Jurnal
Teknologi

Full Paper

COMPUTATIONALLY EFFICIENT PATH PLANNING ALGORITHM FOR AUTONOMOUS VEHICLE

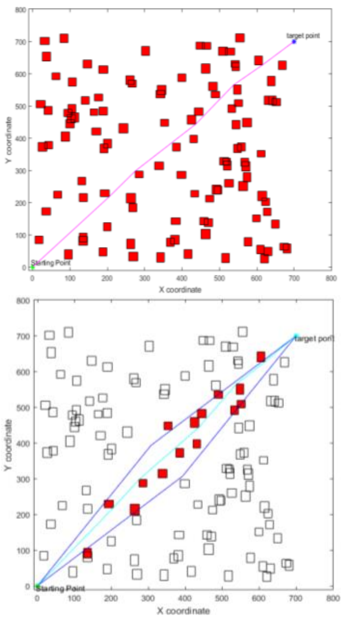
Sanjoy Kumar Debnath[a], Rosli Omar[a*], Nor Badariyah Abdul Latip[a], Susama Bagchi[a], Elia Nadira Sabudin[a], Abdul Rashid Omar Mumin[a], Abdul Majid Soomro[b], Marwan Nafea[c], Bashir Bala Muhammad[d], Ranesh Kumar Naha[e]

[a]Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400, Johor, Malaysia
[b]Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, 86400, Johor, Malaysia
[c]Department of Electrical and Electronic Engineering, University of Nottingham Malaysia, Semenyih, Selangor, Malaysia
[d]Faculty of Air Engineering, Air Force Institute of Technology, Kaduna Nigeria
[e]Discipline of ICT, University of Tasmania, Hobart, TAS7005 Australia

Article history
Received
29 January 2020
Received in revised form
13 October 2020
Accepted
4 November 2020
Published online
17 December 2020

*Corresponding author
roslio@uthm.edu.my

Graphical abstract



Abstract

This paper analyses an experimental path planning performance between the Iterative Equilateral Space Oriented Visibility Graph (IESOVG) and conventional Visibility Graph (VG) algorithms in terms of computation time and path length for an autonomous vehicle. IESOVG is a path planning algorithm that was proposed to overcome the limitations of VG which is slow in obstacle-rich environment. The performance assessment was done in several identical scenarios through simulation. The results showed that the proposed IESOVG algorithm was much faster in comparison to VG. In terms of path length, IESOVG was found to have almost similar performance with VG. It was also found that IESOVG was complete as it could find a collision-free path in all scenarios.

Keywords: Visibility graph, computionally efficient path planning, C-space, graph search algorithm, autonomous vehicle

Abstrak

Makalah ini membentangkan prestasi perancangan laluan secara eksperimen antara algoritma *Iterative Equilateral Space Oriented Visibility Graph* (IESOVG) dan *Visibility Graph* (VG) dari segi masa pengiraan dan panjang laluan untuk kenderaan autonomi. IESOVG adalah satu algoritma perancangan laluan yang dibangunkan bertujuan mengatasi kelemahan VG iaitu perlahan dalam senario yang mempunyai banyak halangan. Penilaian prestasi dilakukan dalam beberapa senario yang sama melalui simulasi. Hasil eksperimen menunjukkan bahawa IESOVG adalah sangat cepat berbanding VG. Dari segi panjang laluan, IESOVG mempunyai prestasi yang sama dengan VG. Didapati juga bahawa IESOVG adalah lengkap kerana ia dapat mencari laluan yang bebas perlanggaran dalam semua senario.

Kata kunci: Graf penyambungan, perancangan laluan berkesan secara pengiraan, ruang konfigurasi, algoritma pencarian graf, kenderaan autonomi

© 2021 Penerbit UTM Press. All rights reserved

1.0 INTRODUCTION

Path planning is necessary to complete a given mission for an autonomous vehicle [1]. Autonomous vehicle aids human operators in dangerous circumstances during natural disasters and hazardous areas [2], such as collapse of the World Trade Centre in 2001 [3], Hurricane Katrina in 2005, and the Tohoku tsunami and earthquake [4] in 2011. There are several types of path planning algorithms such as combinatorial, sampling based and bio-inspired method as shown in Figure 1. Each path planning algorithm has its own benefits and drawbacks in terms of the resulting path length, computation time and completeness [6-10]. Besides that, a path planning method must be able to find collision-free path in real time where environment changes.

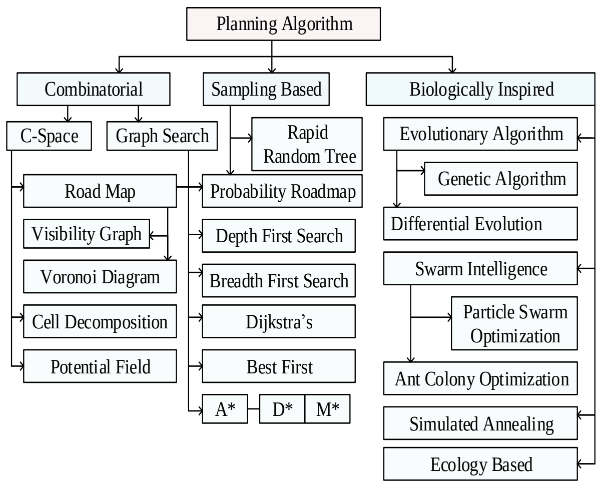


Figure 1 Planning algorithm based on classical taxonomy, adapted form [9]

Among the biologically inspired methods, genetic algorithm(GA) is currently used for an energy efficient path planning but it cannot guarantee to produce optimal paths because the local minima may occur in narrow environments. Moreover, GA is computationally expensive and practically not complete. Particle swarm optimization (PSO) has real time effect but it can easily fall into the local optima in many optimization problems [21]. Ant colony optimization (ACO) does a blind search [22] and thus, it is not optimistic and suitable for energy saving path planning. Simulated Annealing (SA) is also not capable of finding an optimal path.

In sampling based type, Rapidly Exploring Random Tree (RRT) does not always provide an optimal result. Probability Road Map (PRM) is expensive without any guarantee of finding the path.

Combinatorial type consists of mainly configuration space (C-space) and graph search algorithm [9]. Furthermore, this type has two main phases, where C-space works with pre-processing stage and graph search is associated with query stage [26-27].

The pre-processing phase specifies the environment of configuration spaces (C-spaces) while the query phase finds a collision-free path by using a graph search algorithm. In the pre-processing step, after specifying the C-Space, the nodes and the edges are built. In this phase, the starting point (S_p) and the target point (T_p) are defined. Then a map of the graph is generated [11]. C-space denotes the actual free space zone for a moving robot and also gives a guarantee that the vehicle or robot must not collide with any obstacle. However in dynamic and unknown environment, it is hard to find a collision-free path where information is uncertain [12]. Table 1 tabulates the pros and corn of different path planning methods.

Table 1 Properties of different path planning methods

Method	Optimality	Time	Complete
VG	√	×	√
VD	×	√	√
RG	×	√	×
ACD	×	√	√
ECD	×	×	√
PF	×	√	×
DFS	×	×	√
BFS	×	×	√
Dijkstra's	√	×	√
Best First	×	×	√
A*	×	×	√
RRT	×	×	√
PRM	×	×	×
GA	×	√	×
PSO	×	×	√
ACO	×	×	√
SA	×	√	√

Some other methods in C-Space such as Potential Field (PF) could not find the goal because of the local minima issue [25]. The Cell Decomposition (CD) approach also does not provide an acceptable performance in a dynamic state and in real-time. For a path planning using CD, although regular grid (RG) is easy to apply, but the planner may not be complete if the cell is too big. Adaptive cell decomposition (ACD) needs to adjust with the situation as required. Exact Cell Decomposition(ECD) is complete but not suitable for outdoor environment and hence, it is not optimal for path length [6, 16, 28]. VG is more energy efficient than Voronoi diagram (VD) in combinatorial method under roadmap technique [9]. The disadvantage of VD is that the generated path is not optimal [18-20].

In query phase, a path will be planned after C-space representation. A graph search algorithm is applied to find a shortest collision-free path in the C-space. Under graph search method, Depth Fist Search (DFS) is good to pick up a solution among many possibilities. DFS is good because a solution can be found without considering all nodes [9]. Breadth-first Search (BFS) is suitable for the limited available solutions that uses a comparatively small number of steps. It is able to find a path if one exists and does not get trapped in dead ends. BFS algorithm does not assure to discover the shortest path because it bypasses some branches in the search tree. A-star (A*) is sub optimal because it needs to be executed a number of times for each target node to get them all. A* is complete because it always finds a path if one exists. Dijkstra's algorithm is a systematic search algorithm and it gives the shortest path between two nodes. In optimal cases, when there is no prior knowledge of the graph, it cannot estimate the distance between each node and the target [9], [20].

The advantages of VG is that it is able to solve path planning problem by finding the shortest distance without the possibilities of the local minima occurrance [13-15]. It also satisfies two criteria of optimal path planning, i.e. complete path planning and generation of shortest path [16]. Besides that, VG is collision-free [24] and easy to be implemented due to the simplification of equation [5, 13].

A conventional VG path planning is demonstrated using MATLAB as shown in Figure 2. When the number of obstacles in C-space increases, the computation time is also increased. This is due to the fact that VG finds a path by connecting all the mutually visible edges and therefore, as the number of obstacles rises, the computation time of VG also increases.

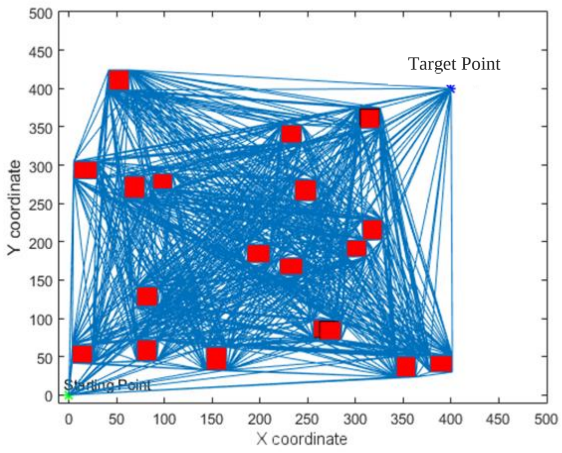


Figure 2 VG calculates all obstacles inside the C-space

To overcome this problem, the Iterative Equilateral Space Oriented Visibility Graph (IESOVG) was proposed. IESOVG was developed based on the conventional VG algorithm. It creates an equilateral space depending on the density of obstacles residing in the space. Then, it finds a path using Dijkstra's algorithm by ensuring that the produced path is always (1) within the equilateral space, (2) collision free, (3) complete and (4) minimum in computation time compared to the the conventional VG.

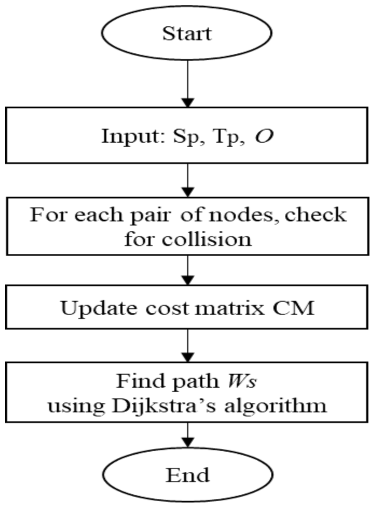
2.0 METHODOLOGY

2.1 Visibility Graph (VG)

This section describes how VG and Dijkstra's algorithm work for path planning. Figures 3(a) and 3(b) show the algorithm and the flowchart of the conventional VG, respectively. In the algorithm, *O* represents obstacles while *Ws* is the resulting path in the form of waypoints.

Algorithm:	**VG**
Input:	*O*, S_p, T_p
Output:	*W*
1	**For** all nodes *NL* of *O*
2	Create an edge *e* for each pair of nodes
3	Check the visibility of the edge
4	**If** ~= collision
5	Add *e* into cost matrix *CM*
6	**End**
7	Repeat Step 2
8	**End**
9	Find *W* from *CM* using Dijkstra's algorithm

(a)


Start
Input: Sp, Tp, O
For each pair of nodes, check for collision
Update cost matrix CM
Find path Ws using Dijkstra's algorithm
End

(b)

Figure 3 (a) The algorithm and (b) the flowchart of the VG

The inputs of the VG are the starting point S_p, the target point T_p and the obstacles *O* in a configuration space. All of the edges of obstacles and the edges connecting the nodes including S_p and T_p are available in the C-space and it is checked whether collisions may happen. If there is a collision between two nodes, the information of the nodes is updated in the cost matrix, CM. Then Dijkstra's algorithm is applied to find a collision free path. An illustration on path planning is shown in Figure 4 by using VG.

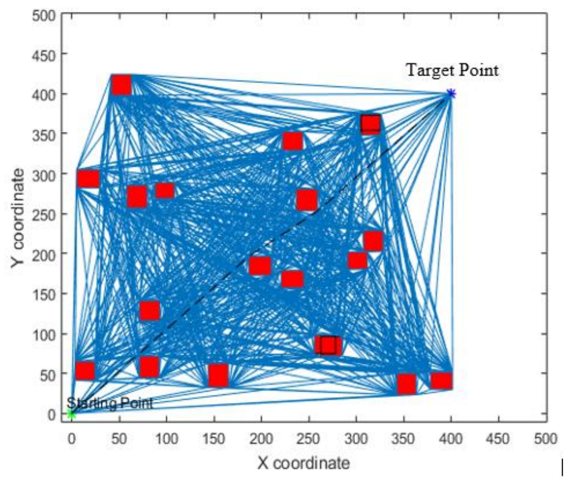


Figure 4 Path planning using VG with graph search algorithm

Hence, the relationship between the number of obstacles and the computation time of VG is not proportionally linear because VG considers all the obstacles inside the C-space. To reduce the computation time of VG, equilateral spaces-oriented visibility graph (ESOVG) is proposed.

2.2 Equilateral Space Oriented Visibility Graph (ESOVG)

ESOVG creates an equilateral space in the C-space based on the Sp, Tp, input angle ρ and obstacles *O*. Figure 5 illustrates the equilateral space represented by the darker region.

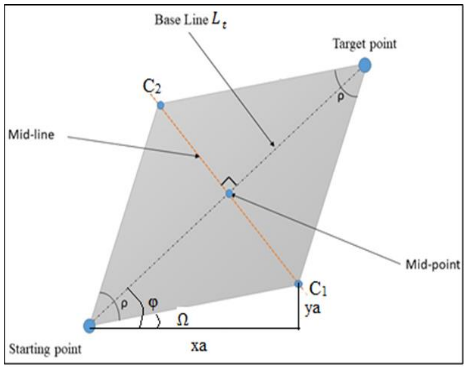


Figure 5 The equilateral space

The space is enclosed by a couple of pairs of imaginary lines that connect the S_p and T_p to both C_1 and C_2 as depicted in the Figure 5. The first pair of lines emerges from S_p towards the mid-line, separated by an input angle of ρ degrees while the second pair emerges from T_p towards the midline with identical ρ. Both pairs of lines intersect at the points denoted by C_1 and C_2. Here Ω is to show the orientation of reference angle. Also, to show *the relative orientation, xa and ya* lines are drawn. The ESOVG algorithm is described in Figure 6.

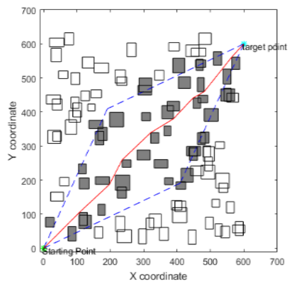
```
Algorithm:   ESOVG
Input:       O, S_p, T_p
Output:      W_S
1            Create a base line connecting S_p and T_p
2            **Identify** a mid-point between S_p and T_p
3            Create a mid-line passing through the mid-
             point and perpendicular to the base line
4            From S_p and T_p, create a pair of imaginary
             lines with an opening angle of ρ towards
             the midline.
5            Create an equilateral space S from the
             enclosed area by the four imaginary lines
             drawn in step 4
6            Find obstacles O_s located within S
7            **For** all nodes of O_s
8              Create an edge e_S for each pair of nodes
9              Check the visibility of the edge
10             **if** no collision
11             Add e_S into cost matrix CM_S
12             Repeat Step 8
13             **end**
             **end**
14           Find W_S from CM_S using Dijkstra's algorithm
```

Figure 6 ESOVG algorithm

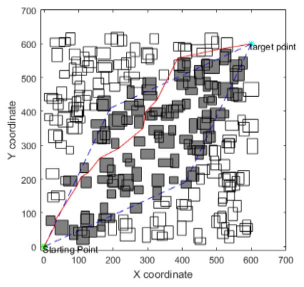
Through ESOVG algorithm, the size of the search space is reduced from the entire workspace into the equilateral space. The input angle ρ determines the size of the space i.e., ρ is proportional to the size of the space where a greater ρ produces a larger space and vice versa. Therefore, ESOVG reduces the obstacles number in such a way that only the obstacles residing within and overlapping with the equilateral space are considered for path planning.

As a result, ESOVG uses a reduced number of obstacles and hence, it has a lower computation time. The application of ESOVG for path planning is illustrated in Figure 7(a).

ESOVG algorithm selects obstacles which are shown in dark to be used for path planning. Note that the area enclosed by the dotted lines represents the equilateral space. Nonetheless, ESOVG has a limitation as shown in Figure 7(b) where the resulting path may not be collision-free. Therefore, IESOVG is developed to overcome that limitation.



(a)



(b)

Figure 7 (a) ESOVG identified obstacles within the equilateral space and (b) failed to get collision free path within an equilateral space

2.3 Iterative Equilateral Space Oriented Visibility Graph (IESOVG)

An improvement in ESOVG is necessary because, sometimes, a few segments of the path are outside the space which may hit obstacles. To address this problem, an improvement of ESOVG was proposed where the resulting path is checked iteratively.

In the proposed algorithm, the planning stops when the path is within the space. Otherwise, the equilateral space will keep on expanding by a certain amount of area through the increment in the input angle ρ. Subsequently, the planning will continue until the resulting collision-free path is completely within the space.

The input angle ρ in IESOVG is used iteratively to expand the area of the equilateral space. Figure 8 shows the IESOVG and Figure 9 illustrates its flowchart. Finally, Figure 10 shows the details and complete flowchart of IESOVG in which the ESOVG is embedded.

```
Algorithm:  IESOVG
Input:      $S_P$, $T_P$, $O$, $\rho = 14°$
Output:     $W_{IE}$
1           Call ESOVG
2             While Ws is outside S
3             Increase $\rho$ by 8°
4             Call ESOVG
5                 If $W_S$ is inside S
6                     Find obstacles $O_s$ located within $S$
7                     Break
8                     $W_{IE}=W_S$
9                 End
10          End
```

Figure 8 Iterative-ESOVG (IESOVG) algorithm

In order to determine the nominal value of ρ, a simulation was performed. Figure 11 depicts the computation time versus number of obstacles and the resulting computation time is listed in Table 2. It is found that the smallest average computation time was obtained when the nominal value of ρ=14°.

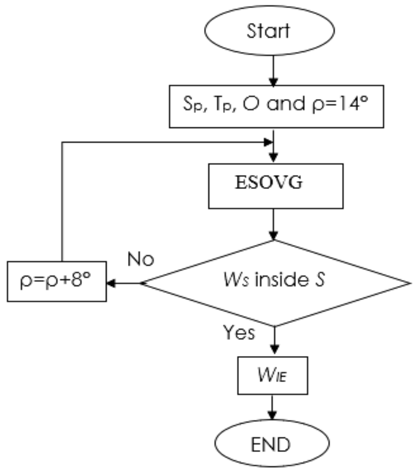


Figure 9 IESOVG flowchart

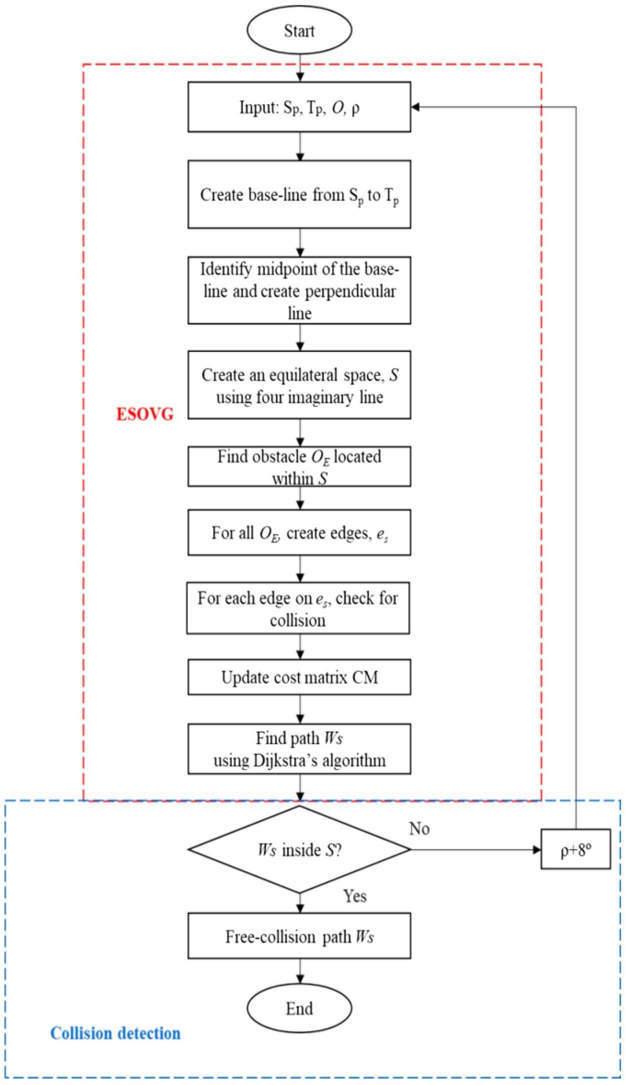

Start
Input: Sp, Tp, O, ρ
Create base-line from Sp to Tp
Identify midpoint of the base-line and create perpendicular line
Create an equilateral space, S using four imaginary line
ESOVG
Find obstacle OE located within S
For all OE, create edges, es
For each edge on es, check for collision
Update cost matrix CM
Find path Ws using Dijkstra's algorithm
Ws inside S?
No
ρ+8°
Yes
Free-collision path Ws
End
Collision detection

Figure 10 IESOVG in which the ESOVG is embedded

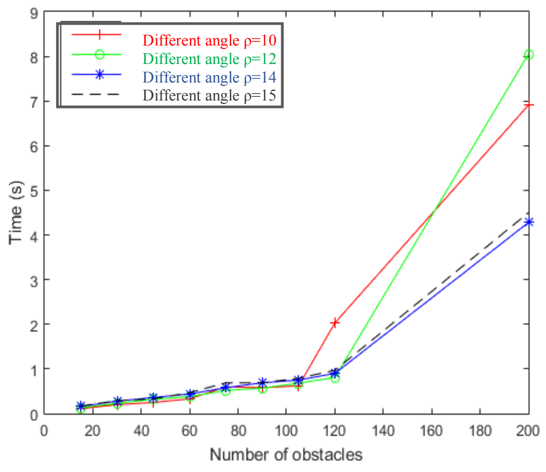


Figure11 Simulation results of ESOVG with different nominal value of ρ

Table 2 Computation times with different angle ρ and number of obstacles

Obstacles no.	ρ=10°	ρ=12°	ρ=14°	ρ=15°
15	0.11s	0.13s	0.16s	0.17s
30	0.20s	0.23s	0.28s	0.30s
45	0.25s	0.32s	0.35s	0.33s
60	0.34s	0.38s	0.44s	0.47s
75	0.60s	0.52s	0.58s	0.69s
90	0.58s	0.57s	0.69s	0.69s
105	0.62s	0.68s	0.75s	0.79s
120	2.03s	0.81s	0.90s	0.97s
200	6.92s	8.05s	4.29s	4.50s
Average	**1.29s**	**1.30s**	**0.94s**	**0.99s**

In order to determine the increment angle of ρ, a further analysis was done through a simulation. The simulation results are listed in Table 3. In the simulation, different incremental angle was applied, i.e. 6°, 8 ° and 10 ° where the numbers of obstacles were set to be higher than 100 to force the planned path to be located outside the equilateral space. Based on the previous findings, ρ was initialized to 14° throughout the simulations.

Table 3 Findings of different incremental ρ

Obstacles no.	ρ+6	ρ+8	ρ+10
120	0.85s	0.66s	0.87s
150	6.36s	2.34s	7.99s
165	10.65s	2.72s	8.86s
200	14.12s	3.38s	12.10s
Average	**8.00s**	**2.26s**	**7.46s**

It is evident from Table 3 that the increment of ρ by 8° resulted in a lower computation time compared with that of the other two. To get a clearer picture on this, Figure 12 depicts the simulation result for different incremental angles.

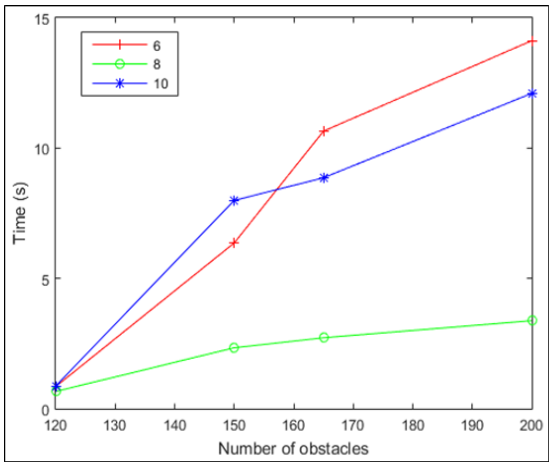


Figure12 ESOVG with different incremental values of ρ

3.0 RESULTS AND DISCUSSION

3.1 Performance of VG

As mentioned earlier, VG is slow especially in an obstacle-rich environment. We evaluated the performance of VG through a simulation in random scenarios with different number of obstacles and the outcomes are recorded in Table 4.

Table 4 Computation time of VG with different number of obstacles

Obstacles no.	Computation time (s)
15	0.88
30	1.74
45	3.92
60	7.25
75	11.08
90	15.86
105	20.41
120	27.16
135	33.63
150	40.33

From Table 4, it is observable that if the numbers of obstacles are increased then the computation time also increases drastically. For instance, when the number of obstacles was 15, the computation time was 0.88 s. With 150 obstacles, the recorded computation time was 40.33 s.

It clearly shows that the increase in the computation time is not linearly proportional with the number of obstacles, but almost exponential as visualized in Figure 13.

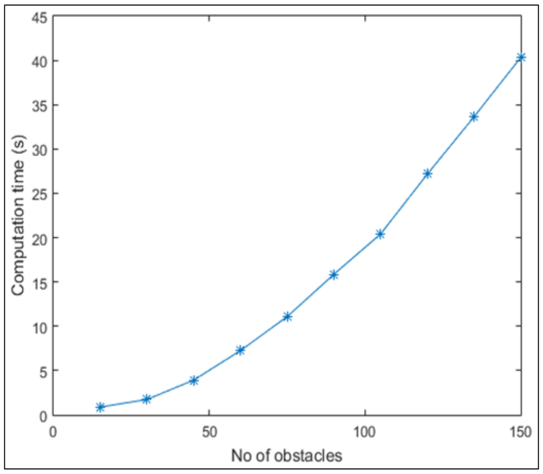


Figure 13 VG Computional time Increases exponentially as the number of obstacles increases

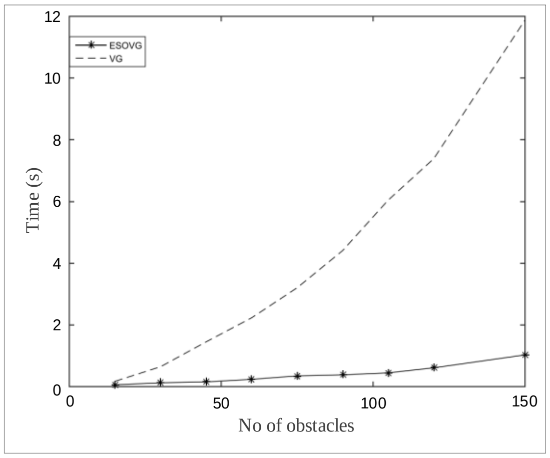


Figure 14 computation times of VG and ESOVG

3.2 Performance of ESOVG

ESOVG generates equilateral space depending on the input angle ρ that limits the area in C-Space. Whereas in VG, all the area in C-Space was calculated as elaborated in Table 5.

Table 5 Improvement rate of ESOVG

	VG		ESOVG		
Obstacles no.	**Computation time** (s)	**Path length** (unit)	**Computation time** (s)	**Path length** (unit)	**Improvement Rate** (%)
15	0.1861	520.1342	0.0678	520.1342	63.5
30	0.6522	520.1342	0.1310	520.1342	79.9
45	1.4569	627.4493	0.1609	627.4493	89.0
60	2.2356	631.0025	0.2439	631.0025	89.0
75	3.2086	645.6740	0.3541	645.6740	89.0
90	4.4142	645.6751	0.3880	645.6751	91.2
105	6.0464	645.6751	0.4545	645.6751	92.5
120	7.3872	750.2257	0.6156	750.2257	91.7
150	11.8672	910.8724	1.0334	910.8724	91.3

When the number of obstacles was 15, the computation time for finding a path by conventional VG was 0.1861 s, and by ESOVG was 0.0678 s. When the number of obstacles was 150, the computation time by conventional VG and ESOVG were 11.8672 s and 1.0334 s, respectively. The improvement recorded by ESOVG was 63.5 %-91.3 %.

Figure 14 clearly shows the trends of computation times for both the conventional VG and ESOVG which indicate that when the number of obstacles escalates, the computation time of conventional VG increases exponentially, whereas it rises almost linearly for ESOVG.

Table 5 reveals that both VG and ESOVG produce paths of identical lengths. This proves that while ESOVG reduces the computation time, it maintains the optimality of the resulting path in terms of length.

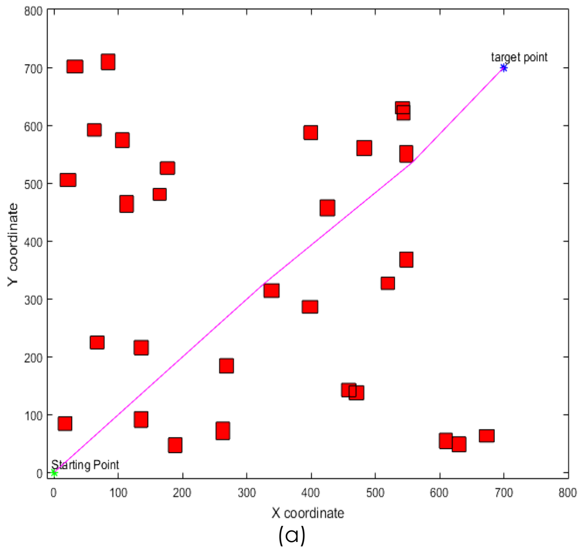
3.3 Performance of IESOVG

Simulation results for both VG and IESOVG in an identical random scenario with 30, 75 and 110 obstacles are shown in Figure15 (a)-(f). Note that, the red colored obstacles were used for path calculation. Figure 15(a) depicts the simulation for VG where 30 obstacles were used whereas in the same scenario shown in Figure 15(b), IESOVG used only 4 obstacles to calculate a collision-free path. When the numbers of obstacles were increased to 75 in 15(c), it was only 9 obstacles used by IESOVG as shown in Figure 15(d). In Figure 15(e), 15 obstacles were used by IESOVG. Table 6 shows the comparison of the computation time and path length of the conventional VG and IESOVG. It reveals that IESOVG is relatively faster compared to the VG in terms of computation time.

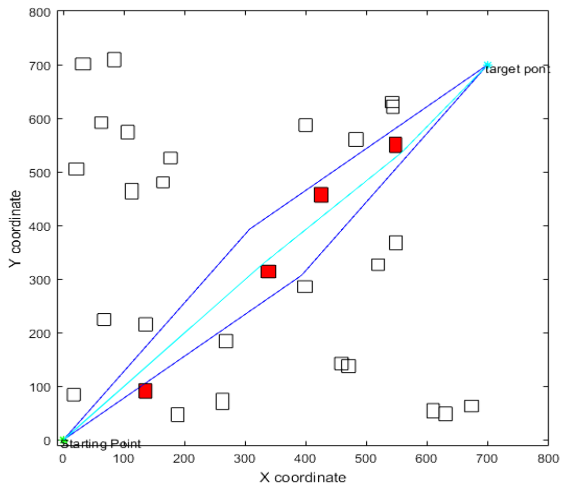
From the table, in a scenario with 30 obstacles, the computation time by conventional VG was 1.74 s, and 0.16 s by IESOVG. IESOVG improved the computation time up to 90.8 % and the path length for both IESOVG and conventional VG were identical at 990.86 m. As the number of the obstacle increased to 75, the computation time of conventional VG was 11.08 s while the IESOVG's was 0.3 s, which improved the VG computation time by 97.9 %. In this case, the path length for both IESOVG and VG were again identical, i.e. 992.08 m. When the number of the obstacle was further increased to 135, the improvement rate recorded was the highest which was 97.77 %, where the computation times for conventional VG and IESOVG were 33.63 s and 0.75 s, respectively. Simulation results with 135 and 150 obstacles can be observed in Table 6.

Please note that the scenarios used in both performance comparisons i.e., ESOVG Vs VG and

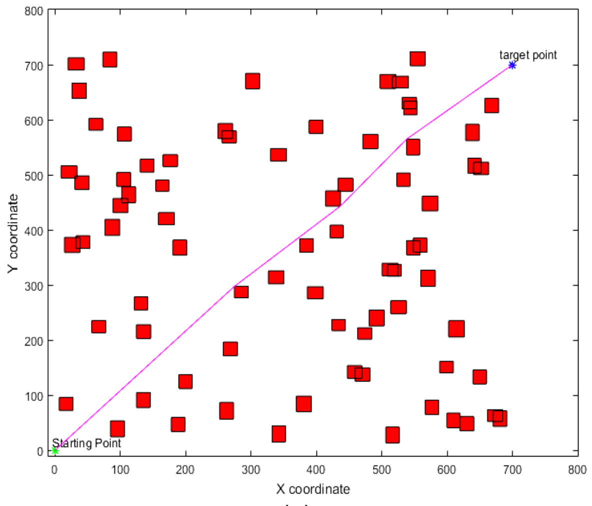
IESOVG Vs VG are different hence they produced different results.



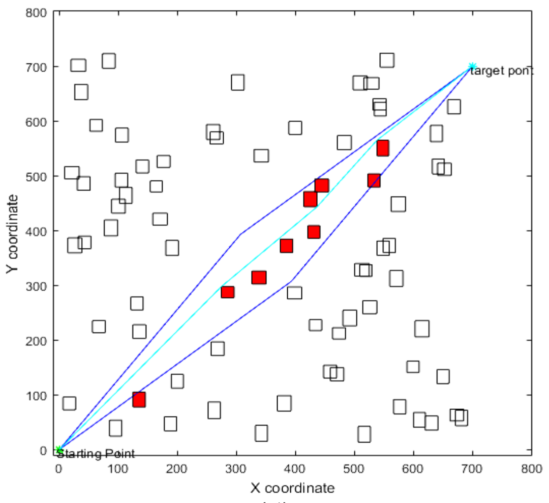
(a)



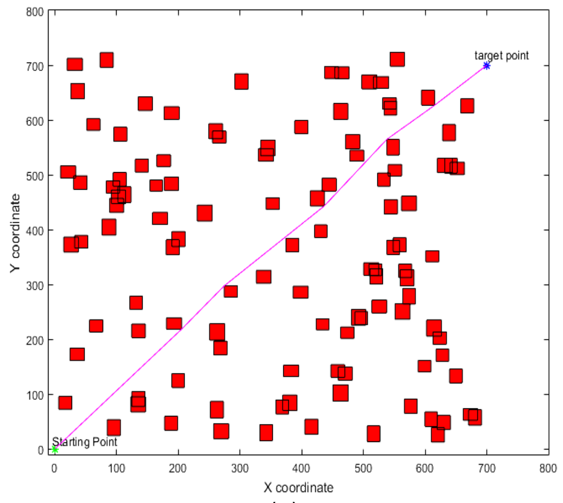
(b)



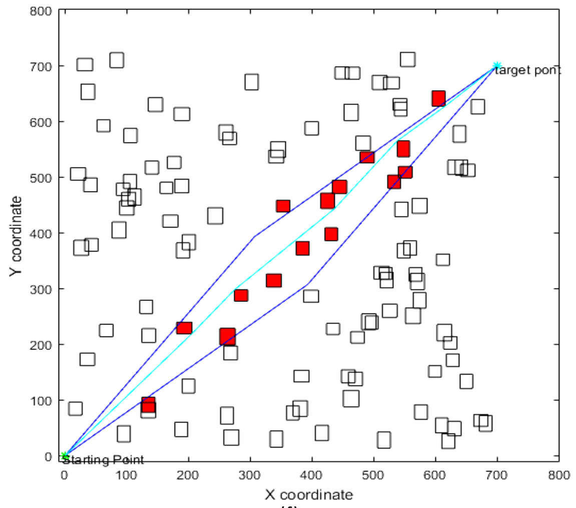
(c)



(d)



(e)



(f)

Figure 15 IESOVG Simulation at different scenarios

Table 6 Improvement rate of IESOVG

	VG		IESOVG		IMPROVEMENT	
Obstacles no.	Computation time(s)	Path length (m)	Computation time(s)	Path length(m)	Computation time efficiency (%)	Path length difference (%)
30	1.74	990.86	0.16	990.86	90.80	0.00
45	3.92	992.08	0.21	992.08	94.64	0.00
60	7.25	992.08	0.25	992.08	96.55	0.00
75	11.08	992.08	0.30	992.08	97.29	0.00
90	15.86	992.12	0.39	992.12	97.54	0.00
105	20.41	992.17	0.50	992.17	97.55	0.00
120	27.16	992.17	0.66	992.17	97.57	0.00
135	33.63	992.49	0.75	992.51	97.77	2×10^{-3}
150	40.33	994.81	2.34	994.83	94.20	2×10^{-3}

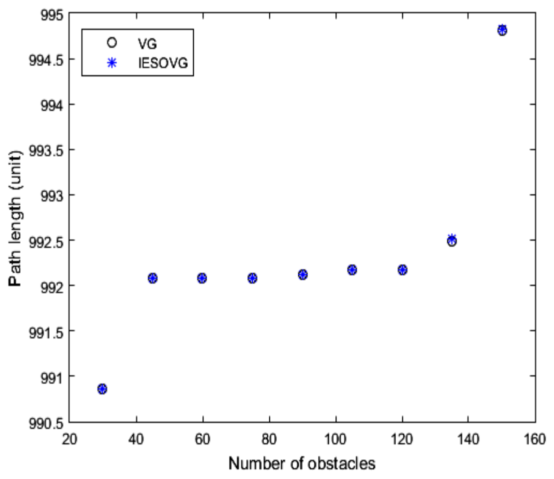


Figure16 Path length of VG and IESOVG

The path lengths obtained by VG and IESOVG are tabulated in Table 6 and plotted in Figure 16. It is clear that the path generated by the IESOVG was almost identical to the conventional VG. The largest path length difference between VG and IESOVG was 0.02 %, which is insignificant. Therefore, IESOVG is considered as an optimal path planner.

Figure17 shows the improvement made by the IESOVG for the computation time. Here, the graph indicates that the IESOVG managed to produce a path more than 90 % faster than that of VG throughout different numbers of obstacles. It also shows that the computation time was increased exponentially with the increment of the obstacles' numbers for VG, while the computation time of IESOVG was increased almost linearly.

In Table 5 and in Table 6, the path length and the random distribution number of obstacles are not same. Thus, computation time and path length are different in both scenarios. In both setups, the developed algorithm worked successfully.

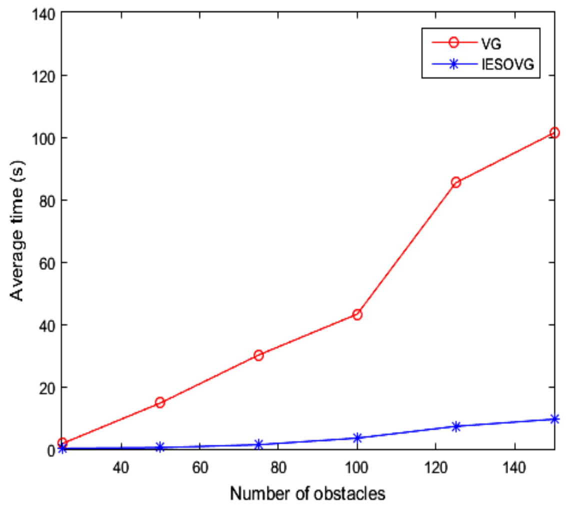
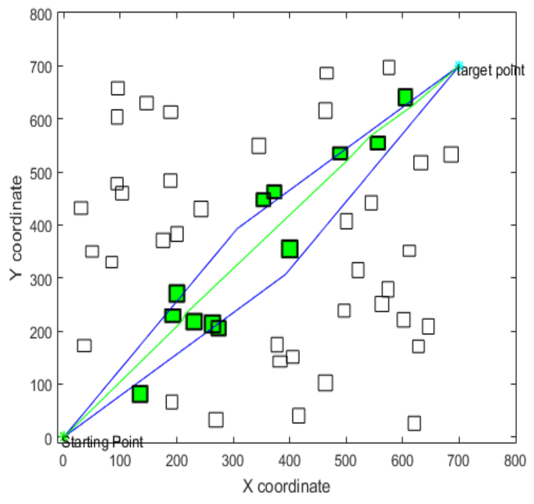
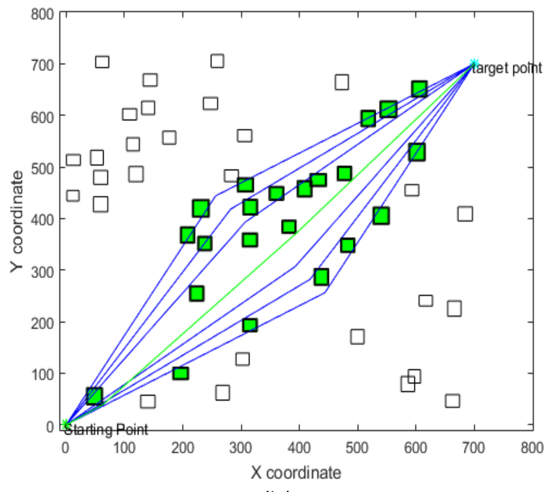


Figure17 Average Computational time comparison between VG and IESOVG

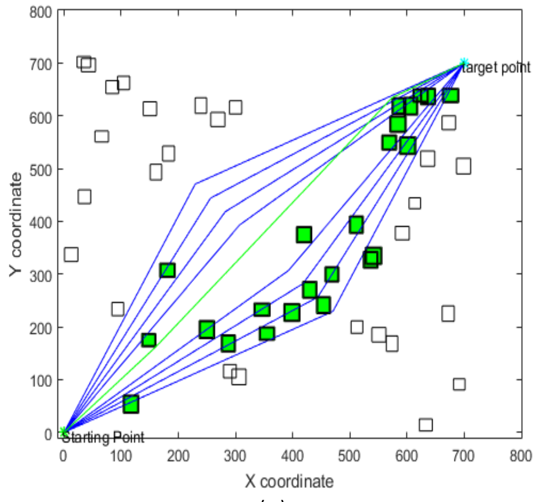
Finally, we will discuss about the completeness of IESOVG. Completeness is an important criterion of any path planning algorithm. An algorithm that holds this criterion has the capability to find a path in any scenario and it also reports if no path exists there. Simulation to test the completeness of IESOVG algorithm was run in 100 random scenarios with 50 obstacles in the C-spaces. From the findings, it can be concluded that IESOVG is a complete algorithm since it is able to find a path in every scenario. Figure 18(a) – 18(d) illustrate some of the simulation results of finding paths using IESOVG in random scenarios.



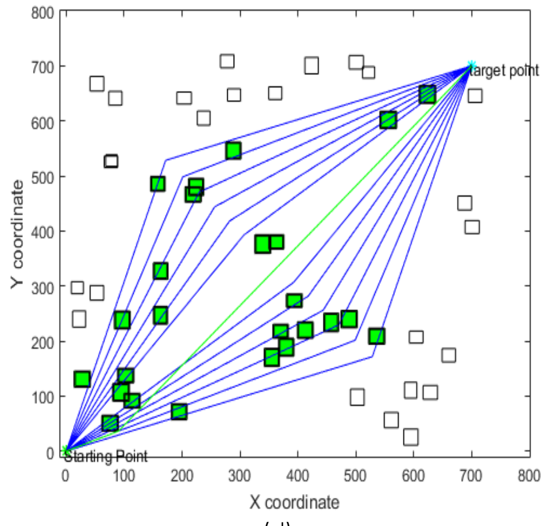
(a)



(b)



(c)



(d)

Figure 18 IESOVG found complete paths

4.0 CONCLUSION

The performance of IESOVG and conventional VG in terms of path length and computation time was compared and analyzed in this work. The obtained results prove that the IESOVG is relatively faster than the conventional VG and it is also capable to produce optimal paths. IESOVG connects a line between the starting point and the target point of shortest path lengths to form an equilateral space and hence, it is complete. The angle of equilateral space was initially set to ρ=14°. When a path is not found within the equilateral spaces, the angle is expanded by 8°. IESOVG manages to reduce the computation time than the conventional VG while maintaining the shortest distance. This is because the IESOVG only considers few numbers of obstacles in a C-space covered within an enclosed area and it results in less complex visibility graphs. Regarding computation time, the IESOVG provided improved result of 97.77 % in comparison with VG. In average, the IESOVG performed above 90 % better computation time while path length was kept almost the same.

Acknowledgement

This research is supported by grant TIER-1 VOT-H131. The authors fully acknowledged Universiti Tun Hussein Onn Malaysia (UTHM).

References

[1] Mueggler, E., Faessler, M., Fontana, F., & Scaramuzza, D. 2014. Aerial-guided Navigation of a Ground Robot Among Movable Obstacles. *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics*. 1-8. DOI: 10.1109/SSRR.2014.7017662.

[2] Nikolic, J., Burri, M., Rehder, J., Leutenegger, S., Huerzeler, C., & Siegwart, R. 2013. A UAV System for Inspection of Industrial Facilities. *Aerospace Conference*. 2013 IEEE. 1-8. DOI: 10.1109/AERO.2013.6496959.

[3] Murphy, R. R. 2004. Trial by Fire [Rescue Robots]. *IEEE Robotics & Automation Magazine*. 11(3): 50-61. DOI: 10.1109/MRA.2004.1337826.

[4] Murphy, R. R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., & Erkmen, A. M. 2008. *Search and Rescue Robotics*. In B. Siciliano and K. Oussama (Editors). Handbook of Robotics. Springer Verlag. 1151-1173. DOI: 10.1007/978-3-540-30301-5_51.

[5] R. Latip, N. B. A., Omar, R., & Debnath, S. K. 2017. Optimal Path Planning using Equilateral Spaces Oriented Visibility Graph Method. *International Journal of Electrical & Computer Engineering*. 7(6): 2088-8708. DOI: 10.11591/ijece.v7i6.pp3046-3051.

[6] Giesbrecht, J. and Defence R&D Canada. 2004. Path Planning for Unmanned Ground Vehicles. *Technical Memorandum DRDC Suffield*-TM 2004-272.

[7] LaValle, S. M. 2006. *Planning Algorithms*. Cambridge University, Press. http://planning.cs.uiuc.edu/.

[8] Hu, L., Liu, Y., Peng, C., Tang, W., Tang, R., & Tiwari, A. 2018. Minimising the Energy Consumption of Tool Change and Tool Path of Machining by Sequencing the Features. *Energy*. 147: 390-402.

https://doi.org/10.1016/j.energy.2018.01.046.

[9] Debnath, S. K., Omar, R. and Latip, N. B. A. 2019. A Review on Energy Efficient Path Planning Algorithms for Unmanned Air Vehicles. *Computational Science and Technology*. Springer, Singapore. 523-532. DOI.org/10.1007/978-981-13-2622-6_56.

[10] Moore, E. F. 1957. The Shortest Path through a Maze. *Proceedings of an International Symposium on the Theory of Switching*. Cambridge: Harvard University Press. 285-292.

[11] Švestka, P., & Overmars, M. H. 1998. Probabilistic Path Planning. *Int. Robot Motion Planning and Control*. Springer. 255-304. DOI: 10.1007/bfb0036074.

[12] Li, G., Yamashita, A., Asama, H., & Tamura, Y. 2012. An Efficient Improved Artificial Potential Field-based Regression Search Method for Robot Path Planning. *Int. Conf. on Mechatronics and Automation*, *IEEE*. 1227-1232. DOI: 10.1109/ICMA.2012.6283526.

[13] Ma, Y., Zheng, G., & Perruquetti, W. 2013. Cooperative Path Planning for Mobile Robots Based on Visibility Graph. *Int. Control Conference (CCC), IEEE*. 4915-4920.

[14] Debnath, S. K., Omar, R., & Latip, N. A. 2019. Comparison of Different Configuration Space Representations for Path Planning Under Combinatorial Method. *Indonesian Journal of Electrical Engineering and Computer Science* 1(1): 401-408. DOI: 10.11591/ijeecs.v10.i1.pp401-408.

[15] Tran, N., Nguyen, D. T., Vu, D. L., & Truong, N. V. 2013. Global Path Planning for Autonomous Robots using Modified Visibility-graph. *Int Control, Automation, and Information Sciences (ICCAIS), IEEE*. 317-321. DOI: 10.1109/ICCAIS.2013.6720575.

[16] Debnath, S. K., Omar, R., Ibrahim, B. S. S. K., Bagchi, S., Nadira, E., Amin, F., & Muhammad, B. B. 2020. Flight Cost Calculation for Unmanned Air Vehicle Based on Path Length and Heading Angle Change. *International Journal of Power Electronics and Drive Systems*. 11(1): 382-389. DOI: 10.11591/ijpeds. v11.i1. pp382-389.

[17] Mac, T. T., Copot, C., Tran, D. T., De Keyser, R. 2016. Heuristic Approaches in Robot Path Planning: A Survey. *Robotics and Autonomous Systems*. 1(86): 13-28. DOI.org/10.1016/j.robot.2016.08.001.

[18] Marbate, P., & Jaini, P. 2013. Role of Voronoi Diagram Approach in Path Planning. *International Journal of Engineering Science and Technology*. 5(3): 527.

[19] Niu, H., Lu, Y., Savvaris, A., & Tsourdos, A. 2018. An Energy-efficient Path Planning Algorithm for Unmanned Surface Vehicles. *Ocean Engineering*. 161: 308-321. DOI: org/10.1016/j.oceaneng.2018.01.025.

[20] Zear, A., & Ranga, V. 2020. Path Planning of Unmanned Aerial Vehicles: Current State and Future Challenges. *First International Conference on Sustainable Technologies for Computational Intelligence*. Springer, Singapore. 409-419. DOI: org/10.1007/978-981-15-0029-9.

[21] Patley, A., Bhatt, A., Maity, A., Das, K., & Ranjan Kumar, S. 2019. Modified Particle Swarm Optimization based Path Planning for Multi-Uav Formation. *AIAA Scitech 2019 Forum*. 1167.

[22] Mirjalili, S., Dong, J. S., & Lewis, A. 2019. Ant Colony Optimizer: Theory, Literature Review, and Application in AUV Path Planning. *Nature-Inspired Optimizers*. Springer, Cham. 7-21. DOI: org/10.2514/6.2019-1167.

[23] Debnath, S. K., Omar, R., Latip, N. B. A., Shelyna, S., Nadira, E., Melor, C. K. N. C. K., Chakraborty, T. K. and Natarajan, E. 2019. A Review on Graph Search Algorithms for Optimal Energy Efficient Path Planning for an Unmanned Air Vehicle. *Indonesian Journal of Electrical Engineering and Computer Science*. 15(2): 743-749. DOI: 10.11591/ijeecs. v15.i2. pp743-749.

[24] Lee, H. Y., Shin, H., & Chae, J. 2018. Path Planning for Mobile Agents Using a Genetic Algorithm with a Direction Guided Factor. *Electronics*. 7(10): 212. DOI: org/10.3390/electronics7100212.

[25] Sabudin, E. N., Omar, R., Joret, A., Ponniran, A., Sulong, M. S., Kadir, H. A., & Debnath, S. K. 2020. Improved Potential Field Method for Robot Path Planning with Path Pruning. *Proceedings of the 11th National Technical Seminar on Unmanned System Technology 2019*. 113-127. DOI: https://doi.org/10.1007/978-981-15-5281-6_9

[26] Debnath, S. K., Omar, R., Bagchi, S., Nafea, M., Naha, R. K., & Sabudin, E. N. 2020. Energy Efficient Elliptical Concave Visibility Graph Algorithm for Unmanned Aerial Vehicle in an Obstacle-rich Environment. *2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*. 129-134. DOI: 10.1109/I2CACIS49202.2020.9140112.

[27] Sharma, K., & Doriya, R. 2019. Reducing Traverse Space in Path Planning using Snake Model for Robots. *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, *IEEE*. 1-4. DOI: 10.1109/CCCS.2019.8888083.

[28] Debnath, S. K., Omar, R., Bagchi, S., Sabudin, E. N., Kandar, M. H. A. S., Foysol, K., & Chakraborty, T. K. 2020. *Different Cell Decomposition Path Planning Methods for* Unmanned Air Vehicles-A Review. *Proceedings of the 11th National Technical Seminar on Unmanned System Technology*. 99-111. DOI: 10.1007/978-981-15-5281-6_8.