2020 16th International Conference on
Control, Automation, Robotics and Vision (ICARCV)
December 13-15, 2020. Shenzhen, China

Bidirectional Homotopy-Guided RRT for Path Planning

Zhen Lin[1], Yanjun Li[2], Ji Xiang[1], Gui Ling[2,3] and Feiyang Suo[2,3]

***Abstract*— As a popular robot path planning algorithm, RRT (Rapid-exploring Random Tree) and various RRT-based extensions have achieved remarkable results. However, existing algorithms rely too much on randomness and often do not make use of known map information, this makes the growth of trees too blind. To this end, we introduce Bidirectional Homotopy-Guided RRT (BH-RRT) that combines Bidirectional RRT (Bi-RRT) with information obtained from obstacle contours. Compared with the previous methods, BH-RRT can reflect most of the map's information with a small number of feature points, and then form a set of points. This set can provide a better local goal point in each iteration, so that the tree can grow in a more favorable direction, rather than only being affected by the goal point. Experimental results show that BH-RRT outperforms RRT, HRRT and Bi-RRT in the success rate within a limited time.**

I. INTRODUCTION

In the past two decades, considerable progress has been made in the field of mobile robot. Path planning, as one of the prolonged topic for mobile robot research and development, widely used in various industrial fields, such as transportation [1], [2], game design [3], [4], and industrial robots [5]–[7]. The path planning algorithm of the robot is actually according to the constraints to find an optimal or suboptimal collision-free path in the reachable area from the starting point to the end point. The efficiency of path planning algorithm will significantly affect the performance of robots in completing tasks.

Path planning problems are commonly solved by grid-based searches or stochastic searches algorithms. Graph-based searches such as A* [8] and Dijkstra [9] has been studied for a long time, which can discretely find the best solution if a solution exists, otherwise returns failure. But these graph-based algorithms do not scale well to the problem of large state space, due to memory space and computation time grow exponentially. Stochastic searches, such as Rapid-exploring Random Tree (RRT) [10], Probabilistic Roadmap (PRM) [11], use sample-based methods to avoid discretization of state space and provide a weaker form of completeness. Among these sampling-based path planning algorithms, the RRT algorithm is one of the most representative, and its variants [12]–[15] achieve efficient search and planning in various applications.

Utilizing the information of known obstacles to drive the growth of RRT has made some progress. Voronoi-graph based method is widely used [16]–[19], which divides the obstacle space into many connected sub-regions. Given a starting point and an ending point, the subregion to which they belong can be determined and collision free paths can be quickly planned along the constructed edges. However, it needs a lot of extra time to generate and reprocess the Voronoi graph, and the generated paths sometimes introduce excessive additional costs due to its heavy reliance on obstacle information. In [20], [21], the topological constraints contained in homotopy information are used to limit the growth of the tree, and the reference frames that segment different homotopy classes are used to restrict the tree to belong to the same homotopy class as the pre-input path, thus reducing the sampling range and speeding up the search. However, these methods can only be carried out in the case of pre-input, too dependent on human participation.

In this paper, we present an efficient path planning method called Bidirectional Homotopy-Guided RRT (BH-RRT) that combines Bidirectional RRT (Bi-RRT) [22]–[26] and information obtained from obstacle contours. BH-RRT uses homotopy information to determine an invalid triangle area, and the feature points on the obstacles are used to perform sampling bias. Different from some previous methods, BH-RRT introduces points on obstacles to guide the tree to avoid the nearest obstacle, instead of simply using the goal point to guide the growth of the tree. Such a design makes the node more likely to explore the *local optimal path*, thereby reducing the number of nodes.

The remainder of this paper is organized as follows. In section II, we introduce the specific contents of RRT algorithm and Bi-RRT algorithm, which are both the basis of the proposed BH-RRT algorithm. Section III introduce proposes a new method to improve search efficiency, and introduces the concept of feature points. Section IV introduces the algorithm of BH-RRT and some details. Section V conducts experiments on different complex environments, and compares the results of different algorithms. Finally, in Section VI, we summarize this paper and put forward the prospect of future path planning.

*This work is supported by the Key R&D Program of Zhejiang Province (2019C02002), the Fundamental Research Funds for the Central Universities (2018XZZX001-06).

[1]College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China

[2]School of Information and Electrical Engineering, Zhejiang University City College, Hangzhou 310015, China

[3]College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China

Emails: 21910136@zju.edu.cn, liyanjun@zucc.edu.cn, jxiang@zju.edu.cn, 21932101@zju.edu.cn, 21932116@zju.edu.cn.Ji Xiang is the corresponding author.

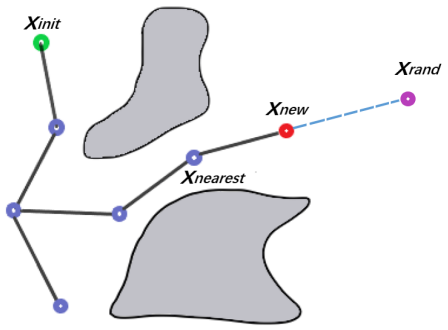


Fig. 1. Node generation using generic RRT

II. Background

We briefly review RRT algorithm and its bi-directional structure in this section.

A. RRT Algorithm

Standard RRTs algorithm constructs a tree using random sampling in search space, the tree starts from the initial point x_{init}, and expands to find the collision-free path toward the goal point x_{goal}, as shown in Fig. 1. Algorithm1 presents its pseudo code. In the initialization, x_{init} is stored in the vertices set V, and the edges set E is set as empty, V and E together constitute tree $\mathcal{T}$. Further detail of some major functions is described as the following:

$RandomSample$: For a given space configuration X, randomly select a point x_{rand} in the obstacle-free state space X_{free}.

$Nearest$: Find the nearest node $x_{nearest}$ to x_{rand} in the tree $\mathcal{T}$ according to a cost function.

$Steer$: Return a control input in the direction from $x_{nearest}$ towards x_{rand} at an increment distance Δq.

$Collosion\_free$: Determine whether the given path from $x_{nearest}$ towards x_{rand} passes through obstacle region X_{obs} or not.

$InsertNode$: Add the node x_{new} to vertices set V and add the edge that connect it with its parent x_{min} to edges set E, where V and E both in tree $\mathcal{T}$.

Algorithm 1 Basic RRT(x_{init})

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$; $\mathcal{T} \leftarrow (V, E)$
for $k = 0$ **to** K **do**
  $x_{rand} \leftarrow RandomSample()$
  $x_{nearest} \leftarrow Nearest(x_{rand}, \mathcal{T})$
  $x_{new} \leftarrow Steer(x_{rand}, x_{nearest})$
  if $Collision\_free(x_{new})$ **then**
    $\mathcal{T} \leftarrow InsertNode(x_{new}, \mathcal{T})$
    if $x_{new} == x_{goal}$ **then**
      $break$
return $\mathcal{T}$

B. Bidirectional RRT

The Bidirectional RRT (Bi-RRT) algorithm starts by growing two RRTs, one from x_{init} and another from x_{goal} , and this is depicted in Fig. 2. The two trees $\mathcal{T}_a$ and $\mathcal{T}_b$ (starting from x_{init} and x_{goal} respectively) are maintained at all times until they become connected and a solution is found.

Algorithm 2 depicts the algorithmic of the steps involved in Bi-RRT. In each iteration, we select a random point x_{rand}, and the nearest neighbor from the $\mathcal{T}_a$ to x_{rand} is termed as $x_{nearest}$. Then x_{new} generated according to x_{rand} and $x_{nearest}$, and if collision check is false, added to $\mathcal{T}_a$. The nearest neighbor from the $\mathcal{T}_b$ to x_{new} is termed as $x_{nearest1}$. After that $x_{nearest1}$ is going to keep expanding towards where x_{new} is until connected or encounter obstacles. At the end of each iteration, the tree with more nodes becomes $\mathcal{T}_b$ in the next iteration, and the other tree becomes $\mathcal{T}_a$.

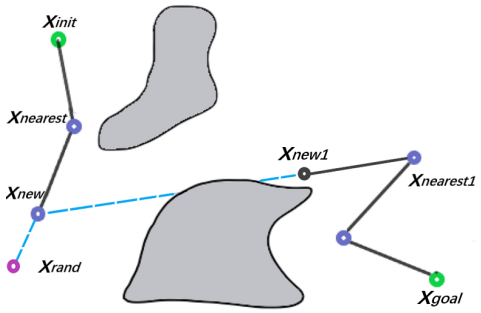


Fig. 2. Bidirectional RRT

Algorithm 2 Bidirectional RRT(x_{init})

$V_a \leftarrow \{x_{init}\}$; $E_a \leftarrow \emptyset$; $\mathcal{T}_a \leftarrow (V_a, E_a)$
$V_b \leftarrow \{x_{goal}\}$; $E_b \leftarrow \emptyset$; $\mathcal{T}_b \leftarrow (V_b, E_b)$
for $k = 0$ **to** K **do**
  $x_{rand} \leftarrow RandomSample()$
  $x_{nearest} \leftarrow Nearest(x_{rand}, \mathcal{T}_a)$
  $x_{new} \leftarrow Steer(x_{rand}, x_{nearest})$
  if $Collision\_free(x_{new})$ **then**
    $\mathcal{T}_a \leftarrow InsertNode(x_{new}, \mathcal{T}_a)$
    $x_{nearest1} \leftarrow Nearest(x_{rand}, \mathcal{T}_b)$
    $x_{new1} \leftarrow Steer(x_{new}, x_{nearest1})$
    if $Collision\_free(x_{new1})$ **then**
      $InsertNode(x_{new1}, \mathcal{T}_b)$
      while *not* $x_{new1} == x_{new}$ **do**
        $x_{new2} \leftarrow Steer(x_{new}, x_{new1})$
        $x_{new1} = x_{new2}$
        if $Collision\_free(x_{new1})$ **then**
          $InsertNode(x_{new1}, \mathcal{T}_b)$
        else $break$
    if $x_{new1} == x_{new}$ **then**
      return $\mathcal{T}_a$
  if $len(V_a) > len(V_b)$ **then**
    $Swap(\mathcal{T}_a, \mathcal{T}_b)$

III. Homotopy Bias

This section mainly discusses how to use homotopy information for sampling range analysis, and only the two dimensional state space is taken into considered.

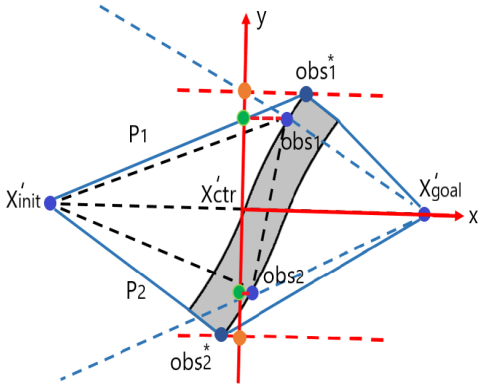


Fig. 3.

A. Single obstacle

Fig. 3 describes a path planning problem with known positions of starting point x'_{init}, ending point x'_{goal}, and the only one obstacle X_{obs} in the plane space. Then the coordinate system with the midpoint x'_{ctr} of x'_{init} and x'_{goal} as the origin is set up, where the direction pointing to x'_{goal} is the positive direction of the x-axis. Let obs_1 and obs_2 be two points arbitrarily taken on an obstacle. If there is a solution, let Pr(·) be the probability of getting the point on the optimal path in any homotopy class through random sampling in the plane space. Then we can get the following:

$$\frac{Pr(X_{free}-S')}{Pr(X_{free})}=\frac{X_{free}}{X_{free}-S'}\geq 1, \tag{1}$$

where S' be the set of all points in the area $X_{S'}$ formed by x'_{init}, x'_{goal}, obs_1 and obs_2. Since the plane space is uniformly distributed, the size of point set S' is proportional to $X_{S'}$. $X_{S'}$ can be calculated by

$$X_{S'}=\frac{1}{2}d(x'_{init},x'_{goal})|y(obs_1)-y(obs_2)|, \tag{2}$$

where $y(\cdot)$ is the value of the point in the y-axis direction of the coordinate system.

Since our goal is to maximize the effect shown in (1), we need to maximize $X_{S'}$. The *optimal feature points* on the obstacle that maximize and minimize $y(\cdot)$ can be found as obs_1^* and obs_2^*, then obs_1 and obs_2 are replaced by them respectively.

B. Multiple obstacles

More general, we extend this method to the state space with multiple obstacles $X_{obs}=\{X^1_{obs},X^2_{obs},...,X^n_{obs}\}$, as shown in Fig. 4. First, randomly take two points obs^i_1, obs^i_2 on each obstacle $X^i_{obs}\in X_{obs}$. Then, the intersection I can be obtained by

$$I=\{\forall X^i_{obs}\in X_{obs}, L(obs^i_1,obs^i_2)\cap L(x'_{init},x'_{goal})\}. \tag{3}$$

where $L(\cdot,\cdot)$ be the line segment between two points. There may be many elements in I, but in order to simplify subsequent calculations, we will select a point $p_j\in I$ closest to x'_{init}. The obstacle X^j_{obs} which contain the points obs^j_1, obs^j_2 is our direct obstacle, where $p_j\in L(obs^j_1,obs^j_2)$. According to the symmetry, we can also select the direct obstacle X^i_{obs} of x'_{goal}. Therefore, $X_{S'}$ can be regarded as composed of $X_{S'_1}$ and $X_{S'_2}$, where $X_{S'_1}$ is a triangular area composed of x'_{init},obs^j_1 and obs^j_2, and $X_{S'_2}$ is a triangular area composed of x'_{goal}, obs^i_1 and obs^i_2. Then we can find the *optimal feature points* obs^*_1, obs^*_2, obs^*_3 and obs^*_4 on the obstacles through the approach proposed in III-A.

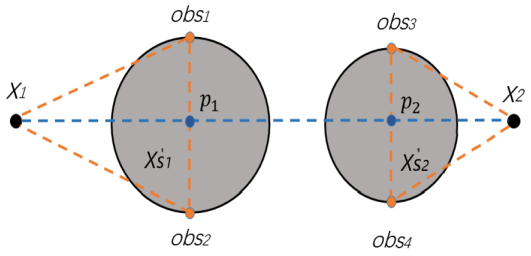


Fig. 4.

IV. Bi-directional Homotopy-Guided RRT

For path planning based on the bidirectional tree, in each iteration, the starting point of local path x'_{init} and the target point of local path x'_{goal} are different, and it requires a lot of calculation to constantly update the optimal feature points on each obstacle that maximize $X_{S'}$ given in (2) as the coordinate system is constantly changing. However, it can be seen from (1) that even if the two feature points are randomly selected, it can have a certain effect. Therefore, we only select the optimal feature points on each obstacle *in the initial stage*, as is shown in Fig. 5.

More details here are shown about how to select a better local goal point x''_{goal} by Algorithm 3. According to the method in III-B, we can get the obstacle X^c_{obs} in the direction from the starting point to the target point, where $\{obs^c_1, obs^c_2\}\in X^c_{obs}$ (Obs Algorithm 3), and X^c_{obs} is likely to be the obstacle closest to the starting point x'_{init} of the local path. How to use obstacle X^c_{obs} to determine the more reasonable local goal point x''_{goal} is a key point, and in this paper we simply choose among obs^c_1 ,obs^c_2 and x'_{goal}. This idea is the core part of the method in this paper.

Algorithm 3 New_Goal

Input: x'_{init}, x'_{goal}
Output: x''_{init}, x''_{goal}
while $Obs(x'_{init},x'_{goal})\neq\emptyset$ **do**
  $obs^c_1, obs^c_2 \leftarrow Obs(x'_{init},x'_{goal})$
  $x'_{goal}, ch \leftarrow Best(obs^c_1, obs^c_2, x'_{goal})$
if $ch == 1$ and $P_2(x'_{goal}) == 2$ **then**
  $x'_{goal} \leftarrow Nearest(x'_{init}, \mathcal{T}_b)$
$x''_{init} \leftarrow Nearest(x'_{goal}, \mathcal{T}_a)$
$x''_{goal} \leftarrow x'_{goal}$

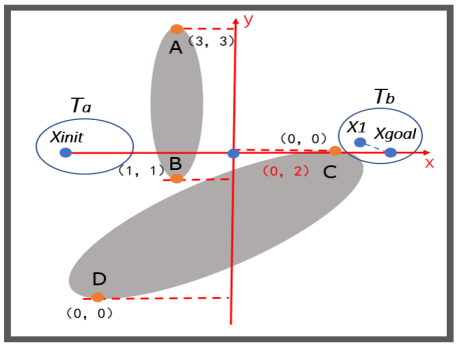


Fig. 5. Feature points select and priority setting of BH-RRT. The feature points (yellow) are all selected in the initial stage according to the x_{init} and x_{goal}, which may not be consistent with the optimal feature points when the starting and ending points of the local path are x_{init} and x_1. Each black bracket contain the initial priority of the feature point for $\mathcal{T}_a$ and $\mathcal{T}_b$ respectively, while the red one shows the priority change caused by the $\mathcal{T}_b$'s growth and proximity to the feature point C.

As shown in Fig. 5, we providing an *initial priority* for each feature point, and it is a critical step. For feature points that are not close to boundaries or other obstacles, set its *priority to 0* to indicate the highest priority, otherwise, the priority of the feature points that will lead the tree to the narrow channel is *set to 1*. Here we set the relative distance that can be considered close to the boundary and other obstacles to within 2 steps. If a feature point is approached by the node which lead by itself, i.e. the distance between the two is within a predetermined range, the priority of the feature point is *reduced to 2*. While the *lowest priority 3* is set for those feature points that are already on the boundary and do not have the possibility to guide the growth of Bidirectional tree.

Now we will explain how to choose a new goal point x''_{goal} for the local path planning. A heuristic cost function $Cost(\cdot)$ is introduced to calculate the heuristic cost of obs_1^c and obs_2^c:

$$Cost(p) = d(x_{new}, p) + d(x_{nearest1}, p) \tag{4}$$

The lower cost point is denoted as L and another point is denoted as H. Then x'_{goal} is updated according to the following Table I:

TABLE I

$P_1(H)$ \ $P_1(L)$	0	1	2	3
0	L	H	H	H
1	L	L	H	H
2	L	L	x'_{goal}	x'_{goal}
3	L	L	x'_{goal}	x'_{goal}

If x'_{goal} remains unchanged, $ch = 0$, otherwise $ch = 1$ (`Best` Algorithm 3). And $P_1(\cdot)$ be the priority of the feature point for the tree. It should be pointed out that for bidirectional trees, each side of the tree has its own priority.

In the bidirectional tree structure, in order to improve efficiency, x'_{init} and x'_{goal} are usually the closest points on the two trees to each other. And we use the method proposed above to update x'_{goal}, so this will also cause x'_{init} to update. We just update x'_{init} to the point closest to x'_{goal} on the tree where it is located, and after the loop structure, the final x'_{init} is denoted as x''_{init}.

Algorithm 4 Bi-directional Homotopy-Guided RRT(x_{init})

Input: $\mathcal{R}, x_{init}, x_{goal}, X_{obs}, max$
Output: $\mathcal{T}_a, \mathcal{T}_b$
$V_a \leftarrow \{x_{init}\}$; $E_a \leftarrow \emptyset$; $\mathcal{T}_a \leftarrow (V_a, E_a)$
$V_b \leftarrow \{x_{goal}\}$; $E_b \leftarrow \emptyset$; $\mathcal{T}_b \leftarrow (V_b, E_b)$
for $k = 0$ **to** max **do**
  $x_{rand} \leftarrow RandomSample()$
  $x_{nearest} \leftarrow Nearest(x_{rand}, \mathcal{T}_a)$
  $x_{new} \leftarrow Steer(x_{rand}, x_{nearest})$
  if $Collision\_free(x_{new})$ **then**
    $\mathcal{T}_a \leftarrow InsertNode(x_{new}, \mathcal{T}_a)$
    $x_{nearest1} \leftarrow Nearest(x_{new}, \mathcal{T}_b)$
    $x_{new}, x_{nearest1} \leftarrow New\_Goal(x_{nearest1}, x_{new})$
    $x_{new1} \leftarrow Steer(x_{new}, x_{nearest1})$
    if $Collision\_free(x_{new1})$ **then**
      $InsertNode(x_{new1}, \mathcal{T}_b)$
      while *not* $x_{new1} == x_{new}$ **do**
        $x_{new2} \leftarrow Steer(x_{new}, x_{new1})$
        $x_{new1} = x_{new2}$
        if $Collision\_free(x_{new1})$ **then**
          $InsertNode(x_{new1}, \mathcal{T}_b)$
        else *break*
    if $x_{new1} == x_{new}$ **then**
      if x_{new} in $\mathcal{R}$ **then**
        $P_2(x_{new}) = 2$
      else
        return $\mathcal{T}_a$,$\mathcal{T}_b$
  $Swap(\mathcal{T}_a, \mathcal{T}_b)$

The BH-RRT algorithm starts by growing two RRTs, looks like Bi-RRT in Section II-B, one from x_{init} and another from x_{goal}. Two trees $\mathcal{T}_a$ and $\mathcal{T}_b$ (starting from x_{init} and x_{goal} respectively) are maintained at all times until they become connected and a solution is found.

The BH-RRT has presented in Algorithm 4. HRRT is a single-tree version of BH-RRT, and the specific algorithm can be derived according to Algorithm 4, it would not be discussed in detail here.

V. Simulation Results

BH-RRT and HRRT were compared to RRT and Bi-RRT on a variety complex path planning problems Fig. 6, and we show the performance of different algorithms in three scenarios, and from the top row to the bottom row are Scenario 1, Scenario 2, Scenario 3, respectively. And the results obtained by each algorithm are in a separate column, for the bidirectional tree algorithm, we use different colors to distinguish the trees.

In Table II, the details corresponding to the three scenarios considered are captured.

For each scenario, above algorithms are executed to determine the success rate within the time limit, and the trees

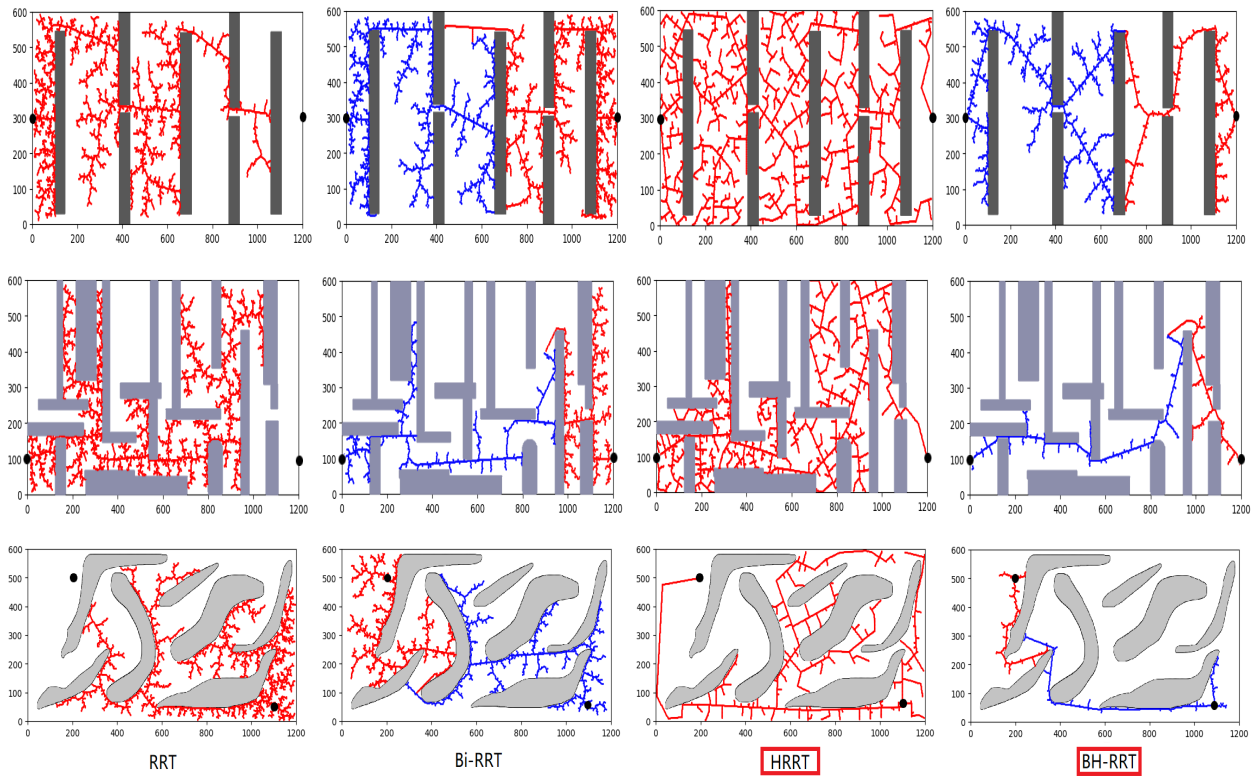
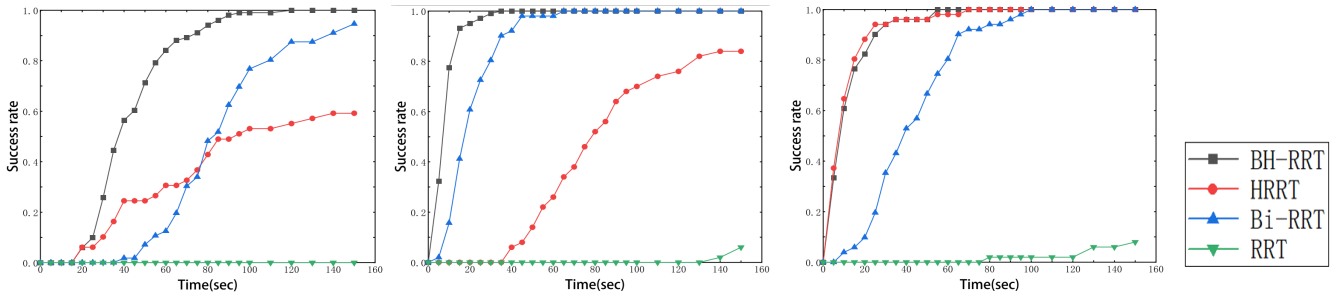


Fig. 6. Planning results of different algorithms(The red box indicates that this is the algorithm we proposed). The black dots represent the starting and ending points, and for the bidirectional versions, we use red and blue to represent different trees. Most of these results are selected from the samples where the algorithms successfully implement the path planning, but due to time constraints, the RRT algorithm has a high probability of not being able to plan the results in these scenarios.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d)

Fig. 7.

TABLE II
DETAILS OF THE TEST SCENARIOS

Sl. No	Start Point (x,y)	Target Point (x,y)	Step Size
Scenario 1	(0,300)	(1200,300)	5
Scenario 2	(0,100)	(1200,100)	5
Scenario 3	(1100,50)	(200,500)	5

generated by different algorithms. The size of scenarios is 1200×600. We set a maximum iteration limit of 50000 and the bias towards the goal was set to 0.5. The maximum time for a single run is 150 seconds. On this basis, the results of RRT, HRRT, Bi-RRT and BH-RRT algorithms are compared.

In Fig. 6, the results of Scenario 1 corresponding to above algorithms are presented. Note that, compared with RRT, HRRT makes use of feature points on obstacles to enable its tree to expand to most areas of the map faster, while Bi-RRT makes use of information interaction between the two trees to achieve efficient synchronous promotion of nodes. BH-RRT combines the advantages of these two, so that the generated tree will neither rely on random search or continuous approach along the wall when encountering large obstacles like Bi-RRT, nor generate a large number of nodes in unnecessary areas like HRRT. The results corresponding to Scenario 2 and Scenario 3 are presented in Fig. 6, which are similar to the result of Scenario 1.

Fig. 7 depicts the relationship between the success rate of each algorithm and the running time in three scenarios. We notice that the results are different in each scenario, HRRT and Bi-RRT under different scenarios each have advantages, but BH-RRT and RRT were basically the best and the worst, respectively. In Scenario 3, as shown in Fig. 7(c), when there are many obstacles but no narrow channels, the HRRT performance is even close to BH-RRT, because despite the introduction of feature points, HRRT still has a strong randomness, which is very important in this kind of environment. And the performance of Bi-RRT in Scenarios 2 is better, as shown in Fig. 7(b), this is because in between the two trees don't have many large obstacles bidirectional search can greatly improve efficiency.

VI. CONCLUDE

In this paper, we proposed an efficient path planning method. The proposed method estimates a better local goal point from feature points generated from the contours of obstacles and original local goal point. Because of the property of RRT, the probability completeness of the planned path are guaranteed. Experimental results demonstrate that our method outperforms other methods and can have a high success rate in a short period of time. Extending our method into a higher dimensional space and taking semantic context of the environment into consideration are our future works.

REFERENCES

[1] A. M. de Souza, R. S. Yokoyama, G. Maia, A. Loureiro and L. Villas, "Real-time path planning to prevent traffic jam through an intelligent transportation system," 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, 2016, pp. 726-731.

[2] Hyeonbeom Lee, Hyoin Kim and H. J. Kim, "Path planning and control of multiple aerial manipulators for a cooperative transportation," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 2386-2391.

[3] G. Foderaro, A. Swingler and S. Ferrari, "A model-based cell decomposition approach to on-line pursuit-evasion path planning and the video game Ms. Pac-Man," 2012 IEEE Conference on Computational Intelligence and Games (CIG), Granada, 2012, pp. 281-287.

[4] Qingchun Meng et al., "Game strategy based on fuzzy logic for soccer robots," Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0, Nashville, TN, 2000, pp. 3758-3763 vol.5.

[5] Z. Doulgeri and T. Matiakis, "A web telerobotic system to teach industrial robot path planning and control," in IEEE Transactions on Education, vol. 49, no. 2, pp. 263-270, May 2006.

[6] Sören Larsson, J.A.P. Kjellander, Path planning for laser scanning with an industrial robot, Robotics and Autonomous Systems, Volume 56, Issue 7, 2008, Pages 615-624, ISSN 0921-8890.

[7] Bing Fu, Lin Chen, Yuntao Zhou, Dong Zheng, Zhiqi Wei, Jun Dai, Haihong Pan, An improved A* algorithm for the industrial robot path planning with high success rate and short length, Robotics and Autonomous Systems, Volume 106, 2018, Pages 26-37, ISSN 0921-8890.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," TSSC, 4(2): 100–107, Jul. 1968

[9] Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1:269–271.

[10] Karaman, S. and Frazzoli, E. (2011) 'Sampling-based algorithms for optimal motion planning', The International Journal of Robotics Research, 30(7),

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", TRA, 12(4): 566–580, 1996.

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

[13] Y. Abou El Majd, H. E. Ghazi and T. Nahhal, "PaRRT: Parallel rapidly exploring random tree (RRT) based on MapReduce," 2017 International Conference on Electrical and Information Technologies (ICEIT), Rabat, 2017, pp. 1-5.

[14] C. Moon and W. Chung, "Kinodynamic Planner Dual-Tree RRT (DT-RRT) for Two-Wheeled Mobile Robots Using the Rapidly Exploring Random Tree," in IEEE Transactions on Industrial Electronics, vol. 62, no. 2, pp. 1080-1090, Feb. 2015.

[15] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma and N. Zheng, "A fast RRT algorithm for motion planning of autonomous road vehicles," 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, 2014, pp. 1033-1038.

[16] S. Garrido, L. Moreno, M. Abderrahim and F. Martin, "Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching," 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, 2006, pp. 2376-2381.

[17] S. R. Lindemann and S. M. LaValle, "Incrementally reducing dispersion by increasing Voronoi bias in RRTs," IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04.

[18] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space Voronoi bias," 2009 IEEE International Conference on Robotics and Automation, Kobe, 2009, pp. 2061-2067.

[19] D. Kim, J. Lee and S. Yoon, "Cloud RRT*: Sampling Cloud based RRT*," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 2519-2526.

[20] V. Narayanan, P. Vernaza, M. Likhachev and S. M. LaValle, "Planning under topological constraints using beam-graphs," 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, 2013, pp. 431-437.

[21] D. Yi, M. A. Goodrich and K. D. Seppi, "Homotopy-aware RRT*: Toward human-robot topological path-planning," 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Christchurch, 2016, pp. 279-286.

[22] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno and M. Pavone, "An asymptotically-optimal sampling-based algorithm for Bi-directional motion planning," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 2072-2078.

[23] A. H. Qureshi et al., "Adaptive Potential guided directional-RRT," 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, 2013, pp. 1887-1892.

[24] C. Moon and W. Chung, "Kinodynamic Planner Dual-Tree RRT (DT-RRT) for Two-Wheeled Mobile Robots Using the Rapidly Exploring Random Tree," in IEEE Transactions on Industrial Electronics, vol. 62, no. 2, pp. 1080-1090, Feb. 2015.

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

[26] D. Ghosh, G. Nandakumar, K. Narayanan, V. Honkote and S. Sharma, "Kinematic Constraints Based Bi-directional RRT (KB-RRT) with Parameterized Trajectories for Robot Path Planning in Cluttered Environment," 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 8627-8633.