

Learning Geometric Constraints for Safe Robot Interactions

Lernen geometrischer Beschränkungen für sichere Roboterinteraktion

Master thesis in the field of study "Computational Engineering" by Kuo Zhang

Date of submission: July 13, 2022

1. Review: M.Sc Puze Liu
2. Review: Ph.D Davide Tateo
3. Review: Prof. Ph.D Georgia Chalvatzaki
Darmstadt



TECHNISCHE
UNIVERSITÄT
DARMSTADT



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

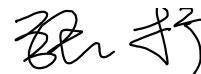
Hiermit versichere ich, Kuo Zhang, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner 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, 13. Juli 2022



K. Zhang

Abstract

The safety of robot is becoming increasingly important, especially in human-robot interaction workspace. The issue of robot collisions is also a hot topic in the field of robot safety. Many methods are now available for collision avoidance control through reactive controller. For Reinforcement Learning (RL), some approaches have achieved collision avoidance by establishing geometric constraints during exploration. However, there is no method that can accurately represent the distance field or geometric constraints for both static and dynamic objects throughout the workspace. In this work, we propose a novel type of Signed Distance Field (SDF): Regularized Deep Signed Distance Field (ReDSDF), which is a single neural implicit function and can compute smooth distance field and define dynamic geometric constraints at any scale.

Nowadays, there are also many methods that compute distance fields by using neural networks, but these methods can only estimate the area near the object and fail for long distance. Moreover, there is no method to calculate the distance field for dynamical articulated objects. Therefore, the author proposes the approach of data generation, network architecture and training methods to remedy these shortcomings.

The ReDSDF can be applied not only as repulsive potential function in reactive motion generation, but also as constraints in RL. In the reactive controller, the ReDSDF of each object can be directly used as an Artificial Potential Field (APF) to control the robot to avoid collisions. And the gradient of the APF can be calculated by deriving the neural network. In RL, ReDSDF can be applied as implicit equations to define the constraints through determining the safe distance between the robot and the object. The application of ReDSDF in robotics is not limited to this. It can be employed by any constrained planner, both as a differentiable constraint and as an energy function. To sum up, ReDSDF can achieve great efficacy in the field of robot safety.

Zusammenfassung

Die Sicherheit von Robotern wird immer wichtiger, vor allem im Bereich der Mensch-Roboter-Interaktion am Arbeitsplatz. Das Problem der Roboterkollisionen ist auch ein heißes Thema im Bereich der Robotersicherheit. Es gibt inzwischen viele Methoden zur Kollisionsvermeidung durch reaktive Regelung. Beim Reinforcement Learning (RL) erreichen viele Methoden die Kollisionsvermeidung durch die Festlegung geometrischer Nebenbedingungen während der Exploration. Es gibt jedoch keine Methode, die das Abstandsfeld oder geometrische Beschränkungen für beide statische und dynamische Objekte im gesamten Arbeitsraum genau darstellen kann. In dieser Arbeit schlagen wir eine neue Art von Signed Distance Field (SDF) vor: Regularized Deep Signed Distance Field (ReDSDF), das eine einzige neuronale implizite Funktion ist, verwendet ein neuronales Netzwerk, um den Abstand zur Oberfläche eines Objekts zu schätzen.

Heutzutage gibt es auch viele Methoden, die neuronale Netze zur Berechnung von Abstandsfeldern verwenden, aber diese Methoden können nur den Bereich in der Nähe des Objekts schätzen und versagen bei großen Entfernungen. Außerdem gibt es keine Methode zur Berechnung des Entfernungsfeldes für dynamische, gelenkige Objekte. Daher schlagen wir einen eigenen Ansatz zur Datengenerierung, eine Netzwerkarchitektur und eine Trainingsmethode vor, um diese Mängel zu beheben.

Die ReDSDF kann nicht nur als repulsive Potentialfunktion in der reaktiven Bewegungsgenerierung, sondern auch als Beschränkung im RL eingesetzt werden. In der reaktiven Steuerung kann die ReDSDF jedes Objekts direkt als künstliches Potentialfeld (APF) verwendet werden, um den Roboter zur Vermeidung von Kollisionen zu steuern. Und der Gradient des APF kann durch die Ableitung des neuronalen Netzes berechnet werden. In RL können wir den sicheren Abstand zwischen dem Roboter und dem Objekt bestimmen, indem wir ReDSDF als implizite Gleichungen zur Definition der Nebenbedingungen verwenden. Die Anwendung von ReDSDF in der Robotik ist nicht darauf beschränkt. Sie kann von jedem Planer mit Nebenbedingungen verwendet werden, sowohl als differenzierbare Nebenbedingung als auch als Energiefunktion.

Contents

1. Introduction	2
2. Related Work	5
2.1. Safe Human-Robot Interaction	5
2.2. 3D Human Body Reconstruction	6
2.3. Reactive Motion Generation	7
2.4. Distance Fields and Manifold	8
2.5. Safe Robot Reinforcement Learning	9
3. Preliminaries	11
3.1. Implicit Function and Signed Distance Field	11
3.2. Artificial Potential Field	12
3.3. Fundamentals of Reinforcement Learning	14
3.4. Acting on the Tangent Space of the Constraint Manifold	15
4. Regularized Deep Signed Distance Fields	17
4.1. Generation of Dataset	17
4.1.1. Sampling Points and Estimating Normal Direction	18
4.1.2. Filtering out Outliers and Data Rejection	19
4.1.3. Generating Augmented Data and Down Sampling	20
4.1.4. Data Generation for Dynamic Articulated Objects	21
4.2. Architecture of the Network	23
4.3. Training of the Network	26
5. Application of Geometric Constrains	28
5.1. Robot Motion Control with ReDSDF	28
5.1.1. Construction of Artificial Potential Fields	28
5.1.2. Applying Distance Field and APF on Robot	30

5.2. Safe Exploration with Learned Constraints	31
5.2.1. Pipeline of Safe Reinforcement Learning	32
5.2.2. Applying Constraints on Robots to Avoid Collisions	33
5.2.3. Define Actions on the Constraints	35
6. Experiments and Results	37
6.1. Results of ReDSDF	37
6.1.1. Results of Human Model	37
6.1.2. Results of TIAGo with Single Arm	40
6.1.3. Results of Static Models	41
6.2. Whole Body Control	45
6.3. Reaching Target Point with a Shelf	47
6.4. Human Robot Interaction	49
6.4.1. Reactive Motion Generation in Shared Human-Robot Workspace . .	50
6.4.2. Reinforcement Learning in Human Robot Interaction Environment	51
7. Conclusion	55
7.1. Summary	55
7.2. Future Work	56
A. Appendix	65
A.1. Recording of Training Process of ReDSDF	65
A.2. Hyperparameters for RL	67

Preface

List of Figures

3.1. An Example of SDF that Represents a Ball	12
3.2. An Example of APF Visualization	13
3.3. The Framework of RL	15
4.1. Pipeline of Generating the Datasets	17
4.2. Demonstration of Sampling Points	18
4.3. The Process of Removing Outliers	19
4.4. The augmented points are generated along the normal direction. The red arrow is the normal direction, the blue point is the original point. When the nearest original point to the augmented point is not the original point that generated it, the point will be rejected. The orange sphere is the accepted augmented point, and the orange fork is the rejected augmented point . .	20
4.5. Demonstration of Weight Assignment	21
4.6. Articulated Object with Different Poses	21
4.7. Example of Data Augmentation of Human with Different Levels: 0, 0.05, 0.12 and 0.25	22
4.8. Architecture of ReDSDF	24

4.9. The parameters of Sigmoid function defines the transition area between close area and far region. The distance field in close area is determined by neural network, and the distance field in far region is determined by the distance to the center of object.	25
4.10. Three Examples of ReDSDF with Different Parameters of Sigmoid Function.	26
5.1. Explanation of the Repulsive Velocity	29
5.2. PoI of Robot and its Distance Field with Mirroring	30
5.3. Sphere-Based Method	31
5.4. Pipeline of SafeRL	32
5.5. Two Methods to Define the Constraints on Robot	33
5.6. Using Spheres to Define Constraints	34
6.1. Visualization of Human Model	39
6.2. Human Models with Different Levels	40
6.3. Visualization of TIAGo Model	41
6.4. Comparison of the Static Models with Baselines	42
6.5. Comparison of the Table Distance Functions in Detail	43
6.6. Comparison of the Sofa Distance Functions in Detail	44
6.7. Comparison of the Human Distance Functions in Detail	45
6.8. Experiment Screenshot of Whole Body Control	47
6.9. Demonstration of Robot-Shelf Experiment	48
6.10. Shared Human-Robot Workspace	50
6.11. Scene of Human-Robot Interaction	52
A.1. Loss Curve during Training	66
A.2. Results of RL-Experiment with Shelf	69
A.3. Results of RL-Experiment in Human-Robot Interaction	70

List of Tables

4.1. Contents of the Final Dataset	23
6.1. Quantitative Results of Human Model	38
6.2. Quantitative Results of TIAGo++	40
6.3. Quantitative Comparison of the Results of Static Models	42
6.4. Results of Whole Body Control Experiment	46
6.5. Results of RL-Experiment with Shelf	49
6.6. Results of Shared Workspace Experiments	51
6.7. Results of RL-Experiment in Human-Robot Interaction	53
A.1. Hyperparameters for Training ReDSDF	65
A.2. Hyperparameters for RL	68

Abbreviations

Notation	Description
RL	Reinforcement Learning
APF	Artificial Potential Field
SafeExp	Safe Exploration
VAE	Variational Autoencoder
SMPL	Skinned Multi-Person Linear Model
GA	Graph Aggregation
SDF	Signed Distance Field
TSDF	Truncated Signed Distance Function
ECoMaNN	Equality Constraint Manifold Neural Network
CMDP	Constrained Markov Decision Process
MDP	Markov Decision Process
CPO	Constrained Policy Optimization
ATACOM	Acting on the Tangent Space of the Constraint Manifold
ReDSDF	Regularized Deep Signed Distance Field
SafeRL	Safe Reinforcement Learning
RMP	Riemannian Motion Policies
CEP	Composable Energy Policies
PoI	Points of Interest
SAC	Soft Actor-Critic Algorithms

Symbols

Notation	Description
$\mathbf{x} = [x, y, z]^\top$	Vector of three dimension coordinates
d	Distance to the surface of object
γ	Discount factor for RL or a hyperparameter to balance regularisation term during learning ReDSDF
τ	Trajectory of robot
T	Time horizon
\mathbf{s}	Observed states of MDP
\mathbf{a}	Actions on MDP
$r(\mathbf{s}, \mathbf{a})$	Reward of the state \mathbf{s} with action \mathbf{a}
\mathbf{q}	Direct controllable state
$\epsilon(\boldsymbol{\mu})$	Function of slack variable
$\boldsymbol{\mu}$	Vector of slack variable
β	Hyperparameter of slack variable
\mathbf{c}	Implicit function of constraints
\mathbf{J}	Jacobian matrix
\mathbf{N}	Tangent-space
K_c	Error correction factor
$\boldsymbol{\alpha}$	Original actions drawn by the policy network
\mathbf{n}	Normal direction of object surface
ω	Weight for training SDF
\mathbf{q}_o	Pose of object or pose input of SDF
\mathbf{p}	Center position of object
\tilde{d}	Output of SDF

θ	Learnable parameters of network
r	Radius of bounding sphere of object
\tilde{n}	Gradient of SDF
\bar{v}	Repulsive velocity coefficient
κ	Threshold to activate repulsive velocity
v	Joint velocity or linear velocity
δ	Safe distance

1. Introduction

Benefiting from big data, powerful computing, new algorithmic techniques, mature software packages and architectures, and strong financial support, Reinforcement Learning (RL) has grown rapidly and is widely used in many fields [1], especially in robotics. Thanks to application of RL, robots can also perform increasingly complex tasks. But at the same time, the safety of robots has always been an issue that cannot be ignored. Robots may collide with surrounding objects during their work, and some may even collide with themselves. If a collision occurs it can cause damage to the robot and corresponding economic losses. In many occasions, robot and human work together in the same workspace with complementary advantages, which raises higher requirements for the safety of the robot, because any collision can cause significant damage and even personal safety issues. What's more, applying RL in the real world remains a challenging task [2]. Most of the RL in robots nowadays is done in simulated environments, as the damage caused by unsafe exploration in real world is tremendous. So far, there are many safety features that have been proposed, including limiting torque output through impedance control, collaborative robot detection of abnormal forces and so on.

But there are still many challenges in the field of robotics in terms of safety. As robots work in more complex environments, the scene of cooperation with people becomes more frequent, which requires robots to be able to adapt to more unstructured and dynamic environments. In the human-robot interaction environment, the security of human has to be always guaranteed [3], so the robot should detect the state of the human in real time. Therefore, distance-based constraints are essential to allow the robot to leave the structured laboratory and evaluate collisions with surrounding static and dynamic obstacles online. But constraint functions can be difficult to design manually, and unspecified constraints may often make the robot challenging to plan, optimize, learn or act reactively [2, 4, 5, 6, 7]. This problem can be well solved by Signed Distance Field (SDF), which is a mathematical function that can be easily converted to an implicit equation defining the constraints. The SDF allows us to easily check the distance to the

obstacle. Through determining the safety distance, we can achieve safety control for the robot.

However, different applications currently require different distance resolutions, which has led to various heuristics to measure distance fields. There are also methods to learn distance fields through neural networks. These methods are either computationally expensive and cannot estimate distance in any scale, or cannot be applied to the distance calculation for dynamic obstacles. Therefore, we propose a novel type of SDF: Regularized Deep Signed Distance Field (ReDSDF), which is a single neural implicit function, and can compute smooth distance field in any scale even for dynamic articulated object.

ReDSDF is learned through neural network, where we add the dimension of pose at the input so that it can represent different poses of dynamical articulated objects, and regularize at the output so that it can generate distance fields at any scale. We propose our own data generation approach, network architecture and training method to enable better performance of distance fields.

This kind of distance field can be used in many methods of robot collision avoidance control. One of the more popular methods used in robot collision avoidance is reactive motion generation. Reactive motion generation provides a good framework for controlling robots to avoid collisions through field functions. ReDSDF, as an implicit neural network, can easily calculate its gradient, so it can be applied as a field function in any methods with reactive motion generation. In RL, there are also many approaches to achieve collision-free exploration of robots by imposing constraints. We refer to this kind of exploration as Safe Exploration (SafeExp), i.e., the safe collection of experiences without violating task and geometric constraints when training robot strategies. The constraints need to be expressed in terms of implicit equations. ReDSDF as a mathematical function, can be easily converted to an implicit equation and define the constraints. Aiming at the above two applications: reactive motion control and SafeExp, we designed four experiments to validate our approach and also the performance of ReDSDF. Both these applications have been experimented in static and dynamic environments, i.e. human-robot interaction environment. We demonstrate the performance of ReDSDF qualitatively and quantitatively by comparing with two baselines. And we also demonstrate the capabilities of ReDSDF when employed in reactive motion generation and SafeExp task. The results of these tasks are also in comparison with the baseline methods.

The structure of this thesis is as follows. Firstly, in Chapter 2, the current related work is presented in five aspects which contains the safe human-robot interaction, three-dimension human body reconstruction, reactive motion generation, manifolds to define constraints and SafeRL. Next, Chapter 3 presents four preliminaries used in this thesis. Chapter 4

focuses on ReDSDF. In this chapter, the data generation approach, the network architecture and the training method of ReDSDF will be presented. Two methods of applying ReDSDF will be introduced in Chapter 5, including reactive motion generation and SafeRL. In this part, we will describe, how the ReDSDF can be used as potential field in reactive controller and as constraints in SafeExp. In Chapter 6, four kinds of experiments will be performed. These experiments include two experiments applying ReDSDF in reactive controller and two experiments on SafeExp. Both static and dynamic environments are used here including human-robot interaction workspace. Finally, in Chapter 6, the results of the thesis work and experiments are summarized and future work is presented.

2. Related Work

In this chapter, some of the work related to this thesis will be presented. All thesis-related work will be divided into five categories as follows. Section 2.1 is a discussion of current approaches regarding safe human-robot interaction. Section 2.2 is about some methods of 3D human reconstruction in virtual environments. In section 2.3, some methods of reactive motion generation in robot control will be covered. Section 2.4 introduces some of recent distance networks, and how authors of other papers have used these manifolds to express some of the robot's constraint spaces. Finally, section 2.5 describes some of the existing safe reinforcement learning algorithms.

2.1. Safe Human-Robot Interaction

Robots and humans can complement each other's capabilities when executing a common task. The advantage of robots is that they can perform repetitive and hazardous works. People can benefit from this advantage and perform more intellectually demanding tasks. However, detecting human poses and understanding the effective safe workspace of the human is essential. The human-robot interaction scenarios are human-centered, and the effective space of human is designated firstly [8]. During human-robot interaction, the robot must ensure the safety of the human.

Corrales et al. [3] proposed sphere-swept lines (SSLs) as bounding volumes to make sure the safe human-robot interaction. The method envelops each link of the human body and robot with a cylinder. The diameter of the cylinder can be changed according to the speed of the link. The most common method of preventing robot collisions today is to use spheres to model humans and robots. This approach is described in [9]. The safety constraints of the robot can be defined by designing the radius of the sphere so that the sphere space can envelop the human or robot. Papadakis et al. [10] proposed a method for training a social map based on human social behavior and thus to study the safety of

human-robot interaction. In [8], all the workspace is voxelized and analyzed by traversal. All locations of the workspace will be evaluated based on two aspects: local dexterity of the human arm and energy consumption of arm postures. Vogt et al. [11] presented an approach for robots to learn human-robot interaction by imitating through human-human Demonstrations.

There are still many approaches to ensure safe human-robot interaction. The experimental part of this thesis will also use sphere-based approach that described in [9] as a baseline for comparison with the method of this thesis.

2.2. 3D Human Body Reconstruction

In order to enable neural networks to dynamically represent the constraints in human-robot interaction, 3D reconstruction of the human body is also essential. Details about building human models can be found in the book [12]. The reach volumes of body links are also measured in [13]. Reconstructing high quality human shapes is a challenge due to non-rigid body deformation, low quality input data and joint articulation [14]. Skinned Multi-Person Linear Model (SMPL) in [15] has solved these problems. It uses pose and shape parameters to reconstruct human body with many vertices. The dataset to train the network in this thesis is also generated by SMPL. What's more, there are now many other ways to generate human body from pointcloud or pictures.

Jiang et al. [14] designed and trained a neural network to reconstruct SMPL from pointcloud. The network allows online, real-time detection of point clouds and 3D reconstruction of the human body. The authors use PointNet++ and Graph Aggregation (GA) to extract features from pointcloud. These features will be fed into a skeleton diagram module that estimates the pose and shape parameters of the SMPL. Bhatnager et al. [16] proposed a method to combine implicit function learning and parametric models to reconstruct human body. They used the SMPL to generate the dataset and added classifiers to the network to classify the points in different body parts to train the implicit function. Li et al. [17] presented an approach to reconstruct human body with multi-view images. In this approach, the authors also use implicit functions to represent the human body. They designed a multiple convolutional network with pooling in parallel to gradually extract detailed features. A method of real-time human model reconstruction is mentioned in the [18]. This method can estimate the human pose from the picture information and generate SMPL models. The authors first use the yolo network to detect the human in the picture, determine the position of the human in space based on the position of the

bounding box, and estimate the pose of the human body through the network. Choutas et al. [19] proposed an approach to regress body shape from an image of a person by using metric and semantic attributes. However, these methods have limitations in human-robot interaction, as it is difficult to accurately estimate the position of a person through pictures.

2.3. Reactive Motion Generation

Reactive motion generation is a common method of obstacle avoidance in robot control. The robot generates the corresponding motion directly based on the current state and the surrounding environment. The most commonly used method for reactive motion generation is to obtain the gradient from the Artificial Potential Field (APF) [20] to obtain the corresponding robot control motion. This method has low computational cost, but it tends to fall into local optima. To compensate for these shortcomings, other methods of reactive motion generation are introduced.

Ratliff et al. [6] proposed a motion policy on Riemannian Manifold. This policy is referred to Riemannian Motion Policy (RMP) that is a second-order dynamical system coupled with a corresponding Riemannian metric. The method can combine the motions generated by various policies and steadily transform the motion from one space to another. Through a combination of these policies, the robot is able to perform obstacle avoidance while completing tasks. Urain et al. [21] proposed a novel framework for modular reactive motion generation: Composable Energy Policies (CEP), which computes control actions by optimizing the product of a set of stochastic policies. The method can combine different policies to eventually generate an action that maximizes the satisfaction of all objectives. In [22], Beik-Mohammadi et al investigated the robot motion learning paradigm from a Riemannian manifold perspective. They used geodesics to generate a learned Riemannian metric produced by a novel variational autoencoder (VAE). The geodesics are natural motion skills and the VAE is used to recover full-pose end-effector states and joint space configurations.

There are many other ways to control robots through reactive motion generation. Each of these methods requires the definition of field functions to allow the robot to perform the appropriate tasks and avoid obstacles. We will describe the construction of these field functions in the following section.

2.4. Distance Fields and Manifold

The usage of Signed Distance Fields (SDFs) is to reconstruct the network of surrounding objects and has been widely studied in the field of computer graphics [23]. With a given value, the SDF can represent a manifold. Many robot motion planning and control, collision checking and obstacle avoidance problems can be solved by SDF. The SDFs are locally accurate because of the truncation effects. However, it is challenging when the entire shape of an object cannot be determined from a single viewpoint. An approach of volumetric integration was proposed in [23]. The method improves the local update based on partial observations by sacrificing the full coverage of space. This field obtained by truncation is called Truncated Signed Distance Function (TSDF), which realizes a better modeling for sensor noise [24]. In contrast, an alternative Euclidean Signed Distance Function (ESDF) provides an approach to evaluate free space rather than fine obstacle areas. This approach is widely used in aerial robot mapping and planning [25].

With the development of deep learning, SDFs can be represented in a better way. The fields can be trained to approximately mesh with the dataset in form of pointcloud, which is generated by the object meshes [26]. Park et al. [27] proposed DeepSDF that uses the deep neural network to learn the SDF. They generate pointcloud as dataset from mesh models and then define areas not covered by the dataset by truncation. The input of this network contains not only the position of the 3D point, but also the codes of several dimensions, and these codes can be used to represent different objects. Sutanto et al. [28] proposed an approach called Equality Constraint Manifold Neural Network (ECoMaNN) to restrict robot motion planning on manifold. The training set of this network contains not only the 3D position of the pointcloud, but also the normal direction of the point in the distance field. Many augmented points are also generated uniformly through this normal direction to train the network. The authors also proposed a method for aligning the local tangent and normal space. The manifold is also suitable for some high-dimensional cases. The robot performs motion planning in a high-dimensional space on the manifold and ensure an equality constraint. The two distance networks mentioned above will also be compared with the distance network proposed in this thesis as baselines in the subsequent parts.

In addition to the SDF, there are also many approaches to establish the manifolds and to perform controlling on the manifolds. Mescheder et al. [29] proposed an occupancy networks to realize 3D reconstruction in function space. The network is similar to a classification network, except that the boundary of the classification is used as the surface of the 3D-object. Holden et al. [30] learned a motion manifold with convolutional

autoencoders. The manifold can be used as a priori probability distribution to eliminate errors, compute similarity, and interpolate. In [31], the manifold was applied in the optimization problem of motion planning. In the work of this thesis, the manifold is also added to the robot as constraints so that the robot can perform safe work.

2.5. Safe Robot Reinforcement Learning

In recent years, the artificial intelligence has grown by leaps and bounds. At the same time, we must ensure that their ability to be safe is similarly increased, because the tremendous positive impact of these applications comes with an increased need for security measures, as the failure of any of these intelligent systems can be catastrophic [32]. Therefore, Safe Reinforcement Learning (SafeRL) has been a topic of great interest for a long time and many different ways to achieve SafeRL were proposed.

Some methods use safe states set to specify safe states. This safe states set is either predefined before training or extended during training. Hans et al. [33] proposed an approach to evaluate the safety degree of the state and define a backup policy. The backup policy can lead the system from critical state back to safe under control. Garcia et al. [34] defined a risk function and a baseline agent. During the learning, control actions will be sampled according to the evaluation of risk. Akametalu et al. [35] proposed a method to learn the system's unknown dynamics based on a gaussian process model. The safe set will be extended in training iteratively and approximated to the maximal safe set. Pecka et al. [36] combined two basic concepts: safe function and classifier. The classifier determines whether the current state is safe through the evaluation of the safe function.

Another way to achieve SafeRL is to formalize the problem to Constrained Markov Decision Processes (CMDP). CMDP adds constraints to the Markov Decision Process (MDP) and learns the optimal policy. Altman et al. [37] investigated the Lagrangian approach and a related Linear Programming that appear in CMDP to obtain better learning results. Achiam et al. [38] proposed the first general-purpose policy search algorithm for constrained reinforcement learning called Constrained Policy Optimization (CPO). This method can guarantee that each iteration has a near-bounded satisfaction. There are also many other approaches that train the CMDP based on Lagrangian optimization [39, 40, 41]. And some algorithms penalize the actions that break the constraints [42, 43]. There are also many approaches to transfer the action to the safe constraints at each step, and each of these methods has its own way of defining constraints, for example, the Hamilton-Jacobi [44, 45], Control Barrier Function [46, 47]

There are another two methods that are the focus of this thesis. The first approach is to adjust the action to the safe constraints at each step and this adjustment process is formulized as a constrained optimization problem (safe explorer) [48]. The method will be used as a baseline for comparison with the method applied in this thesis. The second method is Acting on the Tangent Space of the Constraint Manifold (ATACOM) [2]. This method will be applied as the focus of this thesis.

3. Preliminaries

In this chapter, all the preliminaries used in the thesis will be introduced, namely, implicit function that represents the SDFs, APF, fundamentals of RL and ATACOM. These four approaches will be presented in four separate sections.

3.1. Implicit Function and Signed Distance Field

An implicit function is a function that is defined by an implicit equation that relates one of the variables, considered as the value of the function, with the others considered as the arguments [49]. In constrained optimization problems, the constraints are generally given in the form of implicit functions. The position constraint problem on three-dimensional space is also a type of constraint problem. Meanwhile, many of today's 3D reconstruction methods are achieved by applying the implicit functions. The most intuitive mathematical expression of the implicit equation takes the form:

$$f(\mathbf{x}) = 0, \quad (3.1)$$

where \mathbf{x} can be a multidimensional vector or a scalar. The variables in the implicit equation can also be derived. In the 3D reconstruction, \mathbf{x} is a vector with three dimensions. The set of all points that satisfy the implicit equation forms a surface. This surface can also be called 3D manifold. This manifold is also the surface of the object we want to reconstruct.

SDFs are also a type of implicit equation, but with more requirements for implicit equations in 3D reconstruction. The implicit equation expresses not only the surface of the object, but also the distance between any point in space and the object. So the implicit equation of SDF becomes the following form:

$$f(x, y, z) = d. \quad (3.2)$$

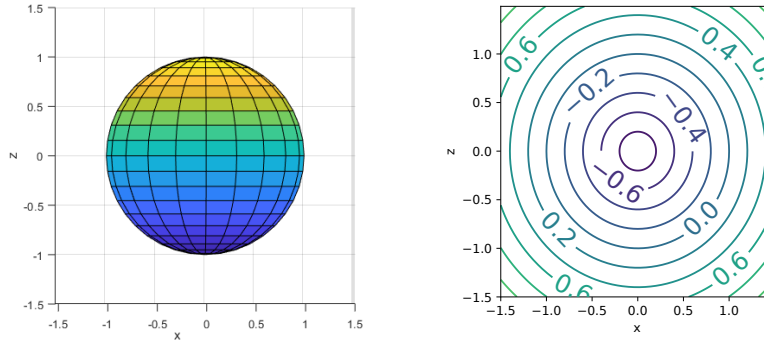


Figure 3.1.: An Example of SDF that Represents a Ball

The formula expresses the meaning that the minimum distance of the point at position (x, y, z) from the surface of the object is d . The value of d is negative if the point is inside the surface of the object, and positive if it is outside the surface of the object. So the different value of d can represent different manifold. For example, the implicit equation of SDF for a ball of radius 1 is:

$$\sqrt{x^2 + y^2 + z^2} - 1 = d. \quad (3.3)$$

The surface of this ball is shown on the left side of Figure 3.1. If we observe only the xz -plane with y -value of 0, the SDF looks like the field shown on the right side of the figure. From the figure, we can see that the fields inside the ball are all negative and the fields outside the ball are all positive. The direction of the gradient of the field is the direction of manifold normal. This theory is proven in the book [50]. Therefore, if the objects around the robot can be converted into mathematical representation of SDFs, we can define constraints in 3D space with SDF to keep the robot away from collisions. Also we can define the safety distance arbitrarily to improve the reliability of the system.

3.2. Artificial Potential Field

The Artificial Potential Field (APF) [20] method is a classical robot path planning algorithm. The algorithm regards the target and the obstacle as objects that have attractive and repulsive forces on the robot, respectively, and the robot moves along the combined forces of attraction and repulsion. APF is also one of the core ideas of reactive motion generation.

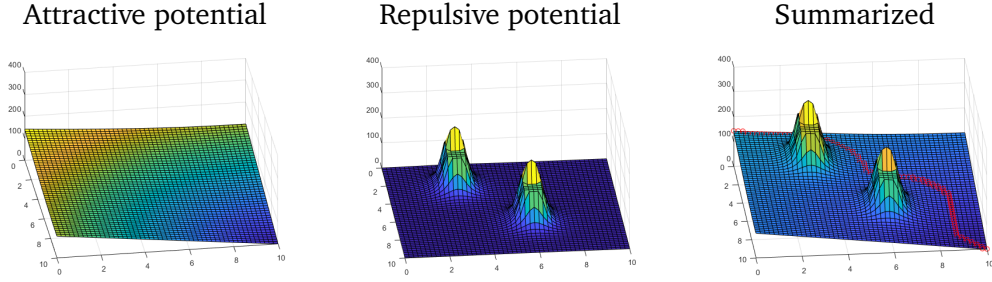


Figure 3.2.: An Example of APF Visualization

The attractive potential function can be constructed as follows:

$$f_a(\mathbf{x}) = \xi(\|\mathbf{x} - \mathbf{x}_g\|^2), \quad (3.4)$$

where ξ is a constant scaling parameter, \mathbf{x} is the position of robot and \mathbf{x}_g is the target position. The position can be either a location in the task space or a higher dimensional location in the joint space.

In addition to enabling the robot to reach the target point, we also want the robot to be able to avoid obstacles in the environment. Therefore, a repulsive field function is designed as follows:

$$f_r(\mathbf{x}) = \begin{cases} 0 & \rho(\mathbf{x}) > d_0 \\ \eta \left(\frac{1}{\rho(\mathbf{x})} - \frac{1}{d_0} \right)^2 & \rho(\mathbf{x}) \leq d_0 \end{cases}, \quad (3.5)$$

where η is a constant scaling parameter and d_0 is a parameter that controls the influence of the repulsive potential. $\rho(\mathbf{x})$ is a function that returns the distance to the closest obstacle from the given point \mathbf{x} . There are also many ways to construct smooth repulsive functions to lead the robot away from obstacles. The repulsive functions of different obstacles can be superposed on each other. With these potential field functions, we can find the gradient on them to obtain the direction of the robot action. Here the action we express in terms of velocity.

$$\mathbf{v} \propto -\nabla f(\mathbf{x}) = -\nabla \left(f_a(\mathbf{x}) + \sum_{i=1}^N f_{r,i}(\mathbf{x}) \right) \quad (3.6)$$

An example of APF visualization can be seen in Figure 3.2. The left figure shows the attractive potential field generated by the target point, and the middle figure shows the

repulsive field generated by the obstacle. The final APF can be obtained by superimposing these two fields, and the direction of control can be found by finding the gradient of the APF. The motion of the robot is only related to the current position, so it is particularly fast to find the motion. In this thesis, we use the repulsive fields that are different from the ones described above.

3.3. Fundamentals of Reinforcement Learning

Reinforcement Learning (RL) is the process by which an agent learns through trial and error by observing the system environment in order to maximize the expectation of long-term rewards [51]. From the definition of RL, it can be seen that RL has two key attributes:

- Learning through trial and error
- Maximizing long-term returns

And the framework of RL generally consists of 5 components: environment, agent, observation, action and reward. The demonstration of RL framework can be seen in Figure 3.3. The RL problem can be defined as policy search in a Markov decision process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where p is transition function that represents the probability density of the next state $s_{t+1} \in \mathcal{S}$ given the current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$. The definition of reinforcement learning can also be expressed as a mathematical formula:

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r(s_t, a_t) \right], \quad (3.7)$$

where γ is the discount factor and π is the policy. This discount factor weighs the value of long-term returns against short-term returns. The environment returns a reward r for a state s with an action a . The reward function can be defined by the tasks that the robot needs to complete. The observation of the system state by the agent can be divided into two cases: full observable and partial observable. Depending on the agent's knowledge of the environment, RL can be divided into model-based RL and model-free RL. The RL methods used in this thesis are all fully observable and model-free. We will address learning the policies in continuous action space. There are now many algorithms to learn the policy. The algorithm that used in this thesis is Soft Actor-Critic (SAC) [52], a deep RL algorithm. The SafeExp approach in this thesis is also applicable to other RL

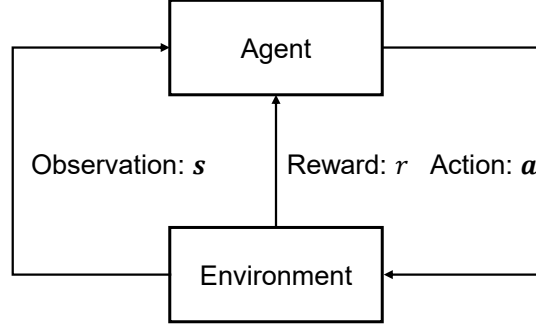


Figure 3.3.: The Framework of RL

algorithms, which will not be described here. SAC is an off-policy actor-critic algorithm based on the maximum entropy RL framework. With this framework, actors can act as randomly as possible while achieving their tasks.

3.4. Acting on the Tangent Space of the Constraint Manifold

The role of ATACOM is to keep the state of MDP in the constraints in RL. The problem of RL with constraints can be formulized as follows:

$$\begin{aligned}
 & \max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^T \gamma^t r(s_t, \mathbf{a}_t) \right], \\
 & \text{s.t.} \quad \mathbf{f}(\mathbf{q}_t) = 0, \quad \mathbf{g}(\mathbf{q}_t) \leq 0 \\
 & \quad \mathbf{s}_t = [\mathbf{q}_t, \mathbf{x}_t]^\top, \quad t \in \{0, 1, \dots, T\},
 \end{aligned} \tag{3.8}$$

where $\mathbf{f}(\cdot)$ is equality constraints and $\mathbf{g}(\cdot)$ is inequality constraints. In this equation, $\tau = \{\mathbf{s}_0, \mathbf{a}_0, \dots, \mathbf{s}_T, \mathbf{a}_T\}$ is the trajectory under the policy π . \mathbf{q} and \mathbf{x} are directly controllable states and uncontrollable states respectively [2].

In order to put the inequality constraint on the manifold as well for restriction, the slack variables $\boldsymbol{\mu}$ are added to transform the original inequality into equality constraints. Thus, all the constraints can be summarized together:

$$\mathbf{c}(\mathbf{q}, \boldsymbol{\mu}) = [\mathbf{f}(\mathbf{q}) \quad \mathbf{g}(\mathbf{q}) + \epsilon(\boldsymbol{\mu})]^\top = 0, \tag{3.9}$$

where $\epsilon(\boldsymbol{\mu})$ is designed that it is always positive, for example we can use quadratic slack variable:

$$\epsilon(\boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{\mu}^2, \quad (3.10)$$

or exponential slack variable:

$$\epsilon(\boldsymbol{\mu}) = \exp(\beta\boldsymbol{\mu}). \quad (3.11)$$

In the Equation 3.11, the hyperparameter β can be adjusted that how close to the constraint boundary the robot should decelerate. Therefore, we can consider the constraints as a manifold and let the state motion on the manifold. If the action is in the tangent space of the manifold, then the current state is guaranteed to remain on the manifold. Then we need to solve for the Jacobian of constraints to obtain the tangent space of the manifold:

$$\mathbf{J}(\mathbf{q}, \boldsymbol{\mu}) = [\nabla_{\mathbf{q}}c(\mathbf{q}, \boldsymbol{\mu})^\top \quad \nabla_{\boldsymbol{\mu}}c(\mathbf{q}, \boldsymbol{\mu})^\top], \quad (3.12)$$

where the tangent-space $\mathcal{N}(\mathbf{q}, \mathbf{s})$ is the nullspace of the Jacobian. With nullspace, we can project the action into tangent space:

$$[\dot{\mathbf{q}}^\top \quad \dot{\boldsymbol{\mu}}^\top]^\top = \mathbf{N}\boldsymbol{\alpha} - K_c\mathbf{J}^\dagger\mathbf{c}. \quad (3.13)$$

In this equation, $\boldsymbol{\alpha}$ is the action sampled from the policy, and the second term is the error correction term, which forces the state on the manifold. The hyperparameter K_c affects this error correction term: how fast the agent will return to the constraint if the present state breaks the constraint. Through ATACOM we can not only verify whether the constraints learned in this thesis meet the application requirements, but also validate the reliability and performance of SafeExp through experimental results.

4. Regularized Deep Signed Distance Fields

In this chapter, we will describe how ReDSDF is implemented to be able to compute smooth distance fields for static and articulated objects at any scale. At the beginning of this chapter, the method for constructing the dataset used to train the network is described in Section 4.1. And then Section 4.2 introduces the architecture of ReDSDF. Later in Section 4.3, we will talk about how to train this network.

4.1. Generation of Dataset

If the network is used as a constraint, the smoothness and accuracy of the field must be guaranteed, otherwise the robot will experience vibrations and unplanned collisions. So the generated data must be accurate. As SDF, the input of the neural network is a coordinate of point x in three dimensions and the output is a scalar d representing the distance. Therefore, our dataset is a 3D pointcloud with labels, the distance from the point to the surface of the object. To enable the network to represent dynamical articulated objects, we add the dimension of pose to the input. To make the trained distance field smoother and the gradient direction more accurate, we add the normal direction of the object in the dataset just like [28]. To enable the network to represent some details of the object surface, we use the method of non-uniform generation of augmented points and introduce weights. Figure 4.1 shows the pipeline of generating the datasets for one pose. Each step will be described in detail below.

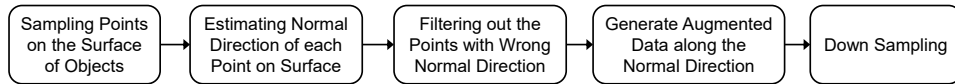


Figure 4.1.: Pipeline of Generating the Datasets

4.1.1. Sampling Points and Estimating Normal Direction

The initial object model to be trained is a mesh model, or a point cloud model. If the object is already represented by a point cloud, there is no need to sample it. For files represented in mesh form, the sampling here is performed only on the surface of the object, and no consideration is given to the internal structure. If too many points are sampled, it will cause problems of slow data generation and training, but if too few points are sampled, the normal direction is not estimated correctly. Empirically, it is generally appropriate to take 10000 data points for each pose. To better predict the normal direction, for complex objects, the number of sampling points should be larger and the detailed parts should be sampled more. Here we use depth camera to capture multi-view pointcloud, and Figure 4.2 illustrates this process. We can adjust the angle and parameters of the camera to adjust the sparsity of the point cloud on each part of the object surface. If there are too many points in the end, they can be downsampled once to make the total number of points close to 10,000.

For many objects represented in mesh form, the normal direction of the mesh is already stored in a file or the normal direction can be calculated directly, e.g. SMPL. These objects do not need to predict the normal direction or filter out the points with wrong normal direction. For other objects, the normal direction will be estimated through the method proposed in [28, 53]. And we use the normal estimation method in the Open3D library [54].

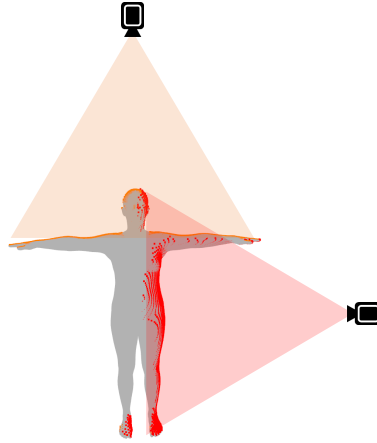


Figure 4.2.: Demonstration of Sampling Points

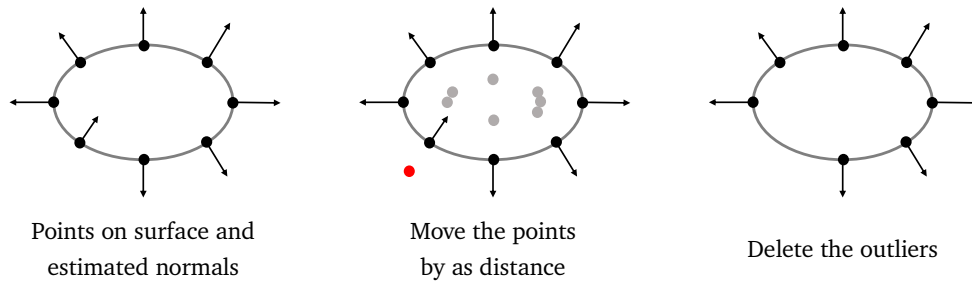


Figure 4.3.: The Process of Removing Outliers

4.1.2. Filtering out Outliers and Data Rejection

The normal direction estimated by the pointcloud is not particularly accurate. Because what we need to train is the distance field with sign, the direction of the normal must point out of the object. Any opposite normal direction can cause outliers of augmented data, which makes the quality of the distance network very poor. Therefore, the point with wrong normal direction should be deleted. The method of deletion here is to use the kd-tree. The process of filtering out the outliers is shown in Figure 4.3. First the original point will be moved by a distance in the opposite direction of the estimated normal. And then the closest original point will be found in the kd-tree for each point after the move. If the closest original point is not where it was before it moved, exclude this point. Then, look for the outlier in the remaining moved points. Finally, find the outlier in the remaining moved points and delete the original points of these outliers from the dataset, because these outliers will influence the quality of the SDF.

The SDF predicts the distance closest to the surface of the object. Therefore, for some augmented points, if the nearest original point is not the one that generated it, these augmented points will be rejected. The demonstration of this process can be seen in Figure 4.4. This rejection process is also implemented through the kd-tree. All the original data points are saved into the kd-tree before the augmented points are generated. Whenever an augmented point is generated, the nearest original point from the kd-tree is queried, and if the original point is not the one from which this augmented point is generated, this augmented point is rejected.

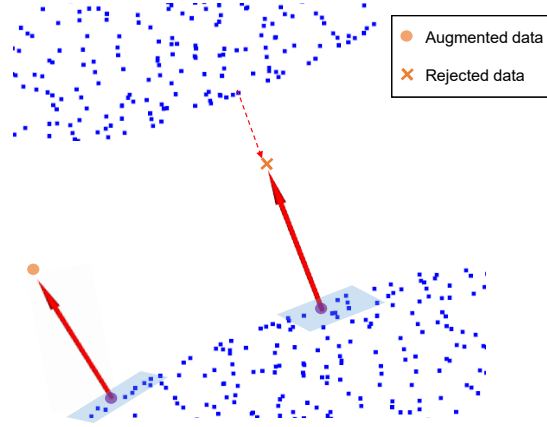


Figure 4.4.: The augmented points are generated along the normal direction. The red arrow is the normal direction, the blue point is the original point. When the nearest original point to the augmented point is not the original point that generated it, the point will be rejected. The orange sphere is the accepted augmented point, and the orange fork is the rejected augmented point

4.1.3. Generating Augmented Data and Down Sampling

Next, the augmented data are generated along the positive and negative normal directions. The data augmentation is visualized in Figure 4.4, where the red arrow is the normal of purple point, and the orange point is augmented data in the direction of normal. In practice, many layers of augmented data are generated in different levels. In order for the SDF to have better quality close to the surface of the object, there are more augmented points near the object and fewer augmented points at position far away from the object such as in level $[-0.10, -0.05, -0.02, -0.01, 0.00, 0.01, 0.02, 0.05, 0.10, 0.20, 0.50]$. In order to have the same training effect at every position on the surface of the object, weights are used here, those with fewer augmented points correspond to higher weights and those with more augmented points correspond to lower weights. The demonstration of weight assignment can be found in Figure 4.5. Eventually sum of the weights of each original point and the augmented points it generates are equal. The Figure 4.7 shows an example of data augmentation of human with the levels: 0, 0.05, 0.12 and 0.25.

For dynamic articulated models, if all the generated data is used for training, the dataset will be particularly large. In order to improve the training efficiency, we should downsample the generated dataset. Downsampling is performed for all original points as well as

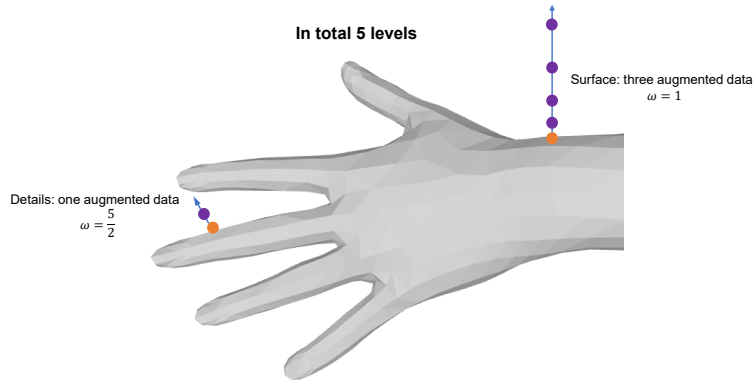


Figure 4.5.: Demonstration of Weight Assignment

augmented points of each pose, and finally get 80,000 points of each pose for training. Up to this, data generation is finished with respect to static objects. But for dynamical articulated objects, the above processes are repeated for each pose to generate the dataset.

4.1.4. Data Generation for Dynamic Articulated Objects

For static objects, we do not need the dimension of pose. So the above process only needs to be performed once. But for different articulated objects, there is different number of dimension for the pose. We also need to sample the different poses of the object, and

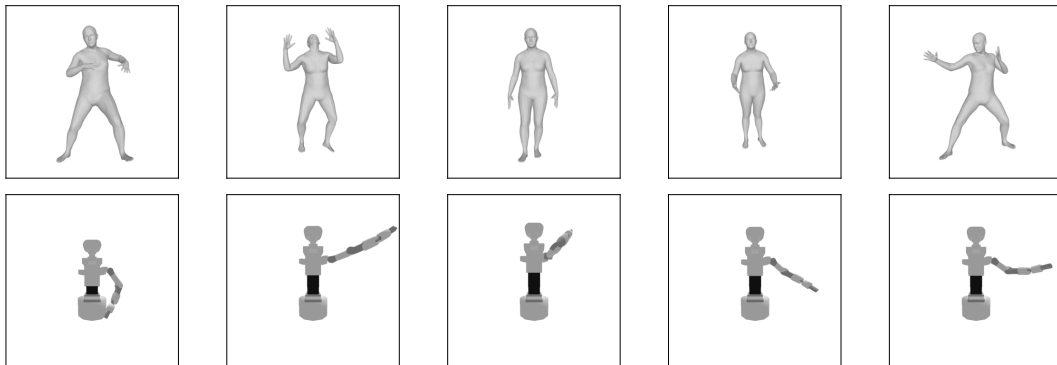


Figure 4.6.: Articulated Object with Different Poses

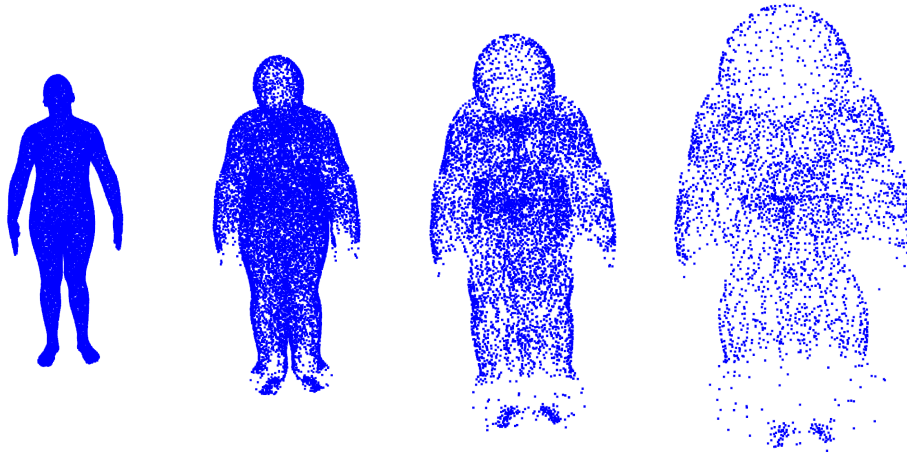


Figure 4.7.: Example of Data Augmentation of Human with Different Levels: 0, 0.05, 0.12 and 0.25

then repeat the process for each pose. In this thesis, we trained in total two dynamic articulated objects, namely the human body and the robot: TIAGo++. These two objects with different poses are shown in Figure 4.6. The subsequent paragraphs describe how to obtain pose information in the dataset.

This thesis uses SMPL as ground truth of human body. We use SMPL to generate dataset. The first row in Figure 4.6 shows the SMPL models with different poses. These specific postures and actions of the human body are taken from the dataset AMASS [55]. This dataset contains most of the daily human behaviors and actions by using the SMPL model. We came up with a total of 10,000 such different poses and constructed the final dataset in AMASS to train the ReDSDF. Since we do not need to consider the root rotation or model the finger, the dimension of the pose in the generated dataset is 63 dimensions (72 dimensions in total with 3 dimension of root and 6 dimension of fingers). The meaning of each dimension is the same as the meaning of pose parameter in SMPL. We then perform the above process of generating data points for each of the 10,000 selected poses to form the final dataset, which is used to train ReDSDF.

The other articulated object that was trained in this thesis is robot: TIAGo. In order to represent the distance field of the robot with fewer dimensions and to make the training efficient, we simplified the mesh model of the robot. The simplified robot is shown in the second row of Figure 4.6, with only a single arm. If we want to obtain the distance field

Contents	Dimension	Mathematical notation
Position	3	\boldsymbol{x}
Normal direction	3	\boldsymbol{n}
Distance to the surface	1	d
Weight	1	ω
Pose	n_q	\boldsymbol{q}_o

Table 4.1.: Contents of the Final Dataset

of the other arm, we only need to mirror the distance field. So the robot has a total of 8 pose dimensions (1 dimension for torso and 7 dimension for single arm). We randomly selected 10,000 poses in the robot’s joint-limited space without collision. We randomly selected 10,000 poses in the joint-limited space of robot in the collision-free case. For each pose we perform the process of generating data points as described above to construct the final dataset.

We eventually obtain a dataset for training ReDSDF as shown in Table 4.1. Static objects have no pose dimension. The pose dimension of dynamic articulated object is determined by the type of object. In this thesis, the pose dimension of human body is 63, and the pose dimension of robot is 8.

4.2. Architecture of the Network

In order to apply SDF to robot control as well as RL, the distance field must provide precise distances anywhere in workspace. The input of the network should include the pose of articulated, like human. For the area near the object, the details of the object are of interest. But for regions far from the object, the scale of the distance becomes larger and the details of the object are no longer important. So we only need to consider the object as a point. Based on these requirements, we propose a distance network across all regions: ReDSDF. The structure of ReDSDF is shown in Figure 4.8. Here we assume that the network has 5 hidden layers.

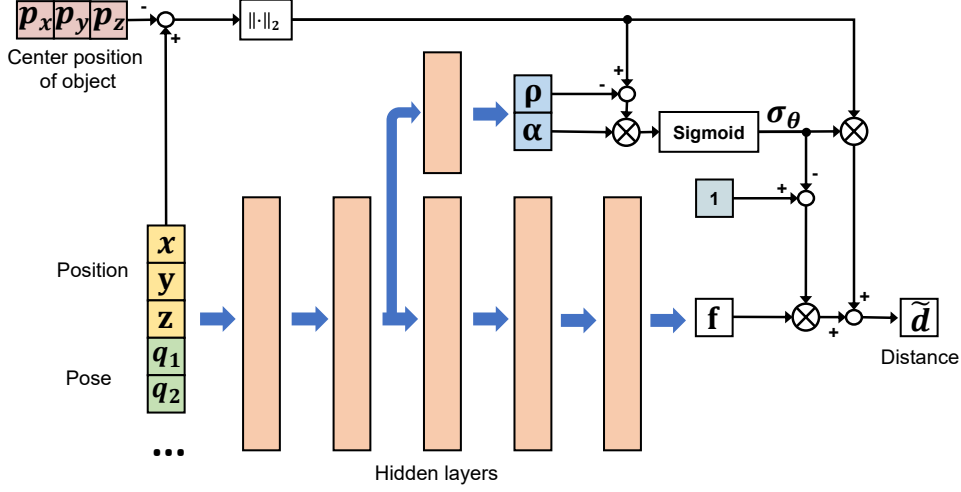


Figure 4.8.: Architecture of ReDSDF

The idea of using a distance field to express a distant in the region that the augmented data doesn't cover is to consider the object as a point, and we find a threshold as a transition from the near area (where the augmented points cover) to distant region (where the augmented data don't cover). Assumed that an object with its center at position \mathbf{p} can be wrapped in a bounding sphere with radius r . If we observe the object at a great distance with $\|\mathbf{x} - \mathbf{p}\|_2 \gg r$, the distance to the surface of the object is approximately equal to the distance to the center of the object. So we have the distance:

$$d \approx \|\mathbf{x} - \mathbf{p}\|_2. \quad (4.14)$$

On top of this, we add a transition from near area to far region, and this transition function is Sigmoid function:

$$\tilde{d}(\mathbf{x}, \mathbf{q}) = (1 - \sigma_{\theta}(\mathbf{x}, \mathbf{q}))f_{\theta}(\mathbf{x}, \mathbf{q}) + \sigma_{\theta}(\mathbf{x}, \mathbf{q})\|\mathbf{x} - \mathbf{p}\|_2, \quad (4.15)$$

where \tilde{d} is the estimated distance, θ represents the learnable parameters of network and $f_{\theta}(\mathbf{x}, \mathbf{q})$ is the neural network approximator. The transition of $\sigma_{\theta}(\mathbf{x}, \mathbf{q})$ is defined as follows:

$$\sigma_{\theta}(\mathbf{x}, \mathbf{q}) = \text{sigmoid}(\alpha_{\theta}(\mathbf{x}, \mathbf{q})(\|\mathbf{x} - \mathbf{p}\|_2 - \rho_{\theta}(\mathbf{x}, \mathbf{q}))), \quad (4.16)$$

where α_{θ} and ρ_{θ} are shaping functions, implemented as neural networks, and the features are extracted from the hidden layer. Here, ρ_{θ} determines the threshold from the close

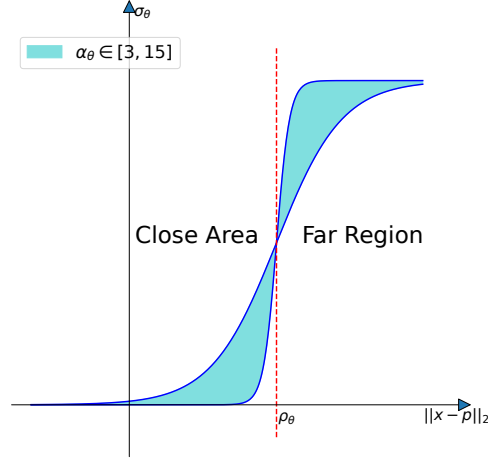


Figure 4.9.: The parameters of Sigmoid function defines the transition area between close area and far region. The distance field in close area is determined by neural network, and the distance field in far region is determined by the distance to the center of object.

area to far region and α_θ determines the Smoothness of the transition. We can see in the Figure 4.9 the shape of Sigmoid function with different value of these two parameters. Figure 4.10 shows how the parameters α_θ and ρ_θ of Sigmoid function affect the transition between the close area and far region.

The main part of the network of ReDSDF that calculates f_θ is consists of 4 fully-connected hidden layers for static models, 5 fully-connected hidden layers for dynamic articulated models. There are 512 elements in each hidden layer. The activation function is ReLU and the last layer is linear connected without activation function. The hidden layer to calculate α_θ and ρ_θ composed of 32 elements. The output of α_θ is constrained to be positive by using a Softplus function, and the output of ρ_θ is limited between 0.5 and 1.5 by a Sigmoid function with a bias. Batch normalization and drop out are not included in the network.

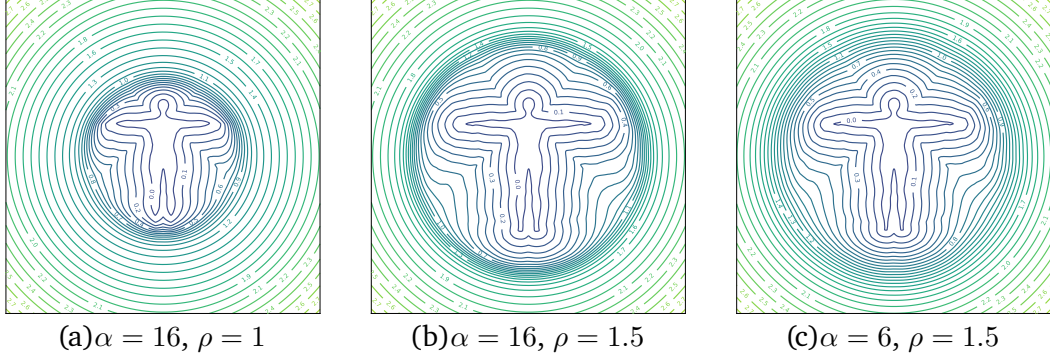


Figure 4.10.: Three Examples of ReDSDF with Different Parameters of Sigmoid Function.

4.3. Training of the Network

Before training, all datasets are divided into training set, validation set and test set in the ratio of 0.8, 0.1, 0.1, where the training set is used to train the neural network, the validation set is used to evaluate the performance of the network and select the hyperparameters, and the test set is used for the final results presentation. In the training process, the network will be trained by optimizing the following loss function. The loss function can be divided into three sections:

$$\mathcal{L}(\mathcal{D}) = \mathcal{L}_1(\mathcal{D}) + \mathcal{L}_2(\mathcal{D}) + \mathcal{L}_3(\mathcal{D}), \quad (4.17)$$

where

$$\mathcal{L}_1(\mathcal{D}) = \sum_{i \in \mathcal{D}} \omega_i \left(\tilde{d}_{\theta}(\mathbf{x}_i, \mathbf{q}_i) - d_i \right)^2, \quad (4.18)$$

$$\mathcal{L}_2(\mathcal{D}) = \sum_{i \in \mathcal{D}} \left(\| \mathbf{N}_i \tilde{\mathbf{n}}_{\theta}(\mathbf{x}_i, \mathbf{q}_i) \|_2^2 + \| \tilde{\mathbf{N}}_{\theta}(\mathbf{x}_i, \mathbf{q}_i) \mathbf{n}_i \|_2^2 \right), \quad (4.19)$$

$$\mathcal{L}_3(\mathcal{D}) = \sum_{i \in \mathcal{D}} \gamma \rho_{\theta}(\mathbf{x}_i, \mathbf{q}_i)^2. \quad (4.20)$$

The loss of the first item represents the weighted MSE-loss between the target value and estimated results. And the second item of loss is similar to the one proposed in [28],

where N_i is the null-space of the normal n_i and \tilde{n}_θ is the gradient of the network. The gradient is only calculated in three dimensions:

$$\tilde{n}_\theta(x_i, q_i) = \nabla_{x_i} \tilde{d}_\theta(x_i, q_i) = \frac{\partial \tilde{d}_\theta(x_i, q_i)}{\partial x_i}. \quad (4.21)$$

And \tilde{N}_θ is the null-space of \tilde{n}_θ . This loss term expresses: The normal direction of the training point is orthogonal to the tangent plane of the distance field, and the normal direction of the distance field is orthogonal to the null space of the normal direction of the training point. The last item of loss function is a regularization term, where γ is a regularization coefficient. Here we use the value of 0.02 for regularization coefficient. This item makes the transition position as close to the object as possible. Otherwise, without this item, the transition region will be pushed particularly far, which causes overfitting.

Through several trials, the Adam optimizer was finally chosen. The learning rate is 10^{-4} . Batch size is 4096 for dynamic models and 2048 for static models. The result of training and comparison with baseline can be find in 6.1.

5. Application of Geometric Constrains

This chapter concentrates on the two applications of ReDSDF: reactive motion generation and SafeRL. The reactive motion generation will be discussed in Section 5.1 and SafeRL will be discussed in Section 5.2. The SafeRL method will be divided into three sub-section for detailed introduction.

5.1. Robot Motion Control with ReDSDF

ReDSDF can be used for real-time control of robots to avoid collisions and it can be integrated to any other type of reactive motion generation, such as Riemannian Motion Policies (RMP) [6] and Composable Energy Policies (CEP) [21]. Here the whole-body motion control based on Artificial Potential Fields (APF) [20] will be used. The controlled robot is TIAGo++. As introduced in Section 3.2, ReDSDF as distance field can be directly constructed as repulsive field in APF.

5.1.1. Construction of Artificial Potential Fields

First of all, we define a PID controller to control the robot end-effector to target position. Further on this basis, we define a velocity signal determined according to the direction of the gradient of the distance field. This velocity signal can help the robot to avoid the obstacles. Since the distance field is a neural network, we can directly find the gradient of the distance field. Before doing robot control, we will define some points on the robot that need to be queried for collision. When any of these points is less than the threshold

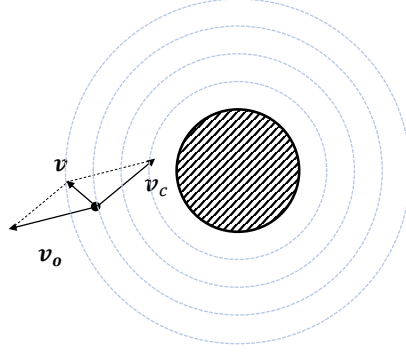


Figure 5.1.: Explanation of the Repulsive Velocity

κ , the control of obstacle avoidance will be turned on. For each obstacle, the energy for obstacle avoidance is defined as follows:

$$E_o(\mathbf{x}) = \begin{cases} 0 & \tilde{d}_{\theta}(\mathbf{x}, \mathbf{q}_o) > \kappa \\ \frac{\bar{v}}{2\kappa} \left(\tilde{d}_{\theta}(\mathbf{x}, \mathbf{q}_o) - \kappa \right)^2 & 0 \leq \tilde{d}_{\theta}(\mathbf{x}, \mathbf{q}_o) \leq \kappa \end{cases}, \quad (5.22)$$

where \bar{v} is the repulsive velocity coefficient. This is the repulsive field in APF. From this we can obtain the velocity of avoiding the obstacle by calculate the gradient of the field:

$$\mathbf{v}_o = -\nabla_{\mathbf{x}} E_o(\mathbf{x}) = \frac{\bar{v}}{\kappa} \left(\tilde{d}_{\theta}(\mathbf{x}, \mathbf{q}_o) - \kappa \right) \tilde{\mathbf{n}}_{\theta}(\mathbf{x}, \mathbf{q}_o). \quad (5.23)$$

The Figure 5.1 shows, how the repulsive velocity works. \mathbf{v}_c is the original control velocity. In this example, Only one point needs to be controlled. If the point is close to the obstacle, a repulsive velocity in the same direction as the distance field gradient will be superimposed on the original control velocity. The resulted velocity will naturally dodge obstacles.

If there are multiple points on a control object that require obstacle avoidance, we call these points: Points of Interest (PoI). If there are multiple obstacles \mathcal{O} in the environment, the repulsive velocity can be vector summed for each point in the PoI:

$$\mathbf{v}_i = \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} \mathbf{v}_{i,o}, \quad (5.24)$$

where index i represents the i -th point in the PoI. In the robot control, the velocity in task-space needs to be translated into joint-space with \mathbf{q}_r through Inverse Kinematics (IK).

Recently, there are many ways to achieve IK. In this thesis, the Jacobian transpose method [56] is used:

$$\dot{\mathbf{q}}_{r,i} = \mathbf{J}_i^T(\mathbf{q}_r)\mathbf{v}_i \quad (5.25)$$

Eventually, we can summarize the control velocity and all the repulsive velocity in the robot joint-space:

$$\dot{\mathbf{q}} = \mathbf{J}_c^T(\mathbf{q})\mathbf{v}_c + \sum_i^N \dot{\mathbf{q}}_{r,i}, \quad (5.26)$$

where N is the number of PoI and \mathbf{J}_c is the Jacobian matrix calculated in the end-effector frame. The experiments and results on reactive motion generation will be described in detail in Section 6.2.

5.1.2. Applying Distance Field and APF on Robot

The distance field can only query the distance for a certain point. If we want to control the whole robot for obstacle avoidance, we must select some points on the critical area of the robot surface for calculation. These points are mentioned in Section 5.1.1 and called PoI. We have here an example of taking PoI on TIAGo. The example is shown in the middle of Figure 5.2. In this figure, the green points are PoI. The example experiment is a self-collision prevention experiment with only one robot controlling the whole body, so in robot work, only the arm can collide with the rest part of the robot. Therefore, the critical area of the robot is the arm surface, and all the PoI are taken from the arm surface. The arm will only collide with the other arm, and the robot is symmetric, so we only need to

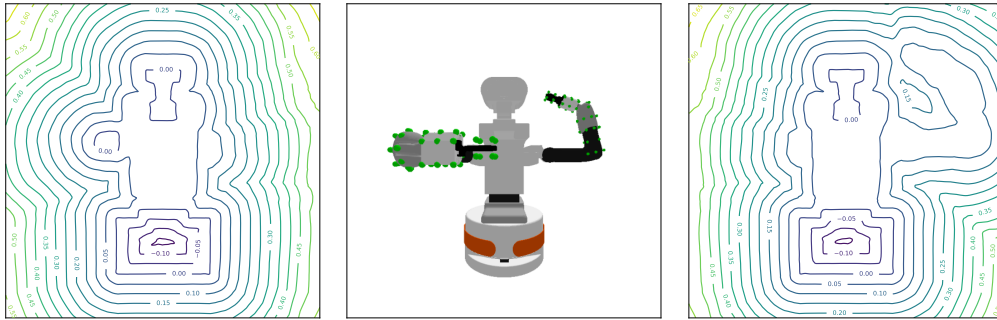


Figure 5.2.: PoI of Robot and its Distance Field with Mirroring

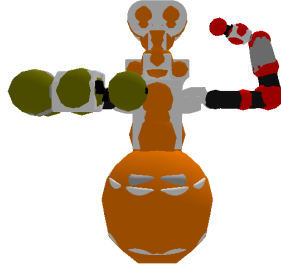


Figure 5.3.: Sphere-Based Method

train the simplified robot with only a single arm as shown in Figure 4.6. While we only train distance field of robot with left arm, we can calculate the distance field of the right arm by mirroring. Left and right image of Figure 5.2 shows the distance fields with left arm and right arm respectively. In this example, the PoI on the right arm only needs to query the distance field with the left arm, and similarly, the PoI on the left arm only needs to query the distance field with the right arm. For other robots, the this approach can also be used to train distance fields and define poi in critical surface for obstacle avoidance control.

There is also a popular method of modeling objects with spheres [9]. In the control of the robot, the relationship between the position of each part of the robot is calculated by finding the distance between the two centers of the spheres. The method is shown as an example of TIAGo in Figure 5.3, where different groups of spheres are shown in different colors. Each sphere only needs to query the distance from other groups of spheres. These two methods will be compared with each other in Section 6.2 and Section 6.4.2.

5.2. Safe Exploration with Learned Constraints

Now that we have the distance field, we'll go into detail on how to apply the RedSDF to SafeExp as constraints. In Section 3.4, we introduced Atacom, which is the theoretical premise for SafeExp in this thesis. The method views the constraint as a manifold and

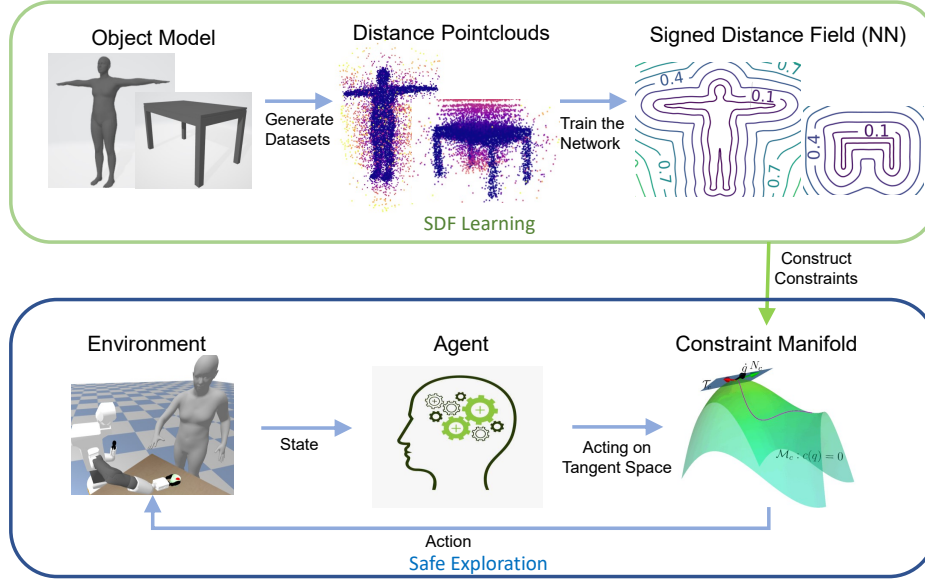


Figure 5.4.: Pipeline of SafeRL

all exploration is performed on the manifold. Discussed in this section is the SafeExp of robots in RL. Therefore, the problem we should be concerned now is how to translate the task-space manifold into the joint-space of robot. Here we use the state constraints in ATACOM and the control value of robot is joint velocity.

5.2.1. Pipeline of Safe Reinforcement Learning

The aim of SafeRL is to allow robots to achieve SafeExp without breaking task and geometric constraints in the process of learning. Then SafeExp can be summarized into two main problems: how to define the constraints and how to calculate the action so that the state of the MDP is always on the constraint.

The task-oriented constraints are easy to define, such as joint limits that can be defined directly. But for the geometric constraints, it is complex. As introduced in Section 3.4, the approach to SafeRL applied in this case is to define collision-free region through implicit equations, while real-world objects are represented by SDF and converted into constraints.

So the main idea to construct the geometric constraints is training the neural network as SDF of object. The object can be either dynamic or static. The datasets used to train the network are generated from the object-models. Once we have the constraints, we can use ATACOM, mentioned in Section 3.2, to project the actions onto the manifold. The pipeline of SafeExp can be found in Figure 5.4.

The top half of the figure shows the process of learning the geometric constraints. This part was discussed in Chapter 4. Once we have the SDF, we can define the constraints according to the corresponding safe distance to allow the robot to avoid collisions. The derivation of constraints and Jacobian will be described in detail in the next two sub-sections. The actions drawn by the agent will be projected into the tangent space by means of ATACOM, and then applied to the environment, so that all states at the time of exploration can be satisfied with safety requirements. It is worth mentioning that these constraints can also be dynamic constraints. RL experiments for dynamic scenes will be presented in the Section 6.3.

5.2.2. Applying Constraints on Robots to Avoid Collisions

The method of obstacle avoidance for the robot is to keep a sufficient distance between the robot and the obstacle. However, when the robot is close to an obstacle, the shape of the robot cannot be ignored. In this case, the distance to the obstacle is no longer reliable if we use the robot's center point to calculate. From this, we proposed two ways to keep the

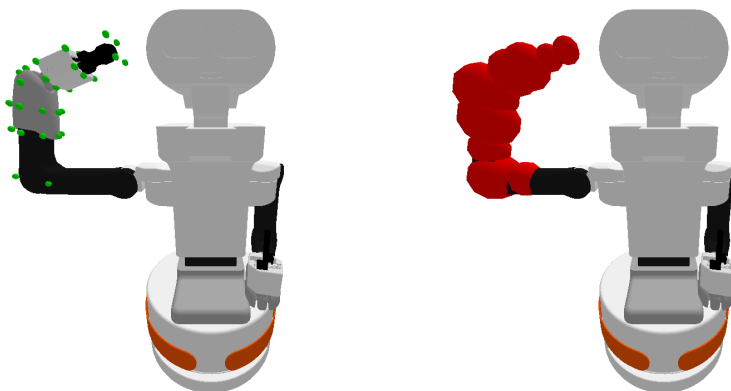


Figure 5.5.: Two Methods to Define the Constraints on Robot

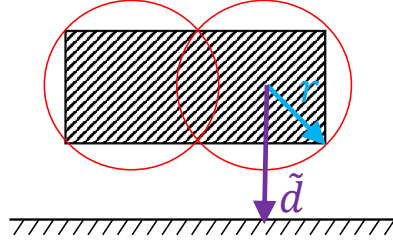


Figure 5.6.: Using Spheres to Define Constraints

robot at a safe distance from the obstacle. These two methods are shown in the Figure 5.5. The first method is to define some query points on critical parts of the robot surface. This method is same as PoI. We will constrain these points to maintain a sufficient distance from the obstacle. By applying the distance network, we can express the constraints by the following equation:

$$c_i(\mathbf{q}) = \delta_i - \tilde{d}(\mathbf{x}_{p,i}(\mathbf{q}), \mathbf{q}_o) \leq 0, \quad \forall i \in (0, 1, \dots, N) \quad (5.27)$$

where $\mathbf{x}_{p,i}(\mathbf{q})$ is forward kinematics to compute the position of query points, $\tilde{d}(\cdot)$ is the function of ReDSDF, which output the distance to the obstacle surface, the index i represents the point in PoI, and δ is the minimal distance that the robot should keep from the obstacle. However, the number of constraints with this method is same as the number of PoI, and when there are especially many points, the computation slows down. What's more, these points do not wrap the entire robot arm. Therefore we propose the method shown on the right side of the Figure 5.5.

The idea of this method to define the constraints is using spheres to wrap the robot and query the center position of spheres. The illustration can be seen in the Figure 5.6. Here we constrain the distance between the position of the center of the sphere and the surface of the obstacle. The position of the sphere's center can be determined by forward kinematics. By subtracting the radius of the ball from the distance obtained we can obtain the distance to be constrained:

$$g_i(\mathbf{q}) = \delta_i - \tilde{d}(\mathbf{x}_{p,i}(\mathbf{q}), \mathbf{q}_o) + r_i \leq 0, \quad \forall i \in (0, 1, \dots, N) \quad (5.28)$$

where r_i is the radius of the constraint sphere. And we use forward kinematics $\mathbf{x}_{p,i}(\mathbf{q})$ to calculate the position of sphere center on the robot arm. We thus constrain the position of these points to allow the robot to avoid collisions. If we use spheres as constraints, the number of spheres is much smaller than the number of PoI in the previous method. Therefore, this method is more computationally efficient than PoI.

5.2.3. Define Actions on the Constraints

Now that we have the constraint, we need to convert the constraint to the manifold in joint-space. According to the ATACOM introduced in Section 3.4, we can convert the inequality constraint of Equation 5.28 into an equation constraint by using slack variable:

$$c_i(\mathbf{q}, \mu_i) = \delta_i - \tilde{d}_i(\mathbf{x}_{p,i}(\mathbf{q}), \mathbf{q}_o) + r_i + \epsilon_i(\mu_i) = 0, \quad \forall i \in (0, 1, \dots, N). \quad (5.29)$$

The control of the robot is carried out in discrete time. So we assume that the obstacle is stationary in one time step. So we derive Equation 5.29 for time:

$$\dot{c}_i(\mathbf{q}, \dot{\mathbf{q}}, \mu_i, \dot{\mu}_i) = -\tilde{\mathbf{n}}_i(\mathbf{x}_{p,i}(\mathbf{q}), \mathbf{q}_o) \mathbf{J}_{FK}(\mathbf{q}) \dot{\mathbf{q}} + \epsilon'_i(\mu_i) \dot{\mu}_i. \quad (5.30)$$

Assume that the state of the environment doesn't break the constraints at the previous moment, so $\dot{c}_i(\mathbf{q}, \dot{\mathbf{q}}, \mu_i, \dot{\mu}_i)$ should be equal to 0. And then we write the formula in the form of a matrix:

$$\dot{\mathbf{c}}(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\mu}, \dot{\boldsymbol{\mu}}) = \begin{bmatrix} -\tilde{\mathbf{n}}_1(\mathbf{x}_{p,1}(\mathbf{q}), \mathbf{q}_o) \mathbf{J}_{FK}(\mathbf{q}) & \epsilon'_1(\mu_1) & 0 & \cdots & 0 \\ -\tilde{\mathbf{n}}_2(\mathbf{x}_{p,2}(\mathbf{q}), \mathbf{q}_o) \mathbf{J}_{FK}(\mathbf{q}) & 0 & \epsilon'_2(\mu_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\tilde{\mathbf{n}}_N(\mathbf{x}_{p,N}(\mathbf{q}), \mathbf{q}_o) \mathbf{J}_{FK}(\mathbf{q}) & 0 & 0 & \cdots & \epsilon'_N(\mu_N) \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix}, \quad (5.31)$$

$$\dot{\mathbf{c}}(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\mu}, \dot{\boldsymbol{\mu}}) = \mathbf{J}_c(\mathbf{q}, \boldsymbol{\mu}) \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\boldsymbol{\mu}} \end{bmatrix} = \mathbf{0}. \quad (5.32)$$

In order to solve the Equation 5.32, we need to obtain the null-space of \mathbf{J}_c . There are now many ways to solve for the null-space of a matrix, such as SVD [57] and QR [58].

According to the idea in [59] that the null space bases generated by the projection-based method has continuity, the projected null space of \mathbf{J}_c is determined by:

$$\mathbf{N}_c = (\mathbf{I} - \mathbf{J}_c^T (\mathbf{J}_c \mathbf{J}_c^T)^{-1} \mathbf{J}_c) \mathbf{Z}, \quad (5.33)$$

where the $\mathbf{Z} = [\mathbf{I}_n \ \mathbf{0}]^\top$ is the augmented bases that combines the original joint space bases with a zero matrix. With zero-space of Jacobian, we can project the original action onto the manifold and solve the Equation 5.32:

$$\begin{bmatrix} \dot{q} \\ \dot{\mu} \end{bmatrix} = \mathbf{N}_c \alpha, \quad (5.34)$$

where α is the original action that draw from the network in RL. The above is the theoretical part of the SafeExp. Some experiments and results of RL will be shown in the next chapter.

6. Experiments and Results

This chapter presents some of the designed experiments and results. These experiments not only validate the performance of our proposed method, but also validate the quality of ReDSDF at the same time. First in Section 6.1 some training results of ReDSDF will be shown. In this section, not only static objects are shown here, but also dynamic articulated objects such as human body. The Section 6.2 introduces the experiment that robot controls the whole body to avoid self-collision. In Section 6.3, we use SafeExp proposed in this thesis to make the robot arm grasp specific points. In the environment, there is a shelf and the learned policy will be transformed on the real robot. The learned geometric constraints will be used for reactive motion generation and RL in human-robot interaction environment in Section 6.4.

6.1. Results of ReDSDF

In this section, the qualitative and quantitative results of ReDSDF will be shown. There are two main dynamic models trained in this thesis: human body and TIAGo++. Some comparisons between ReDSDF and other baselines will be made at the end of this section. The training processes can be found in Appendix A.

6.1.1. Results of Human Model

In training the human model, we use the human pelvis as the base position; the top of the person is the positive y-axis direction; the front is the positive z-axis direction; and the left is the positive x-axis direction. The movement and rotation of the base are not considered in the network. Removal of palm joints, the human pose has in total 63 dimensions. Compared to training other objects, the dataset of human body is huge, so it takes a long

Training time	3d 1h 51m 38s
Number of Epochs	189
Minimal loss for training set	0.04597
Minimal loss for validation set	0.05164
Eventual loss for test set	0.05433
RMSE of SDF	0.00499

Table 6.1.: Quantitative Results of Human Model

time to train the human model. The quantitative results of the training are shown in Table 6.1.

The dataset used to calculate RMSE here is a series of actions in AMASS that were not used to generate points for any previous dataset. Here the RMSE is calculated as a comparison between the distance output by the network and the real distance. The formula for calculating RMSE is as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{D}_{test}|} \sum_{i \in \mathcal{D}_{test}} \left(\tilde{d}_{\theta}(\mathbf{x}_i, \mathbf{q}_i) - d_i \right)^2}, \quad (6.35)$$

where d_i is the real minimal distance between object surface and the point \mathbf{x}_i . From the results, we can see that the errors of the network are in the level of 10^{-3} , so the network is able to estimate the distance precisely.

Next, the results will be shown qualitatively. The method of visualizing the SDF is the ray-marching algorithm [60]. This algorithm is widely used in graphics for rendering. This algorithm is also used in this thesis to visualize surface at specific distances. Figure 6.1 shows the visualization of human in four different poses. These four postures are just examples. The network can show all human actions in normal postures. The images in the first row show the position at a distance of 0 (0-level) from the field using the Ray-marching algorithm. The shape displayed in this position should be the same as the original object. The picture in the second row show the contour of the distance field. This contour shows only the cross section in the xy-plane at the position where the z-value is 0. The third row of images shows the original object. From the figure, it can be seen that the network can represent different poses of the same object, and it can also represent

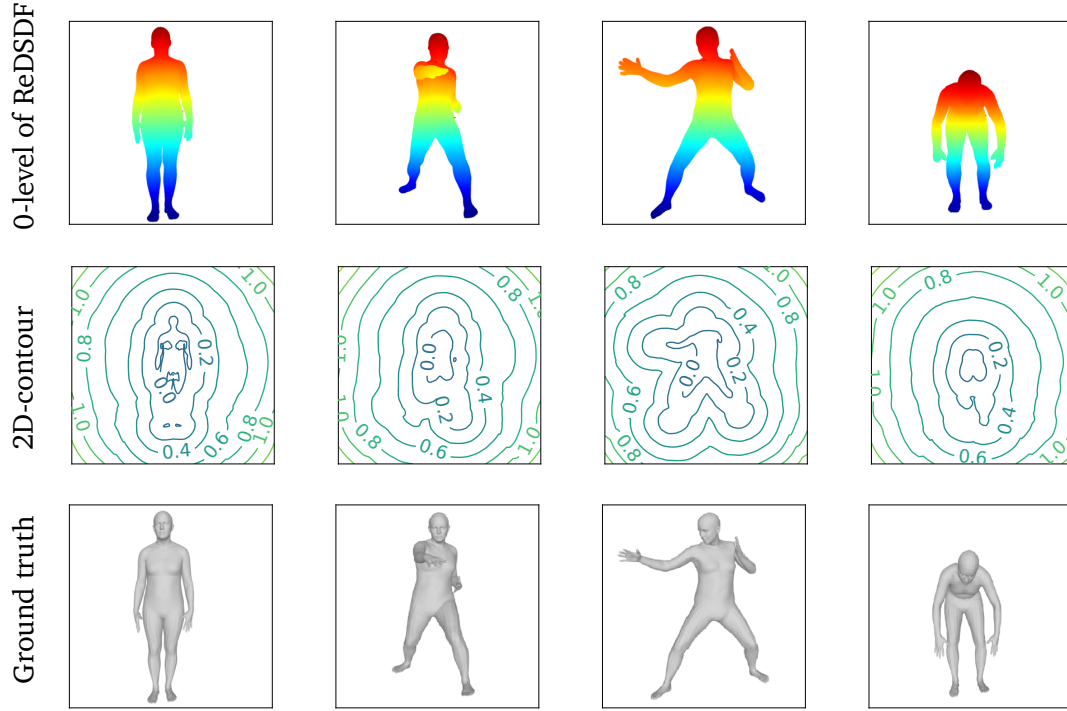


Figure 6.1.: Visualization of Human Model

objects with high-dimensional poses like the human body. The human body reconstructed by the network is almost identical to the ground truth. And many details on the human body are also expressed, such as fingers. Since this network is all capable of expressing a dynamic model of the human body in such large pose dimensions, this also means that ReDSDF can express other dynamic objects with high joint dimensions.

The above comparison is the different pose of the human body, and the following will compare the different levels with the same pose. Figure 6.2 shows the result of ReDSDF with different output value. The levels in the figure of the four shapes in the network are: 0, 0.05, 0.2 and 0.7. It can also be seen from the figure that the network can give good results for different values of distances. For close areas, the network can portray the details of the human body, for distant areas, the network can roughly estimate the distance to the human body. And as the distance gets larger, the shape presented gradually approaches a sphere. This is consistent with the expression of Equation 4.14.

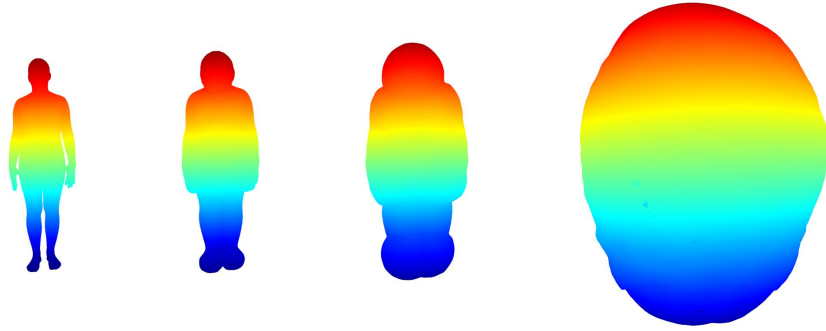


Figure 6.2.: Human Models with Different Levels

6.1.2. Results of TIAGo with Single Arm

For training the robot, the origin of the SDF is the center of the robot base and z-axis facing up, x-axis pointing to the front of the robot, y-axis pointing to the left of the robot. For a single arm, the robot was trained with a total of 8 joints. We randomly selected 10,000 poses to generate the dataset. The quantitative results of the training are shown in Table 6.2.

Training the robot took a relatively short time. And the training results are shown in Figure 6.3 qualitatively. The figure shows four different configurations of TIAGo. The 3D-reconstruction of robot and 2D-contour are visualized in the first and second rows in the figure. It can be seen that ReDSDF can express the configurations in almost all joint limits, and can give the distance to the robot surface with high quality. The model will

Training time	10h 11m 35s
Number of Epochs	58
Minimal loss for training set	0.01971
Minimal loss for validation set	0.01991
Eventual loss for test set	0.04258
RMSE of SDF	0.00414

Table 6.2.: Quantitative Results of TIAGo++

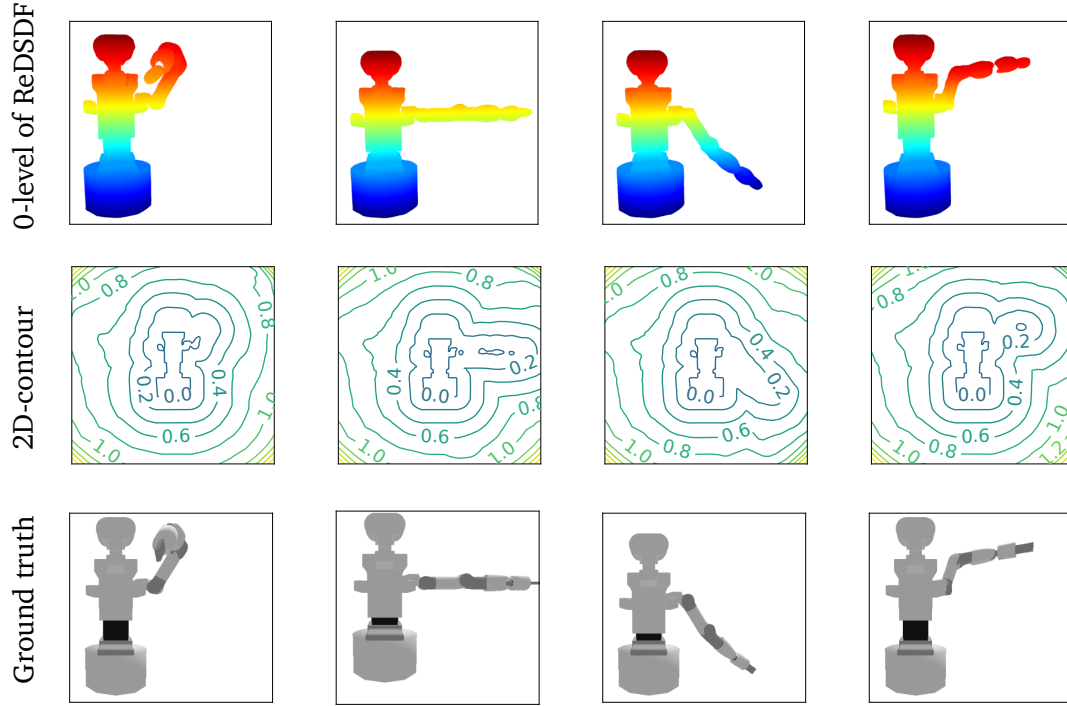


Figure 6.3.: Visualization of TIAGo Model

be used to do self-collision avoidance. In addition to these two dynamic models, we also trained many static models.

6.1.3. Results of Static Models

The static models do not have the input dimension of the poses. So they only have three input dimension of position. In this section, the results of static models will be shown. And these models will be compared with two baselines: DeepSDF [27] and ECoMaNN [28]. We use the datasets from ShapeNet [61], and train two models: table and sofa. In addition, a static human body will be also trained through the baseline methods and compared. The ReDSDF used here is a network with four hidden layer. And each hidden layer has 512 elements. For the baseline approaches, we use the original method proposed in the literature to generate training data. The quantitative comparison of the results can be seen in the Table 6.3. This table gives a comparison of RMSE between the three objects

RMSE \ Object	Network	table	sofa	human
	ECoMaNN	0.02423	0.01485	0.00832
	DeepSDF	0.00632	0.00749	0.00592
	ReDSDF	0.00503	0.00773	0.00499

Table 6.3.: Quantitative Comparison of the Results of Static Models

and the three networks. From the table we can see that the error difference between DeepSDF and ReDSDF is not significant. But the error of ECoMaNN is relatively large. This difference can also be seen from the qualitative comparison in Figure 6.4. The surface of the object is reconstructed by ray-marching algorithm. According to this figure, we can see that the difference of reconstructed surface quality between DeepSDF and ReDSDF is not significant. However, the reconstructed surface by ECoMaNN is not flat. Further, we can compare the details from the Figure 6.5, 6.6 and 6.7.

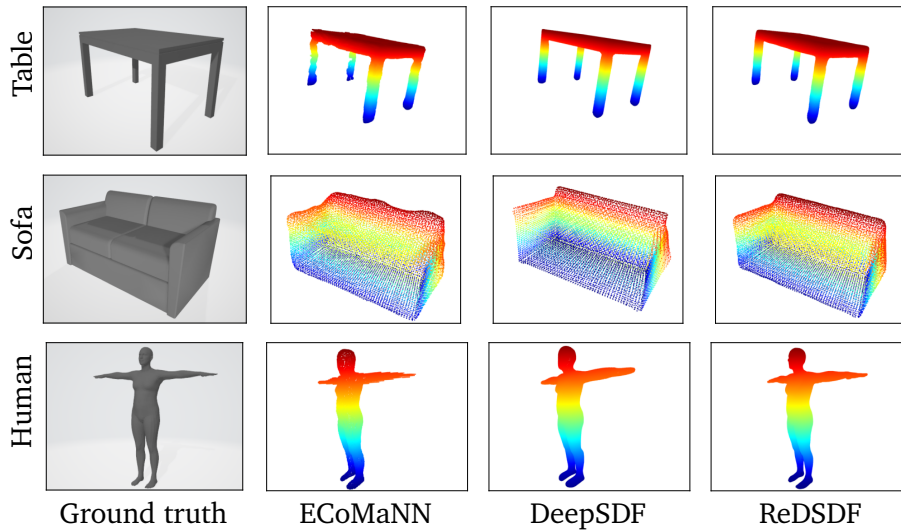


Figure 6.4.: Comparison of the Static Models with Baselines

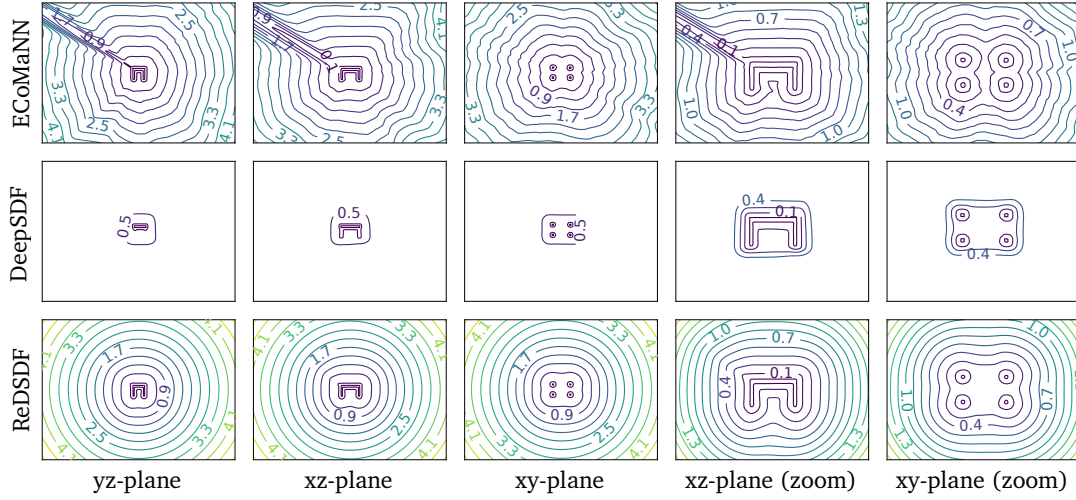


Figure 6.5.: Comparison of the Table Distance Functions in Detail

Figure 6.5 shows the distance field of table. To better show the area near the surface of the object, the field is zoomed in on two planes: xz-plane and xy-plane. As we can see, the distance field learned by ECoMaNN is not very smooth and there are some outliers in the field. This is due to the outliers in dataset and missing of training data in the distant region. DeepSDF can only estimate the distance near the object, while for distant distances can only be obtained by truncation. ReDSDF not only computes the near distance with the same high quality as DeepSDF, but also gives a rough estimate of the distant distance. And at any position in space, ReDSDF gives the accurate normal direction of the distance field. This feature is critical in the application of distance fields to robot control.

Figure 6.6 shows the distance field of sofa. This field is zoomed in on yz- and xy-plane. From the figure can be seen that the contours of ReDSDF distribute nicely. Similarly, ECoMaNN cannot give a smooth distance field. Although DeepSDF can estimate the distance near the surface of an object very well, the distance cannot be estimated in distant region. ReDSDF has the best performance, which can calculate smooth distance fields at any scale.

The comparison of human model is shown in Figure 6.7. For the distance field computed through ECoMaNN, the distance estimated for locations farther from the surface of the object is no longer meaningful. All the other methods can only roughly reconstruct the human body, but ReDSDF can reconstruct details, such as the face and hands. This is due

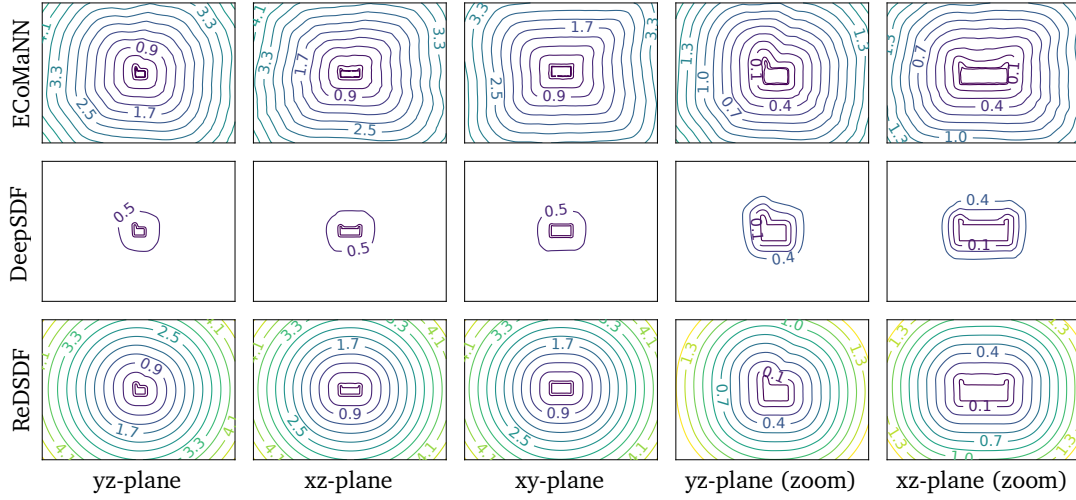


Figure 6.6.: Comparison of the Sofa Distance Functions in Detail

to the fact that we use weights and uneven expansion of augmented data to enhance the training of detailed positions.

To sum up, ReDSDF compensates for the shortcoming that ECoMaNN cannot learn complex manifolds on the one hand, and DeepSDF cannot express distances at a distant region on the other hand. We use the Sigmoid-function to regularize the output of the network, so that not only can the network estimate the distance at a distant region, where the augmented data can't reach, but also the calculated distance field becomes smooth. So we don't need to generate augmented data in the whole space, and the network can express distance anywhere. For the purpose that the network can adapt the boundary between near area and distant region, we use the learnable parameters and limit their scope, so that the network can be used to learn many different models. In order to make the distance field near the object surface more accurate, we produce more augmented points in the area near the object surface. What's more, we assign more weight to the training points at locations with fewer augmented points. Therefore, the network can reconstruct more details on the surface of the object. ReDSDF can also calculate the normal direction of the distance field more precisely by computing the gradient, because we put the normal direction into the training set. During training, the normal direction is also added to the loss function as a metric for training the network.

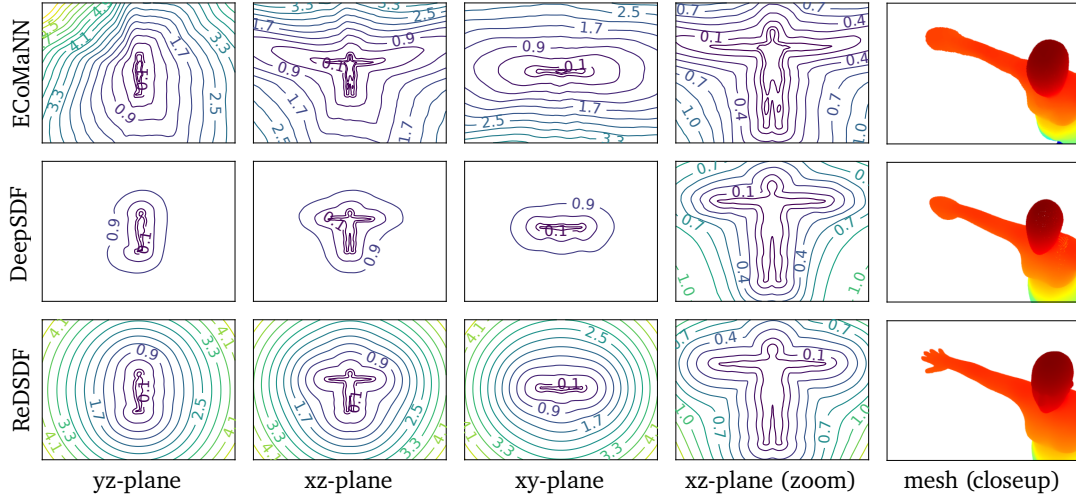


Figure 6.7.: Comparison of the Human Distance Functions in Detail

6.2. Whole Body Control

In this section, we will describe the experiment of whole body control. The learned ReDSDF is used in reactive motion generation to control robot. This method was mentioned in Section 5.1 and the distance model was trained in Section 6.1.2. This experiment was conducted in a virtual environment and the robot was TIAGo++. We control the robot's joints velocity on whole body with a total of 15 degrees of freedom to make the end-effector reach the target point. The 15 degrees of freedom include the joints of both arms and torso. The robot base is fixed.

The method to control robot was mentioned in Section 5.1. In this experiment, we take a total of 124 points on the robot as PoI (of which 62 points per arm). Because the distance field was only trained for the robot left arm, we use mirroring when querying the distance of the right arm. The aim of this experiment is to achieve self-collision avoidance by querying the distance of these points. Again, we did another experiment as a baseline. This experiment uses the sphere to model the robot as mentioned in Section 5.1.2, and achieves obstacle avoidance by querying the distance between the centers of each sphere. There are in total 30 spheres and these spheres are grouped into 3 subgroups: body with 18 spheres, left and right arm with 6 spheres respectively. When calculating the distance, the spheres in each subgroup perform a distance query on the spheres in the

	No avoidance	Sphere-based (task-oriented)	ReDSDF (task-oriented)	Sphere-based (cautious)	ReDSDF (cautious)
Success rate	44.8%	80.9%	82.8%	38.3%	72.9%
# collisions	548/1000	49/1000	7/1000	0/1000	0/1000
Final err. (cm)	10.42 \pm 0.59	1.42 \pm 0.14	1.74 \pm 0.15	7.21 \pm 0.42	2.05 \pm 0.17
Reach time (s)	6.17 \pm 0.42	12.58 \pm 0.58	13.95 \pm 0.58	23.98 \pm 0.55	15.86 \pm 0.63
Smoothness	10.50 \pm 0.06	811.88 \pm 183.40	12.20 \pm 0.25	3557.51 \pm 284.15	13.10 \pm 0.28
C. time (ms)	0.10 \pm 5.49e-5	5.79 \pm 6.28e-3	2.56 \pm 2.45e-3	5.97 \pm 5.68e-3	2.61 \pm 2.42e-3

Table 6.4.: Results of Whole Body Control Experiment

other subgroups. By adjusting κ and \bar{v} we achieve two control modes: task-oriented and cautious. For the task-oriented control, the robot will tend to complete tasks at the expense of obstacle avoidance, and conversely, the robot will prefer obstacle avoidance in cautious control mode. Figure 6.8 is a screenshot of experiment, which demonstrates how the experiment was conducted.

A total of 1,000 experiments were performed for each method and each control mode. Two targets in the robot workspace will be randomly selected for each experiment. These targets will not be too close to the surface of the robot,

and the robot is controlled to bring the two end-effector to that targets. When the both end-effectors is less than 3 cm from the target, it is marked as successful. If the robot collides, the environment will be reset for next experiment. Each experiment is conducted for a maximum of 30 seconds and the simulated environment is controlled at 60 Hz. The results of the experiments can be found in Table 6.4. We finally report these items: success rate, number of collisions, final error, time spent for reaching target, smoothness of the generated trajectory and time spent for querying distance. The smoothness of trajectory τ is defined by the following equation:

$$s(\tau) = \sum_{\mathbf{q}_r(i) \in \tau} \|\ddot{\mathbf{q}}_r(i)\|^2, \quad (6.36)$$

where $\mathbf{q}_r(i)$ is the robot joint position in i -th time step. The simulation was performed on an AMD Ryzen 7 3700X 8-Core Processor with a graphics card of GeForce RTX 2080 SUPER. We used the graphics card to speed up the query distance process.

From the results, it can be seen that if no obstacle avoidance measures are used, the robot will have multiple collisions and the success rate is very low. Comparing the two methods, ReDSDF has a higher success rate than sphere-based, regardless of the control mode. Although the final error of sphere-based is a little smaller in task-oriented control mode, the number of collisions is much higher than that of ReDSDF. In cautious control

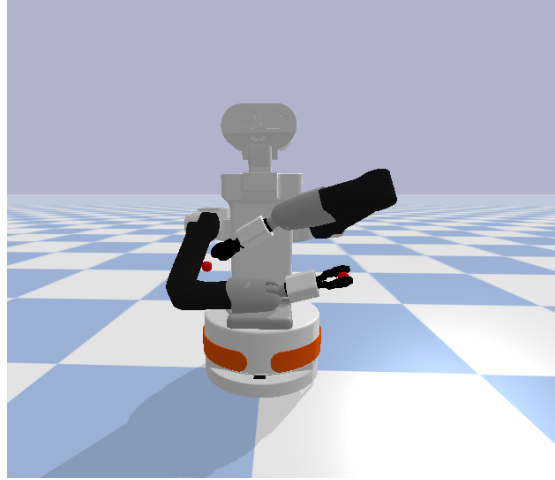


Figure 6.8.: Experiment Screenshot of Whole Body Control

mode, both methods achieve 0 collisions. However, the success rate of ReDSDF is much higher than sphere-based, and the error is smaller than sphere-based. What's more, the trajectory generated by ReDSDF is smoother than by sphere-based and the computing time of RedSDF is shorter than sphere-based. In general, ReDSDF has better control performance than sphere-based, both for obstacle avoidance and reaching the target.

6.3. Reaching Target Point with a Shelf

This experiment is a RL experiment. The experiment includes a robot TIAGo++ and a shelf. The robot learns to bring the end-effector to one specified position on the shelf in a simulation environment. While reaching the target point, the robot has to avoid collision with the shelf and itself. After learning the policy, we transform the agent onto the robot in real world, and then observe whether the robot can achieve the specified function in the real world. The scene of experiment can be seen in Figure 6.9 with the simulated world on the left and the real world on the right.

In this experiment, we use the algorithm SAC [52] to learn the policy. And the method of SafeExp mentioned in Section 5.2 and the robot distance model trained in Section 6.1.2 were used. What's more, we also trained a distance model for the shelf as constrains. We also use joints limitation and ground as the constrains. And the method will be compared

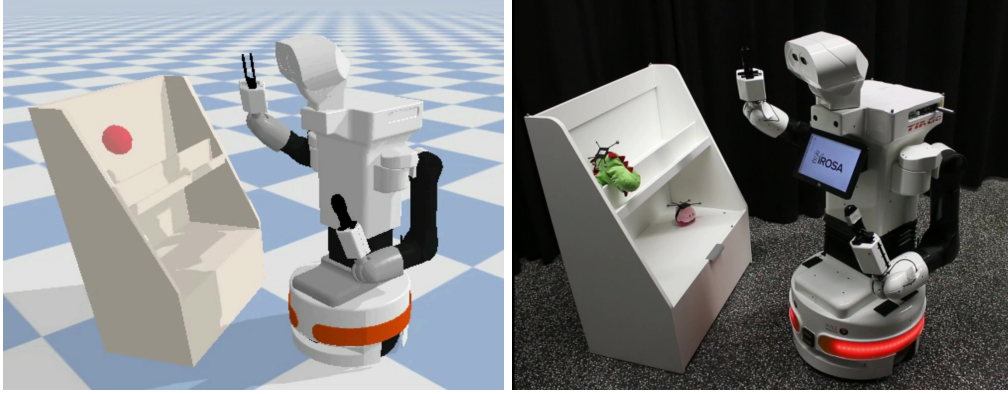


Figure 6.9.: Demonstration of Robot-Shelf Experiment

with the baselines: no constraint and safe-explorer [48]. We control the velocity of robot joint. The degrees of freedom of the robot, in other words, the dimension of the action space is 7, which includes the seven joints of the robot's right arm. The base of robot is fixed. The dimension of observation space is 10, which includes the state of joints and the position of target. During the training, the position of target was selected on the shelf randomly. We performed every experiment with 10 random seeds. The hyperparameters of the training and the training process can be seen in the Appendix A. The final experimental results can be seen in Table 6.5 and Figure A.2.

From the results, it can be seen that only ATACOM can learn the expected results among the three methods. The final learned policy can keep the error below 0.1 mean throughout the episode. Whether it's a task specific constraints or a geometric constraints, the robot barely breaks them by using our method during training. Even zeros collisions are possible in some random seeds. If no SafeExp method is used, the robot will move away from the target point in order to avoid collisions, which causes the robot not to learn the expected policy. This conclusion can be seen in the first column of the table with trivial SAC. If we use safe explorer, the number of collisions will be reduced accordingly. But still the expected strategy cannot be learned. The learned policy also shows good results when we transfer the experiment to the real robot. In the real world, robots can also avoid collisions while grasping objects at target points.

	Trivial SAC	Safe Explorer	ATACOM
Average error to target	0.7682	0.6034	0.09564
Minimal error to target	0.7472	0.5263	0.08251
Average cumulative reward	-525.5	-493.2	1488
Maximal cumulative reward	-280	-450.7	1682
Average discounted reward	-213.2	-190.2	455.4
Maximal discounted reward	-126.2	-177.5	535.2
Average collision during learning	3527.5	1417.6	3.4
Minimal collision during learning	2325	1100	0
Average breaking joint constraints	313.2	212.3	0
Minimal breaking joint constraints	261	174	0
Average maximal value of constraints	/	0.08737	0.01818
Minimal maximal value of constraints	/	0.07923	0.01359
Average number of steps	331.5	428.9	499.8
Maximal number of steps	340.8	443.8	500

Table 6.5.: Results of RL-Experiment with Shelf

6.4. Human Robot Interaction

In this section, two experiments on human-robot interaction will be presented. These experiments were designed to demonstrate that the reactive motion generation and safeRL approaches proposed in this thesis can also be applied in dynamic environments. In the first experiment, a human and a robot work in a shared workspace, where the robot is able to accomplish its own task while avoiding collisions with the human. The second experiment applies SafeExp to a human-robot interaction environment.

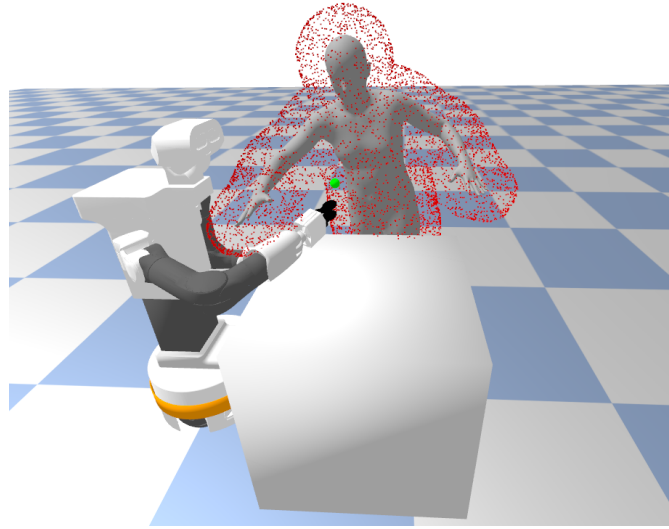


Figure 6.10.: Shared Human-Robot Workspace

6.4.1. Reactive Motion Generation in Shared Human-Robot Workspace

The experiment was performed in a simulated environment and used a robot TIAGo++, a human model and a cube. These objects were trained separately as distance networks: the ReDSDF of robot in Section 6.1.2, human body in Section 6.1.1 and a newly trained cube. In this experiment the human is completing his work and the robot's end-effector needs to reach the specified position without collision. It is like a sequential pick-and-place-like actions. Because the human keeps moving, robots must also be able to evade human in the shared workspace. We control joints velocity on the right arm and torso of robot with a total of 8 degrees of freedom. The robot base is fixed and can't move. The scene of this experiment is shown in Figure 6.10. The green dot is the target, the place where the end-effector of robot should arrive. When the end-effector is less than 3 cm from the target, a new target point is refreshed. The red point cloud on the surface of the human body is 10 cm from the surface of the human body generated by the ReDSDF. Once the robot collides with any object, the experiment resets.

In this experiment, we use the same method as whole body control to perform obstacle avoidance, and also compare with two baselines: trivial baseline and sphere-based baseline. Two control modes are also used here: task-oriented and cautious. There are in total

	No avoidance	Sphere-based (task-oriented)	ReDSDF (task-oriented)	Sphere-based (cautious)	ReDSDF (cautious)
# collisions	935/1000	171/1000	95/1000	86/1000	27/1000
# targets	2.172 ± 0.19	6.35 ± 0.18	5.78 ± 0.16	4.44 ± 0.14	4.73 ± 0.16
Smoothness	71.95 ± 0.85	2120.00 ± 198.35	135.56 ± 3.67	2286.69 ± 174.82	624.14 ± 43.50

Table 6.6.: Results of Shared Workspace Experiments

1,000 experiments that have been performed for each method and each control mode, and each experiment will have 9 consecutive target points randomly generated. The control frequency of the robot is 60 Hz. The experiments were performed on the computer with AMD Ryzen 7 3700X 8-Core Processor and GeForce RTX 2080 SUPER graphics card. And we use graphics card to accelerate the process of querying distance. The results of experiment can be found in Table 6.6.

Here we evaluate a total of three items: number of collisions, error of targets and smoothness of trajectory. The smoothness of trajectory is computed through the Equation 6.36. From the results, it can be seen that the number of collisions of ReDSDF is less than other methods regardless of the control mode used. Although the error of reaching the target point is similar, the ReDSDF-controlled trajectory is smoother compared to sphere-based one. In the course of the experiment, the robot was not only able to reach the target and achieve obstacle avoidance, but also actively evade human by using ReDSDF. To sum up, the obstacle avoidance control by using ReDSDF is on the one hand suitable for static scenes, on the other hand for dynamic scenes, such as human-robot interaction.

6.4.2. Reinforcement Learning in Human Robot Interaction Environment

In this section, an experiment on SafeRL in human-robot interaction environment is presented. This experiment is performed in simulation environment. There are four objects in the environment: a robot TIAGO++, a human, a table and a cup. At the beginning of each episode, the robot holds the cup, and then the robot learns to place the cup in a designated location near the human and the water in the cup should not be poured. The human keeps moving and working. The robot needs to avoid collisions with the human, table and itself during the learning process. The scene of this experiment is shown in Figure 6.11. In this figure the red point is the target position where the cup should be placed.

In this experiment, we use the algorithm SAC [52] to learn the policy. And the method of SafeExp mentioned in Section 5.2 and some distance fields trained in Section 6.1 are

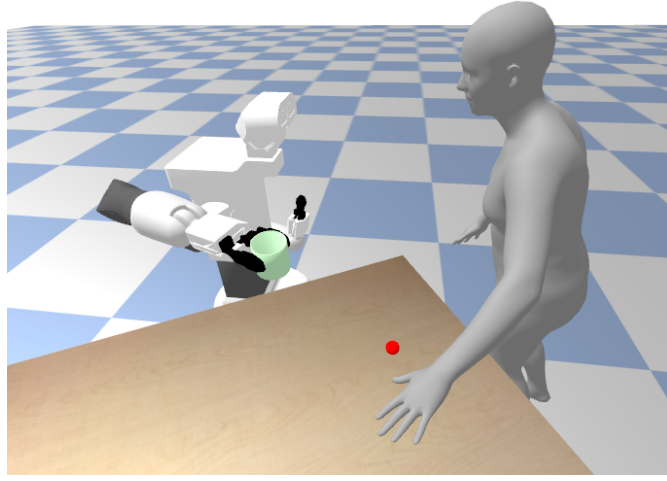


Figure 6.11.: Scene of Human-Robot Interaction

used here. The ReDSDF of human is evaluated in Section 6.1.1, the ReDSDF of robot is evaluated in Section 6.1.2 and the ReDSDF of table is evaluated in Section 6.1.3. These distance fields are used as constraints for collision avoidance. We have also added joint limits and ground positions to the constraints. The method mentioned in Section 5.2 is compared with two baselines: the first is trivial learning without constraints, the second is safe explorer [48]. In the original paper of safe explorer, the constraint must be learned through neural networks before exploration, which causes the problem of inaccuracy of constraints and slow query speed. Here we directly use the trained ReDSDF as a constraint for safe explorer. During the learning, the velocity of robot joint is controlled and the degrees of freedom of the robot is 7. So the dimension of action space is 7, and these 7 joints are all on the right arm of robot. The robot base is fixed and can't move. The dimension of observation space is 81, which includes the state of joints, the position of target, the orientation of cup, the position of human, orientation of human and pose of human. The target position where the cup should be placed is selected randomly. The hyperparameters and some details of training are shown in the Appendix A. We performed every experiment with 10 random seeds.

	Trivial SAC	Safe Explorer	ATACOM
Average error to target	0.3288	0.07887	0.1292
Minimal error to target	0.3819	0.07360	0.08271
Average number of sprinkling	4969.2	130.3	0
Minimal number of sprinkling	585	22	0
Average cumulative reward	1.284	1085	903.5
Maximal cumulative reward	475.4	1204	1014
Average discounted reward	40.98	348.9	267.2
Maximal discounted reward	138	382.6	337.3
Average collision during learning	1561	270.8	6.4
Minimal collision during learning	1146	130	1
Average breaking joint constraints	8	2.4	0
Minimal breaking joint constraints	1	0	0
Average maximal value of constraints	/	0.1301	0.1096
Minimal maximal value of constraints	/	0.1169	0.0836
Average number of steps	452.5	490.9	499.9
Maximal number of steps	465.9	495.5	500

Table 6.7.: Results of RL-Experiment in Human-Robot Interaction

The comparison of results and the learning process can be found in Table 6.7 and Figure A.3. The experimental results were taken as the average of 10 times and the best result of 10 times. In the table, we have considered eight results: error to target, number of sprinkling, cumulative reward, discounted cumulative reward, number of collision during learning, number of breaking joint constraints, maximal value of constraints and average number of steps for each epochs. All eight results are presented by means of 10 random seed experiments, taking the average and optimal values. And the figure shows 6 different items during learning including discounted cumulative reward, cumulative reward, number of collision, number of breaking joint limits, average number of steps in each epoch and

maximal value of constraints. As we can see from the results, although safe explorer has a higher cumulative reward than ATACOM, ATACOM can also learn good results. And ATACOM is able to achieve nearly 0 collision learning. If no safety algorithm is used, the robot will not learn the expected results. In particular, the robot won't break the joint limits if we apply ATACOM. So for safety reasons, our method has better performance.

7. Conclusion

In this chapter, the method proposed in this thesis will be summarized based on the experimental results, and future work will be prospected. The summary and future work will be presented in two separate sections respectively.

7.1. Summary

In this thesis, we propose Regularized Deep Signed Distance Fields (ReDSDF) framework, a novel distance field, which is network and estimates distance with high performance. Furthermore, we propose methods to apply this network to reactive motion control and Reinforcement Learning (RL) to achieve robot obstacle avoidance and Safe Exploration (SafeExp).

For ReDSDF, we propose new methods for generating augmented points and design the architecture of this network. We add the dimension of the pose in the input of network, so ReDSDF can express different postures of arbitrary articulated objects. Since the output of the network is limited by inductive bias in training, the network can estimate the distance to any position in space and the distance field can become smooth. The performance of ReDSDF has been validated by training different dynamic as well as static models. In addition, the quality of network has been compared with two baselines: ECoMaNN and DeepSDF. The distance network can be used not only in computer graphics, but also in robot control and RL. The results of the comparison lead to the conclusion that ReDSDF has better performance. It can not only compensate the disadvantage that DeepSDF can only estimate the distance near the object, but also improve the smoothness of distance field compared with ECoMaNN.

Safety is important for autonomous robots, both when planning actions that considers themselves and the world around them, and when robots interact with humans in shared

workspace. Applying the ReDSDF proposed in this thesis can solve these collision problem. On the one hand, we propose to apply this network to generate reactive motions in robot control to enable the robot to achieve obstacle avoidance. The method has been validated with two different experiments in both static and dynamic environments, and the performance of this method has been compared with baseline. On the other hand, we use this network to implement SafeExp. We constrain the state space of the robot by ReDSDF to achieve obstacle avoidance during the process of learning policy. Also this method is compared with baseline, and the performance of the method has been validated through two different experiments. The method has proven to be applicable not only to static environments, but also to dynamic environments such as human-robot interaction. Applying this method, the robot is able to achieve SafeExp in a static environment without collision. In a dynamic environment, although there are a small number of collisions, the robot will not go to actively hit human and will avoid human.

I also learned a lot from this thesis. Overall, I have gained a deeper understanding of neural network training, robot control, and RL, as well as an improved ability to program and work with others. The specific experience can be summarized as follows: First, machine learning is a discipline that combines theory and practice. Each attempt must be based on theory for continuous attempts to gradually improve the learning results. Second, experiments with robots in the real world differ greatly from those in simulated environments. Experiments in the real world require many factors to be considered to ensure the proper functioning of the robot. Finally, we need to make full use of the computer's resources to complete the task efficiently. If we use a reasonable strategy we can largely reduce the computation time.

7.2. Future Work

This thesis has only partially completed the study of robot safety. For the future work is summarized in the following three main aspects.

First, the estimation of the normal direction of the dataset. The method used in this thesis cannot accurately estimate the normal direction of all points. And looking at the existing techniques, there is no method that can accurately estimate all normal directions of a pointcloud especially for some thin-walled objects. Even though some points with wrong normal estimation are removed by deleting outliers in this thesis, it still has an impact on the training of the network. Therefore, if a method to accurately estimate the normal direction can be studied, the performance of ReDSDF can be greatly improved.

The second is to apply the methods of robot control and safeRL that have been proposed in this thesis to more and more complex environments. The method proposed in this thesis can also be applied in real-world robots. The number of obstacles involved in this thesis is small, so the validation of the stability of the algorithm is not comprehensive enough. Therefore, the next work can apply the method to more robot control and reinforcement learning algorithms.

The final research direction is to find methods to accurately estimate the human pose. Now we can estimate the distance network of the human body from the human pose parameters. So if we want to apply the method proposed in this thesis to real-world human-robot interaction scenarios, we need to obtain the position and pose of the human body through computer vision methods or point tracking methods.

Acknowledgments

The advisors have guided me to a great extent while writing this thesis. I would like to express my sincere gratitude to Puze Liu, Davide Tateo and Georgia Chalvatzaki here. I am also grateful that they give me this opportunity to work with them. During the thesis, they gave me a lot of constructive suggestions. Every time I asked them for advice, they were patient and explained. I also learned a lot about robotics and machine learning through this thesis. My appreciation also extends to PhD student Snehal Jauhri from iROSA Group. He also collaborated with me on some of my thesis work.

In the future, I wish Puze can graduate as a doctor successfully, and the best of luck for Davide and Georgia in their future endeavors.

Bibliography

- [1] Y. Li, “Deep reinforcement learning: An overview,” *arXiv preprint arXiv:1701.07274*, 2017.
- [2] P. Liu, D. Tateo, H. B. Ammar, and J. Peters, “Robot reinforcement learning on the constraint manifold,” in *Proceedings of the 5th Conference on Robot Learning* (A. Faust, D. Hsu, and G. Neumann, eds.), vol. 164 of *Proceedings of Machine Learning Research*, pp. 1357–1366, PMLR, 08–11 Nov 2022.
- [3] J. Corrales, F. Candelas, and F. Torres, “Safe human-robot interaction based on dynamic sphere-swept line bounding volumes,” *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 177–185, 2011.
- [4] M. d. S. Arantes, C. F. M. Toledo, B. C. Williams, and M. Ono, “Collision-free encoding for chance-constrained nonconvex path planning,” *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 433–448, 2019.
- [5] 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, 2009.
- [6] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, “Riemannian motion policies,” *arXiv:1801.02854*, 2018.
- [7] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, and G. Chalvatzaki, “Regularized deep signed distance fields for reactive motion generation,” *arXiv preprint arXiv:2203.04739*, 2022.
- [8] N. Vahrenkamp, H. Arnst, M. Wächter, D. Schiebener, P. Sotiropoulos, M. Kowalik, and T. Asfour, “Workspace analysis for planning human-robot interaction tasks,” in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 1298–1303, 2016.

-
-
- [9] L. Balan and G. M. Bone, “Real-time 3d collision avoidance method for safe human and robot coexistence,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 276–282, 2006.
- [10] P. Papadakis, A. Spalanzani, and C. Laugier, “Social mapping of human-populated environments by implicit function learning,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1701–1706, 2013.
- [11] D. Vogt, S. Stepputtis, S. Grehl, B. Jung, and H. Ben Amor, “A system for learning continuous human-robot interactions from human-human demonstrations,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2882–2889, 2017.
- [12] N. I. Badler, C. B. Phillips, and B. L. Webber, *Simulating humans: computer graphics animation and control*. Oxford University Press, 1993.
- [13] D. Kee and W. Karwowski, “Analytically derived three-dimensional reach volumes based on multijoint movements,” *Human factors*, vol. 44, no. 4, pp. 530–544, 2002.
- [14] H. Jiang, J. Cai, and J. Zheng, “Skeleton-aware 3d human shape reconstruction from point clouds,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5431–5441, 2019.
- [15] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, “Smpl: A skinned multi-person linear model,” *ACM transactions on graphics (TOG)*, vol. 34, no. 6, pp. 1–16, 2015.
- [16] B. L. Bhatnagar, C. Sminchisescu, C. Theobalt, and G. Pons-Moll, “Combining implicit function learning and parametric models for 3d human reconstruction,” in *European Conference on Computer Vision*, pp. 311–329, Springer, 2020.
- [17] Z. Li, M. Oskarsson, and A. Heyden, “Detailed 3d human body reconstruction from multi-view images combining voxel super-resolution and learned implicit representation,” *Applied Intelligence*, vol. 52, no. 6, pp. 6739–6759, 2022.
- [18] M. Kocabas, N. Athanasiou, and M. J. Black, “VIBE: Video inference for human body pose and shape estimation,” in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5252–5262, IEEE, June 2020.
- [19] V. Choutas, L. Müller, C.-H. P. Huang, S. Tang, D. Tzionas, and M. J. Black, “Accurate 3d body shape regression using metric and semantic attributes,” in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

-
-
- [20] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.
 - [21] 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.
 - [22] H. Beik-Mohammadi, S. Hauberg, G. Arvanitidis, G. Neumann, and L. Rozo, “Reactive motion generation on learned riemannian manifolds,” *arXiv preprint arXiv:2203.07761*, 2022.
 - [23] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303–312, 1996.
 - [24] K. Saulnier, N. Atanasov, G. J. Pappas, and V. Kumar, “Information theoretic active exploration in signed distance fields,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4080–4085, IEEE, 2020.
 - [25] L. Han, F. Gao, B. Zhou, and S. Shen, “Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4423–4430, 2019.
 - [26] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, “Convolutional occupancy networks,” in *European Conference on Computer Vision*, pp. 523–540, Springer, 2020.
 - [27] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: Learning continuous signed distance functions for shape representation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.
 - [28] G. Sutanto, I. M. R. Fernández, P. Englert, R. K. Ramachandran, and G. S. Sukhatme, “Learning equality constraints for motion planning on manifolds,” *arXiv preprint arXiv:2009.11852*, 2020.
 - [29] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, “Occupancy networks: Learning 3d reconstruction in function space,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4460–4470, 2019.
 - [30] D. Holden, J. Saito, T. Komura, and T. Joyce, “Learning motion manifolds with convolutional autoencoders,” in *SIGGRAPH Asia 2015 technical briefs*, pp. 1–4, 2015.

-
-
- [31] T. Osa, “Learning the solution manifold in optimization and its application in motion planning,” *arXiv preprint arXiv:2007.12397*, 2020.
- [32] P. S. Thomas, “Safe reinforcement learning,” 2015.
- [33] A. Hans, D. Schneegaß, A. M. Schäfer, and S. Udluft, “Safe exploration for reinforcement learning,” Citeseer.
- [34] J. Garcia and F. Fernández, “Safe exploration of state and action spaces in reinforcement learning,” *Journal of Artificial Intelligence Research*, vol. 45, pp. 515–564, 2012.
- [35] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, “Reachability-based safe learning with gaussian processes,” in *53rd IEEE Conference on Decision and Control*, pp. 1424–1431, IEEE, 2014.
- [36] M. Pecka and K. Zimmermann, “Safe exploration for reinforcement learning in real unstructured environments,”
- [37] E. Altman, “Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program,” *Mathematical methods of operations research*, vol. 48, no. 3, pp. 387–417, 1998.
- [38] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained policy optimization,” in *International conference on machine learning*, pp. 22–31, PMLR, 2017.
- [39] A. Stooke, J. Achiam, and P. Abbeel, “Responsive safety in reinforcement learning by pid lagrangian methods,” in *International Conference on Machine Learning*, pp. 9133–9143, PMLR, 2020.
- [40] D. Ding, X. Wei, Z. Yang, Z. Wang, and M. Jovanovic, “Provably efficient safe exploration via primal-dual policy optimization,” in *International Conference on Artificial Intelligence and Statistics*, pp. 3304–3312, PMLR, 2021.
- [41] A. I. Cowen-Rivers, D. Palenicek, V. Moens, M. A. Abdullah, A. Sootla, J. Wang, and H. Bou-Ammar, “Samba: Safe model-based & active reinforcement learning,” *Machine Learning*, vol. 111, no. 1, pp. 173–203, 2022.
- [42] Y. Liu, J. Ding, and X. Liu, “Ipo: Interior-point policy optimization under constraints,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 4940–4947, 2020.
- [43] C. Tessler, D. J. Mankowitz, and S. Mannor, “Reward constrained policy optimization,” *arXiv preprint arXiv:1805.11074*, 2018.

-
-
- [44] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, “A general safety framework for learning-based control in uncertain robotic systems,” *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.
- [45] S. Herbert, J. J. Choi, S. Sanjeev, M. Gibson, K. Sreenath, and C. J. Tomlin, “Scalable learning of safety guarantees for autonomous systems using hamilton-jacobi reachability,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5914–5920, IEEE, 2021.
- [46] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, “End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3387–3395, 2019.
- [47] A. Taylor, A. Singletary, Y. Yue, and A. Ames, “Learning for safety-critical control with control barrier functions,” in *Learning for Dynamics and Control*, pp. 708–717, PMLR, 2020.
- [48] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, “Safe exploration in continuous action spaces,” *arXiv preprint arXiv:1801.08757*, 2018.
- [49] K. Wainwright *et al.*, *Fundamental methods of mathematical economics*. Boston, Mass. McGraw-Hill/Irwin, 2005.
- [50] J. Stewart, *Calculus*. Cengage Learning, 2015.
- [51] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [52] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.
- [53] A. Boulch and R. Marlet, “Fast and robust normal estimation for point clouds with sharp features,” in *Computer graphics forum*, vol. 31, pp. 1765–1774, Wiley Online Library, 2012.
- [54] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3d: A modern library for 3d data processing,” *arXiv preprint arXiv:1801.09847*, 2018.
- [55] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, “AMASS: Archive of motion capture as surface shapes,” in *International Conference on Computer Vision*, pp. 5442–5451, Oct. 2019.

-
-
- [56] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.
- [57] R. P. Singh and P. W. Likins, "Singular Value Decomposition for Constrained Dynamical Systems," *Journal of Applied Mechanics, Transactions ASME*, vol. 52, no. 4, pp. 943–948, 1985.
- [58] S. S. Kim and M. J. Vanderploeg, "QR Decomposition for State Space Representation of Constrained Mechanical Dynamic Systems," *Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 108, pp. 183–188, 1986.
- [59] R. H. Byrd and R. B. Schnabel, "Continuity of the null space basis and constrained optimization," *Mathematical Programming*, vol. 35, no. 1, pp. 32–41, 1986.
- [60] M. McGuire, "Numerical methods for ray tracing implicitly defined surfaces," *Williams College*, 2014.
- [61] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

A. Appendix

This appendix contains hyperparameters and recorded data from the deep learning and reinforcement learning process.

A.1. Recording of Training Process of ReDSDF

In this Section, the hyperparameters and recorded data for training the ReDSDF are shown.

The hyperparameters can be found in Table A.1:

	Human body	TIAGo	Table	Sofa
Batch size	4096	4096	2048	2048
Hidden layer	5*512+2*32	5*512+2*32	4*512+2*32	4*512+2*32
Learning rate	1e-4	1e-4	1e-4	1e-4
Optimizer	Adam	Adam	Adam	Adam
Alpha	0.02	0.02	0.02	0.02

Table A.1.: Hyperparameters for Training ReDSDF

The curve of loss can be found in Figure A.1:

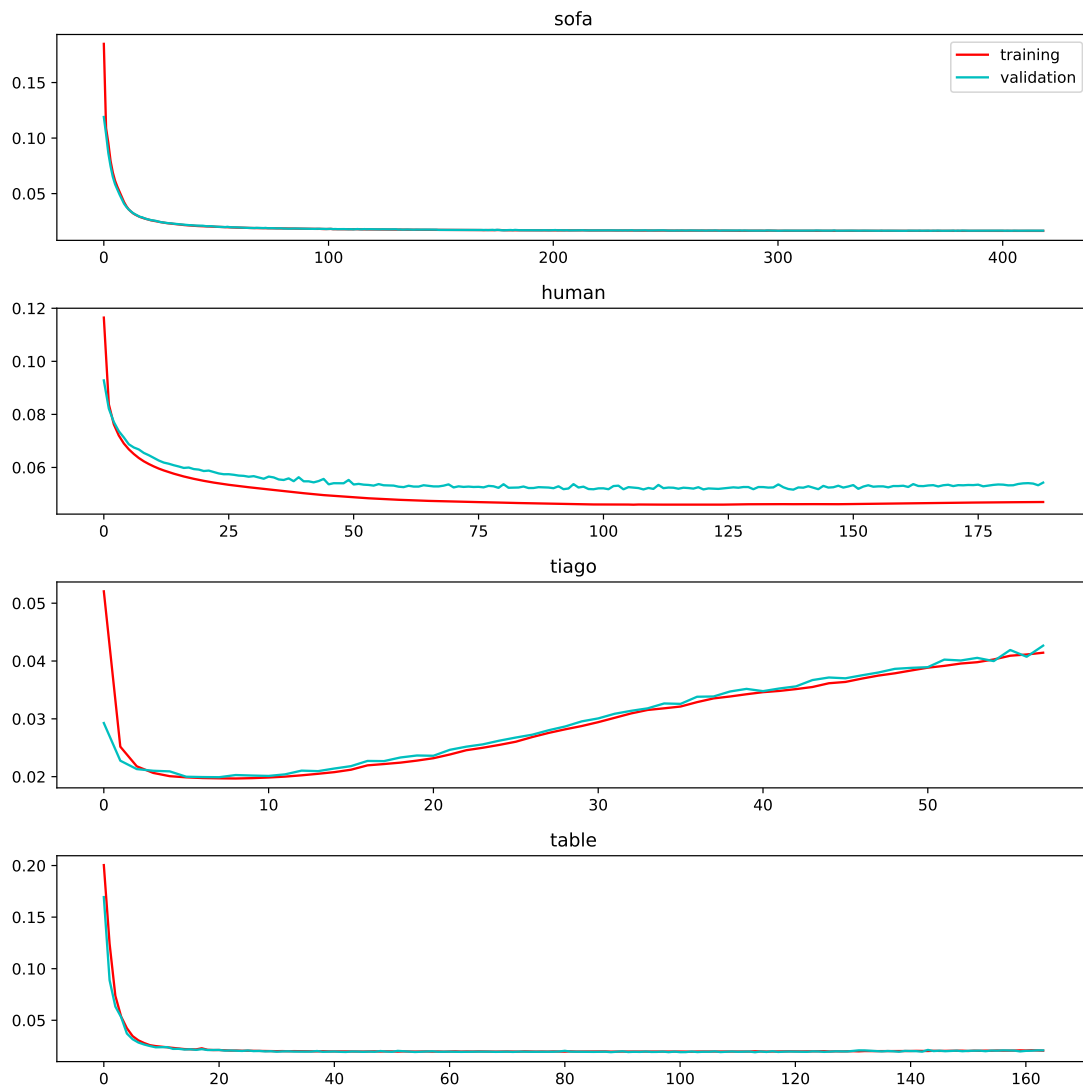


Figure A.1.: Loss Curve during Training

A.2. Hyperparameters for RL

In this section, the hyperparameters and recorded process for RL will be shown. The hyperparameters of the two RL experiments are shown in Table A.2. And the data recorded during training is shown in Figure A.2 and Figure A.3

	With Shelf	Human-Robot Interaction
Algorithm	SAC	SAC
Number of epochs	200	200
Number of steps in each epoch	10000	10000
Number of episodes for test	10	10
Number of features in policy network	256-256-256	512-512-256
Learning rate for actor network	3e-4	1e-5
Learning rate for critic network	3e-4	5e-5
Batch size	64	64
Initial replay size	5e3	5e3
Maximal replay size	2e5	2e5
τ for SAC	0.001	0.001
Warm-up Transitions	5e3	5e3
Learning rate of alpha	5e-6	1e-5
Target entropy	-10	-10
Discount factor γ	0.995	0.995
Maximal steps for one episodes	500	500
Control frequency	30Hz	30Hz
Type of slack variable for ATACOM	softcorner	exponential

K_c for error correction of ATACOM	30	20
β for slack variable of static object	30	20
β for slack variable of dynamic object	/	5
β for slack variable of joint limits	30	10
Threshold of slack variable of static object	1e-3	1e-5
Threshold of slack variable of dynamic object	/	2e-2
Threshold of slack variable of joint limits	1e-3	1e-5

Table A.2.: Hyperparameters for RL

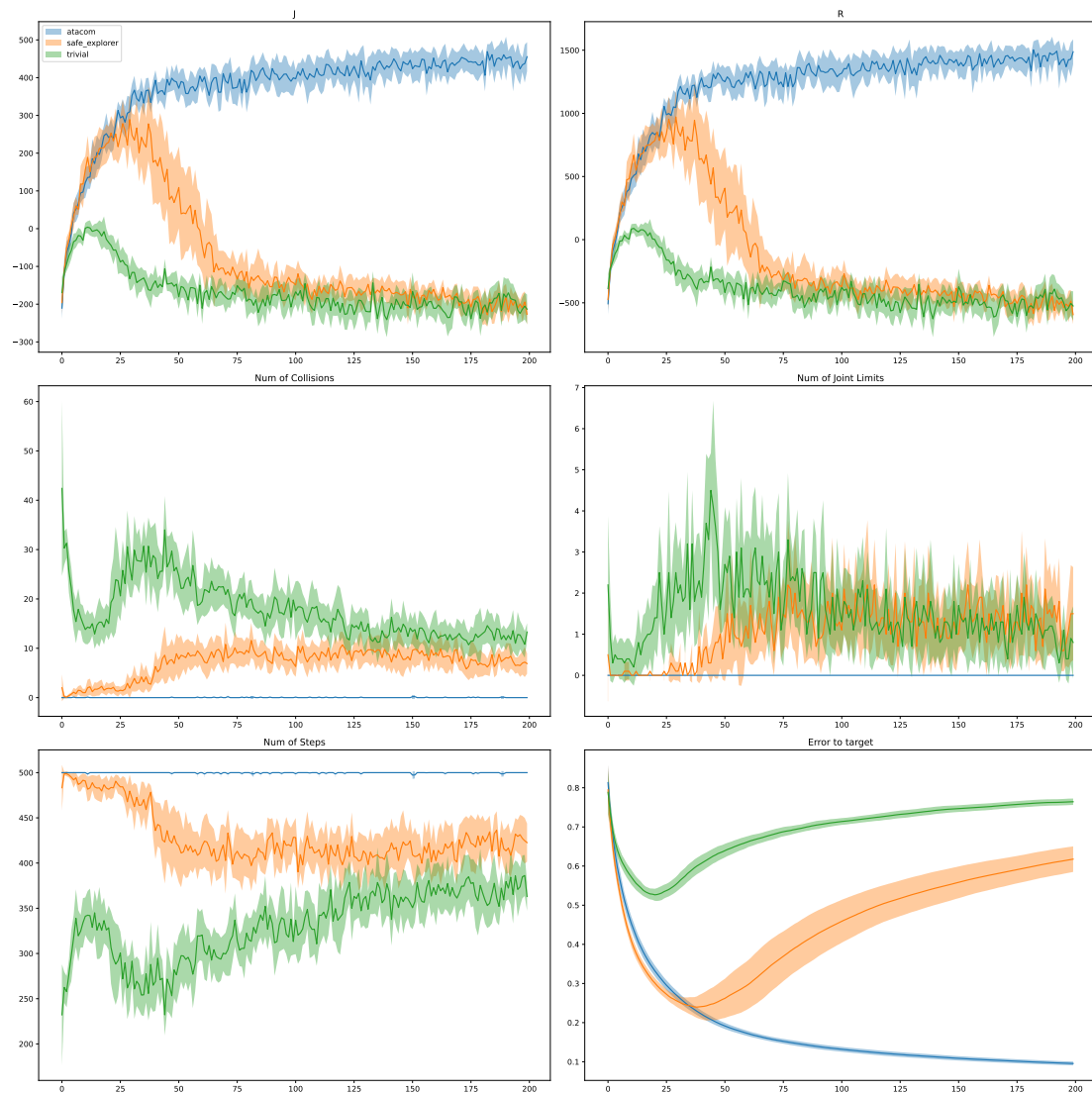


Figure A.2.: Results of RL-Experiment with Shelf

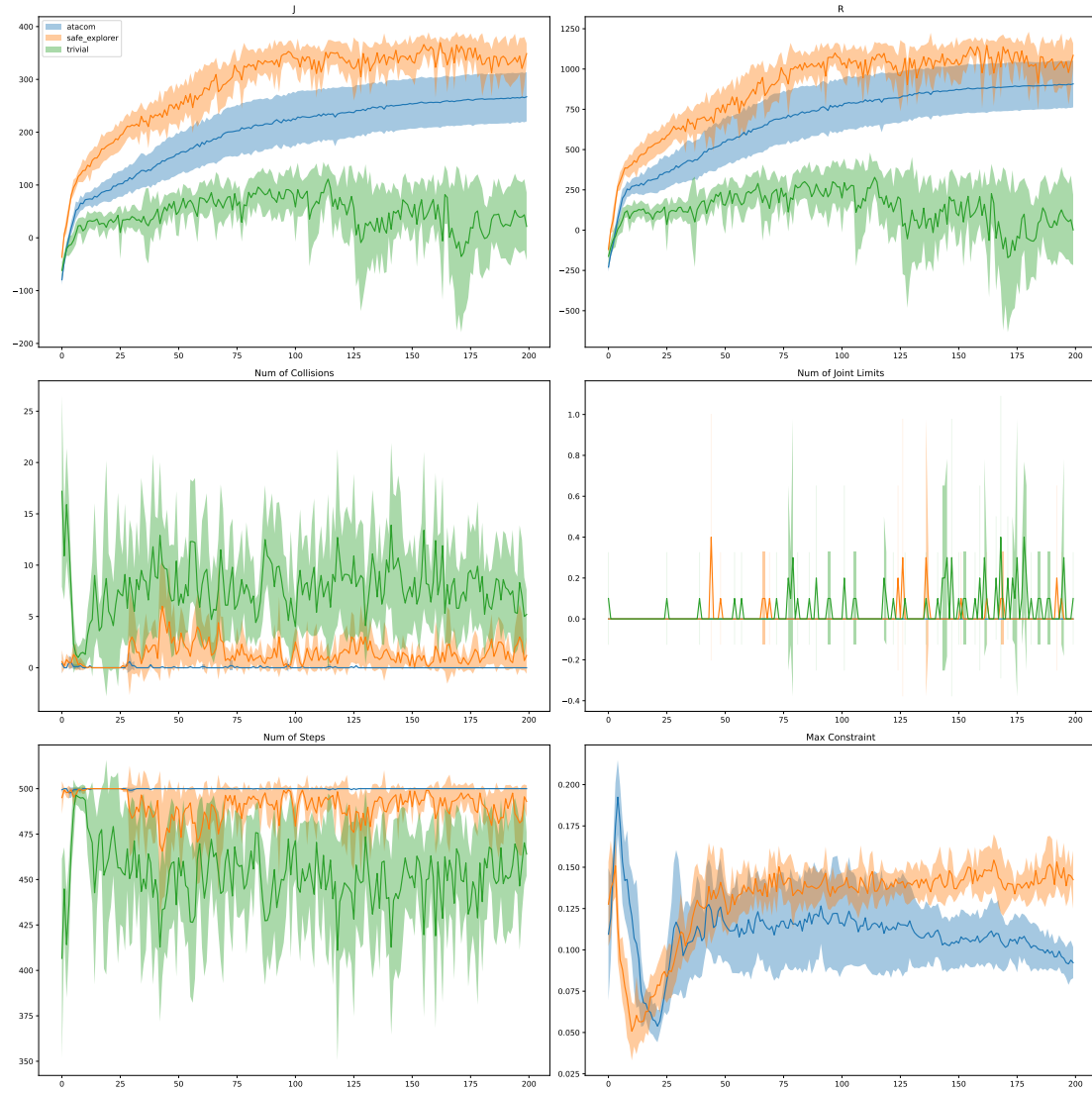


Figure A.3.: Results of RL-Experiment in Human-Robot Interaction