

Bimanual Control and Learning with Composable Energy Policies

Bimanual Control and Learning with Composable Energy Policies

Master thesis by Yifei Wang

Date of submission: 03 Jan 2022

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, Yifei Wang, 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.

Darmstadt, 03 Jan 2022


Yifei Wang

Master-thesis

für

Mr Yifei Wang, B.Sc.

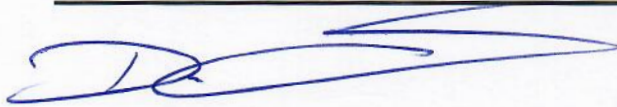
Theme: Bimanual Control and Learning with Composable Energy Policies

Bimanual Control and Learning with Composable Energy Policies

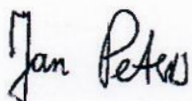
Composable Energy Policies (CEP) is a new paradigm for multi-objective robot control. CEP extends the concepts of artificial potential fields incorporating Monte-Carlo methods to compute the robot action. Up to now, CEP have been applied for 7DoF robot and there are no evidence of CEP working in higher DoF robots. In this direction, this project is introduced to evaluate the performance of CEP algorithms for Bimanual robots with 14 DoF. In the project, we aim to develop a set of energy priors for bimanual tasks and apply learning algorithms to improve bi-manipulation problems.

Begin: 07.05.2021
Submit: 07.11.2021

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, *Robotic-Systems*
Institute for Mechatronic Systems



Prof. Dr. Jan Peters
Intelligent Autonomous Systems

Abstract

Study on reactive motion generation of mobile robots and manipulator has a long history. In past years one of traditional solutions to these issues is to calculate corresponding actions over a sum of policies for multi-objective tasks. However, these policies are designed independent from each other and even during optimization also separately updated. Therefore these solutions may could cause conflicts or unexpected results in some complex cases. Composable Energy Policies (CEP) provides new insight to improve the performance in filed of reactive motion generation. Instead of sum of policies, CEP optimize the product over different policies to compute appropriate actions, which satisfies all tasks. In this project, we aims focus on the simple applications of CEP for a bimanual robot with 14 DOF along with extra joystick. Besides CEP framework, we also build obstacle map via depth camera and use them as for the obstacle.

Zusammenfassung

Studien zur reaktiven Bewegungserzeugung von mobilen Robotern und Manipulatoren haben eine lange Geschichte. Von Post steht eine der traditionellen Lösungen für diese Probleme darin, die entsprechende Bewegungen des Robots nach einer Summe von bestimmter Strategies für mehrere verschiedene Aufgaben zu errechnen. Composable Energy Policies (CEP) liefern neue Lösungen dazu. Anstelle einer Summe von Strategies optimiert CEP das Produkt über verschiedene Strategies, um geeignete Aktionen des Agents zu berechnen, die alle Aufgaben gleichzeitig erfüllen. In diesem Projekt konzentrieren wir uns auf die einfachen Anwendungen von CEP für einen bimanuellen Roboter mit 14 DOF mit einem zusätzlichen Joystick. Neben dem CEP-Framework erstellen wir auch eine Hinderniskarte über eine Tiefenkamera und verwenden sie mit dieser Hinderniskarte bei Hindernisvermeidung von Robot.

Contents

1	Introduction	2
1.1	Related Work	3
1.2	Overview	5
2	Foundations	7
2.1	Composable Energy Policies	7
2.2	Camera Calibration	11
2.3	Artificial Potential Fields	14
3	Visual processing	17
3.1	Camera Calibration	17
3.2	Obstacles Map	23
4	Leaves for motion generation	31
4.1	Leaf for Task Go-To Far	32
4.2	Leaf for Task Joystick	32
4.3	Leaf for Task Go-To Close	33
4.4	Key parameters for Three Leaves	34
5	Experiments	37
5.1	Experiment Setup	37
5.2	Tests for the CEP control	38
5.3	Tests for obstacle avoidance	39
6	Discussion	41
7	Outlook	42

Figures and Tables

List of Figures

1.1	Overview of Experiment Platform Robot Darias	2
2.1	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.	13
2.2	Combination of the Repulsive Force and Attractive Force	14
3.1	Five Coordinate Systems In Camera Calibration	17
3.2	Five Coordinate Systems In Camera Calibration	19
3.3	Snapshots of the Calibration Pattern From Different Poses	20
3.4	Transform Chain Inside Five Coordinate Systems	21
3.5	Two Coordinate Systems on the Chessboard Plane	22
3.6	Workflow for the Extrinsic Matrix's Acquisition	23
3.7	Workflow of Generation of Obstacle Map	24
3.8	Workflow of Generation of the point clouds from the Robots' links	24
3.9	Difference by Publisher's Frequency	26
3.10	Workflow for Post-processing of Point Clouds	27
3.11	Detailed Processing Steps for Two Point Clouds	28
3.12	Processed Two Point Cloud from Environment and Robot Links	28



3.13 Final Obstacle Octomap	29
4.1 Mapping Tree from the Configuration Space	33
4.2 The Projection of Buttons with Different Functions	34
4.3 Mapping Tree from the Configuration Space	35
5.1 Task Go-To Close With large Weight Coefficient	38
5.2 Sum of Three Leaves	39
5.3 The progress of Obstacle Avoidance With Single Sphere	39
5.4 Sum of Three Leaves	40

List of Tables

1 Introduction

Due to expanding demands for bimanual robot in field of research and commercial applications, the robot are required to master more and more skill than before and accomplish more complex task. Nowadays these robots should finish tasks containing different step or even finish tasks in parallel. Such as fetch a bottle and pour water into cup. This task consists of a few sequential steps: reach to the bottle, grab bottle, move close to cup and finally rotate the wrist to pour the water right into cup. Apart from these sequential objectives, the manipulator should also avoid collision between the arm links themselves and the obstacles in the environments.

In our work we use robot Darias from IAS institute of TU Darmstadt as the main ex-

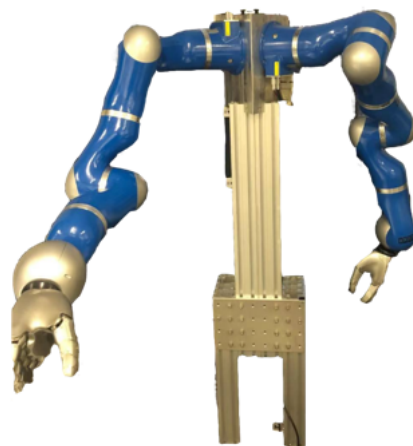


Figure 1.1: Overview of Experiment Platform Robot Darias

periment platform. This robot has two arms and totally 14 rotation joints. Arms are manufactured by Kuka and equipped with a delicate hand as end effector, which has five

flexible fingers to accomplish complex task accurately, such as picking and grabbing objects. Besides the robot itself, this platform also has powerful visual perception components. For the acquisition of simple information from environment we use the depth camera from Intel RealSense as the main RGB image-stream with 30 Hz. As for calibration and more complicated motion capture we use commercial solution from OptiTrack motion capture system. OptiTrack enable extremely precise motion capture via tracking markers attached to the targets' surface.

1.1 Related Work

Currently the use of manipulator in industrial productions and complicated applications' scenes grows rapidly in the past years. Persistent Growth of applications requires bimanual robots to learn more and more skills and be able to solve task with multi objectives in parallel. In our case we mainly focus on the motion generation tasks, such as reach certain position, and obstacles avoidance problems. In this study filed, there are two kinds of mainstream theories to solve this problems: path planning method and reactive motion generation.[1]

Path planning approach aims that: the robot decides a collision-free and optimal global trajectory from the start to the destination. During the computation will the algorithm convert various sensors' feedback into useful perception information and follow specific criteria defining safety and efficiency. In the planning the robot should based on certain criteria to compute the required cost time and efficiency to avoid some necessary obstacles. According to the completeness of perceptions of the environment, the classic path planning could be divided into global path planning and local path planning.

Compared to the path planning approach, the reactive motion generation method is more suitable for rapidly changing environment and robot with middle or short ranged sensors, such as integrated depth camera. Reactive motion generation method requires perception input with much higher refresh rate aiming for quick response to the dynamic outside world.

In this work we investigate mainly into the framework suitable for local path planning. Because bimanual robots face often to dynamically changing environment and the target pose of the robot arm would also change due to the progress of the whole task. For example when the robot is supposed to grab a bottle or cup, in order to smooth the action of robot's fingers, the robot should response quickly enough to adjust its behaviors.

1.1.1 Path Planning

In past decades, path planning for the manipulators and mobile robots is constantly discussed and improved by researchers. Many researchers optimized existing methods or proposed new approach to balance the merits and drawbacks of path planning algorithms. Nowadays there are three mainstream sort of algorithms to solve this problems: graph search algorithms, like A^* [2], RRT and B^* ; bionic approaches as genetic algorithm [3] along with ant colony algorithm [4] and classical approaches as artificial potential field. Traditional path Planning approaches [5] [6] aims to compute the whole trajectory from the start to the goal and ensure that whole path is collision-free. This kind of framework is often associated with large computation's burden and lack of flexibility when dealing with dynamically changing environment.

A^* algorithm and other similar algorithms from graph searching approach are most well known path planning methods and could be easily applied with existing grid map [2]. However, the application of these algorithms are limited due to the huge computation cost, especially for time-sensitive scenes. To overcome this drawback many optimization-based approaches were proposed, such as CHOMP (Covariant Hamiltonian Optimization for Motion Planning) [7], using gradient descending method to generate and optimize path for robot. Schulman used convex optimization functions to learn from naive straight-forward initial trajectory. The key of this algorithm is applying penalty to the path with collision and dynamically changing the penalty coefficients[8].

Path planning methods benefits from its simple form and also suffer from corresponding relatively large computation cost. Besides path planning approach still proposes trajectory in long-term sight and could not provide satisfying result in case of dynamic environment.

1.1.2 Reactive Motion Planning

Compared to path planning method, reactive motion planning has much shorter execution cycles and better adaption to robot with high Degree-of-Freedom and complex sophisticated real-time scenarios [1].

Artificial potential field is one of most popular approaches in field of reactive motion planning [9] [10] [11]. Artificial potential field method is intuitive and easy to build corresponding potential field with given map. Compared to classical path planning method it has little burden on computation. Furthermore the calculations in this approach are barely euclidean distance, which could be easily vectorized for optimization by many matrix calculation libraries, such as Numpy and Eigen. However, this approach could

not ensure the long-term trajectory fulfilling tasks' requirements and could be stuck in local minimum cases sometimes. Even though, there are some improved methods, such as Chanclou's method, which provides two planner separately for both local and global path planning.

1.2 Overview

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 of our approach.

In this thesis the basic theoretical framework is based on the CEP (composable energy policies) framework [12] and our work focus mainly on the simple application of CEP frame with extra joystick for human to control some behavior of robot. CEP provide new insight to solve sophisticated tasks and complete framework to combine many delicate techniques into together: operational space control of robot joints, reactive motion planning, optimization for motion planning and so on. CEP enables the robot to learn from the motion primitives from subsequent task in parallel and combine the policies of these relatively independent motion policy into single one by a product of their policies. The optimization of motion generation is achieved by this single policy. This CEP framework also provides flexibility to deal with task defined in different action spaces and transfer them into another action space. For example, end-effector related task like releasing and grasping is defined at end-effector itself and should be converted into configuration space for following control, while the obstacle avoidance task for the whole robot arms is defined at the task space. CEP allows user to define convenient mapping between different space. The main work of this paper can be summarized as follows:

- 1) Use depth camera to generate obstacle map.
- 2) Use artificial potential field for obstacle avoidance.
- 3) Applying CEP for control of two robot arms with joystick.

This thesis has been divided into six parts. Chapter 1 start with the current state of the related study and presents the motivation of this work. Chapter 2 lists the theoretical foundations of operational space control(OSC), composable energy policies(CEP), artificial potential filed(APF) and camera calibration. Chapter 3 explains the concrete step in our experiment in details. Chapter 4 and Chapter 5 focus on the results of the experiments. Chapter 6 introduces briefly the outlook for the future work.

2 Foundations

This section contains three parts: robot control related theoretical foundation, obstacle avoidance and camera calibration related theory.

2.1 Composable Energy Policies

Composable Energy Policies(CEP) [12] enables a new framework, which could combines different policies into one single policy with the product of these independent policies. Compared to CEP, the previous work solve the reactive generation problems by maximizing the sum of bunch of separate policies, which focus only on their own task and lack of interaction. Every policy would follow its own criteria to search the optimal actions. This independence might lead to non-optimal actions or even cause conflicts between different policies. For example, the attractive force pointing to the destination may lead the end effector to into local minimum and stay trapped.

For CEP, the optimization process is to search an action which could maximize the product of few expert policies.

$$a^* = \underset{a}{\operatorname{arg\,max}} \prod_k \pi_k(a|s) \quad (2.1)$$

where $\pi_k(a|s)$ is one of expert policy with given system state.

We start by defining a bunch of independent stochastic policies. Let $\pi_1(\mathbf{a}|s), \dots, \pi_k(\mathbf{a}|s)$ to be the desired independent stochastic policies, which are constructed in form of Boltzmann distribution.

$$\pi_i(\mathbf{a}|s) = \exp(E_i(\mathbf{a}, s)) \frac{1}{Z_i(s)} \quad (2.2)$$

with $E_i(\mathbf{a}, \mathbf{s}) : \mathcal{S} \times \mathcal{A} \rightarrow R$ is an arbitrarily energy function for this policy and $Z_i(\mathbf{s}) = \int_{\mathbf{a}} \exp(E(\mathbf{a}; \mathbf{s})) d\mathbf{a}$ is integral of this energy function and functions as factor for normalization. According to energy-based learning [13], Boltzmann equation could be used to describe arbitrary functions under appropriate parameter. Furthermore, the importance of different policies could be valued through different weight coefficients. Hence, from Equation 2.1 and 2.2, we rewrite the formulation as the below:

$$\pi(\mathbf{a}|\mathbf{s}) = \prod_{k:0 \rightarrow K} \pi_k(\mathbf{a}|\mathbf{s})^{\beta_k} \propto \exp\left(\sum_k \beta_k E_k(\mathbf{a}; \mathbf{s})\right) \quad (2.3)$$

Thanks to Boltzmann equation in exponential form, this product of expert policy could be converted into the sum of each policy component with logarithmic operation. This kind of linear form also contributes to acceleration of calculation. Because the each policy could be designed as independent progress and run in parallel to make the best use of processor;s potential performance.

As discussed in Chapter 1, for reactive motion planning problem all the policies haven been assumed to be in the same action-space. For example, for the obstacles avoidance in global path the policy is defined in joint space, whereas for grasping task for manipulator with hand-shaped end effector in task space. In order to solve this problem, the mapping between these two space is necessary, namely inverse kinematic Matrix and pseudo-inverse Jacobian matrix with given joint states of robot. Therefore, it is necessary to define appropriate mapping functions between different action-states.

Firstly we define an stochastic policy $\pi^z(\mathbf{a}^z|\mathbf{s}^z)$ in latent space, namely task space, $(\mathbf{a}^z|\mathbf{s}^z) \in \mathcal{Z}$. Next we define the mapping, which transforms the action-states pair from the configuration space $(\mathbf{a}^x|\mathbf{s}^x) \in \mathcal{X}$ to the latent space \mathcal{Z} . The corresponding mapping functions shows like $\mathbf{f}_x^z : \mathcal{X} \rightarrow \mathcal{Z}$. Therefore we can rewrite the Equation 2.2 in the task space

$$\pi^x(\mathbf{a}^x|\mathbf{s}^x) = \pi^z(\mathbf{a}^z|\mathbf{s}^z) \sqrt{\det(\mathbf{J}^T \mathbf{J})} \quad (2.4)$$

$$\pi^z(\mathbf{f}_x^z(\mathbf{a}^x|\mathbf{s}^x)) = \pi^z(\mathbf{a}^z|\mathbf{s}^z) \sqrt{\det(\mathbf{J}^T \mathbf{J})} \quad (2.5)$$

where the $\mathbf{J} = \partial \mathbf{f}_x^z(\mathbf{a}^x, \mathbf{s}^x) / \partial \mathbf{a}^x$ stands for the Jacobian expression of \mathbf{f}_x^z in action \mathbf{a}^x . With Equation 2.5, we could transform the Equation into :

$$\pi^x(\mathbf{a}^x|\mathbf{s}^x) = \exp(E^z(\mathbf{a}^z, \mathbf{s}^z)) \frac{1}{Z(\mathbf{s}^x)} = \exp(E^z(\mathbf{f}_x^z(\mathbf{a}^x, \mathbf{s}^x))) \frac{1}{Z_i(\mathbf{s}^x)} \quad (2.6)$$

with

$$Z = \frac{\int_{\mathbf{a}^z} \exp(E^z(\mathbf{a}^z, \mathbf{s}^z)) d\mathbf{a}^z}{\sqrt{\det(\mathbf{J}^T \mathbf{J})}} \quad (2.7)$$

As the Equation above describes, any action-states pair could be easily transferred from task space to expected latent space. This mapping functions enable the simple and aligned representation of policies. Besides we could also calculate corresponding log-probability of the action with given energy functions.

Compared to other methods, for the converse of action from latent space to the task space, there is no explicit inverse mapping function for \mathbf{f}_x^z . Instead, the implicit form is listed below:

$$\mathbf{a}^x = \arg \max_{\mathbf{a}^x} E^z(\mathbf{f}_x^z(\mathbf{a}^x, \mathbf{s}^x)) \quad (2.8)$$

Given a bunch of policies defined in different task space $\mathcal{Z}_1, \mathcal{Z}_2, \dots, \mathcal{Z}_N$. According to all the conversion above, we could the final policy over the product of independent policies:

$$\pi^x(\mathbf{a}^x | \mathbf{s}^x) = \exp\left(\sum_{k=1}^K \beta_k E^{z_k}(\mathbf{f}_x^{z_k}(\mathbf{a}^x, \mathbf{s}^x))\right) \frac{1}{Z_i(\mathbf{s}^x)} \quad (2.9)$$

The optimization procedure could be summarized as the equation below.

$$\mathbf{a}_{ML}^x = \arg \max_{\mathbf{a}} \prod_{k=0}^K \pi_k(\mathbf{a}^x | \mathbf{s}^x)^{\beta_k}, \beta_k > 0. \quad (2.10)$$

where \mathbf{a}^x presents the current state in task space, β_k is the weight coefficient. This optimization problem could be solved with the help from Maximum Likelihood Estimation (MLE) and cross entropy optimization approach [14].

$$\theta_{t+1} = \arg \max_{\theta_t} E_{p(\mathbf{a}^x | \theta_t)} [\log(\prod_{k=0}^k \pi_k(\mathbf{a}^k | \mathbf{s}^x)^{\beta_k})] \propto E_{p(\mathbf{a}^x | \theta_t)} [\beta_k E^{z_k}(\mathbf{f}_x^{z_k}(\mathbf{a}^x, \mathbf{s}^x))]. \quad (2.11)$$

As the equation above presented, appropriate energy functions are definitely independent from each other. This character enables the parallelization of processing of these energy functions. Multiprocessing contributes here largely to cutting off the execution cycle and enhancing the adaption to the reactive motion generation algorithms. Besides CPU, GPU could also help to save tons of computational time cost, especially in case with large dimensional matrix computation.

CEP for Reactive Motion Generation

In this work one main goal is to integrate reactive motion planning algorithm right into the CEP frame. For classic robot control joint value and joint speed are often chosen as the state, while the acceleration of the joint as the action. Therefore the corresponding mapping between the task space and latent space should be determined.

In this thesis, the whole robot system would be treated as pure second-order dynamic system. We study only on the behavior of the robot kinematics, while the robot dynamics would be ignored.

$$\ddot{\mathbf{q}} = g(\mathbf{q}, \dot{\mathbf{q}}) \quad (2.12)$$

with \mathbf{q} is the joint state, $\dot{\mathbf{q}}$ is the joint velocity and $\ddot{\mathbf{q}}$ is the joint acceleration. In common latent state-action space, namely joint space, the state and action are $(\mathbf{q}, \dot{\mathbf{q}})$ and $\ddot{\mathbf{q}}$. The main task of bimanual robot is mainly achieved by changing the pose of the hand-shaped end effector. Therefore the state from task space is $(\mathbf{x}, \dot{\mathbf{x}})$ and action refers to $\ddot{\mathbf{x}}$.

According to the robot kinematics, we could determine: $\mathbf{x} = \mathbf{f}_{fw}(\mathbf{q})$. \mathbf{f}_{fw} presents the forward kinematics of the robot. We could get the derivation of this equation.

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \quad (2.13)$$

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} \approx \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} \quad (2.14)$$

where $\mathbf{J} = \partial\mathbf{x}/\partial\mathbf{q}$ stands for the Jacobian matrix with given robot pose. In order to simplify the computation, the acceleration of end effector here equals to the product of Jacobian matrix and joint acceleration. Because the derivative of Jacobian is too small and could be ignored.

Theoretically, the action-state pair should be the same as expression above. However, in our application, the main three tasks is designed based on the joint speed as the action. In order to save computation sources and avoid extra conversion, the action should be set to the joint speed $\mathbf{a} = \dot{\mathbf{q}}$ and joint speed along with joint value as the state $\mathbf{s} = (\mathbf{q}, \dot{\mathbf{q}})$.

2.1.1 Lie Group and Lie Algebra

In order to solve estimation problems in robotics, such as modeling of states and estimation about uncertainties. In the past study researchers developed a manifold, which is smooth plane of Lie group. Based on the Lie theory, calculus equations can be constructed, which are mainly used in rotation $SO(3)$ and translation $SE(3)$. In modern robotics, the application of Lie group and Lie algebra is widely applied especially to solve pose related

problems [15]. In this work, we mainly focus the application of lie algebra to calculate required speed from one pose to another in 3D frame. The three-by-three Lie group $SO(3)$ represents exactly rotation in euclidean space, where $SE(3)$ refers to the six-dimensional euclidean matrix including the rotation part and translation part.

$$SO(3) = \{R \in \mathbb{R}^{3 \times 3} | R \times R = I | \det(R) = 1\} \quad (2.15)$$

$$SE(3) = \left\{ T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4} | R \in \mathbb{R}^{3 \times 3}, t \in \mathbb{R}^3 \right\} \quad (2.16)$$

The Lie algebra $\mathfrak{se}(3)$ and group $SE(3)$ could be converted into each other via via the exponential and log operations [8]. The conversion $\mathfrak{se}(3) \rightarrow SE(3)$ is exponential; $SE(3) \rightarrow \mathfrak{se}(3)$

$$\text{islogoperation.}\mathfrak{se}(3) = \left\{ \xi = \begin{bmatrix} \rho \\ \phi \end{bmatrix} \in \mathbb{R}^6, \rho \in \mathbb{R}^3, \phi \in \mathfrak{so}(3), \xi^\wedge = \begin{bmatrix} \phi^\wedge & \rho \\ 0^H & 0 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \right\} \quad (2.17)$$

where ξ refers to a 6-by-1 spinor containing line speed and angular speed. When two 6D pose in world frame is given, through Lie algebra we could compute expected speed, with which one pose could transformed into another.

2.2 Camera Calibration

This section begin with the camera calibration, namely hand-eye calibration problem. Meanwhile, the transformation of different coordinate systems will be discussed in this part. The following part focuses mainly on the generation of the octomap for the obstacle avoidance, which based on the point cloud data collected by the Intel real-sense camera. Further processing of these point cloud data is intended to eliminate points from robot arms. Because these points don't belong to obstacle groups and may cause unexpected error in obstacle avoidance.

2.2.1 Perspective Camera Model

The perspective camera model or pinhole camera model [?] is a simple but widely applied camera model, which simplifies the projection relationship of lights and also gives a simple but clear mathematical description of the transformation matrix between the three-dimensional world coordinate system and the two-dimensional image plane system. There are three important coordinate systems, namely the world coordinate system, the camera coordinate system, and the image coordinate system.

2.2.2 Extrinsic and Intrinsic Matrix

The acquisition of extrinsic and intrinsic parameters is a necessary step in camera calibration. Intel SDK Toolbox provides a look-up function to get a default value of this two-parameter matrix. In order to get more precise parameters, a traditional calibration procedure from [?] would be performed.

As the figure below shows, the intrinsic matrix represents the projection relationship between the image plane and the camera coordinate system. In order to simplify the mathematical equation, a conversion to homogeneous coordinates would be used. The original two-dimensional image coordinate system would be extended to a three-dimensional form. Normally, an additional conversion to pixel coordinate system would be performed, where its origin point is located at the upper-right corner of the image plane. Because the pixel coordinate system is widely used in the field of image processing, like the OpenCV library. Lately, the acquisition of the extrinsic matrix would be based on the pixel coordinate system and OpenCV library built-in functions.

We start by defining the intrinsic matrix, which describes the transform between the pixel coordinate and the image plane. This matrix only relates to the camera itself instead of the pose of the camera. For example, important camera parameters like principal point offset, focal length, and so on. As the equation shown above, this 3-by-3 matrix K is the expected intrinsic matrix.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{Z_c} \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \frac{1}{Z_c} K \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} \quad (2.18)$$

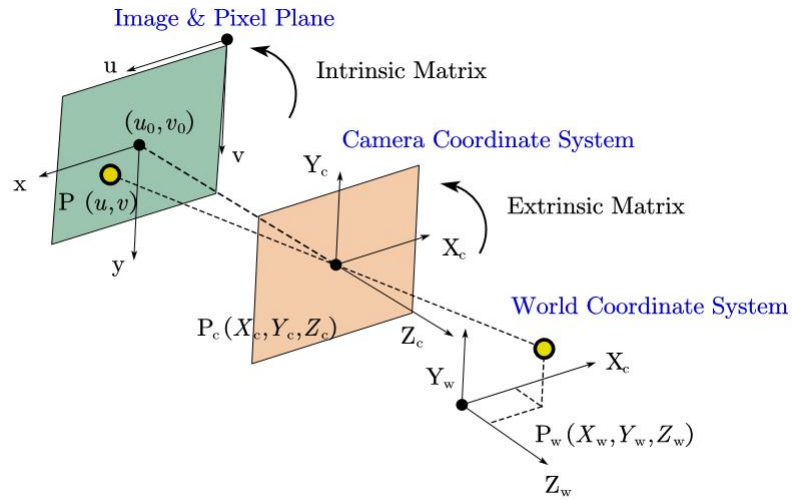


Figure 2.1: RRT tree serves as a prior map. The green point is the start point, the blue point is the end point, and the red line denoted the path.

where K is the expected intrinsic matrix, Z_c is the depth of objective position in the camera frame; u, v is the coordinates in pixel plane; $X_c Y_c Z_c$ is the coordinates in camera frame; c_x, c_y is the principal point offset; f_x, f_y is the x-axis and y-axis focal length of the camera in pixels.

The extrinsic matrix describe the projective relationship between the three-dimensional camera coordinate and world coordinate system. Two parameter identify uniquely this transformation : rotation matrix and translation vector. Translation vector refers the vector between the origins of the two frames and rotation matrix makes axes of two frames, namely camera coordinates system and world coordinates system into alignment [16].

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} R & T \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} \quad (2.19)$$

Given the skew part in intrinsic matrix is zero, When we combine the equation 2.19 and 2.18, we could get a simple but clear expression of projective relationship between pixel

coordinate system and world coordinate system. With help of this transformation, we could get any points' pose in world frame. In Chapter 3 we would introduce the details about how to acquire these two important camera matrices in our experiment setup and their usage in generation of map for obstacle avoidance.

2.3 Artificial Potential Fields

Artificial Potential Fields is published by Khati in 1986 and a widely used method for real time path planing and obstacle avoidance problems in field of mobile robots and manipulators[17]. In artificial potential fields the end-effector or mobile robot is moving inside of field of force: the target position for the end-effector would be treated as attractive pole, whereas the surface of all obstacles as repulsive field to the robot arm links. These two filed sum to be the final artificial potential field.

The force applied on the end-effector equals gradient of field at current position. In

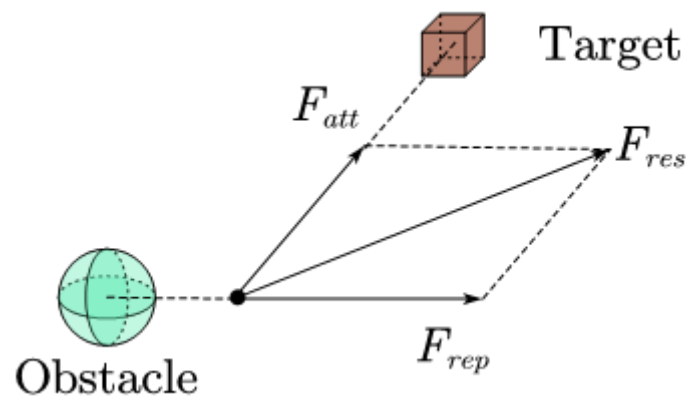


Figure 2.2: Combination of the Repulsive Force and Attractive Force

general, the manipulator will move towards the target position. When it get close to the obstacles, the repulsive force would be dominating and push the manipulator away from the obstacles. For every point, the force applied on the end-effector equals to the sum of attractive force and repulsive force. Therefore, the key of artificial potential field is how to generate appropriate attractive and repulsive field in three-dimensional real world.

2.3.1 Attractive Force

Attractive field generates corresponding attractive force, which leads the robot to expected target position. We define this attractive force function as a function of distance between the end-effector and target position. When the distance increases, the gravity potential also grows up. When the distance decreases, the potential also drops. Normally there is a simple and popular attractive field meetings our needs: quadratic potential fields, as the equation below shows.

$$U_{att}(q) = \begin{cases} \frac{1}{2}\zeta d^2(q, q_{goal}) & d(q, q_{goal}) \leq d_{goal}^* \\ d_{goal}^* \zeta d(q, q_{goal}) - \frac{1}{2}\zeta (d_{goal}^*)^2 & d(q, q_{goal}) > d_{goal}^* \end{cases} \quad (2.20)$$

where d_{goal} is an threshold, ζ is scale factor; q, q_{goal} is the current robot position and target position.

As the equations above and below shown, this attractive field and force are both piecewise function based on the distance. When the distance is smaller than the threshold, the attractive force is linear to the distance. When the end-effector get closer to the target, the attractive force would decrease and robot's movement would also be slowed down. Hence, it could be easier for the robot to adjust its pose and get more change to reach the target. Meanwhile, when the distance gets larger than the threshold, in order to avoid too large attractive force, the quadratic shaped potential would be modified as linear form.

$$\nabla U_{att}(q) = \begin{cases} \zeta (q - q_{goal}) & d(q, q_{goal}) \leq d_{goal}^* \\ \frac{d_{goal}^* \zeta d(q, q_{goal})}{d(q, q_{goal})} & d(q, q_{goal}) > d_{goal}^* \end{cases} \quad (2.21)$$

For the real time path planing, the attractive field and force should be updated the same time. These two functions are both only related to the target position and current robot state. Once the controller gives the desired robot's pose, the attractive field part should be calculated.

2.3.2 Repulsive Force

Different from attractive field, repulsive field strongly relates to current environment information, such the three-dimensional coordinates of the obstacle's surface. Based on processed environment's info, such camera data along with the depth data, we should

generate corresponding repulsive field for the obstacle avoidance. When the robot runs close to the obstacles' surface, the repulsive forces would push the target away from the obstacles. In contrast, when the robot is far enough from the obstacles, the repulsive should keep in zero in order to simply the calculation, especially, in case like real-time path planing problem, the generation of potential field should be simple enough and time-saving in order to response timely to complex and dynamically changing environments. Similar to Equation 2.20 and 2.21, let repulsive field and force also functions of the distance between the robot and obstacle.

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta \left(\frac{1}{D(q)} - \frac{1}{Q^*} \right)^2, & D(q) \leq Q^* \\ 0, & D(q) > Q^* \end{cases} \quad (2.22)$$

$$\nabla U_{rep}(q) = \begin{cases} \eta \left(\frac{1}{q^*} - \frac{1}{D(q)} \right) \frac{1}{D^2(q)} \nabla D(q), & D(q) \leq Q^* \\ 0, & D(q) > Q^* \end{cases} \quad (2.23)$$

Where $D(q)$ is the distance between between the robot and obstacle, Q^* is the threshold. When robot keeps away from the obstacles, the repulsive force would be deactivated. When the robot moves closer, the repulsive would increase rapidly in order to avoid collision.

3 Visual processing

This section describes two parts: camera calibration of Intel RealSense depth camera and generation of obstacles map in form of octree. Camera calibration includes acquisition of the extrinsic and intrinsic parameters. As for the obstacles map it contains the necessary step about post processing of point clouds.

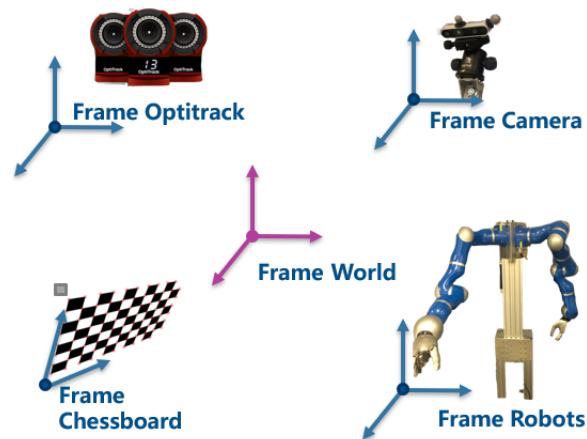


Figure 3.1: Five Coordinate Systems In Camera Calibration

3.1 Camera Calibration

As the mentioned in Chapter 2, the main goal of camera calibration in this work is to get intrinsic and extrinsic matrix for the Intel RealSense depth camera. To accomplish these

tasks, in our case there are two sets of hardware in function: OptiTrack camera system and Intel RealSense depth camera.

3.1.1 Hardware Setup

In this work, there are two sets of camera systems: OptiTrack motion capture System and Intel RealSense depth camera. As in section 2.2.2 mentioned, the OptiTrack is only used for early stage camera calibration, where as the Intel RealSense camera is used for generation of obstacle maps.

OptiTrack is a mature and complete motion capture solution to acquire object's three-dimensional coordinate via camera and markers[18]. In our case, OptiTrack motion capture system is applied to acquire the transform matrix between world frame and camera frame. The Intel depth camera with marker mounted on its surface, the center of markers is placed as close enough to the center of depth optical axis. Under this circumstance, the OptiTrack system knows the pose of the Intel depth camera in world coordinates system.

In this work, the depth camera is manufactured by Intel RealSense with model D435i, which offers both RGB image and depth point cloud data. Intel provides complete development SDK toolbox and various user-friendly post-processing algorithms like to ensure images and depth information with good quality and stability. For example ,spatial Edge-Preserving filter aims at increasing smoothness of images and temporal filter is intended to stable the depth data. This camera is fixed on the head of the robot and receives a top view from above.

Besides two fixed camera system, we also need a 2-D flat calibration chessboard for acquisition both intrinsic and extrinsic matrix for Intel RealSense Camera. Chessboard calibration is a widely used method in computer vision [19]. Many mainstream image processing libraries like OpenCV offer chessboard pattern for print and also convenient functions to calculate related camera matrix.

We define 5 different coordinates system for following processing, namely : world frame, robot frame, camera frame, OptiTrack frame and chessboard frame, as the figure shown below.

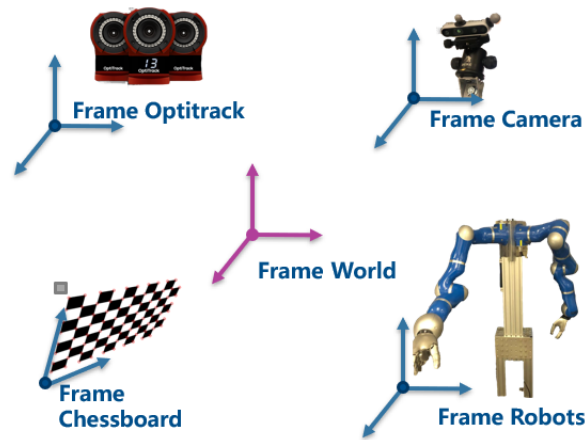


Figure 3.2: Five Coordinate Systems In Camera Calibration

3.1.2 Intrinsic Matrix Acquisition

The traditional camera calibration method, such as methods from Zhang [20] and Tsai [21], are extensively discussed topics in field of computer vision. These two method both require extra calibration marker or pattern. In this work, we use the method from Zhang and printed chessboard pattern with 9-by-7 grids from the OpenCV library.

The basic workflow is shown as follow:

- 1) Print chessboard pattern and attach it to a flat surface, like piece of carton.
- 2) Move the surface and take at least 10 snapshots of chessboard grid from different angles.
- 3) Detect all critical features points, namely 4 corner points of the grid.
- 4) Choose at least 4 pair data of the corner points' pixel coordinates and calculate homography matrix with help from OpenCV functions.
- 5) Take pictures from different angles as input and calculate corresponding scale factor, distortion coefficients and intrinsic matrix of the camera with help of nonlinear algorithm like Levenberg-Marquardt.

Usually in pinhole camera model, the radial distortion and tangential distortion should be taken into consideration. Under frame of OpenCV, there are totally five distortion

parameters for use: k_1, k_2, k_3 for the radial distortion and p_1, p_2 for the tangential distortion. Before the calculation of the intrinsic matrix, we should calculate the corrected coordinates of the image and undistort the snapshots.

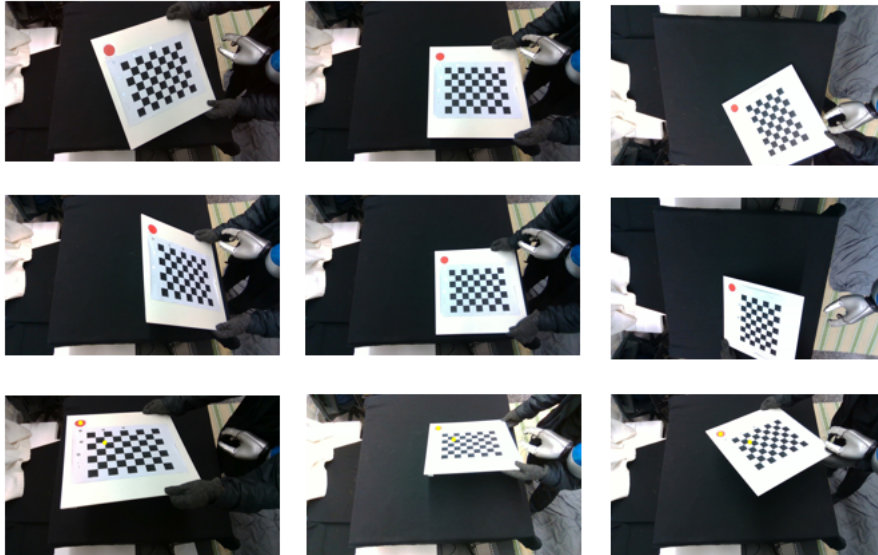


Figure 3.3: Snapshots of the Calibration Pattern From Different Poses

After OpenCV function processed the input pictures of the chessboard, the results are: $f_x = 921.563$, $f_y = 921.750$, $c_x = 642.073$, $c_y = 343.416$. The Intel RealSense SDK offer function to look up default value of the intrinsic parameters: $f_x = 920.788$, $f_y = 920.285$, $c_x = 639.458$, $c_y = 343.739$. Compared to the default value by Intel official documentation, the calculated camera parameters are quit close to them. In order to simply the programming, in following sections the intrinsic parameters would be the same with the value we calculated. In our case, this matrix represents the transform relationship between the camera frame and pixel frame.

3.1.3 Extrinsic Matrix Acquisition

In this work, expected extrinsic matrix defines the transform relationship between the camera frame and world frame. As the figure below illustrated, based on the transform chain the desired the extrinsic matrix could be determined directly. The frame chessboard

pattern nested serves as the bridge between world frame along with OptiTrack and camera.

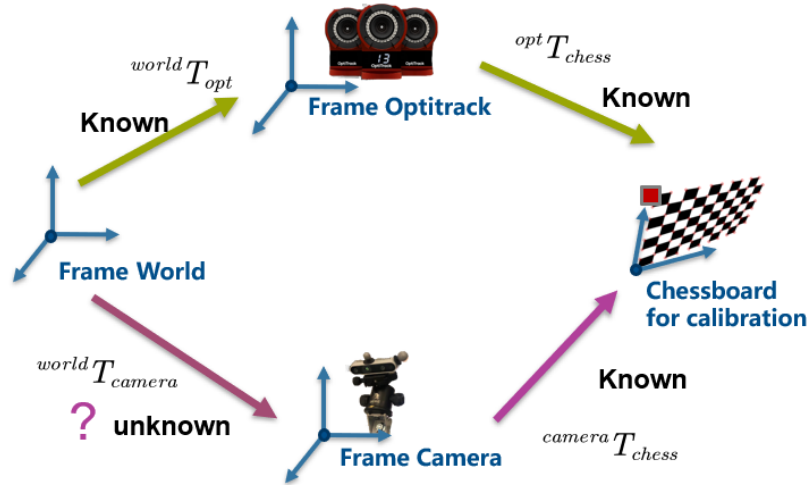


Figure 3.4: Transform Chain Inside Five Coordinate Systems

The transform relationship between these frames is shown as equation below.

$${}^{world}T_{camera} \cdot {}^{camera}T_{chess} = {}^{world}T_{opt} \cdot {}^{opt}T_{chess} \quad (3.1)$$

where ${}^{world}T_{opt}$ is transform matrix between the world and OptiTrack. It is already known, the OptiTrack system is mounted; ${}^{opt}T_{chess}$ known, when the OptiTrack system recognises it via capture markers attached to the surface of the chessboard; ${}^{chess}T_{camera}$ is identical to the intrinsic matrix we get.

The desired matrix ${}^{world}T_{camera}$ could not directly determined by the Equation 3.1. Because the origin point of the chessboard coordinate system in two transform chain could not be the same point on the chessboard plane. The origin point of chessboard in sight of OptiTrack is in the center of three tracking markers, whereas the origin point by camera is always of upper left corner point of the image. When the pose of chessboard plane changes, the selected corner point may be different from the last time. For example when the board rotates 90 degrees clockwise, the original lower left corner point would be picked up.

During the data collection, it is quite hard to ensure these two origin point to be the same one. For the sake of simplicity, as the Figure 3.6 presents, an extra marker for locating

the origin of the chessboard frame. Therefore, there are two chessboard frames in our case: frame based on the center of OptiTrack markers denoted by $chess_1$ and frame based on the OpenCV calibration functions denoted by $chess_2$. Can be observed in Figure 3.6, the transform between these two frames is constant with help from extra circle markers.

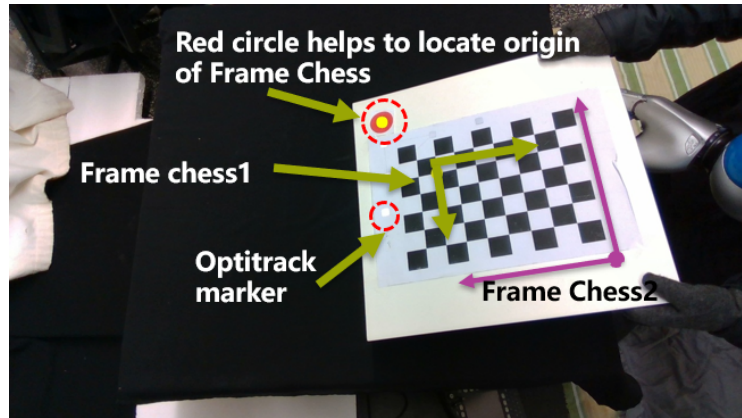


Figure 3.5: Two Coordinate Systems on the Chessboard Plane

As summarized in Chapter 2, this is a typical $AX = XB$ problem, but slightly different from the traditional hand-eye calibration problem. The solution to this $AX = XB$ problem is the same with two step method from Tsai [21], by solving the rotation matrix first and translation vector later.

$$chess1_1 T_{chess2_1} = chess1_1 T_{world} world T_{camera} camera T_{chess2_1} \quad (3.2)$$

$$chess1_2 T_{chess2_2} = chess1_2 T_{world} world T_{camera} camera T_{chess2_2} \quad (3.3)$$

The Equation 3.3 and 3.2 represent two snapshot of the chessboard. We could combine these two equations into together and determine the formulation in form of $AX = XB$ problems, as the equations below describe. The desired transform matrix $world T_{camera}$. Based on the method from Tsai-Lenz [21], the rotation part from the 4-by-4 transform matrix would be solved first with help from Rodrigues parameters. Later on the translation part could easily determined.

$$A = \left({}^{chess1_2}T_{world} \right)^{-1} \cdot {}^{chess1_1}T_{world} \quad (3.4)$$

$$B = {}^{camera}T_{chess2_2} \cdot \left({}^{camera}T_{chess2_1} \right)^{-1} \quad (3.5)$$

$$X = {}^{world}T_{camera} \quad (3.6)$$

In summary, the basic workflow to acquire extrinsic matrix is as the figure below illustrated:

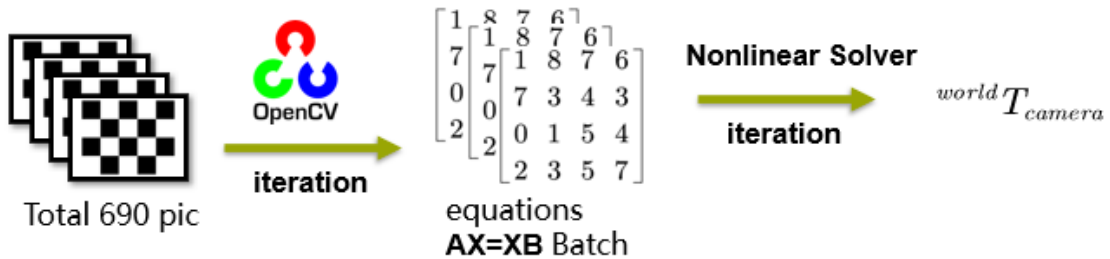


Figure 3.6: Workflow for the Extrinsic Matrix's Acquisition

Since the required two camera matrices is known, the data-stream from camera could be efficiently and correctly converted into correct coordinates in world frame. So far the most important pre-processing is accomplished.

3.2 Obstacles Map

In this section we presents the workflow about the generation of obstacle maps. As detailed in Chapter 2, the ultimate goal of the visual processing is to acquire the obstacle map, which is an octomap presenting all discrete points on the surface of the obstacles. Meanwhile, the points from the robot arm links should also be eliminated from the obstacle map. The basic workflow is illustrated in the figure below.

As in last section discussed, the camera parameters are right now given, any points from point cloud in camera coordinate system could be transferred into world frame. The input from camera could be precisely converted into data-stream in form of point cloud. The Intel RealSense SDK offers demo to launch corresponding ROS node to publish environment's point cloud. In order to achieve objectives above, the following sections will focus on three tasks:

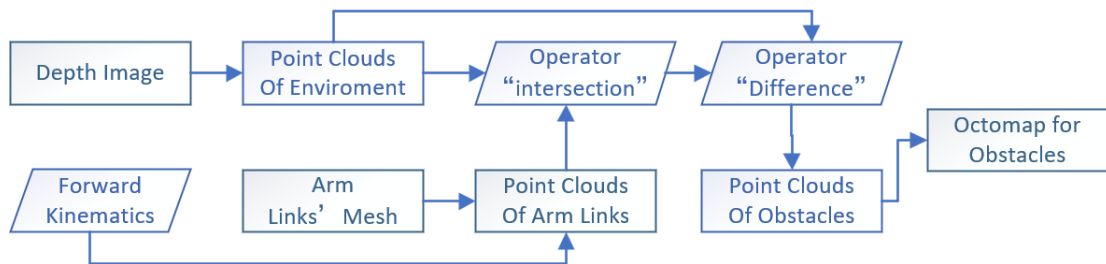


Figure 3.7: Workflow of Generation of Obstacle Map

- 1) Generation of robot links' point clouds.
- 2) Message Synchronization
- 3) Post-processing of Point Clouds

3.2.1 Point Clouds from Robot's Link

The final goal of visual processing in this work is an obstacle map but without points from the robot's links. As Figure 3.8 illustrated, the input is the geometry mesh files of the 14 robot's arm links.

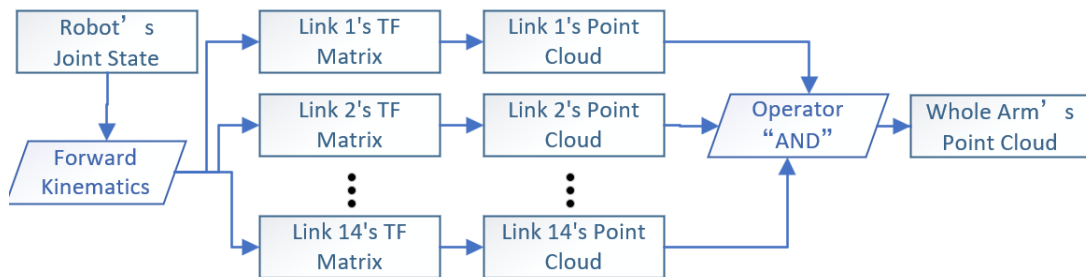


Figure 3.8: Workflow of Generation of the point clouds from the Robots' links

With help from open-sourced software CloudCompare, it could convert the geometry file in STL format into point cloud format. When the CloudCompare receives the STL, we

need to sub-sample the surfaces of the arm links. The density of the desired sample point cloud could be manually set with default value 10000 points. For the sake of efficiency and saving run-time memory, after experiments, this value could be reduced to 1000 and the final point clouds' density still meets our end.

Beside the point clouds of 14 different links the processor still needs transform matrix between the links and world frame. These transform could be directly exported in robot forward kinematics. In this work, the forward kinematics is based on the open-sourced robot kinematics library Pinocchio. This light-weighted library supports c++ and python and provides powerful built-in functions to calculate the robot's states efficiently, such as Jacobian matrix and transformation between different frames.

The robot's controller would persistently publish 14 joints' state via ROS message. The corresponding ROS node subscribes this joint states' publisher and pushes these data into queue array of robot kinematics functions. The Pinocchio library receives thees joints' value and calculate corresponding Jacobian matrices and 14 desired transform matrices between joints' reference frames and world frame.

According the workflow displayed in Figure 3.8, we have already collect desired point cloud of each arm link. Finally we should merge these 14 links' point cloud into united one and eliminate redundant points. The goal could achieved with help from PCL built-in function for point clouds' basic set operations. The Point Cloud Library (PCL) is an open-sourced and powerful library for the two- and three-dimensional point clouds' processing. Apart from the "AND" set operation, this library offers a bunch of useful and user-friendly functions, which would also be used in the following section, such as KD-Tree, filters and octree.

3.2.2 ROS Message Synchronization

This section explains the problem in ROS message synchronization and discusses the concrete steps to solve these issues.

In this work, the Intel RealSense camera publishes the point cloud of environment at 30Hz, whereas the robot states' publisher runes at least 1000Hz. These difference could cause unexpected delay in two point clouds. In our case, only one ROS node runs to process and generate obstacle map. This node subscribes two ROS publisher and pushes these data into queue at first. Later it would follow the rule "last in, first out" to select newest data from the queue and afterwards publishes processed obstacle map. The advantage of only ROS node is obvious, compared two separate ROS node: saving run-time memory and reducing time cost by intercommunication between two ROS nodes.

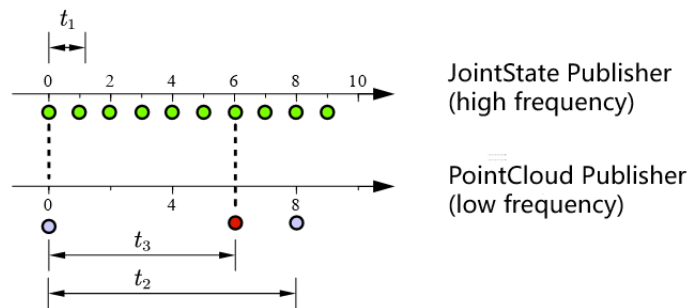


Figure 3.9: Difference by Publisher's Frequency

As the figure above reveals, we assume that two topics run at 1Hz and 8Hz to simplify the illustration. As can be observed in the figure above, the callback functions of two subscribers are triggered at the beginning. The processing time by the joint states' publisher is close to zero, while the point cloud node publisher needs t_2 , around 6 seconds, to process the incoming data. However, the upcoming joint states are with a timestamp about 6 seconds later, when the point cloud publisher finishes the processing. This difference brings about a time difference between the robot arm's point cloud and the point cloud from the depth camera. Hence, the following set operations around these two point clouds would definitely lead to errors.

One of the most intuitive solutions is to move the callback function of the robot states' subscriber inside the callback of the point cloud subscriber. Only when the point cloud's callback is triggered, then the robot states' callback would be triggered.

Another ROS built-in function for message filters could also help to solve this problem. According to the ROS message filters' documentation, we should define two subscribers with `type message_filters::subscriber`, build an extra `TimeSynchronizer` and also define a joint callback function via `boost::bind`.

Two ways described above provide simple solutions to this problem. A perfectly aligned data queue is barely impossible in this case due to the unpredictable size of upcoming point clouds. These two methods could bring satisfying enough results. Once the two point clouds are aligned closely enough, the following post-processing of the point cloud and conversion could also continue.

3.2.3 Post-processing of Point Clouds

This section describes the detailed procedure in post-processing of obstacle point cloud and concrete step by set operations for these point clouds.

As the Fig 3.10 illustrated, before acquiring final obstacle map, there are a bunch of sequential processing parts in the workflow. These processing operations could be divided into 3 type based on their aims: down-sampling function, set operation function and octomap conversion maps.

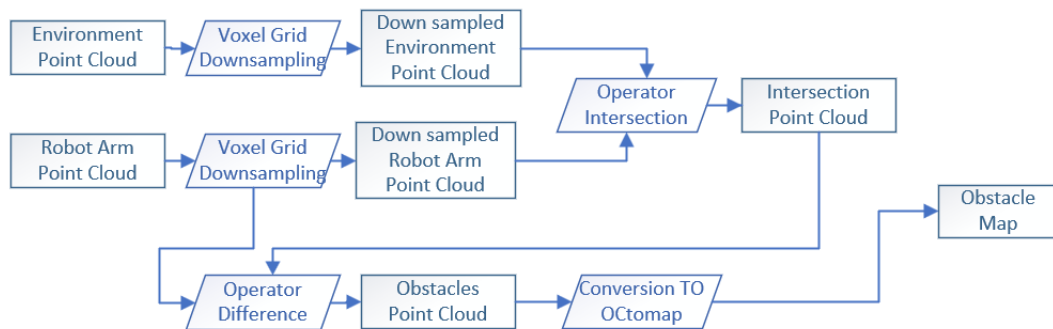


Figure 3.10: Workflow for Post-processing of Point Clouds

Firstly, we introduce the voxel grid down-sampling. Voxel is a type of three-dimensional occupancy grid map with fixed resolution and leaf size. When one voxel grid is occupied, namely inside this voxel there is at least one point, the corresponding value would be true. With given 0.02 leaf size, the down-sampled point cloud could be compressed quite a lot. Thanks to voxel grid down-sampling, the original point cloud size is reduced from 50,000 points to around 1800 grids. This modification contributes greatly to accelerating the post-processing of these point clouds.

As the Fig 3.11 illustrated, before the "AND" operation for two point clouds, there are still a few critical steps to take: transformation to world frame and pass-through filter for the environment's point cloud. Another huge advantage of placing voxel grid process before applying the transformation is to save great deal of time in processing. Based on the experiments we made, the processing time with voxel grid operation is about 8 ms, whereas around 181ms without this step. We obtain quite satisfying result through this simple pre-processing.

Before from transformation, the environment's point cloud requires an extra pass-through

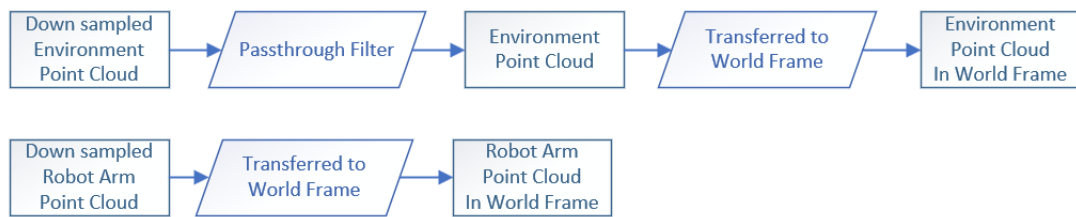


Figure 3.11: Detailed Processing Steps for Two Point Clouds

filtering. Because in the testing scenes, all the target objects and obstacles are above the table's surface. Meanwhile the depth camera collects data in top view and points from the ground would also be saved into data-stream. In order to eliminate these useless points, the pass-through filter would delete all points with value in z axis less than 0.5meter. The output is illustrated as the figure below.

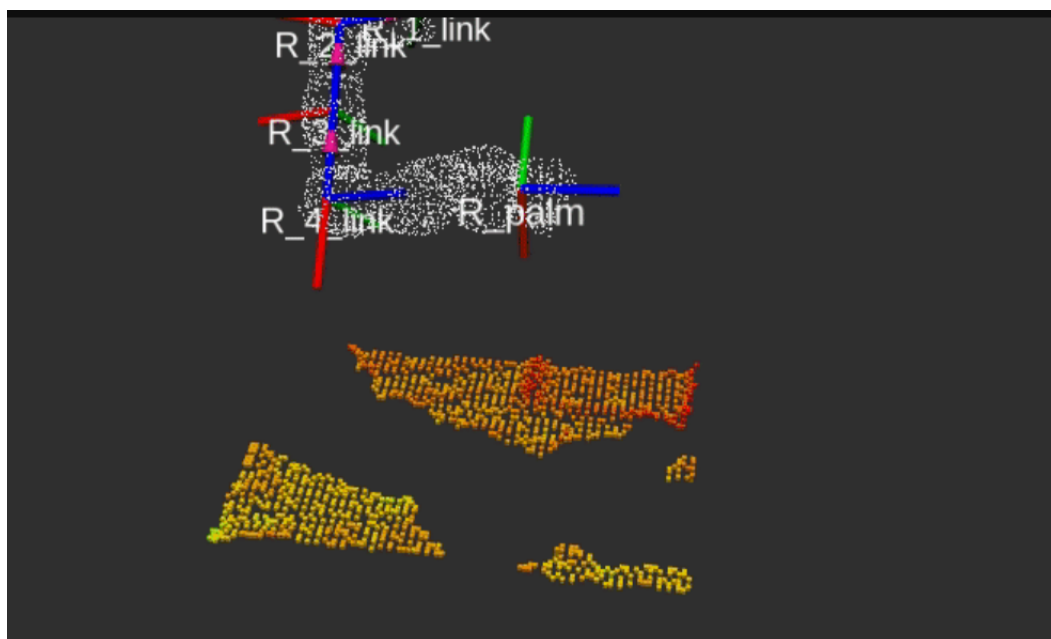


Figure 3.12: Processed Two Point Cloud from Environment and Robot Links

The following part is the key step in the whole post-processing of point clouds. This step contain two parts:

- 1) Get Intersection Point Cloud of Environment's and Robot Link's Point Cloud.
- 2) Subtract the Intersected Point Cloud from the Environment's Point Cloud.

To achieve the first task, the general idea is iterate all the points in the robot link's point cloud and later compare them to the enforcement's point cloud to extract overlapped points. In our case, the down-sampled point cloud of 14 robot links contains more than 14,000 points. The time cost to iterate all these points is unacceptable, especially in our time-sensitive application.

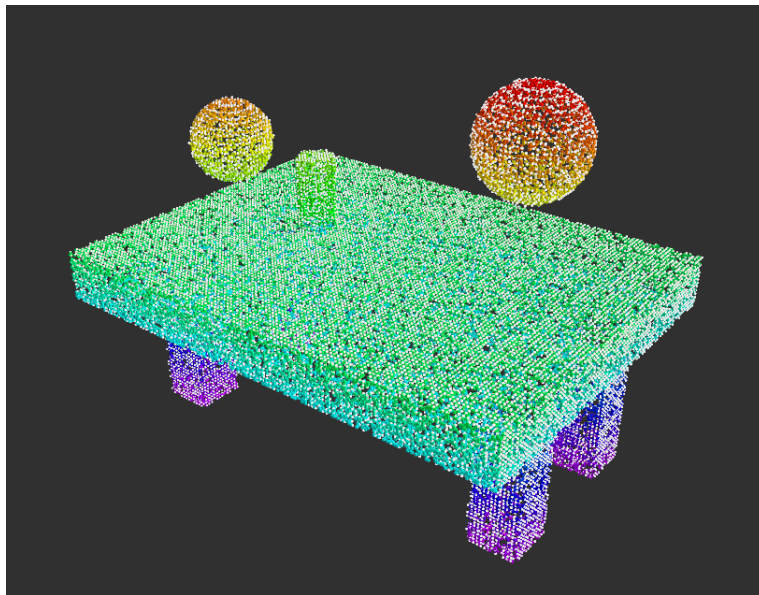


Figure 3.13: Final Obstacle Octomap

Alternatively, we choose to use KD-Tree with radiusSearch function to find overlapped points in this two point clouds. Firstly, we define the criteria to determine whether two points are overlapped. IN general, if euclidean distance between two points is smaller than a given threshold, these two points would be treated as overlapped points. The comparison process could be achieved by nearest neighbour search. The complete steps is as displayed in the algorithm below.

Input: point cloud from environment P_{env} with size n_{env}
point cloud from robot arm P_{arm} with size n_{arm}

Output: final point cloud P_{final}

Initialize KD-tree K_{tree} from P_{arm} ;

Initialize empty point cloud P_{common} fro intersection part ;

for $i = 1$ to n_{arm} **do**

- $p_{search} \leftarrow P_{arm}[i]$;
- use K_{tree} to do the nearest neighbour search for p_{search} ;
- search result saved as $p_{overlap}$;
- if** $p_{overlap}$ **not null then**

 - push $p_{overlap}$ into P_{common} ;
 - delete p_{search} from P_{arm} ;

save the size of P_{common} as n_{common} ;

for $i = 1$ to n_{common} **do**

- $p_{search} \leftarrow P_{common}[i]$;
- delete p_{search} from point cloud P_{env} ;

reshape the P_{env} and save as required final point cloud P_{final}

Algorithm 1: Set Operations by Two Point Clouds

After acquisition of the common part by two point clouds, we should delete points in intersection group from the environment's point cloud. When every point from the robot arm link is removed, the ultimate goal of this section is basically achieved. The rest conversion task could be directly accomplished by PCL functions, which takes point cloud as input and returns octomap with given leaf size. For example, assuming that there is no obstacles block infra ray from the depth camera, the final obstacles map in form of octomap should looks like the Figure 3.13.

4 Leaves for motion generation

This section begins with the brief introduction of task, which the CEP algorithm should achieve under this paper. As explained in detail in Chapter [?], CEP could combine few independent motion policies into one single policy via the product of these expert policies to solve multi-task problems. In stead of extremely sophisticated task with fingers, such as grasping and releasing, this work focus more on the interaction with the human-controlled joystick and the importance of these policies would change with the distance between the end effector and the target position.

In our case, there three different policies in function:

- 1) Task Go-To when the end-effector is far away from the target.
- 2) Task Joystick
- 3) Task Go-To when the end-effector is far close to the target.

For simplification the first task is denoted as task Go-To Far, where as the third task as task Go-To Close. The rest task for joystick's control would be denoted as task Joystick. In this paper, the ultimate goal for the CEP application is fulfilling three requirements described as followed:

- 1) When the end effector is far away from the target, the Task Joystick control completely the motion of robot arm.
- 2) When the end effector is not quite far from the target, the Task Joystick and Task Go-To Far control the motion of one robot arm together.
- 3) When the end effector is close enough to the target, the Task Go-To dominates the control of robot arm.

The weight coefficients of these three tasks are functions of distance between the target and end-effector. Key of this CEP application is to design these three distance functions appropriately.

4.1 Leaf for Task Go-To Far

As the last section explained, CEP allows deeper structure in latent space. Therefore the mapping between different action-state space has a tree-shaped structure, just like the figure below illustrated. The root of this mapping tree is the common space, namely configuration space for all joints.

For the task Go-To Far, the action-state pair $(\mathbf{a}^{z2}, \mathbf{s}^{z2})$ is defined in task space for the end effector. In order to explain the mapping relationship clearly, we start with the root of the tree: configuration space. In our case, the control of two robot arm is running in two different progress. Because the cooperation between two arms are quite complicated and hard to define corresponding mapping functions. Through the robot kinematics, we could transform the configure space $(\mathbf{a}^z, \mathbf{s}^z)$ into the task space $(\mathbf{a}^{z1}, \mathbf{s}^{z1})$. Once the controller received the joint state from the broadcaster, the controller could compute the current pose of each joint in world frame. The sequential mapping is one of the pick up map. In order to control the behavior of the end effector, we only need the last row of data from the task space $(\mathbf{a}^{z1}, \mathbf{s}^{z1})$.

4.2 Leaf for Task Joystick

In this work, we use joystick for the manipulation of two robot arms via a joystick from Xbox.

Task Joystick takes charge of the motion of end effector in x, y axis, when the end effector is not so closed to the target. Adding joystick related policy into the CEP system is aiming for more freedom for the user. For example, the user could manually set an appropriate path for the robot at the beginning. Lately the robot might learn from the previous path set by the user. It contributes to accelerate the learning progress, especially dealing with sophisticated environment. Without the joystick, the robot might need much more time

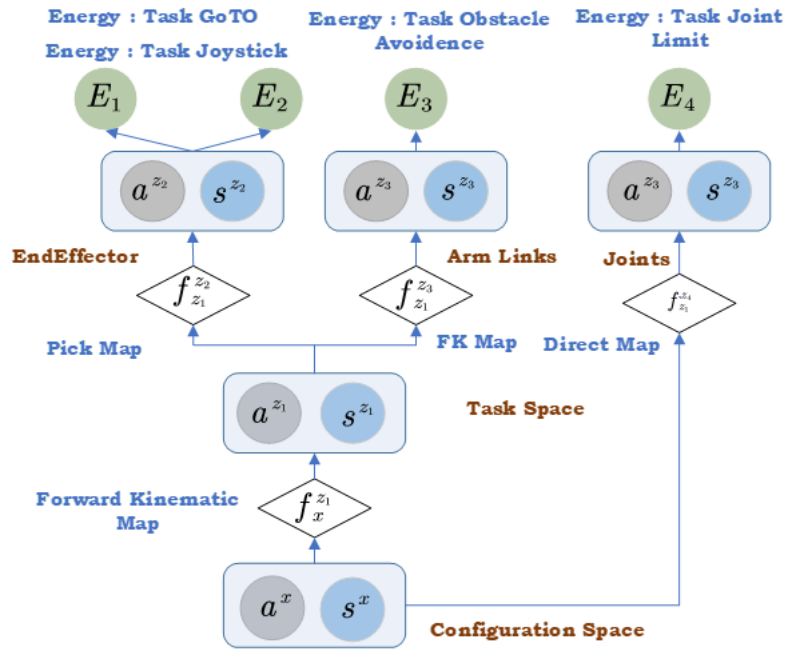


Figure 4.1: Mapping Tree from the Configuration Space

for the policy searching and optimization of the actions. By contrast, the guide from the user could be easily passed to the robot motion planner. The mapping relationship for the Task Joystick is the same as the Task Go-To Far, both two policy interests only in behavior of end effector.

4.3 Leaf for Task Go-To Close

Task Go-To Close is designed for the task with demand for high accuracy, such as complicated motion primitives grasping and releasing. In this work Task Go-To only focuses on the accurate motion of the robot' hand. Especially when the hand is quite close to the target, the controller requires smooth, safe and quick response to the environment to avoid unexpected errors like hitting on the target. By contrasts, Task Go-To Far and Task Joystick are designed for the rough motion control, whereas Task Go-To Close for the

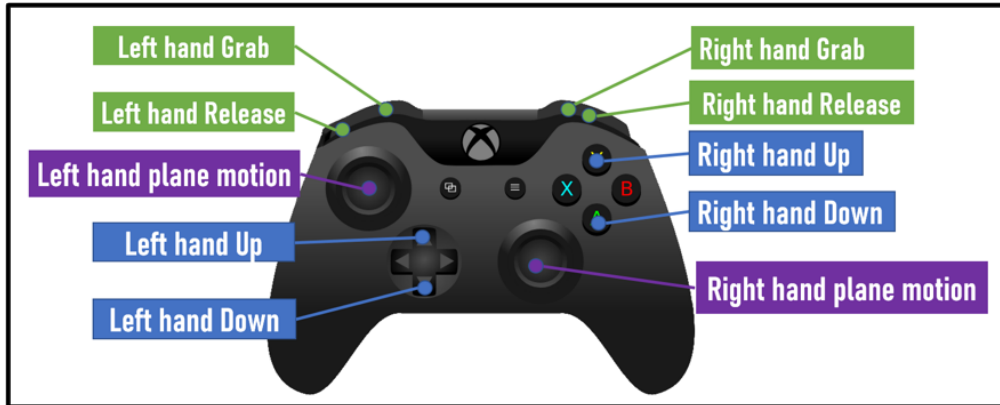


Figure 4.2: The Projection of Buttons with Different Functions

delicate and sophisticated tasks.

4.4 Key parameters for Three Leaves

This section introduces the four main key parameters for these leaves: distance threshold, the scale factor for the joystick's input, the PD speed controller for the Task Go-To Far and Close and the variable weight coefficient for these three leaves.

Joystick Input as Speed

The action for this leaf is the speed of the end effector $\dot{q}_x = \alpha x, \dot{q}_y = \alpha y$, where the x, y is the value from stick ranging from -1 to 1, α is corresponding scale factor. Considering the safety of the operation, this value could be set to 0.01 to ensure a smooth and not so quick motion of the end effector.

Task Go-To Speed Controller

As explained in Chapter 2, according to Lie algebra, with two given 6D pose (start pose and target pose), we could compute the desired 6D speed spinor (translation and rotation speed) via log operation from $SE(3)$ to the space $\mathfrak{se}(3)$.

$$x_{des} = (1 - \beta)\dot{x} + \beta\rho \quad (4.1)$$

where ρ is the linear speed part of the ξ , β stands for the damping coefficient, x_{des} presents the desired linear speed in task space and \dot{x} is the speed in last timestamp.

Due to the ignorance of angular speed of the ξ during the Lie algebra calculation, the speed of the end effector may be not smooth enough. The damping factor β is applied here to smooth the speed of end effector. This value could be set to 0.3 after experiments.

Variable Weight Coefficient

As mentioned in the beginning of this chapter, the weight of each leaf is a function of distance between between the end effector and target. Here this distance will be denoted by d_{ee} for the sake of simplicity. For example, when d_{ee} is bigger than 80 cm, the weight of Task Go-To Far will be set to zero. As for the middle and small range, the weight of

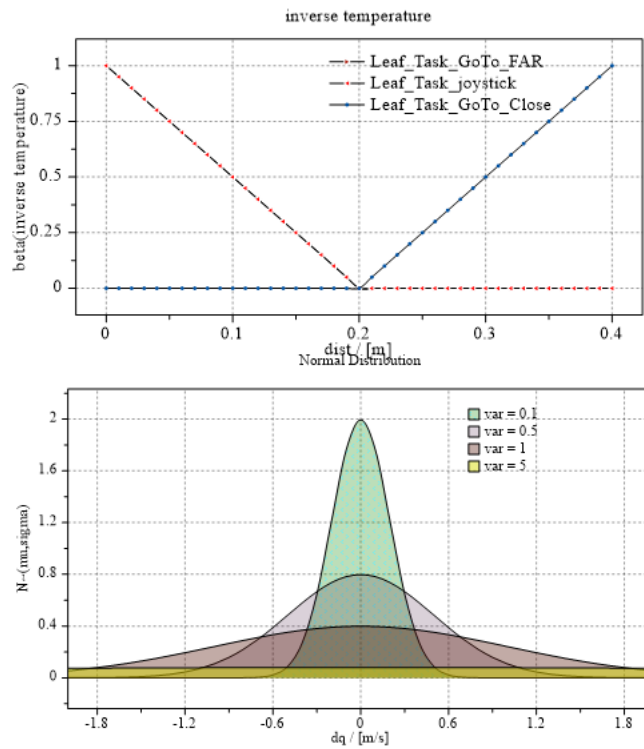



Figure 4.3: Mapping Tree from the Configuration Space

leaves would be function to this distance d_{ee} . There are two methods for this issue as the figure above shown: change the variance in the Gaussian distribution or set it to a simple



slope function. We choose the second one as the solution due to its simplicity and reduced the number of variables in optimization progress.

5 Experiments

This section begins with the brief introduction of set up of the experiments. Later three different motion task primitives will be discussed in detail and corresponding the combination of these tasks will also be introduced. Another topic about obstacle avoidance will be analyzed in followed section.

5.1 Experiment Setup

Due to complexity of real robot's control, the experiments will all be conducted in the simulation environment. Because the algorithm in this thesis contains much uncertainty due to the policy search strategy and random initialization of action distribution. This uncertainty might caused unexpected errors or even failures causing harm to human and surrounded equipment. Before long-term validation for the robustness and safety any algorithms should be conducted in simulation.

The simulation environments for the robot dynamics is the PyBullet, which provides convenient interface for the robot control and building of the test scenes. Furthermore PyBullet offers also user-friendly GUI interface and give clear display about critical information, such as joints' state or speed of the end effector.

As for the simulation for the obstacle avoidance, some environment data would be pre-collected via depth camera. These image stream will be replayed during the simulation as the off-line data. In our case, all the required data along with necessary transform tree would be recorded via ROSBag function and lately replayed by the ROSBag node during simulation. For some a bit more complicated scenes with multi obstacles, we will use tools like PyBullt for quick start-up to build the scene. For example in our work, the single ball scene is collected from the real world, whereas the a wall of obstacles are built via software.

5.2 Tests for the CEP control

In this work, the test for the CEP control focuses on if the robot whether act as designed when the distance between the end effector and target pose changes. As described in Chapter 4, when the distance d_{ee} is larger enough, in our case this value was set to 80cm, the action computed from Task Go-To Far policy would no impact on the final action of the robot's arm. When the robot's hand is close enough to the target, even though there

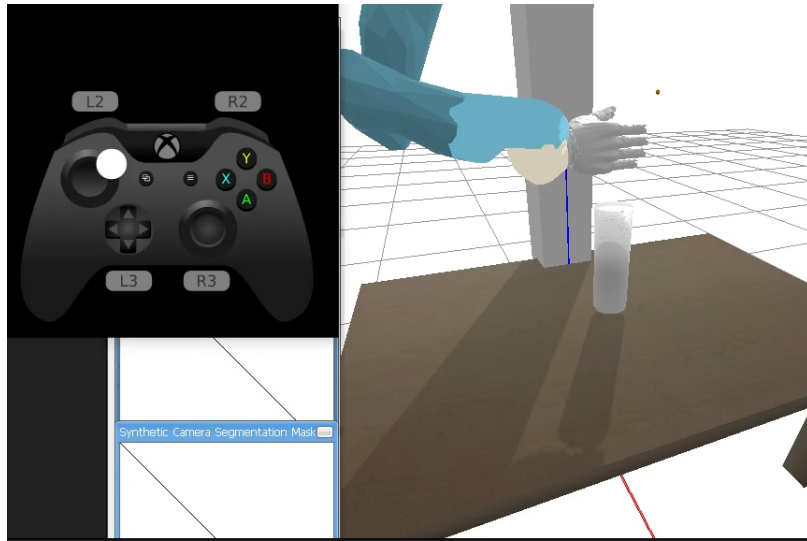


Figure 5.1: Task Go-To Close With large Weight Coefficient

are input from the joystick, the end effector only followed the instruction of Task Go-To Close. The robot's hand moved smoothly to the target with decreasing velocity. When the distance d_{ee} is in the middle range, the Task Go-To Far drove the end effector slowly in the z axis. Meanwhile the input from joystick's stick button still functions.

As the Fig. 5.2 illustrated, when the distance d_{ee} is small, the corresponding leaf rank highest among three policies. CEP optimizes the motion planning problem as a sum of bunch of Gaussian distribution here. The action with high probability is more likely to fulfill all the assigned task in parallel, whereas lower probability may fail few tasks or even cause unexpected errors.

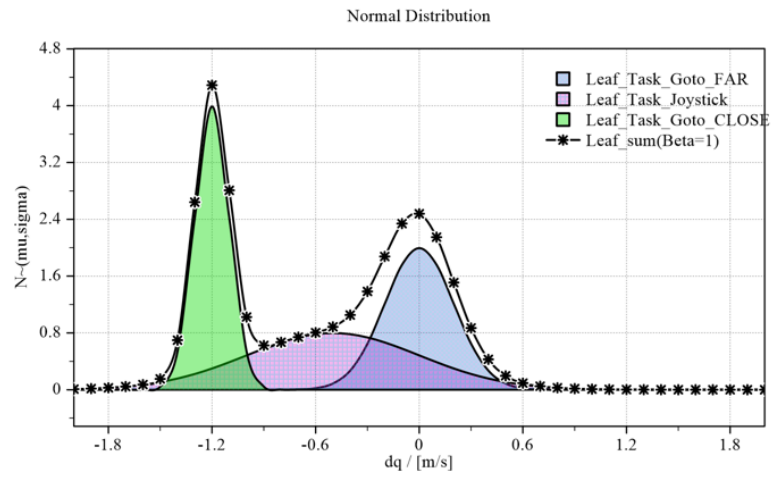


Figure 5.2: Sum of Three Leaves

5.3 Tests for obstacle avoidance

For the obstacle avoidance alone. for both single obstacles and multi obstacle avoidance, the artificial potential filed method works well.

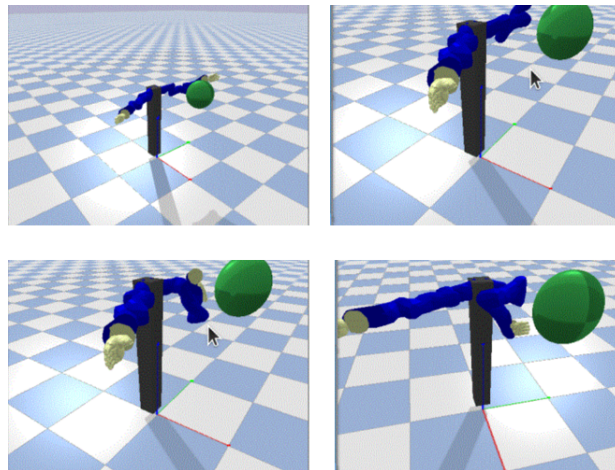


Figure 5.3: The progress of Obstacle Avoidance With Single Sphere

As the Fig. 5.3 shows, the sphere-shaped obstacles is placed at the middle between the start and goal. The robot arm moved around the obstacle and kept as closed as possible to the original straight line trajectory.

As the Fig. 5.4 illustrated, for multi obstacles, the artificial potential field method also

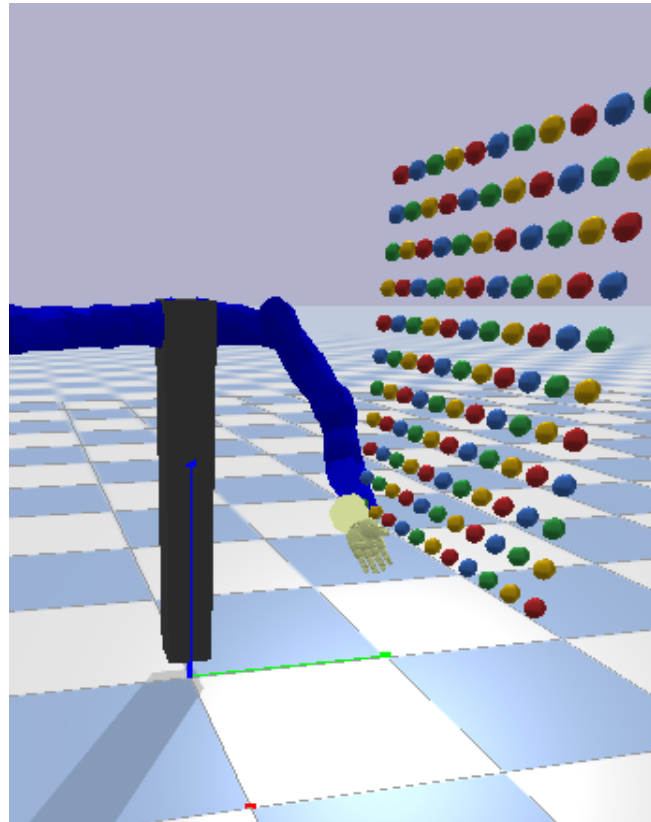


Figure 5.4: Sum of Three Leaves

functions well. APF methods yields satisfying results while dealing with large numbers of obstacles. Because the calculation of the distance could be easily vectorized and lots of matrix calculation library has official optimization for these operations.

6 Discussion

In this thesis, we achieve the application of CEP with three policies Task Go-to Far Task Joystick and Task Go-To Close. The controller could based on the current state of the end effector and the distance to the target change its own behaviors. When the distance is far enough, the user have the freedom to control partial motion of the robot arms. When the distance is close enough, corresponding algorithms will take over the control and interact with the environment with much higher frequency than human.

Apart from the application of CEP, we also designed and achieved complete processing of converting the depth image data into obstacles map in form of octree. We also found appropriate method to eliminate point cloud from robot's links. Moreover, in this thesis we also increase the computation efficiency via various post-processing techniques for point clouds, such as simplification via voxel grid and pass-through filters. In order to speed up the comparison between two point cloud, we applied KD-Tree for it.

As for the obstacle avoidance, we combine the artificial potential filed method with the obstacle maps. We could generate the potential filed based on the grid in the obstacle maps without extra object segmentation procedures.

7 Outlook

As for the visual processing part, more modification on the point clouds could be made to accelerate the computation. For example, like object-based segmentation. When certain range of points clouds could be recognized as a object, we could use simple geometry such as cylinder, box and sphere as approximation of this object. In this way the burden by processing the point cloud could be reduced, such as the set operation, searching and comparison. Further more, the obstacle map could also be updated with large resolution. Besides the depth information, image processing technique could also contributes to the object segmentation. As for the CEP policy, it is possible to combine two arm into one progress, namely from 7 DoF planning problem converted into one 14 DoF planning problem. This integration could help to solve the conflicts of these two arms. Instead of using obstacle avoidance techniques, these problem can be directly solved from the perspective of algorithms. Another advantage of this integration is saving of communication cost between different progress.

Bibliography

- [1] 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.
- [2] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, “Path planning with modified a star algorithm for a mobile robot,” vol. 96, pp. 59–69, Elsevier, 2014.
- [3] Y. Hu and S. X. Yang, “A knowledge based genetic algorithm for path planning of a mobile robot,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*, vol. 5, pp. 4350–4355, IEEE, 2004.
- [4] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, “An improved ant colony algorithm for robot path planning,” vol. 21, pp. 5829–5839, Springer, 2017.
- [5] N. Buniyamin, W. W. Ngah, N. Sariff, and Z. Mohamad, “A simple local path planning algorithm for autonomous mobile robots,” vol. 5, pp. 151–159, 2011.
- [6] P. Raja and S. Pugazhenthii, “Optimal path planning of mobile robots: A review,” vol. 7, pp. 1314–1320, Academic Journals, 2012.
- [7] 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.
- [8] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, “Motion planning with sequential convex optimization and convex collision checking,” vol. 33, pp. 1251–1270, SAGE Publications Sage UK: London, England, 2014.

-
-
- [9] Z.-z. Yu, J.-h. Yan, J. Zhao, Z.-F. Chen, and Y.-h. Zhu, "Mobile robot path planning based on improved artificial potential field method," *Harbin Gongye Daxue Xuebao (Journal of Harbin Institute of Technology)*, vol. 43, no. 1, pp. 50–55, 2011.
- [10] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, vol. 1, pp. 256–263, IEEE, 2000.
- [11] Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su, "Uav path planning using artificial potential field method updated by optimal control theory," *International Journal of Systems Science*, vol. 47, no. 6, pp. 1407–1420, 2016.
- [12] J. Urain, A. Li, P. Liu, C. D'Eramo, and J. Peters, "Composable energy policies for reactive motion generation and reinforcement learning," 2021.
- [13] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, no. 0, 2006.
- [14] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and computing in applied probability*, vol. 1, no. 2, pp. 127–190, 1999.
- [15] F. C. Park, J. E. Bobrow, and S. R. Ploen, "A lie group formulation of robot dynamics," *The International journal of robotics research*, vol. 14, no. 6, pp. 609–618, 1995.
- [16] Y. Liu, Z. Wang, K. Han, Z. Shou, P. Tiwari, and J. Hansen, "Sensor fusion of camera and cloud digital twin information for intelligent vehicles," 07 2020.
- [17] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.
- [18] G. Nagymáté and R. M. Kiss, "Application of optitrack motion capture systems in human movement analysis: A systematic literature review," vol. 5, pp. 1–9, 2018.
- [19] A. De la Escalera and J. M. Armingol, "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," vol. 10, pp. 2027–2044, Molecular Diversity Preservation International, 2010.
- [20] Z. Zhang, "A flexible new technique for camera calibration," vol. 22, pp. 1330–1334, 2000.

-
- [21] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," vol. 3, pp. 323–344, IEEE, 1987.