2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

An Improved RRT-based Motion Planner for Autonomous Vehicle in Cluttered Environments

Mingbo Du, Jiajia Chen, Pan Zhao, Huawei Liang, Yu Xin, and Tao Mei

***Abstract*—In this paper, we present an improved RRT-based motion planner for autonomous vehicles to effectively navigate in cluttered environments with narrow passages. The planner first presents X-test that can identify passable narrow passages, and then perform an efficient obstacles-based extension operation within passable narrow passages. In order to generate a smooth trajectory for the vehicle to execute, a post-process algorithm with trajectory optimization is proposed. For the purpose of demonstrate benefits of our method, the proposed motion planner is implemented and tested on a real autonomous vehicle in cluttered scenarios with narrow passages. Experimental results show that our planner achieves up to 13.8 times and 7.6 times performance improvements over a basic RRT planner and a Bi-RRT planner respectively. Moreover, the resulting path of our planner is more smooth and reasonable.**

I. INTRODUCTION

With the constant pursuit of people for intelligent life, autonomous vehicles have become increasingly important research topic in various civilian and military operations. A fundamental part of the vehicle autonomy consists of motion planning and navigation system that enables it to safely drive through different types of environment. General motion planning methods have been studied in the literatures and it plays a vital role in different fields from computer animation, artificial intelligent to autonomous systems.

Intrinsically, a motion planning task can be stated as finding a path for a robot, such that the robot can move along this path from its initial configuration to goal configuration without colliding with any obstacles in the scene. Dijkstra's shortest path algorithm, potential fields and A* algorithm are traditional methods for motion planning [1]-[3]. In 2007 DARPA Challenge, a motion planning method based on Anytime Dynamic A* algorithm is well applied by CMU team [5]. Their approach generates good results in practice for urban driving, so that they finally won the competition. Sampling-based methods in robotic motion planning literatures, such as the Probabilistic RoadMap (PRM) [6] or Rapidly-exploring Random Trees (RRTs) [7], which have received a considerable amount of attention in recent, as they have been designed and successfully used to motion planning in a variety of environments [17]. In 2007 DARPA Urban Challenge, a RRT-based motion planner is developed by MIT team [8]. This method is not only effective for obstacle avoidance, but quickly finds a path to the goal configuration in the random steps. Thus, make them successfully complete the competition.

Figure 1. The autonomous vehicle named "Intelligent Pioneer II".

The above mentioned methods are effective for route maps, such as urban roads and highways. In cluttered environments with narrow passages or complex multi-obstacle environments, however, these methods are not suitable for motion planning since the limitations of their own. In these environments, quality of A* algorithm is largely restricted to the grid resolution. In addition, since the number of grid points grows exponentially with the dimensionality of the state-space, so does the running time of the algorithm. Yet, potential field methods easily suffer from local minima [4]. For RRT-based planning algorithms, their performance will degrade if the free space of the robot has narrow passages. Because of the sampling points are less likely to fall into the narrow passage, which hinders the tree to grow through the passage.

Motivations for the work presented here come from motion planning for autonomous vehicle in cluttered environment with narrow passages. Therefore, we present a new Improved RRT-based algorithm (which is abbreviated as

*This work was supported by "Key technologies and platform for unmanned vehicle in urban integrated environment", a National Nature Science Foundation of China (91120307).

Mingbo Du is a PhD student of the University of Science and Technology of China, Hefei 230027,China; Institute of Advanced Manufacturing Technology, Hefei Institutes of Physical Science, Chinese Academy of Sciences,Changzhou 213000, China.(E-mail: dumingbo@mail.ustc.edu.cn).

Jiajia Chen is a PhD student of the University of Science and Technology of China, Hefei 230027,China; Institute of Advanced Manufacturing Technology, Hefei Institutes of Physical Science, Chinese Academy of Sciences,Changzhou 213000, China.(E-mail: Nicky127@mail.ustc.edu.cn).

Pan Zhao is with the Institute of Advanced Manufacturing Technology, Chinese Academy of Sciences. (E-mail: qiushui@mail.ustc.edu.cn).

Huawei Liang is with the Institute of Advanced Manufacturing Technology, Chinese Academy of Sciences. (E-mail: hwliang@iim.ac.cn).

Yu Xin is a PhD student of the University of Science and Technology of China, Hefei 230027,China; Institute of Advanced Manufacturing Technology, Hefei Institutes of Physical Science, Chinese Academy of Sciences, Changzhou 213000, China. (E-mail: autoxinyu@qq.com).

Tao Mei is with the Institute of Advanced Manufacturing Technology, Chinese Academy of Sciences. (E-mail: tmei@iim.ac.cn).

IRRT in the following sections) in this paper, and develop a motion planner based on IRRT at the same time. This motion planner can efficiently generate a smooth and feasible trajectory in cluttered environments with narrow passages or complex multi-obstacle environments, and thus successfully solve the motion planning problem.

Specific contributions of this paper include:

- A novel X-test algorithm can identify narrow passages and estimate the width and depth of the difficult regions, which determine whether to meet driving requirements of the robot to further increase the recognition accuracy.
- We present a rapid and effective extension method for the narrow passages which greatly reduce the search time.
- A reasonable and effective post-process algorithm is proposed, which makes the resulting path can be directly executed by the vehicles to guarantee the integrity and practicality of the planner.
- We have implemented our motion planner and test the real performance of the planner on our autonomous vehicle "Intelligent Pioneer II" (see Fig.1). In practice, our algorithm improves the performance by 13.8 times and 7.6 times as compared to basic RRT planner and Bi-RRT planner respectively.

The rest of the paper is organized as follows. In Section II, we provide a brief survey of related work in sampled-based planning. In Section III, we present our improved RRT-based algorithm, and post-process algorithm is described in Section IV. Experimental results of the motion planner in a real autonomous vehicle and comparison with reference methods are given in Section V.

II. Related work and Preliminaries

In this section, we will formulate the motion planning problem and then discuss prior work on RRT which especially be designed to efficiently handle the complex environment with narrow passages.

A. Motion planning problem formulation

For clarity, we give some definitions and terms at first. For a mobile robot with n degrees of freedom (DOFS), its configuration can be represented as a point in an n-dimensional space $\boldsymbol{C}$, called the configuration space. A configuration q is free, if the robot placed at q has no collision with the obstacles or with itself. We define the free space $\boldsymbol{F}$ to be the set of all free configurations in $\boldsymbol{C}$. The obstacle space $\boldsymbol{B}$ is defined to be the complement of $\boldsymbol{F}$: $\boldsymbol{B}=\boldsymbol{C}/\boldsymbol{F}$. Let the starting configuration be $q_{start} \in \boldsymbol{F}$ and the goal configuration be $q_{goal} \in \boldsymbol{F}$. Thus, the motion planning problem is to find a path connecting q_{start} with q_{goal}, that is, a continuous collision-free path $\boldsymbol{\pi}:[t_0, t_f] \rightarrow \boldsymbol{F}$, such that $\boldsymbol{\pi}(t_0) = q_{start}$ and $\boldsymbol{\pi}(t_f) = q_{goal}$, where t_0 is the starting time and t_f is the final time.

Though the motion planning problem is an interesting thing from a practical view, the problem is still known to be computationally challenging, especially in complex environment. In order to achieve computational efficiency, sampling-based approaches have been extensively used.

Algorithm 1 IRRTExtend (q_n , q_r)

1. $q_r \leftarrow$ a randomly generated configuration in $\boldsymbol{C}$
2. $q_n \leftarrow$ the nearest neighbor of q_r
3. **if** (q_n ,q_r) is a collision-free line **then**
4. **return** $q_{new} \leftarrow$ Standard RRTExtend(q_n ,q_r)
5. **end if**
6. $q_c \leftarrow$ the obstacle boundary point from q_n to q_r
7. $q_d \leftarrow$ X-test(q_c ,q_r)
8. **if** q_d is NULL **then**
9. **CONTINUE**
10. **end if**
11. Standard RRT Extension(T, q_d)
12. **return** $q_{new} \leftarrow$ Obstacles-based Extension (q_d , d)

Rapidly-exploring Random Trees (RRTs) have been widely applied in single-query problems, while Probabilistic RoadMaps (PRMs) are useful for multi-query problems [17].

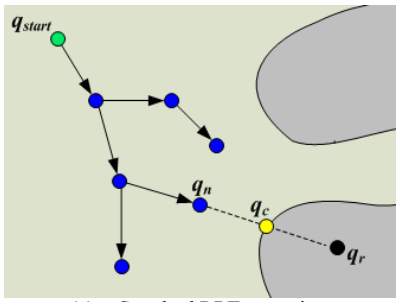
B. RRT and Narrow passages problem

At first, we give a brief description on RRT algorithm. It incrementally searches $\boldsymbol{C}$ from starting configuration q_{start} (the root node of search tree) by iteratively selecting a random configuration q_{rand} followed by finding $q_{nearest}$, the nearest node in the tree to q_{rand}, and steering $q_{nearest}$ towards q_{rand} to produce q_{new} which is then added to the tree. More specifically, when $q_{nearest}$ is selected, an Extend operation is executed on $q_{nearest}$ towards q_{rand} which iteratively takes steps based on the environmental resolution, checking validity at each step. This continues until either a tolerance distance, the $\boldsymbol{B}$ boundary, or q_{rand} is reached. Even though, RRT have drawbacks in passing through narrow spaces of the environment.

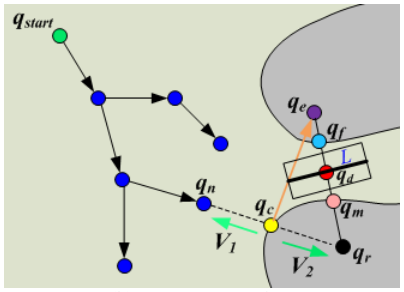
In fact, if the path to find in $\boldsymbol{C}$ passes through a narrow passage, the algorithm has to sample configurations inside the small region, impacting the connectivity of $\boldsymbol{F}$. However, it is difficult because the volume of the narrow passage is small compared with $\boldsymbol{C}$.

In order to efficiently handle this challenge, many strategies have been presented to improve the performance of RRTs for narrow passages. They include utilizing a bidirectional search with two trees (Bi-RRT)[7], using the work space obstacle geometry information to guide the sampling (OBRRT) [9], adaptive sampling strategies towards more critical regions[10], [11], biasing sampling techniques[12],[13], etc.

Particularly,Retraction-based methods [14],[15],[16],[20], [21], have shown great success in handling the environment with narrow passages. One approach, is called as Selective Retraction-based RRT, filters the use of Retraction-based RRT by taking a possibly expensive Bridge-line test, to determine whether a node of the tree is within or near a narrow passage [16].Our idea is just inspired by it. The biggest difference is that our method can not only identify the narrow passage, but also can estimate the width and depth of the narrow passage which analysis whether it can satisfy the conditions of traffic. Moreover, in this special region, we also propose a novel and inexpensive extension method to meet the requirements of practical application.



(a) Standard RRT extension



(b) IRRT extension

Figure 2. (a)Given a randomly in-colliding sample q_r, the standard RRT extension strategy grows the tree from its nearest node q_c to q_r, stopping at the boundary point q_c. (b)In our IRRT extension, we first identify the narrow passages by performing X-test. Note that (b) also is a illustration of X-test method.

Algorithm 2 X-test(q_c , q_r)

1. d ← the length of line (q_c , q_r)
2. β ← select random direction except $\overline{q_c q_r}$ and $\overline{q_c q_n}$
3. Create a line based on β , the size of robot
4. **if** CollisionPath($\overline{q_c q_e}$) AND
5. d > Distance(q_m , q_r) **then**
6. **if** intersecting line L is collision-free AND
7. Distance(q_m , q_f) > $d_{threshold}$ **then**
8. **return** q_d ← Midpoint(q_m , q_f)
9. **end if**
10. **end if**
11. **return** NULL

Algorithm 3 Obstacles-based Extension (q_d , d)

1. q_d ← a configuration located in narrow passage
2. θ ← select random direction based on goal configuration
3. q_t ← Extend a configuration based on a fixed step size d and θ
4. **if** Distance(q_t, ***B***) < σ **then** //σ is safety distance
5. CONTINUE
6. **end if**
7. **return** q_t

III. IMPROVED RRT-BASED ALGORITHM

In this section, we present our improved RRT-based algorithm which is designed for efficiently handling narrow passages.

It is necessary to add some basic constraints, including the width and depth of narrow passages, to identify the narrow passage, since our intention is for the vehicle motion planning in this environment. In order to avoid spending the sampling time on those impassable regions, we present a X-test method, which is inspired by the bridge line-test [11].

A. Improved RRT-based Extension

We first explain our extension method of IRRT, whose pseudo code is at Algorithm 1. When a random configuration q_r and its nearest neighbor node q_n are computed, we perform our extension method, IRRTExtend(q_n,q_r), shown in Algorithm 1. In the first step, if the line of (q_n,q_r) is collision-free, the extension method is the same to the extension method of the standard RRT explained in Sec. II-B. Otherwise, we perform X-test to decide whether a passable narrow passage exists. If it indicates that there is one, we find the estimated entrance point of the narrow passage. Subsequently, we can perform a new extension method in this special case.

B. X-test

The X-test algorithm not only determines whether a passable narrow passage exists, but find out the estimated entrance point of the narrow passage. Its pseudo code is shown in Algorithm 2. In order to perform the X-test, we first create a line segment whose one end is located at q_c and the length is related to the size of robot. Instead of generating the line direction in a uniform manner, we consider local information around q_c to make the line direction to through narrow passages with higher probability. Since the line segment between q_c and q_n is collision-free, we can think that there are no narrow passages along this direction, which is v_1=(q_n-q_c) (see Fig. 2(b)). In like manner, then, the line segment between q_c and q_r is in ***B***, we can also think that there are no narrow passages along the direction, which isv_2=(q_r-q_c) (see Fig.2(b)). This local information guides us to select a random line direction between these two line directions.

After constructing a line whose two end points are q_c and q_e, we check whether there is a collision at point q_e. If there is a collision, we treat the region around these q_c and q_e as they are in a narrow passage, and thus perform the succeeding operation, which will return the estimated entrance point q_d of the narrow passage or null. When connect the configuration q_e and the configuration q_r, we can obtain two obstacle boundary points q_m and q_f. If the line segment between q_m and q_f is collision-free and their length is longer than $d_{threshold}$, which is equal to the sum of the width of robot and safety distance, we can check the intersecting line L of $\overline{q_m q_f}$ whether there are any collisions in L. Note that we generate the

Algorithm 4 Pruning(T)

```
T ← obtained from improved RRT-based
Var Q, Q': Path
Q(q0, q1,…, qN) = Path(T)
qtemp ← q0
Q'.AddNode(q0)
While qtemp != qN do
  for each node qi∈qtemp, … , qN do
    if Collision (qtemp, qi) then
      qtemp = qi-1
      Q'.AddNode(qtemp)
      break
    end if
Q'.AddNode(qN)
return Q'
```

direction of line L in a specialized manner, selecting the vertical direction in a higher probability. The main aim of checking the collisions in such a way is that we can identify a passable narrow passage and exclude those fake passages. And then we call this X-test, because the line segment $\overline{q_m q_f}$ and line L resemble a capital letter X.

C. Obstacles-based Extension

Intuitively, if we expand from a configuration in free space, we should apply a simple growth method such as randomly selecting a direction. However, if we expand within a narrow passage, expensive growth method may be more useful. In these special regions, then, we can take advantage of the information of obstacles to do more efficient extension. Therefore, we present a new extension method, Obstacles-based Extension (q_d, d), shown in Algorithm 3. Our method is based on the information of goal configuration and the information of obstacles, such that, once a configuration is identified in a narrow passage one can perform a extension with a fixed length size d. Note that, the newly generated configuration must keep a safe distance between it and obstacles. The main reason of extending the configuration in this way is that it not only meets the constraints of obstacles space, but can speed up the progression of our algorithm in constrained regions.

IV. Post-Process Algorithm

It is well known that RRT-based planners often generate jerky and unnatural paths that may contain unnecessary turns, or that the velocities at the points may change arbitrarily. All of these issues are bad for the vehicle to execute. Especially, these issues become more significant when the environment has narrow passages as the narrow region for path planning becomes more constrained [18]. Therefore, a post-process algorithm is proposed for generating smooth paths. And this algorithm consists of two stages: pruning stage and smoothing stage.

A. Pruning Algorithm

As mentioned earlier, in order to address the unnecessary turning points problem, we introduce our pruning algorithm, *Pruning (T)*, shown in Algorithm 4.

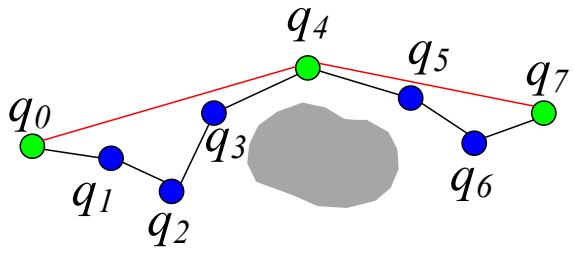


Figure 3. Pruning Algorithm: Due to the line segment (q_0,q_4) and (q_4,q_7) are collision-free, then the unnecessary nodes q_1,q_2,q_3,q_5,q_6 can be discarded and the critical nodes q_0, q_4,q_7 are retained.

Algorithm 5 IRRT Planner

```
tree T.init (qstart)
Repeat
  qr ← RandomNode( );
  qn ← NearestNeighbor(qr, T)
  qnew ← IRRTExtend(qn , qr)
  T.AddNode(qnew); T.AddEdge(qn , qnew)
Until find a collision-free path from qstart to qgoal
Q ← Pruning(T)
return S ← Bezier(Q)
```

In this algorithm, we need to take into account the obstacles constraint, which is caused by the change of connection of node. Then, we make a detailed introduction of our pruning algorithm.

We can obtain an ordered tree T= {q_0, q_1, q_2, ... , q_N} (q_0 is the initial node and q_N is the goal node) from the aforementioned IRRT. In order to discard the unnecessary turning nodes, we apply greed strategy in this algorithm. Let q_{temp} represents q_0 and start from q_{temp}, we orderly try to connect to q_1, q_2, ... , q_N, till the line segment between q_{temp} and q_i has a collision with obstacles. Then we can replace the zigzag path between q_{temp} and q_{i-1} with the straight line. Observe from Fig.3 that the unnatural nodes (q_1, q_2, q_3) between q_0 and q_4 are discarded and the critical nodes are retained. Thus the issue of unnatural and unreasonable paths is effectively solved.

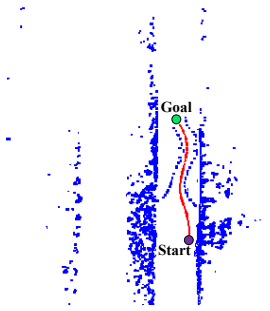
B. Path Smoothing

In this final stage, we need to handle the path from the aforementioned operation to generate a smooth and continuous-curvature path. By the experiences of experiments, we use the method of Bezier Curve to smooth the path because it's inexpensive overheads and well geometric property. Bezier Curve is defined as follows: we construct n-stage Bezier Curve by n+1 control points P_i (i=0,1,…,n), the equation is:

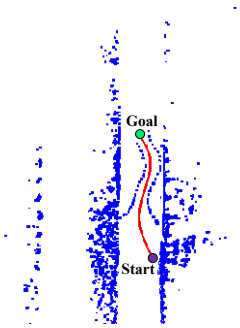
$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t) \quad (0 \le t \le 1) \qquad (1)$$

(a) Experiment environment (b) Corresponding perception map

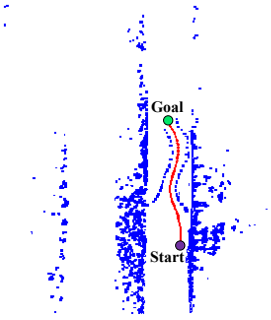
Figure 4. Outdoor autonomous driving experiments to evaluate the proposed method



(a) RRT planner



(b) Bi-RRT planner



(c) IRRT planner

Figure 5. Final path after optimization for different planners

Where $B_{i,n}$ is the Bernstein primary function, and the equation is shown as follows:

TABLE I. RESULTS OF THREE DIFFERENT PLANNERS

Planner	*Success-rate*	*nodes*	*time in ms*
IRRT	100%	457(21)	630.19 (65.28)
RRT	87%	6312(220)	2167.33(901.32)
Bi-RRT	100%	3506(134)	1125.61(500.13)

$$B_{i,n} = C_n^i t^i (1-t)^{n-i} \tag{2}$$

In order to use easily, the equation of Bezier Curve turns into the following form of recursion:

$$P_i^k = \begin{cases} P_i & k = 0 \\ (1-t)P_i^{k-1} + tP_{i+1}^{k-1} & k = 1,2,\ldots,n,\ i = 0,1,\ldots,n-k \end{cases} \tag{3}$$

Where $t = m/M$, M is the number of discrete nodes in the path.

According to our experimental results (Section V), it can well satisfy the vehicle dynamics by using the smoothing method of Bezier Curve.

C. Overall Algorithm

A pseudo code of the overall algorithm of our IRRT planner is shown at Algorithm 5. We generate a random sample q_r and find out its nearest neighbor node q_n. Then we perform our extension algorithm, IRRTExtend(·), and update the tree T. We iteratively perform these steps until we find a collision-free path between the start and goal configurations. Finally, we perform the post-process algorithm to obtain a smooth path S for the vehicle to execute.

V. EXPERIMENTAL RESULTS

A. Experimental Condition

We have implemented the motion planner in our autonomous vehicle platform, called "Intelligent Pioneer II"(see Fig. 1). It is equipped with a GPS/INS receiver, three LIDAR sensors (two Sick LMS, one Velodyne HDL-64), and three cameras [19]. We did some experiments for outdoor autonomous traveling through the cluttered environment with narrow passages. And we implement our method and compare this method with basic RRT and Bi-RRT. Fig. 4(a) shows the experiments environment for autonomous driving with sizes 3.3 x 45 meters and Fig. 4 (b) depicts the corresponding perception map. In Fig. 4 (b), the white area indicates safety area and blue area indicates obstacles area or infeasible area.

In the cluttered environment, we set a start point (purple point) and a goal point (green point). All algorithms are implemented in Visual C++ on an Intel i7 computer that has 2.53 GHz CPU.

B. Evaluation Results

The final paths after optimization and smooth of three different methods are shown in Fig.5. Observe from Fig.5 that our planner is obviously more reasonable than the other two planners, RRT planner and Bi-RRT planner. The

resulting path of our planner (Fig.5 (c)) is almost in the middle of the sinuous road. However, the path of RRT planner (Fig.5 (a)) is close to the obstacles and the path of Bi-RRT planner is much sharper (Fig.5 (b)). Moreover, we also compare the relevant characteristics of our planner with the other two planners, which is shown in TABLE I. Note that, in the third column of Table I, the number of retained nodes after pruning is listed in parentheses. And in the fourth column, the time cost of according planner expansion is also listed in parentheses. As RRT is a randomized algorithm, we performed the same experiment 500 times for three different methods respectively and compared their average characteristics. Firstly, in the term of success-rate, IRRT planner and Bi-RRT planner are all 100%, but the success-rate of RRT is only 87%. Secondly, the number of nodes of IRRT planner is drastically less than the others. And the unnecessary nodes are effectively discarded by Pruning Algorithm, the few critical nodes are reserved. From the expansion time, our planner shows 13.8 times and 7.6 times improvement over RRT planner and Bi-RRT planner respectively.

VI. Conclusion And Future

In this paper, we present an improved RRT-based motion planner to further the overall performance of RRT planners in cluttered environment with narrow passages. Our planner can identify the passable narrow passages by using X-test, and then perform the obstacles-based extension within the difficult regions. After finding an original path from start point to goal point, we handle the path by post-process algorithm to obtain a smooth path. We have implemented this method and applied it for a real autonomous vehicle in a difficult scenario. Our experimental results show that our planner significant speedups over prior RRT planners in cluttered environment.

There are several limitations of our planner. Though our method can handle the narrow passages for autonomous vehicle, the total computation time can't well satisfy the real-time (Table I). The main reason is that Pruning Algorithm of the post-process spends large time on collision detection. Furthermore, to some extent, our method is lack of robust due to the interference of environment noises.

There are many avenues for future research work. We would like to take the RRT as a tool of exploring the information of space to combine with other intelligent algorithms, such as machine learning and neural fuzzy network. Moreover, we would like to further enhance the robustness of our method. Finally, we also would like to apply our X-test method to PRM planners.

Acknowledgment

The authors gratefully acknowledge the help of our team members, particularly Hu Wei and Jun Wang for developing the perception map, Yan Song for his contributions to the low-level controller.

References

[1] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research,* 5(1):90-98, 1986

[2] E.W. Dijkstra. A note on two problems in connection with graphs. Numerische Mathematik, 1:269-271,1959.10.1007/BF01386390.

[3] D. Dolgov, S. Thrun, M. Montemerlo, J. Diebel, "Practical search techniques in path planning for autonomous driving", *Proc. of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics*(STAIR-08), June 2008.

[4] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE Conference on Robotics and Automation*, 1991

[5] M. Likhachev, D. Ferguson, G. Gordon, S. Thrun, A. Stenz, "Anytime dynamic A*: An anytime, replanning algorithm", *Proc. of International Conference on Automated Planning and Scheduling*, 2005.

[6] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation* 12(4): 566-580,1996.

[7] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2000

[8] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, J. P. How, "Motion planning for urban driving using RRT", *Proc. of IEEE/RJS International Conference on Intelligent Robots and Systems*, pp 1681-1686, 2008.

[9] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato. An obstacle-based rapidly-exploring random tree. *In Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2006

[10] T. Simeon, J. P. Laumond, and C. Nissoux, "Visibility based probabilistic roadmaps for motion planning", *Advanced Robotics Journal*, vol. 14, no.6, 2000

[11] D. Hsu, Tingting Jiang, J. Reif, and Zheng Sun, "The bridge test for smapling narrow passages with probabilistic roadmap planners", in *IEEE ICRA*, 2003, vol.3, pp.4420-4426 vol.3.

[12] A. Yershova, L. Jaillet, T. Simeon, and S.M. LaValle, "Dynamic-domain rrts: Efficient exploration by controlling the sampling domain", in *IEEE ICRA*, 2005, pp.3856-3861.

[13] S. R. Lindemann and S.M. LaValle, "Incrementally reducing dispersion by increasing voronoi bias in rrts", in *IEEE Int. Conf. on Robotics and Automation*, 2004, pp. 3251-3257.

[14] L. Zhang and D. Manocha. An efficient retraction-based RRT planner. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2008.

[15] J. Pan, L. Zhang, and D. Manocha. Retraction-based RRT planner for articulated models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2529-2536, 2010.

[16] J. Lee, O. Kwon, L. Zhang, and S. Yoon. SR-RRT: Selective Retraction-based RRT planner. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2543-2550, 2012.

[17] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006

[18] J. Pan, L. Zhang, and D. Manocha. Collision-free and smooth trajectory computation in cluttered environments. *The International Journal of Robotics Research*. 2012, 31(10):1155-1175.

[19] Tao Mei, Huawei Liang et al. Development of 'Intelligent Pioneer' unmanned vehicle [J]. *Intelligent Vehicles Symposium (IV)*, 2012 *IEEE*. 2012: 938 – 943.

[20] M. Saha, J. Latombe, Y. Chang, Lin and F, Prinz, "Finding narrow passages with probabilistic roadmaps: the small step retraction method", *Intelligent Robots and Systems,* vol.19, no.3, pp.301-319, Dec 2005.

[21] Ian Garcia and Jonathan P. How, "Improving the efficiency of rapidly-exploring random trees using a potential function planner", *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference* on(pp:7965-7970).