Region-Guided Rapidly Exploring Random Tree for Sampling-based Path Planning

12232419 Zijian Huang

***Abstract*—The rapidly exploring random tree (RRT) is a popular sampling-based path planning algorithm, as it allows researchers to avoid modeling the robot's motion space during the planning process. However, the RRT algorithm also has some drawbacks, such as the need to explore areas that are irrelevant to the final path during the planning process, as well as issues with slow convergence. In this work, we propose a region-guided RRT improvement algorithm based on regional guidance. To avoid the issue of the RRT algorithm randomly sampling and creating many unnecessary branches in the tree, we aim to establish a useful heuristic function to guide the exploration process. Specifically, we use a neural network to predict "high-value" regions in the map. The definition of "high-value" regions is straightforward - it refers to areas in the map that can be directly connected to the endpoint via a straight line (without colliding with obstacles). These areas are defined by a set of points that form a region. Since any feasible path from the starting point to the endpoint must pass through this region, we aim for the RRT-generated tree to explore this area as quickly as possible. When the nodes of a tree explore so-called "high-value" areas, we hope to increase the probability of sampling towards the goal, so that the tree can quickly explore towards the endpoint. We tested our algorithm on 1500 randomly generated maps of size 256x256, including 1000 maps for training and 500 for testing. Experimental results demonstrate that our algorithm outperforms the RRT algorithm in terms of number of nodes, path length, algorithmic time, and success rate of path planning.**

***Index Terms*—convolutional neural network, sampling-based path planning, motion planning**

I. Introduction

PATH planning plays a critical role in the fields of robotics and autonomous driving, aiming to generate a collision-free path from a given starting point to a destination. In this process, path planning algorithms need to consider obstacles in the environment, which may be either static or dynamic. Finding the possible route is one of the most critical challenges to be addressed for unmanned aerial vehicles (UAVs) in order to determine the ideal course between a starting point and an end goal [1]. Currently, there are many excellent path planning algorithms, including graph-based algorithms, optimization-based algorithms, sampling-based algorithms, and artificial potential field methods. Graph-based path planning algorithms, also known as grid-based methods, require the environment to be discretized into grids. Some well-known examples of these algorithms include A* [2] and Dijkstra's algorithms. The advantage of these algorithms is that they can guarantee finding the optimal path. However, their search efficiency decreases as the map size increases. The main idea behind the artificial potential field method [3] is to artificially assign attractive or repulsive forces to objects in the map. Specifically, repulsive forces are assigned to the starting point and obstacles, while an attractive force is assigned to the destination. The repulsive force decreases as the distance to an object increases, while the attractive force increases as the distance to the destination increases. One of the drawbacks of the artificial potential field method is that it may get stuck in a local minimum when the combined force of the repulsive and attractive forces is zero. Sampling-based methods are represented by algorithms such as the Rapidly-exploring Random Tree (RRT) [4] and Probabilistic Roadmap (PRM) algorithms [5]. Sampling-based methods are applicable in high-dimensional spaces, but their drawback is that they often generate many unnecessary samples, leading to slow computation times. Additionally, the paths generated by these methods are typically not optimal. To address these challenges, this paper proposes a novel RRT-based algorithm that utilizes region-guidance to improve path planning.

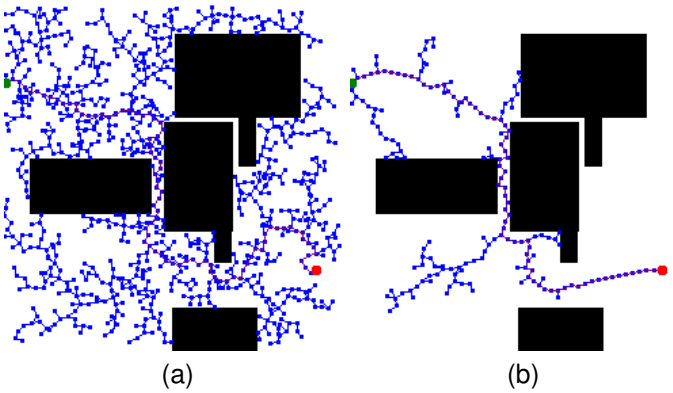

Fig. 1. An comparison between RRT(a) and Region-Guided RRT(b) Algorithm. The starting point is green, the ending point is red, the nodes of the tree are blue, and the final path is purple. Best viewed in color.

To avoid excessive sampling in areas irrelevant to the path in the RRT algorithm, we define a region with high value and increase the probability of sampling in this region. After exploring the high-value region, we increase the probability of sampling towards the destination. As shown in Fig. 1, the Region-Guided RRT algorithm can avoid many unnecessary samples and explore the environment faster to reach the destination.

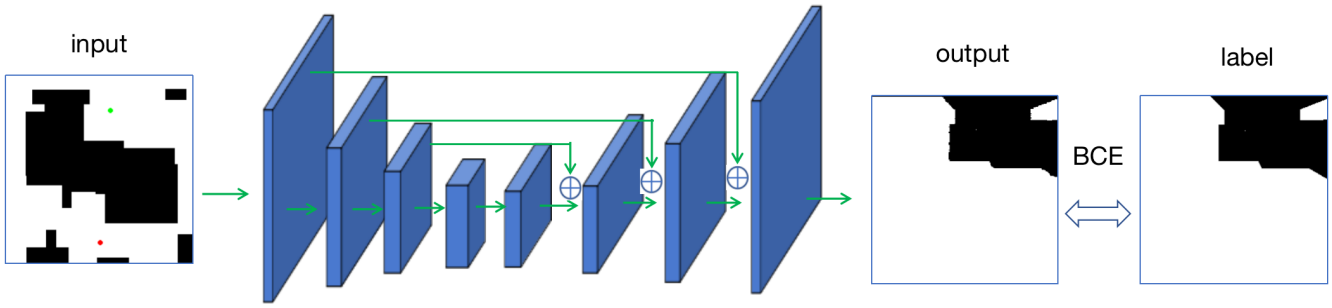

Fig. 2. Pipeline of the region generator. The network consists of an encoder and a decoder. The input to the network includes the map, starting point, and ending point. The output of the network is the high-value region.

II. Related work

To address issues such as slow convergence and non-optimal paths in the RRT algorithm, researchers have proposed various modifications to the original algorithm. One example of such modification is the RRT* algorithm. Building on the RRT algorithm, the RRT* algorithm does not immediately connect a new node X_{new} to the nearest node in the tree. Instead, it attempts to connect the new node X_{new} to all nodes within a circle centered at the new node and with a radius r. The algorithm then checks the cost of the path from the root node to the new node for each potential connection. The connection with the lowest cost is established, and the nodes in the tree are rewired accordingly. This process is known as rewire. It can be proved that when the radius r satisfies certain conditions, the RRT* algorithm is asymptotically optimal. The Informed RRT* [6] algorithm improves the convergence of RRT* by using sampling within an ellipsoidal region instead of uniform sampling throughout the environment. This approach reduces the number of unnecessary samples and accelerates the convergence of the RRT* algorithm. The RRT-Connect [7] algorithm constructs two separate trees at the start and end points, and expands towards a sampled node X_{rand}. The algorithm also incorporates a heuristic search to improve its efficiency. The PathGan [8]

With the development of deep learning technology, some researchers [9] [8] are interested in leveraging the powerful fitting capabilities of deep neural networks to obtain heuristic information for path planning algorithms. PathGAN [8] and Neural RRT* [9] are both examples of heuristic RRT* algorithms. PathGAN trains a GAN network to generate potential paths for the RRT* algorithm and increases the probability of sampling in areas covered by these generated paths. Neural RRT*, on the other hand, uses a deep neural network to predict the probability of each pixel in the map being on the optimal path and adjusts the sampling probability based on these predictions. Both algorithms leverage heuristic information to improve the efficiency and optimality of the path planning process.

III. Proposed method

A. Region Generator

The pipeline of the region generator is shown in Fig. 2 As previously mentioned, we define a high-value region as an area on the map composed of points that can be connected to the destination with a straight line without colliding with obstacles. For a fixed map with a given starting point, ending point, and obstacles, the high-value region is also fixed. Therefore, we can use a deep neural network to predict this region. The design of the neural network is very simple and consists of a fully convolutional network, which allows it to handle maps of various resolutions. The encoder part of the network is a classic VGG network, while the decoder part consists of several de-convolutional layers and batchnorm layers. The input image include start point(red), end point(green), and obstacles(black). The shape of the input data is $H \times W \times 3$. The output of the encoder consists of 5 different feature maps of varying scales. The output of the encoder is passed through a cascaded decoder structure to produce a binary classification result for each pixel($H \times W \times 2$).

B. Region-Guided RRT

Incorporating a region generator, we propose a region-based RRT algorithm. The details of the algorithm are presented in Algorithm1. At the start of the algorithm, we use a region generator to generate a probability prediction for each pixel being in the high-value region. During the sampling process, there is a 50% chance of sampling within the predicted region, which ensures the probabilistic completeness. When a node is explored within the predicted region, the sampling mode changes to increase the probability of sampling directly towards the goal configuration. This approach prioritizes sampling within the high-value region and reduces unnecessary sampling in areas with low probability of being on the optimal path, which can improve the efficiency of the algorithm.

IV. Experiments

A. Dataset

We randomly generated 1500 maps with a size of 256*256 pixels, where each map contained 5 to 20 obstacles represented by black pixels. The starting and ending points for each map were also randomly generated. Out of these 1500 maps, we used 1000 maps as the training dataset for the region generator, and the remaining 500 maps were used as the testing dataset. Every map underwent connectivity testing using the RRT algorithm.

Algorithm 1 Region-guided RRT

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

$V \leftarrow x_{init}$, $E \leftarrow \emptyset$, $\mathcal{T} = (V, E)$, $p_1 = 0.5$, $p_2 = 0.05$,
$\mathcal{O} \leftarrow RegionGenerator(Map)$
for $i = 1...N$ **do**
  if $Rand() > p_1$ **then**
    $x_{rand} \leftarrow NonuniformSample(\mathcal{O})$
  else
    if $Rand() < p_2$ **then**
      $x_{rand} \leftarrow x_{goal}$
    else
      $x_{rand} \leftarrow uniformSample()$
    end if
  end if
  $x_{nearest} \leftarrow Nearest(\mathcal{T} = (V, E), x_{rand})$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$
  if $ObstacleFree(x_{nearest}, x_{new})$ **then**
    $V \leftarrow V \cup \{x_{new}\}$
    $E \leftarrow E \cup \{(x_{new}, x_{nearest})\}$
    if $x_{new} \in \mathcal{O}$ **then**
      $p_1 = 0.8$
      $p_2 = 0.75$
    end if
  end if
end for
return $\mathcal{T}$

B. Experiment Setup

We conducted experiments on a machine with an Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz processor and Ubuntu 22.04.2 LTS operating system. For the training of the region generator, we used an NVIDIA TITAN V@12GB graphics card.

The training configuration for the region generator is as follows: we used a batch size of 4 and trained for a total of 120 epochs. We used the SGD optimizer with a momentum of 0.7 and an initial learning rate of 0.01. The loss function is binary cross entropy loss.

C. Results

The experimental results will be presented in the following sections:

1) Analysis of the quality of the generated regions: This section will analyze the quality of the regions generated by the region generator on the training and testing datasets. As shown in Fig. 3, as the training progresses, the predicted regions become more precise, indicating that our network has sufficient fitting capabilities. It may be challenging for the network to predict the regions accurately for some narrow passages.
2) Case study of region-guided RRT vs. traditional RRT: This section will compare the results of path planning using region-guided RRT and traditional RRT on several sample maps. As shown in Fig. 4 we compared the performance of Region RRT and traditional RRT under three different scenarios: when there are no obstacles between the starting and ending points, when there are obstacles that need to be bypassed, and when there is a narrow passage between the starting and ending points. The results show that our Region RRT outperforms traditional RRT in all three scenarios.

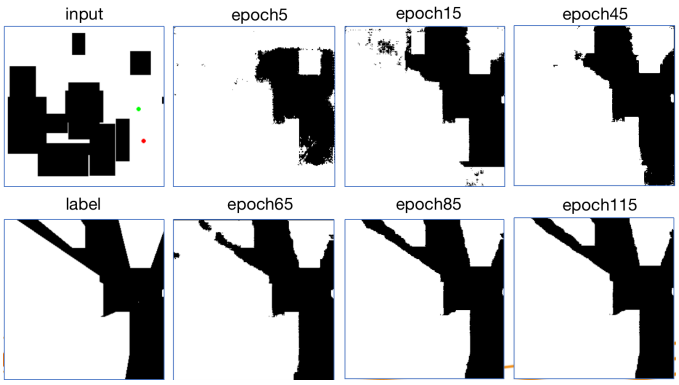

Fig. 3. Quality analysis of generated regions by the region generator

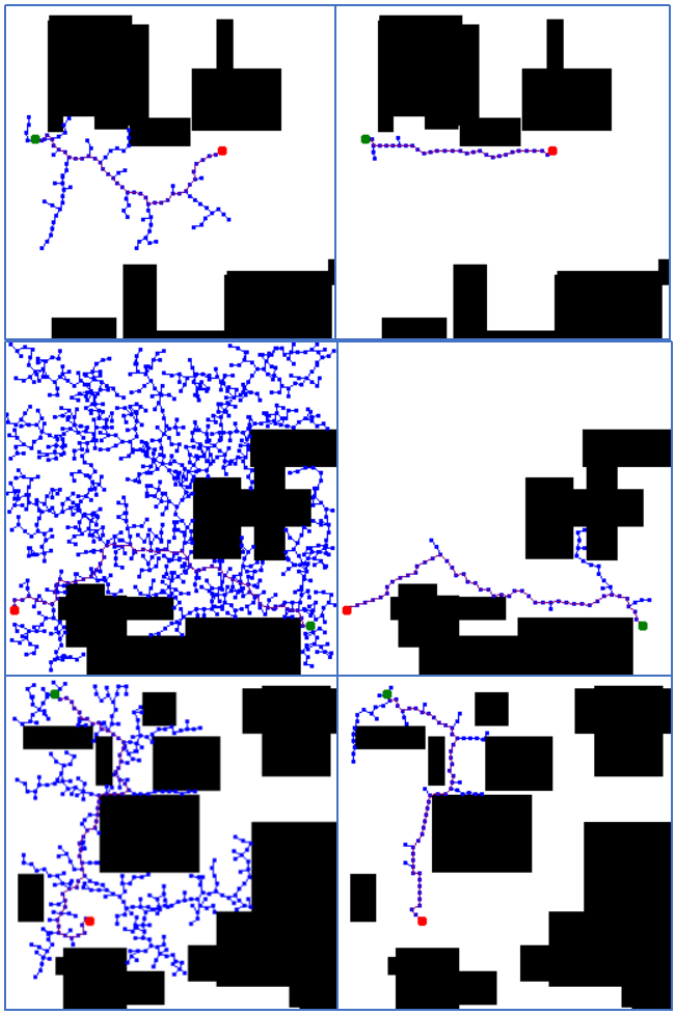

Fig. 4. Comparison of path planning results between traditional RRT (left images) and Region RRT (right images)

TABLE I
COMPARISON OF PATH PLANNING PERFORMANCE METRICS BETWEEN RRT AND REGION RRT ON TRAINING AND TESTING DATASETS

dataset	step	method	success rate	node num	path cost	planning time
training(1000 maps)	**3**	RRT	0.407	579.506	177.273	1.199
		Region RRT	**0.953**	**99.307**	**169.695**	**0.220**
	6	RRT	0.923	412.541	196.606	0.459
		Region RRT	**0.983**	**50.781**	**163.262**	**0.121**
testing(500 maps)	3	RRT	0.380	599.268	189.060	1.284
		Region RRT	**0.912**	**98.668**	**164.364**	**0.232**
	6	RRT	0.942	421.089	197.328	0.448
		Region RRT	**0.972**	**54.726**	**160.852**	**0.132**

3) Comparison of region-guided RRT and traditional RRT on node count, path length, planning time, and success rate: This section will compare the performance of region-guided RRT and traditional RRT on the training and validation datasets in terms of the number of nodes generated, path length, planning time per map, and success rate of finding a feasible path.Table I shows the experimental results, which indicate that Region RRT outperforms traditional RRT on all the performance metrics. Especially when the step size is small, the performance gap between Region RRT and traditional RRT becomes more significant.

V. CONCLUSION

We proposed a region-guided RRT algorithm that utilizes a neural network to predict regions of high value and assigns them higher weights to avoid excessive exploration of irrelevant areas. Additionally, we take advantage of the direct reachability between points on the predicted regions and the goal point to increase the sampling probability of the goal point, which accelerates the convergence of the algorithm. Experimental results show that our region-guided RRT algorithm outperforms the original RRT algorithm in all performance metrics.

However, our algorithm also has some limitations. For environments that the region generator has never seen before, the predicted regions may not be as precise, indicating that the generalization ability of our network is insufficient. In the future, we may consider using more data and data augmentation techniques to improve the generalization ability. Additionally, our algorithm may not perform well when the goal point is completely surrounded by obstacles.

REFERENCES

[1] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, vol. 149, pp. 270–299, 2020.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[4] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[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 *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[7] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[8] T. Zhang, J. Wang, and M. Q.-H. Meng, "Generative adversarial network based heuristics for sampling-based path planning," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 1, pp. 64–74, 2021.

[9] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural rrt*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.