Asian Journal of Convergence in Technology **Volume V Issue I**
ISSN NO: 2350-1146 I.F-5.11

MULTI FUSION PATH PLANNING ALGORITHM FOR UGV

Sharana Basava S Madari, *DM, BEL,* A. Thirija Sharmila, *M (SRS)*, *BEL* and Adlinge S D, *SO, DIAT*

Abstract— **This research paper presents the work on the multi fusion path planning algorithm for Unmanned Ground Vehicle (UGV), which includes, tasks such as studying various existing algorithms, combing algorithms for better results, implementing multi fusion path planning algorithm to produce an optimal path for UGV in a simulated environment. To determine collision free path for a robot from start to goal position in a workspace comprising of obstacles, is the main challenge in the design of an autonomous UGV. The Multi fusion Path planning algorithm subsequently attempts to create free paths for the UGV to travel in the workspace without colliding with obstacles. Probabilistic roadmap (PRM) algorithm along with Dijkstra algorithm is used for the UGV navigation in an environment. Python is used for simulation of the results.**

Keywords— ***Path planning; Probabilistic Roadmap (PRM); Dijkstra algorithm; KNN (K-nearest neighbor); Unmanned Ground Vehicle (UGV).***

I. INTRODUCTION

Path planning is considered as one of the vital tasks for any autonomous robot. Sensors plays an important role for accessibility of environmental and odometric information. This depends upon how accurate the environmental and odometric information is attained by the robot. In our study robot means UGV. Path planning aims for moving UGV from their initial position to the goal position by their own actuators and strategies, and during the process, robots must always be able to avoid obstacles to maintain safety. Robots such as underwater robots [1,2], wall-climbing robot [3], and micro air vehicles [4-6] have been tested with different methods. Also, in path planning the robot must reach the destination location without colliding with obstacles and the path must be the shortest. The shortest path is considered as the time saving constraint for UGV and the safest path is considered as the safety constraints for UGV. To meet such requirement there are many path planning methods are already proposed by researchers.

Path planning algorithms are divided into five categories [7]; Fig.1 shows various categories of path planning algorithms.

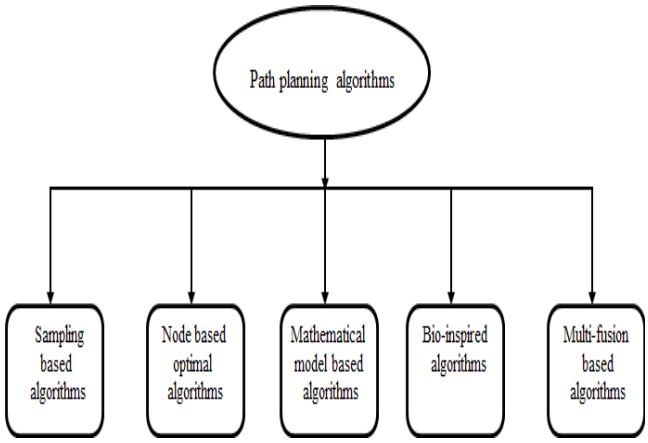

Fig. 1 Categories of path planning algorithms

We have studied and analyzed sampling-based algorithm and node based optimal algorithm from the path planning taxonomy.

II. SAMPLING BASED ALGORITHMS

Sampling-based path planning algorithms, such as Probabilistic Road Maps (PRM) and Rapidly-exploring Random Trees (RRT), have been working well in practice and possess theoretical guarantees such as probabilistic completeness. However, effort has been devoted to the analysis of the quality of the solution returned by such algorithms, e.g., as a function of the number of samples. This kind of methods needs some pre-known information of the whole workspace, that is, a mathematic representation to describe the workspace. This kind usually samples the environment as a set of nodes, or cells, or in other forms. Then map the environment or just search randomly to achieve a feasible path. Fig.2 shows the elements of sampling-based algorithms.

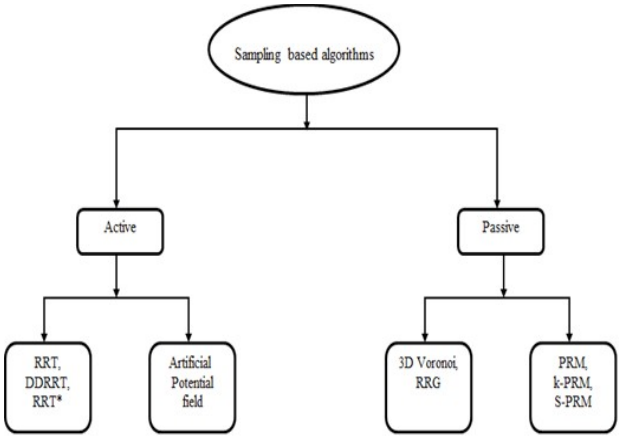

Fig. 2 Elements of sampling-based algorithms

A. Representation of environment:

In path planning, the UGV moves in a **workspace W** i.e**.,** either in two or three dimensions. The location of a UGV is represented as a configuration. A set of all possible configurations for a UGV is called a configuration space. If configurations have d parameters, the configuration space is d- dimensional. In each configuration, the robot occupies some set of points in a workspace W. This set of occupied points in configuration q is denoted as R(q). Workspace W contains obstacles, n obstacles denoted by Oi, $1 \leq i \leq n$. For each Oi there is a counterpart COi in the configuration space C. This configuration space obstacle can be defined as

$COi = \{q \in C \mid R(q) \cap Oi = 0\}$,

means that COi is a set of all configurations where the robot R would collide with the obstacle Oi. The free configuration space can be defined as

$C\ free = C - U\ COi$,

and it is a set of configurations where the UGV does not collide with any of the obstacles. The task in motion planning can be defined as finding a free path from configuration q start to configuration q goal. The path is a continuous function

$\tau: [0, 1] \rightarrow C\ free$,

where $\tau(0) = q\ start$ and

$\tau(1) = q\ goal$.

B. Analysis of Sampling-Based Algorithms

Table 1: Analysis of sampling-based algorithms

Analysis of Sampling Based algorithms		
Method Type	**Short comings**	**Advantages**
RRT	Single Path, single query planner, Nonoptimal, Static threat only, Required much time	fast searching ability
PRM	Expensive collision check, Nonoptimal	Appropriate for complex environment and replanning situations, multi query planner, Probabilistically complete
Voronoi	Incomplete representation, Non-convergence, static threat only, unnatural attraction to open space, suboptimal paths.	Easy to implement on-line and ignoring collision checking
Artificial potential	local minima	Fast convergence

We cannot predict the environment of UGV whether it operates in complex or easy environment, also by considering advantages of replanning situation and multi query planner among the various methods of sampling-based algorithm, we have chosen PRM for our multifusion algorithm.

C. Probabilistic Roadmap:

PRMs are sampling-based methods which do not try to construct an exact representation of C free. Instead, they utilize the fact that it is quite easy to check whether some configuration is collision-free by using some collision detection algorithms [1, 2]. With these algorithms it is also possible to check collisions or a local path which is some simple path segment between two configurations. Using these methods, PRM planners can build a roadmap that lies entirely in C free. A typical PRM planner contains the following parts:

- A sampling method to generate new configurations,
- A method to select neighbor configurations from the roadmap,
- A local planner to connect configurations together with a local path, and
- An ending condition which is used to decide when the roadmap is ready.

The PRM algorithm works in two phases. In the first phase, the roadmap is constructed and in the second phase it is used to answer queries. The learning phase is the most time-consuming part of PRM planners but after the roadmap has been built, it is very fast to solve the motion planning queries. This is very useful especially

when multiple queries must be solved. This also distinguishes the PRM planners from many other motion planning algorithms like RRT which often can solve only one query at a time.

D. Roadmap construction:

Algorithm 1 shows how the PRM method constructs a roadmap.

1. V=0
2. E=0
3. Repeat
4. q=a randomly chosen configuration from C free
5. V=VU {q}
6. Nq=all nearest neighbor configuration of q chosen from V.
7. For all q' in Nq do
8. if the local planner Δ can find a free path between q & q' then
9. E=EU{(q,q')}
10. end if
11. end for
12. until there are enough configurations in V
13. return G

Output:
A roadmap G= (V, E).

Initially the algorithm starts with an empty roadmap. The main loop is in between lines 3–12. In line 4, at the beginning of one iteration, a free configuration q is generated and in line 5 this new configuration is added to the roadmap as a node. The free configuration is generated randomly using uniform sampling method. In line 6, a set of neighbour configurations are chosen for q from the roadmap. We have to select predetermined number of the nearest configurations in program. Then, in lines 7–11, the algorithm goes through all these neighbours. For each neighbour q0, a local planner is used in line 8 to check whether there is a simple and free path between q and $q0$. If the local planner finds a free path, an edge (q, $q0$) is added to the roadmap in line 9. We have used **kd-tree** method for nearest neighbour search. There are different variations of the method but it is always based on the binary tree. In a typical kd-tree implementation, the points, which in PRM planners are the configurations from the roadmap, are stored to the nodes of the binary tree. Each node stores one point and each node also divide the space into two partitions by a plane that goes through the stored point. The plane is used to divide the remaining points into the sub trees of the node. The nearest neighbours can now be searched for quickly by using this structure because it allows to eliminate the large regions of the search space during the search. The nodes are added to the roadmap until some ending condition has been met. This ending condition is defined in program. The roadmap should have a good coverage and connectivity at the end.

E. Solving queries:

Algorithm 2 shows how a roadmap can be used to solve motion planning query. As input parameters, the algorithm

needs a previously constructed roadmap G = (V, E), a start configuration q-start, and a goal configuration q-goal. The algorithm tries to connect q-start and q-goal to the roadmap and then find and return a free path between those configurations.

1. N-start= all nearest neighbour configuration of q-start chosen from V.
2. N-goal= all nearest neighbour configuration of q-goal chosen from V.
3. V=V U {q-start}
4. V=V U {q-goal}
5. For all q'in N-start do
6. If the local planner can find a free path between q- start & q'then
7. E=E U {(q-start,q')}
8. End if
9. End for
10. For all q'in N-goal do
11. If the local planner can find a free path between q-goal and q' then
12. E=E U {(q-goal,q')}
13. End if
14. End for
15. P=a shortest path from q-start to q-goal
16. If P is empty then
17. Return not found
18. Else
19. Return P
20. End if

Input:
G= (V, E); the roadmap
q-start; the start configuration
q-goal; the goal configuration
Output:
A shortest path in G from q-start to q-goal or not found if a path cannot be found.
In lines 1 and 2, the algorithm chooses a set of the nearest neighbour configurations for both q-start and q-goal. Then both configurations are added to the roadmap in lines 3 and 4. Ideally, all configurations from the roadmap should be selected as neighbours to maximize the probability that q-start and q-goal can be connected to the roadmap. In practice, it is usually enough to select only some of the nearest nodes. In lines 5–9, there is a loop that goes through all neighbours for q-start. For each neighbour $q0$, the algorithm checks whether a local planner can find a free path from q-start to $q0$. If such path exists, an edge (q-start, $q0$) is added to the roadmap. In lines 10–14, the same is

done for *q*-goal. The path that the algorithm returns is a sequence q-start = v0, e1, v1, e2, v2. . . en, vn = q-goal, where vi Ɛ V, 0 ≤ i≤ n and ej ƐE, 1 ≤ j ≤ n. To retrieve the actual path in C, a local planner Δ must be used to calculate local paths for each edge. The corresponding local path for edge ej is Δ (vj-1, vj) and the actual path from q-start to q-

goal can be composed by concatenating all these local paths together.

III. NODE BASED OPTIMAL ALGORITHMS:

Node based optimal algorithms explore through the decomposed graph. This kind of methods can always find an optimal path according to the certain decomposition. Figure 3 illustrates the typical elements of node based optimal algorithms

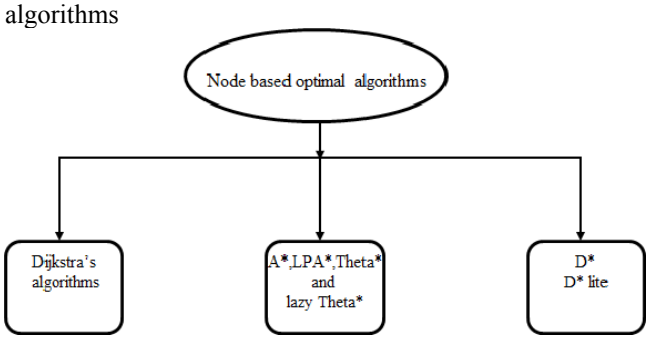

Fig. 3 Elements of Node based optimal algorithms

D. Analysis of node based Algorithms

Table 2: Analysis of node-based optimal algorithms

Analysis of Node Based optimal algorithms		
Method Type	**Short comings**	**Advantages**
Dijkstra's algorithm	High time complexity, static threat only	Easy to implement for various environments.
A*	Heavy time burden, Non-smoothness, Static threat only	Fast searching ability, enabling implementation on-line
D*	Unrealistic distance	Fast searching ability, dealing with dynamic environments

Among the various methods of node-based optimal algorithm, we have chosen Dijkstra algorithm by considering advantages of implementing for various environment. In line 15 of the algorithm 2, we have used Dijkstra algorithm to find a path between *q*-start and q-goal from the roadmap graph. If the path is found, it is returned. Otherwise the algorithm returns Not Found which informs that the path could not be found.

B. Dijkstra's algorithm:

an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks, in our case PRM roadmap. The node at which we are starting be called as initial node. The distance of node Y be the distance from the initial node to Y. This algorithm will assign some initial random distance values and will try

to improve them step by step. Mark all nodes unvisited. Create a set of all the unvisited nodes called the unvisited set. Assign to every node a tentative distance value, set it to zero for our initial node and to infinity for all other nodes.

Set the initial node as current. For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. Otherwise, keep the current value. When we are done considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again. If the destination node has been marked visited or if the smallest tentative distance among the nodes in the unvisited set is infinity then stop. The algorithm has finished. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to initial steps. When planning a route, it is actually not necessary to wait until the destination node is "visited" as above, the algorithm can stop once the destination node has the smallest tentative distance among all "unvisited" nodes.

they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

IV. MULTI FUSION ALGORITHMS

We multifused the PRM from Sampling Based algorithms with Dijkstra's algorithm from Node Based Optimal Algorithms. We simulated multi fused algorithms using python code.

V. SIMULATION:

Simulated results for different ending conditions are as follows:

A. N_SAMPLE = 350 # number of sample points
N_KNN =10#number of edges from one sampled point
MAX_EDGE_LEN=30.0 # [m] Maximum edge length

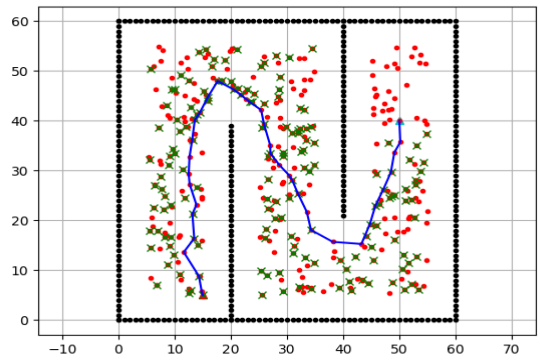

Fig. 4. Simulation Result -1.

probabilistic_road_map_work.py~ start!!
goal is found!

B. N_SAMPLE = 500 # number of sample points
N_KNN =10#number of edges from one sampled point
MAX_EDGE_LEN = 30.0# [m] Maximum edge length

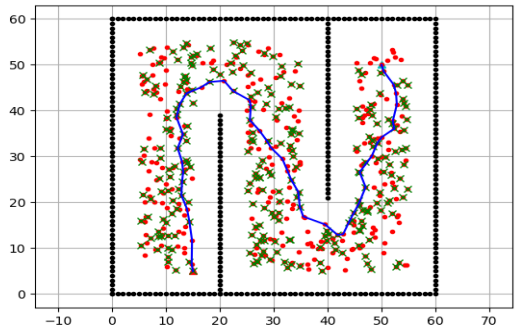

Fig. 5. Simulation Result-2

probabilistic_road_map_work.py~ start!!
goal is found!

C. N_SAMPLE = 100 # number of sample points
N_KNN=20#number of edges from one sampled point
MAX_EDGE_LEN=50.0 # [m] Maximum edge length

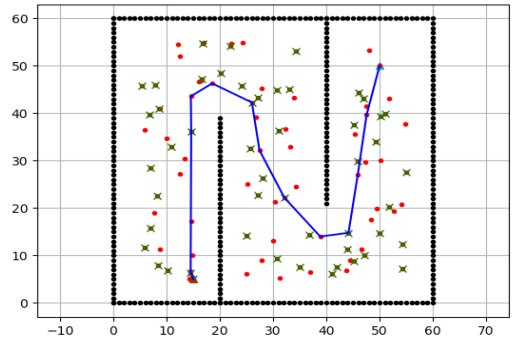

Fig. 6. Simulation Result-3

probabilistic_road_map_work.py~ start!!
goal is found!

D. N_SAMPLE = 50 # number of sample points
N_KNN=20#number of edges from one sampled point
MAX_EDGE_LEN=10.0 # [m] Maximum edge length

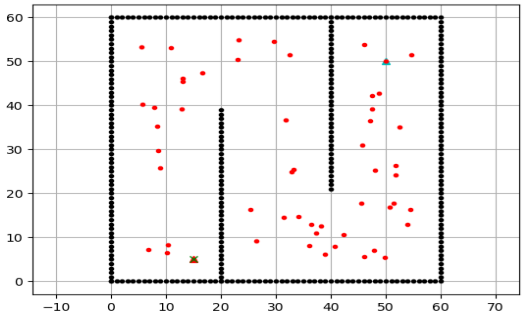

Fig. 7. Simulation Result-4

probabilistic_road_map_work.py~ start!!
Cannot find path

E. N_SAMPLE = 10000 # number of sample points
N_KNN=20#number of edges from one sampled point
MAX_EDGE_LEN = 20.0 #[m]Maximum edge length

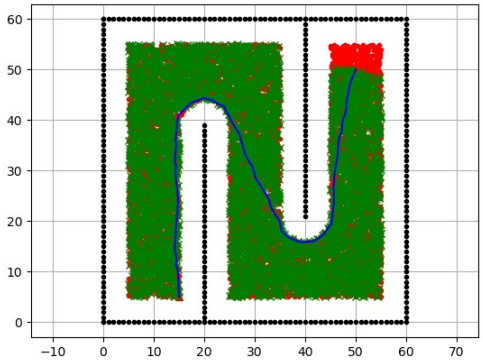

Fig. 8. Simulation Result-5

probabilistic_road_map_work.py~ start!!
goal is found!

F. N_SAMPLE = 500 # number of sample points
N_KNN=20#number of edges from one sampled point

MAX_EDGE_LEN =20.0# [m] Maximum edge length

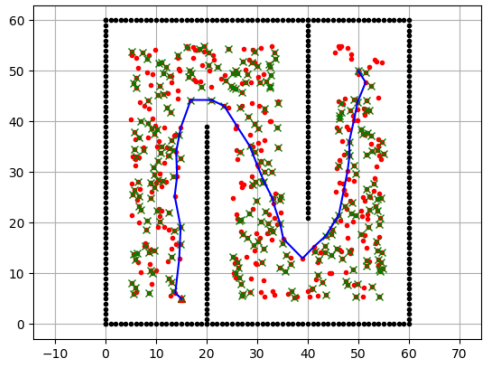

Fig. 9. Simulation Result-6

probabilistic_road_map_work.py~ start!!
goal is found!

VI. CONCLUSION

We carried out simulation for different ending conditions, simulation results of A & B reach goal with sufficient number of sample points and edge length of 30 m. Results in C reaches goal with a smaller number of samples and increased edge length. Results in D not able to reach goal because of less samples and less edge length. In E though it reaches goal but it took more time to find a path due to large number of samples. In this type of multifusion

of algorithm we should not give too a smaller number of samples and too a greater number of samples.
Practically we tried several times by giving 500 samples points, 20 number of edges and edge length of 20 m. With this ending condition it able to find a path to reach a goal without fail. It is very difficult to solve the path planning problem exactly especially if the configuration space is complex and high-dimensional. Therefore, the problem is usually solved with approximate methods in practice.
In this paper, a path planning method based on a Multifusion of PRM & Dijkstra's algorithm is carried out to find a most safe path for UGV of particular dimension. With implementation of proposed method, the robot does not collide with the obstacles. However same algorithm can be extended for the dynamic obstacle by using roadmap with cycles with A* and D* algorithm.

Acknowledgment

The authors wish to acknowledge Mr. Manoj Jain, Chief Scientist of CRL, BEL, Bangalore, Mr. Srivathsa M. R, GM, NS-2 and CRL Robotics Team for their continuous motivation and guidance.

References

[1] N. K. Yilmaz, C. Evangelinos, P. F. J. Lermusiaux, and N. M. Patrikalakis, "Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming,"IEEE Journal of Oceanic Engineering,vol.33,no. 4,pp.522–537,2008.

[2] M. P. Aghababa, "3D path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles," Applied Ocean Research, vol. 38, pp. 48–62,2012.

[3] R.Yue, J.Xiao, S.L.Joseph, and S.Wang, "Modeling and path planning of the city-climber robot part II: 3D path planning using mixed integer linear programming," in Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '09), vol. 6, pp. 2391–2396, Guilin, China, December 2009.

[4] F.Yan,Y.-S.Liu,andJ.-.Xiao,"Pathplanningincomplex3D environments using a probabilistic roadmap method," International Journal of Automation and Computing,vol.10,no.6,pp. 525–533,2013.

[5] H. Duan, Y. Yu, X. Zhang, and S. Shao, "Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm," Simulation Modelling Practice and Theory, vol. 18, no.8, pp.1104–1115,2010.

[6] F. Schler, 3d path planning for autonomous aerial vehicles in constrained spaces [Ph.D. thesis], Department of Electronic Systems, Faculty of Engineering and Science, Aalborg University, Aalborg, Denmark,2012.

[7] " Survey of Robot 3D Path Planning Algorithms"
LiangYang, JuntongQi, DaleiSong, JizhongXiao, JiandaHan and YongXia

BIOGRAPHY

Sharana Basava S Madari has received his B.E degree in Electronics and Communications from VTU, Belgaum. He joined Bharath Electronics Limited, Bangalore,India in the year 2007 and he is working as the Deputy Manager . Currently he is pursuing MTech in Robotics from DIAT, Pune. His areas of interest includes path planning for autonomous UGV, Machine learning, Artificial intelligence.

A.Thirija Sharmila has received her M.E degree in VLSI Systems from National Institute of Technology, Tiruchirapalli. She is working as Member (Senior Research Staff) at Central Research Laboratory of Bharat Electronics Limited, Bangalore, India since 2001. She had worked in areas covering bulk encryption, SDH, OTU and LTE technologies. Currently she is working in the field of Robotics and Artificial Intelligence.

Sudam D Adlinge Received ME degree in Electronics Instrumentation from Andhra University, Visakhapatnam, Presently he is working as a Scientific Officer at DIAT, Pune. Currently he is pursuing PHD in Instrumentation and Control Engineering from Pune University. He has worked for Indian Navy, DRDO and Mahindra Defence Naval System. His areas of interest are Non linear control systems, sensors for defence Robotics, unmanned sea surface vehicles and Autonomous underwater vehicles.