Collision-Free Motion Generation Based on Stochastic Optimization and Composite Signed Distance Field Networks of Articulated Robot

Baolin Liu, Gedong Jiang, Fei Zhao, *Member, IEEE*, and Xuesong Mei

***Abstract*—Safe robot motion generation is critical for practical applications from manufacturing to homes. In this work, we proposed a stochastic optimization-based motion generation method to generate collision-free and time-optimal motion for the articulated robot represented by composite signed distance field (SDF) networks. First, we propose composite SDF networks to learn the SDF for articulated robots. The learned composite SDF networks combined with the kinematics of the robot allow for quick and accurate estimates of the minimum distance between the robot and obstacles in a batch fashion. Then, a stochastic optimization-based trajectory planning algorithm generates a spatial-optimized and collision-free trajectory offline with the learned composite SDF networks. This stochastic trajectory planner is formulated as a Bayesian Inference problem with a time-normalized Gaussian process prior and exponential likelihood function. The Gaussian process prior can enforce initial and goal position constraints in Configuration Space. Besides, it can encode the correlation of waypoints in time series. The likelihood function aims at encoding task-related cost terms, such as collision avoidance, trajectory length penalty, boundary avoidance, etc. The kernel updating strategies combined with model-predictive path integral (MPPI) is proposed to solve the maximum a posteriori inference problems. Lastly, we integrate the learned composite SDF networks into the trajectory planning algorithm and apply it to a Franka robot. The simulation and experiment results validate the effectiveness of the method.**

***Index Terms*—Composite SDF networks, integrated planning and learning, motion and path planning.**

I. Introduction

THE ability to generate a safe and fast trajectory is of great significance for robot applications from industry to homes, which has attracted a lot of attention in the robotic community [1]. However, efficient and accurate collision checking for high-dimensional articulated robots is not trivial. Besides, developing optimization-based trajectory planning methods with some ability to escape local optima is also crucial for achieving this goal.

Collision checking is a prerequisite for robots to generate collision-free trajectories, which aims to ensure that the robot does not collide with the environment or human beings. One typical and effective method is the bounding volume hierarchy (BVH), which uses a set of discrete and simplified geometry primitives, e.g., spheres [2], boxes [3], capsules [4], and meshes [5] to represent the geometry structure of the robot and the environment. Those methods divide the robot and environment into a hierarchical tree structure, which enables an efficient estimate of the distance between the simplified robot and obstacles [6]. The estimated distances serve as safety constraints for collision-free motion planning and control. However, those approaches result in a trade-off between fidelity representation of the robot geometry and collision-checking efficacy, especially for irregularly shaped articulated robots. Using many small-sized geometry primitives or meshes to approximate the geometry of the articulated robot finely will increase the computational cost in collision checking. While strong assumptions such as over-conservative robot representations, can ensure safety while reducing computational cost but also reducing the remaining free working space of the robot [7]. A continuous but low-fidelity Danger Field is designed analytically in [8], [9], which acts as a collision avoidance constraint in a collaborative environment.

Machine-learning-based techniques have also been applied for collision checking. The incremental support vector machines are used to represent the collision boundary for collision pairs in configuration space while each pair of objects need a corresponding model [10]. The kernel perceptron-inspired learning method is proposed in [11] which needs active learning and online sampling to adapt to the new environment. The gradient of this extended model is applied to robot motion planning in [12]. The signed-distance field is an intuitive effective shape representation method. The deep feed-forward networks [13] in theory can learn the fully continuous Signed Distance Field (SDF) with arbitrary precision. However, the accuracy of the approximation in practice depends on the limited number of point samples guiding the decision boundary and the limited capacity of the network due to limited computing power [14]. The equality constraints are approximated using the Equality

Manuscript received 9 April 2023; accepted 28 July 2023. Date of publication 4 September 2023; date of current version 21 September 2023. This letter was recommended for publication by Associate Editor J. Wang and Editor H. Kurniawati upon evaluation of the reviewers' comments. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3305000 and in part by the National Natural Science Foundation of China under Grant 52175029. *(Corresponding author: Fei Zhao.)*

The authors are with the State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China, and also with the Shaanxi Key Laboratory of Intelligent Robots, School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: liublhri@163.com; gdjiang@mail.xjtu.edu.cn; ztzhao@mail.xjtu.edu.cn; xsmei@mail.xjtu.edu.cn).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3311357, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3311357

Constraint Manifold Neural Network (ECoMaNN) for robot motion planning in [15]. The neural networks are used to build a mapping from joint space to minimum distance for articulated robots in [16]. A trade-off exists between computational accuracy and efficiency. Most of the learning-based methods are trained in configuration space and the training data are sampled in configuration space. If the robot has high degrees-of-freedoms (DoFs), the training dataset will be very large. To this end, this work proposes composite SDF networks to train a simple and accurate SDF network for each link of articulated robots at the base coordinate frame. Any 3D query points, i.e., obstacles, in the working space, can be transformed into the base coordinate frame to efficiently estimate the minimum distance.

Generating collision-free trajectories is critical for robot applications. The sampling-based and optimization-based trajectory planning algorithms are two typical trajectory planning methods. The sampling-based trajectory planning algorithm can effectively solve high-dimensional motion planning problems, which generates the desired trajectories by connecting task-constrained samples acquired in the configuration space [17], [18]. These methods can guarantee computational completeness but are computationally expensive for high-DOF robots.

The optimization-based trajectory planning methods are another effective method for generating collision-free trajectories [2], [19], [20]. Given an initial trajectory, these methods attempt to find the optimal solution by iterating along the gradient descent direction of the cost function. The sequential quadratic programs are used to tackle the problem of time-optimal and collision-free motions while taking into account end-effector and object acceleration constraints imposed by the object being transported in [21], [22]. Besides, the optimization-based trajectory planning problems can be cast as a probabilistic inference with the Gaussian process prior and task-encoded likelihood [23]. The Stein Variational Inference is used to plan collision-free motion planning with the Gaussian process prior in [24] which is also a gradient-based solution method. In this case, the Gaussian process prior encodes the correlation of the time series at the velocity or the acceleration level, which can better ensure the smoothness of the trajectory [25]. The performance of planning algorithms is highly dependent on the parameter of the prior distribution. Thus the learned Energy-Based Models (EBM) as guiding priors are used for trajectory optimization in [26].

Gaussian process-based trajectory planning is effective for collision-free trajectories, but the Gaussian process prior is determined by many factors, and it is challenging to design a suitable prior distribution. When the variance of the prior distribution is small, sample trajectories with limited diversity will be generated, which will easily cause the optimization to fall into a local optimum. A large variance will result in conservative or even divergent solutions. For example, when the distance between the initial and the goal positions changes greatly, it is necessary to readjust the troublesome prior distribution parameters. Besides, formulating the trajectory planning as a unified optimization problem considering the safety, smoothness, and system dynamic constraints will result in a complex cost function. Those gradient-based solving methods are easy to stick to the local optimum and are far away from the time optimal.

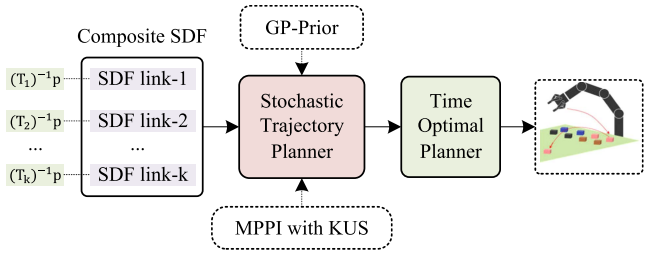


Fig. 1. An overview of the collision-free motion generation framework.

We follow the framework of Gaussian process motion planning [23], but adopt the time-normalized Gaussian process as the prior distribution, and use a larger standard deviation as the initial value to increase the diversity of trajectory samples to escape from the local optimum and explore collision-free trajectory. The trajectories are optimized using MPPI and kernel update strategies. Based on the optimized trajectory, a feasible and time-optimal trajectory is generated using an existing time-optimal planner.

Contribution: In this letter, we introduce a novel stochastic collision-free motion generation method for articulated robots based on composite SDF networks. First, the composite SDF networks are proposed for learning the SDF of the articulated robot, which can accurately estimate the minimum distance between the robot and obstacles in real time. Second, a stochastic trajectory planning algorithm is proposed to generate a fast and collision-free trajectory for a robot that has a certain ability to escape from the local optimum. The trajectory planning algorithm is formulated as a Bayesian Inference problem with a time-normalized prior and exponential likelihood function, which is solved using the kernel updating strategies and model predictive path integral (MPPI). Finally, the learned composite SDF networks are integrated into the trajectory planning algorithms which are successfully applied to generate a collision-free trajectory for the real 7-DoFs Franka robot.

The remainder of the letter is laid out as follows. First, we introduce the framework of the collision-free motion generation method in Section II. How to learn composite SDF networks for an articulated robot is discussed in detail in Section III. Section IV includes the stochastic trajectory planning method. Both simulations and experiments tests of the collision-free motion generation algorithm are performed and analyzed in Section V. Finally, Section VI concludes with concluding remarks and limitations of this method.

II. An Overview of the Motion Generation Method

The main goal of this work is to propose a collision-free motion generation method for the articulated robot which will be achieved with three cascade modules as shown in Fig. 1.

First, the composite SDF networks are learned for the articulated robot which could estimate the minimum distance

from any 3D query point $\mathbf{p}$ (transform it to robot base coordinate frame based on robot inverse kinematics, $\mathbf{T}_k^{-1}$) to the robot in batch form. Then the estimated minimum distances are streamed to a stochastic trajectory planner to generate a collision-free and spatial-optimized trajectory. This stochastic trajectory planner is formulated as a Bayesian Inference problem with a time-normalized Gaussian process prior and exponential likelihood function. The Gaussian process prior is able to enforce waypoints and goal position constraints. The likelihood function is responsible for encoding containing task-related terms, such as collision avoidance, trajectory length penalty, and boundary avoidance. The model-predictive path integral (MPPI) [27] combined with kernel updating strategies (KUS) are proposed to estimate the maximum a posteriori inference problems. Finally, a pre-existing time-optimal planner is used for assigning timing to the spatial optimized trajectory to generate spatial and temporal optimized trajectory.

III. Composite Signed Distance Field Network for Articulated Robot

The main goal of this section is to propose an accurate and efficient minimum distance estimation method for articulated robots based on Neural Networks.

A. Composite Signed Distance File for Articulated Robot Manipulator

Assume the pose of a n-DoFs robot manipulator with and K links is defined by a vector of joint angles, $\boldsymbol{q} = [q_1, q_2, \ldots, q_n]^\top \in \mathbb{R}^n$. The pose of the k^{th} link $\mathbf{T}_k$ with respect to the base coordinate frame $\{0\}$ represented as a homogeneous transformation matrix can be accurately estimated by the forward kinematics of the robot manipulator $f_k(\cdot)$.

$$\mathbf{T}_k = f_k(\boldsymbol{q}), \quad \mathcal{T} = \{\mathbf{T}_k\}_{k=1}^{K} \tag{1}$$

where $\mathcal{T}$ denotes a set of the homogeneous transformation matrices of all rigid links. We aim at learning a function that could accurately represent the SDF of the articulated robot manipulator using the Neural Networks. The intuitive idea is to learn a single network encoding distances directly as a map of robot configuration $\boldsymbol{q}$ and any 3D query point $\mathbf{p} \in \mathbb{R}^3$. In this way, we need to collect samples under different robot poses, and the dataset will become very large as the DOFs of the robot increases. In addition, without losing accuracy, the complexity of the corresponding network needs to be increased. Besides, given the joint position $\boldsymbol{q}$ of the manipulator can uniquely define the pose of each link of the robot manipulator. Thus, to simplify the learning model, instead of learning a holistic SDF network for the whole robot manipulator, we align the coordinate frame of each link to the base coordinate frame of the robot and learn an SDF network for each link.

$$f_{\boldsymbol{\theta}_k}(\mathbf{p}) = \mathbf{sdf}_k(\mathbf{p}) \tag{2}$$

where $f_{\boldsymbol{\theta}_k}(\mathbf{p})$ denotes the learned SDF network of link k with learning parameters $\boldsymbol{\theta}_k$ at base coordinate frame. Given any query 3D position $\mathbf{p}$, the outputs are the approximate minimum distance to the link k as well as the gradient $\hat{\mathbf{n}} = \frac{\partial f_{\boldsymbol{\theta}_k}(\mathbf{p})}{\partial \mathbf{p}}$. The SDF networks for all links are combined to form the composite SDF networks for the whole articulated robot manipulator.

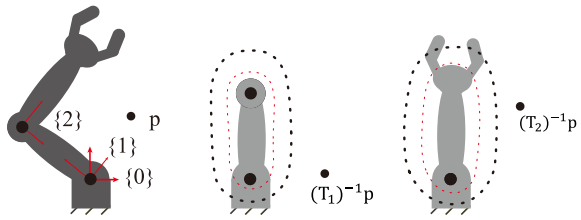


Fig. 2. An example of the composite signed distance field for a 2-DoF articulated robot manipulator.

During the online inference process, firstly, the query point $\mathbf{p}$ is projected into the robot base coordinate frame $\{0\}$ by using the inverse of the homogeneous transformation matrix $(\mathbf{T}_k)^{-1}$ as shown in Fig. 2.

$$\mathbf{p}'_k = (\mathbf{T}_k)^{-1}\mathbf{p} \tag{3}$$

where $\mathbf{p}'_k$ denotes the position with respect to the base coordinate frame of the robot. Then, the transformed point positions $\mathbf{p}'_k$ are the input of the network.

B. Dataset Synthesis and Network Designing

In order to learn accurate and continuous SDF for the articulated robot manipulator, preparing a suitable training dataset is critical. Since the accurate 3D models of each link of the robot manipulator, such as a 3D CAD and URDF model, are easy to obtain. Thus, instead of measuring and collecting the real-world distance samples of the robot, we synthesize the datasets that capture its precise distance through the watertight 3D mesh model of each link. The training samples are collected more aggressively near the surface of the link as we are concerned with detailed distance near the robot for collision checking. First, the normal directions of the mesh vertices are estimated using Open3D library [28]. Those vertices whose normal vectors are inconsistent with the normal vectors of neighboring vertices are excluded. Then a set of points are sampled along the normal directions with a user-defined distance interval based on the link size. Besides, the KDtree is used to perform forward-backward tracing for rejecting the points that do not re-project on the original point [29]. Each sample consists of the query point $\mathbf{p}$, the target distance d, and the normal vector $\mathbf{n}$ corresponding to the query point. using this method, the generated dataset is large enough, but under the premise of not affecting the accuracy, only a subset is randomly selected as the training set for the sake of computational efficiency. The dataset $\mathcal{D}$ for link k is organized as

$$\mathcal{D}_k = \{\mathbf{p}_i; d_i, \mathbf{n}_i\}_{i=1}^{N} \tag{4}$$

where $\mathbf{n}_i$ is the normal vector for the i^{th} sample point and N is the number of samples for the k^{th} link.

The fully-connected neural networks are used to learn the SDF for each robot link. Since we hope to improve the calculation efficiency as much as possible without loss of the distance

estimation accuracy. A deeper network will increase the computational complexity while a shallow network cannot guarantee the accuracy of distance estimation. In order to trade off accuracy and efficiency, the scale of the network is adjusted by trial and error. The final structure is composed of 5 fully-connected layers with all hidden layers being 64-dimensional. The ReLU function is chosen as the activation function. The loss function is defined as follows

$$\mathcal{L}_{\boldsymbol{\theta}_k} = \sum_{\mathcal{D}} \left[f_{\boldsymbol{\theta}_k}(\mathbf{p}) - d\right]^2 + (\mathbf{t}\cdot\hat{\mathbf{n}})^2 + (\hat{\mathbf{t}}\cdot\mathbf{n})^2 \tag{5}$$

where $\hat{\mathbf{t}}$ and $\mathbf{n}$ denote the tangent of the learned SDF network and the measured normal vector at point $\mathbf{p}$, respectively. $\mathbf{t}$ and $\hat{\mathbf{n}}$ denote the measured tangent vector of the point corresponding to the link mesh vertices and the estimated normal vector at point $\mathbf{p}$. The first term aim at penalizing distance errors. The second and third terms are used to penalize the normal vector alignment errors as in [29].

IV. Stochastic Optimization for Robot Trajectory Planning

The optimization-based trajectory planning problem can be formulated as a Bayesian inference problem, following the conventions established in [23] and [30]. Mathematically, this is expressed as:

$$p(\boldsymbol{\tau}|\mathcal{Z}=1) \propto p(\mathcal{Z}=1|\boldsymbol{\tau})p(\boldsymbol{\tau}) \tag{6}$$

where a prior distribution on the trajectory, $p(\boldsymbol{\tau})$, encodes prior knowledge such as initial and goal constraints. $\boldsymbol{\tau} \triangleq [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_H]$ is defined as a sequence of positions of the robot, where $\mathbf{x} = [x, y]^\top \in \mathbb{R}^2$ for a 2D planner mobile robot, and $\mathbf{x} = \boldsymbol{q} \in \mathbb{R}^n$ for an n-DoF robot manipulator. H denotes discrete time steps. The likelihood function, $p(\mathcal{Z}=1|\boldsymbol{\tau})$, encourages the achievement of desired objectives with a set of optimization criteria $\mathcal{Z}$. For instance, this could involve optimizing for obstacle and joint limits avoidance. $\mathcal{Z}=1$ indicates the corresponding criterion is optimized. Thus, the objective is to find the maximum a posteriori (MAP), $p(\boldsymbol{\tau}|\mathcal{Z}=1)$, with a likelihood function that encourages the trajectories to be collision-free.

A. Time-Normalized Gaussian Process Prior

Since the proper prior allows encoding some desired behaviors, to facilitate generating optimal and goal-directed motion trajectory for the robot, we wish to incorporate goal constraints and smoothness, i.e., the correlation of trajectories over time series, into the prior distribution. The main goal of this part is to generate spatial-optimized collision-free trajectories, regardless of trajectory feasibility, i.e., not considering system dynamic constraints here. Besides, instead of constructing the Gaussian process prior at the velocity and acceleration level, the Gaussian process prior is designed at the position level to increase the diversity of trajectory samples for exploring collision-free trajectories. To this end, the simple Time-Normalized Gaussian process with the time t being normalized within interval $[0, 1]$, is proposed to define a prior distribution over the trajectories $\boldsymbol{\tau}$.

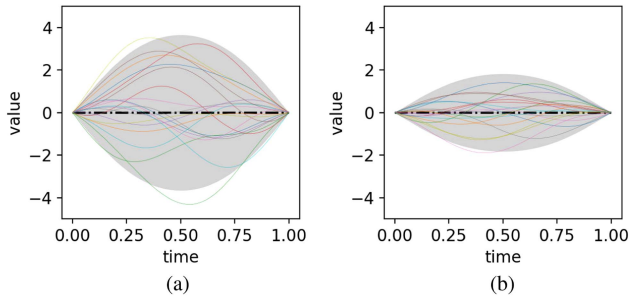


Fig. 3. An example of the time-normalized GP-prior with different σ_f. 20 solid lines are the trajectories sampled from the prior. The black dotted line represents the mean of the prior. (a) $\sigma_f = 4, h = 0.1$. (b) $\sigma_f = 1, h = 0.1$.

Thus, the prior distribution can is modeled as

$$p(\boldsymbol{\tau}) \propto \exp\left\{-\frac{1}{2}\|\boldsymbol{\tau}-\boldsymbol{\mu}\|^2_{\mathcal{K}}\right\} \tag{7}$$

where $\|\boldsymbol{\tau}-\boldsymbol{\mu}\|^2_{\mathcal{K}}$ is the Mahalanobis distance with the Gaussian kernel $\mathcal{K}$. For the general goal-directed trajectory planning problem, the initial and target positions are known in advance. The kernel of the initial Gaussian process prior over trajectory $\boldsymbol{\tau}$ conditioned on the observed initial and goal positions can be estimated by the following formula

$$\mathcal{K} = \mathcal{K}_{**} - \mathcal{K}_*^\top \mathcal{K}_{test}^{-1} \mathcal{K}_* \tag{8}$$

where the Gaussian kernel $\mathcal{K}_*$, $\mathcal{K}_{test}$ and $\mathcal{K}_{**}$ are

$$\begin{aligned} \mathcal{K}_* &= [\kappa\left(t^j, t^i\right)],\ i \in \boldsymbol{\mathcal{C}},\ j \in \boldsymbol{\mathcal{C}}_* \\ \mathcal{K}_{test} &= [\kappa\left(t^j, t^i\right)],\ i, j \in \boldsymbol{\mathcal{C}} \\ \mathcal{K}_{**} &= [\kappa\left(t^j, t^i\right)],\ i, j \in \boldsymbol{\mathcal{C}}_* \end{aligned} \tag{9}$$

where $\boldsymbol{\mathcal{C}}$ denotes the set of test points indexes, i.e., the indexes of the discrete waypoints. $\boldsymbol{\mathcal{C}}_*$ denotes the indexes of the observed initial and goal points, i.e., 0 and H. Each element of the kernel matrix is determined by

$$\kappa\left(t^j, t^i\right) = \sigma_f^2 \exp\left\{-\frac{1}{2\tilde{h}^2}(t^j - t^i)^2\right\},\ t^i, t^j \in [0, 1] \tag{10}$$

where σ_f and h are two hyper-parameters. Therefore, the diversity of the trajectories sampled from the conditional Gaussian process prior could be easily adjusted by the parameter, σ_f, as shown in Fig. 3.

B. The Likelihood Function for Trajectory Planning

The prior distribution primarily enforces goal constraint. Moreover, we also need to define likelihood functions that encourage the achievement of other desired objectives, e.g., collision avoidance and trajectory length penalty. In this case, the Exponential Utility is chosen as the cost-likelihood to quantify each desired objective

$$p(\mathcal{Z} \mid \boldsymbol{\tau}) = \exp\left\{-\frac{1}{2}\sum_{i=1}^{l}\|\ell_i(\boldsymbol{\tau})\|^2_{\boldsymbol{\Sigma}_i}\right\} \tag{11}$$

where l is the number of tasks. The weight matrix $\mathbf{\Sigma}_i$ corresponds to the i^{th} task.

The cost for collision avoidance is defined as follows.

$$\ell_{obs}(\boldsymbol{\tau}) = \sum_{i=0}^{H} \ell_o(\mathbf{x}_i),\ \ell_o(\mathbf{x}_i) = \begin{cases} 1, & \text{if } d(\mathbf{x}_i) \le \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

where $d(\mathbf{x}_i)$ is the minimum distance between K links and obstacle when the robot is at position $\mathbf{x}_i$, estimated using the composite SDF network. ϵ is the distance threshold. The cost for obstacle avoidance is estimated at a series of discrete waypoints. To avoid collisions with delicate objects, e.g., sharp corners and thin-walled obstacles, the evaluation points are added between adjacent waypoints by linear interpolation when calculating the cost for obstacle avoidance.

In addition to the collision avoidance tasks, a trajectory length penalty is also added to penalize the length of the trajectory. Thus, this cost is designed as

$$\ell_{le}(\boldsymbol{\tau}) = \sum_{i=0}^{H} \|\mathbf{x}_{i+1} - \mathbf{x}_i\| \tag{13}$$

Other task-related cost items can be also added as required.

C. Trajectory Optimization and Kernel Updating Strategy

In this part, the goal is to estimate the maximum posterior, $p(\boldsymbol{\tau} \mid \mathcal{Z})$, given the above-mentioned Gaussian process prior and likelihood functions. The main idea is to continuously update this Gaussian process prior so that the trajectories samples from the prior can maximize the likelihood functions. The weighted mean trajectory is the solution result. Thus, it is necessary to update not only the mean of the Gaussian process prior but also the kernel matrix. The mean trajectory is updated using the derivation-free model predictive path integral (MPPI) [27] with weight approximated by Monte Carlo sampling. Let us draw a number of N_s trajectory samples from the Gaussian process prior, $p(\boldsymbol{\tau})$, and estimate the cost-likelihood value of each trajectory. The mean trajectory is updated as follows

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \gamma \sum_{i=1}^{N_s} \omega_i (\boldsymbol{\tau}_i - \boldsymbol{\mu}_t) \tag{14}$$

where the step-size γ is used to get a smooth updating process and the relative probabilistic weight ω_i corresponding to the i^{th} trajectory is determined according to follows

$$\omega_i = \frac{p(\mathcal{Z} \mid \boldsymbol{\tau}_i)}{\sum_{i=0}^{N_s} p(\mathcal{Z} \mid \boldsymbol{\tau}_i)} \tag{15}$$

The kernel matrix, $\mathcal{K}$, can determine the diversity of the sampled trajectories, and it can be adjusted by the parameter σ_f and h. Since h is responsible for controlling the correlation of waypoints in time series that is not the main factor affecting the exploration of collision-free trajectories. To this end, we mainly discuss the strategies of updating σ_f. A large σ_f corresponds to a large sampling region and can results in more diverse trajectories to find the collision-free trajectories. However, a large σ_f also causes the optimization to generate conservative trajectories. To this end, we prefer to use a large σ_f to initially explore collision-free trajectories, and to use a small σ_f to fine-tune the trajectory after finding the collision-free trajectory.

The parameter of σ_f is updated as follows

$$\sigma_f^{t+1} = \begin{cases} \sigma_f^t, & \text{if } \ell_{obs}(\boldsymbol{\mu}_t) > 0 \\ \eta \cdot \sigma_f^t, & \text{if } \ell_{obs}(\boldsymbol{\mu}_t) = 0 \text{ and } \sigma_f^t > \sigma_{\min} \\ \sigma_{\min}, & \text{otherwise} \end{cases} \tag{16}$$

where σ_f^t and σ_f^{t+1} denote the standard deviation of the current and next iteration step. η is an attenuation factor, $0 < \eta < 1$. $\ell_{obs}(\boldsymbol{\mu}_t)$ is the cost value for obstacle avoidance corresponding to the mean trajectory $\boldsymbol{\mu}_t$. $\sigma_{\min}$ refers to the threshold of standard deviation.

D. Time-Optimal Path Parameterization

The above-generated trajectories are spatially optimal, while it is not feasible, i.e., without considering system dynamics. Thus it is necessary to allocate the timing for the trajectories to generate time-optimized trajectories while considering the system dynamics, velocity, and acceleration constraints. The off-the-shelf Reachability Analysis-based time-optimal path parameterization algorithm (TOPP-RA) is used to generate a time-optimal trajectory [31]. The TOPP-RA algorithm uses a cubic spline curve to fit the waypoints which also ensures the smoothness of the trajectory. As a result, the output of the time-optimal planner is the final trajectory which is not only spatial optimal but also temporal optimal.

V. Simulation and Experiment Validation

Both simulations and experiments have been carried out to verify the effectiveness and performance of the collision-free motion generation algorithm. First, we tested the computational efficiency and accuracy of the composite SDF network. Then, the trajectory planning algorithm is tested with a 2D mass point. Finally, the learned SDF networks are integrated into the trajectory planning algorithm with the Franka robot to generate collision-free trajectories. All the algorithms are implemented in Python and Pytorch which allows for batch operations and run on a laptop with AMD Core i7-9700 and Geforce RTX 3060 GPU.

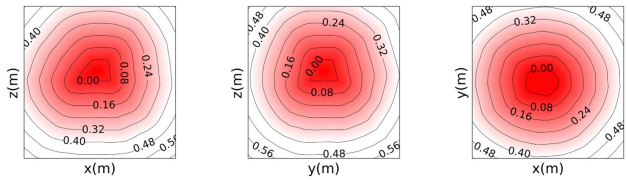
A. Evaluation of Composite SDF Networks

The composite SDF Networks are trained for the Franka robot which only takes a few minutes before converging for each link. This is very convenient for adjusting the hyper-parameters and can be easily applied to other robots. Both the accuracy and computational time of the learned SDF networks are evaluated.

The average query computation time with different numbers of 3D query points for 9 links is investigated, and compared with the most related Neural-JSDF algorithm [16] and the standard Gilbert-Johnson-Keerthi (GJK) algorithm (not parallelized) [5]. The average computation times for both methods are listed in Table I. The average computation time of Composite SDF is slightly longer than that of the Neural-JSDF algorithm and increases with the number of query points. Due to the benefits of batch operations, the average calculation time of SDF is much

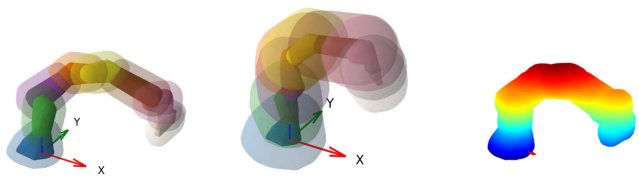
TABLE I
COMPUTATIONAL TIME OF THE LEARNED SDF NETWORK

num. query points	1	100	10000
composite SDF, ms	4.33	4.78	6.63
Neural-JSDF, ms	3.32	3.62	4.02
GJK, ms	3.69	241	21960



(a) (b) (c)

Fig. 4. The reconstructed signed distance field with $f_{\boldsymbol{\theta}}(\mathbf{p}) \geq 0$ m, for three sections of $link0$ of the Franka robot. (a) section xz (y=0 m), (b) section yz (x=0 m), (c) section xy (z=0 m).



(a) (b) (c)

Fig. 5. The reconstructed distance isosurfaces (transparent) with (a) $f_{\boldsymbol{\theta}}(\mathbf{p}) = 0.05$m and (b) $f_{\boldsymbol{\theta}}(\mathbf{p}) = 0.15$m for each link of the Franka Emika Panda. (c) the merged isosurfaces at $f_{\boldsymbol{\theta}}(\mathbf{p}) = 0.05$m.

shorter than that of the GJK algorithm when the number of query points is 100 and 10000. Moreover, if some links are not required for collision detection, the calculation time can be further reduced.

To intuitively demonstrate the accuracy of the learned model, we use the learned SDF network to reconstruct the SDF with $f_{\boldsymbol{\theta}}(\mathbf{p}) \geq 0$ m, on three sections of the $link0$, i.e., the blue link in Fig. 5(a), of the Franka robot, and the results are shown in Fig. 4. The reconstructed SDF can well reflect the profile of the $link0$ near the surface, and the shape of the isoline tends to be circular as it is far away from the link. Besides, the widths between adjacent isolines are almost equal, also indicating that the learned model can reconstruct an accurate SDF for the robot link.

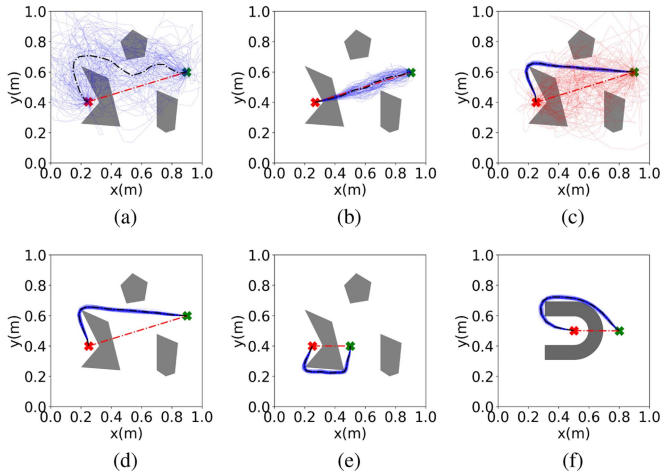
The original mesh geometry (solid) and reconstructed distance isosurface $f_{\boldsymbol{\theta}}(\mathbf{p}) = 0.05$ m, $f_{\boldsymbol{\theta}}(\mathbf{p}) = 0.15$ m (transparent) and the merged isosurface of Franka robot at different poses using ray marching algorithms [32], are shown in Fig. 5(a), (b), and (c). The reconstructed distance isosurface can well reflect the shape of each link. The root-mean-square deviation (RMSD) of the estimated distance $f_{\boldsymbol{\theta}}(\mathbf{p})$ and normal vector $\mathbf{n}$, i.e., the root mean square of the second and third terms in (5), of all links at different distance intervals are listed in Table II. Moreover, our method has higher accuracy at two distance intervals, i.e., [0, 0.1] and [0.1, 0.12] m, compared with the Neural-JSDF

TABLE II
ACCURACY OF THE LEARNED SDF NETWORK

distance interval (m)	[0, 0.4]	[0.4,0.8]	[0.8, 1.2]
RMSD, $f_{\boldsymbol{\theta}}(\mathbf{p})$ (cm)	0.28	0.36	0.38
RMSD, $\hat{\mathbf{n}}$	0.083	0.045	0.042

TABLE III
ACCURACY COMPARISON WITH $0 < d < 0.1$ M | $0.1 < d < 1.2$ M

method	composite SDF	Neural-JSDF
RMSD, $f_{\boldsymbol{\theta}}(\mathbf{p})$ (cm)	**0.21** | **0.36**	1.04 | 1.06



(a) (b) (c) (d) (e) (f)

Fig. 6. (a) and (b) denote the generated trajectories of a point mass without using the kernel updating strategies. (c)–(f) are generated trajectories of a point mass using kernel updating strategies.

algorithm [16], and the results are shown in Table III. The average distance error of the distance isosurface in the robot workspace is not greater than 3.8 mm.

B. Trajectory Planning Simulation for 2D Mass Point

To verify the effectiveness of the trajectory planning algorithm more intuitively. We first evaluate the algorithm for generating collision-free trajectories for a 2D mass point in dummy environments as shown in Fig. 6. In this simulation, the likelihood function consists of collision avoidance and trajectory length penalties. The initial mean trajectory of the Gaussian process prior is a straight line between the start and goal positions as shown by the red dotted lines in Fig. 6. The discrete steps H of the trajectory are 20 and the optimization iteration steps are 200. The parameter h in (10) is 0.01 and N_s in (14) is 200 for all 2D simulation tests.

The black dotted lines in Fig. 6(a) and (b) are the optimized trajectories without using the kernel updating strategies. The parameter σ_f of (a) and (b) are 0.02 and 0.001, respectively. The planning algorithm can generate a collision-free trajectory with $\sigma_f = 0.02$ as shown in Fig. 6(a), but the results are relatively conservative, i.e., the trajectory is far away from obstacles and

not short. The planning algorithm cannot generate a collision-free trajectory with $\sigma_f = 0.001$ as shown in Fig. 6(b), and sticks into a local optimum. This comparison shows that larger σ_f is beneficial to explore collision-free trajectories, and the trajectory is easy to stick into the local optimum with smaller σ_f. For the 2D point mass simulation demonstrations, one can refer to the video in the attachment.

The initial and optimized trajectories of point mass using the kernel updating strategies are shown in Fig. 6(c)–(f). All of them are set with a large initial $\sigma_f = 0.02$ and minimum allowable $\sigma_{\min} = 0.0005$ and the corresponding trajectory samples are shown by the red and blue solid lines in Fig. 6(c), respectively. The difference between (c) and (d) is that 5 linear interpolation points between adjacent waypoints are taken into consideration in (d) while estimating the cost for obstacle avoidance. All the 20 waypoints in (c) and (d) are located in the collision-free white area. As the goal position changes, the trajectories to be optimized have different lengths and complexities. Besides, the initial and target positions of the mass point are placed on both sides of the V-shaped and U-shaped obstacles, as shown in Fig. 6(e) and (f). The proposed trajectory planning algorithms can generate collision-free trajectories in both trap scenarios. The results show that the proposed trajectory planning algorithm has a certain ability to escape from the local optimum.

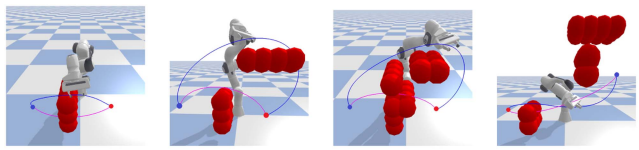
C. Evaluation of the Whole Algorithms

In this part, the learned composite SDF networks are integrated into the trajectory planning algorithm, and the whole algorithm is tested on the Franka robot manipulator to verify the effectiveness and feasibility of the algorithm.

1) Testing Scenarios and Parameter Settings: The robot manipulator attempts to perform tasks of goal-reaching in Cartesian space while avoiding collision with obstacles. Since the Gaussian process prior is constructed in the configuration space, Inverse Kinematics (IK) is used to estimate the target configuration. The workbench consists of a 7-DOF Franka robot manipulator with a fixed base and a Kinect-V2 camera mounted on a fixed position to detect obstacles. In this case, the Aruco markers are used for hand-eye calibration and obstacle position detection.

The entire trajectory is discretized into 20 waypoints, i.e., $H = 20$ steps, and as calculating the cost for obstacle avoidance, two interpolation points between adjacent waypoints are considered. In addition to the cost for obstacle avoidance and trajectory length penalty, the likelihood function also contains a cost for boundary avoidance that constrains the z-position of the robot, i.e., the z-position of the link should not be less than 0.02 m. The distance threshold is $\epsilon = 0.08$ m. The number of particles used to update the mean trajectory is $N_s = 50$. The initial value of $\sigma_f = 0.1$ and the minimum allowable $\sigma_f = 0.012$. The length scale is $l = 0.5$. The joint speed and acceleration limits of the robot for the TOPP-RA-based time-optimal planner are set to 0.1 times the preset limit value of the Franka robot.

2) Evaluation Results: First, the proposed algorithms are applied to generate collision-free trajectories in the 4 simulation scenarios as shown in Fig. 7. The pink curves denote the initial mean trajectory of the end-effector, and the blue curve is the trajectory obtained after optimization. We also repeat the planning 20 times in each scenario with the CHOMP [2], and RRT-Connect [17] algorithms already included in Moveit with default parameters. The success rate and the average relative length of trajectories in joint space are listed in Table IV. The results show the success rate of both the proposed method and the RRT-Connect algorithm is 1, which is higher than the CHOMP. Besides, the planned trajectories have the shortest relative length compared with the successfully planned trajectories using the other two methods.



(a) (b) (c) (d)

Fig. 7. Screenshot of 4 different obstacle avoidance scenarios. The pink curve is the initial mean trajectory, and the blue curve is the optimized collision-free trajectory. The red and blue points are the initial and goal position of the end-effector.

TABLE IV
COMPARISON WITH OTHER TRAJECTORY PLANNING ALGORITHMS

method	proposed	CHOMP	RRTconnect
success rate	1.0	0.25	1.0
path length	1.0	1.06	1.17

(a) (b) (c) (d)

Fig. 8. Screenshot of Franka robot performing collision avoidance motion.

Then, the proposed algorithms are tested in a scenario, as shown in Fig. 8(a) to (c). In this workspace, the black box is an obstacle that is simplified as a set of spheres. The manipulator needs to move from the right side of the box to its left side. The initial (pink curve) and the optimized (blue curve) trajectory of the end-effector are shown in Fig. 8(d). The robot can move from the initial configuration to the goal pose while avoiding collision with the black box, as shown in Fig. 8. Fig. 9 denotes the curve of the cost for obstacle avoidance corresponding to the mean trajectory and the evolution curve of σ_f during the iteration process. The cost for obstacle avoidance quickly converges to 0, i.e., after 3 iterations as shown by the black solid line in Fig. 9. σ_f decreases from 0.1 to 0.012 after the cost for obstacle avoidance reaches 0 as shown by the blue solid line in Fig. 9. The robot can avoid collision with the black box and generate collision-free trajectories. The results of simulations and experiments show

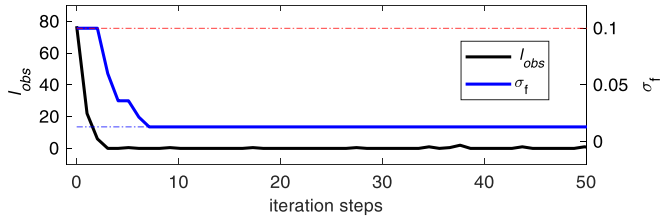


Fig. 9. The curve of cost for obstacle avoidance ℓ_{obs} corresponding to the mean trajectory and the evolution curve of σ_f during the iteration process.

that the whole algorithm is effective. For the demonstrations, please refer to the attached video.

VI. Conclusion

We have proposed composite SDF networks for articulated robots, which can quickly and accurately calculate the minimum distance between the articulated robot and obstacles combined with the kinematics of the articulated robot. Although the learned composite SDF networks contain many parameters, it still runs fast due to the high degree of parallelization. In addition, we also propose a stochastic trajectory optimization method based on the Gaussian process prior and kernel function update strategy, which can efficiently generate goal-directed trajectories for high-dimensional robots. The introduction of the MPPI with kernel update strategy to explore collision-free trajectories endows the optimization process with a certain ability to escape from local optimum and does not generate overly conservative trajectories. Moreover, the composite SDF model is integrated into the trajectory generation algorithm which can generate optimized and collision-free trajectories for high DoF robot manipulators.

Although the composite SDF networks are used for offline trajectory planning, this network can be also used for obstacle avoidance in dynamic environments. Thus, in the future, we would also like to integrate the composite SDF networks into a real-time motion planning and control method to ensure collision-free motion generation in a dynamic environment. The limitation of this network is that it is not able to estimate the gradients of distances with respect to robot joint positions.

References

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, and W. Burgard, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA, USA: MIT Press, 2005.

[2] M. Zucker et al., "CHOMP: Covariant hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, no. 9/10, pp. 1164–1193, 2013.

[3] B. Schmidt, "Real-time collision detection and collision avoidance," in *Adv. Hum.-Robot Collaboration in Manuf.* Berlin, Germany: Springer, 2021, pp. 91–113.

[4] H.-C. Lin, C. Liu, Y. Fan, and M. Tomizuka, "Real-time collision avoidance algorithm on industrial manipulators," in *Proc. IEEE Conf. Control Technol. Appl.*, 2017, pp. 1294–1299.

[5] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE J. Robot. Automat.*, vol. 4, no. 2, pp. 193–203, Apr. 1988.

[6] J. Corrales, F. Candelas, and F. Torres, "Safe human-robot interaction based on dynamic sphere-swept line bounding volumes," *Robot. Comput.-Integr. Manuf.*, vol. 27, no. 1, pp. 177–185, 2011.

[7] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for MAVs in dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 776–783, Apr. 2019.

[8] B. Lacevic and P. Rocco, "Kinetostatic danger field — A novel safety assessment for human-robot interaction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 2169–2174.

[9] M. P. Polverini, A. M. Zanchettin, and P. Rocco, "A computationally efficient safety assessment for collaborative robotics applications," *Robot. Comput.- Integr. Manuf.*, vol. 46, pp. 25–37, 2017.

[10] J. Pan and D. Manocha, "Efficient configuration space construction and optimization for motion planning," *Engineering*, vol. 1, no. 1, pp. 046–057, 2015.

[11] N. Das and M. Yip, "Learning-based proxy collision detection for robot motion planning applications," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1096–1114, Aug. 2020.

[12] Y. Zhi, N. Das, and M. Yip, "DifCo: Autodifferentiable proxy collision detection with multiclass labels for safety-aware trajectory optimization," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 2668–2685, Oct. 2022.

[13] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feed forward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[14] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.

[15] G. Sutanto, I. R. Fernández, P. Englert, R. K. Ramachandran, and G. Sukhatme, "Learning equality constraints for motion planning on manifolds," in *Proc. Conf. Robot Learn.*, 2021, pp. 2292–2305.

[16] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 480–487, Feb. 2023.

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

[18] Z. Kingston, M. Moll, and L. E. Kavraki, "Decoupling constraints from sampling-based planners," in *Proc. Robot. Res.: 18th Int. Symp.*, 2020, pp. 913–928.

[19] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Proc. Conf. Robot.: Sci. Syst.*, 2013, pp. 1–10.

[20] M. Toussaint, "Newton methods for k-order Markov constrained motion problems," 2014, *arXiv:1407.0414*.

[21] J. Ichnowski, M. Danielczuk, J. Xu, V. Satish, and K. Goldberg, "GOMP: Grasp-optimized motion planning for bin picking," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 5270–5277.

[22] J. Ichnowski, Y. Avigal, Y. Liu, and K. Goldberg, "GOMP-FIT: Grasp-optimized motion planning for fast inertial transport," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 5255–5261.

[23] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time Gaussian process motion planning via probabilistic inference," *Int. J. Robot. Res.*, vol. 37, no. 11, pp. 1319–1340, 2018.

[24] A. Lambert and B. Boots, "Entropy regularized motion planning via stein variational inference," 2021, *arXiv:2107.05146*.

[25] S. Anderson, T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch nonlinear continuous-time trajectory estimation as exactly sparse Gaussian process regression," *Auton. Robots*, vol. 39, pp. 221–238, 2015.

[26] J. Urain, A. T. Le, A. Lambert, G. Chalvatzaki, B. Boots, and J. Peters, "Learning implicit priors for motion optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 7672–7679.

[27] G. Williams et al., "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1714–1721.

[28] Q. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," 2018, *arXiv:1801.09847*.

[29] 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 Int. Conf. Intell. Robots Syst.*, 2022, pp. 6673–6680.

[30] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 1049–1056.

[31] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 645–659, Jun. 2018.

[32] J. C. Hart, "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces," *Vis. Comput.*, vol. 12, no. 10, pp. 527–545, 1996.