Conditional Generative Adversarial Networks for Optimal Path Planning

Nachuan Ma, Jiankun Wang, *Member, IEEE*, Jianbang Liu, and Max Q.-H. Meng, *Fellow, IEEE*

***Abstract*—Path planning plays an important role in autonomous robot systems. Effective understanding of the surrounding environment and efficient generation of an optimal collision-free path are both critical parts for solving path-planning problems. Although conventional sampling-based algorithms, such as the rapidly exploring random tree (RRT) and its improved optimal version (RRT*), have been widely used in path-planning problems because of their ability to find a feasible path in even complex environments, they fail to find an optimal path efficiently. To solve this problem and satisfy the two aforementioned requirements, we propose a novel learning-based path-planning algorithm which consists of a novel generative model based on the conditional generative adversarial networks (CGANs) and a modified RRT* algorithm (denoted by CGAN-RRT*). Given the map information, our CGAN model can generate an efficient probability distribution of feasible paths, which can be utilized by the CGAN-RRT* algorithm to find an optimal path with a nonuniform sampling strategy. The CGAN model is trained by learning from ground-truth maps, each of which is generated by putting all the results of executing the RRT algorithm 50 times on one raw map. We demonstrate the efficient performance of this CGAN model by testing it on two groups of maps and comparing the CGAN-RRT* algorithm with the Informed-RRT* algorithm and conventional RRT* algorithm.**

***Index Terms*—Conditional generative adversarial networks (CGANs), optimal path planning, sampling-based path planning.**

Manuscript received November 4, 2020; revised January 31, 2021; accepted February 23, 2021. Date of publication March 2, 2021; date of current version June 10, 2022. This work was supported in part by the Shenzhen Key Laboratory of Robotics Perception and Intelligence, Southern University of Science and Technology, Shenzhen, China; in part by the Shenzhen Science and Technology Innovation projects under Grant JCYJ20170413161616163; in part by Hong Kong ITC ITSP Tier 2 under Grant ITS/105/18FP; and in part by Hong Kong ITC MRP under Grant MRP/011/18. The work of Max Q.-H. Meng was supported by Hong Kong RGC GRF under Grant 14200618. *(Nachuan Ma and Jiankun Wang contributed equally to this work.) (Corresponding author: Nachuan Ma.)*

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

Jianbang Liu is with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: henryliu@link.cuhk.edu.hk).

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, Chinese University of Hong Kong, Shenzhen 518057, 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/TCDS.2021.3063273.

Digital Object Identifier 10.1109/TCDS.2021.3063273

I. Introduction

PATH planning is an essential component for autonomous mobile robots. Its performance directly determines the success rate of robot tasks and the robustness of robot systems. The objective of the path-planning problem is to generate a collision-free path for robots from an initial state to a goal state while avoiding a number of moving or static obstacles. Over the past few decades, researchers have proposed a lot of path-planning algorithms, such as artificial potential field method (APF) [1], probabilistic roadmap method (PRM) [2], RRT [3], A* algorithm [4], and so on. By utilizing a virtual force method, the APF transforms the movement of robots in the environmental space into the virtual potential field to guide the movement of robots. However, it suffers from easily falling into a local minimum. The A* algorithm adopts a heuristic function and computes its value at each node within the configuration space to obtain an optimal path. But it tends to consume a lot of time and huge memory usage with the increase of the size of configuration space. The RRT algorithm iteratively constructs trees to connect samples drawn from the whole space in a short time. Then, a feasible path can be found by traversing the tree from the initial state to the goal state. The PRM algorithm randomly samples points to establish a graph which can be solved by a typical graph search algorithm. It utilizes a local planner to connect random samples from free space.

In the aforementioned path-planning algorithms, sampling-based algorithms, such as the RRT and PRM algorithms, are widely applied to autonomous mobile robots because of their probabilistic completeness and good scalability. Besides, sampling-based path-planning algorithms do not require an elaborate modeling of the environment. However, both RRT and PRM algorithms cannot guarantee an optimal path. To make the initial path converge to an optimal path, the RRT* algorithm [5] is proposed. It can be viewed as a significant improvement over the RRT algorithm. But the RRT* algorithm converges slowly to an optimal path because planners draw random samples from a uniform distribution. It is also constrained by the quality of the initial path. The existing RRT* algorithm cannot solve a path-planning problem quickly, especially when encountering challenging environments such as an environment containing narrow passages or many turns.

To overcome the limitations of the RRT* algorithm, researchers have proposed some learning-based path-planning algorithms which change the sampling strategy of the RRT* algorithm [6]–[8]. Inspired by these improved sampling-based algorithms, this article aims to construct a neural network

2379-8920 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.

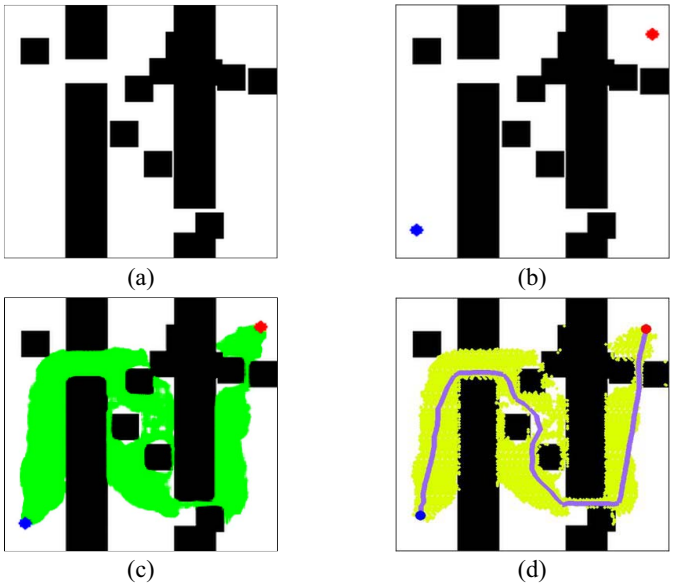


Fig. 1. Four stages of the proposed CGAN-RRT* algorithm. (a) Raw map. (b) Map(a) added randomly assigned initial state and goal state. (c) Map(b) added predicted probability distribution of feasible paths generated from the CGAN model. (d) Map(c) added an optimal path generated from the CGAN-RRT* algorithm.

model that can quickly predict the probability distribution of feasible paths on a map. And the predicted probability distribution is utilized to guide the sampling process of the RRT* algorithm. The problem can be viewed as "translating" an input image into a corresponding output image. In the field of computer vision, researchers have already achieved significant progress in the problem of predicting pixels to pixels with convolutional neural networks (CNNs) [9]. CNN models learn to minimize a loss function of the Euclidean distance between predicted and ground-truth pixels. However, it will tend to generate blurry results because the Euclidean distance is minimized by averaging all plausible outputs. To make the output indistinguishable from the reality, the generative adversarial networks (GANs) [10] can be applied to this problem. GAN models train a generator to generate outputs that cannot be distinguished from the real images by a synchronously training discriminator which tries to detect fake maps. For image-to-image prediction tasks, GAN can be modified in the conditional setting. The CGAN condition on an input image and generate a corresponding output image. CGAN models are more suitable for the path-planning problem than GAN models because they can direct the data generation process under certain conditions. CGAN models can be utilized to generate the predicted probability distribution of feasible paths on given maps.

In this article, we propose a novel learning-based path-planning algorithm that trains a CGAN model to obtain the predicted probability distribution of feasible paths. The CGAN model is trained by learning from 12 000 pairs of input environment maps and corresponding ground-truth maps. The input environment maps contain random initial states and goal states, while the ground-truth maps contain 50 feasible paths generated from the RRT algorithm. For a given path-planning problem, the proposed CGAN model can quickly predict the probability distribution of feasible paths on the map. Then, the predicted probability distribution is utilized to guide the sampling process of the RRT* algorithm. Four stages of the proposed CGAN-RRT* algorithm are shown in Fig. 1. The results of the model are promising and show that it can improve the performance of the RRT* algorithm significantly. This new model can also be applied to other sampling-based algorithms in future researches.

To summarize, the key contributions of this article include the following.

1) A CGAN model to predict the probability distribution of the feasible paths for different types of maps.
2) A novel optimal path-planning algorithm which combines a CGAN model and the RRT* algorithm, denoted as the CGAN-RRT*.
3) Case studies to demonstrate the effectiveness and efficiency of the proposed CGAN-RRT* algorithm.

The remainder of this article is organized as follows. We introduce some related work concerning the GAN model and RRT algorithms in Section II. In Section III, we formulate the path-planning problem and explain the RRT and RRT* algorithms. Then, Section IV presents the details of the proposed CGAN-RRT* algorithm. A number of simulation experiments are conducted in Section V to compare the performance of CGAN-RRT* with that of RRT*. Finally, Section VII concludes this article and discusses future work.

II. Related Work

A. Improved Path-Planning Algorithms

For decades, path planning has been a very active topic in robotics research. There are many well-established algorithms. As some general path-planning techniques have been discussed in Section I, we will focus on improved algorithms based on the RRT algorithm.

In [11], The exploring/exploiting tree (EET) is proposed by balancing exploitation and exploration during path planning. Although it improves the computational efficiency of conventional sampling-based algorithms, it lacks probabilistic completeness. Yershove *et al.* [12] proposed the dynamic-domain RRT algorithm, which improves the performance of the RRT algorithm in several motion planning problems. It adopts a new sampling strategy that limits the expansion of nodes near obstacles during the sampling process. However, the dynamic-domain RRT algorithm is not convenient since a new parameter of it needs to be tuned carefully. To automatically tune the new parameter, an extension of the dynamic-domain RRT algorithm is proposed in [13]. It improves the performance of the dynamic-domain RRT algorithm by adapting the region of influence of each node. By utilizing the initial path generated from the A* algorithm to guide the sampling process of the RRT* planner, Brunner *et al.* [14] proposed the A*-RRT* algorithm. It improves the convergence speed of the RRT* algorithm. However, both algorithms in [13] and [14] suffer from consuming a lot of time when the size of configuration space increases.

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.

Recently, deep learning and reinforcement learning methods have been applied in robotic path-planning problems. Reinforcement learning methods have achieved good performance in decision-making problems, and deep learning methods are suitable for image recognition. By modifying the coefficient of the Bellman equation, Zhang *et al.* [15] proposed an improved method to accelerate the convergence of the Q-learning algorithm in path-planning tasks. Some scholars obtain good path-planning results by combining deep learning methods and reinforcement learning methods. Dubey *et al.* [16] utilized images generated by the Q-learning algorithm as the training data. Then, the training data is used as input to a deep-learning neural network to learn a path. However, both algorithms in [15] and [16] are constrained by the low convergence speed and a lot of time cost of the Q-learning algorithm. Zhu *et al.* [17] focused on path planning of an autonomous underwater vehicle (AUV) system with a bioinspired neural network to realize obstacle avoidance. Then, Chen and Zhu [18] extended this idea to solve the task assignment and path-planning problem of the multi-AUV system. In [19], the conditional variational autoencoder (CVAE) algorithm for robot motion planning is proposed. It utilizes a nonuniform sampling strategy which learns from demonstrations of successful motion plans to guide the sampling process. By implementing a CNN model, Wang *et al.* [7] proposed the Neural-RRT* algorithm. It utilizes the predicted probability distribution generated from the CNN model to guide the sampling process of the RRT* algorithm. Though the Neural-RRT* algorithm achieves better performance than the informed RRT* and RRT* algorithms, the predicted sampling regions generated from the CNN model may be discontinuous sometimes. In [20], a brain-inspired cognitive model with attention is proposed for self-driving cars. The brain-inspired cognitive model combines a recurrent neural network (RNN) with the real-time updated cognitive map to detect the free space and estimate the distances to obstacles.

B. Conditional GAN

In recent years, applying GAN in the conditional setting has gained a lot of attention in the computer vision field. Mirza and Osindero [21] introduced a condition version of GAN. The model can generate MNIST digests conditioned on class labels. Gauthier [22] further applied the CGAN in face recognition. Positive results have been obtained in utilizing the combined conditional data to control particular face attributes from the model. Denton *et al.* [23] presented a Laplacian pyramid of adversarial networks to produce high-quality samples of natural images. The model utilizes a cascade of convolutional networks within a Laplacian pyramid framework. In [24], the style and structure adversarial network (S^2-GAN) is presented. The model utilizes the structure-GAN to generate a surface normal map. Then, the style-GAN translates the normal map into the corresponding 2-D image. Mathieu *et al.* [25] evaluated different loss functions and illustrated that generative adversarial training can be effective for future frame prediction. For product photograph generation problem, Yoo *et al.* [26] proposed a pixel-level domain converter. It translates the information in the source domain into a pixel-level image while preserving the semantic meaning.

To the best of our knowledge, there exist few applications for the path-planning problem by utilizing GAN. Different from the aforementioned path-planning algorithms, in this article, we propose a CGAN model to quickly predict the probabilistic distribution of feasible paths on the map. Our method also differs from the aforementioned CGAN model works in the architectural choice. A "U-net"-based architecture is utilized in our generator [27], which is a popular version of GAN in image-to-image tasks. We show that this approach is effective in the path-planning problem by experimental simulations in Section V.

III. Preliminaries

This section starts with the formulation of the path-planning problem in Section III-A. Then, an explanation of the RRT and RRT* algorithms is presented in Section III-B.

A. Path-Planning Problem and Related Terminologies

In this section, the basic path-planning problem is formulated. We denote $\mathcal{X} \in \mathbb{R}^d$ as the state space. Let $\mathcal{X}_{\text{free}}$ be the free space, and $\mathcal{X}_{\text{obs}} = \mathcal{X}/\mathcal{X}_{\text{free}}$ is denoted as the obstacle space. We denote x_{init} and x_{goal} as the initial state and the goal state, respectively. They both belong to $\mathcal{X}_{\text{free}}$. In addition, $\mathcal{X}_{\text{goal}} \in \mathcal{X}_{\text{free}}$ denotes the goal region, where $\mathcal{X}_{\text{goal}} = \{x \in \mathcal{X}_{\text{free}}, \|x - x_{\text{goal}}\| < r\}$, and r is a positive real number. Then, the path-planning problem can be expressed as to generate a feasible path $\delta(t) \in \mathcal{X}_{\text{free}}$ for $t \in [0, t_g]$ that satisfies the robot dynamics constraints. The path starts at the initial state $\delta(0) = x_{\text{init}}$ and ends at the goal state $\delta(t_g) \in \mathcal{X}_{\text{goal}}$.

We also employ the definition of *optimal path* throughout the rest of this article.

Optimal Path: Given a path-planning problem, the cost function L_c computes the length of a planned path. If we find a feasible path that can minimize the cost function $L_c(\delta(t))$, where $\delta(t) \in \mathcal{X}_{\text{free}}$, we define it as an optimal path $\delta(t)^*$.

The definitions which are utilized in the RRT algorithm are shown as follows.

Samplefree: Sample a random node from the free space $\mathcal{X}_{\text{free}}$.

V: The set of nodes.

E: The set of edges between the nodes.

x_{init}*:* The initial state.

x_{rand}*:* A node sampled randomly at each iteration.

Nearest(V, x): Find the nearest node from V to the node x by utilizing a Euclidean distance.

Steer(x_1, x_2)*:* Steer from x_1 to x_2.

ObstacleFree($\delta(t)$)*:* Determine whether the path $\delta(t)$ is collision-free.

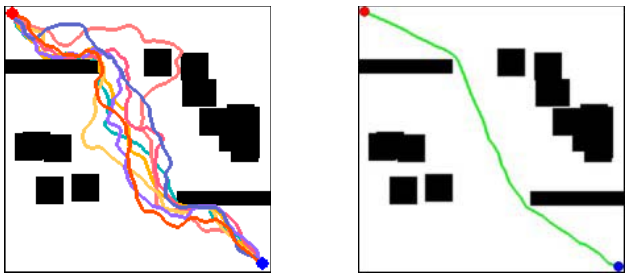
B. RRT and RRT* Algorithms

The pseudocode for the RRT algorithm is presented in Algorithm 1. It is mainly designed for single-query applications and appropriate for dealing with high-dimensional problems. The RRT algorithm randomly expands nodes in the collision-free space and incrementally builds a tree rooted at

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: RRT Algorithm

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$;
for i=1,...,n **do**
  $x_{rand} \leftarrow$ SampleFree$_i$;
  $x_{nearest} \leftarrow$ Nearest(V, x_{rand});
  $x_{new} \leftarrow$ Steer($x_{nearest}$, x_{rand});
  if *ObstacleFree*($x_{nearest}$, x_{new}) **then**
    $V \leftarrow V \cup \{x_{new}\}$; $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$
Return $(G = (V, E))$



(a) (b)

Fig. 2. Comparison between the RRT and RRT* algorithms. (a) Eight paths generated by the RRT algorithm within one map. (b) Optimal path generated by the RRT* algorithm.

the initial state. It begins with an empty edge set E and a vertex set V including the initial state $x_{\text{init}} \in \mathcal{X}_{\text{free}}$. During the execution of the algorithm, a node $x_{\text{rand}} \in \mathcal{X}_{\text{free}}$ is sampled randomly at each iteration. Then, a node from the existing vertex set V which is closest to x_{rand} is selected as x_{nearest}. After that, a steer function is used to adjust x_{rand} to x_{new} and x_{nearest} is connected to x_{new}. If the connection is collision-free, the edge set E will add $(x_{\text{nearest}}, x_{\text{new}})$ and the vertex set V will add the new sampling node x_{new}. Finally, the iteration is stopped when the expanding tree contains a new sampling node in the goal region, and the algorithm returns the edge set E and the vertex set V.

The RRT algorithm is popular for its probability completeness, good scalability, and high efficiency. It can guarantee the feasibility of a solution if there exists a path on the map between the initial state and the goal state. However, though the RRT algorithm can find a feasible path in a short time, it cannot obtain an optimal path. Actually, on the same map, the RRT algorithm tends to generate different paths between the initial state and the goal state as it samples nodes randomly. One example is shown in Fig. 2. The first image shows eight paths generated from the RRT algorithm in the same environment. We can find that some paths have better quality with shorter lengths than other paths. It means that the RRT algorithm performs more efficiently when generating these paths. However, all eight paths are generated randomly from the initial state to the goal state. As a result, the RRT algorithm cannot guarantee the optimal solution and has a large variance. Sometimes it just runs for a short time to generate a feasible path, but it may also run for a long time in some cases.

To generate an optimal path, the RRT* algorithm is proposed. It is a variant of the RRT algorithm and also inherits the merit of the A* algorithm. The main difference from the RRT algorithm is that the RRT* algorithm adds a procedure of *Rewire*. It is an important procedure for the RRT* algorithm to determine which neighboring node has the shortest distance from x_{new}. Although the RRT* algorithm can obtain an optimal path, as shown in Fig. 2(b), it usually consumes a lot of time and huge memory usage. In order to accelerate the convergence speed of the RRT* algorithm, we propose the CGAN-RRT* algorithm.

IV. CGAN-RRT* Algorithm

In this section, we first introduce the objective of our CGAN model in Section IV-A. Then, the structure and loss function of the generator and discriminator is presented in Sections IV-B and IV-C. Finally, we introduce the CGAN-RRT* algorithm in Section IV-D.

A. Objective

GAN models are designed to learn a mapping from random noise z to the output image y. Therefore, the generator of a GAN model is defined as $\{G : z \rightarrow y\}$. It is trained to generate output images that can confuse the discriminator of a GAN model D. D is trained to distinguish the *fake* data produced by G from the *real* data.

The objective function of a GAN model can be expressed as

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_y\big[\log D(y)\big] + \mathbb{E}_z\big[\log(1 - D(G(z)))\big]. \quad (1)$$

The generator is trained to minimize the objective function, while the discriminator is trained to maximize it. Then, the objective function of a GAN model is represented as $\theta^* = \arg\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D)$.

In contrast to GAN models, CGAN models learn a mapping from random noise z to output image y conditioned on the observed image o. The objective function of a CGAN model is expressed as

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{o,y}\big[\log D(o, y)\big] + \mathbb{E}_{o,z}\big[\log(1 - D(o, G(o, z)))\big]. \quad (2)$$

The $L1$ loss is added to encourage less blurring, which is the mean absolute error between the generated image and the target image. Then, the final objective function of our CGAN model is represented as $\theta^* = \arg\min_G \max_D [\mathcal{L}_{\text{cGAN}}(G, D) + \lambda \mathcal{L}_{L1}(G)]$, where λ is a weighting factor. In this article, we set $\lambda = 100$.

For the path generation problem, observed image o represents the input environment map. The *fake* data represents generated maps with the predicted probability distribution of feasible paths produced by the generator. The *real* data represents ground-truth maps.

B. Generator Architectures

The generator aims to translate a high-resolution input environment map to another high-resolution generated map. Both corresponding input environment map and generated map share the same initial state, goal state, and structure of the raw

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.

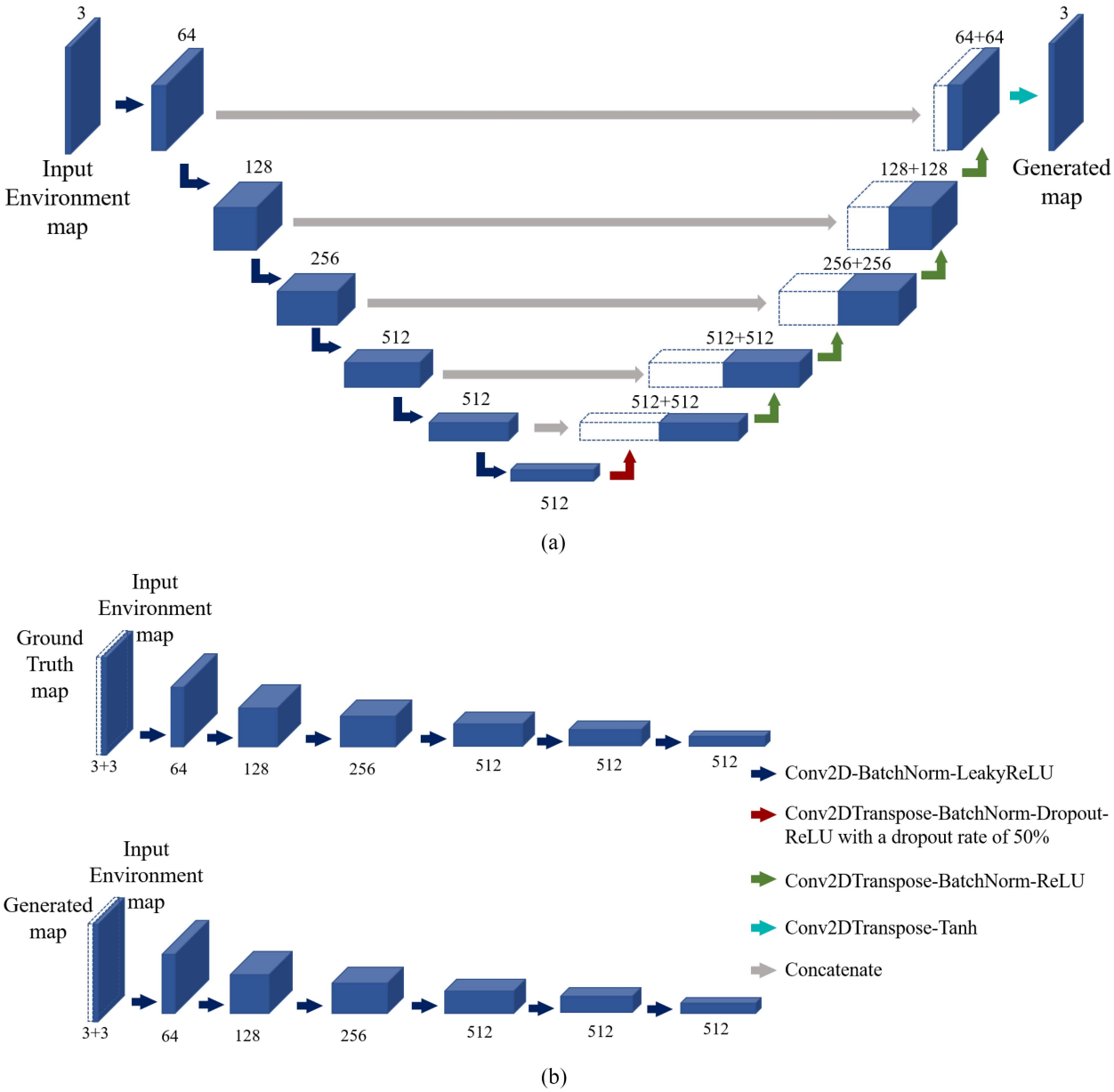


Fig. 3. Illustration of the proposed CGAN model. Blue cubes represent multichannel feature maps, and white cubes represent copied feature maps. (a) Architecture of the generator with a U-net structure. (b) Architecture of two discriminators with two pairs of input.

map. They just differ in the aspect where generated maps have the predicted probability distribution of feasible paths. The proposed model is shown in Fig. 3. All convolution modules are 4×4 spatial filters applied with stride 2. Convolution modules downsample by a factor of 2 in the encoder and upsample by a factor of 2 in the decoder. The dimension of both input and output maps is $(256, 256, 3)$.

For the encoding stage, we use modules of the form Convolution-Batchnorm-ReLU. The output of each encoding layer is called a feature map. Low convolutional layers extract low-level feature maps, and they are fed into higher convolution layers to extract high-level feature maps. If we denote each encoding layer with k filters as C_k, the encoder architecture can be expressed as $C_{64} - C_{128} - C_{256} - C_{512} - C_{512} - C_{512} - C_{512} - C_{512}$. The Batchnorm layer aims to perform the normalization operation for each training mini-batch. It can accelerate the neural network training process and deal with the problem of parameter initialization [28]. The rectified linear unit (ReLU) activation layer is widely used in neural networks, which can help avoid the overfitting problem [29]. It assigns zero output for negative input. The activation function of ReLU is defined as $f(u) = \max(0, u)$, where u is the input value. Leaky-ReLU is a variant of ReLU. It translates negative input into output in the range of $(0, 1)$, that is $f(u) = \max(\eta u, u)$, where η refers to the slope value. All ReLU activation layers in the encoder are leaky, with slope value $\eta = 0.2$.

For the decoding stage, we use modules of the form ConvTranspose-Batchnorm-dropout-ReLU. Let T_k denote each decoding layer with k filters and TD_k denote each decoding layer with a dropout rate of 50%. Then, the decoder architecture can be expressed as $TD_{512} - TD_{512} - TD_{512} - T_{512} - T_{256} -$

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.

$T_{128} - T_{64}$. For a number of CGAN models, random noise z is added into the input of the generator to avoid generating deterministic outputs. However, the generator tends to ignore the random noise, which makes this strategy ineffective. In our decoder architecture, a dropout layer is utilized to act as noise and ensure randomness, which randomly sets a half of input units to zero at each step [30]. We apply the dropout layer in the first three decoding layers at both training and test time. All ReLU layers in the decoding stage are not leaky.

We adapt U-NET as the backbone of our generator network. It is designed for shuttling abundant information between the input and output directly across the network. U-NET is utilized by adding skip connections between each layer i and layer $n-i$, simply concatenating all channels of them, respectively, where n denotes the total number of layers of the generator. Then, the number of channels in decoder is changed, and the final decoder architecture can be expressed as $TD_{512} - TD_{1024} - TD_{1024} - T_{1024} - T_{1024} - T_{512} - T_{256} - T_{128}$. The last layer in the decoder is followed by a convolution layer and a Tanh activation function.

As the generator is trained to produce generated maps that cannot be distinguished from ground-truth maps, we define the generator loss as a sigmoid cross-entropy loss of the generated maps and an array of ones. The function of sigmoid cross-entropy is defined as

$$sce(m, h) = -[h * \ln(M) + (1 - h) * \ln(1 - M)] \quad (3)$$

$$M = \text{sigmoid}(m) = \frac{1}{1 + \exp(-m)}. \quad (4)$$

The L1 loss is added for reducing blurring. Then, the loss function of the generator is shown as follows:

$$L_G = sce(G(o), 1) + \lambda \mathcal{L}_{L1}(G). \quad (5)$$

C. Discriminator Architectures

The discriminator architecture includes two pairs of input. One pair consists of the input environment maps and the corresponding ground-truth maps, which should be classified as real. Another pair consists of the input environment maps and the corresponding generated maps, which should be classified as fake. We concatenate each pair and perform encoding procedures, respectively, using modules of the form convolution-Batchnorm-ReLU. Then, the architecture can be expressed as $C_{64} - C_{128} - C_{256} - C_{512} - C_{512} - C_{512} - C_{512} - C_{512}$. All ReLU activation layers in the architecture are leaky, with slope $\alpha = 0.2$. The last layer is a 1-D output, followed by a sigmoid function.

As the discriminator is trained to distinguish generated maps and ground-truth maps, we define the discriminator loss as the sum of the real loss and the generated loss. The real loss is a sigmoid cross-entropy loss of the ground-truth maps and an array of ones, and generated loss is a sigmoid cross-entropy loss of the generated maps and an array of zeros, respectively. Then, the loss function of the discriminator is shown as follows:

$$L_D = sce(G(o), 0) + sce(y, 1). \quad (6)$$

Algorithm 2: CGAN-RRT* Algorithm

```
V ← {x_init}; E ← ∅;
P ← CGANModel(Map, y);
for i=1,...,n do
    if Rand() < 0.5 then
        x_rand ← Nonuniform(P)
    else
        x_rand ← Uniform_i
    x_nearest ← Nearest(V, x_rand);
    x_new ← Steer(x_nearest, x_rand);
    if ObstacleFree(x_nearest, x_new) then
        V ← V ∪ {x_new};
        E ← E ∪ {(x_nearest, x_new)};
        Rewire();
Return (G = (V, E))
```

D. CGAN-RRT*

The CGAN-RRT* algorithm utilizes the distribution generated by the pretrained CGAN model to guide the sampling process of the RRT* algorithm. There are two sampling strategies in the CGAN-RRT* algorithm. One is the nonuniform sampling strategy guided by the predicted probability distribution of feasible paths generated from the CGAN model, and another is the uniform sampling strategy adopted by the conventional RRT* algorithm.

The details of the CGAN-RRT* algorithm are presented in Algorithm 2. It is initialized with the initial state x_{init} and no edges. CGANModel(Map, y) produces the predicted probability distribution of feasible paths $\mathcal{P}$. At each iteration, a random number Rand() $\in (0, 1)$ is utilized to determine which sampling strategies to choose. If Rand() < 0.5, a node x_{rand} is sampled from Nonuniform($\mathcal{P}$). Otherwise, the uniform sampling strategy is utilized. Then, the algorithm attempts to find the nearest node from the existing vertex set V to x_{rand}, which is denoted as x_{nearest}. After that, x_{rand} is adjusted to x_{new} through a steer function and x_{nearest} is connected to x_{new}. If the connection between x_{nearest} and x_{new} is collision-free, x_{new} will be added to the vertex set V, and $(x_{\text{nearest}}, x_{\text{new}})$ will be added to the edge set E. The procedure of *Rewire* enables the initial path to converge to an optimal path. Finally, the CGAN-RRT* algorithm returns a graph of a new vertex set V and a new edge set E.

V. Simulation Results

In this section, we first introduce the training details of our CGAN model in Section V-A. Then, we present the probability distribution of feasible paths generated from the CGAN model and evaluate its performance in Section V-B. Finally, we compare the CGAN-RRT* algorithm with the conventional RRT* algorithm through several experimental simulations in Section V-C.

A. Training Details

As suggested in the original GAN paper [10], we train the generator to maximize $\log D(o, G(o, z))$, instead of minimizing $\log(1 - D(o, G(o, z)))$. We train our CGAN model on the google co-laboratory platform with NVIDIA TESLA T4 and

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.

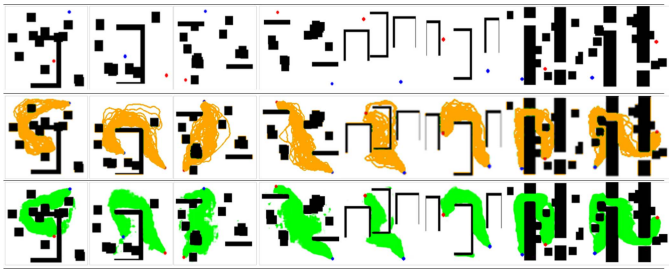


Fig. 4. Examples of the proposed CGAN model. The first row represents four types of input environment maps with random initial states and goal states. The second row represents the corresponding ground-truth maps with 50 feasible paths generated from the RRT algorithm. The third row represents the corresponding generated maps with the predicted probability distribution of feasible paths generated from the CGAN model.

TensorFlow framework. The Adam solver [31] and mini-batch stochastic gradient descent (SGD) are applied in the training process with a learning rate of $\alpha = 0.0002$. In addition, we utilize the suggested momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

To guarantee the divergence and robustness of our CGAN model, 12 000 2-D maps with random initial states and goal states are generated. The training set includes three types of maps. The size of each map is $(256, 256, 3)$. Then, we utilize the RRT algorithm to generate 50 feasible paths on each map. Finally, we combine the input environment maps and corresponding ground-truth maps as our training set. The input environment maps are the raw maps with randomly initial states and goal states, while the ground-truth maps are maps with 50 feasible paths generated from the RRT algorithm. Our CGAN model is trained and tested with Python 3.7. It takes 75 epochs to train the CGAN model, and the whole training process takes around 28 h.

B. Evaluation of the CGAN Model

We generate two groups of combined input environment maps and ground-truth maps as the test set. Each group includes 500 pairs of images with a size of $(256, 256, 3)$. One test group consists of three types of maps which are the same as the training set, and another test group consists of two types of maps that are different from the training set. It only takes the proposed CGAN model 20 ms to generate the predicted probability distribution of feasible paths for a given map. Some examples of generated results from the CGAN model are shown in Fig. 4, which demonstrates the performance of the proposed CGAN-model. The first and second row of Fig. 4 represents the input environment maps and ground-truth maps, respectively. And the generated maps with the predicted probability distribution of paths are presented in the third row. The red and blue circles denote the initial states and goal states, respectively. The orange region of each map in the second row represents 50 feasible paths generated from the RRT algorithm, which is denoted as the ground truth. While in the third row, the green region of each map represents the predicted probability distribution of feasible paths generated from the CGAN model. Fig. 4 illustrates that the predicted probability distribution is close to the ground truth.

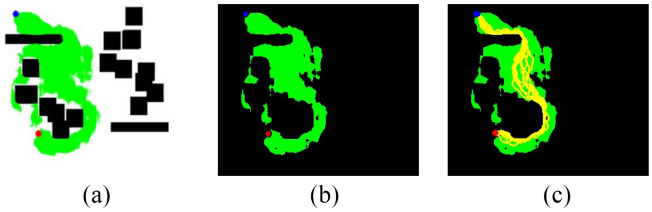


Fig. 5. One example of the test process for the predicted probability distribution of feasible paths. (a) Map with the predicted probability distribution of feasible paths. (b) Map after color filter operation. Space except for the predicted probability distribution of feasible paths, the initial state, and the goal state is turned to black color. (c) Map with paths generated from the RRT algorithm in the limited green space.

A variant of the RRT algorithm is utilized to test the connectivity of the predicted probability distribution of feasible paths generated from the CGAN-model. We set the region of the predicted probability distribution of feasible paths as the promising space $\mathcal{X}_{\text{prom}}$. Other regions are set as collision space. All sampling process is operated in the promising space. If the RRT algorithm can find feasible paths connecting the initial state and goal state in this space, the predicted probability distribution is viewed as feasible and effective. One example of this process is shown in Fig. 5. The red and blue circles denote the initial states and goal states, respectively. We use green to represent the promising space and black to represent the collision space. Feasible paths generated from the RRT algorithm are represented by yellow in Fig. 5(c). The test group consisting of three types of maps which are similar to the training set achieves a 91.8% success rate, while another test group consisting of two types of maps that are different from the training set achieves a 77.8% success rate. The test results illustrate that the proposed CGAN model is effective in generating the probability distribution of feasible paths for a given map.

C. Comparison Between CGAN-RRT* and RRT*

In this section, we test the performance of the CGAN-RRT* and RRT* algorithms on the path-planning problem. The CGAN-RRT* algorithm utilizes the predicted probability distribution of feasible paths to guide the sampling process of the RRT* algorithm, while the conventional RRT* algorithm just uniformly samples nodes from the free space $\mathcal{X}_{\text{free}}$. The step size of the two algorithms is both set to 6.

The illustrations of the predicted probability distribution and optimal path for four maps with different types are shown in Fig. 6. The red and blue circles denote the initial states and goal states, respectively. The green-yellow represents the predicted probability distribution of paths generated by the CGAN model, which indicates that the corresponding region has a higher probability of containing an optimal path than other regions. And we utilize medium-orchid color to represent an optimal path. From the illustrations in Fig. 6, we can find that an optimal path lies in the predicted probability distribution, which indicates that the prediction results generated from the proposed CGAN model can accelerate the convergence to an optimal path and improve the quality of the initial path.

These four maps with different types are utilized to test the performance of the CGAN-RRT* algorithm and the RRT*

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.

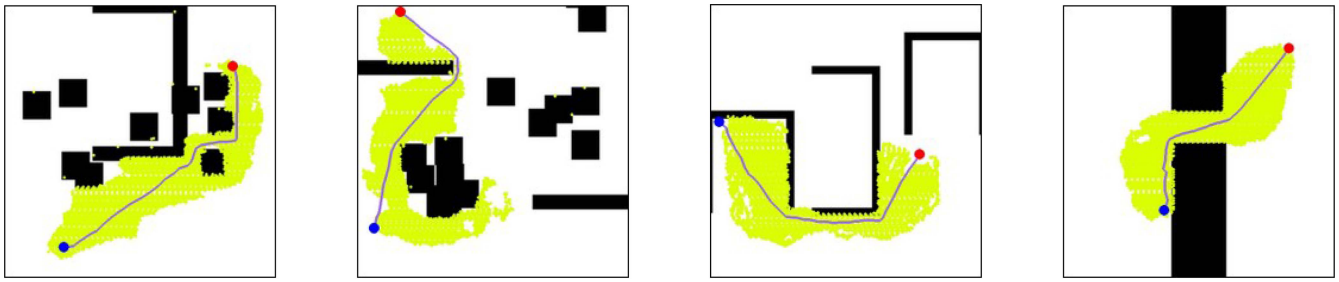


Fig. 6. Optimal paths generated with the predicted probability distribution for four types of maps. Red and blue circles denote the initial state and goal state, respectively. Green-yellow represents probability distribution from the CGAN model, and the medium-orchid represents an optimal path. (a) Map1. (b) Map2. (c) Map3. (d) Map4.

TABLE I
COMPARISON OF PERFORMANCE BETWEEN THE CGAN-RRT* AND RRT* ALGORITHMS

		Number of iterations	Number of nodes	Length of Ini. path	Length of Opt. path
Map1	CGAN-RRT*	**7899**	**14658**	**600**	585
	RRT*	20000	38320	641	
Map2	CGAN-RRT*	**8001**	**15070**	**567**	500
	RRT*	11195	21486	690	
Map3	CGAN-RRT*	**4340**	**6554**	**578**	540
	RRT*	14654	24550	596	
Map4	CGAN-RRT*	**2786**	**5270**	**481**	450
	RRT*	12517	23560	514	

algorithm. The number of iterations, the number of nodes, and the length of the initial path are selected as metrics to evaluate the performance. The number of iterations refers to the required number of iterations during the execution of the entire algorithm. It can indicate the comparison of time cost between different path-planning algorithms. Since the time taken by different programming languages to execute algorithms is a variant, the number of iterations is used to replace the metric of time cost. The number of nodes refers to the total samples when an optimal path is found. It is an essential evaluation metric since it represents the memory usage of path-planning algorithms. The quality of the initial path has a huge impact on path-planning algorithms. If the length of the initial path is smaller, the performance of algorithms tends to be more efficient.

We provide a statistical result in Table I to demonstrate the comparison of performance between the proposed CGAN-RRT* algorithm and conventional RRT* algorithm. Table I illustrates that the CGAN-RRT* algorithm achieves better performance compared with the RRT* algorithm. First, the CGAN-RRT* algorithm needs a much smaller number of iterations to converge to an optimal path. It means that the CGAN-RRT* algorithm can improve the operational efficiency of the path-planning problem. Second, a much smaller number of nodes are used by the CGAN-RRT* algorithm compared with the RRT* algorithm, which reveals that the CGAN-RRT* algorithm utilizes much less memory usage and can save a lot of computation resources. Third, the initial path generated from the CGAN-RRT* algorithm is shorter than the initial path generated from the RRT* algorithm. It means that the initial path generated from the CGAN-RRT* algorithm is closer to an optimal path. Table I shows that the CGAN-RRT* algorithm can always find a high-quality initial path.

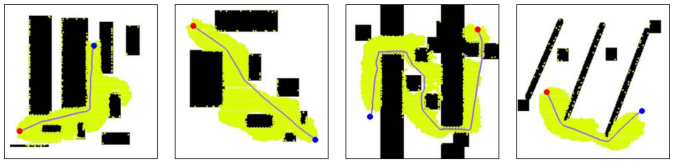


Fig. 7. Optimal paths generated with the predicted probability distribution for four types of complex maps. Red and blue circles denote the initial state and goal state, respectively. Green-yellow represents the predicted probability distribution from the CGAN model, and medium-orchid represents an optimal path. (a) Map5. (b) Map6. (c) Map7. (d) Map8.

In conclusion, the proposed CGAN model is effective as it can generate the predicted probability distribution of feasible paths in a very short time with a good connectivity success rate. Moreover, the CGAN-RRT* algorithm achieves better performance compared with the RRT* algorithm in terms of the number of iterations, the number of nodes, and the length of the initial path.

VI. DISCUSSION

To further validate the performance of the proposed CGAN-RRT* algorithm, some additional experiments are implemented in this section.

First, we compare the connectivity success rate of the predicted probability distributions of feasible paths generated from the CGAN-RRT* algorithm and the Neural-RRT* algorithm [7]. Both the CGAN-RRT* algorithm and the Neural-RRT* algorithm are trained by the same training set, which includes 12 000 pairs of input environment maps and corresponding ground-truth maps. The input environment maps contain random initial states and goal states, while the ground-truth maps contain 50 feasible paths generated from the RRT algorithm. Then, we use a test set consisting of three types of maps that are similar to the training set to evaluate the two algorithms. The test set executed by the CGAN-RRT* algorithm achieves a 91.8% success rate, while it achieves an 86% success rate when applied to the Neural-RRT* algorithm.

Moreover, as shown in Fig. 7, we add illustrations of the predicted probability distribution of feasible paths and the

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.

TABLE II
COMPARISON OF PERFORMANCE BETWEEN THE CGAN-RRT*, INFORMED-RRT*, AND RRT* ALGORITHMS

		Number of iterations	Number of nodes	Length of Ini. path	Length of Opt. path
Map4	CGAN-RRT*	**2786**	**5270**	**481**	450
	Informed-RRT*	5471	7206	519	
	RRT*	12517	23560	514	
Map5	CGAN-RRT*	**3955**	**6965**	**529**	510
	Informed-RRT*	10694	12480	695	
	RRT*	12846	23383	767	
Map6	CGAN-RRT*	**3047**	**5746**	**619**	610
	Informed-RRT*	5310	7664	663	
	RRT*	9787	18760	671	

optimal path for four complex maps. The red and blue circles denote the initial states and goal states, respectively. The green-yellow represents the predicted probability distribution of feasible paths generated by the CGAN model, while an optimal path is shown in medium-orchid. The experimental results show that an optimal path probably lies in the predicted probability distribution of feasible paths for a given complex map.

To further demonstrate the advantage of the proposed CGAN-RRT* algorithm, some experiments are conducted to compare the performance between the CGAN-RRT* algorithm, the Informed-RRT* algorithm [6], and the conventional RRT* algorithm. A statistical result of the performance between them is shown in Table II. Three maps are adopted to execute these path-planning algorithms. Table II illustrates that the CGAN-RRT* algorithm achieves better performance compared with the Informed-RRT* algorithm and RRT* algorithm as the CGAN-RRT* algorithm needs a much smaller number of iterations and nodes and generates a shorter initial path.

VII. CONCLUSION AND FUTURE WORK

In this article, we present a novel algorithm, CGAN-RRT*, for the robotic path-planning problem. It utilizes the predicted probability distribution of feasible paths generated from the CGAN model to guide the sampling process of the RRT* algorithm. The CGAN model is trained by learning from a number of ground-truth maps which are generated by putting all results of executing the RRT algorithm 50 times on raw maps. The experimental results in Sections V and VI suggest that the CGAN-RRT* algorithm is a promising approach for the path-planning problem and it has a much better performance compared with the Informed-RRT* algorithm and the conventional RRT* algorithm in terms of the number of iterations, the number of nodes, and the length of the initial path. Therefore, the proposed CGAN-RRT* algorithm can accelerate the convergence of the RRT* algorithm to an optimal path and provide a high-quality initial path.

In future research, the proposed CGAN model can be applied to other sampling-based path-planning algorithms to guide the sampling process of them and improve their performance. Furthermore, the CGAN-RRT* algorithm can be extended and used in high-dimensional configuration space to deal with high-dimensional path-planning problems.

REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*. New York, NY, USA: Springer, 1986, pp. 396–404.

[2] 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. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

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

[4] 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. 4, no. 2, pp. 100–107, Jul. 1968.

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

[6] 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.*, Chicago, IL, USA, 2014, pp. 2997–3004.

[7] 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.

[8] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Francisco, CA, USA, 2011, pp. 2640–2645.

[9] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn.*, 2001, pp. 327–340.

[10] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2014, pp. 2672–2680.

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

[13] L. Jaillet, A. Yershova, S. M. La Valle, and T. Siméon, "Adaptive tuning of the sampling domain for dynamic-domain RRTs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Edmonton, AB, Canada, 2005, pp. 2851–2856.

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

[15] Q. Zhang, M. Li, X. Wang, and Y. Zhang, "Reinforcement learning in robot path optimization," *J. Softw.*, vol. 7, no. 3, pp. 657–662, 2012.

[16] A. Dubey, R. Mishra, and A. Jha, "Path planning of mobile robot using reinforcement based artificial neural network," *Int. J. Adv. Eng. Technol.*, vol. 6, no. 2, p. 780, 2013.

[17] D. Zhu, X. Cao, B. Sun, and C. Luo, "Biologically inspired self-organizing map applied to task assignment and path planning of an AUV system," *IEEE Trans. Cogn. Develop. Syst.*, vol. 10, no. 2, pp. 304–313, Jun. 2018.

[18] M. Chen and D. Zhu, "A workload balanced algorithm for task assignment and path planning of inhomogeneous autonomous underwater vehicle system," *IEEE Trans. Cogn. Develop. Syst.*, vol. 11, no. 4, pp. 483–493, Dec. 2019.

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

[20] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng, "Brain-inspired cognitive model with attention for self-driving cars," *IEEE Trans. Cogn. Develop. Syst.*, vol. 11, no. 1, pp. 13–25, Mar. 2019.

[21] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014. [Online]. Available: arXiv:1411.1784.

[22] J. Gauthier, "Conditional generative adversarial nets for convolutional face generation," in *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter Semester*, vol. 2014, Stanford Univ., Stanford, CA, USA, 2014, p. 2.

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.

[23] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a Laplacian pyramid of adversarial networks," in *Advances Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2015, pp. 1486–1494.
[24] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 318–335.
[25] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," 2015. [Online]. Available: arXiv:1511.05440.
[26] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon, "Pixel-level domain transfer," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 517–532.
[27] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Interv.*, 2015, pp. 234–241.
[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015. [Online]. Available: arXiv:1502.03167.
[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Meach. Learn. (ICML)*, 2010, pp. 807–814.
[30] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012. [Online]. Available: arXiv:1207.0580.
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: arXiv:1412.6980.

Nachuan Ma received the B.E. degree in electrical engineering and automation from 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.

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 the Ph.D. degree, he spent six months with Stanford University, Stanford, CA, USA, as a Visiting Student Scholar supervised by Prof. O. Khatib. His current research interests include motion planning and control, human–robot interaction, and machine learning in robotics.

Jianbang Liu received the first B.E. degree in microelectronics from Sun Yat-sen University, Guangzhou, China, and the second B.E. degree in electronic engineering from Hong Kong Polytechnic University, Hong Kong, in 2015, and the M.Sc. degree from The University of Hong Kong, Hong Kong, in 2016. He is currently pursuing the Ph.D. degree with The Chinese University of Hong Kong, Hong Kong.

His research interests include sensor fusion, path planning, and robotic perception for decision making and control.

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 is currently a Chair Professor and the Head of 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. He joined The Chinese University of Hong Kong in 2001 as a Professor and later the Chairman of the Department of Electronic Engineering. He was with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, where he served as the Director of the Advanced Robotics and Teleoperation Laboratory and held the positions of an Assistant Professor in 1994, an Associate Professor in 1998, and a Professor in 2000, respectively. He is an Honorary Chair Professor with Harbin Institute of Technology, Harbin, China, and Zhejiang University, Hangzhou, China, and also the Honorary Dean of the School of Control Science and Engineering, Shandong University, Jinan, China. He has published more than 750 journal, conference papers, book chapters, and led more than 60 funded research projects to completion as a Principal Investigator. His research interests include robotics, perception, and intelligence.

Dr. Meng is a recipient of the IEEE Millennium Medal. He has been serving as the Editor-in-Chief and an Editorial Board of a number of international journals and as the General Chair or the Program Chair of many international conferences, including the General Chair of IROS 2005 and the General Chair of ICRA 2021 to be held in Xi'an in May 2021. He served as an Associate VP for Conferences of the IEEE Robotics and Automation Society from 2004 to 2007, the Co-Chair of the Fellow Evaluation Committee, and an Elected Member of the AdCom of IEEE Robotics and Automation Society. He is a Fellow of Hong Kong Institution of Engineers and an Academician of the Canadian Academy of Engineering.

Authorized licensed use limited to: Technische Hochschule Ingolstadt. Downloaded on August 23,2022 at 15:18:05 UTC from IEEE Xplore. Restrictions apply.