2020 59th Annual Conference of the
Society of Instrument and Control Engineers of Japan (SICE)
September 23-26, 2020, Chiang Mai, Thailand

Shortest Path Evaluation with Enhanced Linear Graph and Dijkstra Algorithm

Tanishk Dudi[1†], Rahul Singhal[2] and Rajesh Kumar[2]

[1]Department of Mechanical Engineering, Malaviya National Institute of Technology, Jaipur, India
(E-mail: 2017ume1366@mnit.ac.in)
[2]Department of Electrical Engineering, Malaviya National Institute of Technology, Jaipur, India
(E-mail: 2015ree9544@mnit.ac.in , rkumar.ee@mnit.ac.in)

Abstract: Path planning is one of the vital tasks in the intelligent control of autonomous robots. It is of prime importance from industrial as well as commercial point of view. Various autonomous vehicles are gaining popularity to reduce the on-field intervention of humans. The path planning problem is divided into two sub-problems: finding the feasible node pairs and then evaluate the shortest feasible path from the obtained feasible node pairs. Feasible node pair is the part of the path which do not pass through any obstacle. For the known static arena, the most popular method to solve these kinds of problems is the visibility graph (VGraph), but it is computationally intensive. This paper proposes an Enhanced Linear Graph (ELGraph), a method for finding feasible node pairs. The ELGraph has simplified linear structure which exploits the concept of line segment intersection, which may take low computational time. The simulations were carried out for VGraph and ELGraph to get the feasible node pair, and then Dijkstra algorithm evaluates the shortest feasible path. The results obtained shows that ELGraph takes lesser evaluation time and is computationally lesser intensive than VGraph, without compromising on finding the shortest possible feasible path.

Keywords: Path planning, Dijkstra algorithm, Visibility graph, Shortest path problem, Autonomous robot.

1. INTRODUCTION

Path planning is one of the important tasks in intelligent control of an autonomous robots, it has a vast scope in robotics such as in terrain vehicles [1, 2], unmanned aerial vehicles (UAVs) [3, 4], autonomous underwater vehicles (AUVs) [5, 6]. Various techniques for applications such as threat avoidance [7, 8], finding shortest path [9, 10], collision avoidance have been developed in the field of path planning over the past decade.

Finding the shortest feasible path [11] is important for the purpose of navigation of autonomous robots, which is to find the best possible collision-free path between a starting node and a destination node for a known or unknown arena. Path planning for unknown environment is done using simultaneous localization and mapping (SLAM) [12-14], whereas path planning in known environment (static or dynamic) is done using potential field method [15-17], visibility graph method (VGraph) [16, 18] and grid method [17, 19, 20]. The potential field method is widely applied for determining the real-time shortest possible path because of its simplified structure. The limitation of the potential field method is that for multi-modal surfaces the evaluation of shortest possible path may not be optimal because the method gets trapped inside the local minima which may result in oscillation in shortest path between the tightly packed obstacles boundaries. The visibility graph is a graph of inter visible nodes. Inter visible nodes are the pair of nodes visible from each other in an arena. Visibility graph is computationally intensive due to its exhaustiveness and time complexity of $O(n^3)$. The limitation for the grid method is difficulty in determining the appropriate size of the grid. The gird size plays a critical role in the evaluation of the shortest possible path and precision for the arena information. The algorithms used for path planning includes fuzzy logic, artificial intelligence (AI) which includes artificial neural networks (ANN), reinforcement learning (RL) [21] and meta-heuristic algorithms such as ant colony optimization (ACO), particle swarm optimization (PSO) and genetic algorithm (GA).

Various methods have been developed for shortest path based on the arena specifications, the type of robots, the nature of the application as there is no generic technique which can be applied in all conditions. Typically, for evaluation of shortest feasible path, total distance of the path is of prime consideration for an objective formulation. The other factors which may also act as deciding factor for the shortest feasible path are: computation time, total energy consumption, and smoothness in control action. Dijkstra, Floyd–warshall [22, 23] and A* algorithms are used for shortest path evaluation based on the information regarding the obstacles present in the arena

The shortest path evaluation for the known and static

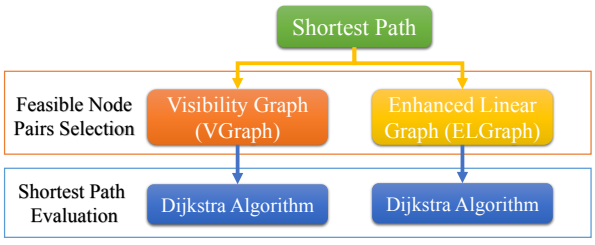


Fig. 1: Shortest path evaluation flow

† Tanishk Dudi is the presenter of this paper.

environment is a two-level problem, which comprises of a selection of feasible node pairs and shortest path evaluation based on the obtained feasible node pairs as shown in Fig. 1. The usual convention is to select the vertices of the obstacles, starting position of robot and destination coordinates as nodes. Then, the role of the selection of feasible node pair is to determine all node pair from the possible combination of the two nodes, which do not pass through any obstacles. The role of the shortest feasible path planning algorithm is to determine the sequence of the node pair which gives the lowest objective value, which usually is finding the least possible distance from the start position and destination coordinated formed by the sequence of node pair.

In this paper, Enhanced Linear Graph (ELGraph) is presented for feasible node pairs selection. The work focuses on node pair selection in an arena containing obstacles as convex polygons so that node pairs intersecting any obstacles is removed and therefore collision can be avoided. The approach is thoroughly discussed in the paper and compared with the commonly used conventional visibility graph (VGraph) [16, 24]. Finally the feasible node pairs obtained are fed as input to Dijkstra algorithm to find the shortest possible path. The flow comparison of the paper is presented in Fig. 1.

The structure of this paper as follows. In section 2, proposed ELGraph is presented. In section 3, visibility graph method is represented which helps in determining the feasible node pairs. In section 4, Dijkstra algorithm is represented for evaluation of shortest path planning. The results are presented and compared in section 5. In section 6, the conclusions were drawn based on the results obtained.

2. ENHANCED LINEAR GRAPH (ELGRAPH)

Feasible node pairs play a crucial role in the shortest path evaluation. These are the line segments which act as the building block for the feasible path not passing through any obstacle, these node pairs are present between the nodes of the same obstacle as well as between nodes of different obstacles.

The proposed ELGraph method focuses on the line segment made by the nodes of the obstacles, so as to reject node pairs passing through obstacle, for the arena in consideration these obstacles are shown by purple region in Fig. 2.

The flow of the ELGraph is as follows. Initially, all nodes are determined from the arena information, which includes the edges of all obstacles, starting node and destination node. Then all possible combination of two-node as node pair checked for the feasible node pair, which mainly check for the node pair not passing through any edges of all obstacles. The feasible node pair evaluation is divided based on the node pair consideration and the obstacle consideration. For checking the node pair whose both the nodes lying on the same obstacles are only checked for that obstacle, as this node pair can only

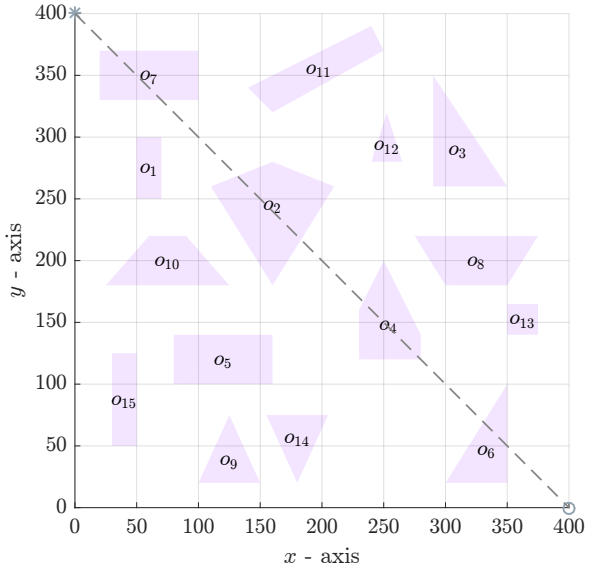


Fig. 2: Obstacles shown as ■, line of sight l as $-\ -$, start node as $*$, and destination node as $\circ$.

be the edge of the obstacle or pass through the obstacle. For the node pair, whose both nodes are from different obstacles is checked through all obstacles. Table 1 gives summary about the strategies used for obstacle consideration in ELGraph algorithm with possible scenarios encounter in its formulation. The node pairs which do not give collision free path and for the purpose of their removal properties of line segments and signs, and some additional aspects which will be thoroughly discussed. For the execution of ELGraph the nodes of each obstacle need to be defined in a sequence i.e. either clockwise or counter-clockwise. The algorithm is briefly explained in the below subsections with various possible scenarios that need to be considered while formulating it, these scenarios deal with the removal of non-feasible node pairs present within an obstacle and between obstacles. The scenarios are formed to give a better insight about the working of ELGraph.

The equation of line passing through the two points (x_1, y_1) and (x_2, y_2) is given by:

$$(y - y_1) - m(x - x_1) - c = 0 \tag{1}$$

$$l(x, y) = (y - y_1) - m(x - x_1) - c \tag{2}$$

Where, m is the slope and can be calculated by

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} \tag{3}$$

From (2), if a point is introduced on a line, then $l(x, y) = 0$, but when $l(x, y) > 0$ or $l(x, y) < 0$ then point is present on either side of the line but not on the line. This property is highly exploited.

Table 1: ELGraph node pair $\overline{p_{j_1}^{i_1}\ p_{j_2}^{i_2}}$ feasibility checking through o_{i_m} obstacle.

Strategy	Obstacle consideration	Description
Intra-Obstacle	$o_{i_m} = o_{i_1}$, where $i_1 = i_2$	Node pair made of same obstacle (o_{i_1}) vertices, feasibility check by edges of same obstacle
Inter-Obstacle	$(i_1 \neq i_2)$	Node pair between obstacles, feasibility check by edges of all obstacles.
Scenario 1 :	$o_{i_m} = o_{i_1}$	Node pair feasibility check by edges of obstacle (o_{i_1}).
Scenario 2 :	$o_{i_m} = o_{i_2}$	Node pair feasibility check by edges of obstacle (o_{i_2}).
Scenario 3 :	$\forall o_{i_m}$, where $i_m \neq i_1 \neq i_2$	Node pair feasibility check by edges of obstacles other than o_{i_1} and o_{i_2}.

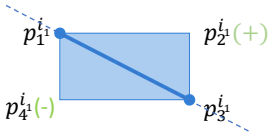


Fig. 3: Intra-obstacle Node Pair

2.1. Intra-Obstacle Feasible Node Pair Selection

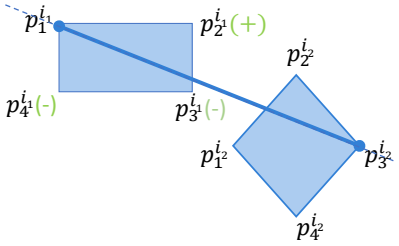
Feasibility of node pair is checked by the same obstacle, Fig. 3 contains an obstacle having four vertices $o_{i_1} = (p_1^{i_1}, p_2^{i_1}, p_3^{i_1}, p_4^{i_1})$, the feasibility of node pair $(p_1^{i_1}, p_3^{i_1})$ is checked by deducing the equation of a line segment made by the points $p_1^{i_1}$ and $p_3^{i_1}$ using (1) and (3) now once the equation is deduce points $p_2^{i_1}$, $p_4^{i_1}$ are inserted in the equation obtained in form (2), these are the points other than the ones forming the segment on the obstacle and the obtained signs are added into an empty list, if the list contains opposite signs the node pair will be redundant and need to be rejected, and if the the signs in the list are all either '+' or '-', then the pair is to be selected.

2.2. Inter-Obstacle Feasible Node Pair Selection

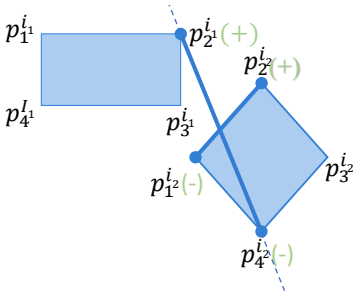
In this feasibility is checked for node pairs between different obstacles. Rejecting redundant node pairs present on different obstacles a slightly different and more complex approach is devised. For the explanation Fig. 4a contains two obstacles having four vertices each namely $o_{i_1} = (p_1^{i_1}, p_2^{i_1}, p_3^{i_1}, p_4^{i_1})$ and $o_{i_2} = (p_1^{i_2}, p_2^{i_2}, p_3^{i_2}, p_4^{i_2})$, here the approach is showcased with the help of 3 scenarios:

SCENARIO 1: The node pair is rejected when checking with nodes of the different obstacle, it should be checked through each edge of obstacle. For this purpose let us consider $\overline{p_1^{i_1}\ p_3^{i_2}}$ in Fig. 4a intersecting both the obstacles, now with the same approach as discussed in intra-obstacle feasible node pair, first nodes of o_{i_1} are injected, then o_{i_2} in the equation of the $\overline{p_1^{i_1}\ p_3^{i_2}}$ and follow the same conditions as in section 2.1, which would reject the node pair $(p_1^{i_1}, p_3^{i_2})$

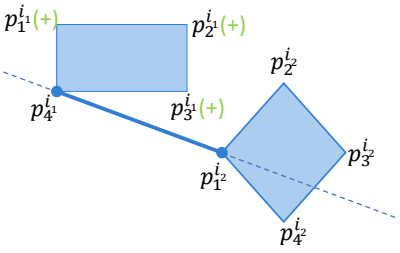
SCENARIO 2: The node pair is accepted when checking with one obstacle but get rejected when checking with nodes of the other obstacle. For this purpose let us consider $\overline{p_2^{i_1}\ p_4^{i_2}}$ in Fig. 4b intersecting obstacle o_{i_2}, also need to consider the line segments which defines the boundary of the obstacles. For this condition a



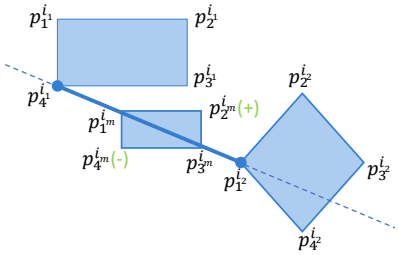
(a) Scenario 1



(b) Scenario 2a



(c) Scenario 2b



(d) Scenario 3

Fig. 4: Inter-obstacle Node Pair

two-way approach is applied. The nodes of the obstacle o_{i_2} are injected in the equation of $\overline{p_2^{i_1}\ p_4^{i_2}}$, the nodes $p_1^{i_2}$ and $p_2^{i_2}$ will give opposite signs and now to be sure that $\overline{p_2^{i_1}\ p_4^{i_2}}$ intersects $\overline{p_1^{i_2}\ p_2^{i_2}}$ the other way round by injecting

the nodes $p_2^{i_1}$ and $p_4^{i_2}$ in the equation of $\overline{p_1^{i_2}\ p_2^{i_2}}$ and in this case it will give opposite signs which means the two line segments intersect each other, so the node pair $(p_2^{i_1}, p_4^{i_2})$ must be eradicated from the solution space.

For node pairs $(p_4^{i_1}, p_1^{i_2})$ in Fig. 4c and the line segment formed by it $\overline{p_4^{i_1}\ p_1^{i_2}}$ intersecting no obstacle, here also the two-way approach needs to be applied, signs are checked by inserting nodes of o_{i_2} (as nodes of o_{i_1} gives similar sign) in equation of $\overline{p_4^{i_1}\ p_1^{i_2}}$ in sequential order, in this case $\overline{p_4^{i_1}\ p_1^{i_2}}$, $\overline{p_1^{i_2}\ p_2^{i_2}}$ and $\overline{p_1^{i_2}\ p_4^{i_2}}$ have a common node, now the other way round considering line segment $\overline{p_1^{i_2}\ p_2^{i_2}}$ and $\overline{p_1^{i_2}\ p_4^{i_2}}$ and inserting the nodes $p_1^{i_2}$ and $p_4^{i_1}$, do not get opposite signs and so the node pair should not be eradicated.

SCENARIO 3: The node pair may be accepted by both the obstacles having each node of the node pair but it may be rejected by the other obstacles present in the arena. For this purpose let us consider three obstacles $o_{i_1} = (p_1^{i_1}, p_2^{i_1}, p_3^{i_1}, p_4^{i_1})$, $o_{i_2} = (p_1^{i_2}, p_2^{i_2}, p_3^{i_2}, p_4^{i_2})$, and $o_{i_m} = (p_1^{i_m}, p_2^{i_m}, p_3^{i_m}, p_4^{i_m})$, and check the node pair $(p_4^{i_1}, p_1^{i_2})$ as shown in Fig. 4d, the line segment $\overline{p_4^{i_1}\ p_1^{i_2}}$ do not intersects the polygon on which these nodes are present but it is intersecting a polygon other than them (here its o_{i_m}) and so this node pair need to be eliminated and so the conditions discussed in above cases for node pair checking needs to be applied to every obstacle present in the arena.

3. VISIBILITY GRAPH

The vertices of an obstacle [25] are connected to that of another by a collection of node pair, these collection of node pairs make visibility graph. The visibility graph obtain contain $O(n^2)$ edges, which can be constructed in $O(n^2)$ time and space in two dimension, where n is the number of vertices.

Generalized visibility graph is the extension of the simple visibility graph, where the obstacles are generalized polygons, i.e., regions bounded by straight segments or circular arcs. The naive algorithm compares every pair of points in the set of obstacles and check if they intersect with any edges of obstacles, it has $O(n^3)$ time complexity.

Visibility graph is used to find feasible node pairs for the set of obstacles in the two dimensional plane, finding shortest distance problem is divided into two subproblems: constructing the visibility graph, and determining the shortest path using algorithm such as Dijkstra's algorithm to the feasible node pairs.

4. DIJKSTRA ALGORITHM

The Dijkstra's algorithm [26] gives the shortest path and distance from the starting node to the destination node while taking the least possible distance between them, it is based on greedy approach. Dijkstra's algorithm is implemented by continuously calculating the shortest distance of the starting node with the various other nodes in the graph and excluding longer distances routes while making an update. The function for distance is formulated as:

$$dist(d) = min(dist(d), (dist(p) + length(p, d))) \quad (4)$$

Where, $dist(d)$ is the least distance from the source to the destination node d, p is the previous adjacent node and $length(p, d)$ represents the distance between the nodes p and d.

Tentative distance values are assigned to every node: infinity for the routes that have never been explored, from algorithm point of view it can be any number of very high in magnitude. The starting node is assigned zero and taken as current node initially. For the current node all its unvisited adjacent nodes are considered and tentative distances is calculated from the current node. The current assigned value is then compared with the recently calculated tentative distances assign the smaller one. The function in (4) is the central idea for assigning the shortest distance in the algorithm. The overall step by step procedure for the Dijkstra algorithm is given in Algorithm 1.

Algorithm 1: Dijkstra Algorithm

```
Result: Shortest distance
Function Dijkstra(Nodeset, source, goal):
for p in Nodeset do
    if p != source then
        shortest_dist[p] = infinity;
        predecessor[p] = undefined;
    else
        shortest_dist[source] = 0;
    end
    Add node p to temp (minimum priority
    queue);
end
while temp != empty do
    node_t = node u having min shortest_dist[u] in
    temp;
    temp.pop(node_t);
    for adjacent node p of node_t do
        wgt = shortest_dist[node_t] +
        length(node_t, p);
        if wgt < shortest_dist[p] then
            shortest_dist[p] = wgt;
            predecessor[p] = node_t;
        end
    end
end
Return shortest_dist[], predecessor[]
```

5. RESULTS AND DISCUSSION

The simulations were carried out for the shortest path evaluation for the static arena with convex obstacles shown in Fig. 2. For this, feasible nodes were obtained by the Vgraph method and proposed ELGraph method, then shortest path had been determined by Dijkstra algorithm for the obtained feasible node pairs. The whole evalua-

Table 2: Results comparison

		Case I : VGraph	Case II : ELGraph
Arena Consideration	No. of Nodes	58	58
Feasible Node Pairs	No. of Feasible Node Pairs	778	778
	Evaluation Time (sec)	0.65982	0.44469
Shortest Path Evaluation	Dijkstra Algorithm (sec)	0.33976	
	Shortest Distance (m)	584.3729	

tion procedure was implemented in Python working on windows 10 operating system executing on the Intel i5 processors with 8 logical cores having base operating frequency of 2.30 Ghz.

The obstacle vertices were considered and checked for the feasible node pairs. The same numbers of feasible node pairs were obtained from the VGraph method and the ELGraph method as shown in Table 2. The obtained feasible node pairs were found identical for both methods and represented in Fig 5. The evaluation time taken by the ELGraph method was found lesser due the simplified linear structure, when compared with VGraph method. The shortest path was obtained using Dijkstra algorithm, which is an exhaustive search algorithm and always yield the same output for the same input information. Therefore, Dijkstra algorithm was executed once because of the same set of feasible node pairs obtained from the both the methods. The obtained shortest path is shown in Fig 6.

From results presented in Table 2, it can be inferred that ELGraph when compared to VGraph the number of nodes taken into consideration are same, the number of node pairs obtain are also same but the execution time for ELGraph was 1.5 times lesser than VGraph as ELGraph uses linear structure which makes it computationally efficient.

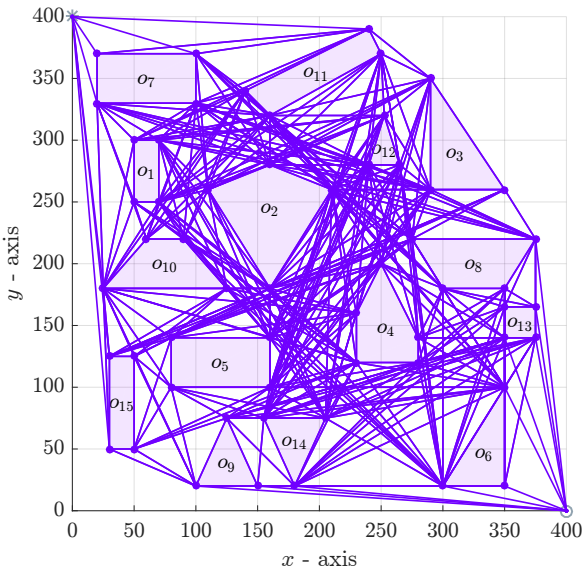


Fig. 5: Feasible node pairs as ●–● for complete arena.

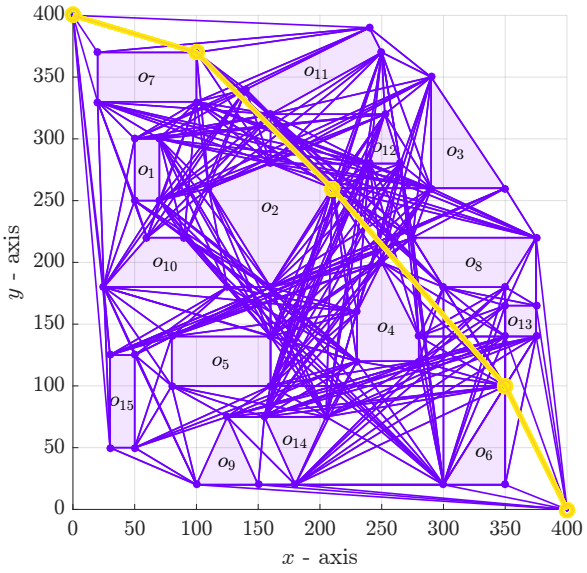


Fig. 6: Shortest feasible path by Dijkstra algorithm as ○–○.

6. CONCLUSION

This paper presented a simple yet effective Enhanced Linear Graph (ELGraph) method for the feasible node pair selection. Feasible node pairs play vital role in path evaluation for the path planning problems. In this, the feasible nodes were obtained for the proposed ELGraph method and the VGraph method for the static arena containing the convex obstacles. The shortest path is obtained by the Dijkstra algorithm for the obtained feasible node pairs. The results have shown that the ELGraph offers reduced evaluation time than VGraph with no compromise on the feasible node pair selection because of its simplified linear structure that shows the ELGraph can be used for the application of path planning of the autonomous vehicles.

REFERENCES

[1] J. Delmerico, E. Mueggler, J. Nitsch, and D. Scaramuzza, "Active autonomous aerial exploration for ground robot path planning," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 664–671, 2017.

[2] K. Berntorp, "Path planning and integrated collision avoidance for autonomous vehicles," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 4023–4028.

[3] E. Dolicanin, I. Fetahovic, E. Tuba, R. Capor-Hrosik, and M. Tuba, "Unmanned combat aerial vehicle path planning by brain storm optimization algorithm," *Studies in Informatics and Control*, vol. 27, no. 1, pp. 15–24, 2018.

[4] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr, "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, 2019.

[5] M. Panda, B. Das, B. Subudhi, and B. B. Pati, "A comprehensive review of path planning algorithms for autonomous underwater vehicles," *International Journal of Automation and Computing*, pp. 1–32, 2020.

[6] J. Kim, S. Kim, and Y. Choo, "Stealth path planning for a high speed torpedo-shaped autonomous underwater vehicle to approach a target ship," *Cyber-Physical Systems*, vol. 4, no. 1, pp. 1–16, 2018.

[7] C. Sun, Y.-C. Liu, R. Dai, and D. Grymin, "Two approaches for path planning of unmanned aerial vehicles with avoidance zones," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 8, pp. 2076–2083, 2017.

[8] U. Z. A. Hamid, Y. Saito, H. Zamzuri, M. A. A. Rahman, and P. Raksincharoensak, "A review on threat assessment, path planning and path tracking strategies for collision avoidance systems of autonomous vehicles," *International Journal of Vehicle Autonomous Systems*, vol. 14, no. 2, pp. 134–169, 2018.

[9] C.-T. Yen and M.-F. Cheng, "A study of fuzzy control with ant colony algorithm used in mobile robot for shortest path planning and obstacle avoidance," *Microsystem Technologies*, vol. 24, no. 1, pp. 125–135, 2018.

[10] G. Qing, Z. Zheng, and X. Yue, "Path-planning of automated guided vehicle based on improved dijkstra algorithm," in *2017 29th Chinese control and decision conference (CCDC)*. IEEE, 2017, pp. 7138–7143.

[11] G.-z. Tan, H. He, and S. Aaron, "Global optimal path planning for mobile robot based on improved dijkstra algorithm and ant system algorithm," *Journal of Central South University of Technology*, vol. 13, no. 1, pp. 80–86, 2006.

[12] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 16, 2017.

[13] S. Wen, Y. Zhao, X. Yuan, Z. Wang, D. Zhang, and L. Manfredi, "Path planning for active slam based on deep reinforcement learning under unknown environments," *Intelligent Service Robotics*, pp. 1–10, 2020.

[14] S. Ortiz, W. Yu, and X. Li, "Autonomous navigation in unknown environments using robust slam," in *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1. IEEE, 2019, pp. 5590–5595.

[15] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on robotics and automation*, vol. 16, no. 5, pp. 615–620, 2000.

[16] L. Li, T. Ye, M. Tan, and X.-j. Chen, "Present state and future development of mobile robot technology research," *Robot*, vol. 24, no. 5, pp. 475–480, 2002.

[17] X.-c. Wang, R.-b. Zhang, and G.-c. Gu, "Potential grids based on path planning for robots," *Journal of Harbin Engineering University*, vol. 24, no. 2, pp. 170–174, 2003.

[18] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[19] X. Bai, W. Yan, M. Cao, and D. Xue, "Heterogeneous multi-vehicle task assignment in a time-invariant drift field with obstacles," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 307–312.

[20] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An improved ant colony algorithm for robot path planning," *Soft Computing*, vol. 21, no. 19, pp. 5829–5839, 2017.

[21] U. Challita, W. Saad, and C. Bettstetter, "Deep reinforcement learning for interference-aware path planning of cellular-connected uavs," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[22] B. M. Sathyaraj, L. C. Jain, A. Finn, and S. Drake, "Multiple uavs path planning algorithms: a comparative study," *Fuzzy Optimization and Decision Making*, vol. 7, no. 3, p. 257, 2008.

[23] P. Höfner and B. Möller, "Dijkstra, floyd and warshall meet kleene," *Formal Aspects of Computing*, vol. 24, no. 4-6, pp. 459–476, 2012.

[24] J. A. Janet, R. C. Luo, and M. G. Kay, "The essential visibility graph: An approach to global motion planning for autonomous mobile robots," in *Proceedings of 1995 IEEE international conference on robotics and automation*, vol. 2. IEEE, 1995, pp. 1958–1963.

[25] E. Masehian and M. Amin-Naseri, "A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning," *Journal of Robotic Systems*, vol. 21, no. 6, pp. 275–300, 2004.

[26] L. Parungao, F. Hein, and W. Lim, "Dijkstra algorithm based intelligent path planning with topological map and wireless communication," *ARPN Journal of Engineering and Applied Sciences*, vol. 13, no. 8, pp. 2753–2763, 2018.