Gradient descent in sample-based single-query path planning algorithm

Rajaneesh Acharya
Department of Electrical and Electronics
National Institute of Technology
Surathkal, Karnataka, India
Email: rajaneesh.u.acharya@gmail.com

Debashisha Jena
Department of Electrical and Electronics
National Institute of Technology
Surathkal, Karnataka, India
Email: bapu4002@gmail.com

***Abstract*—Rapidly-exploring random tree (RRT) is a sampling based single query algorithm that is used to generate a path. This papers describes RRT algorithm, modifies RRT algorithm by applying gradient descent method for generating the path. Simulations are run in iterations and the results are compared with and without gradient descent. Finally this paper concludes with advantages and limitations of the gradient approach.**

***Index Terms*—RRT, PRM, RRT*, RRT-Connect, Lazy RRT**

I. Introduction

Algorithms play an important role in path planning. They provide a way to explore the geometric space in formulating road-map and paths while connecting the start and end points. These road-maps and paths form the basis of motion planning for agents in the field of robotics and unmanned vehicles. Different types of algorithms have been proposed and researched extensively in the past. Few of the algorithms use visibility graph, grid-based approach to build a graph. Few of the algorithms like Dijkstra use cost function and span the geometric or configuration space. Other algorithms like A* use heuristic along with cost function to explore the configuration space. These algorithms provide solutions that are repetitive in nature and generate an optimal solution. However, they take more time to converge at a solution. They are computationally expensive while dealing with a problem in higher dimension space.

Sampling-based algorithms [1] provide a path and offer time-saving for a complex scenario. Some of them are probabilistically complete, some are asymptotically optimal. These algorithms use a distribution and sample the configuration space randomly. These algorithms also provide a mechanism for growing a graph or tree by connecting a set of sample points or derived points. The algorithm grows the tree from source to destination using certain strategies and creates a path. Depending on whether a road-map or a path is generated, the sampling-based algorithms are classified as multi-query algorithms or single-query algorithms. In multi-query path planning algorithms (Probabilistic Road-map PRM [1], Probabilistic Road-map star PRM* [3]) road-map is first generated and then the road-map is queried to find a suitable path. In a single-query algorithm (Rapidly-exploring Random Tree RRT [2] [4] [5], Rapidly-exploring Random Tree star RRT* [3]) path is generated directly.

RRT is a single-query algorithm and uses a sampling strategy to generate samples for path planning. There are many sampling strategies available like uniform sampling, Gaussian sampling, Halton sequence etc. [1]. However, narrow passage [7] [8] presents a challenge for RRT algorithm due to spare points in the passage. Retraction [9–11] technique and Bridge [7] technique have been researched to handle this issue.

This paper proposes the application of gradient descent method to RRT algorithm for retraction of the points. The approach uses multivariate Gaussian distribution [12] to create a gradient. Instead of discarding the points as in RRT algorithm, this algorithm makes use of all the points providing a saving on collision check. While growing the tree, if the points are residing in obstacle configuration space, it applies a gradient descent method to move the point to a free configuration space. The approach is run in multiple iterations and a comparative study is made with respect to RRT. This approach could be used for other algorithms derived from RRT, like RRT*, RRT-Connect etc.

This approach applies gradient descent on RRT to check

- Usage of gradient descent method to move points from obstacle configuration space (C_{obst}) to free configuration space(C_{free}).
- Time to yield path.
- Compare the algorithm with RRT.
- Limitations of the approach.

II. Backround

(A) Problem Statement:

Given a configuration space C, modify RRT algorithm to generate a path from an initial configuration to a goal configuration by generating samples in configuration space. The path that is generated should be in C_{free}. Any of the derived points generated by the algorithm in the

978-1-5386-1138-8/18/$31.00

obstacle configuration space C_{obst} should be moved to C_{free} using gradient descent method. Gradient descent of points should satisfy the following equations

1) $x_{new} = x_{old} - \frac{f(x,y)}{\frac{\partial f(x,y)}{\partial x}}$.
2) $y_{new} = y_{old} - \frac{f(x,y)}{\frac{\partial f(x,y)}{\partial y}}$.

while minimizing the function $f(x, y)$.

Simplifying the concept further, to attain minimum $f(x, y)$, a negative value of $\frac{\partial f(x,y)}{\partial x}$ should increase the value of x and positive value of $\frac{\partial f(x,y)}{\partial x}$ should reduce x. This statement is true for the value of y.

(B) RRT Algorithm:

Rapidly-Exploring Random Trees(RRT) is a path planning algorithm, that grows a tree G which contains nodes V that is randomly generated points by extending the existing tree adding suitable edges E. This algorithm does not create a road-map like multi-query algorithms, hence avoiding closed cycles. The algorithm was introduced by S. M. LaValle [2], and forms a basis of many single-query algorithms like RRT*, RRT-Connect, Lazy RRT. Since random nodes are added to the tree during the path generation, the nodes can be checked on the fly, if they reside in C_{obst} or C_{free}. Hence RRT [1, 3–5] is suitable for dynamic path planning. The RRT has a Voronoi bias growing in the larger Voronoi region, though it is sub-optimal, it is probabilistic complete algorithm.

The pseudo-code of the algorithm is given in Figure 1(a). The algorithm initializes the tree with a null edge set E and starting point q_{init} in set V. It then iterates through a loop which does the following steps

- Generates a sample point q_{rand} using uniform sampling.
- Searches the existing tree in V and finds a vertex $q_{nearest}$ that is closest to q_{rand}
- Extends the tree from $q_{nearest}$ towards q_{rand} by a factor σ as in steering function. The new set of point derived by this extension in the configuration space C is called q_{new}. Figure 1(b) shows the process involving the growth of the RRT tree.
- A check is done to verify if q_{new} resides in obstacle configuration space C_{obst} or free configuration space C_{free}.
- If q_{new} resides in C_{obst}, the sample is discarded and a new sample is taken. Similarly, if the line connecting $q_{nearest}$ to q_{new} intersects any C_{obst}, the sample is discarded and iteration continues with new sample.
- q_{new} is added to the vertex set V and the edge set E is updated with the connection from $q_{nearest}$ to q_{new}.
- The iteration continues for a specified number of time or when the algorithm grows the tree towards q_{final} within an acceptable limit.

(C) Gradient Descent:

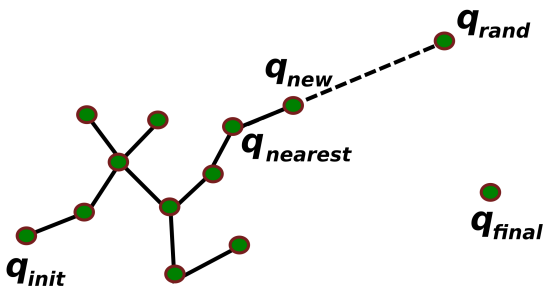
Algorithm Body of *RRT* algorithm

```
V ← {q_init}; E ← ∅; i ← 0;
while i < N do
    G ← (V, E);
    q_rand ← Sample(i); i ← (i + 1);
    (V, E) ← Extend(G, q_rand);
end while
```

Algorithm Body of $Extend(G, q)$

```
V' ← V; E' ← E;
q_nearest ← Nearest(G, q);
q_new ← Steer(q_nearest, q);
if ObstacleFree(q_nearest, q_new) then
    V' ← V' ∪ {q_new};
    E' ← E' ∪ {q_nearest, q_new};
end if
return G' = (E', V')
```

(a) RRT Psuedo Code [1]



(b) RRT Extend Function

Fig. 1: RRT Algorithm

The simple gradient descent algorithm [12] takes a function $f(x, y)$ where $x, y \in R^2$ and tries to minimize $f(x, y)$ in iterations by computing value of x and y. If the gradient $\frac{\partial f(x,y)}{\partial x}$ at a given point x is positive, then we are ascending the function $f(x, y)$ in the direction of X-axis. The gradient -$\frac{\partial f(x,y)}{\partial x}$ gives a descent in the function value. The same applies true for gradient in the direction of Y-axis. The gradient of ascent and descent being $\frac{\partial f(x,y)}{\partial y}$ and -$\frac{\partial f(x,y)}{\partial y}$. Greater the value of $|\frac{\partial f(x,y)}{\partial y}|$ produces greater ascent or descent.
Figure 2 shows the gradient function $f(x, y)$ in R^2 and the direction of movement of point (x, y) such that the movement of point (x, y) produces $f(x, y)_{min}$. The length of the arrow diagrammatically represents the magnitude of the gradient and the direction represents the movement towards minimum function.

The characteristics equation that are validated in iterations for moving the point are
$x_{new} = x_{old} + \Delta x$ if $\frac{\partial f(x_{old}, y_{new})}{\partial x}$ is negative.
$x_{new} = x_{old} - \Delta x$ if $\frac{\partial f(x_{old}, y_{new})}{\partial x}$ is positive.
$y_{new} = y_{old} + \Delta y$ if $\frac{\partial f(x_{new}, y_{old})}{\partial y}$ is negative.

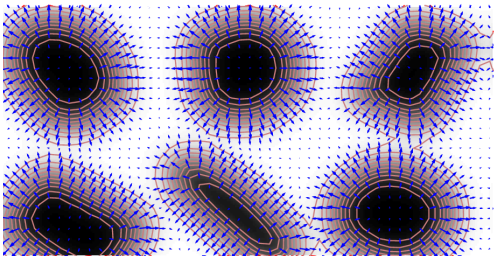


Fig. 2: Gradient Descent

$y_{new} = y_{old} - \Delta y$ if $\frac{\partial f(x_{new}, y_{old})}{\partial y}$ is positive.
Till both the partial derivatives $\frac{\partial f(x_{new}, y_{new})}{\partial x}$,$\frac{\partial f(x_{new}, y_{new})}{\partial y}$ attain minimum which results in local minima.

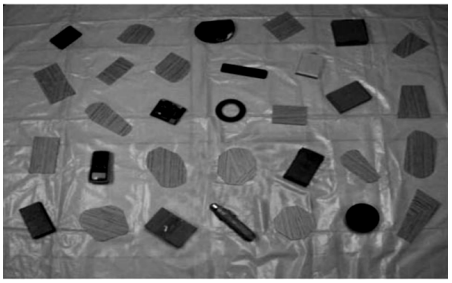
III. Methodology

This section deals with the methodology adopted in this paper to come up with a path using modified gradient descent algorithm. The paper deals with path planning in two stages

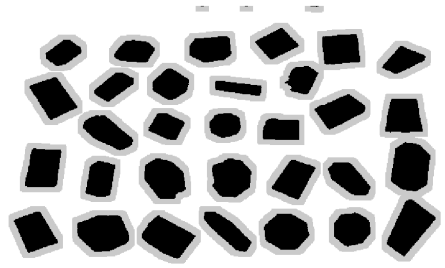
- Pre-processing.
- Gradient function generation.
- Gradient descent method in RRT.

(A) Pre-processing:

In this stage, the image of the work-space of interest is taken first and then the converted to monochromatic and converted to binary image [13] .



(a) Physical Image [13]



(b) Processed Image

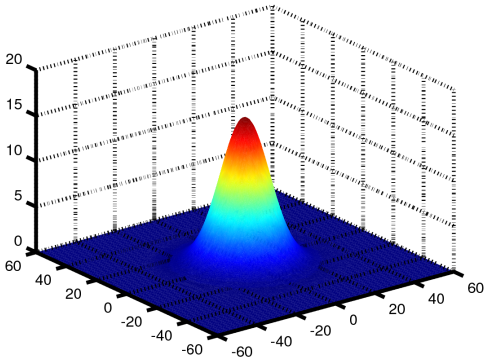
Fig. 3: Pre-Processing

The processed image is segmented as a configuration space C with obstacle configuration space C_{obst} and free configuration space C_{free}. The obstacle space C_{obst} is convoluted with a Minkowski number. Figure 3 shows the Pre-processing steps. The work-space snapshot is given in Figure 3(a). After pre-processing the image is converted to a monochromatic binary image. The back portion of the Figure 3(b) shows the obstacle region C_{obst}. After this region being convoluted with a Minkowski number as per the below equation, the obstacle space dilates. The grey portion in Figure 3(b) shows the dilated obstacle configuration space C_{obst}. the dilated obstacle space will be used for further computational processing.
$A + B = \{a + b | a \in A, b \in B\}$
Where A and B are vector spaces and a and b are vectors in the vector spaces A and B respectively.

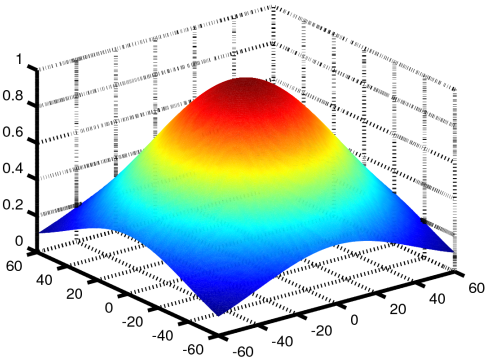
(B) Gradient function generator:

The binary image is subjected to Gaussian blur. The Gaussian function is given by

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x-\mu)^2 - (y-\mu)^2 / 2\sigma^2}$$



(a) Gaussian Distribution with σ=10



(b) Gaussian Distribution with σ=40

Fig. 4: Gaussian Distribution

where (x, y) are the coordinate points of the image, μ is the mean and σ^2 is the variance of the Gaussian distribution. By changing the variance of the Gaussian

distribution, the gradient can be changed. Figure 4 gives two different Gaussian distribution with mean around $(0, 0)$ and variance being standard deviation σ being 10 and 40. The above Gaussian function is applied to the binary image that represents the configuration. space. Figure 5 shows the gradient function of the configuration space. The σ is selected such that it is comparable with average size of the obstacle. The dark blue region in the three-dimensional image of Figure 5 is a free configuration space C_{free} which forms minima for the points (x, y).

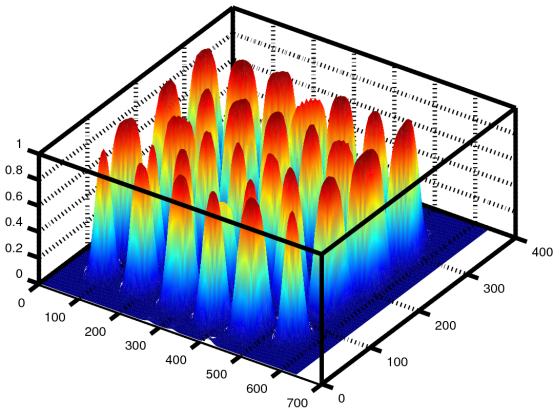


Fig. 5: Gradiant function based on C

(C) Gradient descent method in RRT:

This section of the paper brings out the algorithm used for gradient descent using RRT. The algorithm uses the gradient function as shown in Figure 5. If the point (x, y) lies in obstacle configuration space C_{obst}, then as shown in figure 5, it is on gradient hill with positive gradient or negative gradient based on if the location of the point is on the left side or right side of the gradient. The algorithm repeats the iterations till the point (x, y) is brought outside of the obstacle configuration space C_{obst} to free configuration space C_{free}. One needs to note that the objective of the algorithm is not to move the points to local minima. The objective is just to move the point outside the obstacle configuration space.

The following are the steps that are executed in the algorithm.

- Load the binary image in a matrix.
- Identify the obstacle configuration space C_{obst} and free configuration space C_{free}.
- Apply Gaussian distribution function and round off to dilate the obstacle configurations space. The variance represents the Minkowski number.
- Generate gradient function from the new configuration space.
- Apply basic RRT in the new configuration space. Use the gradient function to move the points q_{new}, when the point lies in the obstacle configuration space C_{obst}.
- Repeat the basic RRT loop till path is established between q_{init} and q_{final}.

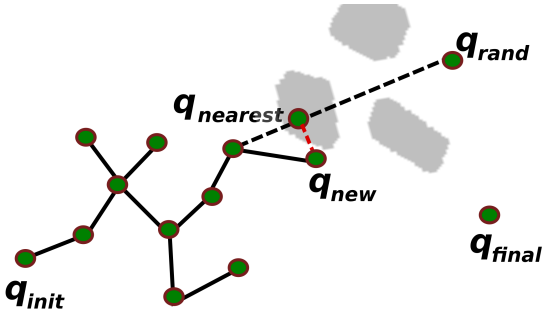
Algorithm Body of gradient descent in *RRT* algorithm

$V \leftarrow \{q_{init}\}; E \leftarrow \varnothing; i \leftarrow 0;$
while $i < N$ **do**
  $G \leftarrow (V, E);$
  $q_{rand} \leftarrow Sample(i); i \leftarrow (i + 1);$
  $(V, E) \leftarrow Extend(G, q_{rand});$
end while

Algorithm Body of $Extend(G, q)$

$V' \leftarrow V; E' \leftarrow E;$
$q_{nearest} \leftarrow Nearest(G, q);$
$q_{new} \leftarrow Steer(q_{nearest}, q);$
if $ObstacleFree(q_{nearest}, q_{new})$ **then**
  $V' \leftarrow V' \cup \{q_{new}\};$
  $E' \leftarrow E' \cup \{q_{nearest}, q_{new}\};$
else
  $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x-\mu)^2-(y-\mu)^2/2\sigma^2} * C$
  $(x_{old}, y_{old}) = q_{new}$
  while $(x_{new}, y_{new}) \in C_{obst}$ **do**
    $Update(x_{old}, y_{old})$
    $Update(\frac{\partial G(x,y)}{\partial x})$
    $x_{new} = x_{old} \pm 1$ if $\left|\frac{\partial G(x,y)}{\partial x}\right| > 0$ at (x_{old}, y_{new})
    $Update(\frac{\partial G(x,y)}{\partial x})$
    $y_{new} = y_{old} \pm 1$ if $\left|\frac{\partial G(x,y)}{\partial y}\right| > 0$ at (x_{new}, y_{old})
  end while
  $V' = (x_{new}, y_{new})$
  $V' \leftarrow V' \cup \{q_{new}\};$
  $E' \leftarrow E' \cup \{q_{nearest}, q_{new}\};$
end if
return $G' = (E', V')$

(a) Gradient descent in RRT Psuedo Code



(b) Gradient descent in RRT Extend Function

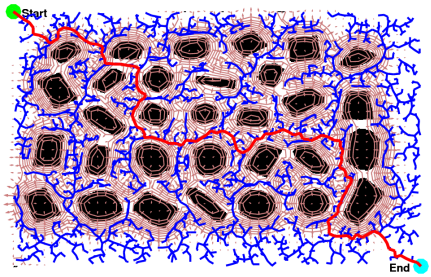
Fig. 6: Gradient descent in RRT Algorithm

The gradient descent using RRT algorithm is shown in Figure 6(a). Highlighted in blue in figure are the changes based on gradient descent method applied on RRT algorithm. The connection strategy of the proposed approach is shown in Figure 6(b). For a given q_{rand}, if the q_{new} lies in the obstacle configuration space C_{obst} as show in grey portion of the image, then the gradient descent method is applied to point on q_{new} so that q_{new} moves towards the local minima function of $G(x, y)$ in

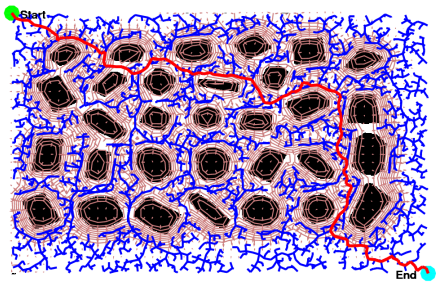
the free configuration space C_{free}. The path traversed by q_{new} is shown in red dotted line in the image. Once q_{new} enters C_{free} the gradient descent method is stopped, and the connection from q_{new} to $q_{nearest}$ is initiated.

IV. Results

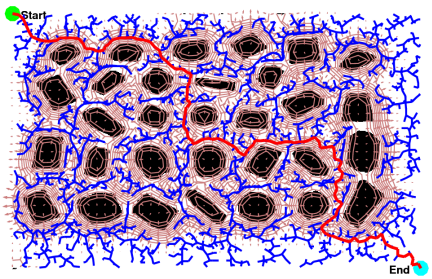
Figure 7 gives the path generated by gradient descent in three different iterations. In each of the iterations, we use three different uniform sample set. The sample set is preserved so that the sets are used in iterations for basic RRT run for comparison purpose.



(a) Iteration 1



(b) Iteration 2

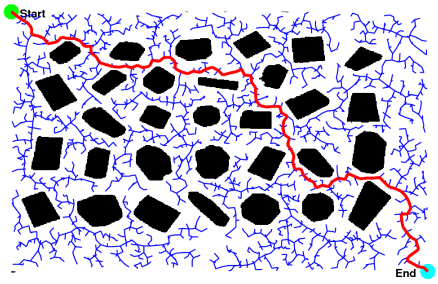


(c) Iteration 3

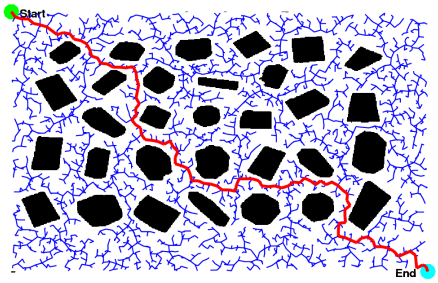
Fig. 7: Path Planning using gradient descent

The gradient descent moves the point q_{new} from the obstacle region C_{obst} to C_{free}. The sample set that was preserved in the gradient descent method is now used to run the basic RRT algorithm. Figure 8 shows the path planning results in basic RRT algorithm for three different iterations.

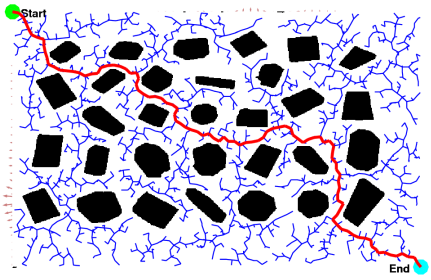
Figure 9 shows the point movement due to gradient descent method. The area marked in black is the obstacle



(a) Iteration 1



(b) Iteration 2



(c) Iteration 3

Fig. 8: Path Planning using basic RRT

region C_{obst}. The area marked in light grey is the dilated or convoluted obstacle region considering Minkowski sum. The dilation is done with an objective to have a minimum clearance from the obstacle to the moving agent.

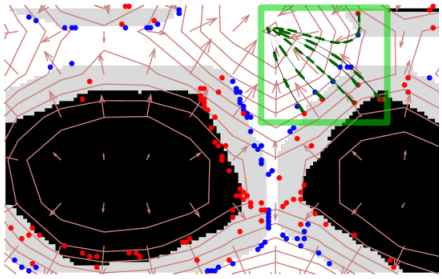


Fig. 9: Point movement using gradient descent

Points marked in red q_{new} are the derived points generated by RRT algorithm. The point in blue are

the points that were derived by the movement of q_{new} from obstacle region C_{obst} to free region C_{free}. The region bounded in square shows the movement of the points from obstacle region C_{obst} to free region C_{free}.

The following table gives the comparison of time taken of gradient descent method and basic RRT. Different iteration uses different sample sets drawn using uniform distribution. In a given iteration, the same set of the samples are used for both of the basic RRT as well and gradient descent based RRT for comparison. It is observed from Table 1, that gradient-based approach gives faster convergence to the solution. This is because of the fact that in basic RRT, every points q_{new} undergo collision check. If q_{new} falls in obstacle space C_{obst}, then the point is discarded to generate new q_{new}. This results in redundancy of collision check for extra points that are not part of the tree in basic RRT. However, the gradient descent method checks if the point q_{new} resides in C_{obst} then it moves that to C_{new} avoiding redundancy. It ignores only those points q_{new} that resides in C_{obst} without sufficient gradient.

TABLE I:

Comparision of gradient descent RRT with basic RRT

Time cost in seconds			
Index	RRT with gradient descent	RRT with collision check	Ratio (%)
1	645	1008	63.99
2	1135.14	1541	73.66
3	652	786.3	82.9

V. Conclusion

Gradient descent method is applied on points generated by RRT to move it from C_{obst} to C_{free}. It is observed that redundancy in collision check of basic RRT is avoided due to the points being used and resulted in better convergence. Multiple iterations in the simulation showed time-saving using gradient descent method. During the run, few of the limitations are observed in this algorithm. First, the step size should not be greater than the minimum obstacle dimension. If the step size is greater, then it would result in a collision. Second, the gradient function used in the algorithm should have a good slope to move around the points. If the slope is minimal, the gradient descent approaches slow convergence or the points will cease to move. This scenario was observed during the simulation. Since simplest gradient descent approach is used, one can explore further, with other gradient descent algorithm that provides faster convergence. Also the concept of this algorithm can be applied to algorithms derived from RRT, PRM like RRT* and PRM*.

References

[1] S. M. Lavelle, "Plannning Algorithm", Cambridge University Press, 2006.

[2] S. M. LaValle, "Rapidly-exploring Random Trees: A new tool for path planning", Technical Report, Computer Science Dept, Iowa State University, pp. 1–4 , 1998.

[3] Sertac Karaman and Emilio Frazzoli . "Sampling-based Algorithms for Optimal Motion Planning". IEEE International Conference on Robotics and Automation (ICRA), vol. 30(7), pp. 846–894, 2011.

[4] M. S. Branicky, M. M. Curtiss, J. A. Levine, S. B. Morgan. "RRTS for nonlinear, discrete, and hybrid planning and control". IEEE International Conference on Decision and Control, vol. 1, pp. 657–663, 2003.

[5] Anna Yershova, Steven M. LaValle. "Improving motion-planning algorithms by efficient nearest-neighbor searching", IEEE Transactions on Robotics. vol. 23(1), pp. 151–157, 2007.

[6] V.Boor, M.H.Overmars and A.F.Van Der Stappen,"The gaussian sampling strategy for probabilistic roadmap planners, IEEE International Conference on Robotics and Automation , pp.1018–1023, 1999.

[7] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati and J. Reif, "Narrow passage sampling for probabilistic roadmap planners", IEEE Trans. on Robotics, vol. 21(6), pp. 1105–1115, 2005.

[8] H. L. Cheng, D. Hsu, J. C. Latombe and G. Sanchez-Ante, "Multi- level free-space dilation for sampling narrow passages in PRM planning", IEEE International Conference on Robotics Automation, pp.1255–1260, 2006.

[9] Liangjun Zhang, Dinesh Manocha, "An efficient retraction-based RRT planner", IEEE International Conference on Robotics and Automation (ICRA)", pp. 3743–3750, 2008.

[10] S. A. Wilmarth, N. M. Amato and P. F. Stiller, "Motion plan ning for a rigid body using random networks on the medial axis of the free space", Symposium on Computational Geometry , pp. 173–180, 1999.

[11] 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(3), pp. 301–319, 2005.

[12] Y. L. Tong, "The multivariate normal distribition", Springer, 1990.

[13] Jianqiang Li, Genqiang Deng, Chengwen Luo, Qiuzhen Lin, Qiao Yan, Zhong Ming, "A hybrid path planning method in unmanned air/ground vehicle (uav/ugv) cooperative systems", IEEE Transactions on Vehicular Technology, vol.65(12), pp. 9585–9596, 2016.