Efficient Collision Detection Framework for Enhancing Collision-Free Robot Motion

Xiankun Zhu, Yucheng Xin, Shoujie Li, Houde Liu†, Chongkun Xia, Bin Liang

Abstract— Fast and efficient collision detection is essential for motion generation in robotics. In this paper, we propose an efficient collision detection framework based on the Signed Distance Field (SDF) of robots, seamlessly integrated with a self-collision detection module. Firstly, we decompose the robot's SDF using forward kinematics and leverage multiple extremely lightweight networks in parallel to efficiently approximate the SDF. Moreover, we introduce support vector machines to integrate the self-collision detection module into the framework, which we refer to as the SDF-SC framework. Using statistical features, our approach unifies the representation of collision distance for both SDF and self-collision detection. During this process, we maintain and utilize the differentiable properties of the framework to optimize collision-free robot trajectories. Finally, we develop a reactive motion controller based on our framework, enabling real-time avoidance of multiple dynamic obstacles. While maintaining high accuracy, our framework achieves inference speeds up to five times faster than previous methods. Experimental results on the Franka robotic arm demonstrate the effectiveness of our approach. Project page: https://sites.google.com/view/icra2025-sdfsc.

I. INTRODUCTION

Efficiently generating safe operational trajectories for robots is a critical challenge in the increasingly common context of human-robot interactionn [1]. Motion planning algorithms need to comprehensively consider external and self-collision issues. Much like how humans perceive obstacles, collision detection serves as the robot's means of sensing environmental obstacles, ensuring it avoids collisions with its surroundings, humans, and itself [2]. Due to their low computational efficiency, traditional collision detection methods, which are generally based on the geometric information of obstacles and robot [3], [4], struggle to meet the current demand for real-time generation of collision-free trajectories in complex and dynamic environments.

The introduction of the Signed Distance Field (SDF) [5] concept offers a novel approach to addressing collision detection challenges. Originally a key research area in computer vision and graphics, SDF has been widely used to efficiently represent complex shapes [6], [7]. Migrating to the field of collision detection, SDF provides an intuitive and effective way of determining the shortest distance from

Xiankun Zhu, Yucheng Xin, Shoujie Li, Houde Liu and Bin Liang are with Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China.

Chongkun Xia is with School of Advanced Manufacturing, Sun Yat-sen University, shenzhen 518107, China

†Corresponding authors: Houde Liu (liu.hd@sz.tsinghua.edu.cn).

SDF-SC

Robot SDF

Self-collision

(a) SDF and Self-Collision

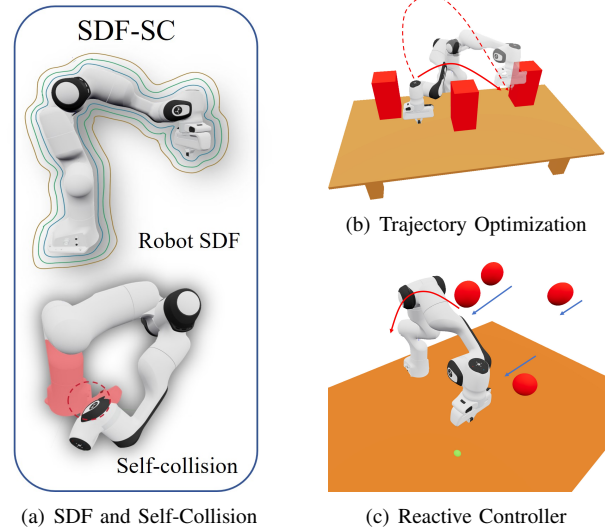(b) Trajectory Optimization

(c) Reactive Controller

Fig. 1: Overview of our work. (a) We combine our own lightweight SDF framework with self-collision detection to obtain a new collision detection framework SDF-SC. (b) Trajectory Optimization based on the SDF-SC. (c) We use SDF-SC to enable reactive control of the robot.

a point to the surface of an object, which perfectly meets the collision detectors' demand for quantifying the collision distance between robots and obstacles. When applied to motion planning, SDF is extended to represent manifolds of general equality constraints [8], facilitating real-time trajectory smoothing and enabling full-body control tasks [9], [10]. In learning SDF models, Koptev et al. [11] employ a multilayer perceptron (MLP) to fit neural implicit signed distance functions. Liu et al. [12] propose a method that employs composite SDF networks to improve the fitting accuracy of articulated robots. Bernstein polynomials are employed as basis functions for SDF, further enhancing storage and computational efficiency [13]. These approaches learn SDF models coupled with different joint configurations. However, the complexity of network architectures or the verbosity of functional representations often limits the precision and speed of collision distance inference. Moreover, SDF has inherent limitations: it can only calculate the distance to external obstacles, but cannot account for self-collision, which is a critical and non-negligible issue in both simulations and the real world [14], [15].

In this paper, we utilize a forward kinematic chain to optimize networks framework. During this process, we discover that extremely shallow networks are sufficient to achieve excellent learning outcomes for the SDFs of each link.

Moreover, we employ a parallel framework to significantly accelerate inference, increasing the speed for thousands of points to five times that of previous methods [11], [12]. The joint boundaries of self-collisions and external collisions are robustly determined using a data-driven approach based on kernel perceptrons [16], [17]. Building on this, we represent the self-collision boundary using support vectors and develop a processing function that leverages the statistical properties of support vector machine (SVM) [18] results to calculate the self-collision distance. By integrating self-collision and the external collision distance, we address a critical limitation of the prior SDF frameworks, which did not account for self-collisions during planning. This leads to the development of our final integrated system, termed the SDF-SC framework.

We apply the SDF-SC framework as inequality constraints within the optimization problem to evaluate its effectiveness in generating collision-free trajectories. Additionally, we integrate SDF-SC with a reactive control algorithm to tackle the challenge of a robot avoiding multiple dynamic obstacles. Experimental results demonstrate that this integration significantly enhances the robot's dynamic response to external and self-collision risks, leading to more robust and adaptable performance in real-world scenarios.

The contributions of our work are as follows:

- We propose an efficient and lightweight parallel networks framework for learning SDF, achieving high fitting accuracy while significantly accelerating inference speed.
- We augment the framework by incorporating integrated self-collision avoidance, while maintaining continuity and differentiability throughout the process.
- We demonstrate the effectiveness of SDF-SC framework through rigorous trajectory optimization experiments and leverage it to design a robust reactive controller.

II. FRAMEWORK COMPOSITION

In this section, we introduce a holistic framework for integrated collision detection, which includes a parallel networks framework for learning the SDF of articulated robots, an algorithm for self-collision evaluation, and dataset construction. The overall algorithmic pipeline for estimating the collision distance is depicted in Fig. 2, with the Franka Emika Panda robot serving as the subject for demonstration.

A. Parallel Signed Distance Field Networks

Consider an articulated robot with n degrees of freedom and K links, where the joint angles are defined as $\boldsymbol{q} = [q_1, q_2, \ldots, q_n]^\top \in \mathbb{R}^n$. Each link is considered as a rigid body, and the coordinate system $\{\mathbf{S}_k\}_{k=1}^K$ is rigidly attached to the rigid body. Let $\mathbf{S}_0$ denote the base coordinate system, and $\mathbf{T}_k \in \mathbb{SE}(3)$ represent the homogeneous transformation matrix from $\mathbf{S}_0$ to $\mathbf{S}_k$.

$$\mathbf{T}_k = FK_k(\boldsymbol{q}) \tag{1}$$

where $\mathbf{T}_k$ is determined by the Denavit-Hartenberg (DH) parameters and joint angles $\boldsymbol{q}$ of the articulated robot, and the computation of the transformation matrix can be achieved
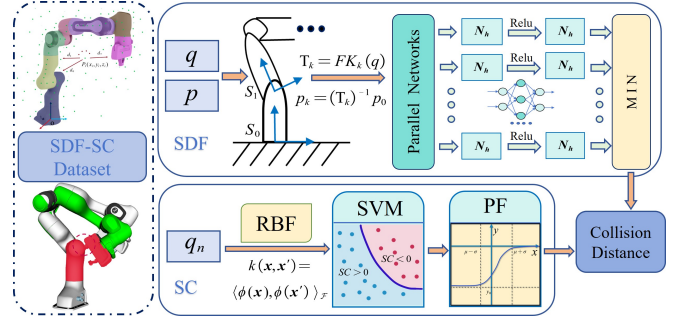

Fig. 2: Overall algorithm pipeline for estimating collision distance. Collision distance refers to the safety margin between the robot and potential collisions.

through the forward kinematic chain. We map 3D query points $\boldsymbol{p} \in \mathbb{R}^3$ from the base coordinate system onto the coordinate systems of each link through matrix transformation, which can be written as

$$\boldsymbol{p}_k = (\mathbf{T}_k)^{-1} \boldsymbol{p} \tag{2}$$

where $\boldsymbol{p}_k$ denotes the coordinate representation of point $\boldsymbol{p}$ in the coordinate system of each link. Owing to the rigid attachment of the coordinate system to the link, the metric distance $\boldsymbol{d}_k$ from a point to the kth link is solely determined by $\boldsymbol{p}_k$. It should be noted that when a point is located within the robot, we represent its distance to the surface of the links with negative values.

The distance from a point to an entire articulated robotic system is defined as the minimum distance from that point to each individual link within the system. For convenience, we employ the function $\Gamma(\boldsymbol{q}, \boldsymbol{p})$ to represent the SDF value from a point $\boldsymbol{p}$ to the articulated robot with joint configuration $\boldsymbol{q}$, with the following equation:

$$\Gamma(\boldsymbol{q}, \boldsymbol{p}) = \min(\boldsymbol{d}_1(\boldsymbol{p}_1), \boldsymbol{d}_2(\boldsymbol{p}_2), \cdots, \boldsymbol{d}_K(\boldsymbol{p}_K)) \tag{3}$$

Deviating from the traditional approach of directly mapping the relationship from the joint configuration $\boldsymbol{q}$ and point $\boldsymbol{p}$ to the fuction $\Gamma(\boldsymbol{q}, \boldsymbol{p})$ via a network [11], [19], our methodology employs the learning of the distance functions $\boldsymbol{d}_k$ for each link to effectively fit $\Gamma(\boldsymbol{q}, \boldsymbol{p})$.

We employ a series of lightweight neural networks to approximate $\boldsymbol{d}_k(\boldsymbol{p}_k)$, with paralleling these networks to expedite the computation speed. The minimum output among them is defined as SDF value for the whole articulated robot.

B. Self-Collision Detection

Calculating the geometric distances associated with self-collisions in robots is an exceedingly laborious and time-consuming process [20]. However, due to the static and unique nature of self-collision regions in the joint space of articulated robots [21], [22], it is feasible to learn a continuous differentiable function that models the self-collision boundary between colliding and free joint configurations.

To learn the self-collision function, we perform uniform sampling of configurations within the joint space and subsequently categorize them into binary classes based on whether they collide or not. The sampled configurations q_i

are constrained within the joint limits $\boldsymbol{q}^-, \boldsymbol{q}^+$, and these constraints are utilized to normalize all configurations to q_n:

$$\boldsymbol{q}^- < q_i < \boldsymbol{q}^+, i = 1, \ldots, N_s \tag{4}$$

The label $y_i = +1$ is used to denote that q_i is in the self-collision-free class, while $y_i = -1$ indicates that q_i is in the self-colliding class. Subsequently, we employ a SVM algorithm to construct a model for self-collision detection, where the decision function takes the following form:

$$S(\boldsymbol{q}) = \sum_{i=1}^{N_v} \alpha_i y_i K(\boldsymbol{q}, \boldsymbol{q}_i) + b \tag{5}$$

where $\boldsymbol{q}_i$ and y_i represent the support vectors obtained from the SVM algorithm and their respective collision labels. N_v is the number of support vectors. $0 \le \alpha_i \le C$ denotes the weight of the support vectors and C is an adjustable parameter that serves as a penalty for errors, helping to prevent overfitting. $b \in \mathbb{R}$ denotes the bias term in the decision rule.

Within the purview of kernel function selection, empirical evidence from prior studies has demonstrated that both the Radial Basis Function (RBF) kernel and forward kinematics (FK) kernel [23] manifest commendable efficacy. However, in deference to the exigencies of rapid self-collision detection, the RBF kernel $K(\boldsymbol{q}, \boldsymbol{q}_i) = e^{-\gamma \|\boldsymbol{q} - \boldsymbol{q}_i\|^2}$ has been selected for its computational efficiency.

C. Dataset Construction

In order to facilitate algorithmic comparison with prior work, it is imperative to ensure the comparability of our dataset with those utilized previously. Consequently, our sampling methodology aligns with the approaches employed in prior studies for dataset construction [11], [12], where data points $\boldsymbol{p}$ are more densely collected near link surfaces. Additionally, we have supplemented our dataset with distances obtained from the GJK in simulation environments, providing a valid supplement. Ultimately, the dataset for each link comprises 50 million entries.

In preparation for self-collision detection, we create training and validation datasets containing 100,000 and 30,000 configurations, along with a test set of 50,000 configurations. Self-collision labels are generated using the self-collision detection function from the FCL library [3].

III. HYPERPARAMETER SEARCH AND FRAMEWORK EVALUATION

A. Hyperparameter Search

Following hyperparameter optimization of the SVM model, we select $C = 50$ and $\gamma = 1.0$ as the optimal parameters, resulting in a total of $N_v = 10446$ support vectors that constitute the self-collision boundary. The score distribution of our trained model, as exhibited on the validation set, is depicted in Fig. 3(a), indicates that $S(\boldsymbol{q}) = 0$ represents the self-collision boundary constituted by the support vectors. When configurations are classified for self-collision based on the boundary, the distribution of misclassified configuration scores approximates a Gaussian distribution $X \sim N(\mu, \sigma^2)$.

TABLE I: Performance metrics of the self-collision detection

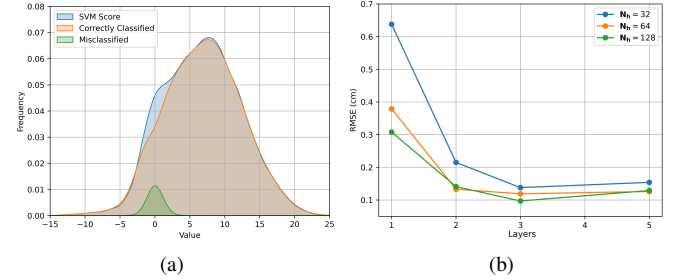| Dataset	Acc	TPR	TNR
Train	0.99	0.95	0.99
Validation	0.97	0.86	0.98
Test	0.97	0.88	0.98

(a)   (b)

Fig. 3: (a) The score distribution of the self-collision detection model on the validation set. (b) The impact of different network architectures on fitting accuracy.

For the seamless integration and enhancement of self-collision detection with collision distance metrics, the robotic configurations are delineated into three distinct regions based on statistical features: distant, proximal, and transgressing the self-collision boundary. The proximal region of the self-collision boundary is delineated according to Gaussian distribution principles as the interval $(u - \sigma, u + \sigma)$.

Within our framework, the collision distance is computed using the following formula:

$$D(\boldsymbol{q}, \boldsymbol{p}) = \Gamma(\boldsymbol{q}, \boldsymbol{p}) + P(S(\boldsymbol{q})) \tag{6}$$

where P denotes a processing function for the self-collision scores. The equation for ∂D is naturally derived as follows:

$$\frac{\partial D(\boldsymbol{q}, \boldsymbol{p})}{\partial \boldsymbol{q}} = \frac{\partial \Gamma(\boldsymbol{q}, \boldsymbol{p})}{\partial \boldsymbol{q}} + \frac{\partial P}{\partial S} \cdot \frac{\partial S}{\partial \boldsymbol{q}} \tag{7}$$

$$\frac{\partial S}{\partial \boldsymbol{q}} = -2\gamma \sum_{i=1}^{N_v} \alpha_i y_i e^{-\gamma \|\boldsymbol{q} - \boldsymbol{q}_i\|^2} (\boldsymbol{q} - \boldsymbol{q}_i) \tag{8}$$

In the case of single-arm robots, such as the Franka, self-collision configurations represent a special case, with their occurrence being significantly less frequent than external collisions. Consequently, we expect the collision distance to approximate the robot's SDF in regions deep within the self-collision boundary, and to invert to negative values as an indicator of constraint when configurations transgress the boundary. Furthermore, the gradient ∂P is aligned with $\partial \Gamma$, both directing towards increasing collision distance.

To this end, we formulate a continuous, monotonically increasing processing function P, bounded both above and below, as presented below:

$$P(s) = -\frac{1}{1 + e^{ks + b_0}} \tag{9}$$

where the parameters k and b_0 are determined by setting a suitable interval for $P(u - \sigma, u + \sigma)$ in the range $[-0.95, -0.001]$ to fit the statistical characteristics of S.

Before evaluating the performance of external collision distance detection, we conduct an optimization search for

the number of layers and nodes in the parallel SDF network to determine the most effective architectural parameters for the networks. The layer count ranges from 1 to 5, while the node count varies between 32, 64, and 128. Each network in the parallel configuration uniformly employs a Root Mean Squared Error (RMSE) loss function, with the Adam optimizer used consistently across all. The ReLU function is selected as the activation function and the maximum number of iterations is set to 100 to ensure network convergence.

The impact of different network frameworks on the final RMSE convergence is illustrated in Fig. 3(b). Experiments reveal that a parallel configuration comprising merely two layers of networks is sufficient to achieve an excellent fit for the articulated robot's SDF, with additional layers providing minimal benefit in terms of RMSE reduction and, conversely, increasing the inference time. To balance accuracy and inference speed, we select a configuration of 2 layers with 64 nodes as the final architecture parameters for our parallel SDF networks, which makes our model significantly lighter compared to previous work [11], [12].

B. Framework Evaluation

All experiments are conducted using an Intel Core i7-13700KF and an NVIDIA GeForce RTX 4070 GPU, with all GPU operations performed using batch processing.

To evaluate the framework's effectiveness for self-collision detection, we evaluate it on the test dataset, employing Accuracy (Acc), True Negative Rate (TNR), and True Positive Rate (TPR) as metrics. The collision distance $D(\boldsymbol{q}, \boldsymbol{p})$ boundary for classification is set at $u - \sigma$. A high TPR prevents the misclassification of collision-free configurations as colliding, while a low TPR may result in overly conservative detection. On the other hand, a high TNR is the most critical metric that should be maximized, as it prevents the erroneous labeling of colliding configurations as free. The self-collision detection model within our framework achieves commendable performance across various datasets, maintaining a TNR of over 0.98 in all cases, as detailed in the Table I.

In our experimental analysis, we evaluate the mean computational latency of our comprehensive framework, denoted as SDF-SC, across a spectrum of 3D query points. This evaluation is conducted in comparison with the relevant Composite-SDF [12], Neural-JSDF [11], RDF [13] and the standard Gilbert-Johnson-Keerthi (GJK) [4] algorithm. The computations within the SDF-SC framework encompass not only the forward kinematics chain and the parallel SDF networks but also self-collision detection. The experimental results are presented in Table II. Despite the comprehensive nature of these computations, our detection efficiency remains highly superior, exhibiting only a slight disadvantage when the number of query points is relatively low in comparison to the Neural-JSDF algorithm. However, in general scenarios where environmental modeling often requires sampling several thousand points, our detection efficiency is generally five times greater than that of previous methods in standard applications.

TABLE II: Comparison of computational times for collision distance by various algorithms

Number of Query Pointss	1	100	1000	10000
SDF-SC **(ours)** ,ms	0.98	1.02	**1.05**	**1.21**
Composite SDF[1],ms	4.33	4.78	-	6.63
Neural-JSDF ,ms	**0.18**	**0.42**	1.13	8.98
RDF(N=8) ,ms	1.12	1.21	1.27	2.12
GJK ,ms	0.95	87.4	815	7984

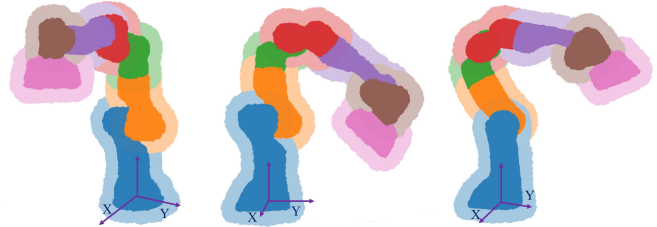[1] The data cited in this row is derived from paper [12].

Fig. 4: The reconstruction of the Franka Emika Panda's links is performed based on the distance isosurfaces $D(\boldsymbol{q}, \boldsymbol{p}) = 0$ cm (solid) and $D(\boldsymbol{q}, \boldsymbol{p}) = 5$ cm (transparent). Different colors represent different links.

The average RMSE across the links of the articulated robot serves as a metric for the quality of SDF fitting. Distances are categorized with $\boldsymbol{d} < 0.1$ m as the region close to the robot and 0.1 m $< \boldsymbol{d} < 1.2$ m as the region distant from the robot. We compare our results within these two regions with those of the Neural-JSDF, Composite SDF and RDF algorithms. The results are as shown in the Table III.

TABLE III: Comparing the SDF accuracy obtained by the algorithms

Region	$\boldsymbol{d} < 0.1$ m	0.1 m $< \boldsymbol{d} < 1.2$ m
SDF-SC **(ours)**, cm	**0.16**	**0.28**
Composite SDF, cm	0.21	0.36
Neural-JSDF, cm	1.04	1.06
RDF(N=8), cm	0.41	0.54

Our method exhibits superior accuracy in both distance intervals, with an overall precision of 0.19 cm, attaining millimeter-level accuracy. To more vividly demonstrate the accuracy of our model, we utilize the integrated framework to generate a 3D point cloud, thereby reconstructing the surface of the Panda robot at $D(\boldsymbol{q}, \boldsymbol{p}) = 0$ cm. As illustrated in Fig. 4, the SDF reconstructed by our method accurately reflects the surface contours of the Panda robot, with very uniform widths between adjacent isosurfaces, indicating that the integrated framework can effectively obtain the precise SDF of the articulated robot.

The evaluations conducted indicate that the integration of self-collision and external collision detection into a unified collision distance $D(\boldsymbol{q}, \boldsymbol{p})$ within our framework is viable, and it performs effectively in detecting both types of collisions.

IV. MONTION PLANNING EXPERIMENTS

In this section, we substantiate the superiority of our framework in robotic motion planning through two categories

of experiments: 1) *Trajectory Optimization* 2) *Reactive Control*. During planning, we utilize $D(\boldsymbol{q}, \boldsymbol{p})$ as the collision distance to simultaneously avoid self-collisions and external collisions.

A. Trajectory Optimization with SDF-SC

To achieve the most efficient trajectory while meeting the requirement of avoiding static obstacles, we define the optimization problem as follows:

$$\min_{q} \quad f(\boldsymbol{q}) = \sum_{t=0}^{T-1} \| FK\left(\boldsymbol{q}_{t+1}\right) - FK\left(\boldsymbol{q}_t\right) \|^2$$

$$\text{subject to} \quad \begin{aligned} D(\boldsymbol{q}, \boldsymbol{p}) + \epsilon &\leq 0 \\ \|\boldsymbol{q}_{t+1} - \boldsymbol{q}_t\| &\leq \boldsymbol{w}_{\max} \\ \|\mathbf{ee}(\boldsymbol{q}_{t+1}) - \mathbf{ee}(\boldsymbol{q}_t)\| &\leq \boldsymbol{v}_{\max} \\ \boldsymbol{q}^- \leq \boldsymbol{q} &\leq \boldsymbol{q}^+ \end{aligned} \quad (10)$$

where $f(\boldsymbol{q})$ denotes the cost function of the optimization problem. T represents the number of planned path points or time duration and FK stands for the forward kinematics chain of each link. ϵ denotes the safety margin, which is employed to augment the level of conservatism in collision avoidance. $\mathbf{ee}$ denotes the position of the end-effector, while $\boldsymbol{v}_{\max}$ represents the maximum velocity for the end-effector.

To demonstrate the broad effectiveness of our optimization algorithm, we design scenarios with a simple environment containing three obstacles and a complex environment containing ten obstacles, as illustrated in Fig. 5. In our approach, we utilize the RRTconnect [24] algorithm to swiftly generate an initial trajectory, followed by employing the SLSQP [25] optimizer for our optimization problem, thereby obtaining the final optimized trajectory. Since our framework is differentiable, the gradients of the constraint functions can be obtained using the automatic differentiation functions in the PyTorch library. As a benchmark, we use the RRT* algorithm [26], which is widely applied in path planning and theoretically proven to achieve asymptotic optimality [27]. The maximum iteration time for RRT* is set to 50 s. Fig. 6(a) shows the variation of collision distance over the movement time during our trajectory planning experiments in a simple scenario for various algorithms. By minimizing the path cost, our approach allows the robot arm's trajectory to approach obstacles more closely, without exceeding safety margins.

Additionally, we perform 50 planning experiments in both simple and complex scenarios, and then take the average of the results from both sets. The comparative results are illustrated in Fig. 6(b). Benefiting from the fully differentiable nature of our framework, which requires only approximately 3 ms per differentiation, we significantly reduce the path cost in trajectory planning using SDF-SC significantly, while maintaining a high success rate and relatively fast computation time.

To validate the self-collision detection capabilities of our framework, we deliberately introduce self-collision configurations into the initial trajectories. Our framework is tasked
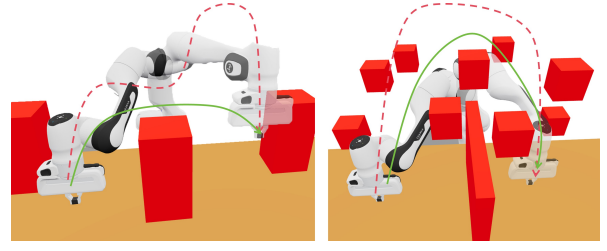
Fig. 5: Trajectory optimization in simple (left) and complex (right) scenarios, where the red dashed line represents the trajectory generated by RRT*, and the green dashed line represents the trajectory optimized using SDF-SS with RRT-connect.
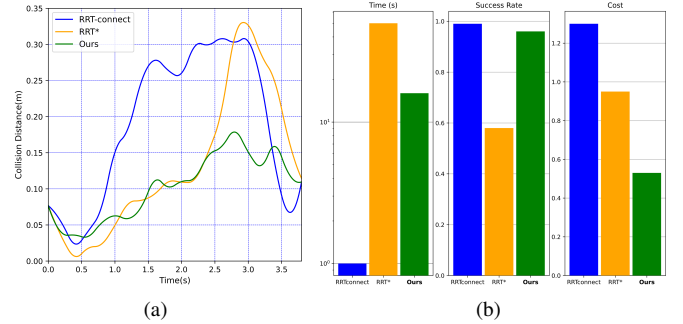
Fig. 6: (a) Collision distance during the movement period. (b) Comparative analysis of trajectory planning methods.

with detecting the presence of self-collisions along these trajectories and optimizing them to eliminate such incidents. We design approximately 20 such trajectories, and our system successfully detects 95% of the self-collisions, while also optimizing the trajectories to avoid these points. However, it is important to note that due to the use of a steep gradient to escape self-collisions, the trajectory transitions were not as smooth as desired, necessitating additional interpolation to refine the affected segments. Our framework is capable of detecting both external and self-collisions within the trajectory, while also using gradient guidance to optimize trajectory and avoid collisions.

B. Reactive Control with SDF-SC

Inspired by NEO [28], we utilize the SDF-SC framework to design a reactive controller, designated as NEO-SS, intended for real-time dynamic obstacle avoidance. We define the following quadratic programming (QP) problem [29] for the controller:

$$\min_{x} \quad g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \mathcal{Q}\boldsymbol{x} + \mathcal{C}^\top \boldsymbol{x}$$

$$\text{subject to} \quad \begin{aligned} \mathcal{J}\boldsymbol{x} &= \boldsymbol{\nu} \\ \mathcal{A}\boldsymbol{x} &\leq \mathcal{B} \\ \mathcal{X}^- \leq \boldsymbol{x} &\leq \mathcal{X}^+ \end{aligned} \quad (11)$$

where

$$\boldsymbol{x} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{\delta} \end{pmatrix} \in \mathbb{R}^{(n+6)} \quad (12)$$

$\boldsymbol{\delta}$ is a slack variable with respect to $\dot{\boldsymbol{q}}$, representing the difference between the desired and actual end-effector velocity

ν. $\mathcal{C}$ maximizes the manipulability of the robot, which is synthesized from $\boldsymbol{J}_m$ [30] and the identity matrix. In NEO, the inequality constraints of a QP problem are represented as:

$$\boldsymbol{J}_{d_l}(\tilde{\boldsymbol{q}})\dot{\tilde{\boldsymbol{q}}}(t) \leq \xi \frac{d - d_s}{d_i - d_s} - \hat{\boldsymbol{n}}_{or_l}^{\top}\dot{\boldsymbol{p}}_{o_l}(t) \qquad (13)$$

where $\tilde{\boldsymbol{q}}$ represents the subset of joint variables that affect the distance to the obstacles, denoted by $\boldsymbol{J}_d$. d_i for the activation distance of the controller and d_s for the safe stopping distance. The variable l denotes the index of obstacles, and the presence of multiple obstacles leads to a substantial increase in the constraints derived from the equations, which significantly impacts the success rate of solving the optimization problem. In our SDF-SC framework, obstacles are not represented as individual geometric shapes but are instead denoted as a collection of point clouds. Consequently, we can improve the inequality constraints as follows:

$$\frac{\partial D}{\partial \tilde{\boldsymbol{q}}}\dot{\tilde{\boldsymbol{q}}} \leq \xi \frac{D - d_s}{d_i - d_s} - \dot{\boldsymbol{p}} - s \qquad (14)$$

where $\dot{\boldsymbol{p}}$ represents the maximum velocity of the detected obstacle points, and s denotes the self-collision boundary, which is utilized to constrain the configurations to prevent self-collisions.

To assess the effectiveness of the improvements, we construct a simulation scenario as depicted in Fig. 7(b). In the scenario, multiple obstacle spheres are moving toward the robotic arm, which must complete the task of moving to a specified posture while avoiding these dynamic obstacles. In order to compare the effectiveness of the algorithm before and after the improvement of the inequality constraints, we set the adjustable parameters of NEO and NEO-SS to be consistent, with d_i =0.4m, d_s =0.02m, and $\xi = 1.0$, and testing experiments are conducted under the same scenario. The experimental data is illustrated in Fig. 7(a). It is evident that NEO-SS exhibits reduced fluctuations, effectively keeping the distance well within the safety threshold. Conversely, NEO exhibits excessive fluctuations and results in a collision.

To evaluate the robustness of the reactive control algorithm, we randomize the positions and velocities of the obstacles within a certain range. The velocity of each obstacle fluctuates randomly around 0.2 m/s. We perform 50 experiments to ascertain the per-step computational latency of two control algorithms and to evaluate the efficacy of collision avoidance in each experiment. The compiled data regarding per-step latency and success rates are articulated in Table IV. The experimental data indicates that in complex scenarios involving multiple dynamic obstacles, the NEO-SS algorithm demonstrates superior robustness compared to the NEO algorithm.

As shown in Fig. 7(c), we implement our reactive control algorithm NEO-SS on a robot arm platform. The platform comprises a Franka Emika robotic arm and a Realsense D435i camera mounted at a fixed location for detecting obstacles. The robotic arm endeavors to execute reactive control measures to avoid the introduced dynamical obstacles concurrent with its mission to reach the designated target.

TABLE IV: Comparison between NEO and NEO-SS

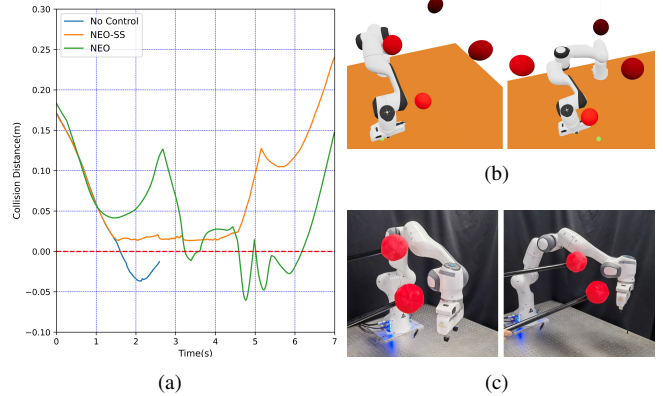| Controller	NEO	NEO-SS**(ours)**
Average time	**3ms**	6ms
Success rate	58%	**86%**

(a)

(b)

(c)

Fig. 7: (a) Collision distance during the movement period. (b) Simulation experiment. (c) Real robot experiment.

During the experimental procedure, the robotic arm exhibits a distinct aversion to obstacles. As the obstacles approach within a certain threshold, the arm articulates its joints to move away from the obstacles, endeavoring to maintain the collision distance within a safe margin. Once the obstacles exit the reactive controller's activation range, the arm resumes its trajectory toward the target posture, calculated based on the inverse kinematics solution. For demonstration videos in both simulation and real-world environments, please refer to the attached video files.

V. CONCLUSIONS

In this paper, we improved the framework of learning SDF based on the forward kinematics of articulated robots, utilizing multiple extremely lightweight networks in parallel to more efficiently approximate SDF. Additionally, we innovatively introduced self-collision detection into the framework, resulting in a collision detector that completely prevents collisions, termed SDF-SC. We then evaluated the detector's performance in both self-collision and external collision detection, achieving highly satisfactory results in both domains. Furthermore, we utilized the SDF-SC as a collision constraint in trajectory optimization, enabling the robot to minimize path cost during planning while ensuring collision-free operation. Lastly, we integrated SDF-SC into the design of a reactive controller, successfully achieving dynamic obstacle avoidance in complex environments.

There are aspects of our approach that offer potential for further improvement. In some cases, the requirement to avoid self-collisions may lead the optimization solver to converge to a local optimum. The reliance on numerical differentiation for gradient computation leads to diminished efficiency. In the future, we plan to extend our framework to dual-arm robots to further enhance their capabilities.

REFERENCES

[1] S. Shi, J. Chen and Y. Li, "Hybrid Safety Certificate for Fast Collision Checking in Sampling-Based Motion Planning," in IEEE Robotics and Automation Letters, vol. 8, no. 1, pp. 113-120, Jan. 2023.

[2] C. Quintero-Peña, W. Thomason, Z. Kingston, A. Kyrillidis and L. E. Kavraki, "Stochastic Implicit Neural Signed Distance Functions for Safe Motion Planning under Sensing Uncertainty," 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 2360-2367.

[3] J. Pan, S. Chitta and D. Manocha, "FCL: A general purpose library for collision and proximity queries," 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 2012, pp. 3859-3866.

[4] E. G. Gilbert, D. W. Johnson and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," in IEEE Journal on Robotics and Automation, vol. 4, no. 2, pp. 193-203, April 1988.

[5] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 165–174.

[6] M. Macklin, K. Erleben, M. M¨uller, N. Chentanez, S. Jeschke, and Z. Corse, "Local optimization for robust signed distance field collision," Proceedings of the ACM on Computer Graphics and Interactive Techniques, vol. 3, no. 1, pp. 1–17, 2020.

[7] D. Joho, J. Schwinn and K. Safronov, "Neural Implicit Swept Volume Models for Fast Collision Detection," 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 15402-15408.

[8] G. Sutanto, I. R. Fern´andez, P. Englert, R. K. Ramachandran, and G. Sukhatme, "Learning equality constraints for motion planning on manifolds," in Proc. Conference on Robot Learning (CoRL). PMLR, 2021, pp. 2292–2305.

[9] S. Fujii and Q. -C. Pham, "Realtime Trajectory Smoothing with Neural Nets," 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022, pp. 7248-7254.

[10] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters and G. Chalvatzaki, "Regularized Deep Signed Distance Fields for Reactive Motion Generation," 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 6673-6680.

[11] M. Koptev, N. Figueroa and A. Billard, "Neural Joint Space Implicit Signed Distance Functions for Reactive Robot Manipulator Control," in IEEE Robotics and Automation Letters, vol. 8, no. 2, pp. 480-487, Feb. 2023.

[12] B. Liu, G. Jiang, F. Zhao and X. Mei, "Collision-Free Motion Generation Based on Stochastic Optimization and Composite Signed Distance Field Networks of Articulated Robot," in IEEE Robotics and Automation Letters, vol. 8, no. 11, pp. 7082-7089, Nov. 2023.

[13] Y. Li, Y. Zhang, A. Razmjoo and S. Calinon, "Representing Robot Geometry as Distance Fields: Applications to Whole-body Manipulation," 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 15351-15357.

[14] C. Fang, A. Rocchi, E. M. Hoffman, N. G. Tsagarakis and D. G. Caldwell, "Efficient self-collision avoidance based on focus of interest for humanoid robots," 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, Korea (South), 2015, pp. 1060-1066.

[15] S. S. M. Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," Int. J. Robot. Res., vol. 37, no. 10, pp. 1205–1232, 2018.

[16] N. Das and M. Yip, "Learning-Based Proxy Collision Detection for Robot Motion Planning Applications," in IEEE Transactions on Robotics, vol. 36, no. 4, pp. 1096-1114, Aug. 2020.

[17] Y. Zhi, N. Das and M. Yip, "DiffCo: Autodifferentiable Proxy Collision Detection With Multiclass Labels for Safety-Aware Trajectory Optimization," in IEEE Transactions on Robotics, vol. 38, no. 5, pp. 2668-2685, Oct. 2022.

[18] T. Joachims and C.-N. Yu, "Sparse kernel SVMs via cutting-plane training," Mach. Learn., vol. 76, pp. 179–193, 08 2009.

[19] J. Chase Kew, B. Ichter, M. Bandari, T.-W. E. Lee, and A. Faust, "Neural collision clearance estimator for batched motion planning," in Proc. Int'l Workshop on Algorithmic Foundations of Robotics, pp. 73–89, 2020.

[20] M. Koptev, N. Figueroa and A. Billard, "Real-Time Self-Collision Avoidance in Joint Space for Humanoid Robots," in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1240-1247, April 2021.

[21] Rakita, Daniel , B. Mutlu , and M. Gleicher . "RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion." Robotics: Science and Systems 2018.

[22] Kim, Jihwan and Frank Chongwoo Park. "PairwiseNet: Pairwise Collision Distance Learning for High-dof Robot Systems." Conference on Robot Learning (2023). 229:2863-2877.

[23] N. Das and M. C. Yip, "Forward Kinematics Kernel for Improved Proxy Collision Checking," in IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 2349-2356, April 2020.

[24] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat., 2000, pp. 995–1001.

[25] D. Kraft, "Algorithm 733: TOMP–Fortran modules for optimal control calculations," ACM Trans. Math. Softw., vol. 20, no. 3, pp. 262–281, 1994.

[26] Karaman S, Frazzoli E, "Sampling-based algorithms for optimal motion planning," The International Journal of Robotics Research. 2011;30(7):846-894.

[27] M. Elbanhawi and M. Simic, "Sampling-Based Robot Motion Planning: A Review," in IEEE Access, vol. 2, pp. 56-77, 2014.

[28] J. Haviland and P. Corke, "NEO: A Novel Expeditious Optimisation Algorithm for Reactive Motion Control of Manipulators," in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1043-1050, April 2021.

[29] J. Haviland, N. Sünderhauf and P. Corke, "A Holistic Approach to Reactive Mobile Manipulation," in IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 3122-3129, April 2022.

[30] J. Park, W. Chung, and Y. Youm, "Computation of gradient of manipulability for kinematically redundant manipulators including dual manipulators system," Trans. Control, Automat. Syst. Eng., vol. 1, no. 1, pp. 8–15, 1999.