaces for RRT motion planning based upon Gaussian Mixture Models (GMM) learned using an incremental Expectation Maximization clustering algorithm. This model is trained online using exemplars provided by a slow, conventional kinematic-based collision detection routine.

We also suggested a learning approach for efficient sampling of RRTs based upon a learned Q-function. We employ several features in learning the Q-function, includ-
ing Radial Basis Function (RBF) for the representation of collision and collision-free regions in the C-space. We also showed the extension of QS-RRT including nonholonomic constraint. Our sampling method chooses the optimal node to extend the tree via the learned state value computed from the node feature representation. Our softmax node selection procedure avoids becoming stuck at local minima and maintains the completeness property of RRTs.

For effective tree extension, we first constructed a probabilistic safety corridor using the margin of the collision space of Gaussian Mixture Models (GMMs). The probabilistic safety corridor is a polyhedron composed of tangent half spaces on the surfaces of ellipsoids representing a given probabilistic confidence level. With a polyhedron of constructed safety space, we offer an optimization-based tree extension algorithm to increase connectivity with less likelihood of collision. Since the tree extends within the probabilistic safety space, this approach can dramatically improve efficiency by reducing the number of collisions.

For grasp and transport tasks with a redundant robotic manipulator, we present a novel constrained, sampling-based motion planning method. We utilize a planning margin for grasping with constraints that allow the best grasp configuration and approach direction to be determined automatically. Our method also considers transporting the grasped object to the desired target position using the RRT algorithm that incorporates soft constraints of end-effector via appropriate cost penalties.

Next, we propose approaches for handling a changed environment and online learning to refine and update the model. We present a novel adaptive approach to fast sampling-based motion planning by learning models of collision and collision-free regions in configuration spaces in an online manner based on a Dirichlet process mixture model. In addition, in practical applications for robotic manipulation, the representation of collision and collision-free regions in configuration space can change due to relative motion between the robot base and workspace. We show how to rapidly adapt to such changes by using inverse kinematics to transform the parameters of the Gaussian mixture model to new configurations. The transformed model is initially used as a prior and then continually updated and refined as the RRT planning
One drawback of our framework is that it requires a learning phase for the probabilistic model. However, as shown in the comparison results, the learning phase of our approach spends less computation time with fewer collision checks. In addition, although there is a variation of PRM for adapting to a changed environment, Lazy PRM, it is slower in a complicated environment. Although GMM-RRT requires learning of GMM, it takes less time and fewer samples compared to PRM, and it is faster than RRT approaches. While PRM can not find a solution if the vertices are insufficient to fully connect a graph, GMM-RRT can always find solutions. In addition, while Lazy PRM is very slow in a complicated environment, our approach is more powerful in a complicated environment and in narrow passages. We suggested our approach for RRTs to achieve several planning missions efficiently using learning of probabilistic generative models in complicated high-dimensional environments. In some practical applications, multiple path trajectories are required for various tasks within the same or slightly changed environment. In these cases, the planner needs to be continually improved as the planning algorithm proceeds. Therefore, we recommend GMM-RRT for conditions which require a complicated environment and repeated planning.
Bibliography


[135] P. Vernaza and D. D. Lee. Learning dimensional descent for optimal motion
planning in high-dimensional spaces. In Proc. AAAI Conference on Artificial
Intelligence, 2011.


[137] M. V. Weghe, D. Ferguson, and S. S. Srinivasa. Randomized path planning
for redundant manipulators without inverse kinematics. In Proc. IEEE-RAS

[138] R. Wein, J. van den Berg, and D. Halperin. Planning high-quality paths and
corridors amidst obstacles. The International Journal of Robotics Research, 27

based RRT path planner for non-holonomic robots. Journal of Intelligent and


[142] A. Yershova and S. M. LaValle. Improving motion-planning algorithms by
efficient nearest-neighbor searching. IEEE Transactions on Robotics, 23(1):

Efficient exploration by controlling the sampling domain. In Proc. IEEE In-
ternational Conference on Robotics and Automation (ICRA), pages 3856–3861,
2005.

168


