Neural RRT*: Learning-Based Optimal Path Planning

Jiankun Wang, Wenzheng Chi, Chenming Li, Chaoqun Wang, and Max Q.-H. Meng, *Fellow, IEEE*

***Abstract*— Rapidly random-exploring tree (RRT) and its variants are very popular due to their ability to quickly and efficiently explore the state space. However, they suffer sensitivity to the initial solution and slow convergence to the optimal solution, which means that they consume a lot of memory and time to find the optimal path. It is critical to quickly find a short path in many applications such as the autonomous vehicle with limited power/fuel. To overcome these limitations, we propose a novel optimal path planning algorithm based on the convolutional neural network (CNN), namely the neural RRT* (NRRT*). The NRRT* utilizes a nonuniform sampling distribution generated from a CNN model. The model is trained using quantities of successful path planning cases. In this article, we use the A* algorithm to generate the training data set consisting of the map information and the optimal path. For a given task, the proposed CNN model can predict the probability distribution of the optimal path on the map, which is used to guide the sampling process. The time cost and memory usage of the planned path are selected as the metric to demonstrate the effectiveness and efficiency of the NRRT*. The simulation results reveal that the NRRT* can achieve convincing performance compared with the state-of-the-art path planning algorithms.**

***Note to Practitioners*—The motivation of this article stems from the need to develop a fast and efficient path planning algorithm for practical applications such as autonomous driving, warehouse robot, and countless others. Sampling-based algorithms are widely used in these areas due to their good scalability and high efficiency. However, the quality of the initial path is not guaranteed and it takes much time to converge to the optimal path. To quickly obtain a high-quality initial path and accelerate the convergence speed, we propose the NRRT*. It utilizes a nonuniform sampling distribution and achieves better performance. The NRRT* can be also applied to other sampling-based algorithms for improved results in different applications.**

***Index Terms*— Convolutional neural network (CNN), optimal path planning, sampling-based path planning.**

Manuscript received September 3, 2019; revised November 28, 2019 and January 20, 2020; accepted February 23, 2020. Date of publication March 16, 2020; date of current version October 6, 2020. This article was recommended for publication by Associate Editor M. Marcos and Editor Y. Ding upon evaluation of the reviewers' comments. This work was supported in part by the Hong Kong Innovation and Technology Commission (ITC) Innovation and Technology Support Programme (ITSP) Tier2 under Grant ITS/105/18FP and in part by the Hong Kong ITC Midstream Research Programme for Universities (MRP) under Grant MRP/011/18. *(Corresponding author: Max Q.-H. Meng.)*

Jiankun Wang, Chenming Li, Chaoqun Wang, and Max Q.-H. Meng are with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, China (e-mail: jkwang@cuhk.edu.hk; licmjy@link.cuhk.edu.hk; cqwang@cuhk.edu.hk; max.meng@cuhk.edu.hk).

Wenzheng Chi is with the School of Mechanical and Electric Engineering, Soochow University, Suzhou 215021, China (e-mail: wzchi@suda.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2020.2976560

I. Introduction

ROBOTIC path planning is to plan a collision-free path for some specific tasks among a number of static or moving obstacles [1]. Many well-established algorithms have been proposed to solve the path planning problems. The artificial potential field (APF) method [2] utilizes a potential function over the whole configuration space to direct the movement of the robot. However, it often ends up in a local minimum. The grid-based algorithms including the A* [3] and D* [4] can guarantee finding an optimal path if it exists. Here, the generated optimal path is called "resolution optimal" because the map discretization under different resolutions will lead to different optimal paths. But their time cost and memory usage increase exponentially with the map size and the dimensionality of the state space. The sampling-based algorithms, such as probabilistic roadmap (PRM) [5] and rapidly random-exploring tree (RRT) [6], have become very popular for efficiently solving high-dimensional and multiconstrained path planning problems. It is noted that the sampling-based algorithms only provide a weaker form of completeness (also called probabilistic completeness) and the generated paths are often nonoptimal. An essential factor in determining the performance of the sampling-based planner is the sampling distribution because all sampling-based planners iteratively construct trees to connect samples drawn from the given sampling distributions. Usually, planners probabilistically or deterministically draw random state samples from a uniform distribution. Such a sampling strategy can guarantee the probabilistic completeness and asymptotic optimality [7]. However, due to environment types, clearance requirements, robot dynamics, and other constraints, robots often work in the small subsets of the state space and drawing state samples in the other parts of the state space is not necessary. For example, in environments with sparse obstacles, it is reasonable to bias the samples toward the goal region because the final path will be relatively straight. Therefore, the placement of the samples in these promising regions is closely related to the performance of the sampling-based path planners.

In [8], Brunner *et al.* propose the A*-RRT* algorithm, where an initial path generated from the A* algorithm is used to guide the sampling process of the RRT* planner. The convergence speed can be accelerated significantly as the A* algorithm always provides an optimal path. However, as the map size or step size used in the search process increases, the A* algorithm consumes much more time and memory to find the optimal path, as shown in Fig. 1. It affects

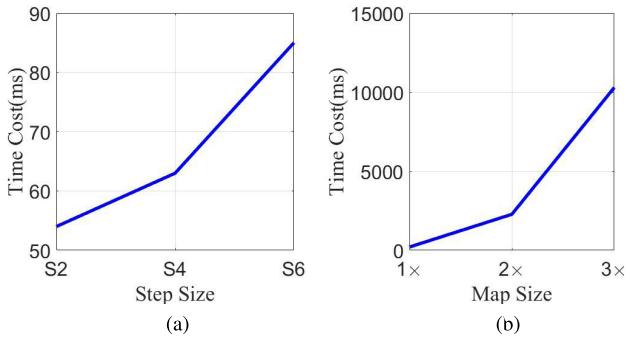

Fig. 1. Time cost of the A* algorithm on finding the optimal path under different step size and map size. (a) Time cost of the A* algorithm increases as the step size becomes larger. S2 means that the step size is 2. (b) Time cost of the A* algorithm increases significantly as the map size becomes double and triple.

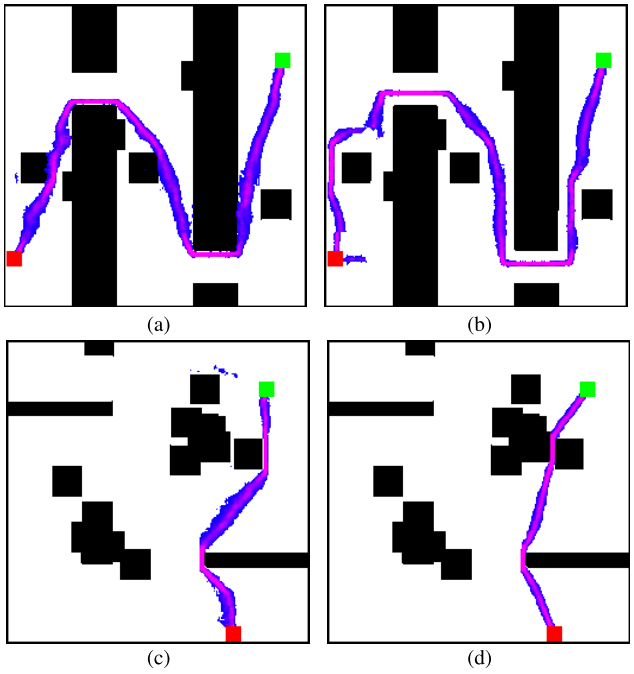

Fig. 2. Probability distribution of the optimal path given different clearance and step size. The red and green rectangles denote the start and goal states, respectively. As the color changes from blue to magenta, the existence probability of the optimal path in current region becomes larger and larger. (a) Clearance = 1, Step size = 2. (b) Clearance = 6, Step size = 2. (c) Clearance = 1, Step size = 1. (d) Clearance = 1, Step size = 4.

the algorithm performance, especially in complex dynamic environments. The test map is shown in Fig. 2(a), and the resolution of map discretization is 1. In this article, it means that the minimum distance between two states is 1 pixel.

In order to overcome the aforementioned limitations and further take advantage of the A* and the RRT* algorithms, in this article, we propose a novel optimal path planning algorithm through learning from the A* algorithm. We train a convolutional neural network (CNN) model by learning a large number of optimal paths generated from the A* algorithm. Given a new path planning problem, our trained model can rapidly provide a predicted probability distribution of the optimal path, which is used to guide the sampling process of the RRT* planner. In the A* algorithm, the generated path has little clearance to the obstacles so that it is dangerous for a robot to track due to the uncertainty in sensor measurement or robot dynamics [9]. It means that we will get different optimal paths if we set a different clearance value in the path planning problem. In addition, when the step size is different in the search process in the A* algorithm, the final optimal path is different for the same environment. Therefore, we take into account these two factors in the training process and our CNN model captures the differences so that it can output different probability distributions of the optimal path for the same environment with different parameter settings, as shown in Fig. 2. Then, the predicted sampling distribution is used to guide the sampling process. The other components of the Neural RRT*(NRRT*) are similar to those of the RRT* [7] algorithm.

A. Related Work

Recently, many researchers have proposed different heuristics to accelerate the convergence of the RRT algorithms to an optimal path.

A common method is to combine the discrete search with a sampling-based path planner. Bekris and Kavraki [10] utilize the informed subdivision tree as a heuristic to guide the growth of the RRT tree. As a two-phase motion planner, the A*-RRT* algorithm uses a geometrically feasible path to bias the sampling process of the RRT*. Rickert *et al.* [11] propose the exploring/exploiting tree (EET) algorithm to utilize the workspace information to bias the tree growth, but the probabilistic completeness is not guaranteed. By using any-angle path biasing, Palmieri *et al.* [12] propose the Thete*-RRT* algorithm, which can find a shorter path faster than the RRT* and the A*-RRT* algorithms. However, these aforementioned methods become inefficient when the map size or the dimensionality of the state space increases.

There are also other good heuristics used in the RRTs algorithms. For example, Qureshi and Ayaz [13] use the potential function as the sampling heuristic. In [14], Wang *et al.* implement a 2-D Gaussian mixture model to find a high-quality initial solution and adjust the sampling behavior of the RRT algorithms. Yershova *et al.* [15] present dynamic-domain RRT, which ignores the samples that are too far from the current tree. In [16], the obstacle boundary information is used to guide the sampling process. Gammell *et al.* [17] present the informed RRT* (IRRT*) to accelerate the convergence to an optimal path using an admissible ellipsoidal heuristic. Urmson and Simmons [18] compute a heuristic quality that is used to evaluate the quality of a path passing through the node. The sampling process will focus on the regions containing high-quality nodes. By limiting the sampling region, the planner can avoid unnecessary exploration in these regions where the feasible path probably does not exist and as a result, the performance becomes better. In [19], the generalized Voronoi graph and multiple potential functions are used to achieve efficient path planning. While the previous work is effective for robots in certain environments, they are not

generally applicable. Sometimes the heuristic can increase the planning time dramatically if it is not suited to the concerned environment.

Recently, reinforcement learning and deep learning methods are widely used in sampling-based planners. Baldwin and Newman [20] use semantic information to learn the sampling distribution. Zucker *et al.* use the reinforcement learning method to optimize the sampling process in the discretized workspace. Another reinforcement learning technique called value iteration networks (VIN) is shown capable of effectively finding near-optimal trajectories in 2-D mazes and 3-D landscapes. However, VIN suffers from several disadvantages such as the training instability and the random seed sensitivity. Ichter *et al.* [21] propose a conditional variational autoencoder (CVAE) to learn explicit sampling distribution for robot motion planning. In [22], Qureshi and Yip propose to use a contractive autoencoder to achieve deeply informed neural sampling. They take into consideration the workspace encoding, the initial and the goal states to generate an end-to-end feasible path. Zhang *et al.* [23] present a policy-search based method as an adaptive way to learn implicit sampling distribution through a rejection sampling strategy. Li *et al.* [24] utilize the neural network to predict the cost function in the RRT algorithm.

Different from the previous work, our methodology does not propose a human-designed heuristic or require any discretization of the state space. Moreover, it learns the probability distribution of the optimal path with different attributes and can be easily combined with other sampling-based algorithms to further improve the algorithm performance.

B. Original Contributions

The contributions of this article are 1) to propose a CNN model to predict a probability distribution of the optimal path for the environment with different parameter settings, 2) present a novel optimal path planning algorithm, the NRRT*, and 3) provide the proof of the probabilistic completeness and the asymptotic optimality of the NRRT*. In fact, the predicted probability distribution helps the path planner achieve nonuniform sampling in the current environment. This method can be generalized to learn the probability distribution of the optimal path from the other path planners with different constraints and it can be also easily applied to other sampling-based algorithms.

The rest of this article is organized as follows. We first formulate the path planning problem in Section II and then explain the details of the proposed NRRT* algorithm in Section III. The simulation results are reported in Section IV. At last, we draw conclusions and discuss the future work in Section V.

II. Preliminaries

In this section, we first formulate the path planning algorithm and then provide an overview of the RRT* algorithm.

A. Problem Formulation

The basic path planning problem can be defined as follows. Let $\mathcal{X} \in \mathbb{R}^n$ be the state space. The obstacle space and the free space are denoted as $\mathcal{X}_{\text{obs}}$ and $\mathcal{X}_{\text{free}}$, respectively. Let x_{init} be the initial state and x_{goal} be the goal state. Usually, a goal region $\mathcal{G}(x_{\text{goal}}) = \{x \in \mathcal{X} \mid ||x - x_{\text{goal}}|| < r\}$ is used. The aim is to compute a path $\sigma : [0, T] \to \mathcal{X}_{\text{free}}$ such that $\sigma(0) = x_{\text{init}}$ and $\sigma(T) \in \mathcal{G}(x_{\text{goal}})$.

Let Σ be the set of all feasible paths. With a cost function, $c(\sigma)$, by mapping each feasible path $\sigma \to \mathcal{X}_{\text{free}}$ to a positive real number $\mathbb{R}$, the optimal path planning problem is defined as

$$\begin{aligned} \sigma^* = \arg\min_{\sigma \in \Sigma} \; & c(\sigma) \\ \text{s.t. } & \sigma(0) = x_{\text{init}} \\ & \sigma(T) \in \mathcal{G}(x_{\text{goal}}) \\ & \sigma(t) \in \mathcal{X}_{\text{free}}(t) \quad \forall t \in [0, T]. \end{aligned} \tag{1}$$

In this article, the cost function between two states is defined as follows:

$$c(x_1, x_2) = \alpha_1 ||x_2 - x_1|| + \alpha_2 \arccos \frac{\overrightarrow{v_1} \cdot \overrightarrow{x_1 x_2}}{|\overrightarrow{v_1}||\overrightarrow{x_1 x_2}|} \tag{2}$$

where $||\cdot||$ is the Euclidean distance between x_1 and x_2, $\overrightarrow{v_1}$ is the robot linear velocity at x_1, and $\overrightarrow{x_1\ x_2}$ is the vector from x_1 to x_2. α_1 and α_2 are two parameters to balance the effect of the Euclidean distance and angle difference, respectively. So the cost function c takes into account the Euclidean distance and angle difference between two states. For an arbitrary path σ, a smaller value of $c(\sigma)$ means that the robot takes less time to move from the start position to the goal position.

Algorithm 1 RRT*.

Input : $x_{init}, \mathcal{G}(x_{goal})$ *and* Map
Output: $\mathcal{T}$

$V \leftarrow x_{init}, E \leftarrow \emptyset, \mathcal{T} = (V, E)$;
for $i = 1 \ldots N$ **do**
  $x_{rand} \leftarrow$ UniformSample();
  $x_{nearest} \leftarrow$ Nearest($\mathcal{T}, x_{rand}$);
  $x_{new} \leftarrow$ Steer($x_{nearest}, x_{rand}$);
  if *ObstacleFree*($x_{nearest}, x_{new}$) **then**
    $\mathcal{T} \leftarrow$ Extend($\mathcal{T}, x_{new}$);
    Rewire();
    if $x_{new} \in \mathcal{G}(x_{goal})$ **then**
      Return($\mathcal{T}$);
Return $failure$;

B. RRT and RRT*

Generally, sampling-based path planners like the RRT and the PRM continuously draw samples x_{rand} from the state space using a uniform sampling distribution to construct the feasible path σ^* in terms of a cost function $c(\sigma)$. The RRT algorithm aims at single-query application while the PRM algorithm focuses on multiquery application. Although multiquery approaches are valuable in highly structured environments, most online planning problems do not require multiple queries since the environment changes dynamically. Moreover, in some applications, computing a roadmap for the PRM algorithm is computationally challenging. However, incremental

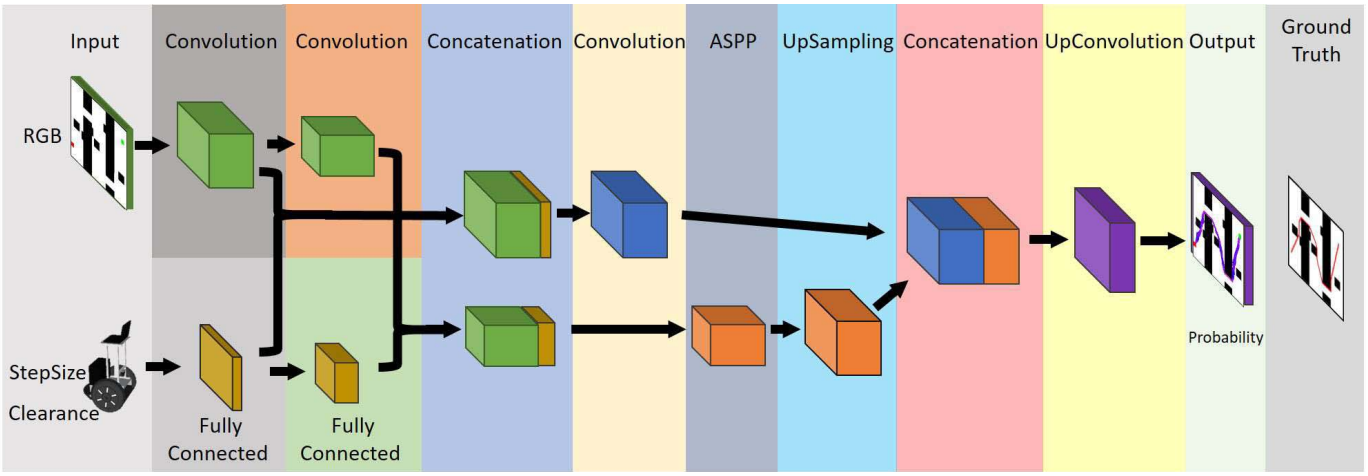

Fig. 3. Illustration of the proposed CNN model.

sampling-based algorithms such as RRTs have become popular and efficient approaches for many applications.

The RRT algorithm is initialized with a tree including the initial state as one vertex and no edges. At each iteration, the sampled state x_{rand} from a uniform sampling distribution attempts to connect the nearest vertex x_{nearest} in the tree. If this connection is successful, x_{rand} will be adjusted to x_{new} through a steering function. Then, x_{new} is added to the vertex set and $(x_{\text{nearest}}, x_{\text{new}})$ is added to the edge set, respectively. The algorithm will stop until the tree contains a state in the goal region or the number of iterations reaches the threshold.

Compared with the RRT, the RRT*, as shown in Algorithm 1, has two extra procedures, *ChooseParent* in the **Extend**$(\mathcal{T}, x_{\text{new}})$ function and *Rewire* process. In the *ChooseParent* procedure, when the RRT* tries to connect x_{new} to x_{nearest}, the vertices around x_{new} within a certain radius will be searched to determine the optimal parent in terms of the cost. After connecting x_{new} to the tree, the RRT* will *rewire* the neighbor vertices to check if one path through x_{new} has a lower cost than the current path. The details of the RRT* can be found in [7]. Therefore, the asymptotic optimality can be guaranteed as the number of iterations goes to infinity. But it consumes much time and memory usage to reach the optimal solution. In the real-world implementation, especially for mobile robots or flying robots, it is necessary to compute high-quality paths efficiently.

III. NRRT* Algorithm

In this section, we first describe the proposed CNN model formulation in Sections III-A and III-B. Then we introduce the NRRT* and analyze its probabilistic completeness and asymptotic optimality in Sections III-C and III-D.

A. CNN Model

In this section, we describe our model formulation. We use 2-D images to represent the 2-D state space, which is denoted as $\mathcal{I}$. Each pixel in the 2-D image has an assigned value. 0 means empty free space and 1 indicates occupied by the obstacles. 2 labels start position and 3 means goal position. Besides we also consider the robotic attributes like clearance denoted as $\mathcal{C}$ and robot step size denoted as $\mathcal{S}$.

The input $\mathcal{I}$ is fed into our neural network together with these robotic attributes $\mathcal{C}$ and $\mathcal{S}$. Our neural planning model outputs the corresponding image denoted as $\mathcal{O}$. Each pixel receives a probability $\hat{p} \in [0, 1]$ indicating the chance that it will be used to construct the optimal path. It means that if the value of $\hat{p}$ of a state is large, the optimal path probably goes through this state.

Our proposed model is shown in Fig. 3. It has an encoding and decoding stage with atrous convolution [25]. This type of deep convolutional networks has been widely used in artificial intelligence-related fields like computer vision and machine learning and gains great success.

During the encoding stage, 2-D residual convolution network [26], [27] is applied into the input $\mathcal{I}$ to extract features of the input map in a hierarchical fashion.

Take an image I_{uv} with a size of $(W, H, 3)$ as an example. Each 2-D convolutional layer has its own small kernel K_{ij}. The convolution operation is defined as follows:

$$(I * K)(u, v) = \sum_{i=-I}^{I} \sum_{j=-J}^{J} I(u - i, v - j) \times K(i, j) \quad (3)$$

where I and J are the size of the convolutional kernel K.

The convolution operation is applied to the whole image and the output of the convolution operation is called the feature map. The feature map is fed into a higher convolution layer to extract another feature map. Low-level feature maps F_l and high level features maps F_h are convolution results at low convolutional layers and high convolutional layers, respectively. Low-level feature maps encode local information while high-level feature maps represent high-level description of the map.

We adopt ResNet50 [26] as the backbone of our network encoder. We extract feature map $C1$ and $C4$ in ResNet50 as the low-level feature and high-level feature. Feature map $C1$ has a dimension of $(W/8, H/8, 256)$, and feature map $C4$ has a dimension $(W/32, H/32, 2048)$.

Atrous convolution is a general form of convolution and is defined in the following way:

$$(I * K)(u, v; r) = \sum_{i=-I}^{I} \sum_{j=-J}^{J} I(u - i \times r, v - j \times r) \times K(i, j) \tag{4}$$

where r is the dilated rate. Atrous convolution allows us to explicitly control the resolution of feature maps computed by deep CNNs and adjust kernel's field-of-view in order to capture multiscale information. Atrous spatial pyramid pooling (ASPP) [28] module is applied to the high-level features to extract multiscale information in our network. The final feature map F_h has a dimension of $(W/32, H/32, 256)$.

On the other input branch, these input robotic attributes $\mathcal{S}$ and $\mathcal{C}$ are fed into fully connected neural networks. The actual inputs of these attributes are float numbers. The corresponding low-level and high-level feature maps are extracted and denoted as F_l^a and F_h^a. F_l^a has a dimension of $(1, 1, 32)$ and F_h^a has a dimension of $(1, 1, 64)$.

Features of the map and features of these robotic attributes are concatenated together because we want our neural network model to be able to adjust the optimal path according to these robotic attributes. Note that feature maps at the same level are concatenated together. F_h^a is repeated along the image width and height axis by $(W/32)$ and $(H/32)$. The corresponding feature map is $(W/32, H/32, 64)$ and is concatenated along the third dimension element-wise. Low-level feature concatenation follows the same principles. The low- and high-level feature maps after the concatenation operation are denoted as F_l^c and F_h^c, respectively. F_l^c has a dimension of $(W/8, H/8, 256 + 32)$ and F_h^c has a dimension of $(W/32, H/32, 256+64)$. The F_h^c is linearly resized to have a dimension of $(W/8, H/8, 256+64)$ and is concatenated with F_l^c element-wise. The final encoding feature is denoted as F^e has a dimension of $(W/8, H/8, 608)$.

In the decoding stage, the feature map F^e is fed into the decoding convolutional network to get the final output $\mathcal{O}$. The final output has a dimension of $(W, H, 1)$ with the same resolution as the input $\mathcal{I}$. Each pixel at (u, v) contains a probability $\hat{p}$ indicating the chance that the pixel is inside the optimal path.

In our proposed neural network, the fully convolutional network fashion [29] is adopted to process the input $\mathcal{I}$ and the output $\mathcal{O}$. A fully convolutional network can handle input $\mathcal{I}$ with different resolutions.

B. Loss Function

We define the loss function as the cross-entropy of predicted probability and ground truth binary labeling per pixel. The cross-entropy loss is summed among all possible locations on the map

$$L = \sum \text{ce}(\mathcal{O}, \mathcal{G}) + \lambda \text{ce}(\mathcal{I}, \mathcal{R}) \tag{5}$$

where λ is a parameter to weigh the loss of reconstruction. $\mathcal{O}$ represents the predicted probability image, and $\mathcal{G}$ denotes the ground truth probability image. The generation procedure of ground truth $\mathcal{G}$ is described in Section IV-A.

Algorithm 2 NRRT*.

Input : $x_{init}, \mathcal{G}(x_{goal}), Map, \mathcal{S}, \mathcal{C}$
Output: $\mathcal{T}$

$V \leftarrow x_{init}, E \leftarrow \emptyset, \mathcal{T} = (V, E)$;
$\mathcal{O} \leftarrow$ NeuralModel$(Map, \mathcal{S}, \mathcal{C})$
for $i = 1 \ldots N$ **do**
  if *Rand()* > 0.5 **then**
    $x_{rand} \leftarrow$ NonuniformSample$(\mathcal{O})$;
  else
    $x_{rand} \leftarrow$ UniformSample();
  $x_{nearest} \leftarrow$ Nearest$(\mathcal{T}, x_{rand})$;
  $x_{new} \leftarrow$ Steer$(x_{nearest}, x_{rand})$;
  if *ObstacleFree*$(x_{nearest}, x_{new})$ **then**
    $\mathcal{T} \leftarrow$ Extend$(\mathcal{T}, x_{new})$;
    Rewire();
    if $x_{new} \in \mathcal{G}(x_{goal})$ **then**
      Return$(\mathcal{T})$;
Return $failure$;

C. Neural RRT*

In the NRRT*, the trained neural network model is used to initialize the sampling process at first. The model outputs a pretty sharp probability distribution of the optimal path for the current map. If we directly use this probability distribution, the probability that quantities of states are selected in the sampling process will approach zero. Then the probability of completeness of the algorithm is hardly guaranteed. In order to maintain the ability of the NRRT* to represent the environment with arbitrarily high fidelity, and thus maintain the theoretical completeness guarantee, we also sample from a uniform sampler. So there are two samplers in the NRRT*, a nonuniform sampler from the neural network model and a uniform sampler. We denote the fraction of uniform samples as α, i.e., αN samples are generated from the uniform sampler and $(1-\alpha)N$ from the nonuniform sampler. In fact, as shown in Section IV, sometimes the predicted probability distribution is not continuous. If we set $\alpha = 0$ (i.e., all samples from the nonuniform sampler), the path planner will never find a solution. In addition, if we set $\alpha = 1$ (i.e., all samples from the uniform sampler), the performance of the algorithm is the same as the conventional path planner. Through a number of simulations, we find that $\alpha = 0.5$ offers the best balance between ensuring full coverage of the environment and leveraging the learned sample regions. Generally, the nonuniform sampler can quickly find a feasible path with a few samples. However, if the nonuniform sampler does not completely identify the solution region, the uniform sampler needs to effectively fill in the gaps, i.e., the nonuniform sampler needs many more samples so that it can find these regions.

The details of the NRRT* is shown in Algorithm 2. First, We initialize a tree $\mathcal{T} = (V, E)$ consisting of a vertex set $V \subset \mathcal{X}_{\text{free}}$ and edge set $E \subseteq V \times V$. Then, in terms of the current map and parameter settings, **NeuralModel**$(Map, \mathcal{S}, \mathcal{C})$ outputs the predicted sampling distribution $\mathcal{O}$. We find that the

value of the state in the predicted sampling distribution $\mathcal{O}$ is polarized, i.e., the value of the state located on the predicted optimal path approaches 1 while the value of the other states approaches 0. Therefore, a simple rejection sampling method is implemented where the value of the state less than 0.5 is rejected during the nonuniform sampling process. In the whole sampling process, a random number **Rand**() $\in (0, 1)$ is used to determine which sampling method to use. If **Rand**() $>$ 0.5, the nonuniform sampler is used. Otherwise, the uniform sampler will be used. The following procedure is similar to the pipeline of the RRT* [7]. If the newest sample belongs to the goal region, i.e., $x_{\text{new}} \in \mathcal{G}(x_{\text{goal}})$, the algorithm will return a tree $\mathcal{T}$ consisting of a feasible path σ connecting x_{init} and $\mathcal{G}(x_{\text{goal}})$. Or the number of iterations reaches the threshold, the algorithm will return $failure$.

It is noted that the difference between the NRRT* and the RRT* is the sampling technique, which can be easily used in other sampling-based algorithms.

D. Probabilistic Completeness and Asymptotic Optimality

In the NRRT*, the nonuniform sampler samples states from the promising region with the probability $1 - \alpha$ while the uniform sampler samples states from the whole state space with the probability α. It means that as the number of iterations goes to infinity, each state in the state space will be sampled. Therefore, the probabilistic completeness is naturally guaranteed.

In reference to the theorem 38 in [7], the theoretical guarantee of the asymptotic optimality of the RRT* algorithm holds if the following inequation holds:

$$\gamma > (2(1 + 1/d))^{1/d} \left(\frac{\mu(\mathcal{X}_{\text{free}})}{\xi_d} \right)^{1/d} \tag{6}$$

where γ represents the search radius in **Extend**$(\mathcal{T}, x_{\text{new}})$, d the dimensionality of the state space, $\mu(\mathcal{X}_{\text{free}})$ the Lebesgue measure of $\mathcal{X}_{\text{free}}$, and ξ_d the volume of unit ball in the d-dimensional Euclidean space. As the same procedures happen in the RRT*, it can be stated that the NRRT* is also asymptotically optimal according to Lemma 56, 71, and 72 in [7].

IV. Simulation Results

In this section, we demonstrate the performance and generality of the NRRT* through several numerical simulations. In Section IV-A, we describe the details of network training. In the following, we compare the NRRT* with two state-of-the-art algorithms, the RRT* and the IRRT* under different clearance and different step sizes. Because the map size does not affect the distribution of the optimal path, the simulation under different map size is not provided.

A. Details of Network Training

The neural network experiments are conducted on NVIDIA Quadro P5000 with TensorFlow. For training, we use the Adam optimizer [30] with its suggested default parameters of $\beta 1 = 0.9$ and $\beta 2 = 0.999$ along with a batch size of 20 training pairs. The learning rate starts at $\epsilon = 0.0001$ and is consistent across the experiments. It is noted that the proposed neural network only takes 50 ms to output a predicted probability distribution of the optimal path whatever the map size, the clearance or the step size. It saves a lot of time cost compared with those which use the A* algorithm to find a prior heuristic path as we have mentioned in Fig. 1.

TABLE I
PATH PLANNING DATA SET

	c1	*c2*	*c4*	*c6*
s1	5576 * 12	5576 * 12	5576 * 12	5576 * 12
s2	5576 * 12	5576 * 12	5576 * 12	5576 * 12
s4	5576 * 12	5576 * 12	5576 * 12	5576 * 12
s6	5576 * 12	5576 * 12	5576 * 12	5576 * 12

To obtain a reasonable and generalized data set, we first generate 5576 different 2-D random maps. The size of each map is 201×201 pixels. As shown in Table I, for each map with corresponding parameter settings ($c1$ means that the clearance is 1 and $s1$ means that the step size is 1), we randomly set 12 different start and goal states. So there are 10 70 592 images in the training data set. We use the A* algorithm [3] to find the optimal path.

The width of the optimal path is one pixel. In order to make the learning easier, we widen the optimal path by one pixel on each side, which means that the final width of the optimal path is three pixels.

B. Path Planning Under Different Clearance

In this section, we test the performance of three algorithms on the path planning problem under different clearance. Four kinds of clearance $(1, 2, 4, 6)$ are used. The input of the step size to the neural network is always 2. A larger clearance means that the planned path is far away from the obstacles while a smaller clearance means that the planned path is close to the obstacles. For the RRT* and the IRRT*, the clearance is a predefined parameter. However, in the NRRT*, the neural network model outputs the corresponding sampling distribution to feed the nonuniform sampler according to different clearance settings.

Fig. 4(a)–(h) shows the illustrations of the predicted sampling distribution for two different maps [Fig. 4(a)–(d) for Map 1 and Fig. 4(e)–(h) for Map 2] under different clearance settings. The red and green rectangles denote the start and goal states, respectively. The magenta indicates that the corresponding region has a higher potential to contain the optimal path and the blue indicates a lower potential. It can be seen that the optimal path lies in the predicted probability distribution. Therefore, using the prediction from the neural network as a nonuniform sampler, the NRRT* can quickly find an initial solution and converge to the optimal solution. However, it is noted that some prediction results are not perfect. For example, in Fig. 4(b) and (f), some regions where the optimal solution cannot lie in are considered as part of the prediction result. In Fig. 4(c), (f), and (g), the predicted sampling regions are not continuous. Through a series of simulations, we find that these imperfect predictions have little effect on the performance

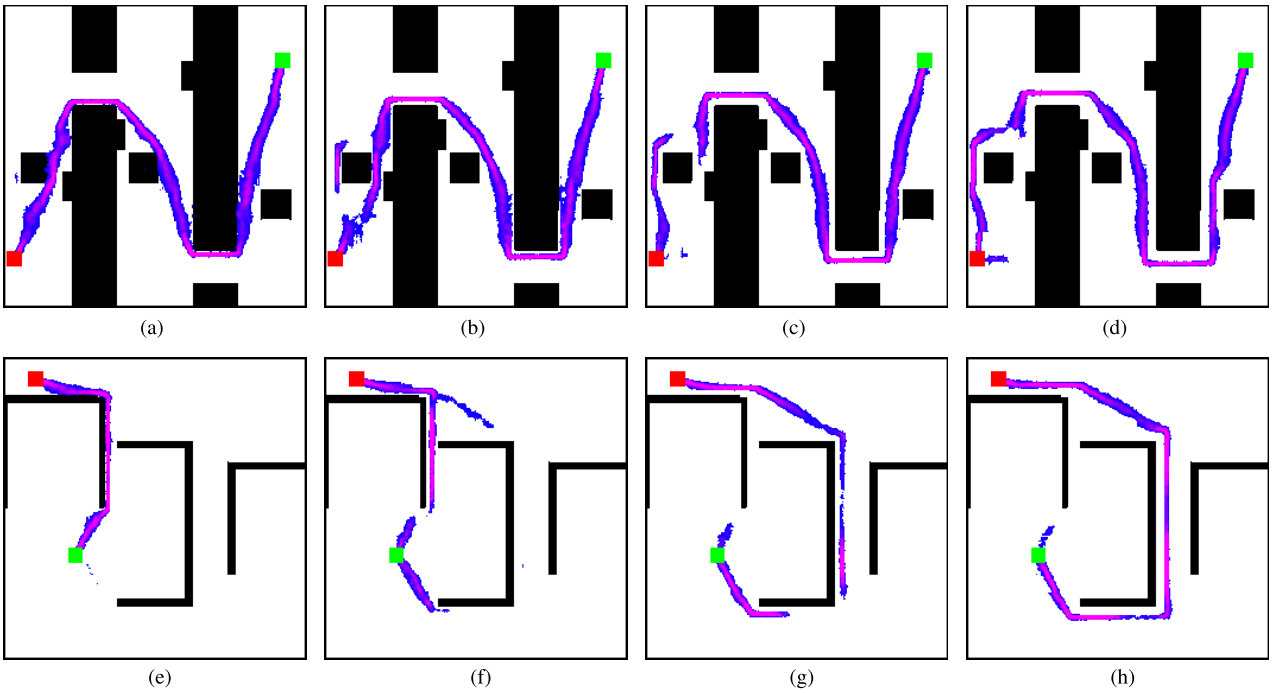

Fig. 4. Predicted sampling distribution under different clearance settings. The magenta indicates that the corresponding region has a higher potential to contain the optimal path and the blue indicates a lower potential. (a) Map 1, C1. (b) Map 1, C2. (c) Map 1, C4. (d) Map 1, C6. (e) Map 2, C1. (f) Map 2, C2. (g) Map 2, C4. (h) Map 2, C6.

of the algorithm. The reason is that RRTs algorithms have the ability to explore the whole state space. For example, in Fig. 4(c), beginning from the start state, the tree grows to the end of the left part of the predicted probability distribution. When the state is sampled from the right part of the predicted probability distribution, the tree can gradually grow to the right. When the tree arrives at the right part of the predicted probability distribution, it can quickly grow toward the goal state. This is also the reason that we use both the uniform sampler and nonuniform sampler in the path planning process, as mentioned in Section III-C.

In addition, we can see that the prediction changes a lot for different clearance settings, especially from Fig. 4(b), (c), (f), and (g). The reason is that the optimal path actually changes under different clearance settings. When the required clearance becomes larger, the feasible path will be far away from the obstacles and the optimal path changes accordingly. So the predictions for different clearance settings reveal that the proposed neural network is capable of dealing with the clearance requirement in the path planning process.

Two different maps (Map 1 and Map 2) are used to test the performance of the RRT*, the IRRT* and the NRRT*. The simulation results under different clearance settings are provided in Fig. 5. The "Time" denotes the sum of CPU time cost and GPU time cost in the path planning process since the GPU is used to execute the CNN. The "Node" denotes the number of the node when the optimal solution is found. The number of the node indicates the memory usage in the path planning process because different algorithms use almost the same memory usage in other places. Fig. 5(a), (b) and (c), (d) shows the performance of three algorithms

TABLE II
COMPARISON OF ALGORITHM PERFORMANCE

	Time vs RRT*	Time vs IRRT*	Node vs RRT*	Node vs IRRT*
Map 1, *c1*	86.5%	81.5%	81.8%	77.2%
Map 1, *c2*	85.2%	81.1%	78.2%	74.7%
Map 1, *c4*	83.3%	83.3%	76.1%	72.3%
Map 1, *c6*	83.0%	80.8%	75.1%	73.9%
Map 2, *c1*	66.4%	57.8%	91.2%	88.8%
Map 2, *c2*	80.3%	56.1%	93.1%	87.8%
Map 2, *c4*	49.2%	50.1%	65.3%	64.4%
Map 2, *c6*	78.8%	75.5%	81.6%	79.8%

on finding the optimal path in Map 1 and Map 2, respectively. First, the NRRT* uses the least time cost and nodes to find the optimal path. It means that the NRRT* can save a lot of computation resources compared with the RRT* and the IRRT*. A statistical result is provided in Table II to show the performance improvement compared with the RRT* and the IRRT*. Second, the standard deviation of the time cost and nodes of the NRRT* are both much smaller, which reveals the better stability of the NRRT*. Third, as the clearance changes, the RRT* and the IRRT* requires different time cost and nodes to find the optimal path. But the time cost and nodes used in the NRRT* do not change a lot, which shows the good robustness of the NRRT*. The simulation results demonstrate that the predicted sampling distribution significantly improves the algorithm performance. In fact, the nonuniform sampling technique provides a strong bias to the optimal path and the initial path generated by the NRRT* is very close to the optimal path. Table III shows the length of the initial

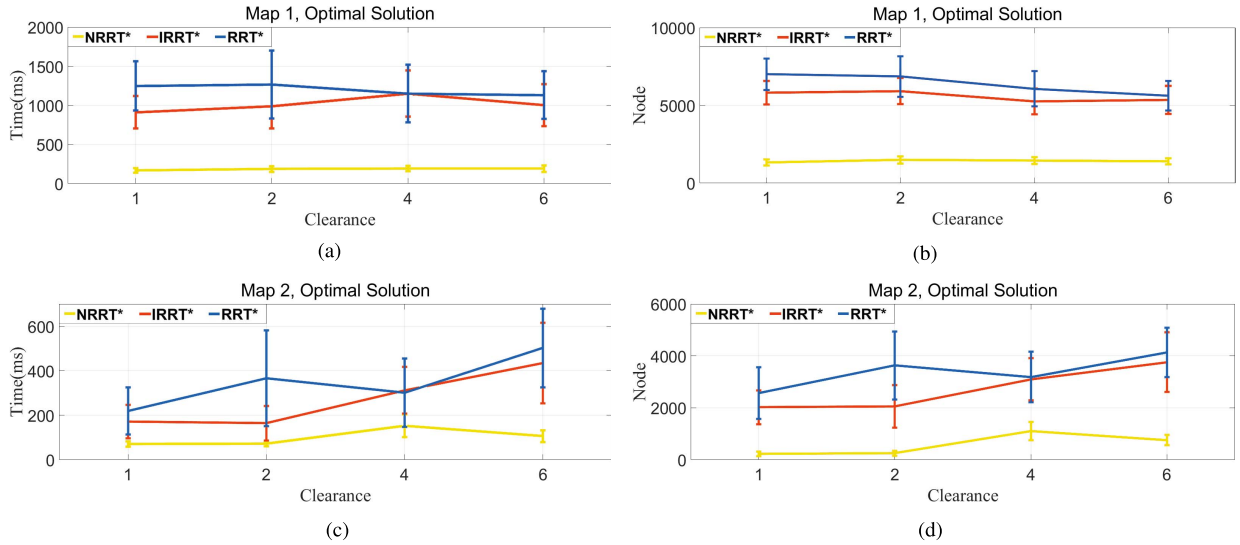

Fig. 5. Simulation results under different clearance settings. (a), (b) and (c), (d) show the performance of three algorithms on finding the optimal path on two different maps, respectively. The time cost and the number of node are selected as the metric to evaluate the performance.

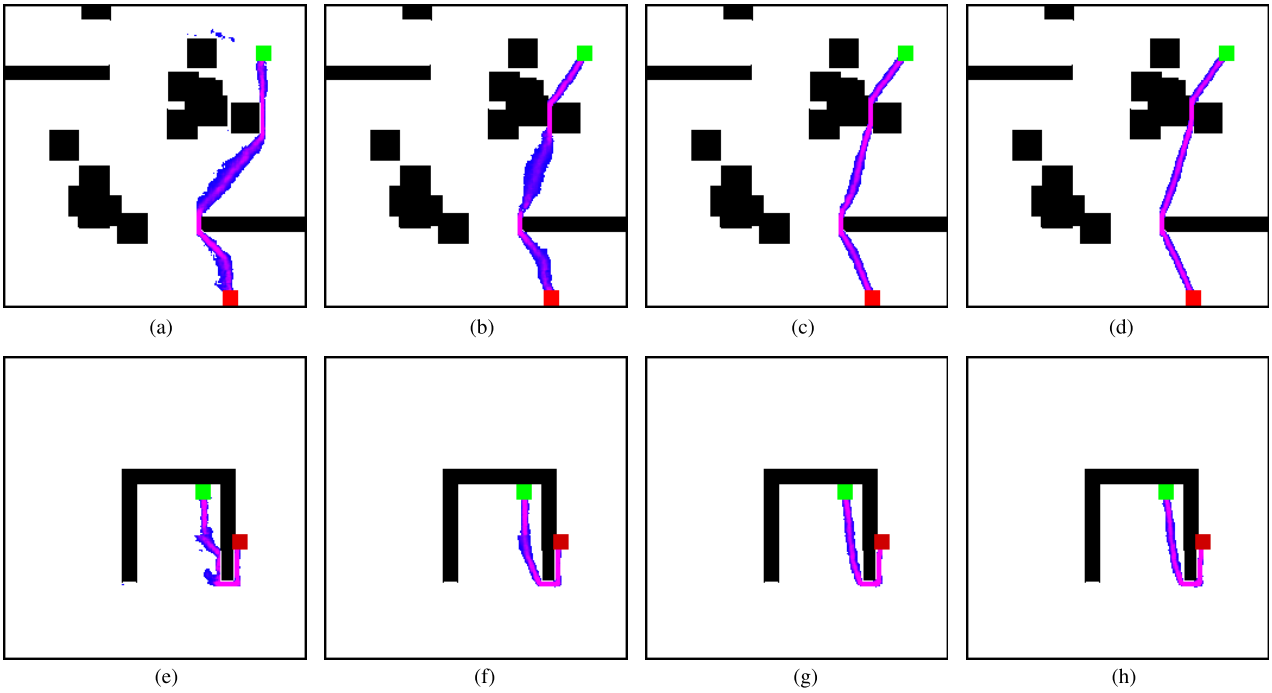

Fig. 6. Predicted sampling distribution under different step size settings. The magenta indicates that the corresponding region has a higher potential to contain the optimal path and the blue indicates a lower potential. (a) Map 3, S1. (b) Map 3, S2. (c) Map 3, S4. (d) Map 3, S6. (e) Map 4, S1. (f) Map 4, S2. (g) Map 4, S4. (h) Map 4, S6.

path generated from three algorithms. We can see that under different clearance settings, the NRRT* always finds a good initial path. It is noted that in Map 2, when the clearance is set to 4, the algorithm performance is not very good because there are many discontinuous parts in the predicted probability distribution of the optimal path. In this case, if the value of α mentioned in Section III-C increases, the performance will be better. So it is still an open problem to measure the quality of the prediction result and choose the appropriate parameter.

In short, the NRRT* achieves better performance compared with the RRT* and the IRRT*. The reason is that the NRRT* uses a nonuniform sampler to bias samples to the predicted sampling region. This region generated from the neural network contains the optimal path with a large probability, so such a biased sampling process naturally achieves better results.

C. Path Planning Under Different Step Size

In this section, we test the performance of three algorithms on the path planning problem under different step sizes.

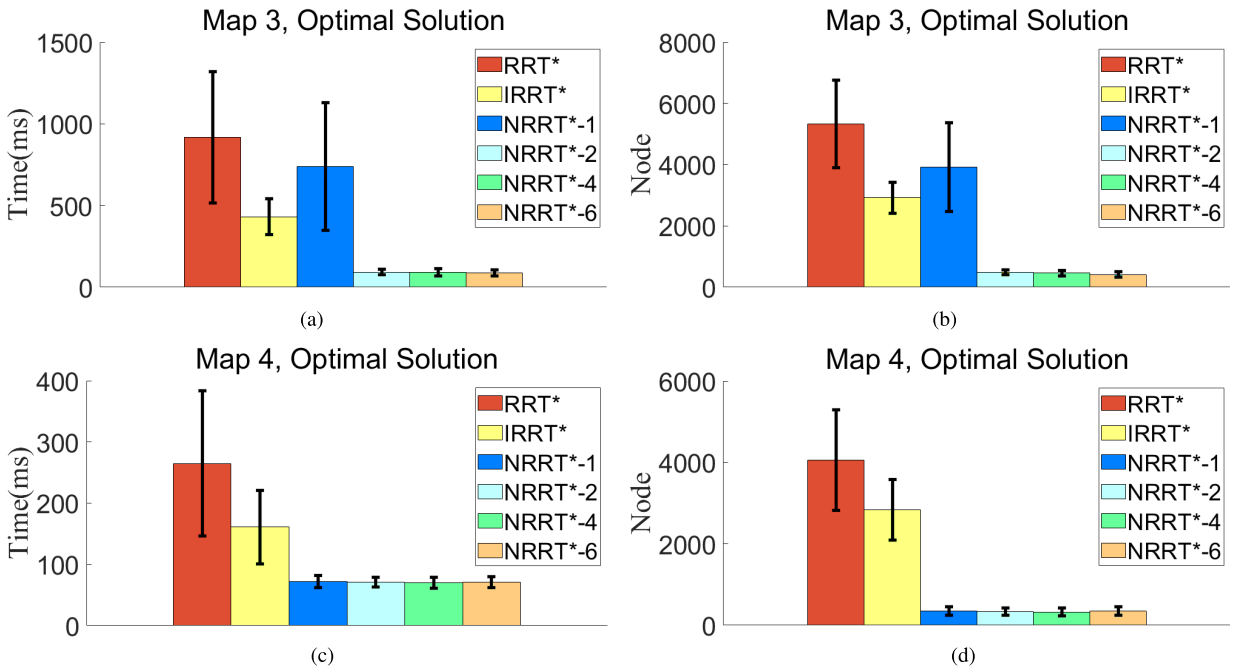

Fig. 7. Simulation results under different step size settings. (a), (b) and (c), (d) show the performance of three algorithms on finding the optimal path on two different maps, respectively. The time cost and the number of node are selected as the metric to evaluate the performance.

TABLE III
INITIAL PATH LENGTH UNDER DIFFERENT CLEARANCE

	Opt.Sol.	RRT*	IRRT*	NRRT*
Map 1, *c1*	1600	1785 ± 89	1777 ± 83	**1604 ± 28**
Map 1, *c2*	1630	1803 ± 79	1798 ± 82	**1635 ± 17**
Map 1, *c4*	1700	1853 ± 62	1853 ± 64	**1713 ± 9**
Map 1, *c6*	1760	1861 ± 44	1866 ± 39	**1770 ± 11**
Map 2, *c1*	625	744 ± 223	749 ± 214	**612 ± 6**
Map 2, *c2*	625	774 ± 281	793 ± 307	**632 ± 15**
Map 2, *c4*	1280	1397 ± 70	1384 ± 69	**1289 ± 13**
Map 2, *c6*	1280	1389 ± 75	1403 ± 79	**1276 ± 11**

Four kinds of step size $(1, 2, 4, 6)$ are used. The input of the clearance to the neural network is always 1. The step size means the search range in the A* algorithm; for example, step size 2 means that the A* planner will search two grids in each direction. Different step size settings in the A* algorithm lead to different optimal paths. Since the optimal paths used in the training data set are from the A* algorithm, the NRRT* is also sensitive to the step size.

Fig. 6(a)–(h) shows the illustrations of the predicted sampling distribution of two maps (Maps 3 and 4) under different step size settings. The red and green rectangles denote the start and goal states, respectively. The magenta indicates that the corresponding region has a higher potential to contain the optimal path and the blue indicates a lower potential. Again, we find that the predicted region contains the optimal path. It is noted that the prediction in Fig. 6(a) is obviously different from those in Fig. 6(b)–(d). It is because the optimal path under step size 1 is different from those under step size $2, 4, 6$, which results in a different prediction result. This is also a limitation of the A* algorithm. If the step size is set to a small value, the final path is not optimal compared with the final path under a large step size setting. The following simulation results will also support this argument. However, if the step size is set to a large value, the time cost of the A* algorithm increases exponentially. But the NRRT* always quickly predicts the sampling distribution under different step size settings.

The simulation results under different step size settings are provided in Fig. 7. The NRRT*-1 means that the NRRT* uses the predicted sampling distribution generated from the neural network under step size 1. For Fig. 6(a)–(d), the NRRT*-2, the NRRT*-4, and the NRRT*-6 use much less time cost and nodes to find the optimal path because the generated initial path is already very close to the optimal path. However, the predicted sampling distribution of the NRRT*-1 does not contain the optimal path and the nonuniform sampler cannot contribute a lot to finding the optimal path. This is also the big limitation of a series of algorithms that use the A* algorithm to find a heuristic path. A large step size means that the preprocessing consumes a lot of time while a small step size cannot guarantee a good heuristic path. However, in the NRRT*, we can set a larger step size to guarantee a good heuristic path without extra time cost. We can also find that the IRRT* performs better than the RRT* and the NRRT*-1 on finding the optimal path in the Map 3. This is because the IRRT* uses an admissible ellipsoidal heuristic to optimize the initial path. The details can be found in [17]. For Fig. 6(e)–(h), the simulation results shown in Fig. 7(c) and (d) demonstrate that the NRRT* can quickly find the optimal path in a "trap" environment compared with the RRT* and the IRRT*.

In summary, the NRRT* achieves convincing performance again under different step size settings.

V. Conclusion and Future Work

In this article, based on the CNN model and the RRT* algorithm, we propose the NRRT* to achieve nonuniform sampling in the path planning process by learning quantities of successful planning cases from the A* algorithm. To satisfy the different constraints in the path planning, the clearance and step size are both considered in the designed CNN model. The simulation results show that the proposed algorithm performs much better in comparison with conventional algorithms. In fact, the proposed algorithm is a novel sampling technique and can be readily applied to other sampling-based algorithms for improved results.

There are also many possible avenues for future work. First, the semantic information of the environment can be taken into account to better understand the task specification and interact with humans, which is also called semantic-aware path planning [31]. Second, a challenging extension is how to directly encode the environment using raw point-cloud data. It is crucial for achieving real-time optimal path planning while satisfying kinodynamic constraints [24] in practice. Furthermore, an interesting topic is to achieve socially compliant path planning [32] with a neural network model.

References

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

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

[3] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 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*. Boston, MA, USA: Springer, 1997, pp. 203–220.

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

[6] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, Jul. 2016.

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

[8] M. Brunner, B. Bruggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5539–5544.

[9] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[10] K. Bekris and L. Kavraki, "Informed and probabilistically complete search for motion planning under differential constraints," in *Proc. 1st Int. Symp. Search Techn. Artif. Intell. Robot. (STAIR)*, Chicago, IL USA, 2008, pp. 1–8.

[11] M. Rickert, A. Sieverling, and O. Brock, "Balancing exploration and exploitation in sampling-based motion planning," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1305–1317, Dec. 2014.

[12] L. Palmieri, S. Koenig, and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 2775–2781.

[13] A. H. Qureshi and Y. Ayaz, "Potential functions based sampling heuristic for optimal path planning," *Auton. Robots*, vol. 40, no. 6, pp. 1079–1093, Nov. 2015.

[14] J. Wang, W. Chi, M. Shao, and M. Q.-H. Meng, "Finding a high-quality initial solution for the RRTs algorithms in 2D environments," *Robotica*, vol. 37, no. 10, pp. 1677–1694, Feb. 2019.

[15] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 3856–3861.

[16] J. Wang, X. Li, and M. Q.-H. Meng, "An improved RRT algorithm incorporating obstacle boundary information," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2016, pp. 625–630.

[17] 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. Robots Syst.*, Sep. 2014, pp. 2997–3004.

[18] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 2, Oct. 2003, pp. 1178–1183.

[19] J. Wang and M. Q.-H. Meng, "Optimal path planning using generalized Voronoi graph and multiple potential functions," *IEEE Trans. Ind. Electron.*, to be published.

[20] I. Baldwin and P. Newman, "Non-parametric learning for natural plan generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 4311–4317.

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

[22] A. H. Qureshi and M. C. Yip, "Deeply informed neural sampling for robot motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 6582–6588.

[23] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning," 2018, *arXiv:1806.01968*. [Online]. Available: http://arxiv.org/abs/1806.01968

[24] Y. Li, R. Cui, Z. Li, and D. Xu, "Neural network approximation based near-optimal motion planning with kinodynamic constraints using RRT," *IEEE Trans. Ind. Electron.*, vol. 65, no. 11, pp. 8718–8729, Nov. 2018.

[25] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[27] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 628–644.

[28] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*. [Online]. Available: http://arxiv.org/abs/1706.05587

[29] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[30] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015, pp. 1–15.

[31] C. Wang, J. Cheng, W. Chi, T. Yan, and M. Q.-H. Meng, "Semantic-aware informative path planning for efficient object search using mobile robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[32] J. Wang and M. Q.-H. Meng, "Socially compliant path planning for robotic autonomous luggage trolley collection at airports," *Sensors*, vol. 19, no. 12, p. 2759, Jun. 2019.

Jiankun Wang 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.

During his Ph.D. degree, he spent six months with Stanford University, Stanford, CA, USA, as a Visiting Student Scholar supervised by Prof. Oussama Khatib. He is currently a Post-Doctoral Fellow with the Department of Electronic Engineering, The Chinese University of Hong Kong. His current research interests include motion planning and control, human–robot interaction, and machine learning in robotics.

Wenzheng Chi received the B.E. degree in automation from Shandong University, Jinan, China, in 2013, and the Ph.D. degree in biomedical engineering from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, in 2017.

During her Ph.D. degree, she spent six months with The University of Tokyo, Tokyo, Japan, as a Visiting Student Scholar. She was a Post-Doctoral Fellow with the Department of Electronic Engineering, The Chinese University of Hong Kong, from 2017 to 2018. She is currently an Associate Professor with the Robotics and Microsystems Center, School of Mechanical and Electric Engineering, Soochow University, Suzhou, China. Her research interests include human recognition, human–robot interaction, and human-friendly motion planning for mobile service robots.

Chenming Li received the B.E. degree in petroleum engineering from the China University of Petroleum, Qingdao, China, in 2017, and the M.Sc. degree in electronic engineering from The Chinese University of Hong Kong, Hong Kong, in 2018, where he is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, The Chinese University of Hong Kong.

His current research interests include indoor localization, and simultaneous localization and mapping.

Chaoqun Wang 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 2019.

He is currently a Post-Doctoral Fellow with the Department of Electronic Engineering, The Chinese University of Hong Kong. He was with The University of British Columbia, Vancouver, BC, Canada, as a Visiting Scholar for six months. His current research interests include autonomous exploration, active perception, and path planning.

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

He has been a Professor of electronic engineering with the Chinese University of Hong Kong, Hong Kong, since 2002, after working for ten years with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, as the Director of the Advanced Robotics and Teleoperation Laboratory and holding the positions of an Assistant Professor in 1994, Associate Professor in 1998, and Professor in 2000. 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 serving as an Elected Member of the Administrative Committee of the IEEE Robotics and Automation Society. He received the IEEE Third Millennium Medal Award.