# Deep Neural Network Enhanced Sampling-Based Path Planning in 3D Space

Jiankun Wang, *Member, IEEE*, Xiao Jia, Tianyi Zhang, Nachuan Ma, and Max Q.-H. Meng, *Fellow, IEEE*

***Abstract*— Robot path planning in 3D space is a challenging problem for its complex configuration. Sampling-based algorithms have gained great success in solving path planning problems in 3D space, but the quality of the initial path is not guaranteed and the convergence to the optimal solution is slow. To address these problems, in this article, we present a novel sampling-based path planning framework enhanced by the deep neural network (DNN) with applications to 3D space. In the proposed framework, we first train the DNN with a number of successful path planning cases in 3D space. Then the DNN is utilized to predict the promising region where the feasible path probably exists for a given path planning problem. This predicted promising region serves as a nonuniform sampling heuristic to bias the sampling process of the path planner. In this way, the path planner can focus on the promising region in the exploration and exploitation process so that the path planning speed gets accelerated. We conduct numerical simulations to evaluate the performance of the proposed algorithm and the results show that it can perform much better than conventional path planning algorithms. Furthermore, we also investigate the performance of different DNN architectures for path planning in 3D space.**

***Note to Practitioners*—In this work, we aim to provide an efficient learning-based method to accelerate the robot path planning process in 3D space. Conventional path planning algorithms need to perceive the environment first, and then implement a series of calculations such as collision checking and data storing to generate a feasible path. When facing complex and high-dimensional environments, they do not perform well. But the proposed neural network method in this article can predict the promising region where the feasible path exists for any given environment. This prediction result is used to guide the path planning process so that the algorithm performance can get significantly improved. Apart from sampling-based algorithms, the proposed neural network model can also be extended to other types of path planning algorithms.**

***Index Terms*— Robot path planning, deep neural network, sampling-based algorithm.**

## I. Introduction

PATH planning plays a fundamental role in robotics, which aims to provide a collision-free path for the robot to complete the specified task [1]. To solve the path planning problem, a geometric model of the configuration space is required. Then the grid-based algorithms like Dijkstra [2], A* [3], and D* [4] can be implemented to search a feasible path. However, constructing the configuration space is time-consuming, especially for robot movement in high-dimensional space like 3D space. Moreover, the grid-based methods only guarantee *resolution completeness*, which means that their performance depends on the discretization of the configuration space. The artificial potential field (APF) algorithm [5] does not need this discretization processing, but it often ends in a local minimum. Sampling-based algorithms can overcome the aforementioned limitations by employing a collision detection module and a graph (or tree) data structure. With the collision detector to compute whether a single state is collision free, the sampling-based algorithms can avoid the complex geometric modeling of the configuration space. The collision-free states are used to construct a data structure that stores collision-free paths. To obtain a feasible path, the rapidly-exploring random tree (RRT) [6] can be used for single-query or the probabilistic roadmap (PRM) [7] for multi-query.

However, the sampling-based path planner provides a weaker form of completeness, namely *probabilistic completeness*: if a feasible solution exists, it will be eventually found as the number of iterations goes to infinity. The reason is that the sampling-based planner uses a uniform sampling distribution to explore the state space randomly. Therefore, the quality of the initial path cannot be guaranteed and sometimes the planner even costs much time but find a bad path. On the other hand, Karaman and Frazzoli [8] propose an advanced version of the sampling-based planner, namely RRT* and PRM*, which can drive the initial path gradually to converge to an optimal path. This property is called *asymptotic optimality*: the feasible solution will eventually converge to an optimum as the number of iterations goes to infinity. In fact, the convergence speed is very slow since the sampling process is random.

Manuscript received 30 August 2021; accepted 13 October 2021. Date of publication 2 November 2021; date of current version 13 October 2022. This article was recommended for publication by Associate Editor L. Li and Editor C. Seatzu upon evaluation of the reviewers' comments. This work was supported in part by the Shenzhen Key Laboratory of Robotics Perception and Intelligence under Grant ZDSYS20200810171800001 and in part by the Southern University of Science and Technology, Shenzhen, China. *(Corresponding author: Max Q.-H. Meng.)*

Jiankun Wang, Tianyi Zhang, and Nachuan Ma are with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: wangjk@sustech.edu.cn; zhangty@mail.sustech.edu.cn; manc@mail.sustech.edu.cn).

Xiao Jia is with the Department of Radiation Oncology, Stanford University, Stanford, CA 94305 USA (e-mail: jiaxiao@stanford.edu).

Max Q.-H. Meng is with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China, on leave from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, and also with the Shenzhen Research Institute, The Chinese University of Hong Kong at Shenzhen, Shenzhen 518172, China (e-mail: max.meng@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2021.3121408.

Digital Object Identifier 10.1109/TASE.2021.3121408

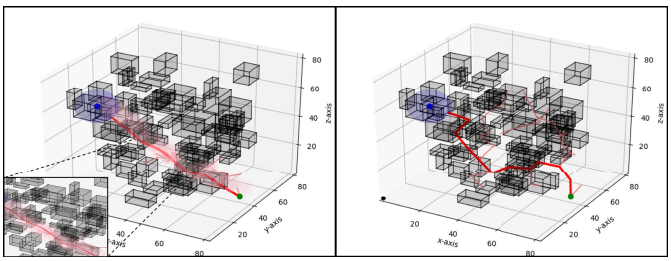


Fig. 1. Comparison between the 3D Neural RRT* algorithm and the conventional RRT* algorithm on finding the initial solution. The start and goal states are denoted as green and blue points, respectively. The goal region is marked as the blue spherical area. The red line represents the initial path, and the promising region is denoted with pink color.

In this article, we address these two limitations in sampling-based algorithms and propose a deep neural network (DNN) enhanced sampling-based path planning framework in 3D space. For a given path planning problem, the DNN can make a prediction of the feasible path and output the promising region. The promising region refers to the region that the feasible path probably exists. In other words, when a human looks at the environment once, he/she will basically know how to navigate from the start position to the goal. Therefore, the proposed DNN imitates human's capability to find the feasible path. But the generated 3D promising region is not a strict definition or guarantee for a feasible path, and it is more like a heuristic. To fill this gap, we tailor the DNN with a sampling-based path planner to achieve fast and efficient path planning. As shown in Fig. 1, under the guidance of the generated 3D promising region by DNN, which is denoted as the pink area, the proposed 3D Neural RRT* algorithm can quickly find a high-quality initial solution.

## II. Related Work

Over the years, sampling-based path planning algorithms have made a great success. RRT, PRM and their variants have been widely used in different areas including but not limited to autonomous vehicle [9], surgical navigation [10], and service robot [11].

To improve the path quality and accelerate the convergence speed to the optimal solution, many kinds of methods have been presented. The intuitive idea is to combine the advantages of conventional path planning algorithms. For example, sampling-based A* algorithm [12] uses an anytime A* heuristic to bias the sampling process to the goal region. Tahir *et al.* propose to use the potential field as a heuristic to guide the bidirectional RRT* [13]. Besides, researchers also try to utilize the topology and statistical knowledge to improve sampling-based path planning. RG-RRT [14] maintains a reachability set to guide the sampling for path planning under differential constraints. In [15], Wang *et al.* introduce the generalized Voronoi graph (GVG) technique to obtain a coarse path and then use a set of potential functions to achieve nonuniform sampling. Informed RRT* [16] is another effective variant of RRT*. It implements RRT* to find an initial solution at first. Then the following sampling will be restricted within an ellipse area which shrinks as the solution gets better. There are also some algorithms particularly designed for high-dimensional path planning. A quantitative analysis of the nearest-neighbors search for sampling-based planning in high-dimensional space is provided in [17]. Ferguson and Stentz [18] propose anytime dynamic RRTs in high-dimensional search spaces. A scalable motion planner [19] based on sampling technique is introduced to achieve effective path planning for high-dimensional kinematic systems.

More recently, learning-based methods have begun to be applied into the field of path planning. They either utilize explicit or implicit representations of policy or state space to aid and accelerate the path planning process. Zhang *et al.* propose to learn implicit sampling distributions for robot planning [20], while an explicit method is introduced in [21] to use a conditional variational auto-encoder (CVAE) to learn sampling distributions. The generative adversarial network (GAN) [22] is also an effective learning tool. In [23], GAN is used to learn an action sampler to guide search in high-dimensional state-action space. Kuo *et al.* [24] introduce deep residual models for sampling-based path planning to learn to plan more efficiently. However, these methods require specially designed local features or parametrized data structure in the learning process, which limits the generalization ability. Besides the methods to directly learn how to sample, there are also some proposed techniques on collision checking or critical region detection. For example, Molina *et al.* propose to use CNN to identify critical regions [25] for robot planning so that the sampling process can be biased to these regions. Kew *et al.* [26] introduce ClearanceNet to incorporate a neural collision clearance estimator to improve the planning performance. Yonetani *et al.* [27] reformulate the A* search algorithm and couple it with a convolutional encoder to form an end-to-end trainable neural network planner. Guo *et al.* [28] combine long short-term memory (LSTM) neural network with a fuzzy control algorithm to improve the efficiency of path planning. Li *et al.* [29] make use of CNN and graph neural network (GNN) to achieve multi-robot path planning.

Different from the aforementioned methods, in [30] we propose Neural RRT*, a pixel-based path planning framework based on CNN architecture. Without extra preprocessing such as local features design or data parametrization, it directly takes an image as input where the start state, goal state, free space, and obstacle space are denoted with different colors. Then the promising region (where the feasible path probably exists) is highlighted in the output, which can be used to guide the sampling process. To further investigate its generalization ability and improve the path planning performance in 3D space, in this work, we extend Neural RRT* to 3D space for efficient path planning.

Accordingly, the contributions in this work include:

- A novel and efficient DNN enhanced sampling-based path planning framework in 3D space;
- An investigation of different neural network architectures and their performance on 3D promising region prediction;
- Case studies to compare the performance of the conventional and DNN-based path planning algorithm;

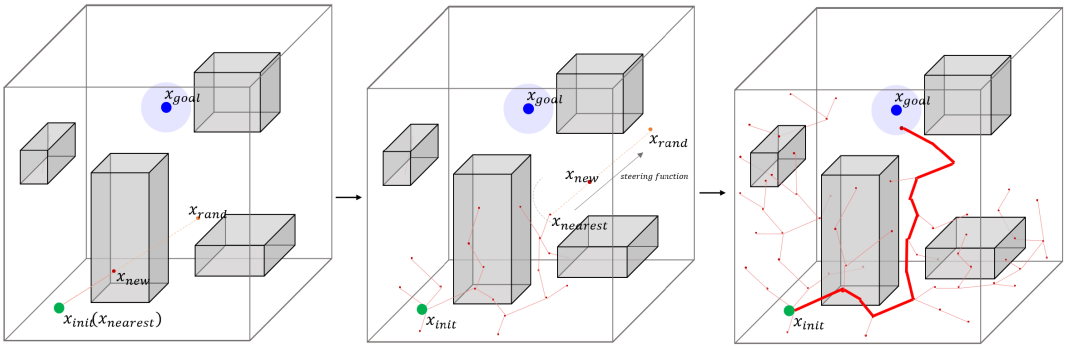



Fig. 2. Illustration of the RRT algorithm. The start and goal states are denoted as green and blue points, respectively. The goal region is marked as the blue spherical area, and the obstacles are represented as grey blocks. The red line denotes the initial path, and the lightcoral lines denote the RRT tree.

The rest of the article is organized as follows. We introduce some preliminaries of sampling-based robot path planning in Section III. Then the details of the proposed path planning framework are presented in Section IV. In Section V, we conduct numerical simulations to evaluate the algorithm performance. In Section VI, we make further discussion on the proposed method. Finally, we come to conclusions and discuss possible future directions in Section VII.

## III. Preliminaries

In this section, we first formulate the robot path planning problem in Section III-A, and then introduce the sampling-based algorithms in Section III-B.

### A. Robot Path Planning

Robot path planning aims to find a set of collision-free states connecting the start and goal states. Let $\mathcal{X} \in \mathbb{R}^3$ denote the state space, $\mathcal{X}_{obs}$ the space colliding with obstacles, and $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$ the free space, respectively. The start state and goal state are denoted as $x_{init}$ and $x_{goal}$, respectively. Usually, the goal region $\mathcal{X}_{goal} = \{x \in \mathcal{X} \big| \, ||x - x_{goal}|| < r\}$ is used, which refers to a ball centered at $x_{goal}$ with a radius $r$. With these definitions, the path planning problem can be formulated as

$$\sigma(t) : [0, 1] \rightarrow \mathcal{X}_{free}, \sigma(0) = x_{init}, \sigma(1) \in \mathcal{X}_{goal}, \tag{1}$$

where $\sigma(t)$ is a feasible path consisting of collision-free states. Let $\Sigma$ denote the set of all feasible paths, and $c : \sigma \rightarrow \mathbb{R}_{\geq 0}$ the cost function to measure the path. The optimal path planning problem can be defined as

$$\begin{aligned} \sigma^* = \arg\min_{\sigma \in \Sigma} \; & c(\sigma) \\ s.t. \; & \sigma(0) = x_{init} \\ & \sigma(T) \in \mathcal{X}_{goal} \\ & \sigma(t) \in \mathcal{X}_{free}, \quad \forall t \in [0, T]. \end{aligned} \tag{2}$$

In this article, we use the Euclidean distance as the cost of two adjacent states

$$\text{Cost}(x_i, x_j) = ||x_i - x_j||_2. \tag{3}$$

Therefore, the total cost of a given path is

$$c(\sigma) = \sum_{i=0}^{n-1} \text{Cost}(x_i, x_{i+1}). \tag{4}$$

### B. Sampling-Based Algorithm: RRT and RRT*

Sampling-based path planning algorithms are widely used to solve robot path planning problems in 3D space. RRT and PRM are two basic sampling-based algorithms where RRT focuses on single-query while PRM applies to multi-query scenarios. PRM requires a complete roadmap of the whole state space to achieve multi-query, which is a time-consuming work. To guarantee a real-time response and efficient space search, RRT becomes very popular in many robotic applications.

The path planning process of RRT is illustrated in Fig. 2. RRT grows a tree rooted at $x_{init}$ and employs a uniform sampler to randomly selects a state $x_{rand}$ from free space. Once $x_{rand}$ is determined, a node closest to $x_{rand}$ on current tree is selected as $x_{nearest}$. Then $x_{rand}$ is normalized as $x_{new}$ using a steering function. If the connection between $x_{nearest}$ and $x_{new}$ is collision-free, $x_{new}$ will be added to the tree. This process is repeated until a state located in $\mathcal{X}_{goal}$ is found, or the number of iterations reaches the threshold and a *failure* is reported. As for the RRT* algorithm, the difference is that a *ChooseParent* process and a *Rewiring* process are conducted when $x_{new}$ is added to the tree. The RRT* algorithm will continue after an initial path is found until the path cost converges to the optimum or the number of iterations reaches the threshold.

## IV. Algorithm

In this section, the details of the proposed DNN-based path planning framework are presented in Section IV-A. Then the formulation of DNN-based promising region prediction is discussed in Section IV-B and Section IV-C.

### A. 3D Neural RRT* Algorithm

*1) Algorithm Presentation:* As discussed in Section I, conventional sampling-based path planning algorithms randomly draw samples from the state space, which makes the convergence speed of finding the optimal solution very slow. In this

**Algorithm 1** 3D Neural RRT*

**Input** : $x_{init}, x_{goal}, Map$
**Output**: $\mathcal{T}$
1 $\mathcal{V} \leftarrow x_{init}, \mathcal{E} \leftarrow \emptyset; \mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E})$;
2 $\mathcal{P} \leftarrow$ 3DDNNModel($x_{init}, x_{goal}, Map$);
3 $\mathcal{H} \leftarrow$ Discretization($\mathcal{P}$);
4 **for** $i = 1 \ldots N$ **do**
5  **if** *A 3D initial path is not found* **then**
6  **if** Rand(0, 1) $< \mu_1$ **then**
7  $x_{rand} \leftarrow$ NonuniformSampling($\mathcal{H}$);
8  **else**
9  $x_{rand} \leftarrow$ UniformSampling($Map$);
10  **else**
11  **if** Rand(0, 1) $< \mu_2$ **then**
12  $x_{rand} \leftarrow$ NonuniformSampling($\mathcal{H}$);
13  **else**
14  $x_{rand} \leftarrow$ UniformSampling($Map$);
15  $x_{nearest} \leftarrow$ Nearest($\mathcal{T}, x_{rand}$);
16  $x_{new} \leftarrow$ Steer($x_{nearest}, x_{rand}$);
17  **if** ObstacleFree($x_{nearest}, x_{new}$) **then**
18  $\mathcal{T} \leftarrow$ Extend($\mathcal{T}, x_{new}$);
19  Rewire();
20  **if** $x_{new} \in \mathcal{X}_{goal}$ **then**
21  Return $\mathcal{T}$;
22 Return *failure*;

article, we propose 3D Neural RRT* algorithm to improve the sampling method. In our 3D Neural RRT* algorithm, we propose a heuristically nonuniform sampling strategy that is based on the promising region generated by DNN.

The proposed 3D Neural RRT* algorithm is shown in Alg. 1. Given a 3D environment map *Map* and a set of start and goal states $\{x_{start}, x_{goal}\}$, we first use a DNN-based predictor **3DDNNModel()** to generate the promising region $\mathcal{P}$, where feasible path probably exists. Then we use **Discretization()** to extract discrete heuristic states $\mathcal{H}$ from $\mathcal{P}$. The heuristic states $\mathcal{H}$ is also called heuristic sampling distribution, a collection of states which is used for biased sampling.

The details of our nonuniform sampling strategy are shown in Line 5 − 14, Alg. 1. We use a random number function **Rand()** to determine the position of $x_{rand}$, so that the sampler selects a state $x_{rand}$ in $\mathcal{H}$ in a certain proportion. Specifically, if the random number generated by **Rand()** is less than the given threshold $\mu$, the sampler will randomly choose a state in $\mathcal{H}$. Otherwise, the sampler will randomly choose a state from the whole state space $\mathcal{X}$. The guidance of the heuristic states $\mathcal{H}$ greatly reduces the sampling space, which can accelerate the speed of the RRT* algorithm to find an initial path and converge to the optimal path. On the other hand, keeping a certain proportion of uniform sampling increases randomness for our algorithm, which can guarantee *probabilistic completeness* and *asymptotic optimality*. In our algorithm, we choose different threshold $\mu$ in the initial stage and the optimization stage. The initial stage refers to the process of finding the first feasible path. In this period, we choose a greater threshold $\mu_1 = 0.9$ to further accelerate the search. The optimization stage refers to the process where the path continuously converges to the optimal solution. We choose a smaller threshold $\mu_2 = 0.5$ to avoid the algorithm stuck in the local optimum.

To sum up, the focus of this work lies in formulating an efficient model **3DDNNModel()** to predict the promising region $\mathcal{P}$ given the specific conditions (*Map*, $x_{init}$ and $x_{goal}$), which will be discussed in the following sections.

*2) Probabilistic Completeness and Asymptotic Optimality:* As introduced in Section I, *probabilistic completeness* indicates whether the proposed sampling-based path planning algorithm is able to find a feasible path if it exists after infinite iterations. In the 3D Neural RRT*, the nonuniform sampler draws samples from the heuristic sets $\mathcal{H}$ with the probability of $\mu_1$ while the uniform sampler draws samples from the whole 3D state space with the probability of $1 - \mu_1$. As the number of iterations $N$ goes to infinity, the following equation will hold

$$\lim_{N\to\infty} \mathbb{P}(\mathcal{V} \cap \mathcal{X}_{goal} \neq 0) = 1, \tag{5}$$

which means that each state in the whole 3D state space will be sampled. Therefore, *probabilistic completeness* of the 3D Neural RRT* algorithm is naturally guaranteed.

According to [8], the RRT* algorithm holds the theoretical guarantee of the *asymptotic optimality* if the following equation holds:

$$s_r > (2(1 + 1/w))^{1/w} \left(\frac{\alpha(\mathcal{X}_{free})}{\xi_w}\right)^{1/w}. \tag{6}$$

In Eq. 6, $s_r$ refers to the search radius in **Extend**($\mathcal{T}, x_{new}$), $w$ represents the dimensionality of the state space, which equals to 3 in this article. $\alpha(\mathcal{X}_{free})$ denotes the Lebesgue measure of $\mathcal{X}_{free}$ and $\xi_w$ indicates the volume of the unit ball in 3D Euclidean space. As the same procedures happen in the RRT* and the 3D Neural RRT*, we can draw the conclusion that the 3D Neural RRT* is also *asymptotically optimal*.

### *B. Data Generation*

For the training and testing of the DNN-based prediction model, we generate the dataset from successful path planning cases in random 3D environments, as shown in Fig. 3. Specifically, we use $80 \times 80 \times 80$ voxel map to represent the 3D environment map $\mathcal{O}$, where 20 to 90 blocks with size from $1 \times 1 \times 1$ to $10 \times 10 \times 10$ are generated to represent obstacles. A state map $\mathcal{S}$ with size $80 \times 80 \times 80$ voxels is also constructed to store the spatial location of the start and goal states.

To obtain the ground truth, we first conduct the A* algorithm [3] on each environment to get a feasible path, and then use the dilation operation to expand the path so that the neural network can better recognize them. Due to the small proportion of the start states and goal states in the state map $\mathcal{S}$, the start states and goal states are also expanded to blocks with size of $3 \times 3 \times 3$.

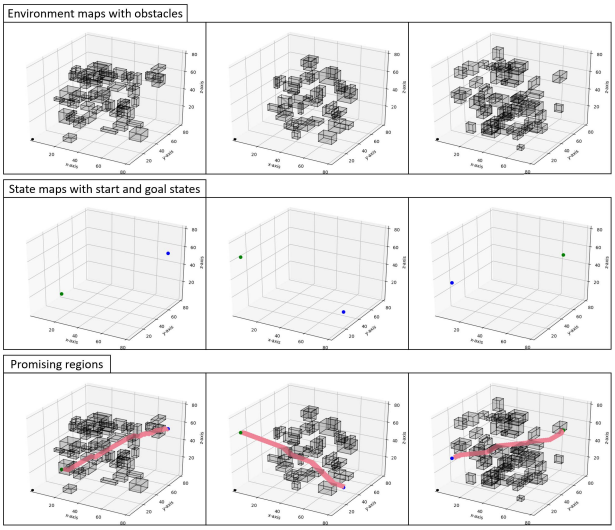



Fig. 3. Examples from the dataset. The first row represents the 3D environments with obstacles. The second row represents the state maps with start and goal states. The third row represents the corresponding promising regions, which are denoted as pink.

### C. 3D DNN Models

As an extension of [30], we first design a 3D CNN model,[1] which is explained in Section IV-C1. Inspired by the theory of generative model, we use 3D GAN architecture as well, which is illustrated in Section IV-C2. In order to facilitate the training process and reduce the network volume, we reshape the maps into $32 \times 32 \times 32$ voxels. For convenience, we denote the ground truth promising region map as $p_{gt}$, and the promising region map generated by the model as $p_o$.

*1) 3D CNN Model:* Fig. 4 illustrates the detailed encoder-decoder architecture of the proposed 3D CNN model. The inputs of the proposed 3D CNN model are $32 \times 32 \times 32$-dimensional environment map $o$ ($o \in \mathcal{O}$) and state map $s$ ($s \in \mathcal{S}$). The output is the corresponding promising region map $p_o$, with a dimension of $32 \times 32 \times 32$.

Inspired by U-Net [31], we convey the context information to higher resolution layers. At the input layer, $o$ and $s$ are fed into the network in separate channels to extract features in different scales. The features are then fed into the encoding network, a 4-layer repeated application of $3 \times 3 \times 3$ convolutions, each followed by batch normalization and rectified linear unit (ReLU). The decoding network consists of 4-layer $3 \times 3 \times 3$ deconvolutions, each followed by batch normalization and ReLU as well. Unlike U-net, we add the features along channels rather than concatenate them together to reduce the network's volume. At the final layer, we use the Sigmoid function to generate the probability map and set a threshold to segment the promising region.

The binary cross-entropy is used as the loss function:

$$\mathcal{L}_{\text{CNN}} = -\mathbb{E}_{p_{gt},p_o}[p_{gt} \cdot \log(p_o) + (1-p_{gt}) \cdot \log(1-p_o)]. \quad (7)$$

*2) 3D GAN Model:*

*a) Model structure:* The GAN model [22] consists of a generator $G$ and a discriminator $D$, with the aim to let $G$ learn a mapping from a latent space $\mathcal{Z}$ to a target data distribution $\mathcal{Y}$, which is denoted as $\{G : z \rightarrow y\}$. The generator $G$ is trained to generate outputs that can fool the discriminator $D$, whereas the discriminator $D$ is adversarially trained to improve the ability to distinguish target $y_{gt}$ and the output from the generator $y_o$. Adding a conditional information $u \in \mathcal{U}$, the GAN model can be extended to the conditional generative model (cGAN) [32]. cGAN is able to generate data that satisfies certain conditions or attributes, which is denoted as $\{G : u, z \rightarrow y\}$. In this article, we design a 3D cGAN-based model to generate promising region maps $\mathcal{P}$. In our model, we take 3D random noise $\mathcal{N}$ as the latent space $\mathcal{Z}$ and 3D environment maps $\mathcal{O}$ and state maps $\mathcal{S}$ as the conditional information space $\mathcal{U}$.

As shown in the upper network of Fig. 5, the generator $G$ takes $32 \times 32 \times 32$-dimensional environment map $o \in \mathcal{O}$, state map $s \in \mathcal{S}$, and noise map $n \in \mathcal{N}$ in separate Convolution3D-BatchNorm3D-ReLU layers to encode information at different scales. Similar to the 3D CNN model in Section IV-C1, the generator follows U-Net architecture. At the final layer, the generator $G$ uses Sigmoid as the activation function and outputs the predicted promising region map $p_o = G(o, s, n)$ and passes $p_o$ to the discriminator $D$.

The bottom network of Fig. 5 shows the structure of discriminator $D$. The inputs of $D$ are the predicted promising region map $p_o$ or the ground truth promising region map $p_{gt}$, conditioning with the information $o$ and $s$. $D$ consists of 4 volumetric convolution layers, each followed with batch normalization and leaky rectified linear unit (LeakyReLU) of parameter 0.2. The last layer is activated by Sigmoid function to output a real number in [0, 1], where $p_o$ is labeled as 0 and $p_{gt}$ is labeled as 1.

*b) Objective function:* According to [32], the objective function of a cGAN model can be expressed as:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D) &= \mathbb{E}_{u,y}[\log D(u, y)] \\ &\quad + \mathbb{E}_{u,z}[\log(1 - D(u, G(u, z))] \\ &= \mathbb{E}_{o,s,p_{gt}}[\log D(o, s, p_{gt})] \\ &\quad + \mathbb{E}_{o,s,p_{gt}}[\log(1 - D(o, s, p_o)]. \end{aligned} \quad (8)$$

The generator $G$ is trained to minimize the objective function and the discriminator $D$ is trained to maximize it, and the objective function is represented as:

$$\theta^* = \arg \min_G \max_D \mathcal{L}_{\text{GAN}}(G, D). \quad (9)$$

Inspired by [33], we add traditional loss to the cGAN objective to address the problem of training difficulty in large-scale GAN network,

$$\theta^* = \arg \min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) + \lambda \cdot \mathcal{L}_{\text{CNN}}, \quad (10)$$

where $\lambda$ is a weight coefficient and $\lambda = 2.5$.

## V. Simulation Results

In this section, the training details of the proposed 3D DNN models are introduced in V-A, and the performance comparison of the proposed 3D Neural RRT* and conventional RRT* algorithm is shown in V-B.

[1] "3D" means that the designed model is used to deal with 3D data input.

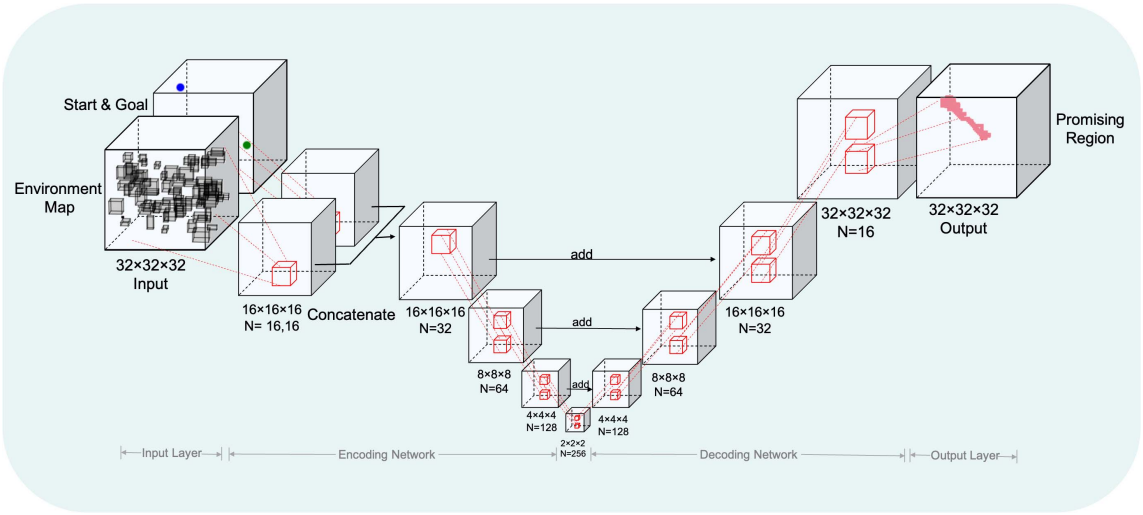



Fig. 4. The architecture of the 3D CNN model.

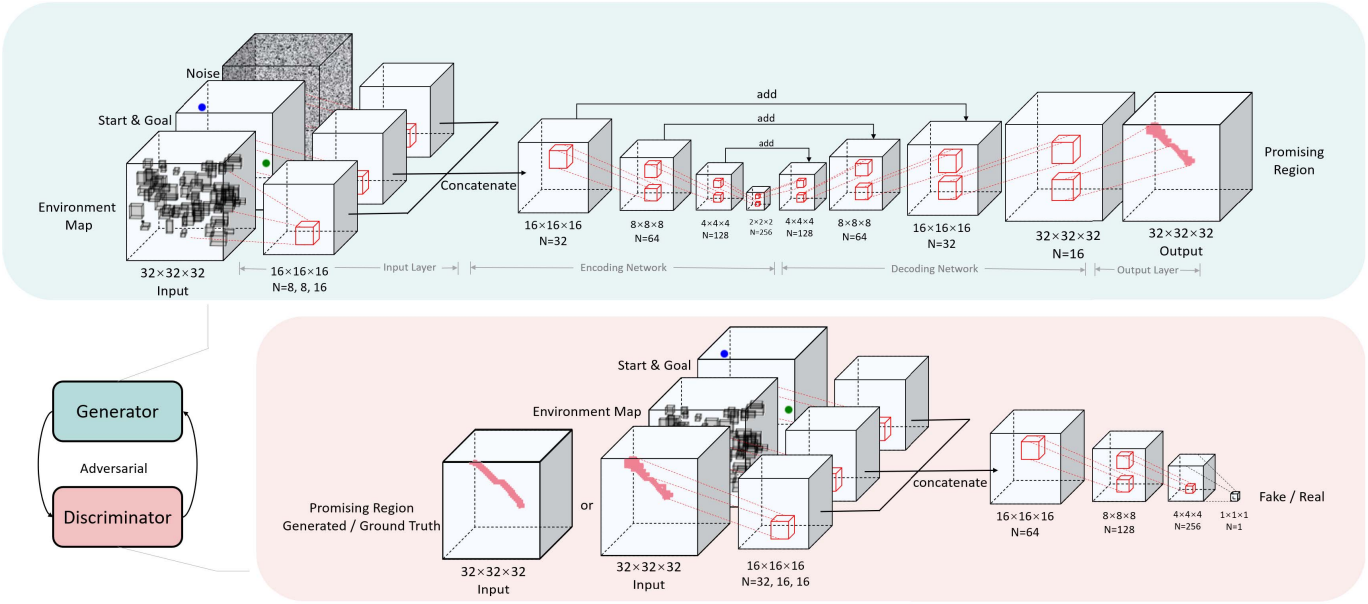



Fig. 5. The architecture of the proposed 3D GAN model. The upper network shows the generator, and the lower network shows the discriminator.

### *A. Training Details*

The proposed DNN models are trained with 15708 sets of 3D images on NVIDIA Geforce RTX 2080Ti in Python 3.8. We train the model for 30 epochs. 3D CNN model takes 1.82 hours to train, and 3D GAN model takes 3.17 hours.

In the proposed 3D CNN model, we set the learning rate to 0.0001 and use a batch size of 32 in the training process. We apply the Adam optimizer [34] for optimization, with parameters of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. As for the proposed 3D GAN models, we use the Adam optimizer with parameters of $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The batch size is set to 16, and the learning rates of the discriminator and generator are set to 0.000002 and 0.00001, respectively.

### *B. Comparison Between 3D Neural RRT* and RRT**

Some examples of the generated results from the proposed 3D DNN models are shown in Fig. 6. 3D images at the first row represent the ground truth maps, and images at the second row are the corresponding promising region maps generated by our DNN models. Fig. 6 demonstrates that the proposed DNN model can generate high-quality predicted promising regions in different environments, and the prediction results are used in Alg. 1 to guide the sampling process.

As the 3D GAN model shows almost the same performance with the 3D CNN model (more details can be found in Section VI-A), in this section, we apply the 3D GAN model to Alg. 1 and make comparisons with the conventional RRT* algorithm. We execute both 3D Neural RRT* and conventional RRT* on random environments 100 times to obtain statistical results of the initial solution and optimal solution, including the path cost, computing iterations and the number of sampling nodes. As shown in Eq. 3 and Eq. 4, the path cost refers to the path length. The computing iterations and the number of

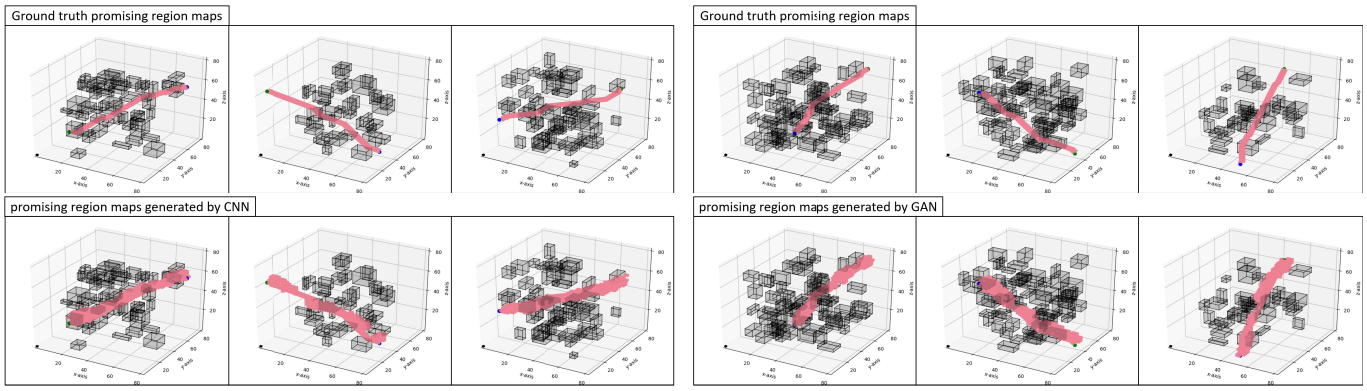
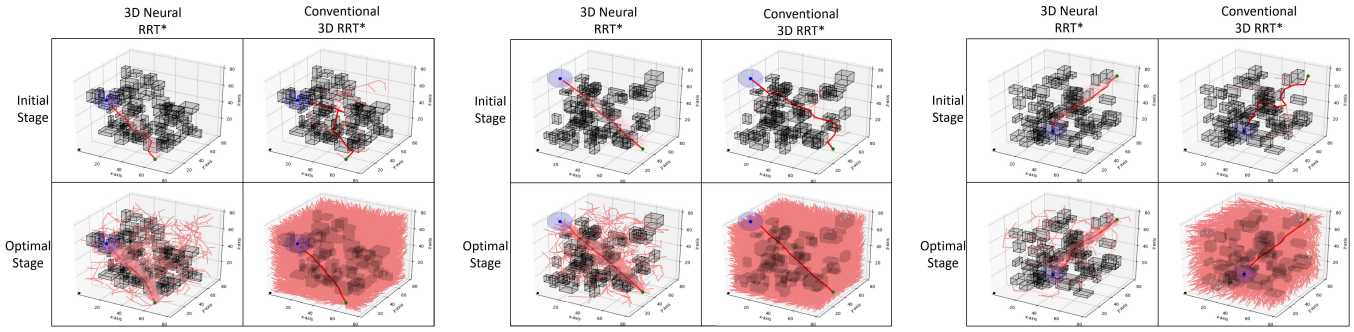



Fig. 6. Examples of the generated results from the proposed CNN model and GAN model. The images in the first row represent the ground truth promising region maps and the images in the second row represent promising region maps generated from the CNN model and GAN model.




(a) Map 1. (b) Map 2. (c) Map 3.

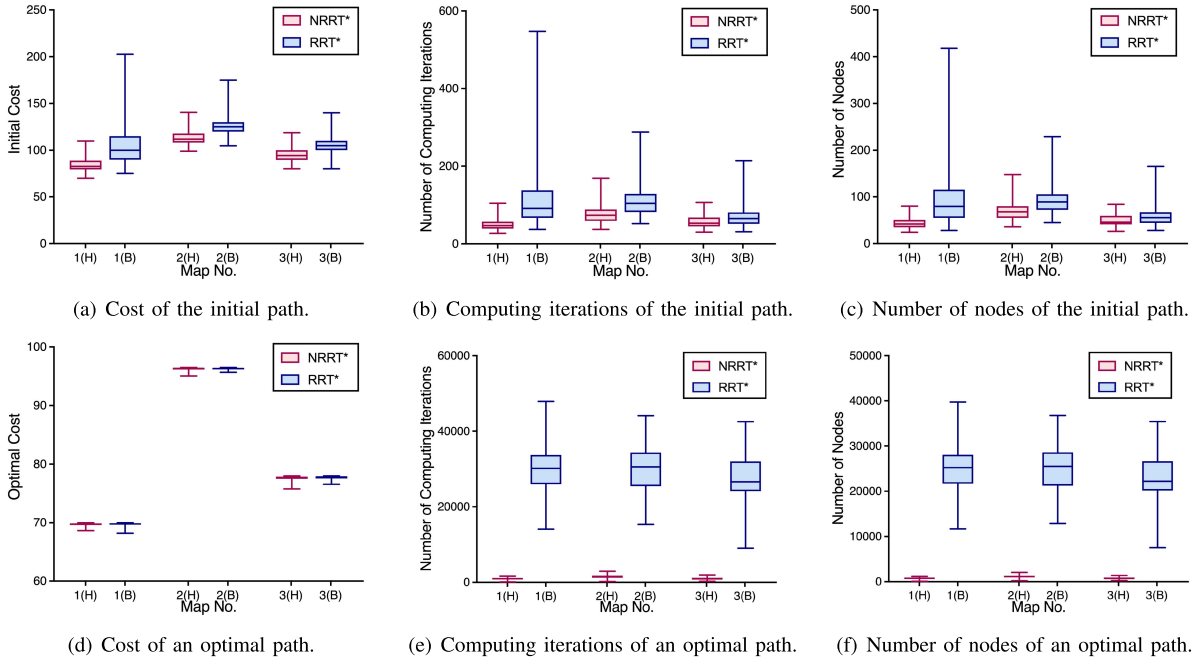
Fig. 7. Comparison of 3D Neural RRT* and conventional RRT* during the initial stage and optimization stage. Red lines refer to the paths, the lightcoral lines represent RRT expansion trees and the pink areas are predicted promising regions. The grey blocks refer to the obstacles, and the start and goal states are denoted as green and blue points, respectively.

sampling nodes represent the consumption of time and memory usage, respectively. Herein, we select three environment maps for illustration. The tree expansion is visualized in Fig. 7 and the statistical results are shown in Fig. 8.

Fig. 8 (a)(b)(c) show that the 3D Neural RRT* algorithm achieve better performance on finding the initial path in terms of the path cost, computing iterations, and memory usage. The heuristic method narrows the sampling space to the region where the feasible path probably exists, and the 3D Neural RRT* has more chances to expand its nodes in this region with the nonuniform strategy. It means that many unnecessary samples are avoided, and the exploration process is improved to facilitate more efficient exploitation. Therefore, the 3D Neural RRT* can find an initial path with a shorter length and fewer computing iterations.

Fig. 8 (d)(e)(f) show the statistical results on finding an optimal path. As mentioned in Section IV-A2, both 3D Neural RRT* and conventional RRT* are *asymptotically optimal*, which means that an optimal path will be eventually found after infinite iterations. However, considering the consumption of time and computing resources, these algorithms cannot be executed for infinite iterations in practical experiments. Therefore, when the path cost is less than the predefined values or the computing iterations reach the threshold, we consider that the corresponding algorithm has found the optimal solution. When the path cost is reduced to the predefined value, the path length will change slightly even if the algorithms continue to be executed. At this time, the path with the predefined value can be called as *asymptotically optimal* path. Herein, the threshold of computing iterations is set to 50000 times. When the descending gradient of the path cost is less than a certain value, we use the path cost at this moment as the predefined value.

It can be seen from Fig. 8 (d)(e)(f) that our algorithm demonstrates great superiority in the optimization stage. The 3D Neural RRT* can quickly converge to the *asymptotically optimal* path, and the computing iterations of the conventional RRT* are 20 to 30 times more than the 3D Neural RRT*. In each environment, the conventional RRT* needs much more computing iterations for convergence, which is impractical for real-time path planning. As the computing iteration goes up, the number of sampling nodes increases so that the conventional RRT* occupies much more memory usage. In the optimization process, RRT* decreases the path cost through sampling and rewiring. The number of sampling nodes in the high-dimensional space increases exponentially, which largely increases the number of nodes that need to be rewired during the rewiring process. Compared with the conventional RRT*, because more sampling nodes are taken in the region where the feasible path probably exists, the 3D Neural RRT* has a




(a) Cost of the initial path. (b) Computing iterations of the initial path. (c) Number of nodes of the initial path.

(d) Cost of an optimal path. (e) Computing iterations of an optimal path. (f) Number of nodes of an optimal path.

Fig. 8. Comparison between the proposed 3D Neural RRT* and the conventional 3D RRT* in terms of path cost, computing iterations and the number of nodes. The first and second rows represent the comparison during the initial stage and the optimization stage, respectively. The blue boxes indicate the statistical results of conventional 3D RRT*, and the red boxes refer to the results of 3D Neural RRT*.

greater probability of finding an *asymptotically optimal* path at an early stage.

Apart from the above advantages, Fig. 8 (a)-(f) shows that the variance of 3D Neural RRT* is much smaller, which means that the 3D RRT* maintains good stability. This is because the heuristic region reduces the randomness of sampling, thereby decreasing the possibility of extreme values. In conclusion, the 3D Neural RRT* shows significant advantages over the conventional RRT*.

## VI. Discussion

In this section, we make some extended discussions on our model. In Section VI-A, we show our method of judging the quality of promising regions directly from the 3D images. In Section VI-B, we make further comparisons with other variations of the RRT* algorithm to test the effectiveness of our algorithm.

### A. Evaluation Method

It will be more beneficial to evaluate the generation ability for the DNN models if we can judge the quality of the promising regions directly from the 3D images using a metric like precision, recall, etc. However, as discussed in Section I, the concept of the promising region is not a strict definition. Therefore, it is hard to build a unified standard to evaluate the result.

For path planning problems, the high-quality promising region can be defined as an unbroken area that covers the optimal path and connects the start and goal states. Therefore, if the generated promising region is good enough, we can place all the sampling nodes in the region, so that the optimal path can be found more quickly. We take this target as our test standard and use connectivity rate for evaluation: we run the RRT* algorithm only on the predicted promising region to test whether the region is connected. In other words, in the testing process, we only take the predicted promising region $\mathcal{P}$ as free space $\mathcal{X}_{free}$ and run RRT* algorithm. If a feasible path can be found, the predicted promising region is regarded as a successful case.

TABLE I
Connectivity Rate of 3D DNN Models

| Model | 3D CNN Model | 3D GAN Model |
|---|---|---|
| Connectivity Rate | 94.10% | 95.30% |

We evaluate our models with 3927 sets of environment maps, and the results are shown in TABLE I. TABLE I reveals the effectiveness of our proposed DNN models and indicates that using GAN structure can improve the accuracy and generalization ability.

It should be pointed out that connectivity rate is not a complete method because it only evaluates the quality of the promising region from one aspect. More systematic evaluation standards need to be further studied in future work.

### B. Further Comparison

Herein, we compare our 3D Neural RRT* with other heuristic RRT* methods to further evaluate our sampling strategy. We choose Informed RRT* [16], an effective variant of RRT* which uses an ellipsoid region to narrow down the sampling state space. We use the same comparison method as

TABLE II
COMPARISON OF DIFFERENT HEURISTIC-BASED RRT* ALGORITHMS

| No. | Method | Init. Cost | Init. Iters | Init. Nodes | Opt. Iters | Opt. Nodes |
|---|---|---|---|---|---|---|
| 1 | RRT* | 106.68 | 118.16 | 100.92 | 29374.28 | 24492.24 |
| | IRRT* | 110.26 | 97.09 | 98.36 | 27410.40 | 23049.48 |
| | NRRT* | **84.26** | **51.16** | **44.52** | **837.24** | **629.24** |
| 2 | RRT* | 123.93 | 110.92 | 100.28 | 30591.04 | 25501.32 |
| | IRRT* | 122.82 | 106.01 | 93.48 | 27109.96 | 22757.60 |
| | NRRT* | **114.11** | **78.36** | **71.60** | **1513.44** | **1131.16** |
| 3 | RRT* | 105.72 | 72.96 | 62.40 | 28428.96 | 23714.88 |
| | IRRT* | 105.49 | 66.88 | 57.76 | 25864.32 | 21776.80 |
| | NRRT* | **94.59** | **56.00** | **50.68** | **960.72** | **727.28** |

No.: Map Number
Init. Cost: Path Cost of Initial Solution
Init. Iters: Computing Iterations of Initial Solution
Init. Nodes: Sampling Nodes of Initial Solution
Opt. Iters: Computing Iterations of Optimal Solution
Opt. Nodes: Sampling Nodes of Optimal Solution
IRRT*: Informed RRT*
NRRT*: 3D Neural RRT*

Section V-B, and the average values of the statistical results are shown in TABLE II. It can be seen that the 3D Neural RRT* can achieve better performance on finding the initial solution and converging to the optimal solution.

## VII. CONCLUSION AND FUTURE WORK

In this article, we present a novel sampling-based path planning framework enhanced by 3D DNN models, which is called the 3D Neural RRT*. The proposed DNN models are able to generate accurate promising regions, which serve as nonuniform sampling distributions to bias the sampling process of RRT*. Numerical experiments are conducted to evaluate the performance of the proposed 3D DNN models and the 3D Neural RRT* algorithm. The proposed DNN models achieve good performance in different environments, and the 3D Neural RRT* algorithm shows great superiority compared with other variants of RRT*.

In future work, the following aspects need to be further studied. The first direction is to extend the proposed the 3D Neural RRT* to more complex situations, such as densely populated dynamic environments. As the predicting time of our DNN models is quite short, the DNN models can be applied to the dynamic process by repeating the DNN models prediction on multiple frames. Another extension is to enable the DNN models to take the motion constraints of robots into account (such as rotation angle, step size, etc.), which is beneficial for the DNN models to predict more refined promising regions. In addition, a challenging topic is to build systematic quality evaluation standards for the promising region, as is discussed in Section VI-A. The concept of the promising region is not a very strict definition, so a clear mathematical description needs to be further discussed.

## REFERENCES


[1] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2016.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[4] A. Stentz, "Optimal and efficient path planning for partially known environments," in *Intelligent Unmanned Ground Vehicles*. New York, NY, USA: Springer, 1997, pp. 203–220.

[5] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[6] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[9] S. U. Lee, R. Gonzalez, and K. Iagnemma, "Robust sampling-based motion planning for autonomous tracked vehicles in deformable high slip terrain," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 2569–2574.

[10] J.-H. Park, W. J. Park, C. Lee, M. Kim, S. Kim, and H. J. Kim, "Endoscopic camera manipulation planning of a surgical robot using rapidly-exploring random tree algorithm," in *Proc. 15th Int. Conf. Control, Autom. Syst. (ICCAS)*, 2015, pp. 1516–1519.

[11] J. Wang, M. Q.-H. Meng, and O. Khatib, "EB-RRT: Optimal motion planning for mobile robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 2063–2073, Oct. 2020.

[12] S. M. Persson and I. Sharf, "Sampling-based A* algorithm for robot path-planning," *Int. J. Robot. Res.*, vol. 33, no. 13, pp. 1683–1708, Nov. 2014.

[13] Z. Tahir, A. H. Qureshi, Y. Ayaz, and R. Nawaz, "Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments," *Robot. Auton. Syst.*, vol. 108, pp. 13–27, Oct. 2018.

[14] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 2859–2865.

[15] J. Wang and M. Q.-H. Meng, "Optimal path planning using generalized Voronoi graph and multiple potential functions," *IEEE Trans. Ind. Electron.*, vol. 67, no. 12, pp. 10621–10630, Dec. 2020.

[16] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst.*, Jun. 2014, pp. 2997–3004.

[17] E. Plaku and L. E. Kavraki, "Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning," in *Algorithmic Foundation of Robotics VII*. Berlin, Germany: Springer, 2008, pp. 3–18.

[18] D. Ferguson and A. Stentz, "Anytime, dynamic planning in high-dimensional search spaces," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 1310–1315.

[19] R. Luna, M. Moll, J. Badger, and L. E. Kavraki, "A scalable motion planner for high-dimensional kinematic systems," *Int. J. Robot. Res.*, vol. 39, no. 4, pp. 361–388, Mar. 2020.

[20] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Dec. 2018, pp. 3654–3661.

[21] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 7087–7094.

[22] I. J. Goodfellow *et al.*, "Generative adversarial networks," 2014, *arXiv:1406.2661*.

[23] B. Kim, L. P. Kaelbling, and T. Lozano-Pérez, "Guiding search in continuous state-action spaces by learning an action sampler from off-target search experience," in *Proc. AAAI*, 2018, pp. 6509–6516.

[24] Y.-L. Kuo, A. Barbu, and B. Katz, "Deep sequential models for sampling-based planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 6490–6497.

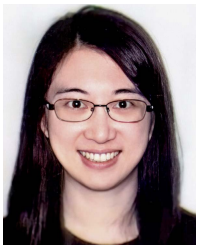
[25] D. Molina, K. Kumar, and S. Srivastava, "Learn and link: Learning critical regions for efficient planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 10605–10611

[26] J. Chase Kew, B. Ichter, M. Bandari, T.-W. Edward Lee, and A. Faust, "Neural collision clearance estimator for batched motion planning," 2019, *arXiv:1910.05917*.

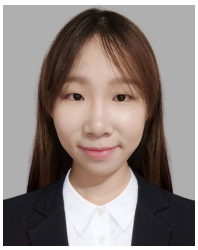
[27] B. Baginski and M. Eldracher, "Path planning with neural subgoal search," in *Proc. Int. Conf. Neural Netw. (ICNN)*, 2021, pp. 12029–12039.
[28] N. Guo, C. Li, D. Wang, Y. Song, G. Liu, and T. Gao, "Local path planning of mobile robot based on long short-term memory neural network," *Autom. Control Comput. Sci.*, vol. 55, no. 1, pp. 53–65, Jan. 2021.
[29] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Oct. 2020, pp. 11785–11792.
[30] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT*: Learning-based optimal path planning," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 1748–1758, Oct. 2020.
[31] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent*. Cham, Switzerland: Springer, 2015, pp. 234–241.
[32] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.
[33] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2016, pp. 2536–2544.
[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

**Jiankun Wang** (Member, IEEE) received the B.E. degree in automation from Shandong University, Jinan, China, in 2015, and the Ph.D. degree from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, in 2019.

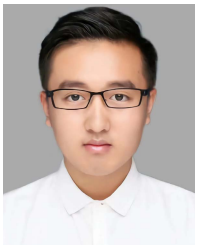
He is currently a Research Assistant Professor with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China. During his Ph.D. degree, he spent six months at Stanford University, CA, USA, as a Visiting Student Scholar, supervised by Prof. Oussama Khatib. His current research interests include motion planning and control, human robot interaction, and machine learning in robotics.



**Xiao Jia** received the B.E. degree in automation from Shandong University, Jinan, China, in 2014, and the Ph.D. degree from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, in 2020. She is currently a Post-Doctoral Fellow with the Department of Radiation Oncology, Stanford University, CA, USA. Her current research interests include medical image analysis, computer-aided diagnosis, and machine learning for image processing, with a recent focus on deep-learning-driven neural networks for lesion diagnosis.



**Tianyi Zhang** received the B.E. degree in automation from the Harbin Institute of Technology, Harbin, China, in 2020. She is currently a Research Assistant with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China. Her current research interests include machine learning in robotics and computer vision.



**Nachuan Ma** received the B.E. degree in electrical engineering and automation from the China University of Mining and Technology, Xuzhou, China, in 2019, and the M.Sc. degree in electronic engineering from The Chinese University of Hong Kong, Hong Kong, in 2020. He is currently a Research Assistant with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China. His current research interests include motion planning, and simultaneous localization and mapping (SLAM).

**Max Q.-H. Meng** (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Victoria, Victoria, Canada, in 1992.

He is currently the Chair Professor with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China, on leave from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, and also with the Shenzhen Research Institute, The Chinese University of Hong Kong at Shenzhen, China. He holds honorary positions as a Distinguished Professor with the State Key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin, China; a Distinguished Provincial Professor with the Henan University of Science and Technology, Luoyang, China; and the Honorary Dean of the School of Control Science and Engineering, Shandong University, Jinan, China. He has published more than 500 journal and conference papers and served on many editorial boards. His research interests include robotics, perception and sensing, human–robot interaction, active medical devices, biosensors and sensor networks, and adaptive and intelligent systems.

Dr. Meng is also serving as an Elected Member for the Administrative Committee of the IEEE Robotics and Automation Society. He received the IEEE Third Millennium Medal Award.