PC-Planner: Physics-Constrained Self-Supervised Learning for Robust Neural Motion Planning with Shape-Aware Distance Function

XUJIE SHEN*, Zhejiang University, China
HAOCHENG PENG*, Zhejiang University, China
ZESONG YANG, Zhejiang University, China
JUZHAN XU, Shenzhen University, China
HUJUN BAO, Zhejiang University, China
RUIZHEN HU, Shenzhen University, China
ZHAOPENG CUI†, Zhejiang University, China

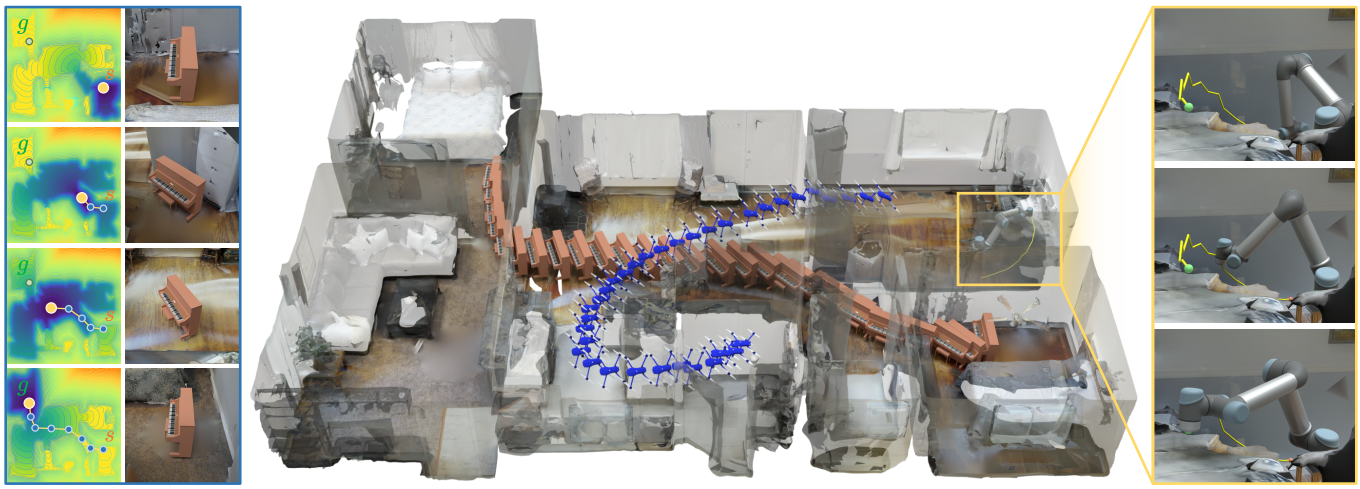


Fig. 1. Given a prebuilt complex 3D environment, our PC-Planner can learn the time fields and execute motion planning for robots of various shapes from any start state to any goal state in a self-supervised manner. Left: Time fields for a piano navigating through the environment, in which the rainbow color scheme is used. Right: Close-up views of the manipulation of a UR5 robot.

Motion Planning (MP) is a critical challenge in robotics, especially pertinent with the burgeoning interest in embodied artificial intelligence. Traditional MP methods often struggle with high-dimensional complexities. Recently neural motion planners, particularly physics-informed neural planners based on the Eikonal equation, have been proposed to overcome the curse of dimensionality. However, these methods perform poorly in complex scenarios with shaped robots due to multiple solutions inherent in the Eikonal equation. To address these issues, this paper presents PC-Planner, a novel physics-constrained self-supervised learning framework for robot motion planning with various shapes in complex environments. To this end, we propose several physical constraints, including monotonic and optimal constraints, to stabilize the training process of the neural network with the Eikonal equation. Additionally, we introduce a novel shape-aware distance field that considers the robot's shape for efficient collision checking and Ground Truth (GT) speed computation. This field reduces the computational intensity, and facilitates adaptive motion planning at test time. Experiments in diverse scenarios with different robots demonstrate the superiority of the proposed method in efficiency and robustness for robot motion planning, particularly in complex environments. Code and data are available on the project webpage: https://zju3dv.github.io/pc-planner.

CCS Concepts: • **Computing methodologies** → **Robotic planning**; **Motion path planning**.

Additional Key Words and Phrases: motion planning, robot navigation, Eikonal equation, self-supervised learning

ACM Reference Format:
Xujie Shen, Haocheng Peng, Zesong Yang, Juzhan Xu, Hujun Bao, Ruizhen Hu, and Zhaopeng Cui. 2024. PC-Planner: Physics-Constrained Self-Supervised Learning for Robust Neural Motion Planning with Shape-Aware Distance Function. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–6, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3680528.3687651

*Xujie Shen and Haocheng Peng contributed equally to this work.
†Corresponding author.

SA Conference Papers '24, December 3–6, 2024, Tokyo, Japan

ACM ISBN 979-8-4007-1131-2/24/12
https://doi.org/10.1145/3680528.3687651

1 INTRODUCTION

Motion planning (MP) is a long-standing problem in robotics that aims to find a trajectory from a start configuration to a goal configuration while satisfying all constraints like collision avoidance. With the rapid advancement of embodied artificial intelligence, this problem has garnered increasing attention. The traditional MP methods normally adopt sampling techniques to explore the robot's obstacle-free state-space and construct feasible paths [Gammell et al. 2015; Karaman and Frazzoli 2011; Kingston et al. 2018]. However, these methods often encounter limitations in high-dimensional spaces with increased computational complexity and reduced efficiency.

Recently learning-based methods [Chaplot et al. 2021; Li et al. 2021; Wang et al. 2020], i.e., neural motion planners, have been proposed to solve the curse of dimensionality. Most of the learning-based methods [Huh et al. 2021; Li et al. 2021] utilize the deep neural network to predict the motions iteratively to improve the efficiency of planning in high dimensions. However, these methods face some significant limitations. Firstly, these data-driven methods heavily rely on expert training data such as the trajectories from the traditional methods, resulting in time-consuming data generation processes, particularly for high-dimensional spaces. Additionally, they typically employ constant velocity paradigms which are not suitable for real scenarios [Vysockỳ et al. 2019]. In contrast, some recently proposed physics-informed methods [Ni and Qureshi 2023a,b] offer a compelling alternative. These methods first predefine a speed field that considers velocity constraints based on the geometry of obstacles. Utilizing this predefined speed field, they employ neural networks to solve the Eikonal equation for motion planning. The training data can be efficiently generated by sampling within the speed field, thereby circumventing the need for expert data and significantly reducing data generation time. However, these physics-informed methods struggle in complex environments with shaped robots due to the multiple solutions inherent in the Eikonal equation. Consequently, they may produce time fields with local minima, as shown in Fig. 2 (left), leading to infeasible paths and thus poor performance in complex planning tasks. Unlike some optimization-based methods, where "local minima" refers to feasible but sub-optimal solutions, in our context, local minima in the time fields lead to infeasible solutions, thereby reducing the success rates.

In this paper, we present PC-Planner, a novel physics-constrained self-supervised learning framework for neural robot motion planning based on a new shape-aware distance field, effectively addressing the local minima in the time fields and thus significantly outperforming existing physics-informed methods in success rates.

Based on a deep analysis of the properties of the Eikonal equation, we propose two physical constraints and incorporate them into the training process of the neural network with the Eikonal equation for motion planning in a self-supervised manner. Specifically, we introduce the *monotonic constraint* to identify the local minima and the *optimal constraint* to preserve the optimality of the solution to the Eikonal equation. These constraints are seamlessly integrated into a self-supervised training framework, allowing the network to recognize situations where the solution is trapped in local minima and enabling it to self-correct by adhering to the defined physical rules, as shown in Fig. 2 (right).

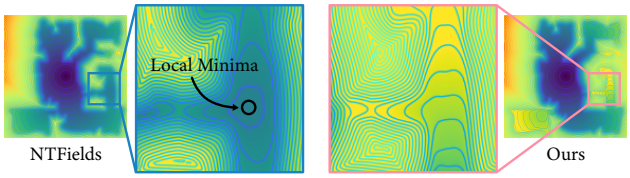


Fig. 2. Comparison of time fields for Gibson. NTFields generates an incorrect time field with local minima due to the inherent multiple solutions in the Eikonal equation, while our PC-Planner learns to generate the correct time field with the proposed physical constraints.

However, these physical constraints are only applied to the collision-free paths, which introduces significant computational overhead for trajectory collision checks during training. To address this issue, we further propose a novel shape-aware distance field (SADF) that models the minimum distance from the robot with any shape and configuration to the environment. With this neural implicit field, we can efficiently conduct collision checking and apply the physical constraints during the training process of the physics-informed neural model. Meanwhile, our SADF can be efficiently converted into the required speed field for training the physics-informed model, and further exploited during the test stage for collision avoidance with adaptive path planning,

We analyze and validate the effectiveness of our method through experiments in diverse scenarios with various robots, demonstrating the superiority of PC-Planner over the existing methods.

Our main contributions can be summarized as follows:

- We introduce a novel physics-constrained self-supervised learning approach for physics-informed neural robot motion planning, which enables efficient and robust motion planning for robots with various shapes in complex scenarios.
- We propose two physical constraints to enable the network to jump out of local minima and converge to the correct solutions that obey the physical rules.
- We develop a new neural shape-aware distance field for collision checking that can predict the minimum distance to the environment for any robot with arbitrary shapes and configurations in the fixed environment, which facilitates both self-supervised training and test stages.

2 RELATED WORK

Our work focuses on the problem of robust motion planning for robots with arbitrary shapes in challenging scenarios. Here, we review the current state of research on motion planning and implicit neural distance fields.

Motion Planning. Existing optimal path planning methods include sampling-based approaches[Gammell et al. 2015; Karaman and Frazzoli 2011; LaValle and Kuffner Jr 2001; Tukan et al. 2022] that explore the environment through random state sampling to retrieve feasible paths, optimization-based methods[Kalakrishnan et al. 2011; Kurenkov et al. 2022; Mukadam et al. 2016] which minimize a defined cost while meeting constraints to find the optimal trajectory,

etc. [Yang et al. 2019]. While effective for high-dimensional tasks, they suffer from high computational costs and unstable solutions due to their sensitivity to initial conditions. Neural motion planners (NMPs)[Chaplot et al. 2021; Ichter et al. 2018; Johnson et al. 2023; Li et al. 2021; Wang et al. 2020] have emerged to balance efficiency and stability, using expert trajectories from classic methods for training. However, generating training data from traditional methods is computationally expensive, limiting their flexibility.

The most pertinent method to our work is the physics-driven method. The earliest of these methods is FMM[Chopp 2001; Sethian 1996; Treister and Haber 2016; White et al. 2020] which numerically solves the Eikonal equation. However, its computational complexity increases dramatically with dimensionality. Recently, NTFields[Ni and Qureshi 2023a] has introduced a continuous-time path planning method encoding the Eikonal equation directly into the network, eliminating the need for expert trajectory supervision. Subsequently, [Ni and Qureshi 2023b] incorporates a viscosity term and progressive learning for smoother path solutions. Despite their success, these methods struggle with challenging scenarios affected by local minima. Our approach addresses the issues, ensuring high success rates and fast inference across diverse scenarios including high-dimensional planning, and various shapes of robots.

Implicit distance representation. The contemporary approach [Chabra et al. 2020; Jiang et al. 2020; Ouasfi and Boukhayma 2022] to learning Signed Distance Fields (SDF) primarily involves employing neural networks to regress the mapping between 3D coordinates to the signed distances. Similar work can be traced back to [Park et al. 2019], where a neural signed distance function was introduced. The function allows querying the shortest distance between the object surface and continuous spatial points. However, it is designed for point queries, and when dealing with a shaped robot as the target, such distance functions cannot be directly applied. Recent works like [Chen et al. 2021; Chou et al. 2022; Zhu et al. 2023] excel in accurately representing the shape of objects, encompassing the capability to handle key properties such as scaling and rotation. These methods serve as inspiration for our proposed shape-aware distance function, allowing us to characterize the distance between the robot and the environment.

3 PRELIMINARIES

3.1 Robot Motion Planning

Let $C \subset \mathbb{R}^d$ and $\mathcal{X} \subset \mathbb{R}^m$ represent the configuration space (c-space) of the robot and its surrounding environment, where $d, m \in \mathbb{N}$ denote the dimensions. The obstructed c-space, formed by the obstacles in the environment $\mathcal{X}_{obs} \subset \mathcal{X}$, is denoted as C_{obs}. Additionally, the feasible space in c-space and the environment is represented as $C_{free} = C \backslash C_{obs}$ and $\mathcal{X}_{free} = \mathcal{X} \backslash \mathcal{X}_{obs}$ respectively. Given the robot start $\mathbf{s} \in C_{free}$ and goal $\mathbf{g} \in C_{free}$ configurations, the objective of robot motion planning algorithms is to find a collision-free trajectory $\sigma = \{\mathbf{c_0}, ..., \mathbf{c_i}, ..., \mathbf{c_n}\} \subset C_{free}\ s.t.\ \mathbf{c_i} \in C_{free}$, where $\mathbf{c_0} = \mathbf{s}$, $\mathbf{c_n} = \mathbf{g}$, and $\mathbf{c_i}$ denotes the waypoint along the trajectory.

3.2 NTFields

The recent work NTFields [Ni and Qureshi 2023a] introduces a new perspective that relates motion planning problems with the solution to the Eikonal equation. The Eikonal equation defines the wave propagation with a first-order, nonlinear PDE formulation:

$$S(\mathbf{g})^{-1} = \left\| \nabla_{\mathbf{g}} T(\mathbf{s}, \mathbf{g}) \right\|, \tag{1}$$

where $\mathbf{s}$ and $\mathbf{g}$ are start and goal configurations, $T(\mathbf{s}, \mathbf{g})$ represents the arrival (travel) time from $\mathbf{s}$ to $\mathbf{g}$, and $\nabla_{\mathbf{g}} T(\mathbf{s}, \mathbf{g})$ denotes the partial derivative of arrival time with respect to the goal point. The solution to the equation yields the minimum arrival time of the wave from the start point to the goal point under the predefined speed model.

NTFields employs the multilayer perceptron (MLP) to model the time field T, which is the solution to the Eikonal equation. The neural network, parameterized by Θ, takes the robot's start and goal configuration $(\mathbf{s}, \mathbf{g})$ as input and outputs the time field, namely, the Time Field Regressor:

$$T_\Theta(\mathbf{s}, \mathbf{g}) = \mathrm{MLP}(\mathbf{s}, \mathbf{g}; \Theta). \tag{2}$$

The ground truth speed model is defined as:

$$S^*(\mathbf{c}) = \frac{S_{const}}{d_{max}} \times clip\left(D_{[r, \mathcal{X}_{obs}]}(\mathbf{c}), d_{min}, d_{max}\right), \tag{3}$$

where r denotes the robot and D is a function that computes the shortest distance between the robot at configuration $\mathbf{c} \in C$ and spatial obstacles $\mathcal{X}_{obs}$. This distance is obtained by sampling points on the surface of the robot and calculating the minimum distance from these points to the obstacles (implemented using BVH [Karras 2012]). The $clip$ function truncates the distance within the range of the minimum distance, d_{min}, and the maximum distance, d_{max}. S_{const} is a speed hyper-parameter, signifying that if the distance between the robot surface and the obstacle surpasses d_{max}, a maximum speed upper limit is imposed. By providing the GT speed model which implicitly encodes obstacle information, NTFields can be trained with these GT speed and predicted speed which is derived from the predicted time according to Eq. (1). However, the BVH-based speed model is computationally expensive, especially when the robot has a complex shape. Our SADF (detailed in Sec. 4.2) offers an accelerated approach to calculating the speed field.

3.3 Viscosity Solution of Eikonal Equation

The non-uniqueness of the solution for the Eikonal equation can cause local minima when applied to motion planning as mentioned in [Sethian 1999]. To solve this problem, the viscosity solution of the Eikonal equation is normally utilized with the following definition:

Definition 3.1. The function T is said to be the viscosity solution of the Eikonal equation provided that for all smooth test function v,

1. *if $T - v$ has a local maximum at a $\boldsymbol{x_0}$, then $G(\boldsymbol{x_0}, \nabla v(\boldsymbol{x_0})) \leq 0$*
2. *if $T - v$ has a local minimum at a $\boldsymbol{x_0}$, then $G(\boldsymbol{x_0}, \nabla v(\boldsymbol{x_0})) \geq 0$*

where $G(\mathbf{x}, \nabla T)$ is the general static Hamilton-Jacobi equation for the Eikonal equation:

$$G(\mathbf{x}, \nabla T) = 0. \tag{4}$$

As proved in [Crandall and Lions 1983; Sethian 1999], we have the following theorem:

Theorem 3.2 (Existence and uniqueness). *There exists a unique viscosity solution T of the Eikonal equation.*

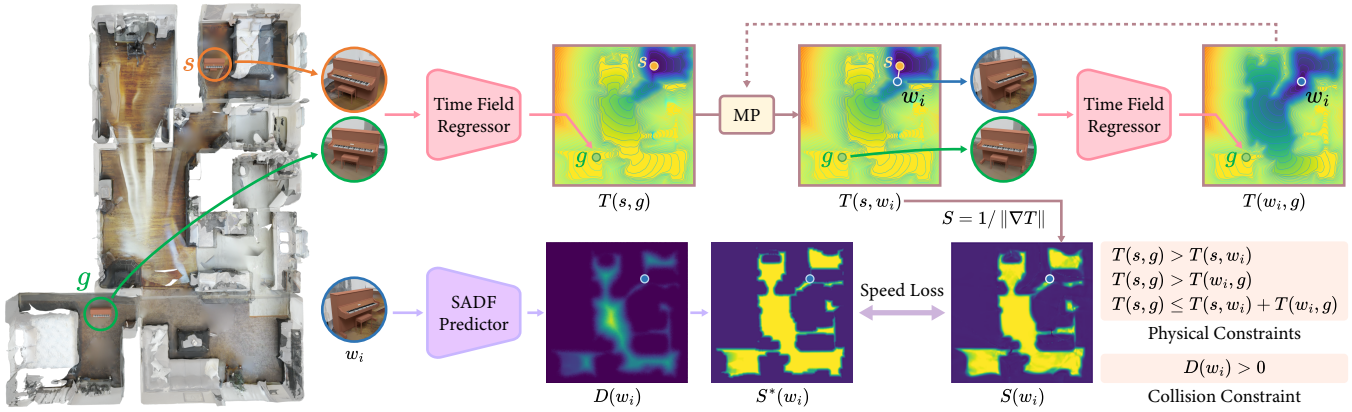


Fig. 3. The PC-Planner integrates a physics-constrained self-supervised learning framework with a shape-aware distance field. The start configuration $\mathbf{s}$ and goal configuration $\mathbf{g}$ are utilized to predict the time $T(\mathbf{s},\mathbf{g})$ through the time field regressor. The travel times $T(\mathbf{s},\mathbf{g})$, $T(\mathbf{s},\mathbf{w_i})$, and $T(\mathbf{w_i},\mathbf{g})$ are employed to incorporate physical constraints during the training of the time field in a self-supervised manner to reduce local minima. It is essential that the waypoint $\mathbf{w_i}$ remains collision-free, which can be ensured by distance $D(\mathbf{w_i})$ predicted through SADF. Moreover, $D(\mathbf{w_i})$ can also be converted into the ground truth speed $S^*(\mathbf{w_i})$ of $\mathbf{w_i}$ to compute the speed loss with the predicted speed $S(\mathbf{w_i})$, which is determined using the time $T(\mathbf{s},\mathbf{w_i})$ and Eq. (1). During training, the motion planning (MP) iterates to derive the waypoint for physical constraint loss, while during testing, it iteratively computes waypoints to generate a path solution.

In other words, the uniqueness of the viscosity solution can inherently address the issue of local minima for the Eikonal equation. To construct the viscosity solution, Fast Marching Methods (FMM) [Sethian 1996] are commonly employed. FMM discretizes the c-space and ensures the viscosity solution by:

Theorem 3.3. *As the discrete space size goes to zero, the numerical solution built by the Fast Marching Methods converges to the viscosity solution of the Eikonal equation.*

The proof of this theorem is provided in [Barles and Souganidis 1991; Tugurlan 2008]. However, FMM has two limitations: 1) the discrete space size cannot practically reach zero implying that the numerical solution may not always converge to the viscosity solution; and 2) the computational burden of FMM increases significantly in high-dimensional spaces, as the number of discrete grids required grows dramatically with the dimensionality.

To overcome those limitations, we utilize neural networks to solve the Eikonal equation, which is more efficient. Meanwhile, instead of pursuing the exact viscosity solution, we introduce our physical constraints to reduce the local minima in the training process, which will be detailed in Sec. 4.1.

4 METHOD

For any start-goal pair, our PC-Planner employs the time field to compute the travel time and its partial derivatives, which are then used in an iterative motion planning process to generate the final path. To train a time field for a new environment, we propose a physics-constrained self-supervised training framework (Sec. 4.1) incorporating a SADF (Sec. 4.2) that is crucial in data preprocessing (GT generation), training (efficient collision-checking), and testing (rapid collision-checking in adaptive motion planning (Sec. 4.3)).

As illustrated in Fig. 3, in our self-supervised framework, pairs of the start and goal configurations are first regressed into the time field through *Time Field Regressor*. Subsequently, the time field is employed to predict the speed and determine the waypoint along the trajectory through motion planning (*MP* module). The physical constraints are then applied to the start, waypoint, and goal configurations in a self-supervised manner, which can help the network escape local minima and converge to the correct solutions consistent with physical principles. Moreover, the proposed *SADF Predictor* is utilized to obtain the shortest distance of any given configuration to the environment. It facilitates collision checking to ensure the collision-free status of the start, waypoint, and goal configurations for the physical constraints and aids in generating GT speed fields in the data processing stage. For ease of illustration, the visualized time field represents the travel time from a fixed start configuration $\mathbf{c_s}$ to any goal configuration $\mathbf{c_i} \in C$ for the robot and the visualized speed field denotes the speed of the robot at any configuration $\mathbf{c} \in C$. Additionally, both fields are visualized in 2D space for clarity.

4.1 Physics-Constrained Self-Supervised Learning

As previously mentioned, the local minima in the solution of the NTFields are caused by the non-uniqueness of the solution for the Eikonal equation. Motivated by traditional methods like FMM, in this section, we introduce a novel self-supervised strategy with two physical constraints related to the viscosity solution, *monotonic constraint* and *optimal constraint*, to enhance the physics-informed neural motion planner by escaping local minima.

Monotonic Constraint. From Theorems 3.2 and 3.3, we have the following property [Sethian 1999]:

Property 4.1. *Travel time solved by FMM increases monotonically away from the start point towards the goal.*

This indicates that the travel time through any waypoint in the path does not shortcut the overall travel time from start to goal. Based on this, we introduce the monotonic constraint which implicitly enforces the monotonicity property in the neural network.

Specifically, for any waypoint along the planned path, the total travel time from the start to the goal configuration must exceed both the travel time from the start to the waypoint and from the waypoint to the goal configuration. More formally, let $T(\mathbf{x}, \mathbf{y})$ denote the travel time from configuration $\mathbf{x}$ to $\mathbf{y}$. We have:

$$T(\mathbf{s}, \mathbf{g}) > T(\mathbf{s}, \mathbf{w}), \tag{5}$$

$$T(\mathbf{s}, \mathbf{g}) > T(\mathbf{w}, \mathbf{g}), \tag{6}$$

where $\mathbf{s}$, $\mathbf{g}$ and $\mathbf{w} \in C_{free}$ represent start, goal, and waypoint configuration respectively in the free space. The significance of the monotonic constraint in maintaining the fidelity of the viscosity solution can be verified by the following proposition:

Proposition 4.2. *If the solution T violates the monotonic constraints, then T is not the viscosity solution.*

Proof. If T violates the monotonic constraints, it fails to satisfy the property described in Property 4.1, which is necessary for being a solution generated by FMM. The limit of the FMM solutions also adheres to Property 4.1, thereby indicating that T cannot be the limit of FMM solutions. Since the solution derived from FMM converges to the viscosity solution via Theorem 3.2, and considering the uniqueness of the viscosity solution according to Theorem 3.2, it follows that T is not the viscosity solution. □

In this way, our monotonic constraint implicitly forces the network to converge to the viscosity solution. We also observe that the occurrence of local minima in the time fields, depicted in Fig. 2, is associated with waypoints that violate the specified monotonic constraints mentioned above. Therefore, we can exploit this constraint to guide the network in a self-supervised learning process, allowing it to correct and attain an accurate time field.

However, since our objective is to find the optimal path through the Eikonal equation, obtaining a ground-truth waypoint on the optimal path before receiving the correct solution from the Eikonal equation proves impractical. A straightforward strategy involves employing a traditional path planning approach, such as FMM, to find the waypoint and thus guide the learning. However, this comes at a significant cost due to the slow and cumbersome nature of traditional methods.

To address this challenge, we integrate a self-correction mechanism in a self-supervised manner into network learning. During training, we utilize the network to plan a path and designate any segment of the path as our waypoint. Our goal is to prompt the network to recognize that the current generated path violates the monotonic constraint. This is achieved by integrating the monotonic constraint loss into the network:

$$\begin{aligned}\mathcal{L}_m = &\sum \max\left(T(\mathbf{s}, \mathbf{w}) - T(\mathbf{s}, \mathbf{g}), 0\right) + \\ &\sum \max\left(T(\mathbf{w}, \mathbf{g}) - T(\mathbf{s}, \mathbf{g}), 0\right).\end{aligned} \tag{7}$$

While, from a global perspective, the chosen waypoint in each training batch does not represent the waypoint of the eventual optimal path, the network perceives it as optimal during the training step. Meanwhile, the selection of the waypoint evolves with the monotonic constraint loss and will be optimized during each training batch of the network. This self-supervised training approach with the monotonic constraint loss demonstrates improvement in addressing local minima, as illustrated in Fig. 2.

Optimal Constraint. To further refine the motion planning process, we introduce the *optimal constraint*, which penalizes the path for deviating from the optimality criterion of the Eikonal equation's solution:

$$T(\mathbf{s}, \mathbf{g}) \leq T(\mathbf{s}, \mathbf{w}) + T(\mathbf{w}, \mathbf{g}). \tag{8}$$

Similar to the previously discussed monotonic constraint loss, we introduce an optimal constraint loss term as follows:

$$\mathcal{L}_o = \sum \max\left(T\left(\mathbf{s}, \mathbf{g}\right) - \left(T\left(\mathbf{s}, \mathbf{w}\right) + T\left(\mathbf{w}, \mathbf{g}\right)\right), 0\right). \tag{9}$$

Following NTFields, we utilize the isotropic speed loss to govern the training of the time field:

$$\begin{aligned}\mathcal{L}_s = \frac{1}{|C|} \sum_{\mathbf{c_i}, \mathbf{c_k} \in C} \Big(&\left\| 1 - \sqrt{S^*(\mathbf{c_i})/S_\Theta(\mathbf{c_i})} \right\| + \left\| 1 - \sqrt{S^*(\mathbf{c_k})/S_\Theta(\mathbf{c_k})} \right\| + \\ &\left\| 1 - \sqrt{S_\Theta(\mathbf{c_i})/S^*(\mathbf{c_i})} \right\| + \left\| 1 - \sqrt{S_\Theta(\mathbf{c_k})/S^*(\mathbf{c_k})} \right\| \Big),\end{aligned} \tag{10}$$

where $|C|$ denotes the number of sampled configurations. $\mathbf{c_i}$ and $\mathbf{c_k}$ represent an arbitrary pair of the start and goal configurations in the c-space. $S^*(\mathbf{c_i})$ and $S^*(\mathbf{c_k})$ are the GT speed values calculated by our SADF (introduced in Sec. 4.2) according to the speed model in Eq. (3). $S_\Theta(\mathbf{c_i})$ and $S_\Theta(\mathbf{c_k})$ are the predicted speed values derived from the predicted time field according to Eq. (1).

With the guidance of our physical constraints, the total loss function for the training of the time field is defined as:

$$\mathcal{L} = \mathcal{L}_s + \lambda_m \mathcal{L}_m + \lambda_o \mathcal{L}_o, \tag{11}$$

where λ_m and λ_o control the penalty weights for the monotonic constraint and optimal constraint respectively.

Training details. To calculate the speed loss during training, we follow these steps: 1) sample configuration pairs $[\mathbf{c_i}, \mathbf{c_k}]$ in the robot's c-space; 2) derive the GT speed values $[S^*(\mathbf{c_i}), S^*(\mathbf{c_k})]$ calculated by Eq. (3); 3) approximate $T_\Theta(\mathbf{c_i}, \mathbf{c_k})$ via time field regressor; 4) obtain the predicted speed values $[S_\Theta(\mathbf{c_i}), S_\Theta(\mathbf{c_k})]$ according to Eq. (1); Finally, the speed loss Eq. (10) can be derived via the GT speed and predicted speed. For the physical constraint loss, we adopt a different paradigm: 1) sample configuration pairs $[\mathbf{c_i}, \mathbf{c_k}]$ in the feasible c-space C_{free}; 2) leverage the time field regressor to obtain $T(\mathbf{c_i}, \mathbf{c_k})$; 3) utilize the gradient of the time field to conduct motion planning, which determines the next waypoint $\mathbf{c_w}$. If the waypoint is in collision, jump to step 1 to resample a new configuration pair; 4) compute $T(\mathbf{c_i}, \mathbf{c_w})$ and $T(\mathbf{c_w}, \mathbf{c_k})$ through time field regressor. Ultimately, the physical constraint loss can be calculated with $T(\mathbf{c_i}, \mathbf{c_k})$, $T(\mathbf{c_i}, \mathbf{c_w})$ and $T(\mathbf{c_w}, \mathbf{c_k})$.

4.2 Shape-Aware Distance Function

When a shaped robot is navigating in the environment, it becomes insufficient only to obtain the distance field of the environment, as a real-world robot cannot be simplistically treated as a particle but rather as a rigid or even deformable body. Therefore, we propose the Shape-Aware Distance Function (SADF) to address the requirements of real-shaped robots. We first introduce how to derive the SADF for the rigid robots and then extend it to the articulated robots.

Rigid Robot. Following the definition of Signed Distance Function (SDF), the Shape-Aware Distance Field (SADF) is defined as the

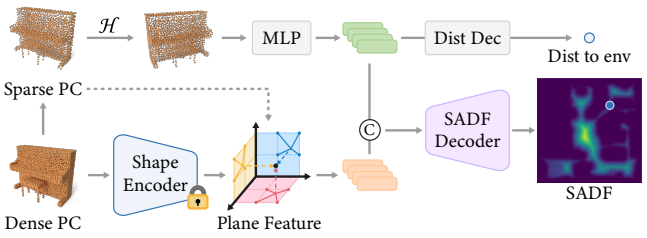


Fig. 4. Training pipeline of SADF. During inference, "Dist Dec" is omitted, with only "SADF Decoder" branch employed as a SADF predictor.

distance between the surface of the robot and the environment:

$$f_{[r,e]}(H) = \min_{\mathbf{x}\in r, \mathbf{y}\in e} d(H(\mathbf{x}), \mathbf{y}), \tag{12}$$

$$\text{with} \quad H(\mathbf{x}) = R\mathbf{x} + \mathbf{t}, \quad R \in \mathbb{RO}(3),\ \mathbf{x}, \mathbf{t} \in \mathbb{R}^3 \tag{13}$$

where r and e denote the surface boundaries of the robot and the environment, $H \in \mathbb{SE}(3)$ denotes the relative transformation from the robot's coordinate system to the environment's coordinate system, and d is the distance function between any two points.

Given a raw point cloud $P = \{p_i \in \mathbb{R}^3\}_{i=1}^N$ comprising N points from the robot's surface and considering the transformations H of the robot, our objective is to learn the SADF between an arbitrary robot and a fixed environment, denoted as $\Phi_e(P, H) = f_{[P,e]}(H)$.

To obtain a precise SADF, it is necessary to sample dense points on the robot. However, directly calculating the distance from every point to the environment and obtaining the minimum for every transformation H, following the definition (Eq. (12)), can be computationally burdensome and inefficient. To tackle the problem, we propose employing a sparse point cloud and local shape feature derived from the dense point cloud to characterize the robot, and use the neural network to approximate the SADF.

Specifically, the sparse point cloud P_s, downsampled from the dense point cloud P_d, first undergoes transformation based on H and is then processed into the distance feature $\mathbf{F}_d$ through a small MLP parameterized by Θ_1. The distance feature, which encapsulates the transformation information between the points and the environment, is then decoded to derive the distance from a point to the environment via a point distance decoder Θ_2. The loss function to govern the MLP and decoder is defined as:

$$\mathcal{L}_d = \frac{1}{|\mathcal{H}||P_s|} \sum_{H\in\mathcal{H}, \mathbf{x}\in P_s} \|\hat{\text{SDF}}_e(H(\mathbf{x}); \Theta_1, \Theta_2) - \text{SDF}_e(H(\mathbf{x}))\|, \tag{14}$$

where $|P_s|$ and $|\mathcal{H}|$ denote the number of sparse point cloud and the number of sampled transformations respectively, $\text{SDF}_e(\cdot)$ represents the GT SDF of the environment, and $\hat{\text{SDF}}_e(\cdot; \Theta_1, \Theta_2)$ indicates the learned SDF function parameterized by Θ_1, Θ_2.

Meanwhile, we utilize the pre-trained encoder from [Chou et al. 2022] to extract the plane features of the robot from densely sampled points P_d. Then, we aggregate the local shape feature $\mathbf{F}_s$ for each sparse point via bilinear interpolation in the plane feature space. At last, we concatenate $\mathbf{F}_d$ and $\mathbf{F}_s$ of each sparse point and feed them to the SADF decoder Θ_3 to decode the ultimate SADF in the environment, as shown in Fig. 4. The loss function for the SADF training is as follows:

$$\mathcal{L}_{SADF} = \left(\frac{1}{|\mathcal{H}|} \sum_{H\in\mathcal{H}} \|\hat{\Phi}_e(P_d, H; \Theta_1, \Theta_3, \Theta_4) - \Phi_e(P_d, H)\| \right) + \lambda_d \mathcal{L}_d, \tag{15}$$

where Θ_4 denotes the frozen parameters of the shape encoder, $\hat{\Phi}_e(\cdot, \cdot; \Theta_1, \Theta_3, \Theta_4)$ represents the learned SADF parameterized by $\Theta_1, \Theta_3, \Theta_4$, and λ_d determines the weights of $\mathcal{L}_d$. Note that the encoded shape feature $\mathbf{F}_s$, derived from the dense point cloud, only needs to be computed once for a specific robot. This will significantly reduce the overall computational cost.

Articulated Robot. To model the articulated robots, we draw inspiration from [Li et al. 2024] and represent them by the union of each part's SADF. Consider the articulated robot with m links, characterized by shapes $\boldsymbol{l} = \{l_0, l_1, ..., l_{m-1}\}$, the SADF for the articulated robot can be represented by the union of each link's SADF, which is defined as:

$$f_{[r,e]}(H_b) = \min\{f_{[l_i,e]}(H_i^b H_b)\},\ i = \{0, 1, ..., m-1\}, \tag{16}$$

where H_b denotes the transformation from the base frame of the robot to the world frame, and H_i^b represents the transformation from the i-th link's frame to the base frame.

4.3 Adaptive Motion Planning

Due to the unpredictability of the network and the fact that our physical constraints serve as a necessary condition for the viscosity solution of the Eikonal equation but not a sufficient one, it remains challenging to avoid all of the local minima entirely. Therefore, we introduce an adaptive motion planning approach illustrated in Fig. 5 to bolster the robustness of the planning procedure, leveraging the fast inference speed of our SADF for collision checking.

For the collision-free case, the ultimate path solution is obtained by performing gradient descent bidirectionally, traversing from the start to the goal and vice versa, which follows the trivial planning strategy of NTFields:

$$\mathbf{s}_{i+1} = \mathbf{s}_i + \alpha S^2(\mathbf{s}_i) \nabla_{\mathbf{s}_i} T(\mathbf{s}_i, \mathbf{g}_i), \tag{17}$$

$$\mathbf{g}_{i+1} = \mathbf{g}_i + \alpha S^2(\mathbf{g}_i) \nabla_{\mathbf{g}_i} T(\mathbf{s}_i, \mathbf{g}_i), \tag{18}$$

where α is a step size hyperparameter, and $S^2(\mathbf{s}_i)$ or $S^2(\mathbf{g}_i)$ is a regulation coefficient to address safety issues arising from low speeds near obstacles, which can cause large gradients due to the inverse speed-gradient relationship in Eq. (1).

If a collision is detected using our SADF, we employ adaptive motion planning. As shown in Fig. 5, we proceed to find a collision-free waypoint $\mathbf{u}$ adjacent to the collision point, thereby forming the longest collision-free path from $\mathbf{u}$ to either the start or goal configuration. Then, we randomly select a point $\mathbf{v}$ along such a collision-free path for replanning. Random sampling employed here is to increase the probability that subsequently sampled points are also collision-free. Similar to the initialization of the sampling-based method, we randomly sample an appropriate number of points within a hypersphere in the configuration space, with $\mathbf{v}$ as the center and r as the radius. These points serve as candidate points to escape local minima. Subsequently, the candidate point $\mathbf{c}$ can be utilized as a waypoint on the path to devise a new collision-free trajectory which is formed as $\mathbf{s}-\mathbf{c}-\mathbf{g}$. To retain the optimality, we calculate the sum of $T(\mathbf{s}, \mathbf{c})$ and $T(\mathbf{c}, \mathbf{g})$ to obtain the total time of the replanned

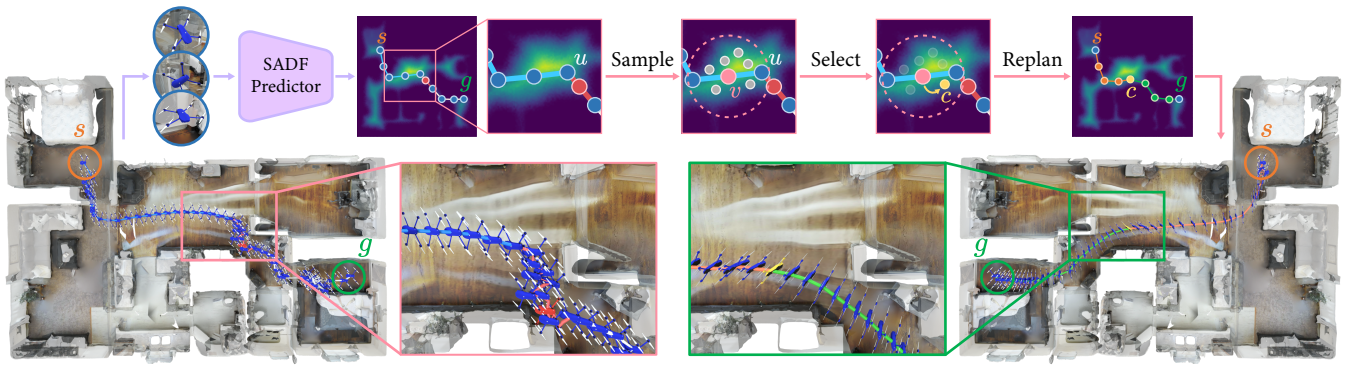


Fig. 5. Adaptive motion planning. We locate a collision-free waypoint **u** adjacent to the collision point and randomly sample points around **u**. These sampled points serve as candidates to escape local minima. The candidate point **c** is selected based on traversal times calculated from **s** to **c** and from **c** to **g**.

path from the learned time field for each candidate point c. We then choose the candidate point with the shortest time to form the final trajectory. Moreover, we adaptively enlarge the search radius r if a collision-free path is not found within the candidate points. This adaptive strategy aids in exploring a larger configuration space to find feasible paths when necessary, enhancing the robustness of our PC-Planner. It's worth noting that this strategy is cost-affordable as it is applied only on the collision path (which is rare in practice thanks to our physics-constrained learning).

5 EXPERIMENTS

In this section, we analyze and validate our method with different robots and environments. We compare our methods with the baselines NTFields [Ni and Qureshi 2023a], P-NTFields [Ni and Qureshi 2023b], FMM [Sethian 1996], RRT* [Kingston et al. 2018], RRT-Connect [Kuffner and LaValle 2000], and LazyPRM* [Bohlin and Kavraki 2000]. For RRT*, RRT-Connect, and LazyPRM* which are probabilistically complete and can theoretically find a solution given infinite time, we impose a practical time limit (10 seconds for rigid robots and 5 seconds for manipulators), since real-world scenarios often require time-constrained solutions. Our evaluation metrics include **path length**, **planning time**, **success rate (SR)**, and **challenging success rate (CSR)**. For additional information on the experimental settings, including metrics description, baseline details, and experimental specifics, please refer to our supplementary material.

5.1 Motion Planning in 3D Environments for Rigid Robots

We perform a comparative analysis of our method against the baselines in two complex 3D Gibson environments (Arona and Eastville) [Xia et al. 2018] in $\mathbb{SE}(2)$ and $\mathbb{SE}(3)$ space with rigid robots. In those two Gibson environments, we employ different robots. In one scenario, we deploy a mobile robot, a bear, and a bird for navigation within the environment. In the other scenario, we showcase the use of a piano, a toy car, and a drone to plan in the environment. Specifically, the mobile robot, bear, piano, and toy car navigate in $\mathbb{SE}(2)$ space with 3 Degrees of Freedom (DoFs) while the bird and drone plan in $\mathbb{SE}(3)$ space with 6 DoFs. These planning examples are depicted in Fig. 9. Additional experiments on 2D environments in $\mathbb{SE}(2)$ space are demonstrated in the supplementary material.

Table 1. Comparison of motion planning in 3D for rigid robots. The optimal results are highlighted with **first**, second.

Methods	Metrics	Arona			Eastville		
		Bear	Mobile Root	Bird($\mathbb{SE}(3)$)	Piano	Toy Car	Drone($\mathbb{SE}(3)$)
RRT*	Length	0.26	0.26	0.23	0.19	0.23	0.23
	Time(ms)	10189.2	10203.1	10121.2	10215.2	10226.5	10196.6
	SR(%)	83.7	81.8	89.6	80.4	80.7	83.7
	CSR(%)	84.4	37.3	29.3	60.5	63.0	29.2
LazyPRM*	Length	0.25	**0.24**	**0.21**	0.18	**0.20**	**0.19**
	Time(ms)	10158.9	10152.6	10121.9	10152.3	10252.3	10130.5
	SR(%)	84.6	87.4	92.9	81.6	87.3	90.3
	CSR(%)	85.6	64.8	51.7	68.4	75.2	57.1
RRT-Connect	Length	0.70	0.72	1.1	0.66	0.70	1.03
	Time(ms)	1543.4	1246.2	959.3	2008.1	1568.7	1259.1
	SR(%)	90.0	91.5	93.6	85.5	89.4	89.7
	CSR(%)	94.6	77.9	60.5	72.8	81.0	68.6
NTFields	Length	0.25	0.25	**0.21**	0.18	**0.20**	**0.19**
	Time(ms)	6.1	6.1	4.5	5.7	6.7	2.5
	SR(%)	85.4	77.3	88.5	86.1	94.3	86.4
	CSR(%)	56.3	45.6	22.5	72.4	88.7	42.0
P-NTFields	Length	**0.24**	**0.24**	**0.21**	0.2	0.21	**0.19**
	Time(ms)	2.8	2.7	**1.4**	6.5	4.2	**1.7**
	SR(%)	94.9	96.7	86.3	80.4	88.9	85.4
	CSR(%)	85.0	**91.1**	7.5	61.5	79.8	35.4
Ours w/o adapt. planning	Length	**0.24**	**0.24**	**0.21**	**0.17**	**0.20**	**0.19**
	Time(ms)	**1.9**	**2.3**	4.9	**1.9**	**1.7**	4.6
	SR(%)	99.7	96.0	**95.8**	91.3	96.2	92.7
	CSR(%)	99.1	88.8	**72.1**	83.0	92.5	67.7
Ours	Length	**0.24**	**0.24**	**0.21**	**0.17**	**0.20**	**0.19**
	Time(ms)	2.8	26.0	4.6	49.0	25.1	38.5
	SR(%)	**99.8**	**96.8**	**95.8**	**92.6**	**96.9**	**93.2**
	CSR(%)	**99.4**	**91.1**	**72.1**	**85.6**	**93.5**	**70.0**

The quantitative results are listed in Tab. 1. In the Arona environment, our proposed method demonstrates the best SR, CSR, and path length with a competitive computation time. It is worth mentioning that without adaptive planning, our method achieves minimal computation time and best or second-best SR, CSR and path length in most tasks. Moreover, our method is more efficient compared to traditional methods, ranging from at least 40 times to as much as 200 times faster than traditional methods, indicating an affordable time cost of our adaptive strategy. This highlights the effectiveness and superiority of our proposed method.

Table 2. Comparison of motion planning for manipulators. The optimal results are highlighted with **first**, second.

Methods	Metrics	Manual Craft custom-arm		Single-cabinet UR5 arm	Dual-cabinet UR5 arm
		4-DoF	6-DoF	6-DoF	6-DoF
RRT*	Length	0.28	0.23	0.35	0.25
	Time(ms)	5120	5120	5140	5130
	SR(%)	90.1	90.5	88.1	85.6
	CSR(%)	55.5	48.4	40.5	41.0
LazyPRM*	Length	**0.25**	0.21	**0.28**	**0.21**
	Time(ms)	5080	5060	5070	5080
	SR(%)	98.2	97.9	98.8	97.5
	CSR(%)	86.4	87.9	85.1	85.6
RRT-Connect	Length	0.83	1.04	1.04	1.03
	Time(ms)	370	460	360	810
	SR(%)	97.4	97.4	98.9	96.7
	CSR(%)	93.6	93.6	89.2	**89.0**
NTFields	Length	**0.25**	**0.20**	**0.28**	**0.21**
	Time(ms)	4.9	1.5	1.7	1.9
	SR(%)	91.2	96.0	97.1	93.1
	CSR(%)	60.0	74.5	60.8	60.7
P-NTFields	Length	**0.25**	**0.20**	**0.28**	**0.21**
	Time(ms)	**1.0**	**1.1**	**1.2**	**1.3**
	SR(%)	89.7	87.9	95.5	84.9
	CSR(%)	53.2	24.2	40.5	14.5
Ours w/o adapt. planning	Length	**0.25**	**0.20**	**0.28**	**0.21**
	Time(ms)	1.2	**1.1**	**1.2**	2.9
	SR(%)	99.0	99.4	**99.6**	97.8
	CSR(%)	95.5	96.2	**96.0**	87.3
Ours	Length	**0.25**	**0.20**	**0.28**	**0.21**
	Time(ms)	2.5	2.0	18.4	58.3
	SR(%)	**99.4**	**99.8**	**99.6**	**97.9**
	CSR(%)	**97.3**	**98.7**	**96.0**	87.9

Table 3. Performance comparison of PC-Planner, specifically under conditions without adaptive motion planning, using our SADF against BVH-distance-query with different sampled surface points. PC indicates the physical constraints. The optimal results are highlighted with **first**, second.

Methods	Points	Metric	Aronna		EastVille	
			Bear	Mobile Robot	Piano	Car
NTFields	1024	SR(%)	85.4	77.3	86.1	94.3
PC + BVH	1024		**99.9**	**97.0**	**96.1**	**96.6**
PC + BVH	32		95.9	93.7	89.5	95.9
PC + SADF	32		99.7	96.0	91.3	96.2

Additionally, we validate our methods on the real-world **TurtleBot4** robot in the complex meeting room and hallway environments with clustered obstacles, as shown in Fig. 6.

5.2 Motion Planning for Manipulators

This section showcases the performance of our method on both custom-built and standard UR5 manipulators across various scenarios. These scenarios include a simple manually crafted environment, a single-cabinet operating environment, and a dual-cabinet opposing operating environment, each with increasing levels of difficulty.

As shown in Tab. 2, we conducted 4-DoF and 6-DoF experiments with a custom-built arm in the manually crafted environment. Our method outperformed in all metrics except for the time. However, the time difference compared to the best performance is almost negligible. The experiments with the 6-DoF UR5 manipulator are presented in the last two columns of Tab. 2. In the single-cabinet environment, our method continues to top all other metrics, even without adaptive planning. In the dual-cabinet opposing environment, ours maintains the highest SR and ranks second in CSR under challenging cases. Planning examples are depicted in Figs. 7 and 8.

Table 4. Comparison of collision-checking time among FCL, BVH-distance-query, and our SADF with different sampled points on various numbers (10, ..., 10000) of relative transformation H between robot and environment.

Transformation Numbers	Time (ms)			
	10	100	1000	10000
FCL	1.0	4.9	40.4	392.6
BVH + 1024 points	5.4	9.1	39.0	382.1
BVH + 32 points	5.8	5.7	6.7	31.2
SADF + 32 points	**0.7**	**0.8**	**1.4**	**13.4**

Table 5. Comparison of training time for robots with different DoFs in a new environment.

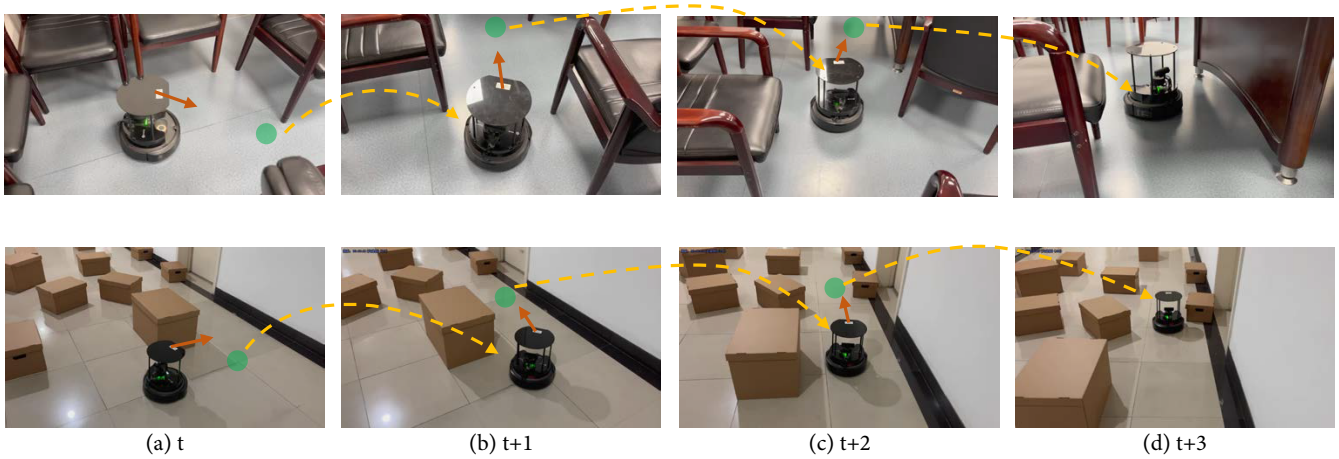
Methods	Time (h)	
	3-DoF robot	6-DoF robot
NTFields	1.0	9.1
P-NTFields	3.7	31.6
Ours	1.3	14.1

5.3 Ablation Studies

The effectiveness of the proposed adaptive planning strategy can be validated through Tabs. 1 and 2. As shown in Tab. 3, all pipelines utilizing physical constraints (with or without SADF) achieve a higher SR than the baseline NTFields, which verifies the effectiveness of our physical constraints.

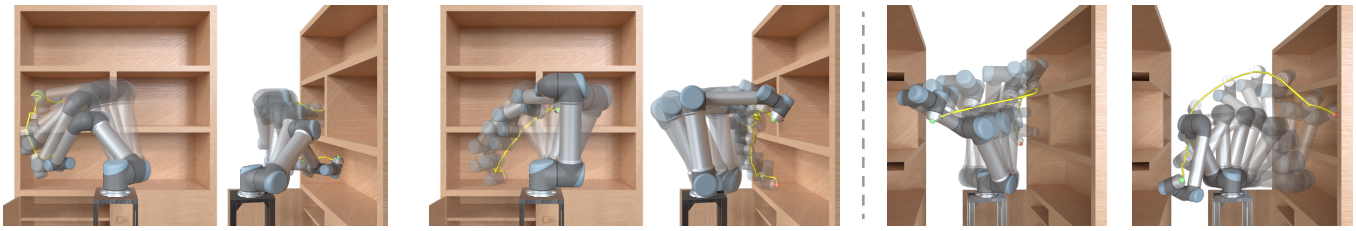
We also investigate the influence of the SADF on our robot motion planning and present the quantitative results. For our SADF, the robot's sparse and dense point clouds consist of 32 and 1024 points, respectively. The Ground Truth (GT) to train our SADF is generated using BVH-distance-query [Karras 2012] with 1024 sampled points of the robots (detailed in supplementary materials). Consequently, we evaluate our SADF against BVH-distance-query using 32 and 1024 sampled surface points within the time fields + physical constraints pipeline. From Tab. 3, it is evident that utilizing our SADF results in a higher SR compared to using BVH-distance-query with only 32 surface points, and it is still competitive with using GT distance field generated with 1024 points.

At last, we compare the collision-checking time cost of our SADF with BVH-distance-query and FCL (a common collision-checking library) [Pan et al. 2012] to demonstrate the lightweight nature of our SADF as illustrated in Tab. 4. The results indicate that both FCL and BVH-distance-query are nearly 30 times slower than our learned SADF, particularly as the number of query points increases, which can be the bottleneck in both the training procedure and the real-time planning scenarios.



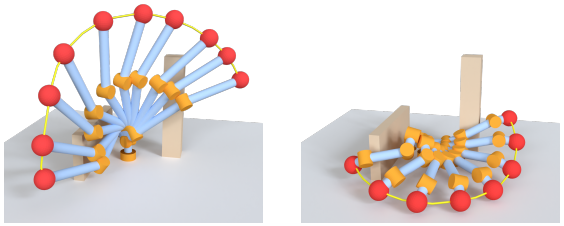
(a) t (b) t+1 (c) t+2 (d) t+3

Fig. 6. A real TurtleBot4 robot navigates in the real-world meeting room with clustered chairs and the hallway with cluster boxes.



UR5 planning in single-cabinet UR5 planning in dual-cabinet

Fig. 7. The UR5 manipulator executes motion planning in the single-cabinet and dual-cabinet environment.



4-DoF robot planning 6-DoF robot planning

Fig. 8. 4-DoF and 6-DoF custom-built arms do motion planning in the manually crafted environment.

6 LIMITATIONS AND FUTURE WORK

We report the training time for NTFields, P-NTFields, and our method in a new environment, as shown in Tab. 5. Our method shows comparable training times to NTFields and a significant reduction compared to P-NTFields. Once trained, our method is able to generate paths in static scenes notably faster than training-free methods like RRT* and RRT-Connect. However, in dynamic environments, training-free methods, which provide solutions within seconds, become valuable alternatives. A promising direction for future work is to further investigate the generalization capabilities of physics-informed methods in new environments.

Additionally, although the proposed neural SADF is agnostic to robot shapes, it is environment-specific, i.e., we need to train the SADF for a new environment. It is also interesting to further make it environment-agnostic.

7 CONCLUSION

This paper introduces a physics-constrained self-supervised learning framework for efficient and robust neural motion planning navigation in complex environments, named PC-Planner. To this end, we propose two physical constraints, namely monotonic and optimal constraints, to mitigate issues related to local minima in the Eikonal equation and introduce a novel shape-aware distance field to expedite the application of the physical constraints. Additionally, we develop an adaptive motion planning strategy to enhance the robustness of our proposed PC-Planner. Experiments with diverse robots in various scenarios demonstrate the efficacy of our method.

ACKNOWLEDGMENTS

We thank all the reviewers for their constructive comments and extend our gratitude to Xiao Liang for his help. We would also like to acknowledge the support of NSFC (No. 62102356, No. 62322207), Information Technology Center, and State Key Lab of CAD&CG, Zhejiang University.

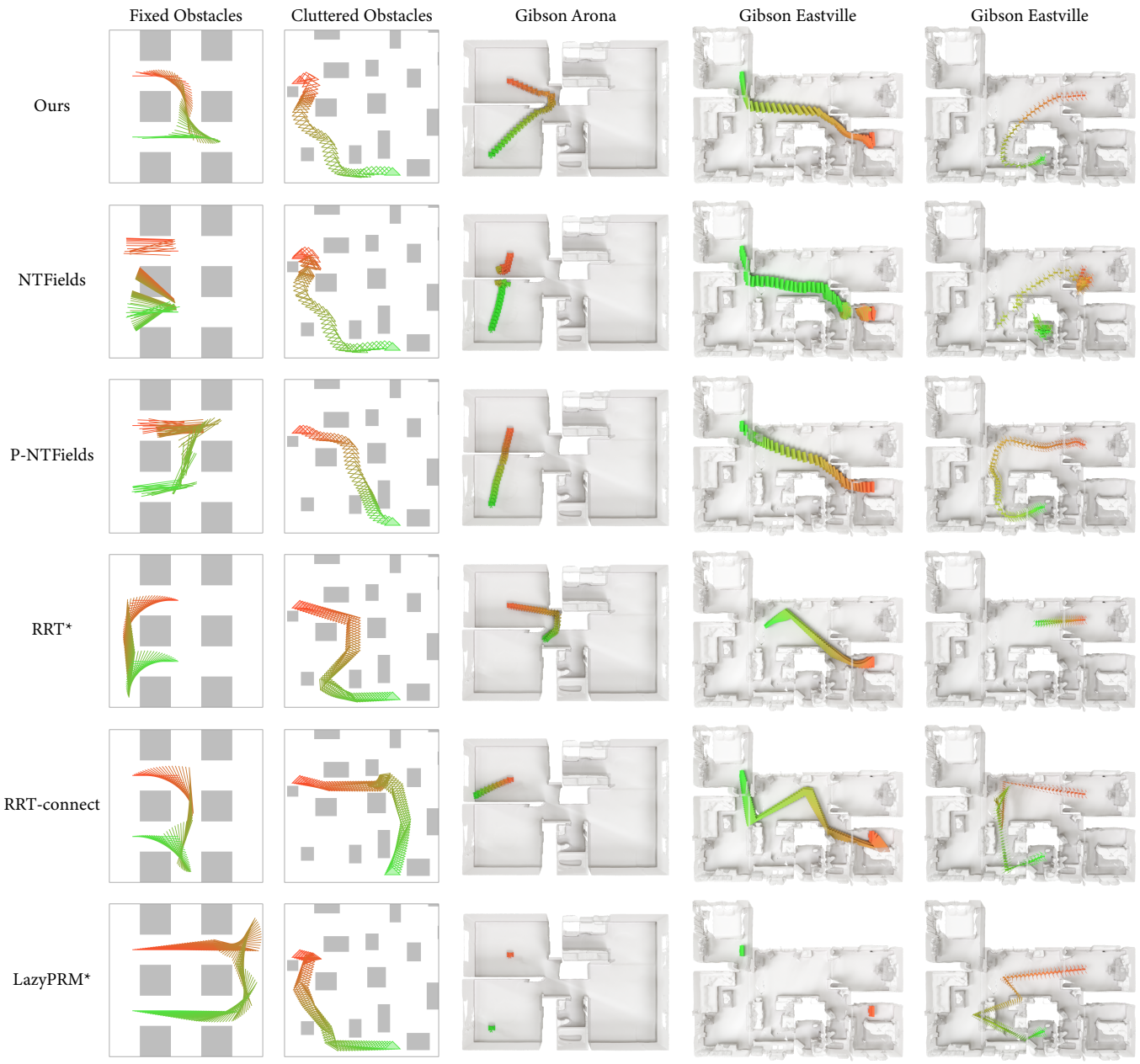


Fig. 9. The comparison results on multiple environments with different robots of various shapes.

REFERENCES

Guy Barles and Panagiotis E Souganidis. 1991. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic analysis* 4, 3 (1991), 271–283.

Robert Bohlin and Lydia E Kavraki. 2000. Path planning using lazy PRM. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, Vol. 1. IEEE, 521–528.

Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. 2020. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*. Springer, 608–625.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015).

Devendra Singh Chaplot, Deepak Pathak, and Jitendra Malik. 2021. Differentiable spatial planning using transformers. In *International Conference on Machine Learning*. PMLR, 1484–1495.

Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Randall Hill. 2021. Equivariant point network for 3d point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14514–14523.

David L Chopp. 2001. Some improvements of the fast marching method. *SIAM Journal on Scientific Computing* 23, 1 (2001), 230–244.

Gene Chou, Ilya Chugunov, and Felix Heide. 2022. GenSDF: Two-Stage Learning of Generalizable Signed Distance Functions. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.),

Vol. 35. Curran Associates, Inc., 24905–24919. https://proceedings.neurips.cc/paper_files/paper/2022/file/9dfb5bc27e2d046199b38739e4ce64bd-Paper-Conference.pdf

Michael G Crandall and Pierre-Louis Lions. 1983. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American mathematical society* 277, 1 (1983), 1–42.

Clemens Eppner, Arsalan Mousavian, and Dieter Fox. 2020. ACRONYM: A Large-Scale Grasp Dataset Based on Simulation. In *2021 IEEE Int. Conf. on Robotics and Automation, ICRA*.

Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. 2015. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 3067–3074.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Jinwook Huh, Volkan Isler, and Daniel D Lee. 2021. Cost-to-go function generating networks for high dimensional motion planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 8480–8486.

Brian Ichter, James Harrison, and Marco Pavone. 2018. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7087–7094.

Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. 2020. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6001–6010.

Jacob J. Johnson, Ahmed H. Qureshi, and Michael C. Yip. 2023. Learning Sampling Dictionaries for Efficient and Generalizable Robot Motion Planning With Transformers. *IEEE Robotics and Automation Letters* 8, 12 (2023), 7946–7953. https://doi.org/10.1109/LRA.2023.3322087

Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. 2011. STOMP: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*. IEEE, 4569–4574.

Sertac Karaman and Emilio Frazzoli. 2011. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research* 30, 7 (2011), 846–894.

Tero Karras. 2012. Maximizing Parallelism in the Construction of BVHs, Octrees, and K-d Trees. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*. Eurographics Association, 33–37. https://doi.org/10.2312/EGGH/HPG12/033-037

Zachary Kingston, Mark Moll, and Lydia E Kavraki. 2018. Sampling-based methods for motion planning with constraints. *Annual review of control, robotics, and autonomous systems* 1 (2018), 159–185.

James J Kuffner and Steven M LaValle. 2000. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, Vol. 2. IEEE, 995–1001.

Mikhail Kurenkov, Andrei Potapov, Alena Savinykh, Evgeny Yudin, Evgeny Kruzhkov, Pavel Karpyshev, and Dzmitry Tsetserukou. 2022. NFOMP: Neural Field for Optimal Motion Planner of Differential Drive Robots With Nonholonomic Constraints. *IEEE Robotics and Automation Letters* 7, 4 (2022), 10991–10998.

Steven M LaValle and James J Kuffner Jr. 2001. Randomized kinodynamic planning. *The international journal of robotics research* 20, 5 (2001), 378–400.

Xueting Li, Shalini De Mello, Xiaolong Wang, Ming-Hsuan Yang, Jan Kautz, and Sifei Liu. 2021. Learning continuous environment fields via implicit functions. *arXiv preprint arXiv:2111.13997* (2021).

Yiming Li, Yan Zhang, Amirreza Razmjoo, and Sylvain Calinon. 2024. Representing Robot Geometry as Distance Fields: Applications to Whole-body Manipulation. In *Proc. IEEE ICRA*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=Bkg6RiCqY7

Mustafa Mukadam, Xinyan Yan, and Byron Boots. 2016. Gaussian process motion planning. In *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 9–15.

Ruiqi Ni and Ahmed H Qureshi. 2023a. NTFields: Neural Time Fields for Physics-Informed Robot Motion Planning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ApF0dmi1_9K

Ruiqi Ni and Ahmed H Qureshi. 2023b. Progressive Learning for Physics-informed Neural Motion Planning. *arXiv preprint arXiv:2306.00616* (2023).

Ruiqi Ni and Ahmed H Qureshi. 2024. Physics-informed Neural Motion Planning on Constraint Manifolds. *arXiv preprint arXiv:2403.05765* (2024).

Amine Ouasfi and Adnane Boukhayma. 2022. Few 'zero level set'-shot learning of shape signed distance functions in feature space. In *European Conference on Computer Vision*. Springer, 561–578.

Jia Pan, Sachin Chitta, and Dinesh Manocha. 2012. FCL: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*. IEEE, 3859–3866.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 165–174.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 234–241.

James A Sethian. 1996. A fast marching level set method for monotonically advancing fronts. *proceedings of the National Academy of Sciences* 93, 4 (1996), 1591–1595.

J. A. Sethian. 1999. Fast Marching Methods. *SIAM Rev.* 41, 2 (1999), 199–235. https://doi.org/10.1137/S0036144598347059

Eran Treister and Eldad Haber. 2016. A fast marching algorithm for the factored eikonal equation. *Journal of Computational physics* 324 (2016), 210–225.

Maria Cristina Tugurlan. 2008. *Fast marching methods-parallel implementation and analysis*. Louisiana State University and Agricultural & Mechanical College.

Murad Tukan, Alaa Maalouf, Dan Feldman, and Roi Poranne. 2022. Obstacle aware sampling for path planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 13676–13683.

Aleš Vysockỳ, Hisaka Wada, Jun Kinugawa, and Kazuhiro Kosuge. 2019. Motion planning analysis according to ISO/TS 15066 in human–robot collaboration environment. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 151–156.

Jiankun Wang, Wenzheng Chi, Chenming Li, Chaoqun Wang, and Max Q.-H. Meng. 2020. Neural RRT*: Learning-Based Optimal Path Planning. *IEEE Transactions on Automation Science and Engineering* 17, 4 (2020), 1748–1758. https://doi.org/10.1109/TASE.2020.2976560

Malcolm C. A. White, Hongjian Fang, Nori Nakata, and Yehuda Ben-Zion. 2020. PyKonal: A Python Package for Solving the Eikonal Equation in Spherical and Cartesian Coordinates Using the Fast Marching Method. *Seismological Research Letters* 91, 4 (06 2020), 2378–2389.

Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. 2018. Gibson Env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE.

Yajue Yang, Jia Pan, and Weiwei Wan. 2019. Survey of optimal motion planning. *IET Cyber-systems and Robotics* 1, 1 (2019), 13–19.

Minghan Zhu, Maani Ghaffari, William A Clark, and Huei Peng. 2023. E2PN: Efficient SE (3)-equivariant point network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1223–1232.

PC-Planner: Physics-Constrained Self-Supervised Learning for Robust Neural Motion Planning with Shape-Aware Distance Function Supplementary Material

A FACTORIZED EIKONAL EQUATION

To make it applicable for motion planning tasks, we follow the factorization in NTFields [Ni and Qureshi 2023a] for $T(\mathbf{s}, \mathbf{g})$:

$$T(\mathbf{s}, \mathbf{g}) = \frac{\|\mathbf{s} - \mathbf{g}\|}{\tau(\mathbf{s}, \mathbf{g})}, \tag{19}$$

where $\tau(\mathbf{s}, \mathbf{g})$ is a factorized time field. The advantage here is that τ can effectively adjust the $T(\mathbf{s}, \mathbf{g}) \in [0, \infty]$ within a constrained range (specifically from 0 to 1) and avoid the singularity issue.

We then employ the MLP to model the underlying physics of τ. The neural network, parameterized by Θ, takes the robot's start and goal configuration $(\mathbf{s}, \mathbf{g})$ as input and outputs the factorized time field τ_Θ:

$$\tau_\Theta = \text{MLP}(\mathbf{s}, \mathbf{g}; \Theta). \tag{20}$$

Subsequently, we can obtain the predicted time field T_Θ through Eq. (20).

B IMPLEMENTATION DETAILS

B.1 Physics-Constrained Time Field

Training strategy. The speed loss is applied throughout the entire training process, while the physical constraint loss is introduced after a certain number of epochs (specifically, 50 epochs). This delay is necessary because the time fields do not perform well in the initial training stages, and the waypoints derived at this stage will mislead the network's training. It is important to note that our monotonic constraint requires a relatively optimal waypoint to guide training effectively, and a poor-quality waypoint will disturb training of the time field.

Regressor Architecture. The network architecture of the regressor for the time field follows NTFields. It contains the c-space encoder, a non-linear symmetric operator, and the time field generator. The c-space encoder, denoted as $g(\cdot)$, is comprised of fully connected (FC) layers with ELU activation and several ResNet-style[He et al. 2016] MLP with ELU, which takes the robot's configuration $\mathbf{c}$ as input and generate the embedding $g(\mathbf{c})$. Given the start and goal configuration $\mathbf{s}$ and $\mathbf{g}$, the non-linear symmetric operator is represented as $[\max(f(\mathbf{s}), f(\mathbf{g})), \min(f(\mathbf{s}), f(\mathbf{g}))]$ where $[\cdot]$ denotes a concatenation operator. This operator ensures the output of the time field is symmetric with respect to the start and goal configuration. The time field generator takes the concatenated embedding as input and outputs the time field value. It is composed of several FC + ELU layers and ResNet MLP + ELU layers.

Hyperparameters. We use AdamW [Loshchilov and Hutter 2019] optimizer with $2 \times 10e^{-4}$ learning rate and 0.1 weight decay. During training, λ_m and λ_o are set to 0.08 and 0.001 respectively.

B.2 Shape-Aware Distance Field

Training Setup. We create the training dataset from Acronym [Eppner et al. 2020] following the procedure of GenSDF [Chou et al. 2022]. Acronym is a subset of ShapeNet [Chang et al. 2015] dataset which consists of 8872 watertight synthetic 3D models of 262 categories. We use 147 models as our training dataset to learn our SADF. For each 3D object, we sample 1024 points on the surface as a dense input, and then downsample to 32 points to create a sparse input. For a given environment, we sample 10^6 configurations for each object that we viewed as our robot, and then calculate the shape-aware distance by attaining the minimum of the queried distances from the 1024 sampled surface points to the environment with BVH-distance-query [Karras 2012]. This distance obtained from the dense point cloud through BVH-distance-query is regarded as the GT for SADF training.

Training Details. We fix the parameters of the pretrained shape encoder in the training process. For each epoch, we randomly select one object from the training dataset and use the shape encoder to derive the shape feature which will be only calculated once in this epoch. Then we derive the $\mathcal{L}_d$ and $\mathcal{L}_{SADF}$ to jointly optimize the network parameters.

Network Architecture. We use the pretrained shape decoder from GenSDF [Chou et al. 2022] which chooses 256 latent size and 64 hidden dimensions. The shape encoder utilizes the tri-plane feature to represent the object's geometry and uses the parallel Unet [Ronneberger et al. 2015] to aggregate shape information. For the shape decoder and distance decoder, we employ the fully connected (FC) layers with ELU activation and several ResNet-style[He et al. 2016] MLP with ELU.

Hyperparameters. We use AdamW [Loshchilov and Hutter 2019] optimizer with $2 \times 10e^{-4}$ learning rate and 0.1 weight decay. During training, λ_d is set to 1.

C MORE EXPERIMENT DETAILS

C.1 Experimental Setup

Evaluation Metrics. Our evaluation metrics include **path length**, **planning time**, **success rate (SR)** and **challenging success rate (CSR)**. The path length quantifies the sum of configuration distances between configurations of the waypoint in different settings, while the planning time measures the time taken by a planner to seek a valid path solution. The SR represents the percentage of collision-free paths connecting the provided start and goal in the test dataset identified by a given planner. The CSR is built upon the SR which constructs a test dataset that removes the easy case. Here the easy case implies the trajectory can be successfully planned just using simple linear interpolation between the start and goal configuration.

The quantitative results for each set of experiments are averaged from the planning outcomes.

Baselines. We compare our methods with the following baselines.

- NTFields [Ni and Qureshi 2023a]: As described earlier, it directly learns to solve the Eikonal equation without relying on expert training data.
- P-NTFields [Ni and Qureshi 2023b]: A physics-informed method based on NTFields that introduces a progressive learning strategy and incorporates a viscosity term into the Eikonal equation to deal with complex scenarios.
- FMM [Sethian 1996]: A numerical method [Sethian 1996] that discretizes the given C-space and computes the solution to the Eikonal equation for path planning.
- RRT* [Kingston et al. 2018]: A sampling-based method that constructs optimal trees and finds a feasible path connecting the given start and goal configuration.
- RRT-Connect [Kuffner and LaValle 2000]: A bidirectional sampling-based planner that iteratively grows trees from both the start and goal configurations, attempting to connect them by extending the trees towards each other until a path is found.
- Lazy-PRM* [Bohlin and Kavraki 2000]: A multi-query method that combines sampling with graph search, which constructs a graph by sampling the environment and connecting nodes. When given start and goal configurations, it queries the graph to find a path.

Experimental Settings. Following the data preparation procedure outlined in NTFields, we generate 10^6(3,4 DoFs) or 10^7 (6 DoFs) training configurations for NTFields, P-NTFields, and our method. For FMM, which involves the process of discretization, we opt to choose the nearest grid cells associated with our start and goal pairs when seeking solutions. Moreover, we execute RRT*, RRT-Connect, and LazyPRM* on our test set until they discover a path solution with the given start and goal configuration in the specified time limit (10 seconds for rigid robots and 5 seconds for manipulators). For the success rate test, we randomly chose 1000 start and goal pairs that are collision-free as our test set. All the experiments are conducted with 3090 RTX GPU and Intel(R) Xeon(R) Gold 6139M CPU.

C.2 Motion Planning in 2D Environments.

We conduct a benchmark of our proposed method on two 2D environments in $\mathbb{SE}(2)$ space. The first environment consists of six obstacles with fixed sizes, while the second one comprises a cluster of 15 randomly placed obstacles with variable sizes. We compare our method with the aforementioned baselines and demonstrate the planning capability with line and triangle as our robot in 2D environments.

From Tab. 6, we can see that our method surpasses all other approaches in terms of both SR and computation time. Additionally, our method achieves almost the best CSR with only a slight margin behind FMM in the clustered environment. Notably, neural methods, including NTFields, P-NTFields, and our proposed method, demonstrate superior computational efficiency compared to traditional motion planning methods like RRT*, LazyPRM*, RRT-Connect, and FMM. Specifically, our method is more than 400 times faster than FMM, the numerical method to solve the Eikonal equation. In 2D environments, our method outperforms traditional planning methods with a remarkable speed-up of over 100 times, while maintaining a comparable path length to other methods within a small threshold.

Table 6. Comparison on the 2D environments in $\mathbb{SE}(2)$. The optimal results are highlighted with **first**, second.

Methods	Metrics	Fixed Obstacles		Cluttered Obstacles	
		Line	Triangle	Line	Triangle
RRT*	Length	0.24	0.31	0.17	0.30
	Time(ms)	10140.1	10131.2	10191.9	10185.6
	SR(%)	93.4	96.7	82.9	83.7
	CSR(%)	81.8	89.8	59.7	62.9
LazyPRM*	Length	**0.22**	**0.28**	**0.16**	0.27
	Time(ms)	10143.0	10112.1	10145.8	10142.1
	SR(%)	94.7	96.7	88.7	89.7
	CSR(%)	85.4	89.8	73.2	76.3
RRT-Connect	Length	0.68	0.79	0.64	0.79
	Time(ms)	612.1	262.7	1183.5	957.7
	SR(%)	97.7	97.7	93.2	93.2
	CSR(%)	95.3	95.1	88.9	88.3
FMM	Length	0.23	0.31	**0.16**	**0.25**
	Time(ms)	1290.0	1381.2	1332.1	1762.3
	SR(%)	98.4	99.2	99.4	99.2
	CSR(%)	**99.7**	97.8	**100**	**99.3**
NTFields	Length	0.25	0.31	0.17	0.26
	Time(ms)	3.3	2.4	5.3	3.1
	SR(%)	98.3	99.9	82.6	94.0
	CSR(%)	95.3	99.7	60.7	86.2
P-NTFields	Length	0.26	0.31	0.17	0.26
	Time(ms)	2.5	2.3	4.3	**2.0**
	SR(%)	96.9	**100**	94.9	97.2
	CSR(%)	92.0	**100**	84.1	93.6
Ours w/o adapt. planning	Length	0.24	0.31	**0.16**	0.26
	Time(ms)	**1.8**	**2.0**	**1.5**	2.2
	SR(%)	**99.9**	**100**	99.3	98.9
	CSR(%)	**99.7**	**100**	98.3	97.5
Ours	Length	0.24	0.31	**0.16**	0.26
	Time(ms)	**1.8**	**2.0**	2.6	3.7
	SR(%)	**99.9**	**100**	**99.8**	**99.6**
	CSR(%)	**99.7**	**100**	99.5	99.1

C.3 Motion Planning on Constraint Manifolds

We apply our methods to geodesic distance learning, framed as a constrained motion planning (CMP) problem, on a bunny-shaped 2D surface mesh manifold in 3D space. Following [Ni and Qureshi 2024], we define the GT speed model S^* for the constrained motion planning problem as follows:

$$\begin{aligned} D_{\mathcal{M}}(\mathbf{c}) &= \min\left(D_{[r,\mathcal{X}_{mnfld}]}(\mathbf{c}), d_{max}\right), \\ S^*(\mathbf{c}) &= \exp(-\frac{D^2_{\mathcal{M}}(\mathbf{c})}{\beta d^2_{max}}), \end{aligned} \tag{21}$$

where $D_{\mathcal{M}}(\mathbf{c})$ determines the distance from the robot to the manifolds. D is a function that computes the shortest distance between the robot r at configuration $\mathbf{c} \in C$ and manifold $\mathcal{X}_{mnfld}$, d_{max} limits the maximum distance ranges, and $\beta \in \mathbb{R}^+$ is a scaling factor. For further details about CMP with time fields, please refer to [Ni and Qureshi 2024]. As shown in Fig. 10, our method successfully finds an

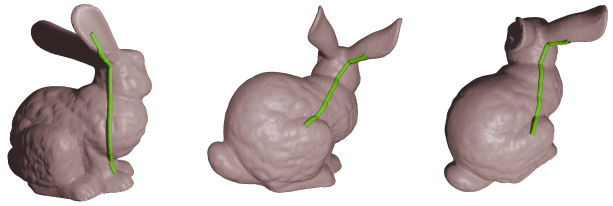


Fig. 10. Constrained motion planning on 2D surface mesh manifold (Bunny) in 3D space, with the planned path highlighted in green.

effective path solution (in green) to determine the geodesic distance.

D LIMITATION

Our physical constraints serve as a necessary condition for ensuring the monotonicity of the time field in the viscosity solution of the Eikonal equation. However, these constraints are not sufficient to guarantee that the network will converge to the viscosity solution. Moreover, although the proposed neural shape-aware distance field (SADF) is independent of robot shapes, it is specific to the environment, requiring training for each new environment. Identifying the sufficient conditions for network convergence to the viscosity solution and developing an environment-agnostic SADF would be an intriguing direction for future research.