Multi-Objective Reactive Motion Planning in Mobile Manipulators

Multi-Objective Reactive Motion Planning in Mobilen Manipulatoren
Master thesis by Jiawei Huang
Date of submission: November 11, 2021

1. Review: M.Sc. Julen Urain De Jesus
2. Review: Prof. Dr. Jan Peters
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Jiawei Huang, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

JiaweiHuang

Darmstadt, 11. November 2021

Jiawei. Huang

TECHNISCHE UNIVERSITÄT DARMSTADT

Master-thesis

für

Mr. Jiawei Huang, B.Sc.

Theme: Multi-Objective Reactive Motion Planning in Mobile Manipulators

Multi-Objective Reactive Motion Planning in Mobilen Manipulatoren

Performing a complex task requires to satisfy multiple objectives in parallel. A relatively simple task like opening a door, requires a robot to properly approximate the door, grasp the handle or avoid colliding against the door while additionally, avoiding to reach joint limits or exceed the joint torques. During this thesis, we are interested in developing algorithms to satisfy multi-objective reactive motion planning in mobile manipulators. We aim to achieve multi-objective planning by Composable Energy Policies, a novel algorithm that frames Multi-Objective Reactive Motion Planning as an inference problem. Composable Energy Policies will allow us to compose classical Motion Planning algorithms with Reinforcement Learning or with Imitation Learning-based policies.

Begin:	10.05.2021
Submit:	10.11.2021

Supervisor: Julen Urain De Jesus, M.Sc. (Intelligent Autonomous Systems)
Prof. Dr.rer.nat. Debora Clever (Institute for Mechatronic Systems)
Prof. Dr. Jan Peters (Intelligent Autonomous Systems)

10.05.2021 [illegible]

Prof. Dr.rer.nat. Debora Clever
Institute for Mechatronic Systems

Prof. Dr. Jan Peters
Intelligent Autonomous Systems

12.05.2021 Jiawei Huang

I would like to register my master thesis.

TECHNISCHE UNIVERSITÄT DARMSTADT

Master-thesis

für

Mr. Jiawei Huang, B.Sc.

Theme: Multi-Objective Reactive Motion Planning in Mobile Manipulators

Multi-Objective Reactive Motion Planning in Mobilen Manipulatoren

Performing a complex task requires to satisfy multiple objectives in parallel. A relatively simple task like opening a door, requires a robot to properly approximate the door, grasp the handle or avoid colliding against the door while additionally, avoiding to reach joint limits or exceed the joint torques. During this thesis, we are interested in developing algorithms to satisfy multi-objective reactive motion planning in mobile manipulators. We aim to achieve multi-objective planning by Composable Energy Policies, a novel algorithm that frames Multi-Objective Reactive Motion Planning as an inference problem. Composable Energy Policies will allow us to compose classical Motion Planning algorithms with Reinforcement Learning or with Imitation Learning-based policies.

Begin:	10.05.2021
Submit:	10.11.2021

Supervisor: Julen Urain De Jesus, M.Sc. (Intelligent Autonomous Systems)
Prof. Dr.rer.nat. Debora Clever (Institute for Mechatronic Systems)
Prof. Dr. Jan Peters (Intelligent Autonomous Systems)

Prof. Dr.rer.nat. Debora Clever
Institute for Mechatronic Systems

Prof. Dr. Jan Peters
Intelligent Autonomous Systems

Jan Peters

12.05.2021 Jiawei Huang

I would like to register my master thesis.

Abstract

The combination of path planning and trajectory optimization considers the motion planning as an optimization problem, instead, Reactive Motion Generation produces appropriate actions that fulfill different policies in parallel, which can also deal with dynamic objects. However, deterministic policies might cause local minima, such as potential fields methods. Moreover, motion planning of mobile manipulators is required to take various policies into account. And hence, we propose to apply a novel motion generation framework, Composable Energy Policies (CEP) in mobile manipulators.
In this thesis, we employ CEP framework in mobile manipulators given a prior map generated by Rapid Random Search Tree (RRT), and compare different methods to find a local target. We found our method can perform 2D planning and whole body control well.

Zusammenfassung

Die Kombination von Bahnplanung und Trajektorienoptimierung betrachtet die Bewegungsplanung als Optimierungsproblem, stattdessen erzeugt die reaktive Bewegungsgenerierung entsprechende Aktionen, die parallel verschiedene Richtlinien erfüllen, die auch mit dynamischen Objekten umgehen können. Deterministische Richtlinien können jedoch lokale Minima verursachen, wie z. B. potenzielle Feldmethoden. Darüber hinaus muss die Bewegungsplanung mobiler Manipulatoren verschiedene Richtlinien berücksichtigen. Daher schlagen wir vor, ein neuartiges Framework zur Bewegungserzeugung, Composable Energy Policies (CEP), in mobilen Manipulatoren anzuwenden.
In dieser Dissertation verwenden wir das CEP-Framework in mobilen Manipulatoren mit einer vorherigen Karte, die von Rapid Random Search Tree generiert wurde, und vergleichen verschiedene Methoden, um ein lokales Ziel zu finden. Wir haben festgestellt, dass unsere Methode die 2D-Planung und die Ganzkörperkontrolle gut durchführen kann.

Contents

Figures and Tables

List of Figures

List of Tables

1. Introduction

A mobile manipulator is a robot system which contains a robot arm and a mounted mobile base. The robot arm is responsible for the manipulation work and the mobile base performs the navigation mission. Most of the mobile manipulators possess a 6 DOFs (Degree Of Freedoms) robot arm and a specific gripper for different purposes. Some mobile manipulators differ from each other in the drive module of the mobile base, which is differential drive scheme with two motivated wheels or three wheels with nonholonomic constrains or even universal wheels. In our work we use TIAGo+ (Take It And Go plus one arm) as the platform, which has a 7 DOFs robot arm and a mobile base with differential drive.

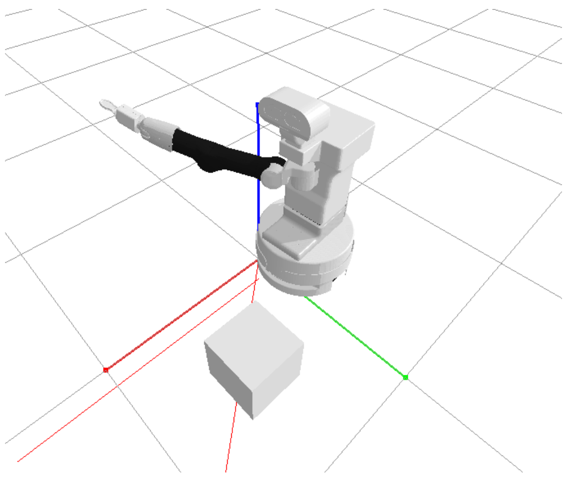

Figure 1.1.: TIAGo+ mobile manipulator and a simple environment in PyBullet

Thanks to the movable base the mobile manipulators could achieve more complicated tasks in unstructured and dynamic environments. Due to the limitation of the velocity, mobile manipulators are suitable for indoor assignments. Therefore, in order to accomplish an

autonomous mobile manipulator, it includes generality across missions and unknown scenarios, operations in high dimensional action spaces, eliminating uncertainty in sensing and actuation, and also integration of localization, perception, prediction, planning and control.
At the moment mobile manipulation is a subject of major focus on development and research environments, and mobile manipulators, either autonomous or teleoperated, are used in many areas, e.g. space exploration, military operations, environment protect, home-care and health-care. However, within the industrial field the implementation of mobile manipulators has been limited, although the needs for intelligent and flexible automation are present. In addition, the necessary technology entities (mobile platforms, robot manipulators, vision and tooling) are, to a large extent, available off-the-shelf components [1].

1.1. Related Work

Motion generation methods can be divided into path planning algorithms and Reactive Motion Generation (RMG) approaches [2]. Distinguished from global path planning methods which provide complete collision-free and feasible paths under known or partially known environment, reactive motion generation methods produce more responsive actions under a dynamic complicated environment based on real-time localization and perception information [2].
As one popular approach of the reactive motion generation methods, artificial potential fields methods (APF) [3] require a low computational cost, however, they compute the action command through combining different deterministic policies, which are usually responsible for target and obstacles. Although APF can provide a efficient solution, they ignore the potential relevance between the different policies which would cause conflicting behaviors when different policies contribution are merely summed. [2] models an attractor based on potential fields and intuitive physical interpretation. [4] emphasizes the importance of the real-time perception ability for reactive motion generation. [5] developed a hybrid strategy based on Circular and Potential Fields to perform collision avoidance and cooperation work with human.
In our work we frame reactive motion generation as an optimization problem over a product of expert policies, which not only computes an appropriate action through optimization, but also takes the dependence between different objectives.

1.1.1. Motion Planning

As to a robotic system, a fundamental problem is how to plan a path and execute the path in order to accomplish a specific task. Planning approaches take the current configuration and desired configuration as input, and return a plan which denotes how to arrive the goal. Informally speaking, it can also be distinguished into path planning and motion planning problems. Path planning algorithms compute a set of waypoints as a path from the start point to the goal point considering object avoidance and sometimes also geometric constraints. Motion planning generates a sequence of vector-based valid motions based on a given path, meanwhile respecting the movement constraints of the dynamic model, and follow the path as much as possible.

One of the off-the-shelf approaches dealing with motion planning is trajectory optimization of the given path generated by path planning algorithms. [6] CHOMP is an efficient method for continuous path refinement that utilizes covariant gradient techniques to promote quality of sampled trajectories, which can be produced by sampling-based algorithms like Probabilistic Roadmap (PRM). Another famous framework is STOMP [7], which generates trajectories with noises to explore the task space around an initial (possibly infeasible) trajectory and then updates the trajectories based on user-defined cost functions which takes smoothness and object avoidance into account. Another practical methodology is reactive motion generation (RMG). Reactive motion generation can handle potential dynamic objects better and perform responsive behaviors since the robot can receive and process the immediate sensor information to obtain a better understanding for the environment.

Given that, in this thesis we adopt this concept and generate reactive motions for the mobile manipulator.

The motion control and planning of the mobile manipulator which is equipped with the robot arm and a mobile platform is a crucial research area, because of the redundant freedoms of mobile manipulators can help them to accomplish more complicated tasks under different static or dynamic environments based on mounted sensor information. Since many robotic missions deal with computing a control action which fulfills multiple objectives, as a more complicated robotic system, the mobile manipulator should be capable to achieve a complex task satisfying different goals in parallel, which implies the manipulation task and navigation task.

1.1.2. Current Methods and Limitations

To solve this problem for mobile manipulators, different solutions are proposed. [8] shows that, compared with the mobile platform and the manipulator, the modifications of the dynamic model and the control algorithms for the mobile manipulators are not demanding. In [9] one of the traditional approaches is computing the control command for each joint through dynamic equations of motion based on the concept of operation space control, and the end-effector can be guided by human robot interactions. [10] combines an elastic roadmap framework, which translates global work space connectivity information into a series of potential functions and then performs globally task-consistent motion for manipulation tasks, with an end-effector-centric control framework which is built on operation space framework for robot control. [11] leverages force-based operation space control to accomplish a tricky task, opening a door, by estimating the parameters of the constraint that the robot is following (i.e. the door radius and position).

Hierarchical Planning Architecture
Informally speaking, due to the different properties of manipulation and navigation, hierarchical structure is suitable for motion planning of mobile manipulator. Hierarchical planning architecture separates manipulation and navigation task, and leverages two-level hierarchy combining a global planner and a local planner, where the global planner generates an approximate solution and the local planner deals with the dynamic obstacles. [12] implements a hybrid navigation method which incorporate the Distance Transform Path Planner as the global planner and the Potential Field navigation method as a local planner. [13] employ a two-level hierarchy planner on two dimensional navigation task and three dimensional manipulation task, where for 2D planner D* Lite [14] and polynomial curve functions are utilized, for 3D planner model predictive and Proportional-Integral-Differential (PID) control are responsible for the local planning and D* Lite performs the global planning.
However, it's still challenging to find a optimal ultimate arm and hand pose and a mobile platform position because of the two tasks are separated, which results in a loss in rapidity and efficiency. Instead, the approach we proposed can carry out the two tasks in parallel and meet different objectives in real-time.

Learning-based approaches
Thanks to the powerful ability of deep learning to extract information from image and video data, earning-based approaches can process environment information quickly and

generate appropriate control commands. Using learning-based approaches, the motion planning for mobile manipulators can be viewed as a whole body control problem, which can reduce execution time cost and broaden work space size. In [15] Convolutional Neural Network (CNN) architecture is employed to map raw sensor information to steering commands based on an expert operator. In [16] the arm's feedback is used to learn a low-level controller using deep reinforcement learning that drives the base to such a place that the arm is able to plan a trajectory up to the object, in which Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimisation (PPO) are compared. [17] proposes an End-to-End deep reinforcement learning architecture which takes 2D LiDAR (Light Detection And Range) data and target position in end-effector task space and output the accelerations of the arm and mobile base.
However, learning-based approaches are heavily dependent to the training data. For deep reinforcement learning architecture it's hard to guarantee the convergence and repeatability consistency; as to End-to-End deep learning architecture it's tricky to find the cause of potential problems. In contrast, although sensor data is not incorporate into our framework, our approach can find relatively robust and reactive solutions given a global prior map information, and we can define the specific objectives explicitly.

Model Predictive Control
Model Predictive Control (MPC) constructs a online optimization problem minimizing the cost taking the reference and geometric constraints into account over a finite predicted horizon and then computes the optimal motion. Most MPC controllers relies on a proper kinematic model. [18] utilizes a MPC controller to generate a safe motion in a partially unknown and dynamic environment, which also deals with object avoidance and joint constraints.
However, the weight matrices of the objective functions are required to be fine tuned or chosen carefully due to its unstable performance. Although the prediction nature brings some heuristic information which attracts the robot to the desired trajectory or set-points, this mechanism may cause uncertain behaviors. And hence, MPC will be compared with CEP in experiment section.

1.2. Overview

Our work is strongly based on the work of [19], which proposed a novel framework for modular reactive motion generation. Composable Energy Policies (CEP) compute

an optimal action through optimizing the product of different expert policies which are specifically designed for different objectives. Moreover, CEP framework realizes to perform a complex task which requires satisfying multiple objectives in parallel. As to mobile manipulators, the extended DOFs of mobile base bring about more possibility for various tasks, and, hence, it's appropriate to adapt the whole body control of mobile manipulators to composable energy policies formulation. A relatively simple task like opening a door, requires the robot to properly approximate the door, grasp the handle and avoid to collide against the door while additionally, avoiding to reach joint limits or exceed the joint torques. During this thesis, we develop algorithms to satisfy multi-objective reactive motion planning in mobile manipulators. We aim to achieve multi-objective planning by composable energy policies, a novel algorithm that frames multi-objective reactive motion planning as an inference problem.
Our work can be summarized as:

- Use operation space control to provide guidance for manipulation.
- Utilize a prior map generated by RRT and PD control in navigation task.
- Based on CEP framework perform 2D planning and the whole body control of TIAGo+.

Chapter 1 begins with the motivation of this thesis, current approaches and their limitations; Chapter 2 introduces necessary basic knowledge of model free policy search, composable energy policies (CEP), operation space control (OSC), path planning algorithms, potential fields methods and model predictive control (MPC). Chapter 3 describes our approaches in details. Then chapter 4 demonstrates the experiment results. Last but not least, in chapter 5 and 6 we discuss about our contribution, and the limitations and outlook or our approach.

2. Foundations

2.1. Model-Free Policy Search

Policy search is a subarea in reinforcement learning which concentrates on searching for good parameters for a given policy parametrization. In robot learning research, there are model-free policy search and model-based policy search. Model-based policy search solve this problem by learning a model or simulator of the robot dynamics from the simulation data. In comparison, model-free policy search learn a optimal policy through sampled trajectories without a forward dynamic model. Hence, model-free policy search is used more widely in practice.
Model-free policy search improves the policy directly based on sampled trajectories $\boldsymbol{\tau}^{[i]}$, and the obtained immediate rewards $r_0^{[i]}$, $r_1^{[i]}$,..., $r_T^{[i]}$ for the trajectories. Model-free policy search methods try to update the parameters $\boldsymbol{\theta}$ such that trajectories with higher rewards have higher probability if following the new policy, and, hence, the average return

$$J_{\boldsymbol{\theta}} = E[R(\tau)|\boldsymbol{\theta}] = \int R(\boldsymbol{\theta})\pi_w(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{2.1}$$

increases.
More specifically, model-free policy search is distinguished from three aspects: exploration strategies, policy evaluation strategies and policy update strategies.

Exploration Strategies

The exploration strategies decide the exploration space and therefore influence the algorithm efficiency. Hence, they are crucial to the performance of the policy search algorithms. Considering the exploration space, policy exploration strategies are distinguished into exploration in action space and in parameter space. Taking the robot control into account, in this thesis the trajectories $\boldsymbol{\tau}^{[i]}$ are sampled from action space, in which each sample $\boldsymbol{\tau}^{[i]}$ represents the joint actions $\mathbf{q}_j$ of the robot.

Exploration in the action space is performed by adding an exploration noise $\boldsymbol{\epsilon}_{\boldsymbol{\mu}}$ directly to the executed actions:

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}(\boldsymbol{x}, t) + \boldsymbol{\epsilon}_{\boldsymbol{\mu}} \tag{2.2}$$

The noise is mostly modeled through a Gaussian distribution with a zero mean. Thus, the exploration space is given as:

$$\boldsymbol{\pi}_\theta(\boldsymbol{\mu}|\boldsymbol{x}) = N(\boldsymbol{\mu}|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_\mu)$$

with $\boldsymbol{x}$ is the state. And hence, the parameter vector which will be evaluated in the exploration step is:

$$\boldsymbol{\theta}^{[i]} \sim \boldsymbol{\pi}_\theta(\boldsymbol{\mu}|\boldsymbol{x}), i = 1...N \tag{2.3}$$

Policy Evaluation Strategies

Once the actions are sampled from the exploration space, policy evaluation strategies are utilized to assess the quality of the samples, which are classified into step-based evaluation and episode-based evaluation. As to step-based evaluation, at each time step the state-action pair $(\boldsymbol{x}_t^{[i]}, \boldsymbol{u}_t^{[i]})$ will be evaluated and then return a evaluation result $\boldsymbol{r}_t$, then for a whole trajectory $\boldsymbol{\tau}^{[i]}$, it contains state-action pairs $(\boldsymbol{x}_t^{[i]}, \boldsymbol{u}_t^{[i]})$ and their rewards or costs; in contrast, for episode-based evaluation, the policy parameter $\boldsymbol{\theta}^{[i]}$ which is used along the whole episode will be assessed and a correspondent reward $R^{[i]}$ will be computed. Due to the higher variance of the returned reward in step-based evaluation strategy, episode-based evaluation strategy is chosen in this thesis.
Thus, for each trajectory $\boldsymbol{\tau}^{[i]}$, the policy parameters $\boldsymbol{\theta}^{[i]}$ are sampled and corresponding rewards $R^{[i]} = \sum_{t=0}^{T} r_t^{[i]}$ are calculated, and hence, the whole composed data are:

$$D_{eps} = \{\theta^{[i]}, R^{[i]}\}_{i=1...N}$$

Policy Update Strategies

Based on the observed data, the initial policy can be improved in a direction of higher returned rewards. In policy search field, policy update strategies are divided into policy gradient methods, expectation-maximization based methods, information theoretic methods, and policy updates which can by derived from the path-integral theory.
Policy gradient (PG) methods leverages gradient ascent in order to maximize the expected return $\boldsymbol{J}_{\boldsymbol{\theta}}$. The policy parameter is updated by the gradient $\nabla_{\boldsymbol{\theta}} \boldsymbol{J}_{\boldsymbol{\theta}}$ pointing a direction in which the expected return increases steepest:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}} \boldsymbol{J}_{\boldsymbol{\theta}} \tag{2.4}$$

with α is the learning rate and the policy gradient is given by:

$$\nabla_{\boldsymbol{\theta}} \boldsymbol{J}_{\boldsymbol{\theta}} = \int_{\boldsymbol{\tau}} \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) R(\boldsymbol{\tau}) d\boldsymbol{\tau} \tag{2.5}$$

where trajectories $\boldsymbol{\tau}^{[i]}$ are sampled from $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$. In general, policy gradient methods differ from the estimation of $\nabla_{\boldsymbol{\theta}} \boldsymbol{J}_{\boldsymbol{\theta}}$.

Expectation maximization (EM) policy search approaches don't require a learning rate parameter which needs to be fine tuned to ensure a convergent result.Instead, on account of the observed data and initial model parameters, new latent data can be inferred, and then model parameters are updated in order to fulfill the new latent data and previous observation. As to standard Expectation-Maximization algorithm, the policy update is carried out through the weighted maximum logarithmic likelihood estimation which has a closed form solution of the most of the the used policies.
Assuming $\boldsymbol{x} = (\boldsymbol{x}^{[1]}, \boldsymbol{x}^{1}, ..., \boldsymbol{x}^{[m]}$ are observed data, $\boldsymbol{z} = (\boldsymbol{z}^{[1]}, \boldsymbol{z}^{1}, ..., \boldsymbol{z}^{[m]}$ are unobserved latent data. For purpose of finding the policy parameter of the samples, the logarithmic likelihood function of maximization of joint distribution of observed and unobserved data:

$$\boldsymbol{\theta} = \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log \boldsymbol{P}(\boldsymbol{x}^{[i]}; \boldsymbol{\theta}) = \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log \sum_{\boldsymbol{z}^{[i]}} \boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta}) \tag{2.6}$$

Using Jensen inequality,

$$\log \sum_j \lambda_j y_j \geq \sum_j \lambda_j \log y_j, \lambda_j \geq 0, \sum_j \lambda_j = 1.$$

the above function can be transformed to

$$\sum_{i=1}^{m} \log \sum_{\boldsymbol{z}^{[i]}} \boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta}) = \sum_{i=1}^{m} \log \sum_{\boldsymbol{z}^{[i]}} \boldsymbol{Q}_i(\boldsymbol{z}^{[i]}) \frac{\boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta})}{\boldsymbol{Q}_i(\boldsymbol{z}^{[i]})} \geq \sum_{i=1}^{m} \sum_{\boldsymbol{z}^{[i]}} \boldsymbol{Q}_i(\boldsymbol{z}^{[i]}) \log \frac{\boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta})}{\boldsymbol{Q}_i(\boldsymbol{z}^{[i]})}$$

where $\boldsymbol{Q}_i(\boldsymbol{z}^{[i]})$ is a distribution which can be computed when the equal sign of Jensen inequality holds:

$$\boldsymbol{Q}_i(\boldsymbol{z}^{[i]}) = \frac{\boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta})}{\sum_{\boldsymbol{z}} \boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta})} = \frac{\boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta})}{\boldsymbol{P}(\boldsymbol{x}^{[i]}; \boldsymbol{\theta})} = \boldsymbol{P}(\boldsymbol{z}^{[i]} | \boldsymbol{x}^{[i]}; \boldsymbol{\theta})$$

Therefor, in order to compute the model parameters, the logarithmic likelihood needs to be maximized, and hence the lower bound is required to be maximized firstly:

$$\arg \max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \sum_{\boldsymbol{z}^{[i]}} \boldsymbol{Q}_i(\boldsymbol{z}^{[i]}) \log \frac{\boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta})}{\boldsymbol{Q}_i(\boldsymbol{z}^{[i]})}$$

with $\sum\limits_{z} \boldsymbol{\theta}_i(\boldsymbol{z}_i) = 1$.
Thus, the lower bound of logarithmic likelihood which needs to be maximized is:

$$\arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \sum_{\boldsymbol{z}^{[i]}} \boldsymbol{Q}_i(\boldsymbol{z}^{[i]}) \log \boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta})$$

This is the maximization step of EM, and the $\boldsymbol{Q}_i(\boldsymbol{z}^{[i]}) \log \boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta})$ represents the expectation step of $\log \boldsymbol{P}(\boldsymbol{x}^{[i]}, \boldsymbol{z}^{[i]}; \boldsymbol{\theta})$ based on conditional probability distribution $\boldsymbol{Q}_i(\boldsymbol{z}^{[i]})$. Based on Expectation-Maximization algorithms, the policy search problem can be formulated as a latent variable inference problem.
In this thesis Reward Weighted Regression (RWR) is used to update the policy which is easy to use. The policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{s})$ is defined as a Gaussian linear model $N(\boldsymbol{\theta}|\boldsymbol{W}^T\boldsymbol{\phi}(\boldsymbol{s}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$. Given the data-set D_{eps} from policy evaluation process and weight parameters $d^{[i]}$ for each sample $(\boldsymbol{s}^{[i], \boldsymbol{\theta}^{[i]}})$, the weighted maximum likelihood estimation is given by:

$$\boldsymbol{W} = (\boldsymbol{\phi}^T \boldsymbol{D} \boldsymbol{\phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\phi}^T \boldsymbol{D} \boldsymbol{\Theta} \tag{2.7}$$

where λ is the ridge factor, $\boldsymbol{\phi}$ is composed pf the feature vectors of the contexts, $\boldsymbol{D}$ is the diagonal matrix of weight parameters, $\boldsymbol{\Theta}$ is the matrix of parameters vectors $\boldsymbol{\theta}^{[i]}$.

The core concept of Information theoretic (IT) methods is trying to stay close to the old 'data', and not to jump away from the previous policy because it's dangerous in robot control. Entropy is a basic definition in information theorem which implies the uncertainty degree of the information. Information theoretic methods improves the policy and in the meanwhile, constrains the distance between the previous policy and the newly updated policy. Such approaches restricts the information loss during the update process, and, hence, avoids that the new policy prematurely concentrates on local optima of the reward landscape, which has been used in natural policy gradient algorithms. However, policy gradient approaches are required to define a learning rate. And hence, to combine the good properties of policy gradient methods and EM-based methods, the information theoretic methods were taken up with the Relative Entropy Policy Search (REPS) algorithm to combine the advantages of both types of algorithms. REPS uses the same information theoretic bound as the NAC algorithm but simultaneously updates its policy by weighted maximum likelihood estimates[20].

There also are popular policy search methods in robot learning area using path-integral theory like Covariance Matrix Adaptation - Evolutionary Strategy (CMA-ES) [21], although will not be discussed in detail in this thesis.

2.2. Composable Energy Policies

Reactive motion generation problems are usually solved through computing actions as a summation of policies. However, sometimes these policies are independent from each other, therefore it's not rigorous when there are contradictory behaviours between them. Avoiding this intractable problem, Composable Energy Policies (CEP), a novel framework for modular reactive motion generation CEP [19], it calculates the actions by the optimization over product of different policies instead of summation. Hence, in CEP framework reactive motion generation problems are transformed to an optimization problem over a product of expert policies:

$$a^{*} = \arg\max_{\mathbf{a}} \prod \pi_k(\mathbf{a}|\mathbf{s}) \tag{2.8}$$

To solve this problem, firstly assume a set of stochastic policies $\pi_1(a|s)$, ... , $\pi_k(a|s)$ modeled by Boltzmann distribution:

$$\pi_i(\mathbf{a}|\mathbf{s}) = \exp(E_i(\mathbf{a},\mathbf{s}))\frac{1}{Z_i(\mathbf{s})} \tag{2.9}$$

with $E_i(\mathbf{a},\mathbf{s}) : \mathbf{S}\times\mathbf{A} \Rightarrow R$ is a arbitrarily represented energy function, $Z_i(\mathbf{s}) = \int_{\mathbf{a}} \exp(E(\mathbf{a};\mathbf{s}))d\mathbf{a}$ is a normalization parameter. Additionally, for different importance policies are assigned with different weight. Therefore, the computation of the product of different policies can be represented by:

$$\pi(\mathbf{a}|\mathbf{s}) = \prod_{k:0\rightarrow K} \pi_k(\mathbf{a}|\mathbf{s})^{\beta_k} \propto \exp(\sum_k \beta_k E_k(\mathbf{a};\mathbf{s})) \tag{2.10}$$

Normally every energy function is defined in the same state-action space, however, for most robot arm control and mobile manipulators tasks, it's worth considering various objectives in different space in order to achieve a more complicated work. For example, as to 7-DOFs robot arm, object avoidance should be guaranteed in each joint and the end-effector should reach a certain target pose; moreover, considering mobile manipulators, it's required that mobile base should perform object avoidance and arrive at a certain region so that the robot could carry out manipulation work.

Hence, a set of policies $\pi^z(\mathbf{a}^z|\mathbf{s}^z)$ in different state-action latent spaces $(\mathbf{s^z}, \mathbf{a^z}) \in \mathbf{Z}$ and corresponding deterministic mappings $\mathbf{f}_x^z : X \Rightarrow Z$ that transform the state-actions from a certain task space into another space.

Therefore rewriting the policy function modeled by Boltzmann distribution:

$$\pi^x(\mathbf{a}^x|\mathbf{s}^x) = \exp(E^z(\mathbf{a}^z,\mathbf{s}^z))\frac{1}{Z_i(\mathbf{s}^x)} = \exp(E^z(\mathbf{f}_x^z(\mathbf{a}^x,\mathbf{s}^x))\frac{1}{Z_i(\mathbf{s}^x)} \tag{2.11}$$

with

$$Z = \frac{\int_{a^z} \exp(E^z(\mathbf{a}^z, \mathbf{s}^z)) d\mathbf{a}^z}{\sqrt{det(\mathbf{J}^{\mathbf{T}}\mathbf{J})}} \tag{2.12}$$

with $\mathbf{J} = \partial \mathbf{f}_z^z(\mathbf{a}^x, \mathbf{s}^x)/\partial \mathbf{a}^x$ is the Jacobian of $\mathbf{f}_z^z(\mathbf{a}^x, \mathbf{s}^x)$ in the action $\mathbf{a}^x$.
Also, CEP assumes an implicit function instead of an explicit function:

$$\mathbf{a}^x = \text{arg} \max_{\mathbf{a}^x} E^z(\mathbf{f}_x^z(\mathbf{a}^x, \mathbf{s}^x)) \tag{2.13}$$

To sum up, assuming a set a policies defined in different task spaces Z_1, ..., Z_K, the final policy could be composed together:

$$\pi^x(\mathbf{a}^x|\mathbf{s}^x) = \exp(\sum_{k=1}^{K} \beta_k E^{z_k}(\mathbf{f}_x^{z_k}(\mathbf{a}^x, \mathbf{s}^x))) \frac{1}{Z_i(\mathbf{s}^x)} \tag{2.14}$$

Then, the optimal actions are computed through Maximum Likelihood Estimation over the product of expert policies. Given the current state s^x,

$$a_{ML}^x = \text{arg} \max_{\mathbf{a}} \prod_{k=0}^{K} \pi_k(\mathbf{a}^x|\mathbf{s}^x)^{\beta_k}, \beta_k > 0. \tag{2.15}$$

This problem can be solved by Cross Entropy optimization. A sampling model $p(\mathbf{a}|\theta)$ is defined beforehand and the parameters of the model can be iteratively calculated:

$$\theta_{t+1} = \text{arg} \max_{\theta_t} E_{p(\mathbf{a}^x|\theta_t)}[\log(\prod_{k=0}^{k})\pi_k(\mathbf{a}^k|\mathbf{s}^x)_k^{\beta}] \propto E_{p(\mathbf{a}^x|\theta_t)}[\beta_k E^{z_k}(\mathbf{f}_x^{z_k}(\mathbf{a}^x, \mathbf{s}^x))]. \tag{2.16}$$

Each energy component is linearly independent, thus, based on different objectives various energy categories can be constructed.

CEP for Reactive Motion Generation
In order to adapt Composable Energy Policy framework in robot motion generation, state-action spaces $(\mathbf{s}^x, \mathbf{a}^x)$ of robot control and potential latent spaces $(Z_1$, ..., $Z_K)$ need to be determined. Therefore, corresponding mappings between common spaces and latent spaces can be decided by robot kinematics.
Considering the robot motion is generated through a second-order dynamic system

$$\ddot{\mathbf{q}} = g(\mathbf{q}, \dot{\mathbf{q}}) \tag{2.17}$$

with $\mathbf{q}$ is the joint state, $\dot{\mathbf{q}}$ is the joint velocity and $\ddot{\mathbf{q}}$ is the joint acceleration. In common state-action space, in other word, the joint space, the state and action are:

$$(\mathbf{q}, \dot{\mathbf{q}}), \ddot{\mathbf{q}}. \tag{2.18}$$

Meanwhile, in the configuration space of end-effector, in which state-action pair is represented by:

$$(\mathbf{x}, \dot{\mathbf{x}}), \ddot{\mathbf{x}} \tag{2.19}$$

respectively. Hence, on account of robot kinematics, the transformation mappings can be modeled by:

$$\mathbf{x} = \mathbf{f}_{kin}(\mathbf{q}) \tag{2.20}$$

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2.21}$$

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} \approx \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} \tag{2.22}$$

where $\mathbf{f}_{kin}$ is the forward kinematics function which transforms joint state into configuration state, $\mathbf{J} = \partial\mathbf{x}/\partial\mathbf{q}$ is the Jacobian matrix given forward kinematic relation.

2.3. Operation Space Control

Operational space control (or task space control) centers on how to resolve redundancy and how to produce appropriate motor commands in configuration space such that both end-effector and DOFs (Degree Of Freedoms) achieve a prescribed level of impedance [22]. In [22] there are eight operational space controllers in three different categories: velocity-based control, acceleration-based control and force-based control. In general, the motor commands can only be enforced in joint space, while in most cases the target behaviors of the robot arm are employed in task space to perform a complicated task. Therefore, by defining the position and orientation of the end-effector, operation space control can derive the corresponding instructions in joint space. And hence, more specifically, given desired pose $\boldsymbol{x}_{des}$ and velocity $\dot{\boldsymbol{x}}_{des}$ in task space of the end-effector, the force commands $\boldsymbol{u}$ in joint space should be computed.
Jacobian matrix $\boldsymbol{J}$ represents the transformation between different spaces, for instance, between different joints. Assuming the force in end-effector $\boldsymbol{F}_x$ and the corresponding Jacobian $\boldsymbol{J}_{ee}$ are known, and, hence,

$$\boldsymbol{u} = \boldsymbol{J}_{ee}^T(\boldsymbol{q})\boldsymbol{F}_x \tag{2.23}$$

According to Newton's second law,

$$\boldsymbol{F}_x = \boldsymbol{M}_{x_{ee}}(\boldsymbol{q})\ddot{\boldsymbol{x}}_{\boldsymbol{des}} \tag{2.24}$$

where $\boldsymbol{M}_{x_{ee}}$ is the inertia matrix in operational space and $\ddot{\boldsymbol{x}}_{\boldsymbol{des}}$ is the desired acceleration in task space. Thus, the control signal gives

$$\boldsymbol{u} = \boldsymbol{J}_{ee}^T(\boldsymbol{q})\boldsymbol{M}_{x_{ee}}(\boldsymbol{q})\ddot{\boldsymbol{x}}_{\boldsymbol{des}} \tag{2.25}$$

Through incorporating the inertia into the control signal, the inertia can be ignored, but in order to cancel out the inertia of the system in operational space, more transformation need to be performed.
Firstly, taking end-effector pose $\boldsymbol{x}$ in task space and joint value $\boldsymbol{q}$ into account, Jacobian matrix establishes their transformation

$$\dot{\boldsymbol{x}} = \boldsymbol{J}_{ee}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.26}$$

Taking the time derivative,

$$\ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}_{ee}(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{J}_{ee}(\boldsymbol{q})\ddot{\boldsymbol{q}} \tag{2.27}$$

For a robot dynamics system,

$$\boldsymbol{u} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) \tag{2.28}$$

where $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is a function representing the Coriolis and centrifugal effect, $\boldsymbol{g}(\boldsymbol{q})$ defines the effect of gravity in joint space. By substituting $\ddot{\boldsymbol{q}}$ and ignoring gravity, therefore, gives

$$\ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}_{ee}(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{J}_{ee}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})) \tag{2.29}$$

Through substituting $\boldsymbol{u}$ with the control signal it gives,

$$\ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}_{ee}(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{J}_{ee}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{J}_{ee}^T(\boldsymbol{q})\boldsymbol{M}_{x_{ee}}(\boldsymbol{q})\ddot{\boldsymbol{x}}_{\boldsymbol{des}} - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})) \tag{2.30}$$

After rearranging and simplification of modeling,

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}_{ee}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{J}_{ee}^T(\boldsymbol{q})\boldsymbol{M}_{x_{ee}}(\boldsymbol{q})\ddot{\boldsymbol{x}}_{des} \tag{2.31}$$

In order to set the dynamics $\ddot{\boldsymbol{x}}$ to be equal to the desired value $\ddot{\boldsymbol{x}}_{des}$, it's required to choose $\boldsymbol{M}_{x_{ee}}(\boldsymbol{q})$ carefully:

$$\boldsymbol{M}_{x_{ee}}(\boldsymbol{q}) = [\boldsymbol{J}_{ee}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{J}_{ee}^T(\boldsymbol{q})]^{-1} \tag{2.32}$$

Therefore, it's the inertial matrix which is required in control signal. In reality, to input motor commands, $\ddot{\boldsymbol{x}}_{des}$ can be decided by Proportional-Differential (PD) control, and, hence,

$$\ddot{\boldsymbol{x}}_{des} = k_p(\boldsymbol{x} - \boldsymbol{x}_{des}) + k_v(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{des}) \tag{2.33}$$

In conclusion, the control signal is given by

$$\boldsymbol{u} = \boldsymbol{J}_{ee}^T(\boldsymbol{q})\boldsymbol{M}_{x_{ee}}(\boldsymbol{q})k_p(\boldsymbol{x} - \boldsymbol{x}_{des}) + k_v(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{des}) \tag{2.34}$$

By setting the desired pose $\boldsymbol{x}$ and velocity $\dot{\boldsymbol{x}}$ the control commands can be computed.

2.4. Path Planning

Path planning is a very crucial task for mobile manipulators. Based on the localization system and user-specified target position, path planning algorithms aim at finding a appropriate path which fulfills several special objectives from the start point to the goal point. Generally, popular path planning algorithms are divided into search-based methods, sample-based methods.

Search-based Methods
One of the representative search-based algorithms is Dijkstra approach which can compute the shortest path because it only considers the adjacent vertices. The Greedy Best-First-Search performs actions according to the estimation of the distance from the goal point, which also called heuristic information. It brings about a faster search than Dijkstra [23], however, it can not guarantee a shortest path. Taking the optimality and efficiency into account, A* [24] takes advantage of these two concepts, using a heuristic function which contains the computation of already-explored vertices and the estimation of the distance from the target to guide the planning process, and different variants modify the approach of the estimation of distance so as to improve the efficiency and optimality.

Sampling-based Methods
Arguably, the most influential sampling-based path planning algorithms include Probabilistic RoadMaps (PRMs) and Rapidly-exploring Random Trees (RRTs) [25]. PRMs approach constructs a graph connected by feasible and collision-free straight lines through sampling in state space. It has been proved that PRMs not only performs well in high-dimensional state spaces [26], but also posses probability completeness and the rate of failing decays to

zero exponentially with the number of sampling increases [27]. However, PRMs requires to perceive a prior environment which may cost too much computationally. Compared to PRMs, incremental sampling-based algorithm RRTs presents its online planning ability and its rapidity. The RRT algorithm also has been proved to be probabilistically complete [28], with an exponential rate of decay for the probability of failure [29]. As RRTs could explore the partially know or unknown environment as much as possible when a low goal sample rate is set and appropriate control input is defined, and hence, in this this RRTs is utilized to provide a prior map in order to adapt to CEP framework.
Based on a partially known or unknown environment, setting the start vertex $\boldsymbol{x}_{start}$, maximum iteration number K and time step Δt, RRTs can return a graph which contains a feasible path connecting start vertex and goal vertex. Referring to 1 to get a more clear view.

```
GenerateRRT($\boldsymbol{x}_{start}$, $\boldsymbol{K}$, $\Delta t$)
Initialize $\boldsymbol{x}_{start}$, $\boldsymbol{x}_{end}$, $\boldsymbol{u}$, $K$, $\Delta t$;
$\boldsymbol{\tau}$.init($x_{init}$);
for iteration=1,2,..., $K$ do
    $\boldsymbol{x}_{rand} \Leftarrow RandomState()$;
    $\boldsymbol{x}_{near} \Leftarrow NearestNeighbor(\boldsymbol{\tau}, \boldsymbol{x}_{rand})$;
    $\boldsymbol{u} \Leftarrow SelectInput(\boldsymbol{x}_{near}, \boldsymbol{x}_{rand})$;
    $\boldsymbol{x}_{new} = NewState(\boldsymbol{x}_{near}, \boldsymbol{u}, \Delta)$;
    $\boldsymbol{\tau}.addVertex(\boldsymbol{x}_{new})$;
    $\boldsymbol{\tau}.addEdge(\boldsymbol{x}_{near}, \boldsymbol{x}_{new}, \boldsymbol{u})$;
end
Return $\boldsymbol{\tau}$;
```

Algorithm 1: RRT

2.5. Potential Fields Methods

Motion planning is carried out as a iterative problem. In comparison, potential fields methods solve the path planning problem by constructing a global artificial potential energy field which guides the robot to move from the high energy region to the lower region. Potential fields methods are suitable for overcoming unknown dynamic scenario and being applied to real-time tasks because of its elegance[30].
By means of the concept of electronic potential field, thinking of the robot and the obstacles as positively charged particles, and modeling the goal point as a negatively charged particle,

according to the potential functions, the whole planning space can be represented as a artificial potential fields. In general, the potential field is established by

$$\boldsymbol{U}(q) = \boldsymbol{U}_{att}(q) + \boldsymbol{U}_{rep}(q) \tag{2.35}$$

where $\boldsymbol{U}_{att}(q)$ determines the attractive potential which drives the robot to move to the goal and $\boldsymbol{U}_{rep}(q)$ defines the repulsive potential which pushes away the robot from the obstacles. As for attractive potential $\boldsymbol{U}_{att}(q)$, its definition is given as:

$$\boldsymbol{U}_{att}(q) = \begin{cases} \frac{1}{2}\xi d^2(q, q_{goal}), d(q, q_{goal}) \geq d^*_{goal} \\ d^*_{goal}\xi d(q, q_{goal}) - \frac{1}{2}\xi(d^*_{goal})^2, d(q, q_{goal}) < d^*_{goal} \end{cases}$$

Where d^*_{goal} is predefined threshold for defining the boundary for different potentials, $d(q, q_{goal})$ computes the distance between the current configuration q_{goal} and goal configuration q_{goal}, and ξ is a scale factor which needs to be fine tuned. When the current distance $d(q, q_{goal})$ is bigger than d^*_{goal}, the attractive potential is high enough to provide a attractive force; Otherwise, when the robot is getting closer to the goal point, the attractive potential is lower and comes to zero at the goal position. And hence, its derivative is given as:

$$\Delta\boldsymbol{U}_{att}(q) = \begin{cases} \xi d(q, q_{goal}), d(q, q_{goal}) \geq d^*_{goal} \\ \frac{d^*_{goal}\xi(q, q_{goal})}{d(q, q_{goal})}, d(q, q_{goal}) < d^*_{goal} \end{cases}$$

In contrast, repulsive potential function implements constraints to drive the robot away,

$$\boldsymbol{U}_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^*})^2, D(q) \leq Q^* \\ 0, D(q) > Q^* \end{cases}$$

where $D(q)$ stands for the distance between the current robot position and the obstacles, Q^* is the threshold value, η is a scale factor. Therefor, its derivative is given by

$$\boldsymbol{U}_{rep}(q) = \begin{cases} \eta(\frac{1}{D(q)} - \frac{1}{Q^*})\frac{1}{D^2(q)}\nabla D(q), D(q) \leq Q^* \\ 0, D(q) > Q^* \end{cases}$$

And hence, the resultant force which is applied to the robot is the negative gradient of the combination of attractive potential and repulsive potential:

$$\boldsymbol{F}(q) = -\nabla\boldsymbol{U}(q) \tag{2.36}$$

2.6. Model Predictive Control

Model Predictive control (MPC), also called dynamic matrix control or receding horizon control, is widely used in some industries, typically for systems with slow dynamics. Given a dynamics controlled system and current state of the robot, at each time step MPC computes a optimal input action by means of solving a planning problem which minimizes a user defined cost function and meanwhile satisfies some constraints over a sequence of predicted actions in finite horizon.
MPC framework is widely used in robot control and also different extensions are added to improve the performance and adapt to various tasks. In [31] an inverse dynamics feedback linearization and a data-driven error model are integrated into a model predictive model to enhance the trajectory tracking precision in robot manipulation. A tricky challenge in MPC formulation is the construct of a control model. For robot manipulation, it can be simply based on a dynamic model of the robot arm. [32] linearizes a nonlinear dynamic model of the robot by using a feedback linearization control and a MPC controller is leveraged to control a twp-link robot arm. Moreover, the dynamic model can also be made using neural network. IN [33], a dynamic model which is formed by multi-layer neural network is incorporated into MPC controller, which is extended to solve model-based reinforcement learning tasks. As to mobile robot, it's well know that a mobile robot with nonholonomic constraints cannot be feedback stabilized through continuously differentiable, time invariant control law [34]. Thus MPC controller could tackle these constraints. [34] and [35] employed a MPC controller in a mobile robot with nonholonomic contraints base and differential drive respectively, and the optimization problem in MPC is solve by quadratic programming(QP).
Informally speaking, composable energy policies framework focuses on the next action through optimizing the summed reward of next step based on different energy components, in which specific constraints and objectives are define. However, model predictive control finds the optimal action of the next step via optimizing the reward sum of a sequence of the predicted actions over a finite horizon and in the meanwhile the calculation needs to meet some constraints. And hence, in [5] CEP and MPC will be used to control the mobile base and the whole robot body and their performance will be compared.
As for model predictive control, let state and input trajectories are $x(\tau)$ and $u(\tau)$, at each

time step t, the optimization planning problem is:

$$\begin{aligned} &\textbf{minimize} && \sum_{\tau=t}^{t=T} l(x(\tau), u(\tau)), \\ &\textbf{wrt.} && u(t) \in \mathcal{U}, x(\tau) \in \mathcal{X}, \tau = t, ..., t+T; \\ &&& x(\tau+1) = A(x(\tau)) + Bu(\tau), \\ &&& \tau = t, ..., t+T-1, x(t+T) = 0. \end{aligned}$$

where $A \in \boldsymbol{R}^{n \times n}$ and $B \in \boldsymbol{R}^{n \times m}$ are dynamics and input matrices, $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{U}}$ are the set of the states and input actions, $x(\tau), x(\tau{+}1), ...x(\tau{+}T) \in \boldsymbol{R}^n$ and $u(\tau), u(\tau{+}1), ...u(\tau{+}T) \in \boldsymbol{R}^m$ represent corresponding states and actions. And hence, the final solution is only one input action $u^{\sim}(t)$, and then the input sequence of actions can be computed step by step for the whole task.

3. Our Approach

Basically, our approach is based on Composable Energy Policies (CEP) and model-free policy search (PS). On the one hand, CEP approach computes the optimal action by maximizing the product over the different expert policies which implies corresponding purposes. On the other hand, model-free policy search can find the approximately optimal expert policy through updating the previous policy based on the evaluation of the sampled trajectories.
Then the way to evaluate the samples should be customized for different goals, e.g. for manipulation task and navigation task. As the initial policy $\boldsymbol{\pi}$ is modeled by multivariate Gaussian distribution of 10 dimensional action, which represents the joint accelerations $\boldsymbol{a} \in R^{10}$, the sampled action in joint space can be directly evaluated. And hence, the sampled action in joint space $\boldsymbol{a}^x$ should be mapped into the task space where the mapped action $\boldsymbol{a}^z$ can be assessed depending on specific objectives. Therefore, the action in joint space $\boldsymbol{a}^z$ is mapped into end-effector task space for manipulation and also planar space for navigation, and hence, the mapped action in task spaces should be compared with desired action in the corresponding task space so that the reward $\boldsymbol{r}$ can be calculated. The desired action in different task spaces can be produced in various way. In our case, as to manipulation task in end-effector task space, the target action can be computed through Operation Space Control (OSC) and PD control; for navigation task in planar space, the target action is also derived from PD control, and the local target point is based on a prior map. Consequently, in our work, the logarithmic probability of the mapped action over a multivariate Gaussian distribution serves as the reward, and hence, each sample action in joint space has a scale reward. Thus, for each iteration in policy search a batch of action can be sampled and evaluated to speed up the iterative process. The following formulated this whole process in details.

As previous statement, at each time step the planning problem which is required to solve is

$$a^* = \arg\max_{\mathbf{a}} \prod \pi_k(\mathbf{a}|\mathbf{s}) \tag{3.1}$$

where $\pi_k(\mathbf{a}|\mathbf{s})$ represents different expert policies, and a^* denotes the final action which is extracted from the resulted policy. For instance, if the expert policy is modeled by Gaussian distribution, then CEP computes a resultant distribution that fulfills two existing expert policies in parallel. And hence, model-free policy search is employed to find the expert policies.

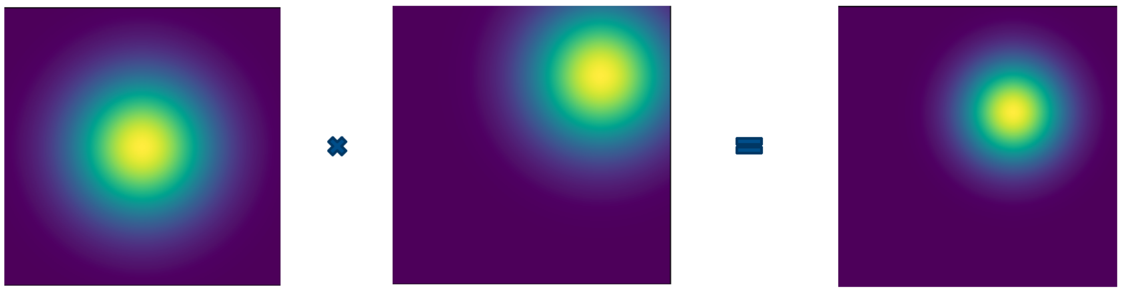

Figure 3.1.: The first two imgages are expert policies modeled by Gaussian distribution, then CEP computes a resultant distribution that fulfills two existing expert policies, which is the last image

Exploration Strategy

The first step of model-free policy search is defining the exploration strategy. In our work Reward Weighted Regression (RWR) is employed, and, hence, the exploration is performed through a multivariate Gaussian noise $\boldsymbol{\epsilon}_a$ in action space

$$\boldsymbol{\epsilon}_a \sim \mathcal{N}(0, \Sigma_a) \tag{3.2}$$

And hence, the exploration space $\mathcal{A} \in \mathcal{R}^n$ is modeled by the combination of the current action $\boldsymbol{a}_t$ and a multivariate Gaussian noise $\boldsymbol{\epsilon}_a$ in action space $\boldsymbol{a}^x$

$$\boldsymbol{\theta}_a = \boldsymbol{a}_t + \boldsymbol{\epsilon}_a, \tag{3.3}$$

where $\boldsymbol{\theta}_a \in \mathcal{R}^n$ denotes a sample or a vector parameter, and $\boldsymbol{\theta}_a^{[i]}(i = 1, 2, ...K)$ implies K samples at one time. For each sample $\boldsymbol{\theta}_a$, the first three joint accelerations belong to mobile base, the other seven joint values pertains to the 7-DOFs robot arm. As the figure of a simplified TIAGo+ model below, mobile base has two continuous joints which control the forward velocity, and one rotational joint controlling the orientation; the 7-DOFs are divided into one prismatic joint for the parallel gripper and six revolute joints realizing high agility.

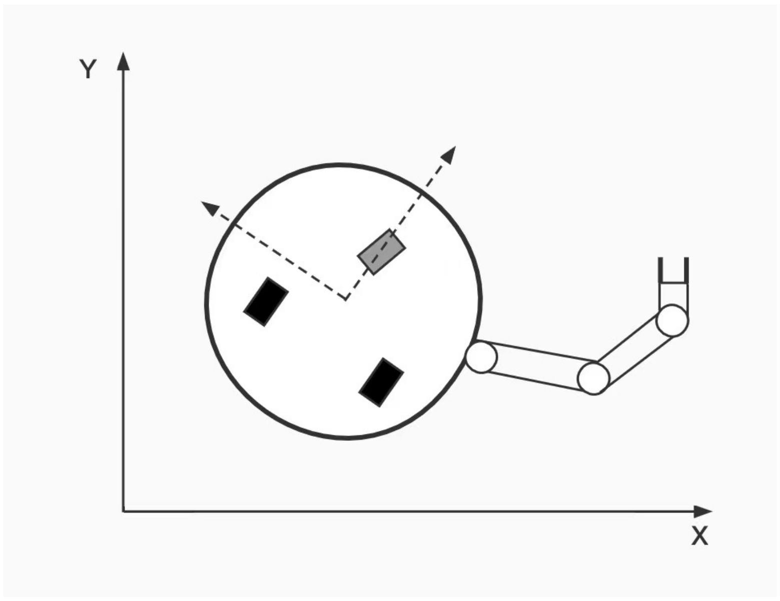

Figure 3.2.: TIAGo+ mobile manipulator with one robot arm and a mobile base

Policy Evaluation
Once the sample action $\boldsymbol{\theta}_a$ is generated, it should be mapped and then evaluated in different latent spaces $\mathcal{Z} \in \mathcal{R}^n$, so that the corresponding reward can be computed. As for the three joint actions responsible for the planning of mobile base, we replace the original three joints with two prismatic joints and one revolute joint $\boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{u}_r$ which are mounted on the center of the mobile base hypothetically, since for the navigation part the thesis merely aims to develop an reactive approach for whole body control of mobile manipulators, and, hence, this simplified assumption using holonomic constraints to replace nonholonomic constraints is an alternative. As to the other seven arm joints, only the final position of the end-effector will be considered, as the orientation of the end-effector is to hard be learned, and, hence, a fixed orientation is set for better convergence. Thus, formulating the final pose and orientation via homogeneorous matrix A, the final is given by

$$A = \begin{bmatrix} 1 & 0 & 0 & s_x \\ 0 & 1 & 0 & s_y \\ 0 & 0 & 1 & s_x \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where (s_x, s_y, s_z) denotes the used-defined final end-effector position.
Let $a_x \in \mathcal{R}^{10}$ denote one sampled action in joint space, in which $(\boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{u}_r)$ are three mobile base joints and $(\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_7)$ are seven robot arm joints:

$$a_x = [\boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{u}_r, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \boldsymbol{u}_4, \boldsymbol{u}_5, \boldsymbol{u}_6, \boldsymbol{u}_7]$$

Note: In order to explicitly define the objective in individual latent space respectively, let $\boldsymbol{u}_b^x = (\boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{u}_r)$ and $\boldsymbol{u}_a^x = (\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_7)$ for better formulation, which will be explained in next paragraph. The subscript b and a denote the part of controlling mobile base and robot arm respectively, the superscript implies the corresponding task space, here x represents in joint space.
Primarily, the original sampled action a_x is mapped through ***Selection_Map*** in order to set the objective in end-effector task space and planar space respectively, which is represented by $\boldsymbol{u}_b^x = \mathit{Selection\_Map\_B}(a_x)$ and $\boldsymbol{u}_a^x = \mathit{Selection\_Map\_A}(a_x)$. The following is the transformation between joint space to different latent spaces. Through forward kinematics the seven joint values can be mapped into end-effector task space:

$$\boldsymbol{u}_a^e = \mathit{ForwardKinematics\_Map}(\boldsymbol{u}_a^x)$$

As the holonomic constraints of two prismatic joints and one revolute joint, the transformation between joint space and planar space is simplily perform by a identity matrix:

$$\boldsymbol{u}_b^e = \mathit{Identity\_Map}(\boldsymbol{u}_a^x)$$

Then the mapped action in latent spaces can be evaluated according to the desired values. Let $\boldsymbol{\mu}_a^e$ and $\boldsymbol{\mu}_b^e$ denote the desired action in end-effector task space and planar space respectively. Through Operation Space Control (OSC) and PD controller, $\boldsymbol{\mu}_a^e$ is derived from

$$\boldsymbol{\mu}_a^e = \boldsymbol{J}_{ee}^T(\boldsymbol{q})\boldsymbol{M}_{x_{ee}}(\boldsymbol{q})(k_p(\boldsymbol{x} - \boldsymbol{x}_{des}) + k_v(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{des})) \tag{3.4}$$

where $\boldsymbol{x}_{des}$ is set as a valid and feasible position of the end-effector and $\dot{\boldsymbol{x}}_{des}$ as zero. $\boldsymbol{\mu}_b^e$ is also computed by PD control, given as

$$\boldsymbol{\mu}_b^e = K_p(X_{des} - X_{cur}) + K_v(V_{des} - V_{cur}) \tag{3.5}$$

where $X_{des} \in R^2$ is the local desired position in planar space and V_{des} is set as zero. Two different methods to generate a local target in planar space X_{des} is developed given a prior map with tree structure, i.e. **Cascade_Control** and **Tracking_Father**. Both ways require a prior map that provides heuristic information, which in our approach is generated by Rapid Random Search Tree (RRT) due to its exploration ability for the environment.

Cascade_Control It refers to the concept of LQR trees [] and Dynamic Window Approach [] as it finds the local target depending on an imaginary circle centering on the geometric center of the robot as the boundary and changes the local target continuously until the robot reaches the goal.
Given current position, end position and a RRT tree prior map, $Cascade\_control$ function returns a local target N_{local}. To regulate the movement, it's required to set the boundary of the dynamic circle as β and the reaching target threshold as Δd. For each planning step, the distances d between the current position N_{cur} of mobile base and the all of the nodes of the RRT tree $\boldsymbol{\tau}$ should be computed and stored using a priority queue $dist\_queue$ in which each pair contains a node N and its distance d and all of the pairs are sorted by increasing distances and also a hash map $dist\_map$ in which same data are stored for further quick search. Firstly, to compute current distance from the destination d_{end} and compare d_{end} with Δd, if $d_{end} \leq \beta$ then N_{local} is found as N_{end}; otherwise, to make use of the prior map for further planning. Popping out the node N_{son} with closest distance d_{son} from current position N_{cur} from $dist\_queue$ and computing the distance d_{cur} from goal N_{end}. In the case that the closest node is outside the circle boundary $\boldsymbol{\beta}$, then the local target N_{local} is founded. Otherwise, to iteratively track the father node N_{fat} of N_{son} until the node N_{fat} is outside the circle boundary or N_{fat} is close enough to the goal N_{end}. The whole logistic process can be found 2

```
Data: $N_{cur}$, $N_{end}$, $\boldsymbol{\tau}$
Result: $N_{local}$
Initialization $\Delta d$, $\beta$;
$d_{end} = CalDist(N_{cur}, N_{end})$;
if $d_{end} \leq \beta$ then
    $N_{local} \leftarrow N_{end}$ ;
    return $N_{local}$ ;
end
$dist\_map, dist\_queue \leftarrow DistMap(N_{cur}, \boldsymbol{\tau})$ ;
$N_{son}, d_{son} = dist\_queue.pop()$ ;
$d_{end} = CalDist(N_{son}, N_{end})$ ;
if $d_{son} \geq \boldsymbol{\beta}$ then
    $N_{local} \leftarrow N_{son}$ ;
end
while $d_{son} \leq \boldsymbol{\beta}$ do
    $N_{fat} \leftarrow FindFather(N_{son}, \boldsymbol{\tau})$;
    $d_{fat} = CalDist(N_{fat}, N_{end})$ ;
    $d_{cur} = CheckMap(dist\_map, N_{fat})$ ;
    if $d_{fat} \geq \Delta d$ then
        $N_{local} \leftarrow N_{fat}$;
        return $N_{local}$ ;
    else
        if $d_{fat} \leq \beta$ then
        $N_{local} \leftarrow N_{fat}$;
        return $N_{local}$ ;
        else
        $N_{fat} \leftarrow FindFather(N_{clost}, \boldsymbol{\tau})$;
        $d_{son} \leftarrow d_{fat}$ ;
        $N_{son} \leftarrow N_{fat}$ ;
    end
end
```

Algorithm 2: Cascade Control

Tracking Father This method takes full advantage of the father-son structure without extra tedious calculation. Given current position N_{cur}, target position N_{end} and prior map $\boldsymbol{\tau}$, firstly judging whether N_{cur} approaches N_{end} enough or not. If not, extracting the closest node N_{son} and its distance d_{son} from $dist\_queue$. Next, in a for-loop, tracking the father node N_{fat} and calculate its distance d_{cur} from goal N_{end}. Only in the case that N_{fat} is close enough to the end point N_{end}, in which N_{fat} is regard as the local target N_{local}, otherwise repeatedly tracking the father nodes along the RRT tree $\boldsymbol{\tau}$ for K times, and K is given by

$$K = \frac{N_{tree}}{N_{tree} - N_{traverse}} \tag{3.6}$$

where N_{tree} denotes the number of nodes of the entire RRT tree, $N_{traverse}$ is the number of nodes if tracking the current position N_{cur} until the end point N_{cur}, and, hence, as the robot moves to the goal, $N_{traverse}$ is decreasing and then parameter K is smaller as well.

```
Data: $N_{cur}$, $N_{end}$, $\boldsymbol{\tau}$
Result: $N_{local}$
Initialization $\Delta d$;
$d_{end} = CalDist(N_{cur}, N_{end})$;
if $d_{end} \leq \Delta d$ then
    $N_{local} \leftarrow N_{end}$ ;
    return $N_{local}$ ;
end
$dist\_map, dist\_queue \leftarrow DistMap(N_{cur}, \boldsymbol{\tau})$ ;
$N_{son}, d_{son} = dist\_queue.pop()$ ;
for iteration=1,2,..., K do
    $N_{fat} \leftarrow FindFather(N_{son}, \boldsymbol{\tau})$;
    $d_{cur} = CalDist(N_{fat}, N_{end})$ ;
    if $d_{cur} \geq \Delta d$ then
        $N_{local} \leftarrow N_{fat}$;
        return $N_{local}$ ;
    end
    $N_{son} \leftarrow N_{fat}$ ;
end
```

Algorithm 3: Tracking Father

Next, using the logarithmic probability of the mapped actions in latent space over the multivariate Gaussian distributions, whose mean values are the desired actions in latent space, i.e. $\boldsymbol{\mu}_a^e$ and $\boldsymbol{\mu}_b^e$. And hence, by defining two multivariate Gaussian distribution $N(\boldsymbol{\mu}_a^e, \Sigma_a)$ and $N(\boldsymbol{\mu}_b^e, \Sigma_b)$, the reward $\boldsymbol{R}_a^e$ and $\boldsymbol{R}_b^e$ are computed by

$$\boldsymbol{r}_a^e = \log(\boldsymbol{u}_a^e | N(\boldsymbol{\mu}_a^e, \Sigma_a)) \tag{3.7}$$

$$\boldsymbol{r}_b^e = \log(\boldsymbol{u}_b^e | N(\boldsymbol{\mu}_b^e, \Sigma_b)) \tag{3.8}$$

Therefore, the resultant reward is the weighted sum of these two rewards

$$\boldsymbol{R}^{[i]} = \alpha_a \boldsymbol{r}_a^e + \alpha_b \boldsymbol{r}_b^e \tag{3.9}$$

And hence, for N samples in batch computation, the whole dataset is

$$D_{ep} = \{a_x^{[i]}, R^{[i]}\}_{i=1...N}$$

Policy Update
After the data set D_{ep} is generated, the new mean value $\boldsymbol{a}_{t+1}$ can be updated based on previous $\boldsymbol{a}_t$ and D_{ep} through Reward Weighted Regression (RWR), and choose the the action a^* with highest reward.

$$a^* = \text{arg} \max_{\mathbf{a}} (D_{ep}) \tag{3.10}$$

4. Experiments

This section demonstrates two kinds of experiments, 2D planning and whole body control via Composable Energy Policies (CEP). Informally speaking, we try to answer these two questions:

- Which method to find the local target is better in CEP, $CascadeControl$ or $TrackingFather$?
- How is the performance of CEP in whole body control for mobile manipulators?

4.1. Environment

In order to observe the planning performance more clearly, firstly the algorithms are implemented using Matlibplot tools, in which the influence of dynamics of robot system can be ignored. Once the algorithms are debugged successfully and perform well, we employ these algorithms in PyBullet. As the purpose of the experiments is only testing the feasibilty of the algorithms, and hence, in Matlibplot and Pybullet, the environment is simple with only one static box object. Moreover, the start point and end point are defined by the user instead of using localization algorithms and sensor information.
A prior map is generated through Rapid Random Search Tree (RRT), and, hence, this prior map is stored as a tree structure to provide guidance for reactive motion generation.

4.2. 2D planning via CEP

As to Composable Energy Policies framework (CEP), two different methods chosing the local target $CascadeControl$ and $TrackingFather$ are used. We set the experiment parameter as follows: optimization steps as 5, sample number as 1000. For $CascadeControl$, the radius of the cascade circle is defined as the same radius as the mobile base, i.e

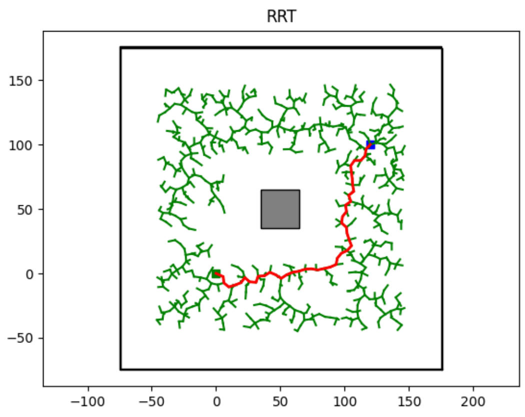

Figure 4.1.: RRT tree serves as a prior map. The green point is the start point, the blue point is the end point, and the red line denoted the path.

$\beta = 0.15m$, this parameter influences the smoothness and the ability to avoid objects. $TrackingFather$, the threshold to check whether the end point can be chosen as local target is defined as $\Delta d = 0.15$.
Therefore, firstly to compare the different methods to identify a local target, $CascadeControl$ and $TrackingFather$.

Based on the experiments, referring to 4.4 and 4.5, 4.8 and 4.9 as well, we find the robot has two different paths to reach the end point given a RRT tree which provides potential path to the goal. Besides, $CascadeControl$ and $TrackingFather$ can both guide the robot to the end point and achieve object avoidance, as the distance from end point decreases to zero. Comparing two different methods it's obvious that $TrackingFather$ results in a more smooth path and needs less iteration times to converge.

4.3. Whole Body Control via CEP

As for whole body control of mobile manipulators, we only concern about a 7-DOFs robot arm and a 3-DOFs mobile base. In order to evaluate the algorithm's performance on whole body control better, we ran 10000 times experiments. As for the combination of

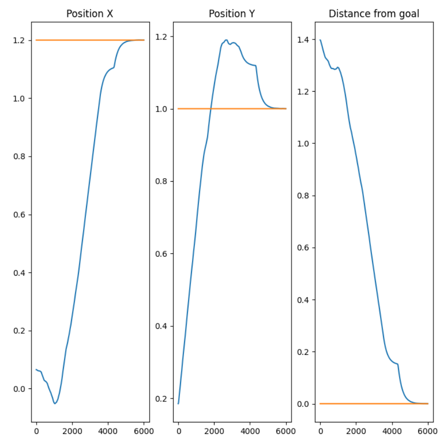

Figure 4.2.: Position 1 of Cascade control

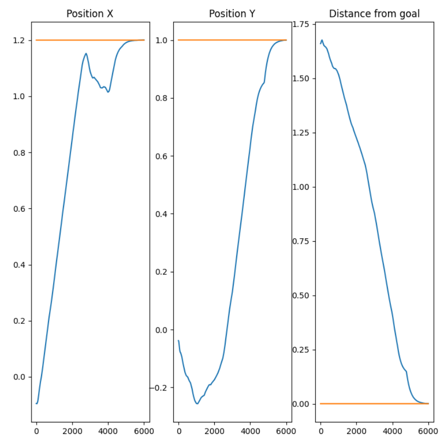

Figure 4.3.: Position 2 of Cascade control

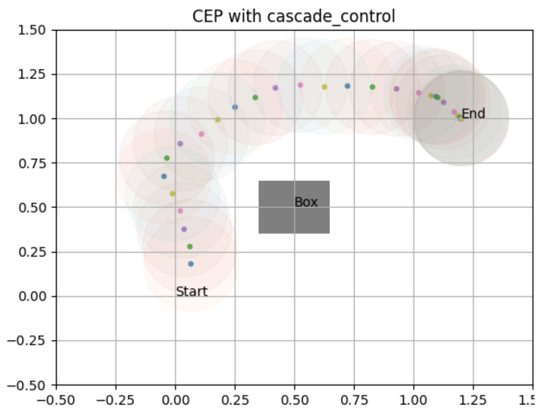

Figure 4.4.: Path 1 of Cascade control

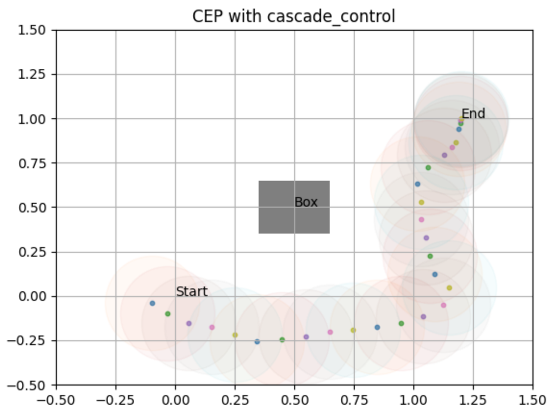

Figure 4.5.: Path 2 of Cascade control

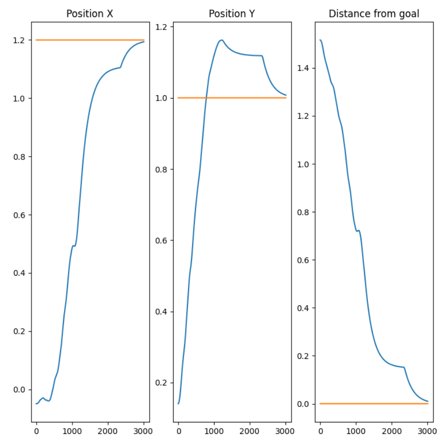

Figure 4.6.: Position 1 of Tracking father

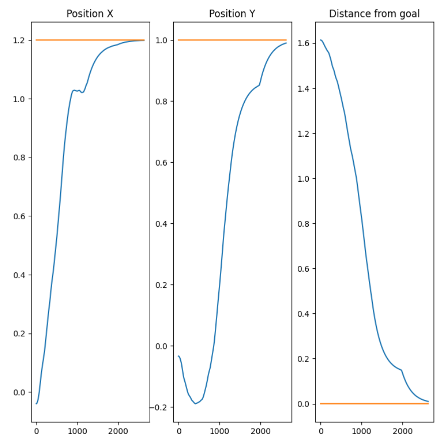

Figure 4.7.: Position 2 of Tracking father

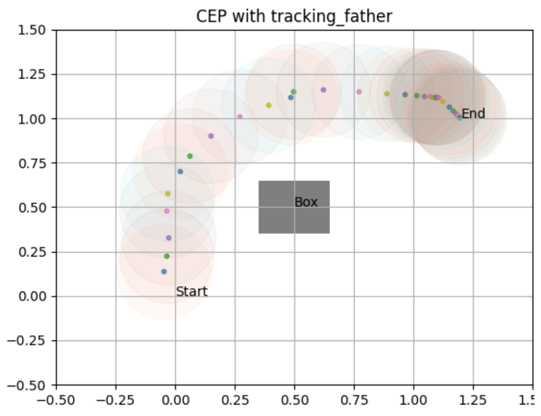

Figure 4.8.: Path 1 of Tracking father

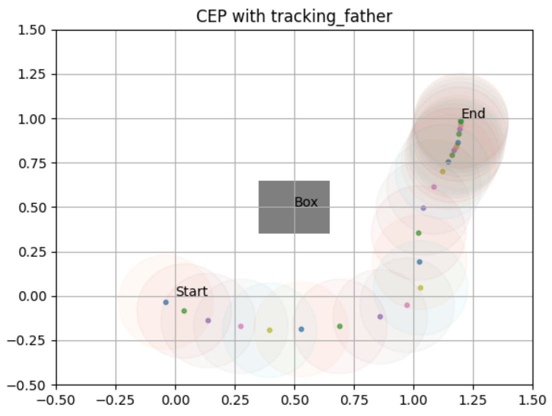

Figure 4.9.: Path 2 of Tracking father

two different energy components via CEP framework, we set the weights of two different rewards as $\boldsymbol{\beta}_a = 0.5$ and $\boldsymbol{\beta}_b = 0.5$.

According to the figure 4.10 demonstrating the convergence process of the position (x, y, x) of the end-effector and the distance from the target point, it's easily to figure out that CEP can realize a satisfied performance in target arrival, although the final error of distance still exits about $0.03cm$, CEP can fulfill a fundamental application in motion planning of mobile manipulators. The reason that the position x and y can converge more accurate than z is that we set a higher variance for z which is more relevant with $TaskGoto$ energy component and set a lower variance for $PathPlan$ energy component.

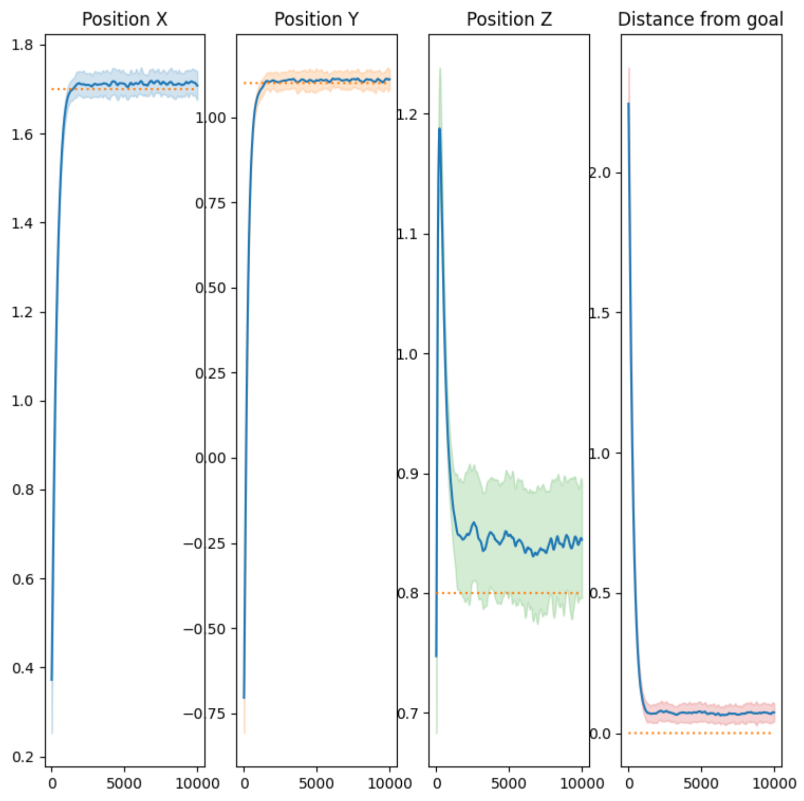

Figure 4.10.: Whole Body Control with CEP framework. Position x, y and z can be convergent, although position z still needs to be improved. The absolute distance from goal point is less than 0.03cm.

5. Discussion

In this thesis, we achieve two dimensional planning and whole body control based on a novel framework, Composable Energy Policies (CEP), for modular reactive motion generation, which can avoid local minima better than potential field methods and define specific objectives in different latent spaces explicitly. Besides, we also compare two different methods to get a local target with CEP. Moreover, we also found the difference between CEP and MPC, as CEP concentrate on the immediate motion planning or in other words, in short prediction and control horizon, in comparison, MPC compute actions in a control horizon based on a finite prediction horizon according to a specific objective function.
As for one of the importance parts in this thesis is choosing a local target for mobile base. Rapid Random Search Tree (RRT) is employed to provide a prior map as it has a good exploration of the whole environment. We use two methods , $CascadeControl$ and $TrackingFather$, to find the local target, and both can achieve target arrival and object avoidance smoothly.
Moreover, for whole body control of mobile manipulators, we accomplish it based on CEP framework, and it can reach the defined end point and avoid the static objects.

6. Outlook

Although we achieve 2D planning and whole body control using Composable Energy Policies framework, to accomplish better performance and be applicable to a more complicated situation, some modifications can be made further:

- Using different approaches to generate a prior map providing motion guidance, which can achieve target arrival and object avoidance
- As Composable Energy Policies can combine different policies smoothly, in order to deal with dynamic object and environment, for each time step more than one node can be treated as potential local target, and, hence, the robot has various choice when it encounters objects suddenly.
- As we a holonomic base to replace the real nonholonomic basic, and hence, a more realistic bicycle model should be implemented in 2D planning part further.

Bibliography

[1] M. Hvilshøj, S. Bøgh, O. Madsen, and M. Kristiansen, "The mobile robot "little helper": concepts, ideas and working principles," in *2009 IEEE Conference on Emerging Technologies & Factory Automation*, pp. 1–4, IEEE, 2009.

[2] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, A. Albu-Schäffer, and G. Hirzinger, "Real-time reactive motion generation based on variable attractor dynamics and shaped velocities," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3109–3116, IEEE, 2010.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.

[4] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg, "Real-time perception meets reactive motion generation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1864–1871, 2018.

[5] S. Haddadin, R. Belder, and A. Albu-Schaeffer, "Reactive motion generation for robots in dynamic environments," in *Proceedings*, 2011.

[6] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, pp. 489–494, IEEE, 2009.

[7] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*, pp. 4569–4574, IEEE, 2011.

[8] I. Duleba, "Modeling and control of mobile manipulators," *IFAC Proceedings Volumes*, vol. 33, no. 27, pp. 447–452, 2000.

[9] Y. Yamamoto and X. Yun, “Coordinating locomotion and manipulation of a mobile manipulator,” in *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*, pp. 2643–2648, IEEE, 1992.

[10] D. Katz, E. Horrell, Y. Yang, B. Burns, T. Buckley, A. Grishkan, V. Zhylkovskyy, O. Brock, and E. Learned-Miller, “The umass mobile manipulator uman: An experimental platform for autonomous mobile manipulation,” in *Workshop on manipulation in human environments at robotics: science and systems*, Citeseer, 2006.

[11] L. Peterson, D. Austin, and D. Kragic, “High-level control of a mobile manipulator for door opening,” in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 3, pp. 2333–2338, IEEE, 2000.

[12] L. C. Wang, L. S. Yong, and M. H. Ang, “Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment,” in *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*, pp. 821–826, IEEE, 2002.

[13] R. A. Knepper, S. S. Srinivasa, and M. T. Mason, “Hierarchical planning architectures for mobile manipulation tasks in indoor environments,” in *2010 IEEE International Conference on Robotics and Automation*, pp. 1985–1990, IEEE, 2010.

[14] S. Koenig and M. Likhachev, “D^* lite,” *Aaai/iaai*, vol. 15, 2002.

[15] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, “From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots,” in *2017 ieee international conference on robotics and automation (icra)*, pp. 1527–1533, IEEE, 2017.

[16] A. Iriondo, E. Lazkano, L. Susperregi, J. Urain, A. Fernandez, and J. Molina, “Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning,” *Applied Sciences*, vol. 9, no. 2, p. 348, 2019.

[17] J. Kindle, F. Furrer, T. Novkovic, J. J. Chung, R. Siegwart, and J. Nieto, “Whole-body control of a mobile manipulator using end-to-end reinforcement learning,” *arXiv preprint arXiv:2003.02637*, 2020.

[18] G. B. Avanzini, A. M. Zanchettin, and P. Rocco, “Reactive constrained model predictive control for redundant mobile manipulators,” in *Intelligent Autonomous Systems 13*, pp. 1301–1314, Springer, 2016.

[19] J. Urain, A. Li, P. Liu, C. D'Eramo, and J. Peters, "Composable energy policies for reactive motion generation and reinforcement learning," *arXiv preprint arXiv:2105.04962*, 2021.

[20] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[21] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary computation*, vol. 11, no. 1, pp. 1–18, 2003.

[22] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.

[23] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[24] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[25] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[26] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[27] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and automation*, vol. 14, no. 1, pp. 166–171, 1998.

[28] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2, pp. 995–1001, IEEE, 2000.

[29] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of guidance, control, and dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[30] E. Sabudin, R. Omar, H. CKAN, and C. K. Melor, "Potential field methods and their inherent approaches for path planning," *ARPN Journal of Engineering and Applied Sciences (2016)*, pp. 10801–10805, 2016.

[31] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[32] E.-H. Guechi, S. Bouzoualegh, Y. Zennir, and S. Blažič, "Mpc control and lq optimal control of a two-link robot arm: A comparative study," *Machines*, vol. 6, no. 3, p. 37, 2018.

[33] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *5th Annual Conference on Robot Learning*, 2021.

[34] F. Kuhne, W. F. Lages, and J. G. da Silva Jr, "Model predictive control of a mobile robot using linearization," in *Proceedings of mechatronics and robotics*, pp. 525–530, Citeseer, 2004.

[35] W. F. Lages and J. A. V. Alves, "Real-time control of a mobile robot using linearized model predictive control," *IFAC Proceedings Volumes*, vol. 39, no. 16, pp. 968–973, 2006.

A. Some Appendix

Observation Space	$q \in R^10$, denotes position of joints.
Action Space	Continues space, $a \in [-0.5, 0.5]^3$ for 2D planning and $a \in [-0.5, 0.5]^10$ for whole body control
Check Done	Check if the robot arrives at the end point without collision.
Dynamics	State transformation is based on Euler discretization $dq = dq + ddq * \Delta t$, $q = q + dq * \Delta t$.
reset	Initial the position of the robot randomly in a valid region

Table A.1.: Environment Setup.

CEP Parameters	
trial number	100
samples N	1000
time step Δt	0.005
optimization steps	20
episode horizon	100
variance of **TaskGoto**	10
variance of **PathPlan**	1.
$\boldsymbol{\beta}_a$	0.5
$\boldsymbol{\beta}_b$	0.5
RWR Parameters	
temperature β	0.1

Table A.2.: Algorithm parameters.