EECS 545 Project Final Report: Learning Based Sampling For RRT* Algorithm

Li Chen, Yue Du, Yeyang Fang, Daiyao Yi, Xuran Zhao
{ilnehc,yueduds,yeyangf,yidaiyao,xuranzh}@umich.edu

Abstract

Compared with traditional uniformly sampled path planning approaches, a biased sampling method which learns from demonstrations and environmental restrictions would be preferable due to its efficiency. In this project, we have combined RRT* algorithm and a conditional variational autoencoder (CVAE) trained to learn from the latent dimensions and propose samples accordingly. We have generated and trained on our own datasets. The modified planning algorithms are then applied on the vehicle parking path planning, and compared with the uniformly sampled RRT* and bi-RRT*. A considerable boost is observed in path planning performance with this inclusion of biased sampling network.

1 Introduction

Traditional sample-based motion planning methods such as rapidly exploring random tree (RRT) and Probabilistic RoadMap Planning (PRM) are drawn probabilistically to uniformly cover the work space, which results in relatively slow convergence speed. Thus, a non-uniform sampling strategy that only sampling in those regions where an optimal solution might lie in is utilized to accelerate the planning process. Specifically, in our project, the sample distribution would be generated based on gradually learning from the occupancy grid and demonstrated trajectories with a conditional variational autoencoder (CVAE). The learned sampler would then be utilized by RRT*. The algorithm would then be implemented onto a nonholonomic system for tasks such as parking and obstacle avoidance, and the results would be compared with traditional uniformly sampled RRT* and bi-RRT*. This learning-based approach is promising due to its generality, as it can capture the obstacles and some sense of dynamics in the system.

2 Related Work

Most popular Sampling Based Planning (SBP) algorithms are PRM, RRT and RRT* [1]. PRM is mainly used in highly structured static environment and holonomic robots like industrial robotic arms, while RRT and RRT* can naturally extend non-holonomic constraints which are suitable for car-like mobile robots and dynamic environments.

Traditional RRT* method is introduced in 3.1 and considerable amount of research has demonstrated its problem of huge computational efficiency, high memory consumption and rate of convergence [1]. Hence many different variants of RRT* methods have developed to improve its performance. Noreen *et al.* [1] gave an review of optimal path planning using RRT* based approaches. There are RRT* Fixed Nodes, RRT*-Smart, RRT# methods for holonomic condition and Spline-based RRT* (SRRT*), RRT*i, Potential Function RRT* algorithm for non-holonomic and kinodynamic (velocity, acceleration and force/torque bounds must be satisfied) conditions.

A small part of recent research has turned towards machine learning based approach to design a suitable heuristic for a sampling based motion planner, also coined as adaptive planning. Patil *et al.* [2] used a 3D convolutional neural network to predict bottleneck points for manipulation planning where sometimes are difficult for SBP to explore. After getting the coordinates of bottleneck points in the environment, they redefined RRT* expansion algorithm to bias sampling towards the 'narrow passage'. Koukuntla *et al.* [3] proposed a deep learning based approach to tune the sensitive parameters for potential piloted-RRT* algorithm. Pan *et al.* [4] introduced an instance based learning method by storing prior collision queries and then performing k-NN algorithm to compute a collision probability for the new query which improves 30-100% speed on rigid and articulated robots.

CAVE [5] is an extension of VAE (Variational Autoencoder) and can model latent variables and data conditioned to some random variables. It has the advantage of representing complex, high-dimensional data distribution and reconstruct original data of arbitrary inputs. Hence, we are applying it to extract the latent information and generate the sampling points.

3 Proposed Method

3.1 Path Planning Method - RRT*

RRT* is a sampling based planner that finds solution by dynamically growing a tree in configuration space. It has advantage of continuously optimizing as sample points increase.The special features of RRT* are near neighbor search and rewiring tree operations.

In detail, let $Z \subset R^n$ denotes n dimensional configuration space. Z_{obs} denotes space with obstacles and $Z_{free} = Z/Z_{obs}$ denotes obstacle-free space. Z_{init} is the starting point and Z_{goal} is the target point. In every iteration, it generates a random sample z_{rand} at first. If z_{rand} lies in Z_{free}, then a nearest node $z_{nearest}$ in tree is searched. If z_{rand} is accessible to $z_{nearest}$, it will be inserted in tree by connecting these two nodes. Otherwise it will generate a new node z_{new} by using a steering function and connect it with $z_{nearest}$. After checking collision between z_{new} and $z_{nearest}$, search near points of z_{new} to find potential parent nodes and the searching radius is defined by

$$k = \gamma(log(n)/n)^{(1/d)} \tag{1}$$

where d is space dimension and γ is planning constant related to environment. The best parent node of z_{new} is z_{min} that has least path cost. Then if changing z_{new} to be the parent node of near neighbors reduces path cost, make this change to rewire the tree. After iterations, when z_{goal} is found, a path from z_{init} to z_{goal} is found. Although near neighbor search and rewiring tree improve the path quality, convergence becomes slow as number of nodes increases. Therefore, the sampling method of RRT* is very important.

3.2 Sampling Distribution Learning

Due to the slow convergence of sample based motion planning method such as RRT* introduced above, we propose to develop a biased-sampling method through learning, which could replace the usual uniform-sampling method at each step of RRT* expanding, referring to [6]. The method is to implement a conditional variational autoencoder (CVAE), which learns the distribution from the sampling points along the successful path.

We refer to a sample point as X and external features as C. z denotes the latent variable. CVAE consists an encoder $q_\phi(z|X, C)$ and an decoder $p_\phi(X|z, C)$. Both of them are trained in the training process, while decoder is used to generate the sample points later. Assuming latent variables are Gaussian, the CVAE is trained to maximize the conditional log likelihood:

$$L = \mathbb{E}[logp(X|z, C)] - \mathbf{D}_{KL}(q(z|X, C)\|p(z|C)) \tag{2}$$

Maximizing the lower bound above is equivalent to maximizing the following equation with respective to θ and ϕ.

$$\|x - f(z, y)\|^2 - \mathbf{D}_{KL}(q(z|X, C)\|p(z|C)) \tag{3}$$

Where as would be mentioned below, the term $q_\phi(z|X, C)$ is the normal distribution with mean $\mu_{X,C}$, variance $\sigma_{X,C}$, p(z—C) is the standard normal distribution. The second part of equation(3) is the divergence penalty in the form of KL divergence, which is a method to measure the difference between two probability distributions.

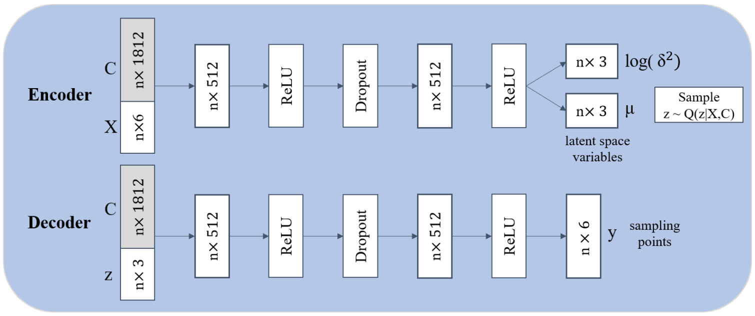


Figure 1: CVAE architecture in our implementation.

The architecture of the network is shown in Figure 1. The encoder takes in input including sampling states from successful planning X and conditions C. For 2D motion, state $X = [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}]$. Conditions C includes obstacle map, initial point and destination point for current path, i.e., $C = [X_{init} \in \mathbb{R}^6, X_{final} \in \mathbb{R}^6, m]$. The map is encoded as an occupancy grid. After passing through the two layers of the network, the log variance and mean of the Gaussian distribution can be obtained. We denote this distribution as $q_\phi(z|X, C) = \mathcal{N}(\mu_{X,C}, \sigma_{X,C})$. The mean and the variance can be inserted to generate KL divergence. Then we could calculate latent variables $z = \mu(X, C) + \sigma(X, C)^2\epsilon$, where ϵ is modeled as $\mathcal{N}(0, I)$ distribution. Combining z with conditions C, we process them through the decoder for obtaining the reconstructed sampling points y.

In the offline test phase, only the decoder is utilized, and z is initialized as randomly distributed noise. Two layers of the neural network are employed in both the encoder and the decoder. The hidden dimension is 512, which is the dimension of the latent feature space assumed. When implementing the loss function, weighted mean square error is used as reconstruction loss, with lower weights for the velocity columns. Furthermore, a weighting parameter ω is used to balance between KL divergence loss and reconstruction loss.

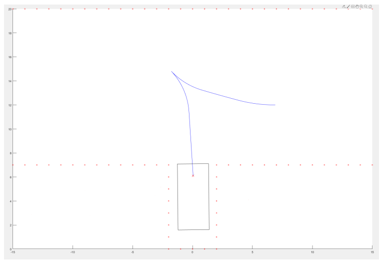
3.3 Learning Based Sampling for RRT*

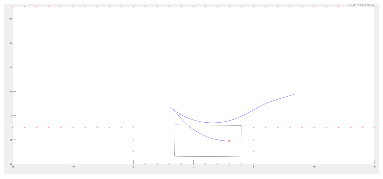
The outline of the proposed method, biased sampling bi-RRT* is shown in Algorithm 1. In a path planning process, the map configuration including starting point, goal point together with obstacles will be input into the decoder of CVAE to get latent sampling points. Then RRT* or bi-RRT* will be used to generate the final path. Specifically, during the sampling process, a threshold ϵ will control sampling from the previous latent points or from random space. The planning procedure completes when RRT* tree reaches the target point, bi-RRT* trees are close enough or the algorithm reaches a certain amount of nodes sampled.

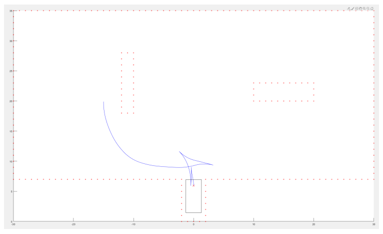
4 Datasets Generation

In our projects, we generate the training and testing datasets on our own. We used Hybrid A* path planning to generate the parking routes. It can generate a smooth path in a given 2-D space for vehicles with nonholonomic constrains. Unlike the regular A*, the hybrid A* can take into account the continuous nature of the search space representing the real world. That's why hybrid A* is widely used in car parking path planning algorithms. In our dataset generation process, the car can move forward or backward. The turning angle for the search process is from -0.6 radius to 0.6 radius with 0.03 resolution.

We have generated data for four scenarios. They are simple map reverse parking, simple map parallel parking, complex map 1 reverse parking, complex map 2 reverse parking. Each folder represents one scenario. Figure 2 shows the configuration of the map and the sample planning trajectory for each scenario.


(a) Scenario 1 map configuration


(b) Scenario 2 map configuration


(c) Scenario 3 map configuration

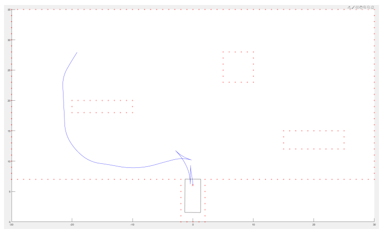

(d) Scenario 4 map configuration

Figure 2: Overview of Hybrid A*.

For each trajectory, some intermediate points along the path are chosen randomly and added to the dataset files. Each line in the dataset includes information of 3 points, i.e., the starting point, the sampling point and the destination point. Each point is represented by position x, position y and angle θ. The dataset will be the input of the network, together with the map configuration.

Table 1 shows the detail of the dataset.

Table 1: Dataset Details.

No.	Dimension(m)	Obstacle	Scenario	Trajectory Generated	Data Generated
1	20 x 30	N	Reverse Parking	154	2310
2	13 x 30	N	Parallel Parking	138	2055
3	35 x 60	Y	Reverse Parking	1001	15015
4	35 x 60	Y	Reverse Parking	507	7605

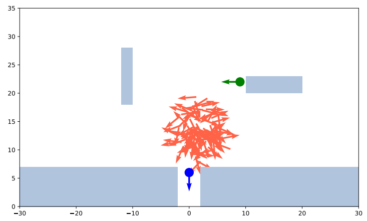
5 Experimental Results

5.1 Sampling Learning

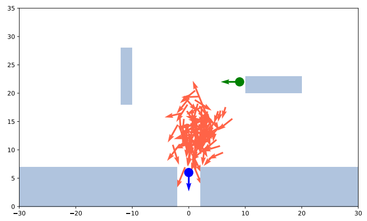
The model is trained in mini-batches of 128. Adam optimizer is used with a learning rate of 10^{-4} and 50000 iterations. Dropout rate is kept to be 0.5 for all the dropout layers. The obstacle map is encoded with a grid size of $1m \times 1m$, and the size of the maps is unified to be $35m \times 60m$. Variant starting positions and end positions are used in the test cases.

5.1.1 Orientation Prediction Improvements

Initially, only the position of the sampling states are included for input into the training network, and the velocity columns are left to be empty, i.e., $X = [x, y, \theta, 0, 0, 0]$. However, it is discovered that inputting orientation in cosine and sine format as velocities, i.e., $X = [x, y, 0, cos(\theta), sin(\theta), 0]$, would result in a better performance of orientation prediction. This modification also conforms to the dynamics of vehicles, since their motion are constrained along the orientation.



(a) training with θ



(b) training with $cos(\theta)$ and $sin(\theta)$

Figure 3: Orientation prediction could be improved by training the orientation as cosine and sine components in velocity columns.

5.1.2 Weighted Loss Tuning

The weight parameter ω in the loss function is used to balance between satisfying reconstruction requirement and the strength of the prior, and it would influence the training result substantially. As shown in Figure 4, with an increasing ω, the sampling cluster would be a better representation of optimal route connecting the initial and destination points; however, the distribution has a poorer coverage of the space. In our case, 10^{-2} would be a great choice, since the sampling clusters connect smoothly from the initial point to the endpoint, while the coverage of the distribution is wide enough for the planner to explore around in the working space.

5.1.3 Architecture of the CVAE

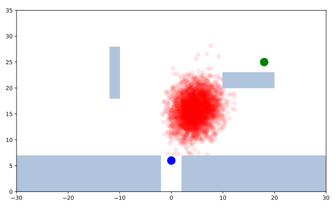
Experiments have also been conducted by modifying the structure of the CVAE and visually inspecting the performance of training through convergence and stability. The results are presented in Figure 5. It seems the modification does not result in an obvious boost on the performance of the network. Though, with more layers, the training tends to be more unstable.

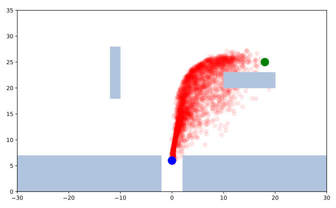
5.2 Sampling based path planning experiments

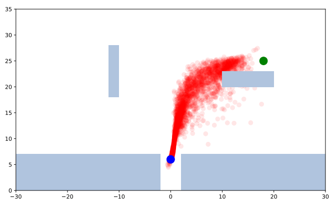
In this section, results of planning algorithm on a simple map and a complex map are presented and analyzed. Traditional algorithm of uniformly sampling RRT* and bi-RRT* are used as baselines, and are compared to the biased sampling RRT* and bi-RRT* with the inclusion of our learnt sampling pipeline.

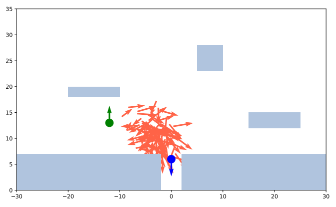
5.2.1 Simple Map Scenario

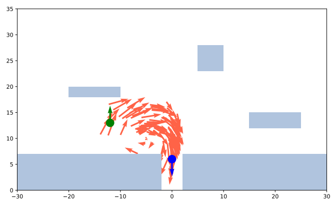
In Figure 6, results are shown for path planning in simple map scenario. The basic single directional RRT* algorithm is shown in Figure 6a. Proposed nodes are randomly selected within the map, and the RRT* tree will gradually extend to cover the most area of the working space before it reaches the destination point. Figure 6c shows the result of bi-RRT* algorithm, which generates two trees from starting and goal points


(a) ω = 10^{-4}


(b) ω = 10^{-2}


(c) ω =1


(d) ω = 10^{-4}


(e) ω = 10^{-2}

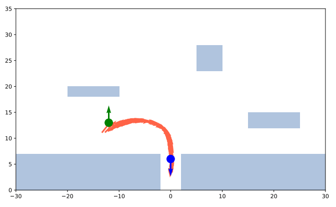
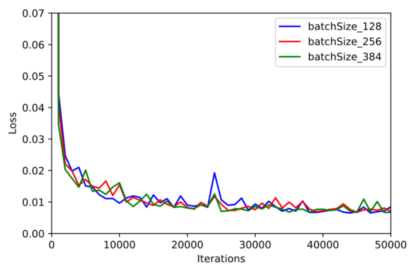
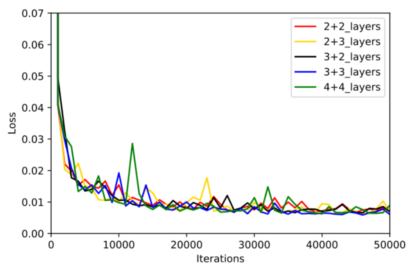

(f) ω =1

Figure 4: Weighted Loss Tuning.



(a) different batch sizes



(b) different number of layers

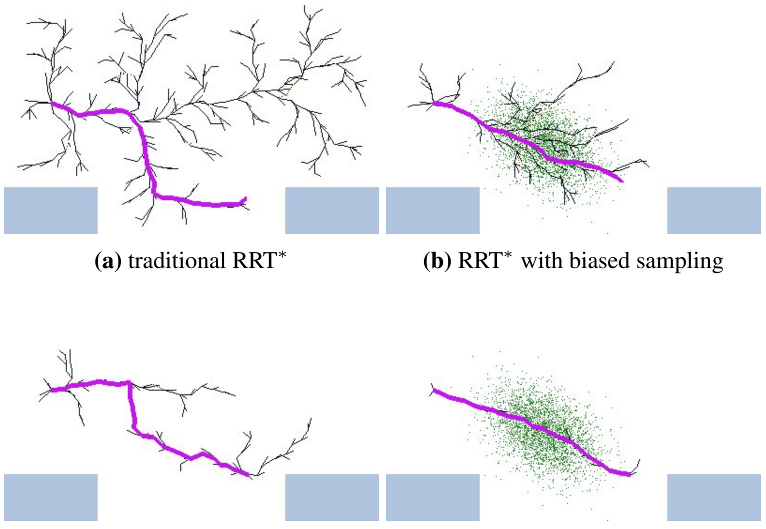
Figure 5: Loss v.s. epoch with modification in CVAE networks.

simultaneously and take turns to expand before they adjoin. The randomness hidden in these two traditional algorithms results in really unstable performance. In three basic performance evaluation parameters, i.e., the number of nodes expanded, elapsed time, and final path length, the original algorithm exhibits higher values and a large standard deviation, as shown in Table 2.

After implementing CVAE for proposal of the sampling points, we chose a greedy algorithm with $\epsilon = 0.2$ to obtain nodes as below.

$$New\ node = \begin{cases} \text{random point among the map} & rand() < \epsilon \\ \text{point from latent samples set} & rand() \geq \epsilon \end{cases} \tag{4}$$

The planning results for biased RRT* and bi-RRT* are shown in Figure 6b and Figure 6d. With biased sampling method through learning included in the planning process, three evaluation parameters have all improved and are more statistically stable, as could be demonstrated by smaller values in the parameters and their standard deviation. However, it could be observed that the biased sampling method through learning does not improve much for single directional RRT* algorithm. The primary reason for this is the simplicity



(a) traditional RRT* **(b)** RRT* with biased sampling

(c) traditional bi-RRT* **(d)** bi-RRT* with biased sampling

Figure 6: Simple map experiment.

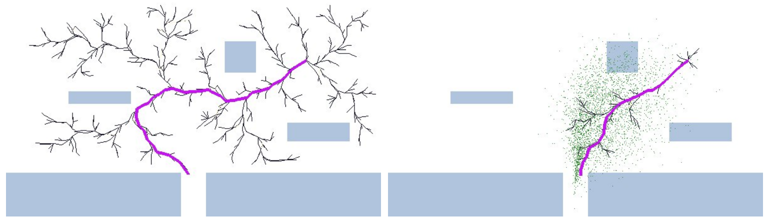
of the map. The sampling points tend to converge to the middle portion of the optimal path, where the proposed distribution is the densest. Hence, it still took quite long for the tree to reach the destination.

Table 2: Mean and standard deviation for simple map (Fig. 6) experiment.

	# nodes		**elapsed(s)**		**path_len**$(\times 0.1m)$	
	random	learnt	random	learnt	random	learnt
RRT* (σ)	251.5629	244.4267	3.6873	3.5023	265.2073	221.5089
	(119.9123)	(90.5567)	(3.8788)	(2.5824)	(28.80450)	(10.4951)
bi-RRT* (σ)	93.03	48.57	0.2290	0.0692	257.7198	228.9368
	(23.6036)	(5.2093)	(0.1084)	(0.01397)	(20.3302)	(6.8773)

5.2.2 Complex Map Scenario

In comparison, in a complex map scenario in Figure 7, both RRT* and bi-RRT* show great improvement with the learnt biased sampling points method, as shown in Table 3. For RRT*, the number of nodes extended, elapsed time and path length have dropped by 72.5%, 91.4% and 18.0% respectively. For bi-RRT*, they have dropped by 55.1%, 70.4% and 17.7% respectively.


(a) traditional RRT* **(b)** RRT* with biased sampling

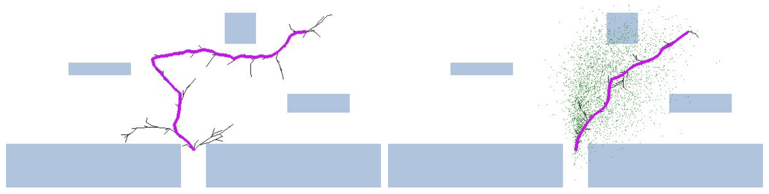

(c) traditional bi-RRT* **(d)** bi-RRT* with biased sampling

Figure 7: Complex map experiment.

Table 3: Mean and standard deviation for complex map (Fig. 7) experiment.

	# nodes		**elapsed(s)**		**path_len**$(\times 0.1m)$	
	random	learnt	random	learnt	random	learnt
RRT* (σ)	492.23	135.2	28.0249	2.3905	317.2249	260.7166
	(137.0647)	(28.8119)	(16.4521)	(0.9523)	(37.1870)	(9.1753)
bi-RRT* (σ)	156.32	70.56	1.7625	0.5169	340.8630	280.5843
	(45.5480)	(11.7760)	(0.9827)	(0.1635)	(55.2340)	(11.2468)

6 Conclusion

In conclusion, we have completed the following tasks in this project: (1) Generated the datasets for training and testing of our method on our own, including four types of maps and $26,985$ sampling states to learn. (2) Trained a CVAE to learn from the dataset and output sampling points through map, initial point and destination point directly, and improved the performance of the network through weight tuning and input modification. (3) Included the learnt sampling pipeline in RRT* and bi-RRT* path planning method; compared them with the baseline approaches - random sampling RRT* and bi-RRT*.

It could be shown that a biased sampling method would substantially improve the efficiency and stability of the path planning algorithm, especially for bi-RRT* and in challenging environments. Future work might include diversity in the obstacle map datasets for training and testing, and the comparison of CVAE to other possible networks models.

Author Contributions

All authors together built the outline and pipeline of the learning based sampling RRT* algorithm and wrote the report. Yeyang Fang implemented the hybrid A* algorithm in MATLAB to generate the dataset; Li Chen conducted the related work survey and performed path planning experiments; Yue Du, Daiyao Yi and Xuran Zhao developed the CVAE neural network and trained the sampling algorithm with the dataset.

References

[1] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: a survey and future directions," *Int. J. Adv. Comput. Sci. Appl*, vol. 7, no. 11, pp. 97–107, 2016.

[2] I. Patil, B. K. Rout, and V. Kalaichelvi, "Prediction of bottleneck points for manipulation planning in cluttered environment using a 3d convolutional neural network," 2019.

[3] S. R. Koukuntla, M. Bhat, S. Aggarwal, R. K. Jenamani, and J. Mukhopadhyay, "Deep learning rooted potential piloted rrt* for expeditious path planning," in *Proceedings of the 2019 4th International Conference on Automation, Control and Robotics Engineering*, CACRE2019, (New York, NY, USA), Association for Computing Machinery, 2019.

[4] J. Pan, S. Chitta, and D. Manocha, "Faster sample-based motion planning using instance-based learning," *Algorithmic Foundations of Robotics X*, 01 2013.

[5] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in neural information processing systems*, pp. 3483–3491, 2015.

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

Appendix

Algorithm 1 Learning based bi-RRT*

```
 Input: $Z,\ Z_{init},\ Z_{goal}$
 Output: path from $Z_{init}$ to $Z_{goal}$
$N \leftarrow 2000$
$\alpha \leftarrow 7$
$\epsilon \leftarrow 0.2$
$y \leftarrow$ CVAE($Z,\ Z_{init},\ Z_{goal}$)
$tree_{init} \leftarrow Z_{init}$
$tree_{goal} \leftarrow Z_{goal}$
while $\#\ nodes\ expanded < N$ do
    foreach $tree \in \{tree_{init},\ tree_{goal}\}$ do
        if $rand < \epsilon$ then
            sample point randomly
        end
        else
            sample point from $y$
        end
    end
    Update nearest neighbors and cost
    if $dist(tree1,\ tree2) < \alpha$ then
        break
    end
end
```