AI PATH PREDICTION AND PLANNING FOR AUTONOMOUS VEHICLES

Author: Parshva Anish Maniar (ORCID: 0009-0005-9963-4189)

1. ABSTRACT

Autonomous navigation of vehicles at various places can tremendously improve daily traversal, especially while driving in unfamiliar territory. Path-planning systems have been continuously improved to cope with increasingly difficult issues and advances in computing technology. One example is the rise of sampling-based planners, which made difficult planning issues easier to solve. As a result, problems that were previously thought difficult can now be addressed, and evendynamical constraints can be overcome.

The ongoing modifications and study in path planning are aimed at achieving robotic systems' fullautonomy. By relying solely on path planning approaches, this goal has not been fully realized. The majority of path-planning algorithms work well in simulations and laboratory robots, but real- world robots operate in unknown surroundings and interact with other agents whose motion is often unpredictable and whose conduct can be uncooperative.

Real robots have inherent kinematical and dynamical (kinodynamical) constraints that make autonomous collision-free motion impossible in particular circumstances. This research demonstrates the implementation, testing, and comparison of two path-planning algorithms for autonomous navigation of a vehicle/rover, namely rapidly-exploring random trees (RRT) and second probabilistic roadmap (PRM).

Using random samples from the search space, an RRT generates a tree rooted at the beginning configuration. As each sample is drawn, a link is made between it and the state closest to it in the tree. This results in the adding of the new state to the tree if the link is viable (goes wholly across free space and obeys all constraints). The main idea underlying PRM, on the other hand, is to pick random samples from the robot's configuration space, test them to see whether they are in the free space, and then use a local planner to connect these configurations to other adjacent configurations. The beginning and goal configurations are brought in, and the resulting graph is subjected to a graph search algorithm in order to find a path between them. To assess the performance of the recommended approaches, simulations and experiments are carried out.

2. INTRODUCTION

We need path planning for autonomous vehicles where human intervention is not required, it can be self-driven cars or drone or rovers which require path planning to navigate the surface of different planets.
To overcome this problem a number of algorithms has been designed to automate the path planning of the vehicles. Path-planning systems have been continuously improved to cope with increasingly difficult issues and advances in computing technology.
The autonomous navigation of rovers / vehicles on unknown surfaces, such as the surfaces of other planets and other unknown terrains can immensely improve day to day transport. Itcan also help in the movement of rovers to and from the lander where human intervention can be difficult. Path-planning systems are continuously being improved to overcome difficult issues and advances in computing technology.

In this project, we will apply two algorithms first rapidly-exploring random trees(RRT) and second probabilistic roadmap(PRM) on a single real-world problem about a unmanned vehicle with basic functionalities which has to navigate through various **terrains with obstacles** to reach its goal by taking **best possible route** in its **best time** and their **efficiency** in reference to **path length** and **execution time** are compared.

3. HARDWARE/SOFTWARE REQUIREMENTS

HARDWARE

Since the application must run over the internet, all the hardware required to connect to the internet will be a hardware interface for the system. As for e.g. Modem, WAN – LAN,Ethernet Cross-Cable. Other minimal hardware requirements that must be fulfilled is- **Processor: Intel I3 or higher,**
Ram: 2GB or more,HDD: 1GB or more

SOFTWARE

Operating System: Windows - macOS - Linux / Android Device with Camera Internet Browser: Google Chrome, Internet Explorer, Mozilla, or Netscape Navigator.

PREFERRED LANGUAGE:

MATLAB

ALGORITHMIC TECHNIQUES CHOSEN:

Rapidly-Exploring random trees(RRT)
Probabilistic Roadmap(PRM)

4. EXISTING SYSTEM

4.1 Literature Review

A comparison of a few algorithms and techniques which are already existing has been done below. We have analyzed and referred through ten papers and summarized the advantages and disadvantages of the proposed algorithms or methods.

Sl. No.	Title	Author	Year Published	Advantages	Disadvantages
1	Simulation Performance Comparison of A*, GLS, RRT and PRM Path Planning Algorithms	• Aisha Muhammad • Nor Rul Hasma Abdullah • Mohammed A.H Ali • Ibrahim Haruna Shanono • Rosdiyana Samad	2022	1) A* is complete and optimal. 2) GLS better than others. 3) RRT is a simple algorithm to implement. 4) PRM builds a single roadmap for travelling through theregion.	1) A* applicable only if branching factor is finite and action hasfixed cost. 2) Comparatively unknown presently. 3) In RRT path foundis not necessarily optimal.
2	The PRM-A* Path Planning Algorithm For UUVs: An Application To Navy Mission Planning	• Mohammad Imran Chowdhury • Daniel G. Schwartz	2020	1) PRM-A* has faster convergenceand a smooth trajectory. 2) No local minima.	1) It cannot accommodate moving obstacles like boats, ships, etc.

				3) Replanning capability.	
3	Review of Path Planning Algorithmsfor Unmanned Vehicles	• Yunyi Shen • Jian Liu • Yasong Luo	2021	1) Dijkstra has high success rate and good robustness. 2) A* is completeand optimal. 3) A* search speed is very fast. 4) RRT can solve problems of high dimensionality.	1) In dijskitra numberof nodes is large, search speed is slow. 2) A* cannot meetreal time requirements if graph is large 3) Convergence speed decreases if there is a large number of obstaclesfor RRTF. 4) PRM is not complete if only fewsampling points.
4	Path Planning Algorithm on Large Scale Terrain Data Based on PRM with	• Sijie Li • Anran Yang • Luo Chen	2021	1) Low cost paths.	1) Cannot maintainefficient planning.

	Non-Uniform Sampling Strategy				2) RRT could be more efficient thanDEM.
5	Utilizing a Rapidly Exploring Random Tree for Hazardous Gas Exploration in a Large Unknown Area	• Yaqub A. Prabowo • Bambang R. Trilaksono • EGI M. I. Hidayat • Brian Yuliarto	2021	1) Better frontier search using RRT 2) Generates an explorative path	1) Goal decision making still need constants selection 2) Slower comparedto other non-grid- based path planning.
6	Application of improved Dijkstra algorithm in intelligent ship path planning	• Zhenyu Zhu • Lianbo Li • Wenhao Wu • Yang Jiao	2021	1) Optimal path point is obtained 2) Path finding speedis increased	1) Cannot handle negative edges. 2) Forms acyclicgraphs.
7	Information-DrivenFast Marching Autonomous Exploration With Aerial Robots	• Ping Zhong • Bolei Chen • Siyi Lu • Xiaoxi Meng • Yixiong Liang	2022	1) Chooses exploration goalsmore wisely.	1) Has a lowerexploration efficiency. 2) Lesser safety.
8	Path Planning Algorithm Using the Hybridization of the	• Muhammad Aria Rajasa Pohan	2021	1) Fast and optimalpath planning.	1) ACS has a stagnation phase.

	Rapidly-Exploring Random Tree and Ant Colony Systems	• Bambang Riyanto Trilaksono • Sigit Puji Santosa • Arief Syaichu Rohman			2) Higher explorationand exploitation rate.
9	Bidirectional Potential GuidedRRT* for MotionPlanning		2019	1) P − RRT ∗ −connect can find a non-collision optimal path in fewer iterations. 2) Running time is reduced	1) Not easy toimplement
10	An Improved Path Planning Algorithmfor UAV Based on RRT	• Jianqing Chen • Jiyan Yu	2021	1) Compared to RRTit has a lesser randomness in sampling. 2) Speeds up convergence.	1) Low dynamicperformance. 2) Not asymptoticallyoptimal.

4.2 Drawbacks in the existing system

The use of the probabilistic roadmap path planning paradigm for an unmanned aerial vehicle application was shown in the study on **Probabilistic Roadmap Based Path Planning for an Autonomous Unmanned Aerial Vehicle (UAV)**. However, there were several limitations which were discovered for the traditional PRM technique during the trial of the PRM path planning prototype. The most difficult feature in this situation is to maintain an efficient planning approach while adhering to the non-holonomic limitations required to ensure seamless transitions between distinct trajectory segments. Another key issue that came up was the efficiency tradeoff between offline and online work. Flexibility in response to contingent changes in the UAV environment during plan execution appears to be sacrificed in favor of efficient runtime planning based on a PRM developed offline.

The RRT algorithm is well-suited for the scenario of a free-floating satellite manipulator system, according to the results of the research paper **Application of Rapidly-exploring Random Trees (RRT) algorithm for trajectory planning of free-floating space manipulator by Tomasz Rybus and Karol Seweryn**. However, a thorough comparison of the performance of the method given herewith that of alternative trajectory planning methods is still needed. Furthermore, they used the basic version of the RRT algorithm in this study, although a newer, upgraded version likeRRT* could yield superior results.

The road test results show that the trajectory planning approach described in the article **An optimal trajectory planning algorithm for autonomous trucks** may provide effective and smooth vehicle lane changes, as well as certain basic autonomous features, in modest industrial parks and ports. The real-time performance of this algorithm, however, cannot match the requirements of the complex road because the trajectory in complex scenarios is too intricate. Delving into RRT's real-time strategy and figuring out how to incorporate it into the Dijkstra algorithm or cost function model would be a step toward cleaner execution.

The ongoing studies on path planning are solely aimed at achieving the full autonomy of the robotic system. We cannot achieve this goal by only relying on the path planning approaches. Most of these existing path planning algorithms may work well in simulations and lab robots, but when these robots are in unknown surroundings and have to interact with other agents whose movement can be unpredictable and uncooperative, these algorithms alone are not enough.

5. PROPOSED MODEL

5.1 Design:

An autonomous vehicle with navigates along different terrains to varying final goals by taking the best possible route in the best possible time. We will do this bydesigning and implementing an algorithm and implementing it in the rover. The algorithm will detect various forms of obstacles in the terrain and will give a collision free path which is optimal.

Previously only one path planning algorithm had been used for a single real-worldrobot. In our proposed model we intend to modify this by applying 2 path- planning algorithms for the autonomous navigation of a single vehicle.

The 2 path planning algorithms which can be used are RRT (rapidly-exploring random trees) and PRM (probabilistic roadmap). We can also use the A* algorithm in place of the PRM algorithm based on the results. Then, we find the path using Dijkstra's shortest path algorithm.

5.2 Module Wise Description of Algorithms used:

I. RAPIDLY EXPLORING RANDOM TREES(RRT)

A rapidly-exploring random tree (RRT) is an algorithm designed to searchnonconvex, high- dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem. RRTs can be viewed as a technique to generate open-loop trajectories for nonlinear systems with state constraints. In the base paper, it is introduced that a randomized data structure for path planning is designed for problems that have nonholonomic constraints. Both RRT & PRM are designed with as few heuristics and arbitrary parameters as possible. The unique advantage of RRT is that they can be directly applied to nonholonomic and Kino dynamic planning. Properties of RRT are:

1) The expansion of an RRT is heavily biased towards unexplored portions of the state space.
2) The distribution of vertices in an RRT approaches the samplingdistribution, leading toconsistent behavior.
3) RRT is probabilistically complete under very general conditions.
4) The RRT algorithm is relatively simple, which facilitates performance analysis.

5) The entire path planning algorithm can be constructed without requiring the ability to steer the system between two prescribed states, which greatly broadens the applicability of RRT.

II. PROBABILISTIC ROADMAP (PRM)

A probabilistic roadmap (PRM) is a network graph of possible paths in a given map based on free and occupied spaces. The robotics PRM class randomly generates nodes and creates connections between these nodes based on the PRM algorithm parameters In the probabilistic road map procedure, the basic idea is to construct a road map of the free space consisting of random samples and edges between them. After one side has been constructed one could connect your desiredstart and endpoints to this graph and plan a path from one end to the other. It should be noted that the first phase constructs the generic road map of the entire free space without considering any particular pair of start and endpoints. The advantage of this approach is that one can reverse the roadmap over and overagain to answer multiple planning problems. When we want to answer only one specific planning problem, it would be wasteful to construct roadmaps that span theentire free space. The probabilistic roadmap planner consists of two phases: construction and a query phase. In the construction phase, a roadmap (graph) is built, approximating the motions that can be made in the environment. First, a random configuration is created. Then, it is connected to some neighbors, typically k nearest neighbors. Configurations and connections are added to the graph until the roadmap is dense enough. In the query phase, the start and goal configurationsare connected to the graph, and the path is obtained by Dijkstra's shortest path query.

5.3 Implementation of the Algortihms:

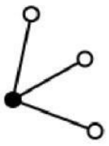
Rapidly-Exploring Random Trees(RRT):

Input: Initial configuration q_{init}, number of vertices in RRT K, incremental distance Δq
Output: RRT graph G

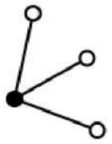
```
$G$.init($q_{init}$)
for $k = 1$ to $K$
$q_{rand} \leftarrow$ RAND_CONF()$q_{near} \leftarrow$ NEAREST_VERTEX(
$q_{rand}$, $G$)$q_{new} \leftarrow$ NEW_CONF($q_{near}$, $q_{rand}$, $\Delta q$) $G$.add_vertex($q_{new}$) G.add_edge($q_{near}$, $q_{new}$)
return $G$
```

Ⓒ Initial Graph G contains -

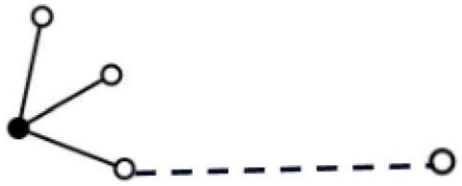
- Start node
- 3 adjoining vertices



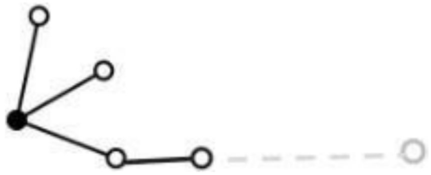
Ⓒ Randomly Generated target node q_{rand} in the environment space



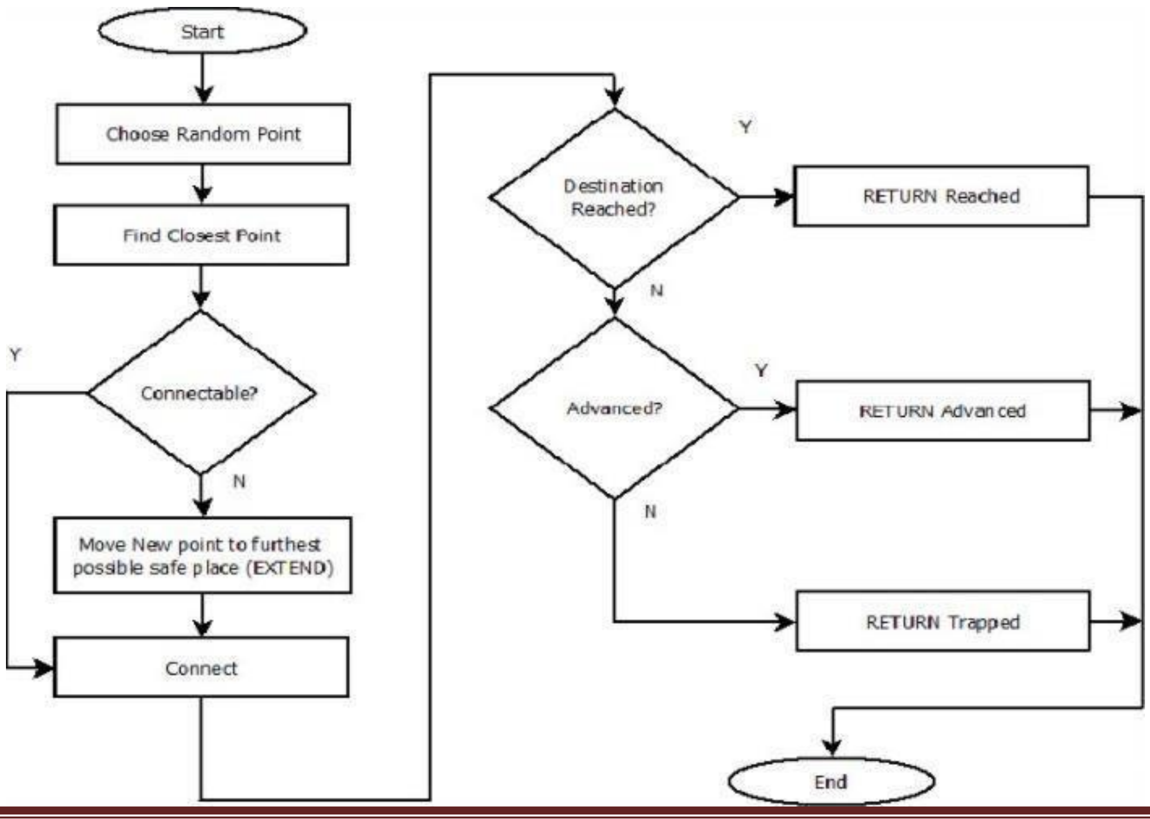
Ⓒ Identifying the nearest node q_{near} from Graph G



④ Generating new node q_{new} at a distance Δq from the nearest node q_{near}



Flowchart:



Probabilistic roadmap(PRM):

Step 1-Roadmap Construction

Input: Initial configuration *n*, number of nodes to put in the roadmap. *k*, number of closestneighbors to examine for each configuration.

```
Output: A roadmap G=(V,E)while |V| < n do
repeat
q ← a random configuration in Quntil q is collision-free
V ← V ∪ {q}end while
for all q ∈ V do
$N_q$ ← the $k$ closest neighbors of $q$ chosen from V according to distfor all $q'$ ∈ $N_q$ do
if (q,q') ∉ E and Δ(q,q') ≠ NIL thenE ← E ∪ {(q,q')}
end if end for
end for
```

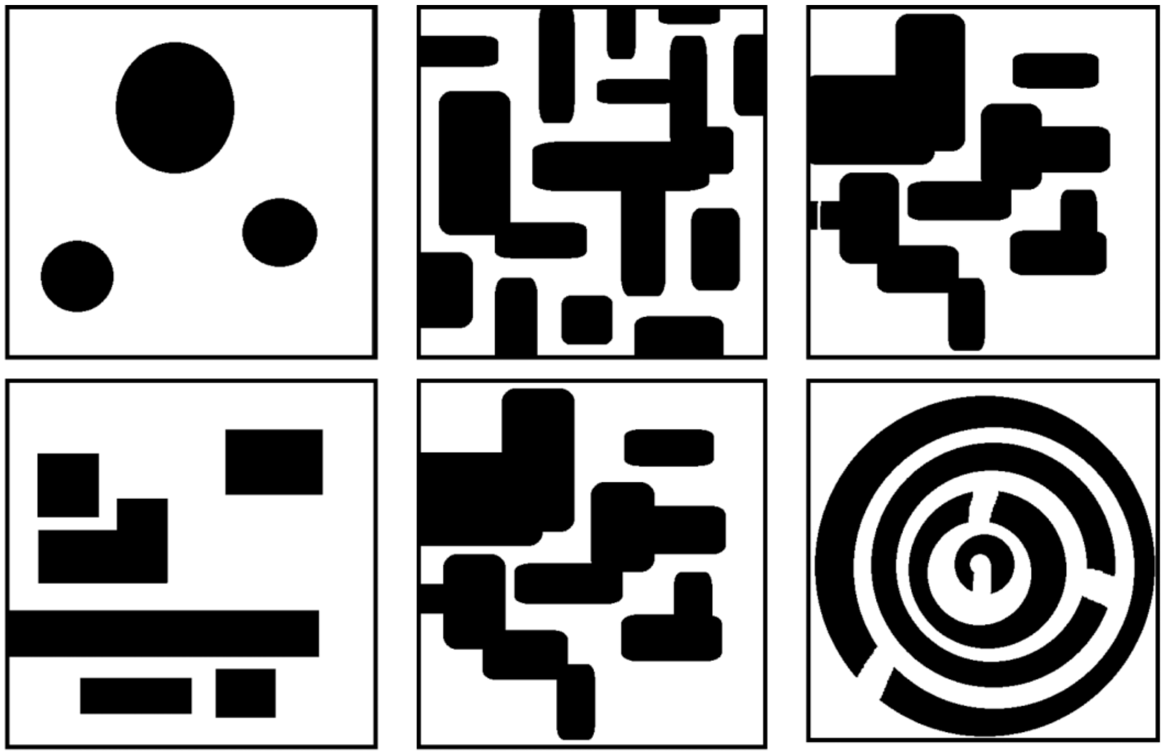
Step 2-Finding a path

Input: Initial configuration q_{init} , the initial configuration. q_{goal}, the goal configuration, k, number of closest neighbors to examine for each configuration. G = (V,E), the roadmap computed aboveOutput: A path from q_{init} *to* q_{goal} or failure

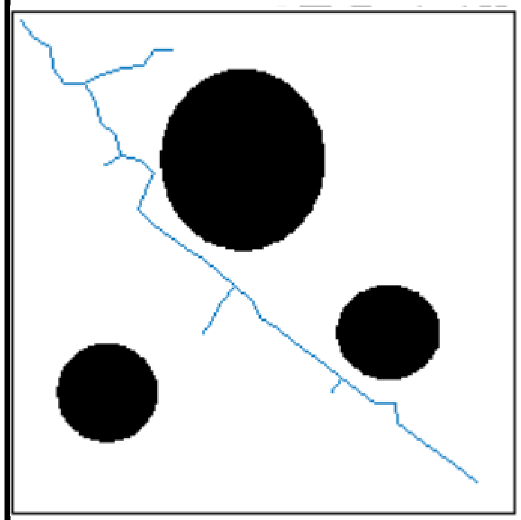
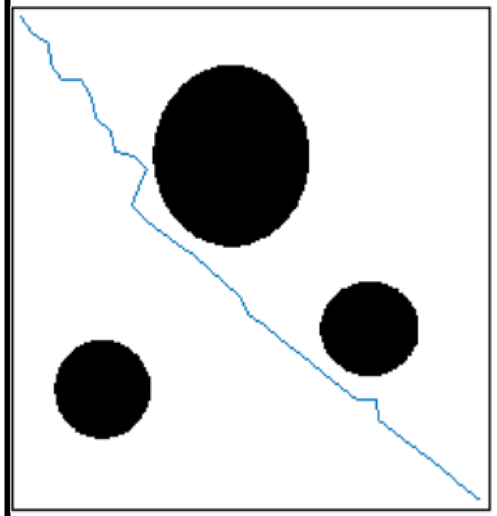
```
N $q_{init}$← the k closest neighbors of $q_{init}$ from V according to dist N $q_{goal}$← the k closest neighbors of $q_{goal}$ from V according to distV←{$q_{init}$ } ∪ {$q_{goal}$} ∪ V
repeat
If Δ($q_{init}$ ,q') ≠ NIL thenE ← ($q_{init}$ ,q') ∪ E
else
set q' to be the next closest neighbor of $q_{init}$in N $q_{init}$end if
until a connection was successful or the set N $q_{init}$is emptyset q' to be the next closest neighbor of $q_{goal}$in N$q_{goal}$repeat
If Δ($q_{goalt}$ ,q') ≠ NIL thenE ← ($q_{goal}$,q') E
else
set q' to be the next closest neighbor of $q_{goal}$ in N $q_{goal}$end if
until a connection was successful or the set N $q_{goal}$ is emptyP ← shortest path($q_{init}$ , $q_{goal}$, G)
If P is not empty thenreturn P

else end if Return failure
```

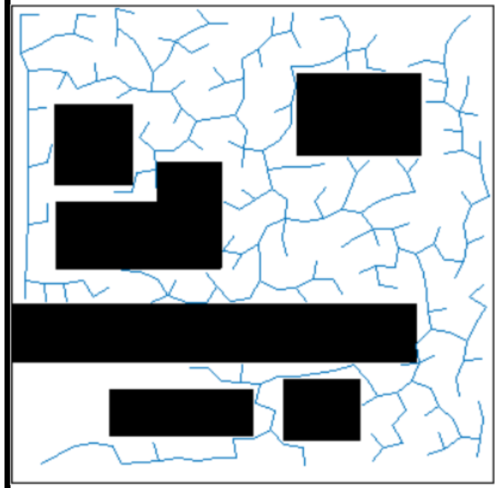
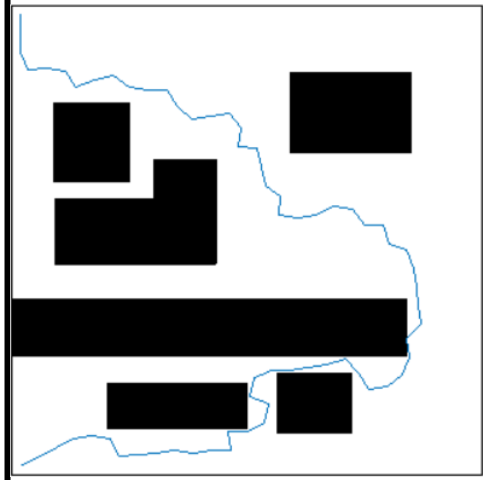
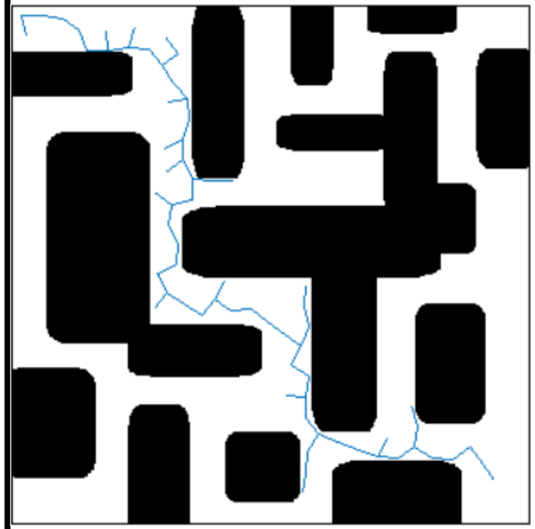
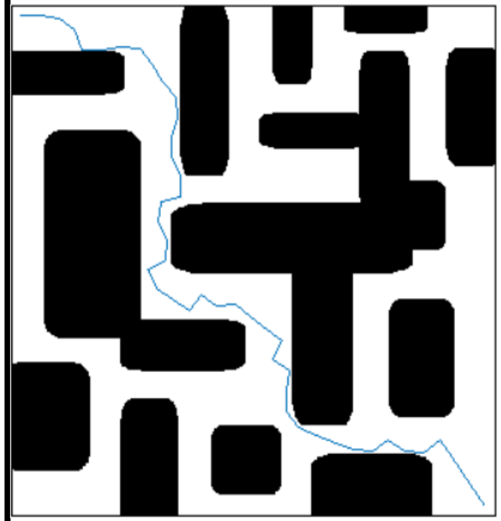
6. RESULTS AND DISCUSSION

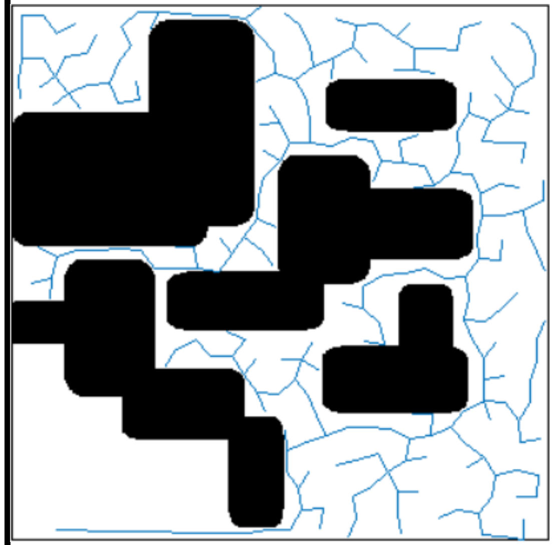
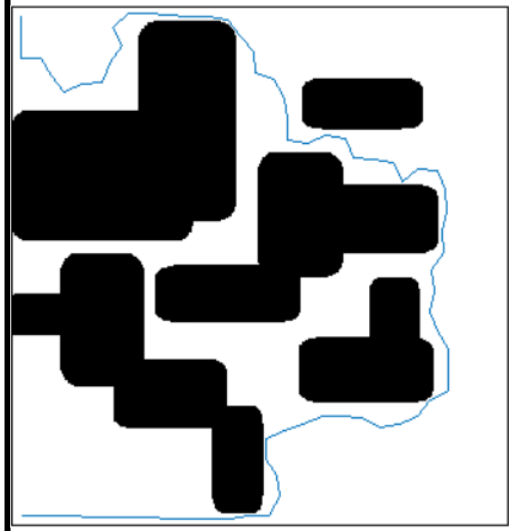
<u>Sample Terrains used:</u>

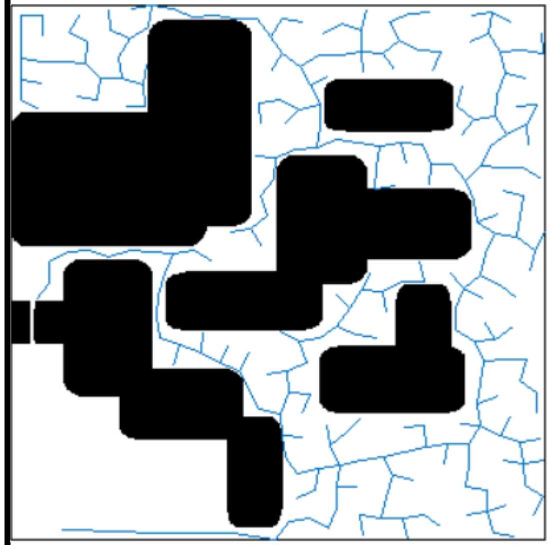
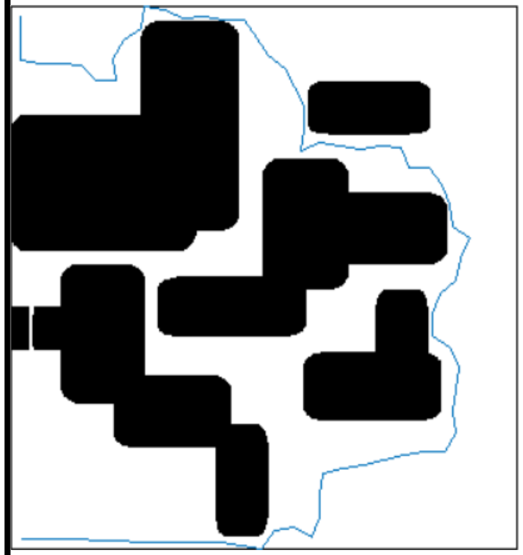


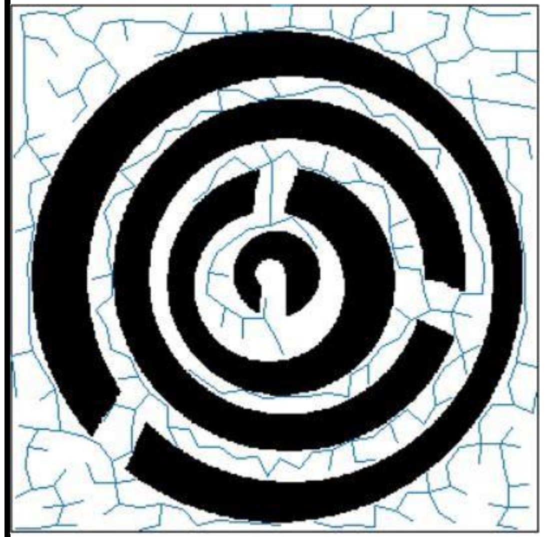
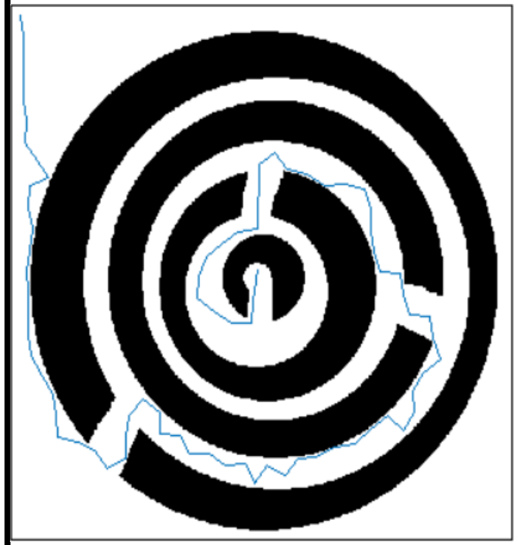
Rapidly-Exploring Random Trees(RRT):

S.no	Step Size	Tree	Path	Path length	Execution time (sec)
1	20			8.101928 e+02	1.188077 e+01

2	20			1.29529 8e+03	3.651574 e+01
3	20			9.251400 e+02	2.333820 e+01

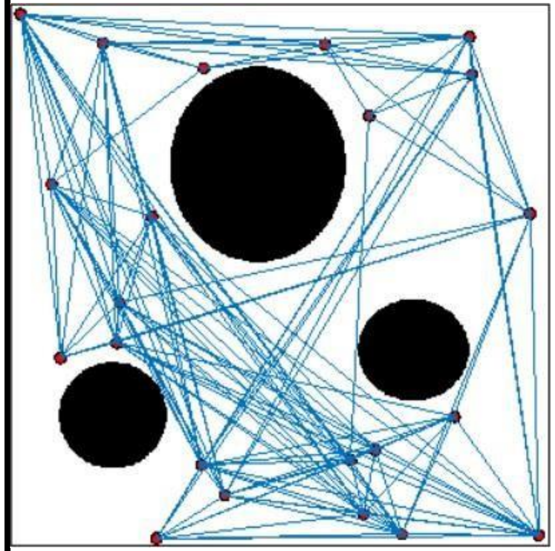
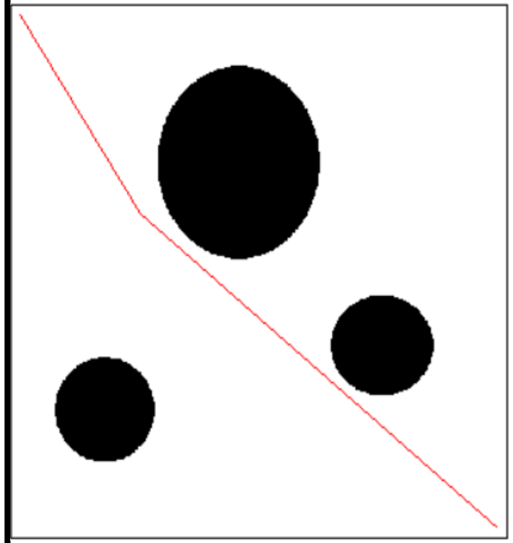
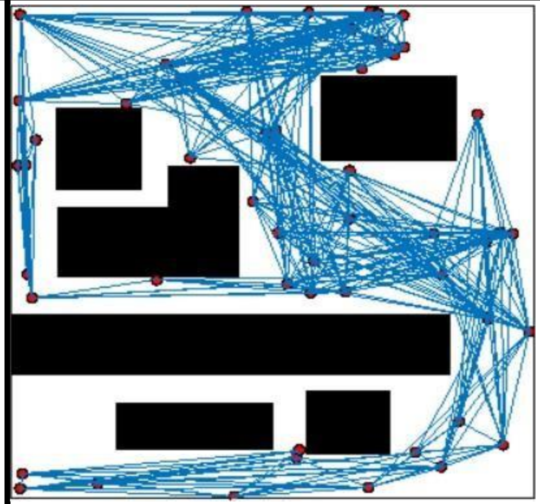
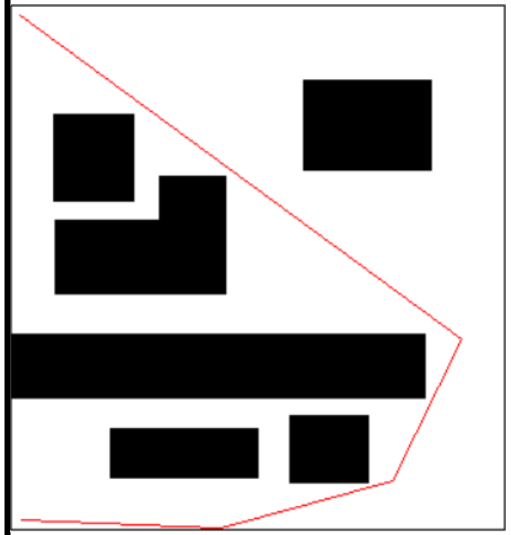
4	20			1.400944 e+03	3.613657 e+01

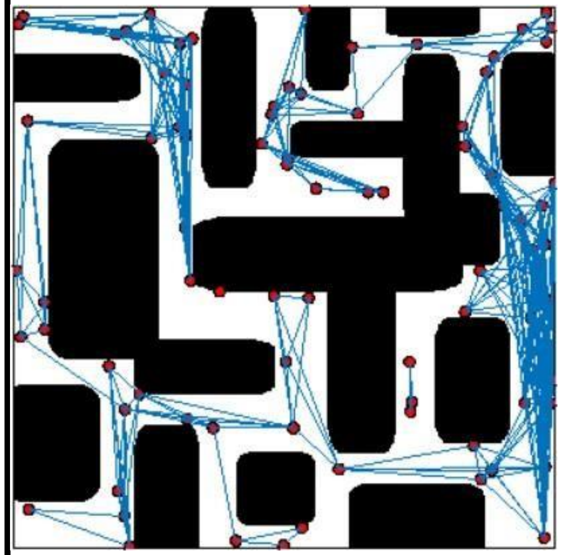
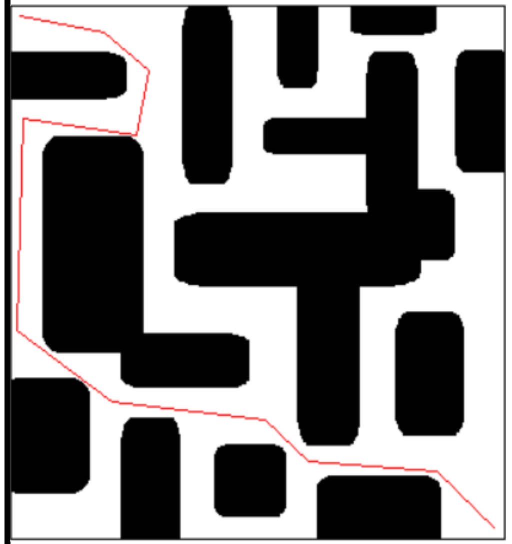
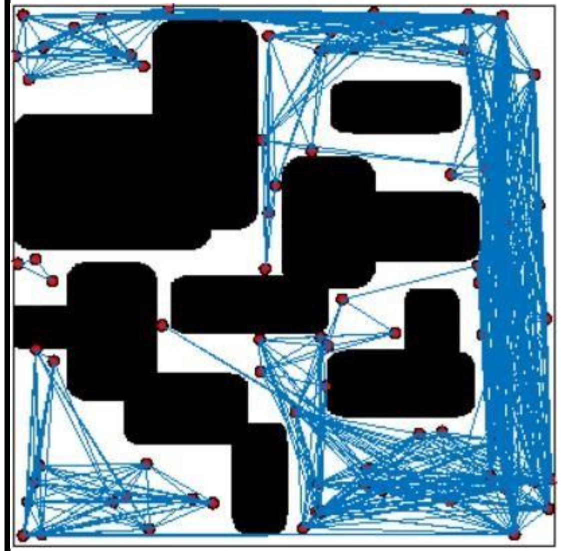
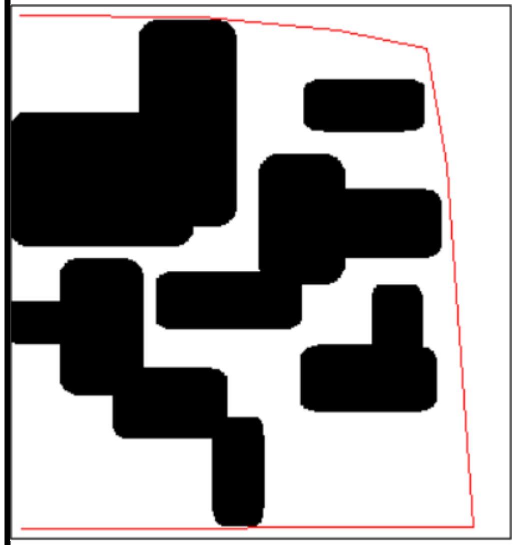
	20			1.388704 e+03	3.613657 e+01

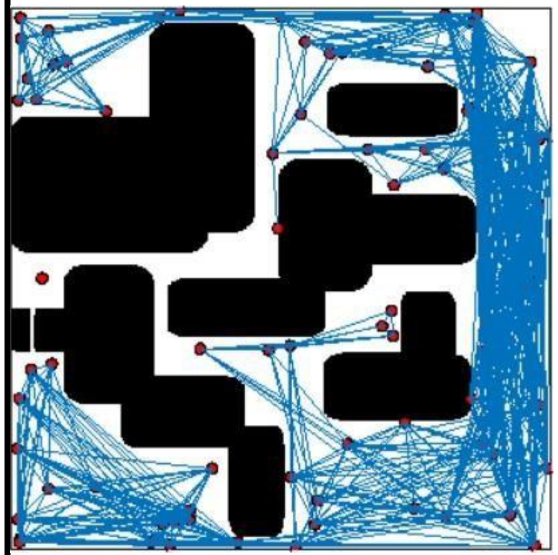
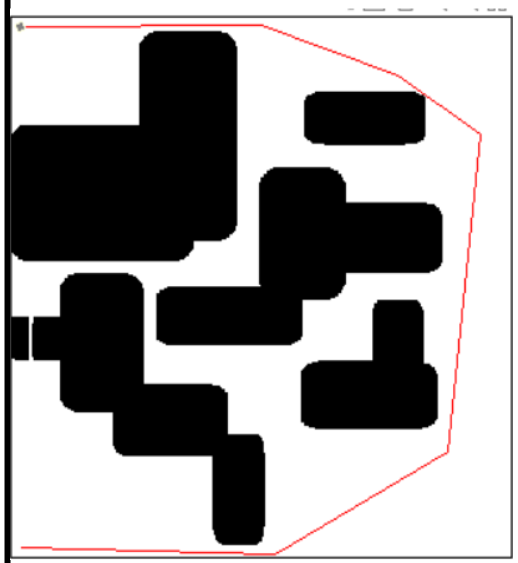
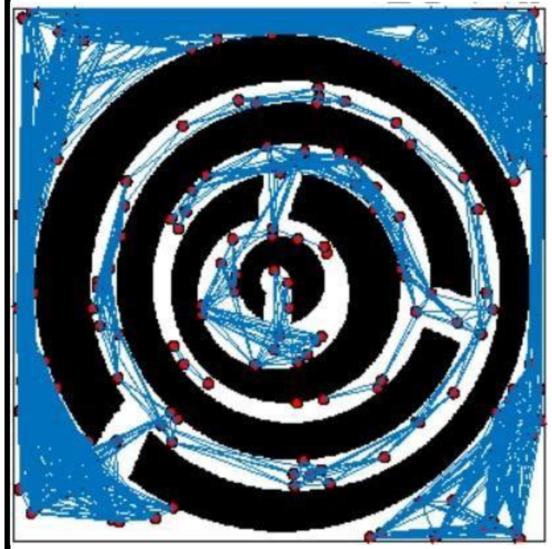
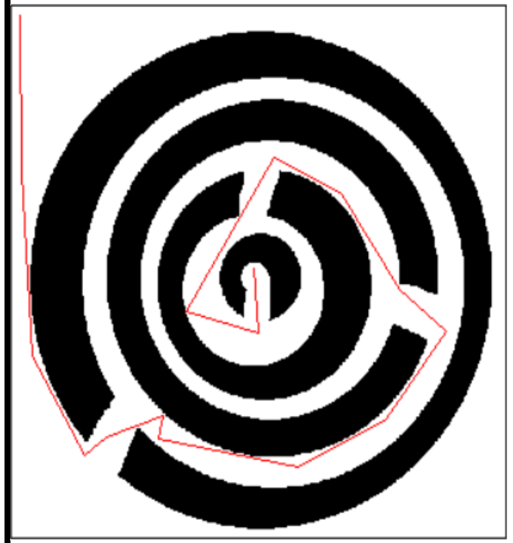
6	20			1.718973 e+03	2.515795 e+01

Probabilistic roadmap(PRM):

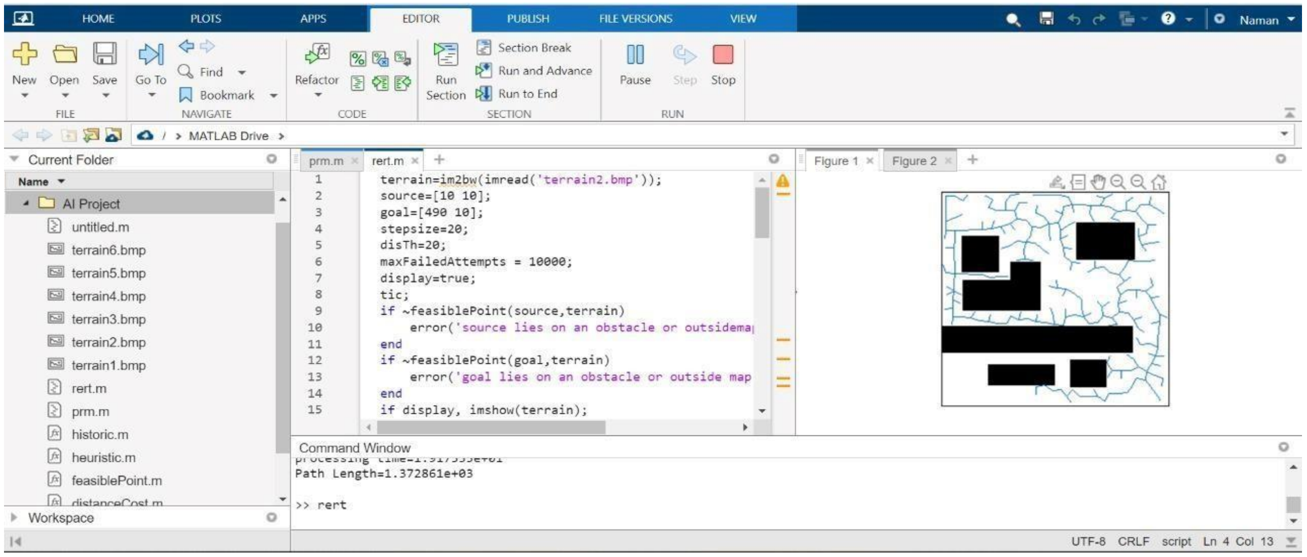
S. no	No. of nodes(k)	Tree	Path	Path length	Execution time (sec)

1	20			6.858932 e+02	7.299953 e+00
2	50			1.07654 0e+03	6.519054 e+00

3	80			9.251400 e+02	2.333820 e+00
4	80			1.311246 e+03	9.929559 e+00

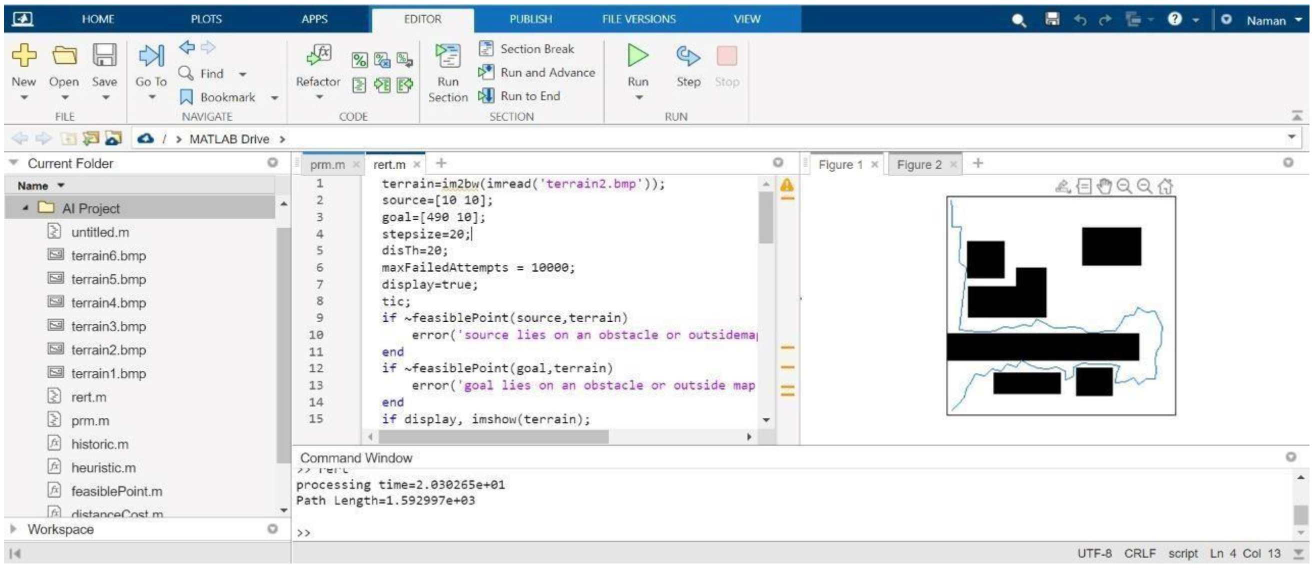
5	80			1.228797 e+03	5.207032 e+00
6	200			1.433969 e+03	9.230730 e+00

Environment Screenshot:



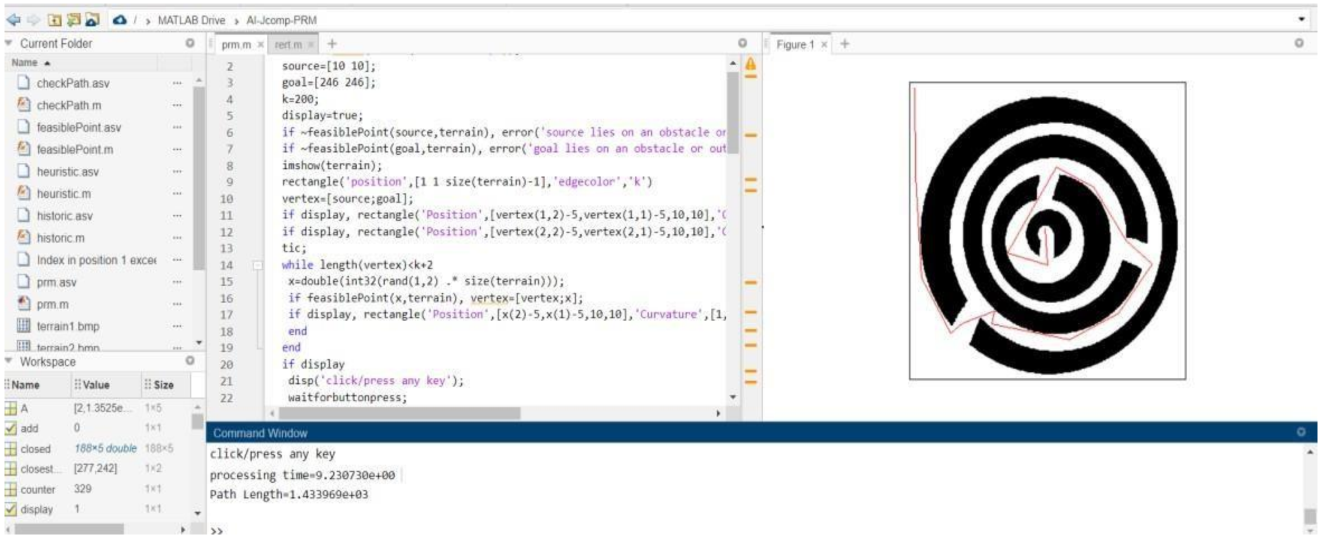
```
terrain=im2bw(imread('terrain2.bmp'));
source=[10 10];
goal=[490 10];
stepsize=20;
disTh=20;
maxFailedAttempts = 10000;
display=true;
tic;
if ~feasiblePoint(source,terrain)
    error('source lies on an obstacle or outsidema
end
if ~feasiblePoint(goal,terrain)
    error('goal lies on an obstacle or outside map
end
if display, imshow(terrain);
```

```
processing time=1.917553e+01
Path Length=1.372861e+03

>> rert
```



```
terrain=im2bw(imread('terrain2.bmp'));
source=[10 10];
goal=[490 10];
stepsize=20;
disTh=20;
maxFailedAttempts = 10000;
display=true;
tic;
if ~feasiblePoint(source,terrain)
    error('source lies on an obstacle or outsidema
end
if ~feasiblePoint(goal,terrain)
    error('goal lies on an obstacle or outside map
end
if display, imshow(terrain);
```

```
>> rert
processing time=2.030265e+01
Path Length=1.592997e+03

>>
```

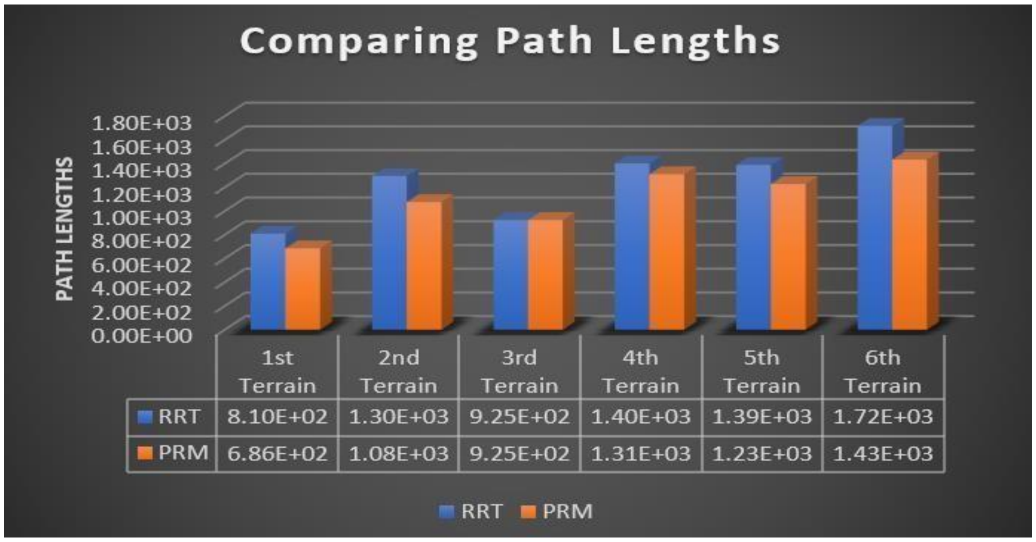


Name	Value	Size
A	[2,1.3525e...	1×5
add	0	1×1
closed	188×5 double	188×5
closest...	[277,242]	1×2
counter	329	1×1
display	1	1×1

```
source=[10 10];
goal=[246 246];
k=200;
display=true;
if ~feasiblePoint(source,terrain), error('source lies on an obstacle or
if ~feasiblePoint(goal,terrain), error('goal lies on an obstacle or out
imshow(terrain);
rectangle('position',[1 1 size(terrain)-1],'edgecolor','k')
vertex=[source;goal];
if display, rectangle('Position',[vertex(1,2)-5,vertex(1,1)-5,10,10],'C
if display, rectangle('Position',[vertex(2,2)-5,vertex(2,1)-5,10,10],'C
tic;
while length(vertex)<k+2
 x=double(int32(rand(1,2) .* size(terrain)));
 if feasiblePoint(x,terrain), vertex=[vertex;x];
 if display, rectangle('Position',[x(2)-5,x(1)-5,10,10],'Curvature',[1,
 end
end
if display
 disp('click/press any key');
 waitforbuttonpress;
```

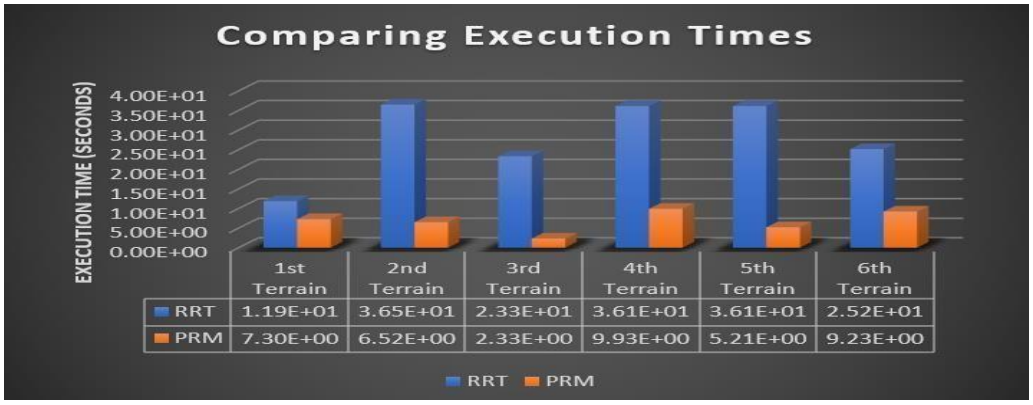
```
click/press any key
processing time=9.230730e+00
Path Length=1.433969e+03
>>
```

7. CONCLUSION



	1st Terrain	2nd Terrain	3rd Terrain	4th Terrain	5th Terrain	6th Terrain
RRT	8.10E+02	1.30E+03	9.25E+02	1.40E+03	1.39E+03	1.72E+03
PRM	6.86E+02	1.08E+03	9.25E+02	1.31E+03	1.23E+03	1.43E+03

From the above comparison bar graph, it can be observed that the path length covered is lesser in the PRM algorithm in comparison to the RRT algorithm. Therefore, for an autonomous vehicle that has its main objective



	1st Terrain	2nd Terrain	3rd Terrain	4th Terrain	5th Terrain	6th Terrain
RRT	1.19E+01	3.65E+01	2.33E+01	3.61E+01	3.61E+01	2.52E+01
PRM	7.30E+00	6.52E+00	2.33E+00	9.93E+00	5.21E+00	9.23E+00

to follow the shortest path length without taking time into consideration,then the PRM algorithm should be used for the implementation.

From the other comparison graph, it is clear that execution time is minimal in the PRM algorithmas compared to the RRT algorithm. For the vehicle to reach the goal in the minimum possible time without considering the path length, the PRM algorithm would be a better choice due to its lessertime complexity.
From the above two graphs, it is distinct that the PRM algorithm has better application compared to the RRT algorithm in terms of execution time and path length. Any one of two algorithms canbe used according to path length or time complexity as a priority.

The most optimal algorithm taking into consideration both the cases would be PRM since it is observed that there is vast variance between the execution time of PRM and RRT whereas there islesser variance between the path length. From the above comparative analysis, it can be concludedthat the PRM algorithm is more efficient than RRT in these conditions. However, there are methods and algorithms that may enable us to increase the execution time for RRT*. An extension of RRT would be to branch out the tree from both starting point and endpoint, and then merge thetwo trees.

Moreover, for this project to be implemented in actual practical situations, it requires a lot more fine-tuning and adjustments to the environment variables. For example, the following could be some features of the terrain that should be considered: tightness of the path, ruggedness, elevation levels, dirt composition, etc. Also, we must take into consideration dynamic obstacles. For the present cause, this proposed solution provides optimal results for simulation.

8. REFERENCES:

➢ Muhammad, N. R. Hasma Abdullah, M. A. H. Ali, I. H. Shanono and R. Samad, "Simulation Performance Comparison of A*, GLS, RRT and PRM Path Planning Algorithms," 2022 IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2022, pp. 258-263, doi: 10.1109/ISCAIE54458.2022.9794473.

➢ M. Imran Chowdhury and D. G. Schwartz, "The PRM-A Path Planning Algorithm For UUVs: An Application To Navy Mission Planning," Global Oceans 2020: Singapore – U.S.Gulf Coast, 2020, pp. 1-9, doi: 10.1109/IEEECONF38699.2020.9388987.

➢ Y. Shen, J. Liu and Y. Luo, "Review of Path Planning Algorithms for Unmanned Vehicles," 2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), 2021, pp. 400-405, doi: 10.1109/ICIBA52610.2021.9688064.

➢ S. Li, A. Yang and L. Chen, "Path Planning Algorithm on Large Scale Terrain Data Based on PRM with Non-Uniform Sampling Strategy," 2021 IEEE 2nd International Conferenceon Information Technology, Big Data and Artificial Intelligence (ICIBA), 2021, pp. 203- 207, doi: 10.1109/ICIBA52610.2021.9688227.

➢ Prabowo, Y.A., Trilaksono, B.R., Hidayat, E.M. and Yuliarto, B., 2022. Utilizing a Rapidly Exploring Random Tree for Hazardous Gas Exploration in a Large Unknown Area. *IEEEAccess*, *10*, pp.15336-15347.

➢ Z. Zhu, L. Li, W. Wu and Y. Jiao, "Application of improved Dijkstra algorithm in intelligentship path planning," 2021 33rd Chinese Control and Decision Conference (CCDC), 2021, pp. 4926-4931, doi: 10.1109/CCDC52312.2021.9602021.

➢ P. Zhong, B. Chen, S. Lu, X. Meng and Y. Liang, "Information-Driven Fast Marching Autonomous Exploration With Aerial Robots," in IEEE Robotics and Automation Letters,vol. 7, no. 2, pp. 810-817, April 2022, doi: 10.1109/LRA.2021.3131754.

➢ Pohan, M. A. R., Trilaksono, B. R., Santosa, S. P., & Rohman, A. S. (2021). Path Planning Algorithm Using the Hybridization of the Rapidly-Exploring Random Tree and Ant Colony Systems. *IEEE Access*, *9*, 153599-153615.

➢ Xinyu, W., Xiaojuan, L., Yong, G., Jiadong, S., & Rui, W. (2019). Bidirectional potentialguided rrt* for motion planning. *IEEE Access*, *7*, 95046-95057.

➢ J. Chen and J. Yu, "An Improved Path Planning Algorithm for UAV Based on RRT," 2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), 2021, pp. 895-898, doi: 10.1109/AEMCSE51986.2021.00182.

9. APPENDIX : SAMPLE CODE

Rapidly-Exploring Random Trees(RRT):

rert.m:

```
terrain=im2bw(imread('terrain5.bmp')); source=[10 10];
goal=[246 246];
stepsize=20;disTh=20;
maxFailedAttempts = 10000;display=true;
tic;
if ~feasiblePoint(source,terrain), error('source lies on an obstacle or outside map'); endif ~feasiblePoint(goal,terrain),
error('goal lies on an obstacle or outside map'); end
if display, imshow(terrain);rectangle('position',[1 1size(terrain)-1],'edgecolor','k'); end RRTree=double([source -1]);

failedAttempts=0; counter=0; pathFound=false;
while  failedAttempts<=maxFailedAttempts if rand < 0.5,
sample=rand(1,2) .* size(terrain);else
sample=goal;end
[A, I]=min( distanceCost(RRTree(:,1:2),sample) ,[],1); closestNode = RRTree(I(1),1:2);
theta=atan2(sample(1)-closestNode(1),sample(2)-closestNode(2));
newPoint = double(int32(closestNode(1:2) + stepsize * [sin(theta) cos(theta)])); if
~checkPath(closestNode(1:2), newPoint, terrain)failedAttempts=failedAttempts+1;
continue;end
if distanceCost(newPoint,goal)<disTh,pathFound=true;break; end
[A, I2]=min( distanceCost(RRTree(:,1:2),newPoint) ,[],1);if distanceCost(newPoint,RRTree(I2(1),1:2))<disTh,

failedAttempts=failedAttempts+1;continue; end

RRTree=[RRTree;newPoint I(1)];failedAttempts=0;
if display,

line([closestNode(2);newPoint(2)],[closestNode(1);newPoint(1)
]);
counter=counter+1;M(counter)=getframe;end
end
if display
disp('click/press any key');waitforbuttonpress;
end
if ~pathFound, error('no path found. maximum attempts reached'); endpath=[goal];
prev=I(1);
```

```
while prev>0 path=[RRTree(prev,1:2);path];prev=RRTree(prev,3);
end pathLength=0;
for i=1:length(path)-1, pathLength=pathLength+distanceCost(path(i,1:2),path(i+1,1:2)); end
fprintf('processing time=%d \nPath Length=%d \n\n', toc,pathLength); imshow(terrain);rectangle('position',[1 1
size(terrain)-1],'edgecolor','k'); line(path(:,2),path(:,1));
```

checkPath.m:

```
function feasible=checkPath(n,newPos,terrain)feasible=true;
dir=atan2(newPos(1)-n(1),newPos(2)-n(2));
for r=0:0.5:sqrt(sum((n-newPos).^2))posCheck=n+r.*[sin(dir) cos(dir)];
if ~(feasiblePoint(ceil(posCheck),terrain) && feasiblePoint(floor(posCheck),terrain) &&feasiblePoint([ceil(posCheck(1))
floor(posCheck(2))],terrain) && feasiblePoint([floor(posCheck(1))ceil(posCheck(2))],terrain))
feasible=false;break;

end
if ~feasiblePoint(newPos,terrain), feasible=false;end
end
```

distanceCost.m:

```
function h=distanceCost(a,b)
h = sqrt((a(:,1)-b(:,1)).^2 + (a(:,2)-b(:,2)).^2 );
```

feasiblePoint.m:

```
function feasible=feasiblePoint(point,terrain)feasible=true;

if ~(point(1)>=1 && point(1)<=size(terrain,1) &&point(2)>=1 && point(2)<=size(terrain ,2) &&
terrain(point(1),point(2))==1)  feasible=false; end
```

Probabilistic roadmap(PRM):

prm.m:

```
terrain=im2bw(imread('terrain5.bmp')); source=[10 10];
goal=[246 246];k=200;
display=true;
if ~feasiblePoint(source,terrain), error('source lies on an obstacle or outside map'); endif ~feasiblePoint(goal,terrain),
error('goal lies on an obstacle or outside map'); end imshow(terrain);
rectangle('position',[1 1 size(terrain)-1],'edgecolor','k')vertex=[source;goal];
```

```
if display,
rectangle('Position',[vertex(1,2)-5,vertex(1,1)-5,10,10],'Curvature',[1,1],'FaceColor ','r'); end if display,
rectangle('Position',[vertex(2,2)-5,vertex(2,1)-5,10,10],'Curvature',[1,1],'FaceColor ','r'); end tic;
while length(vertex)<k+2 x=double(int32(rand(1,2) .* size(terrain)));

if feasiblePoint(x,terrain), vertex=[vertex;x];if display,
rectangle('Position',[x(2)-5,x(1)-5,10,10],'Curvature',[1,1],'FaceColor','r'); end end end

if display
disp('click/press any key');

waitforbuttonpress;end edges=cell(k+2,1); for i=1:k+2
for j=i+1:k+2
if checkPath(vertex(i,:),vertex(j,:),terrain);edges{i}=[edges{i};j];edges{j}=[edges{j};i]; if display,
line([vertex(i,2);vertex(j,2)],[vertex(i,1);vertex(j,1)]); end
end end end
if display
disp('click/press any key');waitforbuttonpress;
end
Q=[1 0 heuristic(vertex(1,:),goal) 0+heuristic(vertex(1,:),goal) -1]; closed=[];pathFound=false;
while size(Q,1)>0[A, I]=min(Q,[],1);n=Q(I(4),:);
Q=[Q(1:I(4)-1,:);Q(I(4)+1:end,:)];
if n(1)==2 pathFound=true;break;end
for mv=1:length(edges{n(1),1})newVertex=edges{n(1),1}(mv);
if length(closed)==0 || length(find(closed(:,1)==newVertex))==0
historicCost=n(2)+historic(vertex(n(1),:),vertex(newVertex,:)); heuristicCost=heuristic(vertex(newVertex,:),goal);
totalCost=historicCost+heuristicCost;

add=true;

if length(find(Q(:,1)==newVertex))>=1I=find(Q(:,1)==newVertex);
if Q(I,4)<totalCost, add=false;

else Q=[Q(1:I-1,:);Q(I+1:end,:);];add=true;end
end if add
Q=[Q;newVertex historicCost heuristicCost totalCost size(closed,1)+1]; endend
end closed=[closed;n];end
if ~pathFound error('no path found')end
fprintf('processing time=%d \nPath Length=%d \n\n', toc,n(4));path=[vertex(n(1),:)];
prev=n(5); while prev>0
path=[vertex(closed(prev,1),:);path];prev=closed(prev,5);
end imshow(terrain);
```

```
rectangle('position',[1 1 size(terrain)-1],'edgecolor','k')line(path(:,2),path(:,1),'color','r');
```

checkPath.m:

```
function feasible=checkPath(n,newPos,terrain)feasible=true;
dir=atan2(newPos(1)-n(1),newPos(2)-n(2));
for r=0:0.5:sqrt(sum((n-newPos).^2))posCheck=n+r.*[sin(dir) cos(dir)];
if ~(feasiblePoint(ceil(posCheck),terrain) && feasiblePoint(floor(posCheck),terrain) &&feasiblePoint([ceil(posCheck(1))
floor(posCheck(2))],terrain) && feasiblePoint([floor(posCheck(1))ceil(posCheck(2))],terrain))
feasible=false;break;end
if ~feasiblePoint(newPos,terrain), feasible=false; endend
```

feasiblePoint.m:

```
function feasible=feasiblePoint(point,terrain)feasible=true;
if ~(point(1)>=1 && point(1)<=size(terrain,1) && point(2)>=1 && point(2)<=size(terrain,2) &&
terrain(point(1),point(2))==1) feasible=false;end
```

heuristic.m:

```
function h=heuristic(X,goal)h = sqrt(sum((X-goal).^2));
```

historic.m:

```
function h=historic(a,b)h = sqrt(sum((a-b).^2));
```