Physics-informed Neural Networks for Robot Motion under Constraints

Ruiqi Ni and Ahmed H. Qureshi

Abstract— **Neural Motion Planners (NMPs) have emerged as a promising tool for solving robot tasks in complex environments. However, these methods often require expert data for learning, which limits their application to scenarios where data generation is time-consuming. Recent developments have also led to physics-informed deep learning models capable of representing complex Partial Differential Equations (PDEs). Inspired by these developments, we propose physics-informed neural motion planning methods for robot navigation and manipulation. Our framework represents a wave propagation model generating continuous arrival time to find path solutions informed by a nonlinear first-order PDE called the Eikonal equation. We evaluate our method in various robot navigation and manipulation tasks, including 3D robot navigation in the Gibson dataset, and 6-DOF robot manipulators constrained motion planning. Furthermore, the results show that our method exhibits high success rates and significantly lower computational times than the state-of-the-art methods, including NMPs that require training data from classical planners.**

I. Introduction

Motion Planning (MP) is one of the core components of an autonomous robot system that aims to interact with its surrounding environments physically. MP algorithms find path solutions from the robot's start state to the goal state while respecting all constraints, such as collision avoidance. The quest for fast, scalable MP methods has led from traditional approaches such as RRT* [10], Informed-RRT* [3], and FMT* [8] to NMPs [5]–[7], [9], [11], [16], [17] that exhibit promising performance in high-dimensional spaces. However, a significant bottleneck in state-of-the-art NMPs is their need for expert trajectories from traditional MP methods, limiting their application to high-dimensional scenarios where large-scale data generation is time-consuming.

Recent developments have provided us with ways to have physics-informed deep learning models [12], [18], [20] that can directly solve complex PDEs such as Navier-Stokes, Burgers, and Schrodinger equations. Inspired by these neural PDE solvers and to overcome the expert data needs of NMPs, this paper introduces Neural Time Fields (NTFields) [13]–[15] for robot motion planning in cluttered environments. Our NTFields are generated by a physics-informed neural model driven by a first-order, non-linear PDE called the Eikonal equation, whose solution represents the shortest arrival time from a source to a destination location under a predefined speed model, and leads to the continuous shortest-path between two locations [1], [19]. Our model generates a continuous time field between the robot's given start and goal configurations while respecting collision avoidance and other kinematic constraints, leading to a path solution in sub-seconds.

The authors are with the Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA {ni117, ahqureshi}@purdue.edu

Our method's salient features and contributions are summarized as follows: 1) A novel physics-informed Eikonal equation formulation for robot motion planning under collision-avoidance constraints in high-dimensional spaces. 2) A novel neural architecture design that encapsulates various properties of our PDE, resulting in a scalable, fast NMP method. 3) A novel bidirectional algorithm that quickly finds path solutions by iteratively following the gradient of neural time fields, marching towards each other from the start and goal configurations. 4) Unlike prior, state-of-the-art NMP methods requiring extensive expert motion trajectories, NTFields only require randomly sampled robot start and goal configurations and learn robot motion time fields by directly solving the PDE formulation to find path solutions in sub-second times. Our data generation takes less than 3 minutes, even in complex high-dimensional scenarios, and yet, our method computes paths significantly faster than traditional planners while retaining high success rates. 5) We demonstrate our method in various 3D environments, including the Gibson dataset, and also solve constraint manipulation problems for 6-DOF robot manipulators where traditional grid-based Eikonal planners such as the Fast Marching Method (FMM) [19] struggle due to computational time complexity. 6) Finally, we compare our method against state-of-the-art NMP methods, which require expert motion trajectories, and against existing neural PDE solvers for the Eikonal equation to highlight their limitations in solving complex motion planning problems.

II. Background

This section formally presents the background to robot motion planning problems and their solutions through physics-informed NMPs.

A. Robot Motion Planning

Let the robot's configuration and environment space be denoted as $\mathcal{Q} \subset \mathbb{R}^d$ and $\mathcal{X} \subset \mathbb{R}^m$, where $\{m, d\} \in \mathbb{N}$ represents their dimensionality. The obstacles in the environment, denoted as $\mathcal{X}_{obs} \subset \mathcal{X}$, form a formidable robot configuration space (c-space) defined as $\mathcal{Q}_{obs} \subset \mathcal{Q}$. Finally, the feasible space in the environment and c-space is represented as $\mathcal{X}_{free} = \mathcal{X} \backslash \mathcal{X}_{obs}$ and $\mathcal{Q}_{free} = \mathcal{Q} \backslash \mathcal{Q}_{obs}$, respectively. The objective of robot motion planning algorithms is to find a trajectory $\tau \subset \mathcal{Q}_{free}$ that connects the given robot start $q_s \in Q_{free}$ and goal $q_g \in Q_{free}$ configurations.

The constraint motion planning (CMP) problem extends the standard motion planning problem by incorporating additional kinematic constraints. These constraints induce a thin manifold inside the robot C-space, which is denoted as $\mathcal{M} \subset \mathcal{Q}$. Like the C-space, the manifold also comprises the obstacle, $\mathcal{M}_{obs} \subset \mathcal{Q}_{obs}$, and obstacle-free space, $\mathcal{M}_{free} \subset \mathcal{Q}_{free}$. Finally, the objective of CMP is to find a robot motion path, $\sigma = \{q_s, \cdots, q_g\}$, between the given start, $q_s \in \mathcal{M}_{free}$, and goal, $q_g \in \mathcal{M}_{free}$, such that $\sigma \subset \mathcal{M}_{free}$.

B. Eikonal Equation Formulation

The Eikonal equation is a first-order, nonlinear PDE approximating wave propagation. Its solution is the shortest arrival time from a source to a destination location under a pre-defined speed model, which corresponds to the continuous shortest path problem [1], [19]. Let $S(x)$ and $T(x,y)$ represent the speed model on point x and the corresponding wave arrival time from the given source x to the target y. Then, the Eikonal Equation relates the speed and arrival time as

$$1/S(q_g) = \|\nabla_{q_g} T(q_s, q_g)\| \tag{1}$$

, where $S(y)$ is the speed at target point y, and $\nabla_y T(x,y)$ is a partial derivative of the arrival time model with respect to the target point y. However, the solutions to the Eikonal equation have singularity near the source point when the arrival time is almost zero. Also, the speed inside obstacles, i.e., formidable space $\mathcal{Q}_{obs}$, needs to be zero, making arrival time infinity.

Therefore, we factorize the arrival time T in the following form and introduce the factorized time $\tau(q_s, q_g)$ as:

$$T(q_s, q_g) = \|q_s - q_g\| / \tau(q_s, q_g) \tag{2}$$

In the above equation, the model time field $T(q_s, q_g) = 0$ when $q_s = q_g$. As we require speed in formidable obstacle space to be almost zero, we can explicitly make $\tau(q_s, q_g) \to 0$ for any arbitrary configurations in obstacle-space, i.e., $\{q_s, q_g\} \in \mathcal{Q}_{obs}$. Therefore, $T(q_s, q_g) \in [0, \infty)$ and by our factorization in Equation 2, $\tau(q_s, q_g)$'s value range from 0 to 1.

III. Proposed Method

We model the physics governed by Eq. 1 using a deep neural network. Our neural architecture outputs the factorized time τ for the given robot configurations' representation concerning the given environment. We also introduce novel neural architecture to capture the property of time field w.r.t motion planning. In addition, we introduce a speed model respecting collision and kinematic constraints.

A. Expert Speed Model for Constraints

We define our ground truth speed model, denoted as $S^*(q)$, at any robot configuration $q \in \mathcal{Q}$ as

$$S^*(q) = \frac{s_{const}}{d_{max}} \times \text{clip}(\mathbf{d}_c(\mathbf{p}(q), \mathcal{X}_{obs}), d_{min}, d_{max}), \tag{3}$$

where $\mathbf{p} \subset \mathbb{R}^m$ represents robot surface obtained via forward kinematics for the given configuration $q \in \mathcal{Q}_{free}$, and $\mathbf{d}_c$ computes a distance between robot surface $\mathbf{p}$ and obstacles $\mathcal{X}_{obs}$ in workspace. The d_{min} and d_{max} are minimum and maximum distance thresholds, and s_{const} is a user-defined speed constant. The clip functions bound the distance function with range $[d_{min}, d_{max}]$. We assume constant speed if a robot's distance from obstacles exceeds d_{max}.

Furthermore, we define the speed model for general kinematic and collision avoidance constraints. The objective is to assign a maximum speed value to configuration samples on the collision-free constraint manifold, $\mathcal{M}_{free}$, and a lower speed value to collision and off-manifold samples. Let a function $\mathbf{d}(q, \mathcal{M}, \mathcal{X}_{obs}, \epsilon)$ determine the distance of a given configuration, q, from the collision-free constraint manifold, and ϵ be a predefined safety margin around obstacles. We define this function as:

$$\mathbf{d}(q, \mathcal{X}_{obs}, \epsilon) = \max(\mathbf{d}_{\mathcal{M}}(q), \epsilon - \mathbf{d}_c(\mathbf{p}(q), \mathcal{X}_{obs})) \tag{4}$$

In the above formulation, the distance $\mathbf{d}_{\mathcal{M}}$ measures the distance of a given configuration to the constraint manifold. We compute this distance following the $f(\cdot)$. Moreover, the function $\mathbf{d}_c$ determines the minimum distance of a given robot configuration from the obstacles. The two distance functions, $(\mathbf{d}_{\mathcal{M}}, \mathbf{d}_c)$, and safety margin, ϵ are combined using the max operator. The safety margin allows slow-speed maneuvering around obstacles, which is usually preferred over sharp turns offered by traditional planners. Moreover, in Eq. 4, if the distance of collision surpasses the margin and creates a negative term $\epsilon - \mathbf{d}_c$, the max operator will return the distance to the manifold since it is more important as the configuration is already far from the obstacle. Next, we define our speed model based on distance function $\mathbf{d}$ as follows:

$$S^*(q) = \exp(-\frac{\mathbf{d}(q, \mathcal{X}_{obs}, \epsilon)^2}{\lambda \epsilon^2}), \tag{5}$$

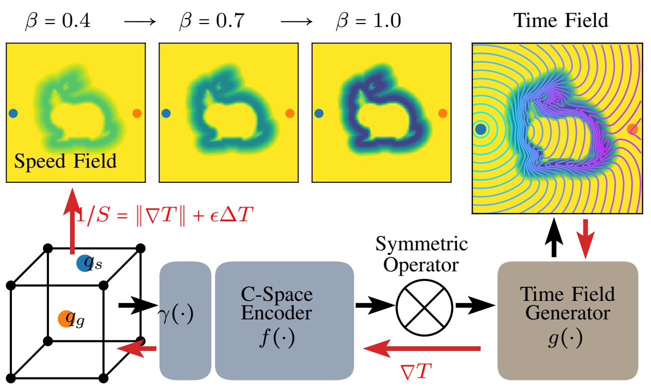
where $\lambda \in \mathbb{R}^+$ is a predefined scaling factor. This speed model uses the negative exponential, which smoothly decays as the distance of the robot configuration from the collision-free manifold increases.

B. Viscosity Eikonal Equation Formulation

The Eikonal equation is ill-posed, i.e., the solution of Eq. 1 around low-speed regions is not unique. To fix these problems, we propose to use a viscosity term that can provide a differentiable and unique approximation of the Eikonal equation's solution. The viscosity term comes from the vanishing viscosity method [2]. It adds the Laplacian $\Delta_{q_g} T(q_s, q_g)$ to the Eikonal equation, leading to a semi-linear elliptic PDE with a unique solution.

$$1/S(q_g) = \|\nabla_{q_g} T(q_s, q_g)\| + \eta \Delta_{q_g} T(q_s, q_g), \tag{6}$$

where $\eta \in \mathbb{R}$ is a scaling coefficient. We use the above formulation in our setting to overcome the challenges induced by the collision and kinematic constraints. Furthermore, we use $\Delta_{q_g} \tau(q_s, q_g)$ instead of $\Delta_{q_g} T(q_s, q_g)$ for computational simplification. Finally, the above equation has a unique solution and aids in training our PINN.



$\beta = 0.4$ $\longrightarrow$ $\beta = 0.7$ $\longrightarrow$ $\beta = 1.0$
Time Field
Speed Field
$1/S = \|\nabla T\| + \epsilon \Delta T$
q_s
q_g
$\gamma(\cdot)$
C-Space Encoder $f(\cdot)$
Symmetric Operator
Time Field Generator $g(\cdot)$
∇T

Fig. 1: The neural architecture comprises the Fourier-based C-space Encoder, symmetric operator, and time-field generator. Three images on the top left show we progressively decrease the speed around a bunny-shaped obstacle to guide the neural network training. The image on the top right shows the final time field from start to goal generated by the trained model.

C. Neural Network Architecture

Our neural architecture can be formalized as follows and Fig. 1:

$$\tau(q_s, q_g) = g(f(\gamma(q_s, Z)) \bigotimes f(\gamma(q_g, Z))) \qquad (7)$$

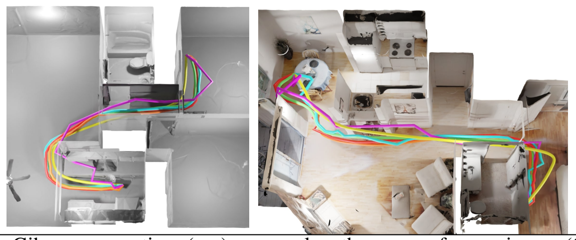
In the above formulation, γ is the Fourier transform-based environment and C-space encoder. It takes as an input the start and goal configurations, (q_s, q_g), and pre-defined environment latent code Z, and outputs the high-frequency Fourier features:

$$\begin{aligned}\gamma(q_s, Z) &= [\cos(2\pi Z^T q_s), \sin(2\pi Z^T q_s)] \\ \gamma(q_g, Z) &= [\cos(2\pi Z^T q_g), \sin(2\pi Z^T q_g)]\end{aligned} \qquad (8)$$

The output features, $(\gamma(q_s, Z), \gamma(q_g, Z))$, are then further embedded by a ResNet-style encoder, f, with skip connections [4]. Next, a symmetry operator, $\otimes$, combines the features using the max and min operators. For instance, some arbitrary inputs, a and b, are combined as $a \otimes b = [\max(a, b), \min(a, b)]$ and $[\cdot]$ is a concatenation operator, which maintains the symmetric property of arrival time, i.e., the arrival time from start to goal and from goal to start must be the same. Finally, the arrival time neural network, g, takes the symmetrically combined features and outputs the $\tau(q_s, q_g)$. This module also leverages the ResNet-style neural network with skip connections. Finally, using the auto-differentiation, we compute the gradient $\nabla_{q_g} \tau(q_s, q_g)$ and the Laplacian $\Delta_{q_g} \tau(q_s, q_g)$ to determine $S(q_s)$ and $S(q_g)$, as described in Eq. 6.

D. Training Procedure

Given the start and goal configuration samples dataset generated on the manifolds and nearby, we train our above-mentioned neural network framework in an end-to-end manner. The NN module takes as an input the environment embedding (Z), the start and goal configurations (q_0, q_T), and outputs the factorized time $\tau(q_0, q_T)$. This factorized time is then used to predict the speed using Equation X. In addition, we also compute the ground truth speed using Eq.



Gibson	time (sec)	length	safe margin	sr(%)
Ours	0.01 ± 0.00	11.68 ± 29.69	0.88 ± 0.16	98.3
IEF3D	0.05 ± 0.00	11.47 ± 27.69	0.87 ± 0.18	92.5
FMM	0.69 ± 0.00	11.21 ± 24.79	0.93 ± 0.13	97.4
RRT*	3.17 ± 0.00	10.36 ± 26.28	0.57 ± 0.29	89.8
LazyPRM*	2.63 ± 25.09	9.94 ± 16.27	0.53 ± 0.35	92.9
RRT-Connect	0.44 ± 0.28	11.95 ± 32.88	0.56 ± 0.34	100

Fig. 2: Comparison in two Gibson environments. The figures show six paths generated by our method (orange), IEF3D (cyan), FMM (green), LazyPRM* (pink), and RRT-Connect (yellow). The statistical results on 2×500 different starts and goals for two Gibson environments.

3 or Eq. 5. Finally, the NN can be trained by minimizing the following isotropic loss between the predicted and ground speed at the given configurations:

$$\begin{aligned}&S^*_{\beta(e)}(q_s)/S(q_s) + S(q_s)/S^*_{\beta(e)}(q_s) + \\ &S^*_{\beta(e)}(q_g)/S(q_g) + S(q_g)/S^*_{\beta(e)}(q_g) - 4\end{aligned} \qquad (9)$$

Furthermore, we use the progressive speed scheduling approach to train our networks and prevent them from converging to incorrect local minima. The scheduling approach gradually scales down the ground truth speed from higher to lower value over the training epoch, e, using the parameter $\beta(e)$, i.e., $S^*_{\beta(e)}(q) = (1 - \beta(e)) + \beta(e) S^*(q)$. This approach overcomes the complex loss landscape of physics-based objective functions and leads to better convergence in low-speed environments such as those with thin manifolds. Additionally, we employ a random batch buffer strategy to train our PINN method. Our findings suggest that selecting a random, smaller data batch for each training epoch is a more efficient and effective approach.

E. Planning Procedure

Once our NN modules are trained, we use the planning pipeline. We begin by computing the factorized arrival time using NN, $\tau(q_s, q_g)$, required to travel from the starting point q_s to the destination point q_g. Next, τ factorizes Eq. 2 and Eq. 1 to compute $T(q_s, q_g)$ and speed fields $S(q_s), S(q_g)$. Finally, the start and goal configurations are bidirectionally and iteratively updated toward each other until the terminal limit is reached, i.e., $\|q_s - q_g\| < r$ to find a path, i.e.,

$$\begin{aligned}q_s &\leftarrow q_s - \alpha S^2(q_s) \nabla_{q_s} T(q_s, q_g) \\ q_g &\leftarrow q_g - \alpha S^2(q_g) \nabla_{q_g} T(q_s, q_g)\end{aligned} \qquad (10)$$

where parameter $\alpha \in \mathbb{R}$ is a predefined step size and $r \in \mathbb{R}$ is predefined the goal region.

IV. Evaluation

In this section, we assess the performance of our method through three sets of experiments. We conduct comparative experiments to evaluate our method against several state-of-the-art MP baselines. These experiments encompass the

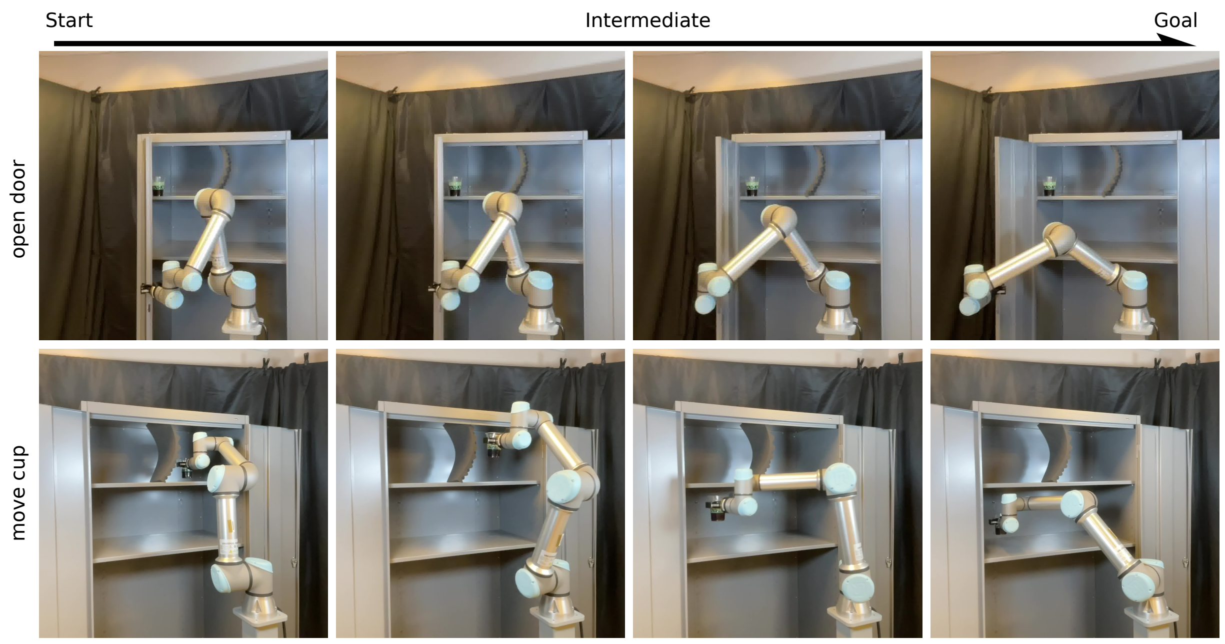


Fig. 3: Two different real-world manipulator cases: the first row shows the door opening task, whereas the second shows the manipulator moving an object from the cabinet's top shelf to the lower shelf by crossing two relatively thin obstacles.

two problem settings: (1) A complex Gibson navigation setting in 3D space (2) A door-opening task with 6-DOF UR5e robot manipulator and an object manipulation task under orientation constraints with 6-DOF UR5e robot in intricate, narrow-passage cabinet environments. Furthermore, we perform all experiments on a computing system with 3090 RTX GPU, Core i7 CPU, and 128GB RAM.

Gibson Navigation:

Our Gibson environments (Fig. 2) demonstrate home-like complex scenarios. We randomly sample 500 start and goal pairs for motion planning in each environment. We compare our method's performance with IEF3D, FMM, RRT*, LazyPRM*, and RRT-Connect. Fig. 2 shows the paths where our method, IEF3D, FMM, LazyPRM*, and RRT-Connect path solutions are illustrated in orange, red, cyan, green, pink, and yellow colors, respectively. We exclude RRT* for the cases presented in the figure as it could not find a valid within 10 seconds limit. Our method, IEF3D, and FMM results show similar smooth paths because of the obstacle safety margin, whereas, RRT-Connect and LazyPRM* have shorter path lengths due to a smaller safety margin. The table in Fig. 2 presents the statistical results. Our results validate that our approach enables physics-informed NMPs to achieve higher performance without needing any demonstration trajectories for learning and outperform prior methods.

Door Opening and Object Manipulation:

These two tasks, defined by distinct manifolds, are solved through a 6-DOF UR5e Manipulator in both simulation and real-world settings. The door-opening task requires a robot to open the door from the current position to the open position. On the other hand, the object manipulation task imposes orientation constraints and requires the robot to maintain the object upright, i.e., without tilting, while moving it from a given start to the goal. We chose a challenging cabinet with narrow passages as our environment for these two tasks, which imposes significant motion planning challenges in terms of collision avoidance and manifold constraints.

Fig. 3 shows our method's executions in real-world experiments. The environment was scanned via the RealSense sensor. The first row shows the snapshots of opening the door of the cabinet. This case took 0.06 seconds. The second row shows snapshots of moving a cup of cola across cabinet shelves: the manipulator moving from the cabinet's top shelf and crossing two relatively thin obstacles to another corner of the cabinet. This case took 0.15 seconds.

V. Conclusions and Future Work

We introduce a physics-informed neural motion planner that requires no expert demonstration data from classical planners and finds path solutions with significantly low computation times and high success rates in complex environments. Additionally, we formulate a physics-driven objective function and reflect it in our architecture design to directly parameterize the Eikonal equation and generate time fields for different scenarios, including a 6-DOF manipulator space, under collision-avoidance and other kinematic constraints. Our future work revolves around solving the limitations of our proposed work. First, our method only generalizes to the new start and goal configurations in given environments. Hence, we also aim to extend our neural field method to generalize across novel environment encoding architectures. Second, aside from addressing a few limitations, we also aim to explore novel PDE formulations to train physics-informed NMPs to solve motion planning and control under dynamic constraints.

References

[1] Z. Clawson, A. Chacon, and A. Vladimirsky, “Causal domain restriction for eikonal equations,” *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. A2478–A2505, 2014.

[2] M. G. Crandall and P.-L. Lions, “Viscosity solutions of hamilton-jacobi equations,” *Transactions of the American mathematical society*, vol. 277, no. 1, pp. 1–42, 1983.

[3] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] J. Huh, V. Isler, and D. D. Lee, “Cost-to-go function generating networks for high dimensional motion planning,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8480–8486.

[6] B. Ichter, J. Harrison, and M. Pavone, “Learning sampling distributions for robot motion planning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[7] B. Ichter and M. Pavone, “Robot motion planning in learned latent spaces,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.

[8] L. Janson, E. Schmerling, A. Clark, and M. Pavone, “Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions,” *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.

[9] R. Kumar, A. Mandalika, S. Choudhury, and S. Srinivasa, “Lego: Leveraging experience in roadmap generation for sampling-based planning,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1488–1495.

[10] S. M. LaValle, J. J. Kuffner, B. Donald, *et al.*, “Rapidly-exploring random trees: Progress and prospects,” *Algorithmic and computational robotics: new directions*, vol. 5, pp. 293–308, 2001.

[11] X. Li, S. Liu, S. D. Mello, X. Wang, M.-H. Yang, and J. Kautz, “Learning continuous environment fields via implicit functions,” in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=3ILxkQ7yElm

[12] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, “Fourier neural operator for parametric partial differential equations,” *arXiv preprint arXiv:2010.08895*, 2020.

[13] R. Ni and A. H. Qureshi, “NTFields: Neural time fields for physics-informed robot motion planning,” in *The Eleventh International Conference on Learning Representations*, 2023.

[14] ——, “Progressive learning for physics-informed neural motion planning,” *arXiv preprint arXiv:2306.00616*, 2023.

[15] ——, “Physics-informed neural motion planning on constraint manifolds,” *arXiv preprint arXiv:2403.05765*, 2024.

[16] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, “Motion planning networks,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[17] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, “Motion planning networks: Bridging the gap between learning-based and classical motion planners,” *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.

[18] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[19] J. A. Sethian, “A fast marching level set method for monotonically advancing fronts,” *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[20] J. D. Smith, K. Azizzadenesheli, and Z. E. Ross, “Eikonet: Solving the eikonal equation with deep neural networks,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 12, pp. 10 685–10 696, 2020.