2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Representing Robot Geometry as Distance Fields: Applications to Whole-body Manipulation

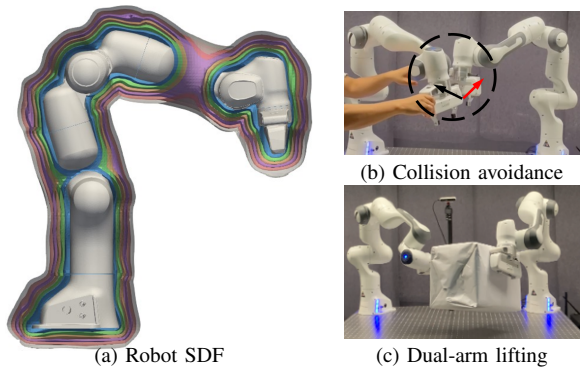
Yiming Li[1, 2], Yan Zhang[1,2], Amirreza Razmjoo[1,2], and Sylvain Calinon[1,2]

***Abstract*— In this work, we propose a novel approach to represent robot geometry as distance fields (RDF) that extends the principle of signed distance fields (SDFs) to articulated kinematic chains. Our method employs a combination of Bernstein polynomials to encode the signed distance for each robot link with high accuracy and efficiency while ensuring the mathematical continuity and differentiability of SDFs. We further leverage the kinematics chain of the robot to produce the SDF representation in joint space, allowing robust distance queries in arbitrary joint configurations. The proposed RDF representation is differentiable and smooth in both task and joint spaces, enabling its direct integration to optimization problems. Additionally, the 0-level set of the robot corresponds to the robot surface, which can be seamlessly integrated into whole-body manipulation tasks. We conduct various experiments in both simulations and with 7-axis Franka Emika robots, comparing against baseline methods, and demonstrating its effectiveness in collision avoidance and whole-body manipulation tasks. Project page: https://sites.google.com/view/lrdf/home**

I. Introduction

In robotics, the representation of a robot commonly relies on low-dimensional states, like joint configuration and end-effector poses. However, this low-dimensional representation lacks internal structure details and is insensitive to external factors, limiting the ability to interact with the environment and respond to real-world. To handle this problem, some geometric representations have been proposed, like primitives and meshes, with various applications [1], [2]. However, they either make simplified assumptions or require significant computational resources to obtain a detailed model.

A natural idea for handling this problem is to encode the geometry of the robot as signed distance fields (SDFs). Several studies in computer vision and graphics have shown the advantages of such a representation [3], [4]. Not only does it offer continuous distance information but also exhibits query efficiency. However, despite several robot representations that can be transformed into SDFs, they are either inaccurate (sphere), computationally complex (mesh), or memory-consuming (voxel SDF). Besides, the SDF representation for articulated objects remains a challenge due to the nonlinear and high dimensionality.



(a) Robot SDF (b) Collision avoidance (c) Dual-arm lifting

Fig. 1: Overview of this work. (a) A precise SDF model of an articulated robot is obtained efficiently by our proposed method. (b) Collision avoidance task based on the SDF representation. (c) Whole-body lifting task with dual-arm.

Following existing robot representations like spheres and meshes, we exploit the kinematic structure to represent the distance fields of robots. In contrast to existing methods that encode the robot shape as its joint angle configuration [5], [6], we adopt a configuration-agnostic approach during the learning phase and utilize the kinematic chain of the robot during the inference phase. This approach simplifies the problem by learning the SDF for each robot link, reducing the dimensionality, and making it robust and reliable for different joint values. During the inference phase, the kinematic information is used to retrieve the SDF values. We utilize Bernstein polynomials as the basis function to represent SDF for each link of the robot for storage and computation efficiency, with facilitated differentiability in task space and joint angle space.

Representing robot geometry as distance fields (RDF) has multiple advantages. First, it provides a continuous and smooth distance representation, granting easy access to derivatives. This characteristic is particularly well-suited for robot optimization problems such as motion planning and collision avoidance. Further, RDF representation encodes the robot geometry implicitly and decouples from spatial resolutions, enabling whole-body manipulation at any scale without explicitly defining surface points. Finally, RDF allows computationally efficient and precise distance query, which is crucial for various robot applications that require precise perception and quick response, especially in dynamic environments.

We experimentally demonstrate the capabilities of our RDF in three aspects. First, we provide a quantitative com-

[1] Idiap Research Institute, Switzerland
[2] Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
Email: {name.surname}@idiap.ch

This work was supported by the China Scholarship Council (No. 202204910113), by the State Secretariat for Education, Research and Innovation in Switzerland for participation in the European Commission's Horizon Europe Program through the INTELLIMAN project (https://intelliman-project.eu/, HORIZON-CL4-Digital-Emerging Grant 101070136) and the SESTOSENSO project (http://sestosenso.eu/, HORIZON-CL4-Digital-Emerging Grant 101070310). We also acknowledge support from the SWITCH project (https://switch-project.github.io/), funded by the Swiss National Science Foundation.

parison of the produced distance fields against other representative methods, showing the advantage of our approach. Then, we conduct collision avoidance experiments to show the real-time control performance. Finally, we present a novel formulation that leverages the RDF representation for manipulation tasks requiring contact, by generalizing the robot's Jacobian matrix from its end-effector to the 0-level set of SDF. We demonstrate the effectiveness of this approach in a dual-arm lifting task, showing how our RDF representation can be seamlessly integrated into first and second-order optimization problems. In summary, our contributions are:

- We propose a simple and flexible structure that leverages Bernstein polynomials to encode SDFs, showing high accuracy and efficiency while ensuring continuity and differentiability.
- The proposed SDF representation is further extended to articulated robots by leveraging the kinematics chain, allowing robust interpolation/extrapolation to any joint configuration while keeping the above properties.
- We demonstrate the effectiveness of our RDF representation in experiments and show how to integrate it into optimization problems for whole-body manipulation tasks without defining any points on the robot surface.

II. Related Work

SDFs for scene/object representation. Representing objects or scenes as SDFs is an active research topic in computer vision and graphics due to its query efficiency and the ability to describe complex shapes [4], [7], [8]. Typically, it is a scalar field defined over a 3D space that assigns signed distance values to points, representing the distance to the surface. The capability of SDFs in modeling object and scenes have shown in versatile applications like mapping [9], [10], grasping [11], [12] and rearrangement [13]. Liu *et al.* [14] has explored optimizing diverse grasping configurations based on the object SDF. Driess *et al.* [15] proposes to learn kinematic and dynamic models as SDFs for robot manipulation.

SDFs for motion planning. SDFs can also be utilized in optimization problems such as robot motion planning and control [16], [17]. CHOMP [18] proposes to use SDF to represent the environment and achieve effective motion planning in trajectory optimization. Schmidt *et al.* [19] uses SDF representation to track articulated objects by exploiting their kinematics structures. Sutanto *et al.* [20] extends learning SDFs to approximate generic equality constraint manifolds. Liu *et al.* [5] presents a regularized SDF with neural networks to ensure the smoothness of SDFs at any scale, testing it in collision avoidance and reactive control tasks. Vasilopoulos *et al.* [21] sample point on the robot surface and compute their SDF values with GPU acceleration for motion planning. Although representing scenes as SDFs has shown several advantages in robotics tasks, the environment is usually diverse and dynamic, and it is inefficient to obtain SDFs for arbitrary scenes.

SDFs for robot geometry representation. An intuitive idea to solve the problem of arbitrary scenes is to represent the robot as an SDF (in addition, or instead of the scene). Koptev *et al.* [6] propose to learn SDFs expressed in joint space with neural networks, allowing query distance values with points and joints as input. Similarly, in [5], an SDF model is trained with joint angles as input to represent a mobile robot. Michaux *et al.* [22] introduce a reachability-based SDF representation that can compute the distance between the swept volume of a robot arm and obstacles.

The aforementioned methods learn an SDF model coupled with joint angles, and the performance is limited due to the high dimensional and nonlinear space. In contrast, our approach exploits the kinematic chain which simplifies the problem and leads to better accuracy. A concurrent work [23], conducted in parallel to our work, also proposes to exploit the kinematic structure with neural networks for collision avoidance. The differences lie in three aspects: 1) Our representation provides interpretable parameters, holds smoothness guarantees and provides an easy way to compute analytic gradients; 2) We further extend this representation to contact-aware manipulation tasks instead of pure collision avoidance; 3) We conduct a detailed comparison with various SDF representations. All approaches exploit parallel computing, implemented with batch operation for both points and joints, consequently showing high computational efficiency. The accuracy of neural network method (one of our baselines) is also better than experimental results in [23][1].

III. Learning Robot Geometry as Distance Fields

In this section, we present our approach to represent the robot geometry as distance fields. Specifically, we first encode the SDF of each robot link through concatenated Bernstein polynomials and then extend it to the whole body based on the robot kinematic chain.

A. Problem Notations

Let $\mathcal{R}(\boldsymbol{q}) \subset \mathbb{R}^3$ be a robot in the 3D Euclidean space at the configuration $\boldsymbol{q}$, and $\partial\mathcal{R}(\boldsymbol{q})$ denotes the surface of $\mathcal{R}$. The distance function $f(\boldsymbol{p}, \boldsymbol{q}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ is defined with $f(\boldsymbol{p}, \boldsymbol{q}) = \pm d(\boldsymbol{p}, \partial\mathcal{R}(\boldsymbol{q}))$, where $d(\boldsymbol{p}, \partial\mathcal{R}(\boldsymbol{q})) = \inf_{\boldsymbol{p}' \in \partial\mathcal{R}(\boldsymbol{q})} |\boldsymbol{p} - \boldsymbol{p}'|^2$ denotes the minimum distance between the points $\boldsymbol{p} = \{x_1, x_2, x_3\} \in \mathcal{R}^3$ and the robot surface. Signs are assigned to points to guarantee negative values within the robot, positive values outside, and zero at the boundary. The gradient $\nabla f_{\boldsymbol{p}}$ points in the direction of maximum distance increase away from the robot surface. In consequence, the normal $\boldsymbol{n} \in \mathbb{R}^3$ with respect to $\mathcal{R}$ can be defined as $\boldsymbol{n} = \nabla f_{\boldsymbol{p}}$.

B. Kinematic Transformation of SDFs

Consider a robot with C degrees of freedom and K links, characterized by joint angles $\boldsymbol{q} = \{q_1, q_2, \cdots, q_C\}$ and shapes $\boldsymbol{\Omega} = \{\Omega_1, \Omega_2, \cdots, \Omega_K\}$. The distance field to represent the robot geometry is the minimum of all links SDFs, which can be written as

$$f_{\mathcal{R}} = \min(f_{\Omega_1^b}, f_{\Omega_2^b}, \cdots, f_{\Omega_K^b}), \tag{1}$$

[1]Codes are available at https://github.com/yimingli1998/RDF.

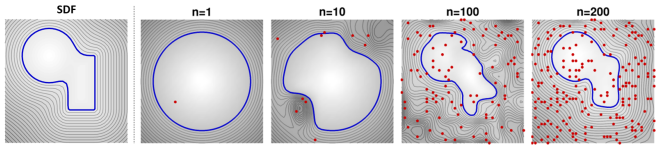


Fig. 2: Illustration of iterative learning for a two-dimensional SDF from samples at different locations. The weights are initialized to resemble a circular object. Red points are sequentially sampled for weight updates. The contour of the estimated object shape is depicted by the blue curve (0-level set of the SDF).

where $f_{\Omega_k^b}$ is the SDF of link Ω_k in the robot base frame.[2] The SDF value for point $\boldsymbol{p}$ in the robot base frame $f_{\Omega_k^b}$ can be computed through the rigid transformation of SDFs [15], which involves transforming the query points as

$$f_{\Omega_k^b}(\boldsymbol{p}, \boldsymbol{q}) = f_{\Omega_k}\Big({}^b\mathcal{T}_k^{-1}(\boldsymbol{q})\boldsymbol{p}\Big), \tag{2}$$

where ${}^b\mathcal{T}_k(\boldsymbol{q}) \in \mathbb{SE}(3)$ denotes a matrix dependent on $\boldsymbol{q}$ that performs the transformation from the frame of the k-th link to the base frame of the robot. The computation of these transformation matrices can be achieved using the kinematics chain of the robot, typically represented by Denavit-Hartenberg parameters [24].

C. Representing SDFs using Bernstein polynomials

Basis functions have been widely used in encoding trajectories in robotics, such as in dynamical movement primitives (DMP) [25] or probabilistic movement primitives (ProMP) [26], see [27] for a review. They provide a continuous, differentiable, and smooth representation of the trajectory, ensuring the encoded motion appears natural without abrupt changes. This compact parameterization also enables efficient storage and computation while accurately capturing complex motions.

Drawing inspiration from these studies, we propose the adoption of geometric primitives, a three-dimensional extension of basis functions, to represent the SDF of each link of the robot. By leveraging basis functions with multivariate inputs, we aim to preserve the aforementioned advantages. In this work, we employ Bernstein polynomials, however, other types of basis functions could alternatively be considered, such as Radial Basis Functions (RBF) for infinite differentiability or Fourier basis functions for multiresolution encoding, see [27] for a review.

The SDF f_{Ω_k} for a point $\boldsymbol{p}^k$ described in the frame of the robot link Ω_k can be represented as a weighted combination of N basis functions as

$$f_{\Omega_k}(\boldsymbol{p}^k) = \langle \boldsymbol{\Psi}_{\boldsymbol{p}^k}, \boldsymbol{w}_k \rangle, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product operation, $\boldsymbol{\Psi}_{\boldsymbol{p}^k} \in \mathbb{R}^{1\times N^3}$ is the matrix of basis functions, and $\boldsymbol{w}_k \in \mathbb{R}^{N^3\times 1}$ is the matrix of superposition weights corresponding to the k-th link. We hereby omit the variables k and Ω_k in order to enhance the readability of the text. We define $\boldsymbol{\Psi}_{\boldsymbol{p}} = \boldsymbol{\phi}(t_1) \otimes \boldsymbol{\phi}(t_2) \otimes \boldsymbol{\phi}(t_3)$ using the Kronecker product $\otimes$, where $t_i = \frac{x_i - x_i^{\min}}{x_i^{\max} - x_i^{\min}}$ is the normalized version of x_i, and reshape it to a row vector. $\boldsymbol{\phi}(t) \in \mathbb{R}^N$ is the vector of basis functions whose n-th element, for Bernstein polynomials, is given by

$$\phi_n(t) = \binom{N-1}{n} t^n (1-t)^{N-1-n}, \quad \forall n \in \{0, \cdots, N-1\}, \tag{4}$$

in analytic form, where $t \in [0, 1]$ is a normalized location of the point. Consequently, the derivative of the n-th basis function can be expressed as

$$\nabla_t \phi_n(t) = \tbinom{N-1}{n}(1-t)^{N-n-2} t^{n-1}\Big(n(1-t) - (N-n-1)t\Big), \tag{5}$$

and the derivatives of $\boldsymbol{\Psi}$ are analytically given by

$$\begin{aligned} \nabla_{t_1}\boldsymbol{\Psi} &= \nabla_{t_1}\boldsymbol{\phi}(t_1) \otimes \boldsymbol{\phi}(t_2) \otimes \boldsymbol{\phi}(t_3), \\ \nabla_{t_2}\boldsymbol{\Psi} &= \boldsymbol{\phi}(t_1) \otimes \nabla_{t_2}\boldsymbol{\phi}(t_2) \otimes \boldsymbol{\phi}(t_3), \\ \nabla_{t_3}\boldsymbol{\Psi} &= \boldsymbol{\phi}(t_1) \otimes \boldsymbol{\phi}(t_2) \otimes \nabla_{t_3}\boldsymbol{\phi}(t_3). \end{aligned} \tag{6}$$

For T data points denoted as $\boldsymbol{P} \in \mathbb{R}^{T\times 3}$ and their corresponding distance values denoted as $\boldsymbol{f} \in \mathbb{R}^T$, the weight tensor $\boldsymbol{w}$ can be learned through least square regression as $\boldsymbol{w}^* = (\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\boldsymbol{f}$, where $\boldsymbol{\Psi} \in \mathbb{R}^{T\times N^3}$ contains all basis functions and points in a concatenated form. Computing the inverse of large matrices can be computationally expensive and suffer from memory issues. Instead of a batch evaluation, a recursive formulation can be used, providing exactly the same result [28], [29]. To do so, we define a new parameter $\boldsymbol{B} = (\boldsymbol{\Psi}^\top\boldsymbol{\Psi})^{-1}$ and process the data sequentially by sampling a small batch of points $\{\tilde{\boldsymbol{P}}, \tilde{\boldsymbol{f}}\}$ and updating the learned weights when new data points become available. The whole process is depicted in Algorithm 1, and a 2D example is shown in Fig. 2. After obtaining the optimal weights $\boldsymbol{w}^*$, the distance can be decoded efficiently with (3) during inference. Similarly, this efficiency also extends the gradients, thanks to the analytic form of the polynomial structure[3].

Algorithm 1 Recursive learning of superposition weights

initialize $\boldsymbol{B}_0 = \frac{1}{\lambda}\boldsymbol{I}$, $\boldsymbol{w} = \boldsymbol{w}_0$;
for $m \leftarrow 1$ *to* M **do**
  Given new mini-batch data points: $\{\tilde{\boldsymbol{P}}, \tilde{\boldsymbol{f}}\}$
  Encode points with Bernstein polynomials: $\tilde{\boldsymbol{\Psi}} = \boldsymbol{\Psi}(\tilde{\boldsymbol{P}})$
  Compute Kalman gain:
    $\boldsymbol{K}_m = \boldsymbol{B}_{m-1}\tilde{\boldsymbol{\Psi}}^\top(\boldsymbol{I} + \tilde{\boldsymbol{\Psi}}\boldsymbol{B}_{m-1}\tilde{\boldsymbol{\Psi}}_m^\top)^{-1}$
  Update $\boldsymbol{B}_m$: $\boldsymbol{B}_m = \boldsymbol{B}_{m-1} - \boldsymbol{K}_m\tilde{\boldsymbol{\Psi}}\boldsymbol{B}_{m-1}$
  Update $\boldsymbol{w}_m$: $\boldsymbol{w}_m = \boldsymbol{w}_{m-1} + \boldsymbol{K}_m(\tilde{\boldsymbol{f}} - \tilde{\boldsymbol{\Psi}}\boldsymbol{w}_{m-1})$
end
return $\boldsymbol{w}^* \leftarrow \boldsymbol{w}_M$

IV. Numerical Experiments

To demonstrate the effectiveness of the proposed method, we conduct several numerical comparisons against baseline methods.

[2]The $\min(\cdot)$ in (1) might cause discontinuous gradient when the closest link changes. A differentiable smooth version of this function can be utilized to avoid this issue.

[3]We refer readers to https://rcfs.ch/ for details about basis functions encoding with multidimensional inputs.

TABLE I: Comparison of baseline methods for representing SDFs

Methods	CD, mean (mm)	CD, max (mm)	Inference Time (ms)	Model size (MB)
TT-SVD	**0.22**	23.3	-	3.8
NN (2.56×10^5 points)	1.86	32.4	0.25	2.4
NN (2.56×10^6 points)	0.57	14.4	0.25	2.4
BP (N=8)	0.91	21.8	**0.21**	**0.024**
BP (N=24)	0.40	**12.6**	0.54	0.49

Implement details. We build the distance field for the Franka Emika Robot with 7 articulations and 9 links (the fingertips of the gripper are ignored). The superposition weights of Bernstein polynomials are separately trained for each robot link. Specifically, we assume a cubic volume around each link to sample training data. The positions of points inside the volume are normalized to $[0, 1]$ and points outside the volume are projected onto the boundary and the distance is approximated by summing the distances from the projected point to the boundary. All operations are implemented with the batch operation and run on an Nvidia GeForce RTX3060 GPU. The training data is generated following DeepSDF [4][4].

Effectiveness of basis functions in encoding SDFs. We first compare the proposed Bernstein Polynomial (BP) method with two other representative state-of-the-art approaches: a volumetric-based method, TT-SVD [30] which utilizes tensor decomposition to compress voxelized SDFs, and a neural network (NN) based method [4], [23]. We evaluate the Chamfer Distance (CD) [4], inference time and model size. For the TT-SVD method, we set the maximum rank R to 40. For the neural network, we found it could not represent the SDF well with the same number of data points we used (2.56×10^5), so we additionally trained it with 10 times more data for a more detailed comparison. We report the result of our lightweight model (with 8 basis functions) and precise model (with 24 basis functions) in Table I. Our approach shows competitive accuracy and efficiency with a more compact structure. Although TT-SVD shows a lower mean CD, it exhibits a higher max CD, indicating sensitivity to high-frequency data. Besides, it represents discrete SDFs while our method is continuous and differentiable. We find BP and NN can encode the shape of the robot links accurately with similar CD. However, our method based on recursive ridge regression shows higher data efficiency. Additionally, our approach offers other benefits. The weights learned by BP correspond to the key points, which directly provide interpretable and controllable parameters. Besides, it also provides a simple and efficient way to compute analytical gradients by directly leveraging the derivatives of the basis functions. The continuity and smoothness of the gradient are also guaranteed by construction.

Quality of RDF. We further compare our approach with several common approaches to represent the geometry of the robot.

Spheres have closed-form signed distance functions and we use 55 spheres to approximate the robot. For meshes, we compute the signed distance by finding the closest vertex and normal, which is another widely used approach. The coarse mesh has 1,249 vertices while the precise mesh has 74,647 vertices. Following Neural-JSDF [6], we report the mean absolute error (MAE) and root mean square error (RMSE) for points near the robot surface (within 0.03m) and points far away (over 0.03m) in Table II.

[4]we use the library mesh_to_sdf for implementation.

TABLE II: Comparison with baselines for the produced distance field. Errors are presented in millimeters (mm).

	Points Near		Points Far		Average		Time
	MAE	RMSE	MAE	RMSE	MAE	RMSE	(ms)
Sphere-based	6.45	11.3	5.49	9.49	5.91	10.4	2.2
Mesh-based (coarse)	13.6	19.2	6.57	16.8	9.70	18.0	2.9
Mesh-based (precise)	4.09	10.21	1.79	11.78	2.82	11.2	8.7
Neural-JSDF	28.2	31.6	18.7	23.4	23.0	27.4	**0.25**
NN + K.C.	1.74	**3.57**	1.30	2.93	1.50	3.24	4.7
BP (N=8) + K.C.	2.85	4.55	2.35	3.93	2.57	4.22	2.4
BP (N=24) + K.C.	**1.71**	3.59	**1.18**	**2.87**	**1.41**	**3.23**	5.8

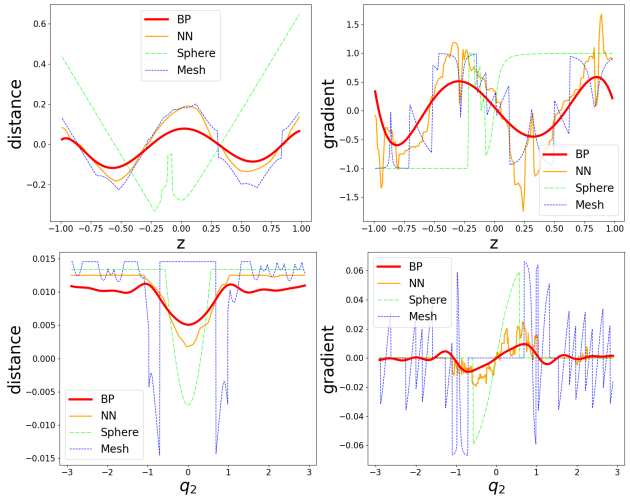


Fig. 3: We show the smoothness of distance and gradient produced by our approach, in both task space (a) and joint space (b), with comparisons to several baselines. The distance and gradient from point $t = [0, 0, z]$ to the surface of link5 with a specific joint are shown in (a). The distance and gradient from a specific point to the robot surface at joint $q = [0, q_2, 0, 0, 0, 0, 0]$ is shown in (b).

The comparison between Neural-JSDF and other approaches demonstrates the importance of the kinematic chain (K.C.) in modeling accurate RDF. With our non-optimized implementation, the computation time is higher for our method, but it is still at the millisecond level, allowing real-time behavior with high frequency. Although coarse mesh has a more precise shape than spheres, it still fails to represent an accurate distance field, since the choice of closest point and normal estimation are usually inaccurate and noisy. Methods incorporating kinematic chain and SDF (NN + K.C. and BP + K.C.) improve the accuracy compared to primitive-based and mesh-based methods. The average MAE for these methods is about 1mm, which is accurate enough for tasks that require establishing contacts with the environment. Figure 3 also shows the distance and gradient produced by several methods, highlighting the smoothness of our approach.

V. Robot Experiments

In this section, we illustrate the effectiveness of our RDF representation through two dual-arm robot tasks: 1) *Collision Avoidance*: While a robot arm tries to reach a target, it

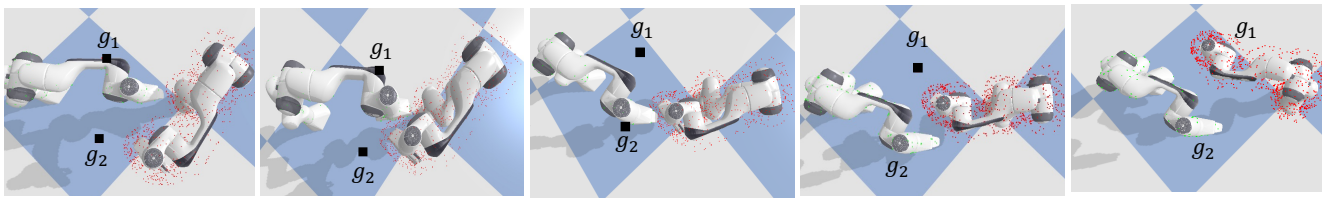


Fig. 4: Collision avoidance experiment in simulation. g_1 and g_2 represent the target points. Red points on the right arm are sampled with the level set $f = 0.05$ to represent the safety threshold surface.

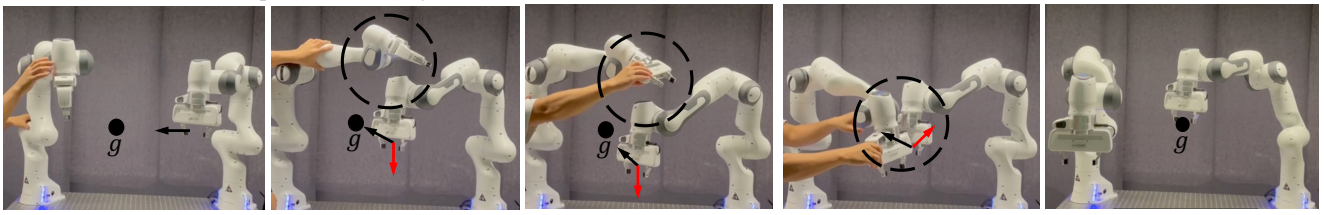


Fig. 5: Real-world collision avoidance experiment. Here, g is the target point for the right arm. Red/black arrows show the reaching velocity with/without collision avoidance. Black dashed circles show the potential collision area.

TABLE III: Results for collision avoidance in simulation.

Methods	Reaching	QP no solution	Time cost (ms)
Mesh-based	63%	37%	7.68
NN	74%	23%	11.34
Sphere-based	84%	**12%**	**4.18**
BP (N=8)	**87%**	**12%**	5.99

must avoid colliding with another. 2) *Dual-arm Lifting*: Two robot arms collaborate to lift a large box that cannot be grasped conventionally. The objective is to plan a pair of joint configurations for both arms such that they can establish contact with the box by exploiting their whole bodies to reach and lift the box.

A. Collision Avoidance

In this section, we integrate the learned distance fields for collision avoidance, which is crucial in motion planning tasks. Specifically, we exploit an augmented quadratic Programming (QP) algorithm [31] to ensure self-collision avoidance between two robot arms during task execution.

The self-collision avoidance experiments are conducted in both simulation and real-world scenarios. In simulation, the goal for both arms is to reach their respective target position while the right arm should actively avoid collision with the left arm. The real-world experiment is conducted with a reactive controller, where the left arm is manually moved by a human operator in gravity-compensated mode, serving as a dynamic obstacle for the right arm. For both experiments, we randomly sampled 256 points on the surface of the left arm as the input of RDF for the right arm and then used the minimal distance produced for self-collision avoidance.

We conducted simulation experiments 100 times, utilizing different initial states for both robot arms, and compared our proposed method with sphere-based, mesh-based and NN-based representations. Results are presented in Table III. The time cost represents the average time for solving the QP problem once. For Neural-JSDF, we observed large distance errors with unsuccessful results.

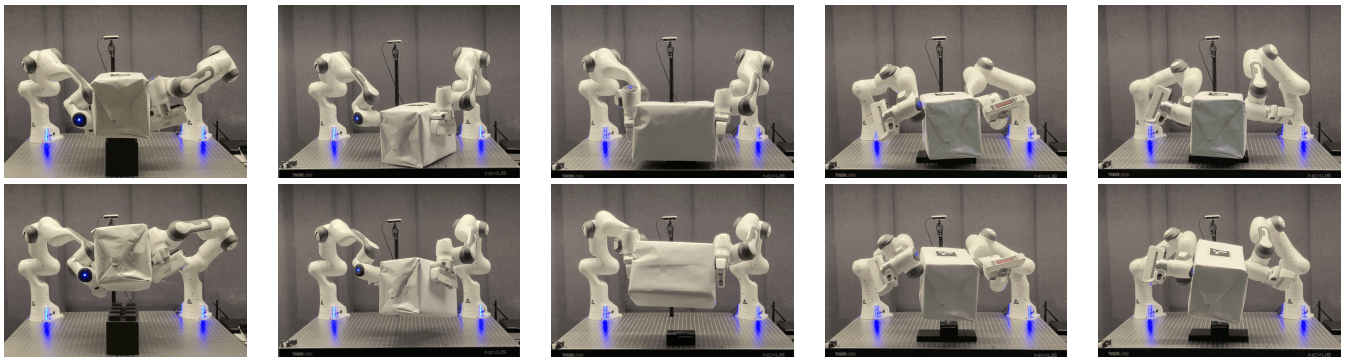
As collision avoidance was established as a hard constraint in the QP controller, all methods exhibited collision-free behavior whenever the QP solver converged to a solution. Nevertheless, attributing to the accuracy and smoothness, our method demonstrated the highest success rate (87%) in reaching, as well as the lowest probability (12%) of not finding a solution for the QP solver, which also led to a shorter planning time. The left 1% case is that the QP solver found a solution but the two arms blocked each other. The poor performance of mesh-based representation and neural networks indicates the importance of continuous gradient, which makes the optimizer find solutions more easily and more efficiently. Figures 4 and 5 depict the collision avoidance process in simulation and real-world, showing our method enables the robot arm to respond to the environment and avoid collisions (see also accompanying video).

B. Dual-arm Lifting

In this experiment, our focus is on the manipulation of a large box using a dual manipulator, utilizing the whole surface of the last four links of the robot. Our underlying assumption is that the contact points on the object are already predetermined, and the robot has the freedom to establish contacts automatically based on the SDF representation without sampling any surface points on the robot. The task can be viewed as an optimization problem whose quadratic cost function can be defined as $c(\boldsymbol{q}) = \boldsymbol{r}^\top \boldsymbol{r}$, where $\boldsymbol{r} = [\boldsymbol{r}_r, \boldsymbol{r}_c, \boldsymbol{r}_j^{\max}, \boldsymbol{r}_j^{\min}, \boldsymbol{r}_j^d]^\top$ is the residual vector consisting of several elements: a reaching residual $\boldsymbol{r}_r$ to establish a contact between the robot arms and the object, a penetration residual $\boldsymbol{r}_p$ for collision avoidance, a joint distance residual $\boldsymbol{r}_j^d$ to regularize the solution near the robot's initial configuration, and joint limit residuals $\boldsymbol{r}_j^{\max}$ and $\boldsymbol{r}_j^{\min}$ to consider joint angle limits, defined as

$$\begin{aligned} \boldsymbol{r}_r = \boldsymbol{f}(\boldsymbol{p}_c, \boldsymbol{q}), \quad \boldsymbol{r}_p = \mathrm{ReLU}(-\boldsymbol{f}(\boldsymbol{p}_i, \boldsymbol{q})), \\ \boldsymbol{r}_j^{\max} = \mathrm{ReLU}(\boldsymbol{q} - \boldsymbol{q}_{\max}), \quad \boldsymbol{r}_j^{\min} = \mathrm{ReLU}(\boldsymbol{q}_{\min} - \boldsymbol{q}), \\ \boldsymbol{r}_j^d = \boldsymbol{q} - \boldsymbol{q}_{\text{init}}, \end{aligned} \tag{7}$$

where $\boldsymbol{f}(\boldsymbol{p}, \boldsymbol{q}) \in \mathbb{R}^T$ represents the spatial distance between points $\boldsymbol{p}$ and the robot surface at configuration $\boldsymbol{q}$. In this context, $\boldsymbol{p}_c$ represents predefined contact points on the object,



Exp. 1 Exp. 2 Exp. 3 Exp. 4 Exp. 5

Fig. 6: Robot experiments for whole-body dual-arm lifting. *Top row:* the planned joint configurations for grasping the box. *Bottom row:* the final states after lifting.

TABLE IV: Results for dual-arm lifting task.

Methods	Sphere-based	BP (N=8)
Success Rate	36%	**77%**
Time (per valid configuration)	0.98	**0.46**

while $\boldsymbol{p}_i$ denotes the points uniformly selected within the box for collision avoidance purposes. $\boldsymbol{q}_{\text{min}}, \boldsymbol{q}_{\text{max}}$ are the physical joint limits and $\boldsymbol{q}_{\text{init}}$ is the robot initial joint configuration. The optimization is solved using the Gauss-Newton algorithm as

$$\boldsymbol{q} = \boldsymbol{q} - \alpha \boldsymbol{J}^{\dagger} \boldsymbol{r} = \boldsymbol{q} - \alpha (\boldsymbol{J}^{\top} \boldsymbol{J})^{-1} \boldsymbol{J}^{\top} \boldsymbol{r}, \tag{8}$$

where $\boldsymbol{J} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}}$ is the Jacobian matrix and α is a line search parameter. The algorithm is terminated when satisfying criteria $\boldsymbol{r}_r^{\top} \boldsymbol{r}_r < 0.01$, and $\boldsymbol{r}_p^{\top} \boldsymbol{r}_p < 0.01$, and $q_{\text{min},c} < q_c < q_{\text{max},c}, \ \forall c \in \{1, \dots, C\}$, and $\sum_{\boldsymbol{p}_c} (1 - \langle \text{norm}(\frac{\partial f(\boldsymbol{p}_c, \boldsymbol{q})}{\partial \boldsymbol{p}_c}), \boldsymbol{n}_c \rangle) < 0.1$ for normal constraints, where $\boldsymbol{n_c}$ is the normal direction on the contact point. We optimize the problem in batch to accelerate the planning procedure with random initialized configurations. Trajectories from initial to goal configurations are interpolated through cubic splines. A joint impedance controller is adopted in conjunction with a smaller desired box size during the planning phase to generate sufficient force at contact points. The lifting action is accomplished by elevating the fourth joint of the robot, which is positioned immediately before the potential contact links.

We report the success rate among 50 planned joint configurations for both arms and the average planning time in Table IV. A configuration is considered successful if it respects all the termination conditions with the exact robot model. Since batch optimization is usually accompanied by a larger memory overhead, we only selected the sphere-based method and our lightweight model for comparison. Our method shows significant improvement in terms of both success rate and computation time, which is attributed to the more accurate robot model compared to the sphere-based representation.

The experimental results of the real robot implementation are presented in Fig. 6. In Experiments 1-3, the robot exhibits the capability to use its last four links to contact the object. These experiments provide empirical evidence of the generalization capability of the method across various poses. In experiments 4 and 5, the robot is constrained to utilize specific links for contacts. Specifically, the contact is limited to the sixth link in experiment 4, while in experiment 5, it is restricted to the seventh link. This restriction narrows down the valid solutions, requiring the robot to adapt its approach accordingly. The optimization problem is still able to find appropriate solutions. It can be attributed to the infinite resolution of the robot arm and the smooth representation provided by the distance field, which enables the optimization algorithm to navigate the constrained search space more effectively, leading to successful solutions even in scenarios with limited contact options.

VI. Conclusion

In this paper, we proposed a novel approach to represent the geometry of a robot as distance fields. We leveraged the kinematic structure of the robot to generalize configuration-agnostic signed distance functions that remain valid for arbitrary robot configurations, which enables more effective learning and more accurate inference of distance fields. The SDF for each link of the robot is represented by a combination of piecewise multivariate polynomials, ensuring interpretability, compactness and smoothness while remaining competitive in terms of efficiency and accuracy. The approach provides analytic derivatives that can directly be used for gradient-based (or higher-order) optimization techniques. Experiments in collision avoidance have shown the effectiveness of our representation. Furthermore, we have demonstrated how to integrate this representation into whole-body manipulation tasks, by defining cost functions based on the SDF of the robot.

There are some limitations that should be acknowledged. First, the capability of basis functions to highly complex shapes has not been thoroughly investigated. Secondly, we simplified the lifting task by planning joint configurations, without considering the dynamic model. Finally, the representation could be further applied to other complex manipulation tasks, such as pushing and pivoting, by estimating the interaction forces between SDFs and formulating it as an optimization problem. We plan to explore this research direction in future work.

REFERENCES

[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[2] S. Zimmermann, M. Busenhart, S. Huber, R. Poranne, and S. Coros, "Differentiable collision avoidance using collision primitives," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8086–8093.

[3] 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*, 1996, pp. 303–312.

[4] 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.

[5] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, and G. Chalvatzaki, "Regularized deep signed distance fields for reactive motion generation," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6673–6680.

[6] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2022.

[7] M. Macklin, K. Erleben, M. Müller, 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.

[8] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 67–76.

[9] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "isdf: Real-time neural signed distance fields for robot perception," *arXiv preprint arXiv:2204.02296*, 2022.

[10] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 559–568.

[11] M. Breyer, J. J. Chung, L. Ott, R. Siegwart, and J. Nieto, "Volumetric grasping network: Real-time 6 dof grasp detection in clutter," in *Proc. Conference on Robot Learning (CoRL)*. PMLR, 2021, pp. 1602–1611.

[12] Z. Jiang, Y. Zhu, M. Svetlik, K. Fang, and Y. Zhu, "Synergies between affordance and geometry: 6-dof grasp detection via implicit representations," *arXiv preprint arXiv:2104.01542*, 2021.

[13] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox, "Object rearrangement using learned implicit collision functions," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6010–6017.

[14] T. Liu, Z. Liu, Z. Jiao, Y. Zhu, and S.-C. Zhu, "Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 470–477, 2021.

[15] D. Driess, J.-S. Ha, M. Toussaint, and R. Tedrake, "Learning models as functionals of signed-distance fields for manipulation planning," in *Proc. Conference on Robot Learning (CoRL)*. PMLR, 2022, pp. 245–255.

[16] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[17] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.

[18] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 489–494.

[19] T. Schmidt, R. A. Newcombe, and D. Fox, "Dart: Dense articulated real-time tracking." in *Robotics: Science and systems*, vol. 2, no. 1. Berkeley, CA, 2014, pp. 1–9.

[20] G. Sutanto, I. R. Fernández, 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.

[21] V. Vasilopoulos, S. Garg, P. Piacenza, J. Huh, and V. Isler, "Ramp: Hierarchical reactive motion planning for manipulation tasks using implicit signed distance functions," *arXiv preprint arXiv:2305.10534*, 2023.

[22] J. Michaux, Q. Chen, Y. Kwon, and R. Vasudevan, "Reachability-based trajectory design with neural implicit safety constraints," *arXiv preprint arXiv:2302.07352*, 2023.

[23] 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," *arXiv preprint arXiv:2306.04130*, 2023.

[24] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," 1955.

[25] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models formotor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[26] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NeurIPS)*, C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013.

[27] S. Calinon, "Mixture models for the analysis, edition, and synthesis of continuous time series," in *Mixture Models and Applications*, N. Bouguila and W. Fan, Eds. Springer, Cham, 2019, pp. 39–57.

[28] W. W. Hager, "Updating the inverse of a matrix," *SIAM Review*, vol. 31, no. 2, pp. 221–239, 1989.

[29] J.-A. Ting, S. Vijayakumar, and S. Schaal, *Locally weighted regression for control*. Springer, 2010, pp. 613–624.

[30] A. I. Boyko, M. P. Matrosov, I. V. Oseledets, D. Tsetserukou, and G. Ferrer, "Tt-tsdf: Memory-efficient tsdf with low-rank tensor train decomposition," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 116–10 121.

[31] J. Haviland and P. Corke, "Manipulator differential kinematics: Part 2: Acceleration and advanced applications," *IEEE Robotics & Automation Magazine*, 2023.