Deep RRT*

Xuzhe Dang, Lukáš Chrpa, Stefan Edelkamp

Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo nám. 13, 121 35 Praha 2, Czech Republic
{dangxuzh, chrpaluk, edelkste}@fel.cvut.cz

Abstract

Sampling-based motion planning algorithms such as Rapidly-exploring Random Trees (RRTs) have been used in robotic applications for a long time. In this paper, we propose a method that combines deep learning with RRT* method. We use a neural network to learn a sample strategy for RRT*. We evaluate Deep RRT* in a collection of 2D scenarios. The results demonstrate that our algorithm could find collision-free paths efficiently and fast, and can be generalized to unseen environments.

Introduction

Efficient and safe motion planning algorithms are crucial for the applications of robots in life and industry. One of the most developed motion planning for these requirements is sampling-based motion planning, such as probabilistic roadmap (PRM) (Kavraki et al. 1996), Rapidly-exploring Random Trees (RRT) (LaValle and Kuffner Jr 2001), and RRT* (Karaman and Frazzoli 2011). The RRT expands the trees by randomly generating new samples. This sampling strategy is fast and efficient to find a collision-free path for the robot in a low-dimensional space. However, the number of samples generated by this random strategy will grow fast with the increasing sizes and dimensions of the space, and the algorithms like RRT and RRT* will be computationally expensive.

The idea to improve the sample generation strategy of RRTs is well studied in motion planning research (Gammell, Srinivasa, and Barfoot 2014; Islam et al. 2012). However, most of these works are focusing on improving the quality of the paths. The runtime of these methods is still quite high.

Recently, learning-based motion planning algorithms have shown their promising ability to solve problems efficiently. Some of these works try to replace the whole motion planner (Qureshi et al. 2019; Huh, Isler, and Lee 2021) while others try to improve some components of classical motion planning algorithms. However, they usually require demonstration data from other motion planning algorithms (Wang et al. 2020; Ariki and Narihira 2019). Thus, the quality of the paths found by these algorithms is strongly dependent on the algorithms used to generate examples.

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In this paper, we propose a deep learning-based RRT algorithm, called Deep RRT*, that finds collision-free paths with a small number of samples. Deep RRT* trains a model that predicts the distribution of the expanded nodes position while giving the current node position, the goal position, and a latent representation of obstacle space. Our current experiment results in a collection of 2D scenarios show the potential of Deep RRT*.

Problem Definition

Let $S \subseteq \mathbb{R}^n$ be the state space of a given problem, $S_{obs} \subsetneq S$ be the obstacle state space, and $S_{free} := S \backslash S_{obs}$ be the free state space. Let $S_{init} \in S_{free}$ be the initial state and $S_{goal} \subsetneq S_{free}$ be the goal region. Let a collision-free path τ be a continuous mapping in state space S such that $\tau : [0, 1] \rightarrow S_{free}$, where $\tau(0) = S_{init}$ and $\tau(1) \in S_{goal}$. Let T be the set of collision-free paths. Let $c(\cdot)$ be a cost function of the path, the optimal motion planning problem is to find a optimal path that has the minimum cost.

$$\tau^* = \text{argmin}_{\tau \in T}\, c(\tau) \tag{1}$$

Deep RRT*

This section introduces our proposed method, Deep RRT*. Instead of using a random sample strategy, our method trains a model to "guide" the sampling strategy. Our model uses a similar architecture to MPNet (Qureshi et al. 2019) that contains two neural networks - the encoder network and the policy network. The first part of our model, the encoder network, embeds the obstacles state space S_{obs} that is represented by point clouds into an m-dimensional latent space Z, where $m \subseteq \mathbb{R}$. The second neural network is a policy network. Given the current tree node state, the goal state, and the latent space Z from the encoder, the policy network predicts the distribution of the subsequent node. Then Deep RRT* expands the tree by sampling a new node with the predicted distribution.

Our model, both encoder network and policy network, is trained by self-play. In each iteration, Deep RRT* finds N successful paths from trees with a set of predefined state spaces S, randomly generated initial states S_{init}, and randomly generated goal states S_{goal}, where $N \subseteq \mathbb{R}^+$, and then store these paths in a rollout buffer in each epoch. Then we

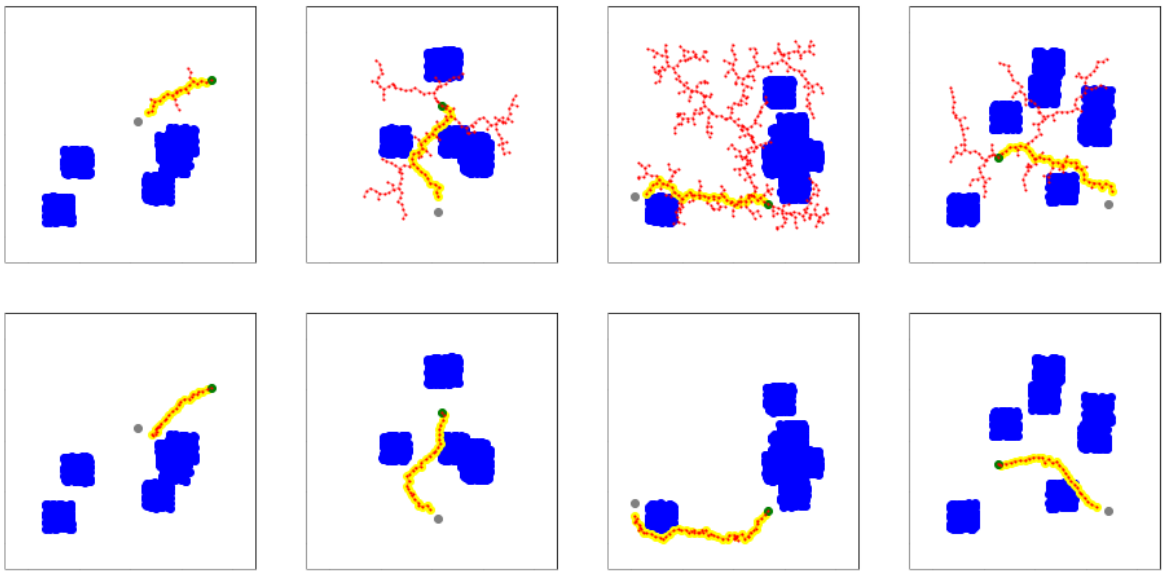

Figure 1: The top row shows the results of RRT*. The bottom row shows the results of Deep RRT*. The blue points represent obstacles. The red points represent the tree nodes. The yellow lines represent found paths. The green nodes are the initial states, and the grey nodes are the goal states.

trained our model π with these data by optimizing the objective

$$min_{\theta^\pi} L = -\frac{1}{N}\sum_{j}^{N}\sum_{i}^{M-1}\log \pi(S_{i+1}^{j} \mid S_i^j, S_{goal}, S_{obs}), \tag{2}$$

where θ^π is the parameter of our model π, and M is the number of nodes in a path.

Results

The model we proposed is implemented in PyTorch (Paszke et al. 2019). We use RRT* implemented in Python as the benchmark algorithm, and compare it with Deep RRT*. We train and test Deep RRT* in the same 2D environments used by MPNet. The 2D environment contains 110 different scenarios. We use 100 scenarios to train our model and keep the remaining scenarios for testing. Each testing scenario contains 2000 different initial and goal configurations. We randomly selected 500 configurations of those. In our experiments, the maximum number of nodes of the tree is set to 1000 for RRT*, and 300 for Deep RRT*. During the planning, once a node expanded is located in the goal region, the path is returned. Otherwise, the problem is considered unsolved.

Figure 1 shows four scenarios, each solved by RRT* (top row) and Deep RRT* (bottom row). Table 1 presents the success rate, search tree size and CPU-time comparison of Deep RRT* against RRT* on the testing scenarios.

As Table 1 shows, Deep RRT* is more "sample efficient" and faster than RRT*. The size of the search tree built by Deep RRT* is only 37.16% of the size of the search tree built by RRT*. The average time to find a path with Deep RRT* is 0.36 seconds, and the average time to find a path with RRT* is 0.60 seconds.

Algorithm	Succ. Rate	Tree Size	Time
RRT*	97.4%	199.87 ± 150.01	0.60 ± 0.72
Deep RRT*	90.1%	74.28 ± 126.93	0.36 ± 1.19

Table 1: Comparison with RRT*.

The success rate of Deep RRT* is, however, reduced by 7.3 %. We find that in most of unsuccessful cases, when the search tree is expanded close to the goal area, the new node generated by Deep RRT* misses the goal area, and then it generates samples randomly in the area close to the goal until it reaches it or the maximum number of samples is exceeded. We believe that adding a weighted entropy loss to the loss function, and reduce the weight of entropy loss during the training may help.

Conclusion and Future Work

In this paper, we present, Deep RRT*, a method that combines the deep learning method with sampling-based motion planning. The neural network used in Deep RRT* contains an encoder to embed the obstacle state space into a latent space and a policy to predict the distribution for sampling the next node while having the current node, and the goal state.

Currently, we are working on the design of a heuristic based mechanism to select the most promising node to expand in the search tree. In future, we plan to train and test Deep RRT* with the select mechanism in more environments like complex 2D environments, 3D environments, and rigid-body environments.

Acknowledgements

This research was funded by Czech Science Foundation (project no. 22-30043S) and by the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

References

Ariki, Y.; and Narihira, T. 2019. Fully convolutional search heuristic learning for rapid path planners. *arXiv preprint arXiv:1908.03343*.

Gammell, J. D.; Srinivasa, S. S.; and Barfoot, T. D. 2014. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2997–3004. IEEE.

Huh, J.; Isler, V.; and Lee, D. D. 2021. Cost-to-go function generating networks for high dimensional motion planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 8480–8486. IEEE.

Islam, F.; Nasir, J.; Malik, U.; Ayaz, Y.; and Hasan, O. 2012. Rrt*-smart: Rapid convergence implementation of rrt* towards optimal solution. In *2012 IEEE international conference on mechatronics and automation*, 1651–1656. IEEE.

Karaman, S.; and Frazzoli, E. 2011. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7): 846–894.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4): 566–580.

LaValle, S. M.; and Kuffner Jr, J. J. 2001. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5): 378–400.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Qureshi, A. H.; Simeonov, A.; Bency, M. J.; and Yip, M. C. 2019. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, 2118–2124. IEEE.

Wang, J.; Chi, W.; Li, C.; Wang, C.; and Meng, M. Q.-H. 2020. Neural RRT*: Learning-based optimal path planning. *IEEE Transactions on Automation Science and Engineering*, 17(4): 1748–1758.