Configuration Space Distance Fields for Manipulation Planning

Yiming Li[1,2], Xuemin Chi[1,3], Amirreza Razmjoo[1,2], and Sylvain Calinon[1,2]
[1]Idiap Research Institute [2]EPFL [3]Zhejiang University

***Abstract*—The signed distance field (SDF) is a popular implicit shape representation in robotics, providing geometric information about objects and obstacles in a form that can easily be combined with control, optimization and learning techniques. Most often, SDFs are used to represent distances in task space, which corresponds to the familiar notion of distances that we perceive in our 3D world. However, SDFs can mathematically be used in other spaces, including robot configuration spaces. For a robot manipulator, this configuration space typically corresponds to the joint angles for each articulation of the robot. While it is customary in robot planning to express which portions of the configuration space are free from collision with obstacles, it is less common to think of this information as a distance field in the configuration space. In this paper, we demonstrate the potential of considering SDFs in the robot configuration space for optimization, which we call the configuration space distance field (or CDF for short). Similarly to the use of SDF in task space, CDF provides an efficient joint angle distance query and direct access to the derivatives (joint angle velocity). Most approaches split the overall computation with one part in task space followed by one part in configuration space (evaluating distances in task space and then computing actions with inverse kinematics). Instead, CDF allows the implicit structure to be leveraged by control, optimization, and learning problems in a unified manner. In particular, we propose an efficient algorithm to compute and fuse CDFs that can be generalized to arbitrary scenes. A corresponding neural CDF representation using multilayer perceptrons (MLPs) is also presented to obtain a compact and continuous representation while improving computation efficiency. We demonstrate the effectiveness of CDF with planar obstacle avoidance examples and with a 7-axis Franka robot in inverse kinematics and manipulation planning tasks. Project page: https://sites.google.com/view/cdfmp/home**

I. Introduction

Distances are the most fundamental and intuitive metrics for expressing the interrelation among multiple variables. In robotics, they are typically used to measure the geometric relationship among diverse representations, such as points, poses, trajectories, surfaces and shapes, which are exploited in various tasks including inverse kinematics [22, 4] and manipulation planning [18]. The signed distance field (SDF) representation has become a popular representation, that can for example be used to encode the Euclidean distance from a point to an object boundary. The differentiability and unit norm gradient properties make it easy to integrate into learning [41, 6, 37], optimization [27, 28, 36], and control [5, 26, 17].

SDFs are conventionally employed in 3D task spaces. In the context of manipulation, a typical control task is composed of two steps, first using an SDF in task space to evaluate the distance to the object, followed by an inverse kinematics step to find the joint angle configuration that can reduce this distance. Because of the nonlinear mapping between task space and joint space, this problem is typically solved with a few iterations by second-order optimization (corresponding to the use of a Jacobian pseudoinverse at each iteration).

Figure 1-*left* depicts this process visualized in the configuration space, where the full distance field has been computed to be depicted as colored level sets (in practice, we evaluate the forward kinematics function and associated Jacobian only at the current joint configuration of the robot).

While conventionally employed in 3D task spaces, SDFs can be considered in other spaces, including robot joint configuration spaces in which planning and control problems take place. We see in Figure 1-*left* that when we transpose an SDF from task space to configuration space, the property of unit norm gradient disappears, while in Figure 1-*right*, this property is maintained. We will refer to this approach as a Configuration Space Distance Field (CDF), a scalar field measuring the angular distance between joint angles and the object geometry in configuration space (Fig. 1-*right*). For manipulation tasks, CDF directly estimates the minimum joint motion required by the robot to establish contact with an object, with gradients consistently pointing toward the object. Unlike SDFs in task space, CDF is directly formulated in configuration space, preserving the Euclidean property of the distance field, ensuring a uniform span of distances and maintaining unit magnitudes in gradient directions.

CDF offers several advantages. It naturally bridges task space and configuration space, providing a unified approach to solving problems that traditionally involve computation in separate spaces. For instance, the inverse kinematics problem is usually solved by evaluating the task space distance and then computing joint space motions. In contrast, CDF solves this problem through one-step gradient projections, avoiding Gauss-Newton iterations. It also offers intuitive geodesics that reflect object geometry in configuration space, see Fig. 1. Furthermore, CDF inherits the merits of SDFs, including implicit structure, Boolean operations for composition involving multiple SDFs, efficient queries, and differentiability. We also propose a neural variant called neural CDF. Analogously to neural SDFs, neural CDFs also offer a compact, continuous, analytical, and latent space representation, thereby facilitating seamless integration into learning, optimization, and control frameworks. Most approaches developed for SDFs can be directly applied to CDF, which directly solves problems in configuration space.

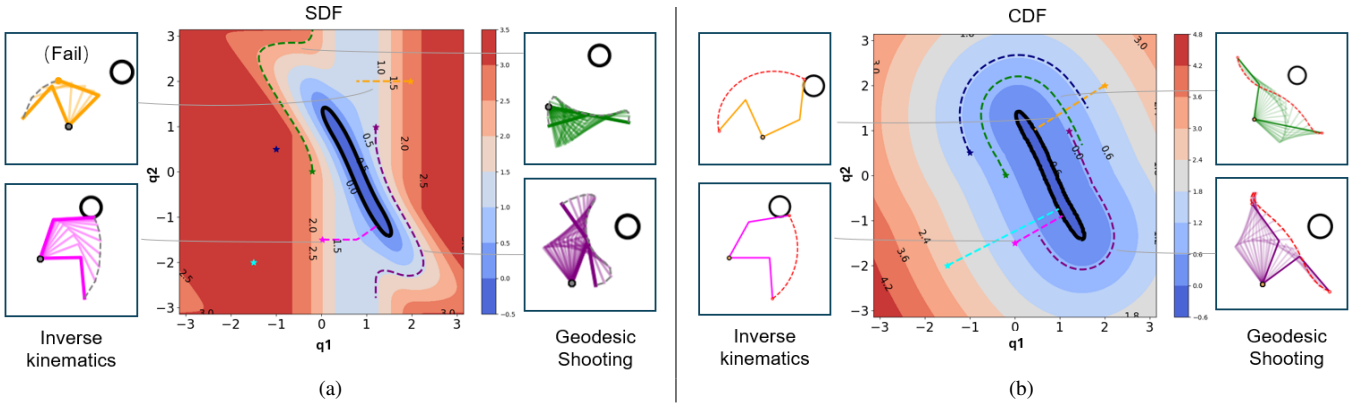


Fig. 1: Differences between SDF and CDF. The colored level sets depict the distances to the object, where the black contours represent joint angles leading to robot-object surface contact. The zero-level-set of SDF and CDF is the same, but the other level sets of CDF are characterized by evenly expanded distances and unit norm gradient. This property leads to gradient projection that can directly be computed, which solves the inverse kinematics problem for the contact task in one step (see trajectories in cyan and pink), whereas SDF requires multiple iterations while encountering singularities, which can even fail due to gradient vanishing (see trajectory in yellow). Geodesics on the CDF naturally wrap around the shape of the object in configuration space (see trajectories in blue, green and purple).

The organization of this paper is as follows. Section II reviews the related work. Section III discusses the formulation, properties, computation and fusion strategy of CDF. Section IV presents the implicit neural CDF representation. Section V and VI demonstrate the effectiveness of CDF in whole-body inverse kinematics and manipulation planning tasks. Section VII discusses and concludes the paper. The primary contributions of this paper are:

- The introduction of the CDF representation, offering a unified framework for addressing robot manipulation challenges within configuration space.
- An efficient algorithm for calculating the zero-level-set geometry of the object in configuration space, corresponding to the set of robot configurations that will lead to a collision. A subsequent fusion strategy is also proposed to combine multiple CDFs online, enabling the generalization of CDF to diverse scenes.
- A neural CDF variant utilizing a multilayer perceptron (MLP), together with the design of the corresponding loss functions, resulting in a concise and continuous representation. This variation provides trade-offs across efficiency, accuracy, and compression capabilities while keeping a simple and flexible structure.
- Experiments comparing CDF with SDFs and their derivatives on a planar robot and a 7-axis Franka robot. The conducted experiments highlight the efficiency of CDF in solving inverse kinematics tasks through gradient projection and its effectiveness in addressing manipulation planning challenges, leading to the generation of natural robotic motions.

II. Related Work

SDFs have garnered extensive attention in the field of computer version and graphics, particularly in shape encoding [25, 9], mesh generation [31], and differentiable rendering [39, 20]. Their efficacy in distance and gradient queries has made them popular in robotics, with applications in planning [27, 45], mapping [10], and manipulation [6]. Recent studies proposed to encode SDF with joint angles [14], or learn an SDF representation of the swept volume of robot manipulators [19], to model the SDF of articulated robots and apply it to manipulation tasks [16].

The prevailing focus on SDFs in robotics centers on task space, where configuration space actions are typically computed independently through mappings between the two spaces [33, 29]. Existing approaches often model the configuration space using binary maps denoting collision status of joint configurations [32, 42], to support sample-based motion planning algorithms [43, 3, 11]. Despite significant progress, these control and planning strategies are computationally expensive in high dimensional space due to the lack of gradient information.

In contrast, considering a distance field in the configuration space introduces new control and planning strategies, by shifting the focus from conventional binary collision masks to continuous and structured representations. For instance, with this approach, the inverse kinematics problem simplifies to an SDF pullback in configuration space, requiring only one-step gradient projection. More generally, CDF enables the transposition of SDF methodologies developed for task space to configuration space. The computation can be viewed as a point-mass system, while obstacles form topological holes in configuration space and geodesics produce natural curved paths around them [28]. The approach can be extended to geometric motion planning frameworks, including Riemannian motion policies [29], geometric fabrics [38], and dynamic-aware motion optimization by assigning metrics on the distance field [13, 1].

III. Configuration Space Distance Field (CDF)

In this section, we introduce CDF and delve into its properties. We then present an efficient algorithm to compute CDF, as well as a fusion strategy for online combination of multiple CDFs.

Algorithm 1 Finding 0 level-set configurations

Input: point $\boldsymbol{p}$, robot SDF model f_s
Output: joint configuration $\boldsymbol{q}'$ that satisfies $f_s(\boldsymbol{p}, \boldsymbol{q}') = \boldsymbol{0}$
Initialization: $\boldsymbol{q} \leftarrow \boldsymbol{q_0}$ ▷ Batch initialization
for $t = 1, \ldots, T$ ▷ T iterations
  $c \leftarrow c(\boldsymbol{q})$ ▷ Compute cost
  $\delta\boldsymbol{q} \leftarrow -\boldsymbol{H}^{-1}\nabla_{\boldsymbol{q}} c$ ▷ Batch L-BFGS update
  $\boldsymbol{q} \leftarrow \boldsymbol{q} + \boldsymbol{\alpha}\delta\boldsymbol{q}$ ▷ Line search
end
$\boldsymbol{q}' \leftarrow \boldsymbol{q} : f_s(\boldsymbol{p}, \boldsymbol{q}) < \epsilon$ ▷ Return final configurations

A. Problem Formulation

CDF is inspired by recent work encoding SDF with robot joint configurations [17, 14, 16]. Let $r(\boldsymbol{q})$ denote a robot at configuration $\boldsymbol{q} \in \mathbb{R}^n$ and $\boldsymbol{p} \in \mathbb{R}^3$ be a point set in the robot workspace, for a robot with n degrees of freedom (DoF). The robot SDF f_s is a function of $\boldsymbol{p}$ and $\boldsymbol{q}$ that measures the distance from $\boldsymbol{p}$ to the closest point on the robot surface $\partial r(\boldsymbol{q})$[1]:

$$f_s(\boldsymbol{p}, \boldsymbol{q}) = \pm \min_{\boldsymbol{p}' \in \partial r(\boldsymbol{q})} \|\boldsymbol{p} - \boldsymbol{p}'\|, \tag{1}$$

where $\pm$ indicates the sign of the distance, which is positive if $\boldsymbol{p}$ is outside, zero on the surface, and negative otherwise. The differentiability of the robot SDF with respect to both $\boldsymbol{p}$ and $\boldsymbol{q}$ enables various gradient-based manipulation planning tasks.

The robot SDF representation encodes the robot geometry through forward kinematics, where the distance is Euclidean in the workspace but highly nonlinear in configuration space. In contrast to using task space distances, CDF is defined as a function f_c that measures the minimal distance in radians from $\boldsymbol{q}$ to zero-level-set joint configurations $\boldsymbol{q}' : f_s(\boldsymbol{p}, \boldsymbol{q}') = \boldsymbol{0}$ at $\boldsymbol{p}$, which would establish contact between the robot and the point:

$$f_c(\boldsymbol{p}, \boldsymbol{q}) = \min_{\boldsymbol{q}'} \|\boldsymbol{q} - \boldsymbol{q}'\|. \tag{2}$$

This distance in radians corresponds to the movement of joint angles, where the constraint $f_s(\boldsymbol{p}, \boldsymbol{q}') = 0$ implicitly solves the inverse kinematics problem by finding the configuration set $\boldsymbol{q}'$ on the zero-level-set of robot SDF model, given a point $\boldsymbol{p}$. CDF is unsigned according to this definition, as we focus more on the value of distance and gradient, where the sign can be determined either by combining it with SDF or by estimating the normal direction on boundary samples. The derivative of CDF with respect to $\boldsymbol{q}$ corresponds to joint velocity.

B. Properties of CDF

An SDF satisfies the eikonal equation $\|\nabla_{\boldsymbol{p}} f_s(\boldsymbol{p}, \boldsymbol{q})\| = 1$ almost everywhere. Thus, the closest point on the robot surface to $\boldsymbol{p}$ can be calculated by projecting $\boldsymbol{p}$ along the gradient direction:

$$\boldsymbol{p}' = \boldsymbol{p} - f_s(\boldsymbol{p}, \boldsymbol{q})\ \nabla_{\boldsymbol{p}} f_s(\boldsymbol{p}, \boldsymbol{q}). \tag{3}$$

[1]All variables support batch operations, i.e. $\boldsymbol{p} \in \mathbb{R}^{b_1 \times 3}$ and $\boldsymbol{q} \in \mathbb{R}^{b_2 \times n}$ accounting for $\boldsymbol{f_s} \in \mathbb{R}^{b_1 \times b_2}$.

For surface points, gradients correspond to normal directions. Similarly, CDF satisfies the eikonal equation $\|\nabla_{\boldsymbol{q}} f_c(\boldsymbol{p}, \boldsymbol{q})\| = 1$ almost everywhere in configuration space. The closest configuration on the zero-level-set manifold can be found by projecting the current configuration along the gradient direction:

$$\boldsymbol{q}' = \boldsymbol{q} - f_c(\boldsymbol{p}, \boldsymbol{q})\ \nabla_{\boldsymbol{q}} f_c(\boldsymbol{p}, \boldsymbol{q}). \tag{4}$$

This property makes CDF useful in manipulation planning tasks. It allows for direct computation of zero-level-set joint configurations through gradient projection, efficiently solving the inverse kinematics problem in one-step computation. For motion generation tasks, this implies having a more structured distance field in the configuration space, where gradients always point toward objects to reach or away from obstacles. Moreover, geodesics in the configuration space will naturally curve around the zero-level-sets, which can for example be used to move around an object while maintaining a constant joint angle distance to the object. From a control perspective, it means that the object remains reachable/avoidable within the robot joint angle velocity limits.

C. Computation of CDF

The derivation of CDF is based on Eq. (2), involving three components: 1. constructing the SDF model f_s of the robot; 2. given a point $\boldsymbol{p}$, calculating zero-level-set configurations $\boldsymbol{q}'$ that satisfy $f_s(\boldsymbol{p}, \boldsymbol{q}') = 0$; 3. Given current joint configuration $\boldsymbol{q}$, finding the closest configuration on the zero-level-set $\boldsymbol{q}'$, coupled with the calculation of the ℓ^2 norm distance to yield the CDF value. We will discuss each step in detail.

1) Robot SDF model: Various approaches exist for calculating the signed distance from a point in the robot workspace to the robot surface. Early approaches involve representing the robot geometry using spheres or meshes to approximate a coarse SDF. Recent investigations employ deep neural networks [17, 14] for encoding the robot SDF. We adopt the method presented in [16] that exploits kinematic chains and basis functions to represent the robot SDF f_s, trading off accuracy and efficiency by providing a balance between explicit and implicit representation.

2) Finding zero-level-set configurations: The challenge of determining zero-level-set configurations parallels the inverse kinematics (IK) problem. While IK only focuses on the end-effector, CDF provides a more expressive approach that focuses on the whole robot geometry. We cast it as an optimization problem and employ the L-BFGS algorithm [23]. L-BFGS is a quasi-Newton method that has demonstrated effectiveness in robot motion planning [36]. The choice of L-BFGS is motivated by its relative simplicity and efficient parallelization. Alternative methods based on Gauss-Newton optimization could also be chosen. We formulate the cost function as a squared sum of SDF values, denoted as $c = \sum f_s^2(\boldsymbol{p}, \boldsymbol{q})$. The search direction is updated using standard L-BFGS steps, and a line search approach is conducted for stable updates. The algorithm, outlined in Algorithm 1, involves initializing a batch of joint configurations $\boldsymbol{q}$, by concurrently optimizing them to

establish dense zero-level-set configurations. Additionally, our SDF model provides the link index of the robot in contact with the point $\boldsymbol{p}$ at configuration $\boldsymbol{q}'$, offering valuable information for subsequent computations.

3) Retrieving CDF value: Given an input point $\boldsymbol{p}$ and configuration $\boldsymbol{q}$, the procedure outlined in Section III-C step 2 identifies zero-level-set configurations $\boldsymbol{q}'$ corresponding to $\boldsymbol{p}$. Calculating the CDF value involves determining the closest $\boldsymbol{q}'$ from $\boldsymbol{q}$. However, the sparse sampling of $\boldsymbol{q}'$ may result in an overly smooth CDF. To mitigate this, we reformulate (2) as

$$f_c(\boldsymbol{p}, \boldsymbol{q}) = \min_{k=1,\dots,K} (\min_{\boldsymbol{q}'} \|\boldsymbol{q}_{:k} - \boldsymbol{q}'_{:k}\|), \tag{5}$$

where k denotes the kth robot link in contact with $\boldsymbol{p}$, K is the total number of robot links, and $\boldsymbol{q}_{:k}$ represents all joint configurations before link k. This adjustment is rooted in the observation that CDF is influenced solely by preceding joint angles before the contact link. This modification exploits the inherent kinematic structure of the robot, leading to a more accurate approximation of CDF and reducing uncertainty, especially when $\boldsymbol{q}'$ samples are limited. The corresponding gradient of CDF is expressed as

$$\begin{aligned} \boldsymbol{q}'_{\min}, k_c &= \arg\min_{\boldsymbol{q}', k} \|\boldsymbol{q}_{:k} - \boldsymbol{q}'_{:k}\|, \\ \nabla_{\boldsymbol{q}} f_c(\boldsymbol{p}, \boldsymbol{q}) &= \frac{\boldsymbol{q}_{:k} - \boldsymbol{q}'_{\min,:k_c}}{\|\boldsymbol{q}_{:k} - \boldsymbol{q}'_{\min,:k_c}\|}, \end{aligned} \tag{6}$$

where $\boldsymbol{q}'_{\min}$ is the closest joint configuration and k_c is the corresponding contact link. The gradient possesses a unit ℓ^2 norm and points against the direction of the nearest joint configuration on the zero-level-set.

D. Fusion of CDF

The computation of CDF described in Section III-C is applicable to both single points and batches of points. However, this process typically requires 1–10 seconds to find joint configurations and is scene-dependent. To address this challenge, we introduce a fusion strategy that computes the CDF independently for each point and combines them, yielding a scene-agnostic CDF representation conducive to efficient online calculations. Specifically, the point cloud $\boldsymbol{p}$ with N points can be partitioned into M subsets ($M \leq N$):

$$\boldsymbol{p} = \{\boldsymbol{p}^1, \cdots, \boldsymbol{p}^M\}, \tag{7}$$

where $\boldsymbol{p}^i$ represents a subset of $\boldsymbol{p}$ with N_i points. The CDF f_c is constructed by fusing the CDFs f_c^i of each subset:

$$f_c(\boldsymbol{p}, \boldsymbol{q}) = \min_{i=1,\dots,M} f_c^i(\boldsymbol{p}^i, \boldsymbol{q}). \tag{8}$$

For the extreme case where $M = N$, each subset contains only one point, allowing for offline computation and storage. The online inference stage only involves subtraction and minimum operations to fuse the CDFs based on the input, ensuring simplicity and efficiency. For example, initializing the workspace into a Cartesian grid, pre-computing corresponding joint configurations for each grid cell, and updating occupied cells during scene changes (see Figure 2). This fusion strategy

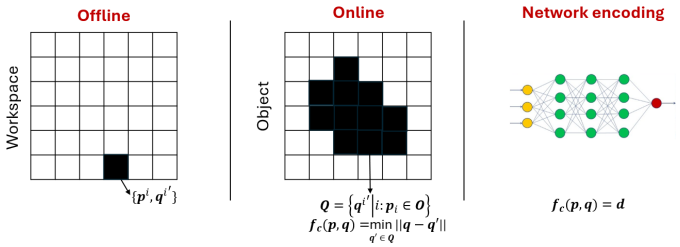


Fig. 2: Illustration of the computation of CDF. During the offline phase, we initialize the workspace of the robot as a volumetric grid and compute zero-level-set joint configurations for each grid point. For online computation, given an object $\mathcal{O}$, we identify the closest configuration in the set $\boldsymbol{Q}$ associated with occupied grids to calculate the ℓ^2 distance. We further encode the CDF with neural networks to obtain a compact and grid-free representation.

enables a non-parametric CDF representation and can be generalized to arbitrary environments.

The fusion of CDF also connects to the union operation of SDFs, albeit performed in configuration space. Consequently, other Boolean operations used to compose and transform SDFs can also be applied to CDF, such as subtraction, intersection, repetition, and rounding.

IV. Neural Configuration Space Distance Field

In this section, we elaborate on the extension of the CDF through a learning-based approach to formulate an implicit representation, referred to as neural CDF. In contrast to the online computation detailed in Section III, employing neural networks for CDF offers additional advantages. It disentangles from spatial resolution constraints, allowing for an expressive representation with reduced memory requirements. The neural CDF, being grid-free, facilitates distance queries between arbitrary joint configurations and points, enhancing flexibility and efficiency. Additionally, it presents a continuous representation, providing access to analytical gradients. Lastly, neural CDF operates in latent space and serves as a feature extractor for downstream tasks. In summary, neural CDF introduces trade-offs between accuracy, efficiency, and compression capabilities while enhancing flexibility.

Neural CDF approximates the batched function $\boldsymbol{f}_c(\boldsymbol{p}, \boldsymbol{q}) : \mathbb{R}^{b_1 \times 3} \times \mathbb{R}^{b_2 \times n} \to \mathbb{R}^{b_1 \times b_2}$ by learning the weights of a multilayer perceptron (MLP) network. It takes the concatenation of $\boldsymbol{p}$ and $\boldsymbol{q}$ as input, with size $\mathbb{R}^{b_1 b_2 \times (3+n)}$ and outputs the CDF value $\mathbb{R}^{b_1 b_2 \times 1}$. The neural CDF remains scene-agnostic, allowing the straightforward online fusion of different points. Subsequent sections will delve into data generation procedures, loss function design, training, and learning results.

A. Dataset Generation

The dataset generation process aligns with the computational and fusion procedures detailed in Section III, comprising both offline and online components. In the offline phase, we construct a $T \times T \times T$ volumetric grid in the 3D robot workspace. Utilizing Algorithm 1, joint configurations $\boldsymbol{q}'$ that satisfy $f_s(\boldsymbol{p}, \boldsymbol{q}') = 0$ for each grid point $\boldsymbol{p}$ are computed. Subsequently, a farthest point sampling algorithm is applied

Algorithm 2 Neural CDF Data Generation

Initialization: volumetric grid G
offline data
for each $p \in G$: ▷ For each point on the grid
  $\boldsymbol{q}' \leftarrow \boldsymbol{q} : f_s(\boldsymbol{p}, \boldsymbol{q}) = \mathbf{0}$ ▷ Find $\boldsymbol{q}'$ using Algorithm 1
  $\boldsymbol{q}' \leftarrow \text{Downsample}(\boldsymbol{q}')$ ▷ Downsample $\boldsymbol{q}$
online data
for $t = 1, \ldots, T$ ▷ Iterate over T epochs
  $\boldsymbol{p}, \boldsymbol{q}' \leftarrow \text{SampleOffline}()$ ▷ Sample $\boldsymbol{p}$,$\boldsymbol{q}'$ from offline data
  $\boldsymbol{q} \leftarrow \text{RandomSample}()$ ▷ Online sample $\boldsymbol{q}$ that satisfies joint limits
  Compute $f_c, \nabla_{\boldsymbol{q}} f_c$ using (5) and (6) ▷ Ground truth
  ...
  ComputeLoss() ▷ Network training
  ...
end for

to downsample the obtained configurations. The resulting zero-level-set configurations for each grid point serve as templates for online computations. In the online phase, b_1 points and b_2 joint configurations, randomly sampled within joint limits, are selected. The closest template is identified, and the ℓ^2 norm distance is computed using (5). Simultaneously, the gradient concerning the joint configuration is calculated. The dataset generation process is outlined in Algorithm 2.

B. Loss Function

We design a loss function for training the neural CDF based on existing neural SDF representations [25, 9, 24]. The loss function consists of four components: distance loss, gradient loss, eikonal loss and tension loss, each serving a distinct purpose.

Distance loss. The distance loss is characterized by the mean squared error between the predicted CDF and the ground truth, expressed as

$$\mathcal{L}_{\text{dist}} = \frac{1}{b_1 b_2} \sum_{i=1}^{b_1} \sum_{j=1}^{b_2} \left(\hat{f}_c(p_i, q_j) - f_c(p_i, q_j) \right)^2, \tag{9}$$

where $\hat{f}_c$ and f_c denote the predicted and ground truth C-space distances for point p_i and configuration q_j, respectively.

Gradient loss. This term constrains the gradient of the predicted CDF to consistently point against the direction of the closest joint configuration on the zero-level set. It employs cosine similarity loss to penalize deviations, given by

$$\mathcal{L}_{\text{grad}} = \frac{1}{b_1 b_2} \sum_{i=1}^{b_1} \sum_{j=1}^{b_2} \left(1 - \frac{\nabla_{\boldsymbol{q}} \hat{f}_c(p_i, q_j)^\top \nabla_{\boldsymbol{q}} f_c(p_i, q_j)}{\|\nabla_{\boldsymbol{q}} \hat{f}_c(p_i, q_j)\| \, \|\nabla_{\boldsymbol{q}} f_c(p_i, q_j)\|} \right). \tag{10}$$

Eikonal loss. This term regulates the predicted CDF by encouraging its gradients to have a unit ℓ^2 norm. This regularization, inspired by the eikonal partial differential equation, ensures a valid signed distance field [9, 24]. The eikonal regularization term is formulated as

$$\mathcal{L}_{\text{eikonal}} = \frac{1}{b_1 b_2} \sum_{i=1}^{b_1} \sum_{j=1}^{b_2} \left| \|\nabla_{\boldsymbol{q}} \hat{f}_c(p_i, q_j)\| - 1 \right|. \tag{11}$$

Tension loss. The tension loss term aims to regularize the curvature of the CDF, promoting smoothness. It penalizes the squared sum of the Laplacian, which measures the second derivatives of the predicted CDF [12, 44], namely

$$\mathcal{L}_{\text{tension}} = \frac{1}{b_1 b_2} \sum_{i=1}^{b_1} \sum_{j=1}^{b_2} \|\nabla^2_{\boldsymbol{q}} \hat{f}_c(p_i, q_j)\|^2, \tag{12}$$

where $\nabla^2_{\boldsymbol{q}}$ is the Laplacian operator computed via automatic differentiation.

Total loss. The network is optimized to minimize the weighted sum of the four loss terms

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{dist}} + \lambda_2 \mathcal{L}_{\text{grad}} + \lambda_3 \mathcal{L}_{\text{eikonal}} + \lambda_4 \mathcal{L}_{\text{tension}}, \tag{13}$$

In the experiments, we set $\lambda_1 = 5.0, \lambda_2 = 0.1, \lambda_3 = 0.01, \lambda_4 = 0.01$.

C. Implementation Details

For the training of the neural CDF model, we employ a simple fully connected MLP. To assess its effectiveness in handling high-dimensional inputs, we evaluate the neural CDF on a 7-axis Franka robot. The resolution of the volumetric grid T is set to 20 for data generation. The input dimension is 3+7, and the output corresponds to the configuration space distance. In line with previous work [14], we adopt a 5-layer MLP architecture, where the input data is enriched with position encoding [20]. During training, we randomly sample $b_1 = 4000$ points with corresponding joint configurations and $b_2 = 100$ configurations. Thus, the batch size is 4000×100. The network is trained for $50,000$ epochs using the Adam optimizer with a learning rate of 0.001, decayed by a factor of 0.5. The training process spans approximately 2 hours on a single NVIDIA RTX 3090 GPU.

D. Learning Results

We evaluate the trained neural CDF model through a comprehensive evaluation for both accuracy and efficiency. The results are presented in Table I. Specifically, we run the forward pass of the network, which outputs predicted C-space distance values f_c for input pairs $\boldsymbol{p}$ and $\boldsymbol{q}$. Then we compute the gradient via automatic differentiation and project configurations $\boldsymbol{q}$ to points $\boldsymbol{p}$ along gradient direction using (4). According to the definition of CDF, the distance between the robot surface, defined by projected configurations $\boldsymbol{q}_{\text{proj}}$ to input points $\boldsymbol{q}$, should be 0. Thus, we measure the mean absolute error (MAE) and root mean squared error (RMSE) as metrics. The success rate (SR) denotes the percentage of configurations successfully projected to input points within a threshold of $3cm$. The projection process is designed to run iteratively for improved accuracy. Each experiment involves the random sampling of 1000 points and 1000 configurations (the results are reported as averages). The outcomes reveal that our model

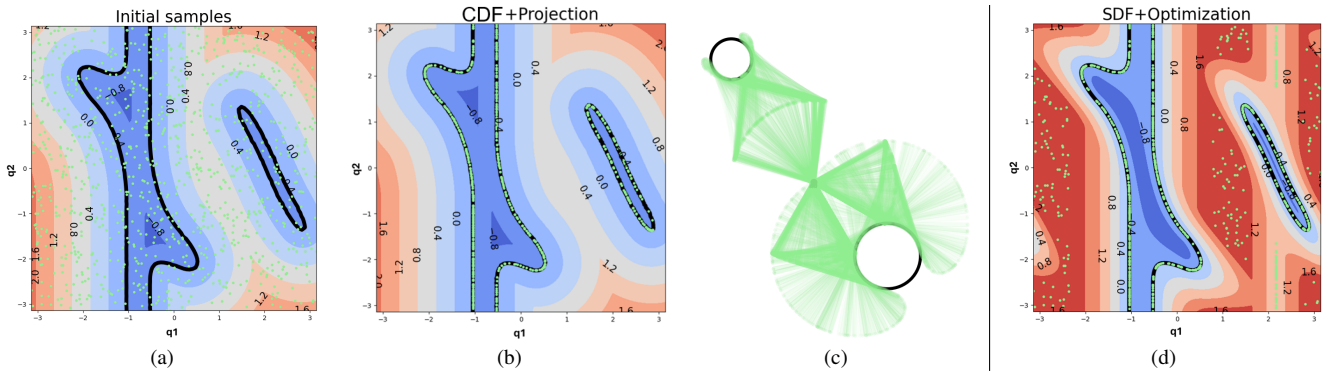


Fig. 3: Comparison between CDF and SDF in solving whole-body inverse kinematics problem. (a) The initial sampled joint configurations. (b) Gradient projection by CDF. (c) Task space visualization of feasible solutions in (b). (d) Results for distance query-based method with L-BFGS optimizer. We can see that with the baseline SDF approach, the system can get stuck when the gradient of the SDF vanishes or reaches the singularity.

TABLE I: Accuracy and computation time (GPU / CPU) of Neural CDF on the Franka robot.

Accuracy				Computation Time		
Projection Iterations	MAE (cm)	RMSE (cm)	SR(%)	Batch Size	Inference Time (ms)	Projection Time (ms)
1	4.99±1.93	8.59±3.15	60.3±12.20	1	0.49/0.37	0.71/0.34
2	1.64±0.62	2.80±1.20	87.8±9.50	10	0.51/0.59	0.72/0.66
3	1.39±0.51	2.09±0.94	91.1±8.12	10^2	0.56/0.95	0.75/1.03
4	1.36±0.49	2.00±0.89	91.6±8.23	10^3	0.58/10.20	0.97/5.48
5	1.34±0.48	1.92±0.79	91.8±8.31	10^4	0.79/25.00	1.01/36.00
10	1.35±0.52	1.89±0.80	91.6±8.39	10^5	4.61/329.00	11.30/310.00

accurately predicts CDF values and gradients, facilitating a projection process that successfully identifies the closest joint configurations on the zero-level-set. Stability is achieved after 2 iterations. As for computation time, results are provided for a single NVIDIA RTX 3090 GPU and a 30-core 2.2GHz CPU. Inference time denotes the duration for a single forward pass of the network, while projection time encompasses the time for automatic differentiation and the projection process. These results underscore the efficiency, high parallelizability, and scalability of our neural CDF, particularly when dealing with large batch sizes.

V. CDF for Whole-body Inverse Kinematics

CDF inherently encodes the kinematic structure of the robot, offering a solution to the inverse kinematics problem through gradient projection without the need for iterative procedures. Given its holistic modeling of the robot geometry, our approach extends the inverse kinematics problem to whole-body inverse kinematics problems, instead of only focusing on the end-effector. We assess our approach with a planar robot and the 7-axis Franka robot.

A. 2-DoF Planar Robot

We start with a straightforward example involving a 2D planar robot with link lengths $l_1 = l_2 = 2$ and joint limits $q_1, q_2 \in [-\pi, \pi]$. The CDF is computed online using the methodology outlined in Section III. Two circular objects with radii $r_1 = 0.8$ and $r_2 = 0.5$ are positioned at $(1.8, -1.8)$ and $(-2.0, 3.0)$, respectively. The objective of the whole-body inverse kinematics task is to identify joint configurations that make the robot reach the objects. We compare our CDF representation with SDF and present qualitative results in Figure 3. The results demonstrate that in this 2D scenario, CDF effectively solves the problem through a one-step gradient projection, while SDF-based optimization struggles to find solutions when the gradient vanishes and gets stuck in local minima due to the nonlinearity of the forward kinematics function.

B. 7-DoF Franka Robot

To further evaluate the performance of CDF, we conducted experiments with a 7-axis Franka robot, utilizing the trained neural CDF model outlined in Section IV. The evaluation focused on the whole-body inverse kinematics performance for various target points, employing $10'000$ randomly initialized configurations. The gradient projection process was iteratively performed in three steps to enhance performance. For comparison, two baseline representations were included in the evaluation: the whole-body SDF representation proposed in [16] and the neural joint space SDF representation (Neural-JSDF) proposed in [14]. Both approaches are followed with an L-BFGS algorithm for optimization, which is also described in cuRobo [36], achieving state-of-the-art performance. Experiments are repeated 100 times and average results are shown in Table II. CDF demonstrated the ability to compute over $700'000$ valid solutions per second, outperforming the state-of-the-art distance query-based approach by 180 times, which could only find $3'700$ solutions. Additionally, the inference and projection process of CDF took only $1-2$ milliseconds, with the primary time cost attributed to the post-processing of distance checking to select valid solutions that satisfy the error threshold. Figure 4 shows the results of a 1-step projection for different target point positions.

C. Applications

We demonstrate two applications that leverage the gradient projection capabilities of CDF. The first application involves a

Fig. 4: Gradient projection for whole-body inverse kinematics using neural CDF. The centers of the red spheres are target points, where the radius of spheres is set to $0.05m$.

Fig. 5: Goalkeeper task in simulation.

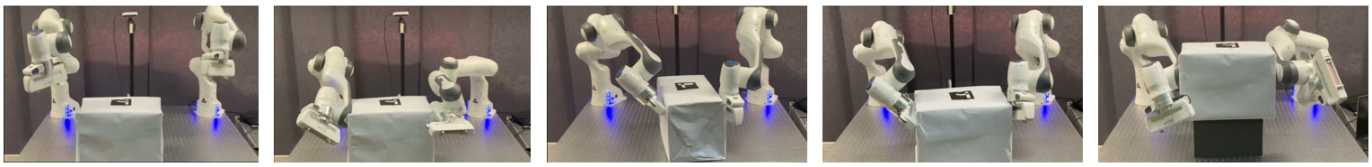


Fig. 6: Planned configurations to reach the box. The first image shows the initial configurations of the two arms.

TABLE II: Comparison of CDF and SDFs in whole-body inverse kinematics task with a 7-axis Franka robot. CDF solves 8773 solutions in 10.6 ms while the SDF based method only finds 3652 solutions in 971 ms.

Methods	Valid Solutions	Time (ms)
CDF + 1-step Projection	6089	**8.72**
CDF + 2-step Projection	8773	10.60
CDF + 3-step Projection	**9163**	12.70
SDF + L-BFGS optimizer	3652	971.00
Neural-JSDF + L-BFGS optimizer	264	272.00

goalkeeper task where the robot intercepts a thrown ball using its arm links. The second one is a dual-arm lifting task exploiting the whole-body structure of the robot to establish contact with a large box, which is hard to accomplish conventionally using an end-effector.

1) Goalkeeper task: In contrast to tasks involving rapid robot responses to avoid obstacles, the present task entails the robot acting as a goalkeeper, utilizing its arm to intercept a propelled ball. This task poses increased difficulty as the robot must promptly determine a whole-body inverse kinematics solution and transition to the requisite configuration to intercept the ball. The experimental configuration is outlined as follows: (1) a rectangular goal, measuring $0.8m$ in width and $0.6m$ in height, is positioned behind the robot, whose configuration is initialized in the middle of the joint angle range; (2) a ball is thrown toward the goal from the front of the robot with a randomly assigned direction and velocity; (3) the robot is tasked to move its arm to intercept the ball.

The conducted evaluations are performed in a simulated environment, with the assumption that the robot can only perceive the current position of the ball, necessitating swift movements to ensure an effective defense. A joint position controller is employed to govern the robot arm, directing it to the designated joint configuration. The task is executed 100 times, resulting in an 82% success rate for our CDF representation. In comparison, the SDF-based method achieves a success rate of only 35%. Snapshots of our approach are depicted in Fig. 5.

2) Large box lifting: The objective of this task is to plan joint configurations for two robot arms to establish contact with a designated box. We assume that the contact points on the box are predefined, and the robots can use any surface points on their body for establishing contact. This task typically involves a multi-objective optimization problem with constraints, including joint limits, collision avoidance, and goal-reaching. The combination of these objectives introduces non-convexity and makes the problem hard to solve. Leveraging the efficient and parallelizable gradient projection inherent to CDF, we instead present a straightforward sample-filter

approach to address this problem. Specifically, we iteratively sample a batch of initial configurations, project them onto the contact points, and filter out configurations in collision with the box or violating joint limits. This process continues until feasible solutions satisfying all constraints are identified. An evaluation of our approach, compared with the Gauss-Newton optimization method outlined in [16], is presented in Table III. The results indicate that CDF reduces the planning time by a factor of 7 and generates shorter paths. During the lifting phase, the Jacobian matrix of the contact point w.r.t. the joint configuration is computed. A joint impedance controller is used in the experiment. Qualitative results are shown in Fig. 6.

TABLE III: Comparison results on large box lifting task.

Methods	Planning Time(s)	Average Distance(rad)
CDF + Filter	**7.65**	**1.37**
SDF + Optimizer	54.80	2.85

VI. CDF for Manipulation Planning

In this section, we investigate the use of CDF for manipulation planning tasks. The key advantage of CDF is the structured representation that alleviates challenges arising from nonlinearity and singularity, making motion optimization in configuration space easier. Similarly to conventional SDF, CDF provides efficient queries of distances and gradients, enabling large-scale parallel computation. To demonstrate its efficacy, we initially explore qualitative results through 2-DOF examples and then progress to 7-DOF robot scenarios, including real-world experiments.

A. Benchmark Approaches and Evaluation metrics

We evaluate the CDF representation on several gradient-based motion optimization approaches:

1) Quadratic programming: We first formulate the motion planning task as reactive quadratic programming (QP) problem, drawing inspiration from the work of Mirrazavi et al. [21]. The QP formulation is:

$$\boldsymbol{u}_k^* = \arg\min_{\boldsymbol{q},\boldsymbol{u}} \boldsymbol{e}(\boldsymbol{q}_k)^\top \boldsymbol{H} \boldsymbol{e}(\boldsymbol{q}_k) + {\boldsymbol{u}_k}^\top \boldsymbol{R} \boldsymbol{u}_k, \tag{14a}$$

$$\text{s.t. } \boldsymbol{q}_{k+1} = \boldsymbol{A}\boldsymbol{q}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{14b}$$

$$\boldsymbol{q}_k \in \mathcal{Q},\ \boldsymbol{u}_k \in \mathcal{U}, \tag{14c}$$

$$-\nabla_{\boldsymbol{q}} f_c(\boldsymbol{p},\boldsymbol{q}) \boldsymbol{u}_k \Delta t \le \ln(f_c(\boldsymbol{p},\boldsymbol{q}) + \gamma), \tag{14d}$$

where $\boldsymbol{q}_k$ and $\boldsymbol{u}_k$ are the state and control input at time step k, $\boldsymbol{H}$ and $\boldsymbol{R}$ are the positive definite matrices for tracking errors and control efforts, $\boldsymbol{e}(\boldsymbol{q}_k) = \boldsymbol{q}_k - \boldsymbol{q}_{\text{desire}}$ is the error vector between the initial and goal configurations, $f_c(\boldsymbol{p},\boldsymbol{q})$ is the CDF, Δt is the time step, γ is a scalar hyperparameter that acts as a safety buffer, and $\mathcal{Q}$ and $\mathcal{U}$ are the admissible state and control constraints. The constraint (14d) ensures collision avoidance, where the robot is allowed to get close to the obstacle when far away, and it is forced to follow the tangent or normal direction of the gradient field when close. For implementation, we use the CasADi [2] library and solve it with popular solvers including OSQP [35], qpOASES [7]. Nonlinear programming solvers like IPOPT [40] and QRQP [8] are also tested.

2) iterative Linear Quadratic Regulator (iLQR): Another benchmark approach involves employing an iterative Linear Quadratic Regulator that solves the optimal control problem by iteratively linearizing the dynamics and cost function around the current trajectory [15]. The dynamic system is defined as $\Delta \boldsymbol{q}_{k+1} = \boldsymbol{A}\Delta\boldsymbol{q}_k + \boldsymbol{B}\Delta\boldsymbol{u}_k$. We minimize the cost function

$$\begin{aligned} c(\boldsymbol{q},\boldsymbol{u}) = {} & \boldsymbol{e}(\boldsymbol{q}_K)^\top \boldsymbol{Q}_1 \boldsymbol{e}(\boldsymbol{q}_K) \\ & + \sum_{k=1}^{K-1} \boldsymbol{h}(\boldsymbol{q}_k)^\top \boldsymbol{Q}_2 \boldsymbol{h}(\boldsymbol{q}_k) + {\boldsymbol{u}_k}^\top \boldsymbol{R} \boldsymbol{u}_k \end{aligned} \tag{15}$$

where $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ are precision matrices for tracking errors and collision avoidance, $\boldsymbol{R}$ is the control effort matrix. $\boldsymbol{h}(\boldsymbol{q}_k) = \min(f_c(\boldsymbol{p},\boldsymbol{q}) - \gamma,\ 0)$ represents the collision avoidance term. The solution of iLQR can be computed either in batch or recursive form, see [1] for details.

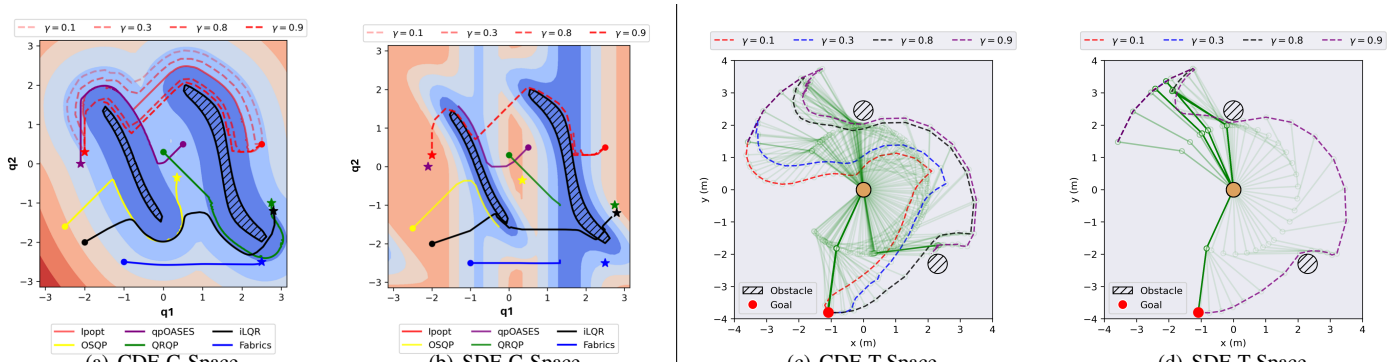
3) Geometric fabrics: Geometric fabrics is a reactive acceleration-based control policy $\ddot{\boldsymbol{q}} = \pi(\boldsymbol{q}, \dot{\boldsymbol{q}})$. $\ddot{\boldsymbol{q}}$ is computed through the motion of equation $\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{F} = 0$, where $\boldsymbol{M}(\boldsymbol{q},\dot{\boldsymbol{q}})$ and $\boldsymbol{F}(\boldsymbol{q},\dot{\boldsymbol{q}})$ model the generalized mass matrix and external forces based on positions and velocities , see [30] for details. The obstacle avoidance geometry is defined as $\boldsymbol{h} = \lambda\|\dot{\boldsymbol{q}}\|^2 \nabla_{\boldsymbol{q}} \psi(f_c(\boldsymbol{p},\boldsymbol{q}))$. When the robot gets close to the obstacle, the value $\psi(\boldsymbol{p},\boldsymbol{q})$ increases and repels the robot away from the collision boundary. The geometric fabrics are adapted from the open-source implementation of optimization fabrics [34].

For a comprehensive evaluation, we estimate both CDF and SDF. The implementation of SDF is achieved by replacing the distance function f_c with f_s. Several evaluation metrics are adopted for comparison.

- **Success Rate**: the success rate shows the percentage of collision-free trajectories generated while reaching the goal. As the evaluation involves randomly sampling initial and goal configurations, for the cases when all algorithms failed, the corresponding samples were excluded when reporting the success rate.
- **Tracking Error**: As reactive approaches may get stuck at a local minimum, we introduce the tracking error to measure the final ℓ^2 norm distance between the final configuration and the desired configuration.
- **Time Step**: The time step denotes the average number of time steps the agent requires to reach the goal configuration.

B. Planar Robot Test

For the 2D experiment, we follow the previous section that defined a 2D planar robot with link lengths $l_1 = l_2 = 2$ and joint limits $q_1, q_2 \in [-\pi, \pi]$. Two circle obstacles with radius of 0.3 are placed at $(2.3, -2.3)$ and $(0.0, 2.45)$. We randomly sample initial and goal configurations for 100 cases and report the experimental results in Table IV. It shows that the approach based on CDF has a higher success rate and better tracking accuracy than SDF. The average number of time steps is smaller, increasing efficiency. To further investigate the



(a) CDF C-Space (b) SDF C-Space (c) CDF T-Space (d) SDF T-Space

Fig. 7: CDF/SDF-based motion planning approaches. Different methods are shown in different colors. We also demonstrate how the safety buffer affects the planning results of CDF and SDF.

TABLE IV: Experiments for motion planning on SDF and CDF.

	2D - CDF			2D - SDF			7D - CDF			7D - SDF		
	Success Rate	Tracking Error (cm)	Time Step	Success Rate	Tracking Error (cm)	Time Step	Success Rate	Tracking Error (cm)	Time Step	Success Rate	Tracking Error (cm)	Time Step
IPOPT	92%	1.27	253	51%	2.19	284	93%	0.98	231	51%	1.12	290
QRQP	94%	1.16	245	52%	2.07	279	91%	0.99	226	38%	1.31	297
OSQP	88%	1.21	300	40%	2.19	321	91%	0.92	279	48%	1.04	301
qpOASES	91%	1.04	241	63%	1.74	263	93%	0.99	229	51%	1.13	289
Geometric Fabrics	95%	1.03	-	68%	1.76	-	88%	2.02	-	74%	2.18	-
iLQR	76%	0.06	-	55%	0.12	-	48%	0.02	-	38%	0.03	-

mechanism behind this, we visualize some cases in Fig. 7-a,b. It shows that the structured distance field can benefit all approaches mentioned above, as their trajectories follow the geodesics of CDF. In contrast, the nonlinearity of the configuration space when using the conventional SDF makes the optimizer get stuck into local minima. Additionally, we visualize the planning results of IPOPT with different safety buffers γ ranging from 0.1 to 0.9. The planner exhibits a more conservative behavior as the value of γ decreases to ensure safety. With CDF, the planned trajectory scaled well with different γ. In contrast, for SDFs, the planner only finds a solution when $\gamma = 0.9$ and the trajectory in joint space is very close to the obstacle. It is also reflected in the task space (Fig. 7-c,d), where the robot reaches a singularity when it is close to the obstacle.

C. 7-axis Franka Robot Experiments

We further conduct experiments on the Franka robot to demonstrate the effectiveness of CDF.

We place several different obstacles such as spheres, walls, and rings, with randomly sampled initial and goal configurations. For a fair comparison, we use the neural representation for both CDF and SDF. Experiments are also repeated 100 times and we report the average results in Table IV. The CDF-based planners demonstrate better performance than SDF-based approaches in terms of success rate, tracking error and number of time steps. For QP controllers, our methods can run over 200Hz frequency, thanks to the high efficiency of neural networks.

For real-world experiments, we set up two different scenarios: static and dynamic environments. In the static environment, we place some obstacles in the robot workspace, such as blocks (for simple cases) and a shelf (for hard cases being highly non-convex). We use a RealSense D435 camera to capture the point cloud of the obstacles. For dynamic environments, we simply detect the obstacle according to the HSV color and convert it to point clouds. The same QP controller with IPOPT optimizer is used for motion planning. For static scenes, we wait for the QP controller to compute the full trajectory and then execute it on the robot. For dynamic scenes, we test the reactive motion generation and send the control commands online. Qualitative results are shown in Fig. 8, showing the effectiveness of our approach.

VII. Discussion and Conclusion

In this paper, we proposed to consider the geometry of robot configuration space as a distance field, and present a new representation called CDF to describe the topological structure of objects in configuration space. After discussing the formulation and properties, we introduced an efficient algorithm to compute and fuse CDFs. An implicit neural network encoding was also proposed to balance the accuracy, efficiency, and compression of CDF. We demonstrated the effectiveness of CDF with planar examples and with a 7-axis Franka robot in whole-body inverse kinematics and motion planning tasks.

CDF serves as a representation implicitly encoding inverse kinematics through a distance field. Both Cartesian grid and neural network representations facilitate the offline computation of inverse kinematics, thus enabling efficient online queries. In contrast, SDFs can be perceived as representations encoding the forward kinematics of a robot.

The pivotal step in CDF computation involves identifying zero-level-set configurations from a given point or object. Recent advancements in differentiable robot SDF representations have paved the way for efficient, accurate, and parallelizable

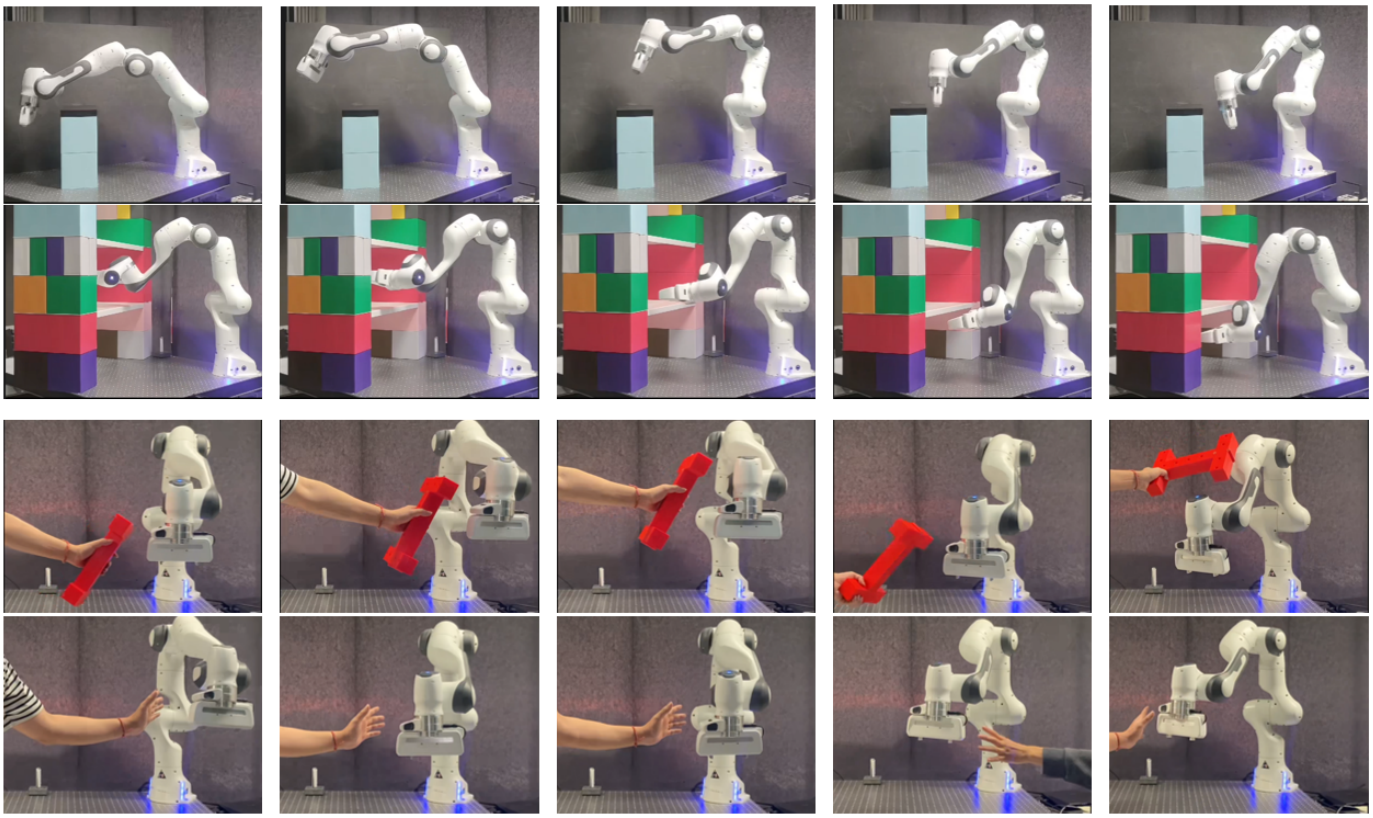


Fig. 8: Motion planning and reactive collision avoidance using CDF. The top two rows show static environments. The two bottom rows show dynamic environments.

robot SDF computation. This advancement ensures efficient solutions in optimization problems. The fusion strategy and neural network representation further enhance the computational efficiency of online queries.

CDF offers a global view of configuration space geometry. In practical applications, it is unnecessary to reconstruct the entire CDF, and selective querying of points of interest suffices. This adaptability enables the application of off-the-shelf techniques developed for task space to be extended to configuration space. Additionally, the combined use of SDFs and CDFs allows for a holistic understanding of geometry information across both task and configuration spaces, avoiding the need for calculating the nonlinear mapping.

Acknowledging the challenges posed by the high-dimensional configuration space and limited data, CDF computations encounter issues such as time consumption for offline computation, memory inefficiency for non-parametric representation, or compromised accuracy in neural network representation. The preference for neural representation is due to the trade-off between efficiency, compression, and accuracy, as well as the flexible structure that can be integrated into other frameworks. The learning results of neural networks are not fully optimized, as we mainly focused in this paper on the representation. Nevertheless, once a model is trained, the robot can re-use it multiple times later as a function approximator.

In addition to the balance between accuracy and efficiency, CDF currently has other limitations that need to be improved. First, the unsigned distance field is always non-negative, and using limited zero-level-set configuration samples can result in an inaccurate approximation for configurations close to the zero-level-set. It can be solved by computing the gradient of zero-level-set configuration samples and adding sign information to the CDF representation. Additionally, although CDF can be generalized to any points in the workspace and arbitrary configurations, the representation is based on the geometry and kinematics of the robot. It has to be computed separately for different robot models. The scalability of CDF to high-dimensional kinematic chains (e.g., full humanoids) also remains a challenge. Finally, the CDF measures the angular distance of joint configurations, which could be further extended to other distance metrics, such as the geodesic distance on the configuration space manifold, for better understanding the topology of the configuration space and benefiting manipulation planning tasks. Future work will delve into exploring the broader applicability of CDF in various robotic problem domains, such as collision-free inverse kinematics, geometric motion planning, operation space control, multi-objective optimization, and robotic learning.

VIII. Acknowledgments

This work was supported by the China Scholarship Council (No. 202204910113), and by the State Secretariat for Edu-

cation, 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).

REFERENCES

[1] Robotics codes from scratch (RCFS). https://rcfs.ch/, Accessed: 2024.

[2] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1): 1–36, 2019. doi: 10.1007/s12532-018-0139-4.

[3] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. STORM: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Conference on Robot Learning*, pages 750–759. PMLR, 2022.

[4] Pasquale Chiacchio, Stefano Chiaverini, Lorenzo Sciavicco, and Bruno Siciliano. Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *The International Journal of Robotics Research*, 10(4):410–425, 1991.

[5] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[6] Danny Driess, Jung-Su Ha, Marc Toussaint, and Russ Tedrake. Learning models as functionals of signed-distance fields for manipulation planning. In *Proc. Conference on Robot Learning (CoRL)*, pages 245–255. PMLR, 2022.

[7] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6:327–363, 2014.

[8] Joris Gillis. Nonlinear programming and code-generation in Casadi. In *Benelux Meeting on Systems and Control*, 2019.

[9] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3789–3799, 2020.

[10] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew 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*, pages 559–568, 2011.

[11] Julius Jankowski, Lara Brudermüller, Nick Hawes, and Sylvain Calinon. VP-STO: Via-point-based stochastic trajectory optimization for reactive robot behavior. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10125–10131. IEEE, 2023.

[12] Bert Jüttler and Alf Felis. Least-squares fitting of algebraic spline surfaces. *Advances in Computational Mathematics*, 17:135–152, 2002.

[13] Holger Klein, Noémie Jaquier, Andre Meixner, and Tamim Asfour. On the design of region-avoiding metrics for collision-safe motion generation on riemannian manifolds. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2346–2353. IEEE, 2023.

[14] Mikhail Koptev, Nadia Figueroa, and Aude Billard. Neural joint space implicit signed distance functions for reactive robot manipulator control. *IEEE Robotics and Automation Letters*, 8(2):480–487, 2022.

[15] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *First International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 222–229. SciTePress, 2004.

[16] Yiming Li, Yan Zhang, Amirreza Razmjoo, and Sylvain Calinon. Representing robot geometry as distance fields: Applications to whole-body manipulation. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 15351–15357, 2024.

[17] Puze Liu, Kuo Zhang, Davide Tateo, Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. Regularized deep signed distance fields for reactive motion generation. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 6673–6680. IEEE, 2022.

[18] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[19] Jonathan Michaux, Qingyi Chen, Yongseok Kwon, and Ram Vasudevan. Reachability-based trajectory design with neural implicit safety constraints. *arXiv preprint arXiv:2302.07352*, 2023.

[20] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[21] Seyed Sina Mirrazavi Salehian, Nadia Figueroa, and Aude Billard. A unified framework for coordinated multi-arm motion planning. *The International Journal of Robotics Research*, 37(10):1205–1232, 2018.

[22] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):163–171, 1986.

[23] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[24] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa

Mukadam. isdf: Real-time neural signed distance fields for robot perception. *arXiv preprint arXiv:2204.02296*, 2022.

[25] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019.

[26] Juan José Quiroz-Omaña and Bruno Vilhena Adorno. Whole-body control with (self) collision avoidance using vector field inequalities. *IEEE Robotics and Automation Letters*, 4(4):4048–4053, 2019.

[27] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 489–494. IEEE, 2009.

[28] Nathan Ratliff, Marc Toussaint, and Stefan Schaal. Understanding the geometry of workspace obstacles in motion optimization. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4202–4209. IEEE, 2015.

[29] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018.

[30] Nathan D Ratliff, Karl Van Wyk, Mandy Xie, Anqi Li, and Muhammad Asif Rana. Optimization fabrics. *arXiv preprint arXiv:2008.02399*, 2020.

[31] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. MeshSDF: Differentiable iso-surface extraction. *Advances in Neural Information Processing Systems*, 33:22468–22478, 2020.

[32] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[33] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008. ISBN 1846286417.

[34] Max Spahn, Martijn Wisse, and Javier Alonso-Mora. Dynamic optimization fabrics for motion generation. *IEEE Transactions on Robotics*, 39(4):2684–2699, 2023.

[35] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[36] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. CuRobo: Parallelized collision-free minimum-jerk robot motion generation. *arXiv preprint arXiv:2310.17274*, 2023.

[37] Giovanni Sutanto, Isabel Rayas Fernández, Peter Englert, Ragesh Kumar Ramachandran, and Gaurav Sukhatme. Learning equality constraints for motion planning on manifolds. In *Proc. Conference on Robot Learning (CoRL)*, pages 2292–2305. PMLR, 2021.

[38] Karl Van Wyk, Mandy Xie, Anqi Li, Muhammad Asif Rana, Buck Babich, Bryan Peele, Qian Wan, Iretiayo Akinola, Balakumar Sundaralingam, Dieter Fox, et al. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics and Automation Letters*, 7(2):3202–3209, 2022.

[39] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Differentiable signed distance function rendering. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

[40] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.

[41] Thomas Weng, David Held, Franziska Meier, and Mustafa Mukadam. Neural grasp distance fields for robot manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1814–1821. IEEE, 2023.

[42] Peter Werner, Alexandre Amice, Tobia Marcucci, Daniela Rus, and Russ Tedrake. Approximating robot configuration spaces with few convex sets using clique covers of visibility graphs. *arXiv preprint arXiv:2310.02875*, 2023.

[43] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[44] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021.

[45] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. CHOMP: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

APPENDIX

A. Differentiability of CDF

The signed distance field is differentiable almost everywhere and satisfies the eikonal equation $\|\nabla f\| = 1$. An illustration of the 1D signed distance field is shown in Figure 9-a (left).

The CDF defined by Eq. (2) is an unsigned distance field. Although it still satisfies the eikonal equation, it is not differentiable at the zero-level-set (shown in Figure 9-a (right)). Since the CDF is computed by finding the closest configuration point on the zero-level-set, it may also lead to non-differentiable points when the closest joint configuration changes (shown in Figure 9-b). The CDF is thus differentiable in configuration space except at the isosurface (zero-level-set) samples and points where the closest joint configuration changes. Nevertheless, the left and right derivatives at those points are well-defined and satisfy the eikonal equation. Additionally, the non-differentiable interval is sparse in the configuration space, and we can use the subgradient instead if needed.

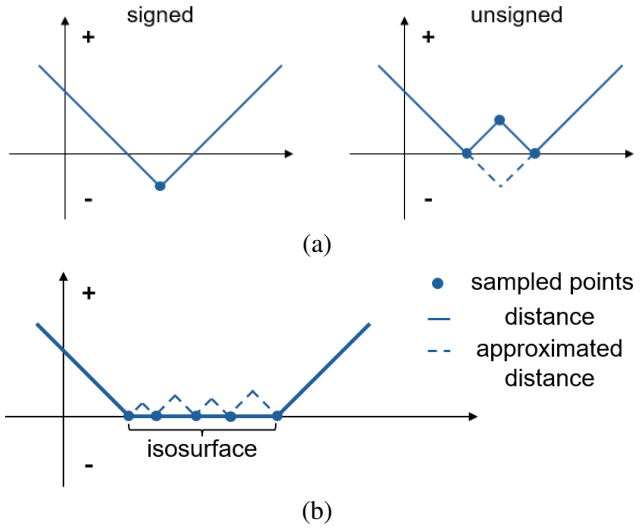


Fig. 9: Illustration of the differentiability of CDF.

The neural CDF approximates the function using a multi-layer perceptron (MLP). The oscillation of training data near the isosurface may cause an inaccurate estimation of CDF and its gradient for inputs close to the zero-level-set. The differentiability of the neural CDF depends on the activation function of the network (for example, the ReLU function is not differentiable at zero) and the gradient can be analytically computed through backpropagation.

B. Sensitivity to Noise

In real-world scenarios, the observed data usually have noise and may affect the performance of CDF. To further evaluate the sensitivity of the neural CDF representation, we introduce Gaussian noise on input points with zero mean value (μ) and standard deviation (σ) ranging from 0.01 to 0.03 and visualize the mean absolute error (MAE) and success rate (SR) in Figure 10. Although performance decreases with the increase of noise, the overall MAE and SR are still acceptable, particularly in scenarios with less noise ($\sigma = 0.01$).

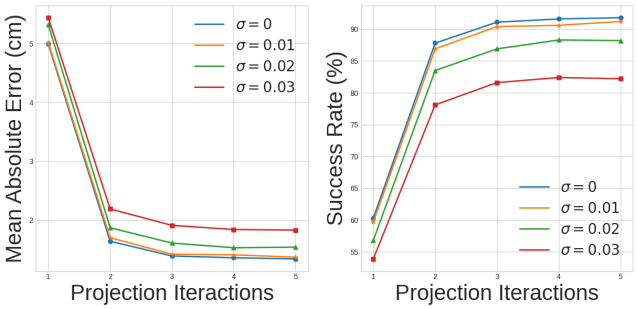


Fig. 10: The sensitivity of Neural CDF to Gaussian noise.