COBEM 2021 26th International Congress of Mechanical Engineering
November 22-26, 2021, Virtual Congress, Brazil

COB-2021-0075
COMPARISON OF PATH PLANNING TECHNIQUES FOR AUTONOMOUS MOBILE ROBOTS

Arnaldo Roberto Rady Peron
Marco Antonio Meggiolaro
Pontifical Catholic University of Rio de Janeiro. Rua Marquês de São Vicente, 225, Gávea. 22451-900 – Rio de Janeiro, RJ – Brasil
arnaldo.peron@gmail.com
meggi@puc-rio.br

Abstract. *This paper presents a comparison study of five state-of-the-art algorithms used on autonomous mobile robots path planning - Dijkstra's Algorithm (DA), Dynamic Programming (DP), Probabilistic Roadmaps (PRM), Rapidly-exploring Random Trees (RRT) and Hybrid A* - regarding processing time, number of steps and path length. The study was carried out through simulations in MATLAB using functions that were already available, on three different maps, each one aiming on a distinct aspect of the path planners: a labyrinth, simulating a complex environment; a labyrinth with narrow passages, increasing the difficulty to navigate; and a zigzag circuit with 180° curves. The results computed during simulations are presented on table and charts with the values of each parameter evaluated for all the three maps.*

Keywords: *Path Planning, Mobile Robotics, Autonomous Robots, Simulation.*

1. INTRODUCTION

With the crescent popularity of mobile robotics, it increases the demand for robots able to navigate autonomously in an environment. Aiming in attending the opportunities, more companies, professionals and organizations are directing their attention to this field of study.

For autonomous mobile robots path planning is a matter of greater importance, which rises the problem of choosing an adequate path planner. This paper presents a comparative study of five global state-of-the-art path planners that can be easily found read to use in multiple platforms: Dijkstra's Algorithm (DA), Dynamic Programming (DP), Probabilistic Roadmaps (PRM), Rapidly-exploring Random Trees (RRT) and Hybrid A*. The objective is to provide readers who are starting out in robotics with information that will help them to choose a planner that meets their needs and is simple to implement. Furthermore, these planners are among the most successful algorithms for robotics (Steffi *et al.*, 2021).

Path planners are classified according to their approach as classical approach, which is the case of the five planners above, and heuristic or artificial intelligence approach (Mohd and Mohantab, 2018). According to Zhang *et al.* (2018), Engineering Village database shows that the amount of papers involving classical path planners are much lower than those involving an approach based on artificial intelligence. Nevertheless, researchers uses classical path planners on the development of hybrid (Patle *et al.*, 2019) and optimized path planners (Steffi *et al.*, 2021).

On recent works, Zaheer *et al.* (2015), compared the performance of four classical path planners – RRT, PRM, A* and Artificial Potential Fields (APF) with a new one named Free Configuration Eigen-Spaces (FCE). Khanmirza *et al.* (2017) conducted a study comparing classical deterministic and probabilistic approaches. Baziyad *et al.* (2019) carried out a comparative study between three heuristic optimization techniques for robot path planning: Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC). Mokwele *et al.* (2020) compared the performance of Potential Field (PF) Method, PRM, RRT* and A* through simulations. Wahab *et al.* (2020) carried out experimental work comparing classical - PF, DA, RRT and PRM - and meta-heuristics approaches – GA, PSO, Differential Evolution (DE) and Cuckoo Search Algorithm (CSA). Pattnaik *et al.* (2021) compared the performance of three meta-heuristics path planners, PSO, GA and Chemical Reaction Optimization (CRO), with a new proposed path planner, which is a hybrid of particle swarm and chemical reaction optimization (HPCRO).

This study involves DP, which is usually not included in other similar comparisons focused on mobile robots (Steffi *et al.*, 2021) and Hybrid A*. Therefore, this paper has the distinction of including in the same comparative study the five planners mentioned above in the same comparative study, including Hybrid A*, which, until the closing of this paper, has not been found in any other comparative study.

2. THEORETICAL BACKGROUND

Path planning can be defined as the work of building a path that leads from a starting point to an ending point, avoiding any obstacles that may exist between them. Path planners are divided in two categories: global path planners,

which consider the robot's capacity to identify the object position relative to its own reference axis while moving towards a goal; and local path planners, which concerns to the robot capabilities of navigation through a planned path, considering environmental dynamics (Patle *et al.*, 2019). The following sections presents each path planner.

2.1 Dynamic Programming - DP

DP is considered as the most robust numerical approach to achieve optimality (Mohamed *et al.*, 2019) and is the base for some searching algorithms, including DA and A*. It uses a model based on a discrete model of dynamic system and a cost function that is additive over time, to anticipate to some extend the outcome of decisions made in stages, aiming to minimize a cost. These decisions are ranked according to the sum of the present cost and the expected future cost. The system's dynamic is given by

$$x_{k+1} = f_k(x_k, u_k, w_k), \; k = 0,1, \dots, N-1 \tag{1}$$

where k is the index of discrete time, x_k is the state of the system in discrete time k, u_k is the control input in discrete time k, w_k is the random or disturbance parameter, N is the horizon or number of times of application of control, and $f_k(,)$ is a function that describes the system. The cost is defined by the *scalar-valued additive* cost function, given by

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \tag{2}$$

where g_N is the terminal cost, g_k is the stage cost, and $\sum_{k=0}^{N-1} g_k$ is the accumulated cost. For the solution of the shortest path problem, a graph defined by a space of vertices $\mathcal{V}$ and a space of weighted edges $\mathcal{C}$ is considered as

$$\mathcal{C} := \{(i, j, c_{i,j}) \in \mathcal{V} \times \mathcal{V} \times \mathbb{R} \cup \{\infty\} | i, j \in \mathcal{V}\} \tag{3}$$

where $c_{i,j}$ is the cost of the edge that connects nodes i and j, which in this case, is the distance from vertices i to j. A path named as Q is defined as a list of q ordered nodes, in a way that each element of Q is part of $\mathcal{V}$:

$$Q := (i_1, \dots, i_q) \tag{4}$$

The length of the path Q, called J_Q, is the sum of the lengths of the segments that make up the path, defined as:

$$J_Q = \sum_{h=1}^{q-1} c_{i_h, i_{h+1}} \tag{5}$$

Given a node S $\in \mathcal{V}$, which is the first one of a path, and a node T $\in \mathcal{V}$, which is the last one, all the paths starting at any node S and ending at any node T are given by $\mathbb{Q}_{S,T}$. The main objective is to find the shortest path, Q^*, between S $\in \mathcal{V}$ and T $\in \mathcal{V}$, i.e.

$$Q^* = \underset{Q \in Q_{S,T}}{\arg min} J_Q \tag{6}$$

To validate the problem, it is necessary to satisfy the assumption that: *for all* $i \in \mathcal{V}$ *and for all* $Q \in \mathbb{Q}_{i,j}$, $J_Q \geq 0$. This ensures that there are no negative cycles in the graph. So $i \in \mathcal{V}$, $c_{i,j} \geq 0$ (D'Andrea, 2020).

2.2 Dijkstra's Algorithm - DA

DA is a graph searching method that finds the shortest path with lower cost between the nodes of a graph, by repetitively calculating the shortest distance from an initial point to a goal point while excluding the longer distances (Gul *et al.*, 2019).

Given a directed graph made by pairs of nodes connected by edges e, a cost $l(e)$ can be defined as the cost to apply an action through e. This cost $l(e)$ can also be represented in the form of a state space $l(x,u)$, which is the cost of applying an action u of state x. The final cost of a path is the sum of all costs of its individual edges. A priority queue Q is classified according to the *cost to come* function $C: X \rightarrow [0, \infty]$. Each state x is then associated with an *optimal cost to come* value $C^*(x)$, starting with the initial state x_I. All costs $l(e)$, in all possible paths from x_I to x, are summed and the path that has the lowest cumulative cost is the $C^*(x)$. If it is not possible to evaluate if a cost is the optimal one, it is defined as $C(x)$.

The algorithm starts with $C^*(x_I)$ null and each time it generates a given state x', an associate cost $C(x') = C^*(x) + l(e)$ is calculated, where e is the edge connecting state x to x'. Therefore, $C(x')$ is the best cost to come at time. However, it cannot be named as C^*, since it is not possible to classify it as optimal. If a state x reaches its lowest $C(x)$ value, it goes to the top of Q and then it is removed from the queue. Therefore state x became a *dead state* and it is not possible to reach it at a cost lower than $C(x)$. Since the calculation of the optimal cost of every *dead state* is assumed to be correct, these

values do not change. Thus, for the first element x of Q the *cost to come* $C(x)$ must be optimal, because any other path with a lower cost would have to travel through some other state in Q. However, the costs of all other states are higher, since all the paths that passes only through the *dead states* were considered in $C(x)$ calculations. At last, $C(x)$ is then converted to $C^*(x)$ (LaValle, 2006).

2.3 Probabilistic Roadmaps – PRM

PRM uses node sampling to create an undirected graph called roadmap, which connects the nodes of the graph in the unoccupied space of an environment, C_{free}. Applying this technique, a robot can travel through the nodes, in both directions, using the edges of the graph as roads. Since the planner is able to find paths with precision avoiding obstacles, it is more applicable to kinetic problems (Lynch and Park, 2017). The algorithm has two phases. The first one is the *learning phase*, which consists in the construction of the roadmap. The second is the *query phase*, in which any two given points in C_{free}, an initial and a goal, are connected to two nodes of the roadmap. Then, a search for paths that interconnect these two node is carried out through the roadmap (Kavraki *et al.*, 1996). A local path planner is required to search among the roadmaps and build a path between the initial and objective nodes. According to Lynch and Park (2017), the A * algorithm is commonly used for this purpose. Since PRM does not use gridmap to represent its configured space C, the resolution of the map is maintained even if the configured space suffers an expansion. This is an advantage of PRM over other planners, which uses gridmap to represent C. With this feature, the area of action of the local planner does not reduce exponentially with the growth of the space, as it occurs with these planners (Lynch and Park, 2017).

2.4 Rapidly-exploring Random Trees – RRT

The Rapidly-exploring Random Trees (RRT) is a sampling-based planner (LaValle, 1998). The algorithm builds a directed graph called tree, in which each node has only one edge connecting it to a *parent node*. Other types of nodes of the tree are the *root node*, which is the only one that do not have a parent node, and the *leaf nodes*, which do not have *children nodes*. Since the construction of the trees do not have cycles, their nodes connect only once (Lynch and Park, 2017). RRT is able to attend many path-planning problems. It has been developed to deal, mainly, with non-holonomic constraints, including dynamics, and a relatively high number of degrees of freedom. The tree generated expands iteratively through commands that provides a connection directed at randomly selected nodes (LaValle, 1998). Starting from an initial state x_{start}, the planner search for a path without collisions to an objective region X_{obj}. For kinematic applications the state x is a just a setting of Cartesian coordinates and orientation in the free space, C_{free}. For dynamic applications, the state also includes speeds (Lynch and Park, 2017). To build the tree, the planner randomly selected a point in the configured space C called x_{sample} and verifies if it belongs to free space C_{free}. The vertex of the tree that is closer to x_{sample} is called x_{near} and an attempt to connect these two nodes is carried out. If there no obstacles between x_{sample} and x_{near} and the distance between them is within a given arbitrary value, x_{sample} is connected to the tree. If not, a new node, called x_{new}, located between x_{sample} and x_{near}, is selected and a new attempt of connection is made. Thus, the planner incrementally connects the initial and goal points, node by node, like branches of a tree (Mohamed and Milan, 2014).

2.5 Hybrid A*

The Hybrid A* algorithm was created as an extension of the traditional A*, in order to attend vehicles with non-holonomic constrains. While A* most common use is in discrete state spaces, Hybrid A* performs a continuous search in space, using a set of small pre-computed curves called motion primitives. The motion primitives are able to specify which states are possible for the robot to reach, in order to build a tree.

The algorithm demands that the vehicle's state shall have a continuous representation given by position x in Cartesian coordinates and its orientation θ. Moreover, the space shall be discretized for the planner be able to make research over it. Since the position x is defined by pairs of Cartesian coordinates, the translation movement is already discretized. The orientation θ has a discrete equivalent, $\tilde{\theta}$ ("tilde" means discrete variable), which is assumed to be equal to its continuous counterpart because the number of motion primitives is limited so that they will always end up with an orientation that fits the discretization scheme. In addition Hybrid A* uses the robot's discrete position, $\tilde{x}$, for searching in discretized space. This parameter is sufficient for environmental research, but the algorithm also stores the coordinates of the continuous position x for each visited cell. Then, these cells are used as root for the nodal expansion, which in turns is, basically, research in the neighborhood. The hybrid nature of the algorithm, discrete and continuous, guarantees the viability of the generated path. It would not be possible if only the centers of each cell were considered in the research. (Petereit *et al.*, 2012).

3. METHODOLOGY

The study is carried out through simulations in MATLAB (version 2019b), where three parameters are submitted to evaluation in order to compare the planners' performance. The first is the processing time, which is the amount of time

that it takes the planner to generate a path and it is related to the computational capacity required to achieve the goal. The second is the number of steps, which means how many nodes (points in a Cartesian plane, for instance) does the path have to go through in order to achieve the goal - this parameter is important for mobile robots that navigate through straight lines. For these robots, each step may represent a point of stop and restart of the motion, forcing the motors and demanding more power consumption. The last parameter, the path length, is literally the total length of the path from the start to the ending point.

Three maps were created to carry out the simulations, each one focusing on a specific aspect to be evaluated by planners. They all have 25 x 25 units long based on gridmaps with 1 x 1 unit long cells. The maps are shown in Figure 1, where the green stars indicates the starting points and the yellow stars the objective points. The first map of Figure 1, called Labyrinth, aims to evaluate the capability of algorithms to plan a path in a complex environment with multiple ways. The second map, called "Narrow Labyrinth", inflates the obstacles of the first map narrowing the passages in order to evaluate the planners' capacity of planning a path in a more challenging environment. The third map, called “Zigzag”, has the objective to ascertain the behavior of planners in a long way, in which it is necessary to reverse the direction and make sharp turns. The start and objective points of the first two maps have Cartesian coordinates (2,2) and (23,23), respectively, and for third map the coordinates are (2,2) and (3, 23).

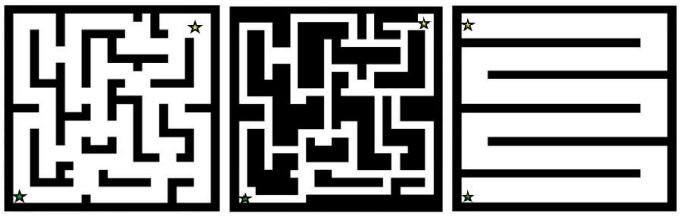


Figure 1 – Simulation maps

3.1 Simulation Procedures

For each planner, series of 10 simulations are performed and for each series, one or more parameters of the planners are modified. If possible, the first series is performed with the planner’s default configuration. If not, the parameters are adjustable until there is a reasonable number of outputs (path generation). For all individual simulations the processing time is measured, the number of steps are recorded, and the path length is calculated. This procedure is repeated until a satisfactory quantity of data is obtained, allowing the generation of graphs that represent the behavior of the planner.

DA and DP simulations use a program developed by Balcitar (2020). The program generates nodes randomly in the free space of the map and make as many connections as possible between them. The chosen algorithm is then applied to plan a path between these conections. The number of nodes is the parameter that varies for each series.

The PRM planner is the MATLAB object “*mobileRobotPRM*” which allows two parameters to be set: the number of nodes and the maximum distance between two nodes, called “delta”, measured in units long. The program developed for the simulations increases the number of nodes by ten if the attempt to reach the goal failures. This procedure is repeated until the goal is reached. In the first series, delta is equal to 10 units long and for the following series its values are increased by five until reaching 25 units long, which is the length of the external edges of the maps. Therefore, there are four series, with delta equal to 10, 15, 20 and 25 units long.

The simulations with RRT are carried out using the MATLAB Navigation Toolbox object “*plannerRRT*”, which uses state space to generate the tree branches and to create the path. The parameters of choice to be varied at each series of ten simulations are the maximum distance between nodes, “delta”, the maximum number of nodes of the tree and the maximum number of iterations. The values of delta of each series are 0.5, 1, 2, 3, 5, 7, 10, 15, 20 and 25 units long.

The Hybrid A* planner is the “*plannerHybridAStar*”, an object of MATLAB's Navigation Toolbox focused on non-holonomic vehicles, such as automobiles, which works with gridmaps. It is the planner with the highest number of adjustable variables. It has been chosen to modify only one variable for each series of ten simulations and on the last series, all changes that cause a positive impact are implemented. The motion primitives used by this planner have a restriction on its length as well as on its minimum turning radius, given by

$$L_{prim} = \lceil \sqrt{2} \, . \, D_{cell} \rceil \tag{7}$$

where L_{prim} is the motion primitive length and D_{cell} is the dimension of the cell of the gridmap.

$$R_{min} = \frac{(2 \, . \, L_{prim})}{\pi} \tag{8}$$

where R_{min} is the minimum turning radius

With the exception of Hybrid A*, all the other four planners generate paths composed of straight lines. Since all the nodes of the paths are given in Cartesian coordinates, the total lengths of their paths can be obtained adding the segments of lines, which in turns is their Manhattan distance. The Hybrid A* path has some segments that are straight, which length is calculated the same way, and the curved segments which length is given by the angle between two nodes times R_{min}.

4. SIMULATION RESULTS

This section presents the values of processing time, number of steps and path length computed during the simulations. An important detail is that DA, DP and PRM presents mean values of the ten simulations that compose a series. For RRT and Hybrid A* this is only true for the processing time because these two planners always repeat the path if none of their parameters are changed. Therefore, since the parameters are unchanged along a series, the path is the same for each simulation and so is the number of steps and the path length. This behavior is valid for the simulations in all three maps. In all graphs the parameters *tp(s)* is for processing time in *seconds*, *steps* is the number of steps and *d(u.l)* is the path length in *units of length*. Figure 2 shows the simulation results for DA (a, b and c) and DP (d, e and f).

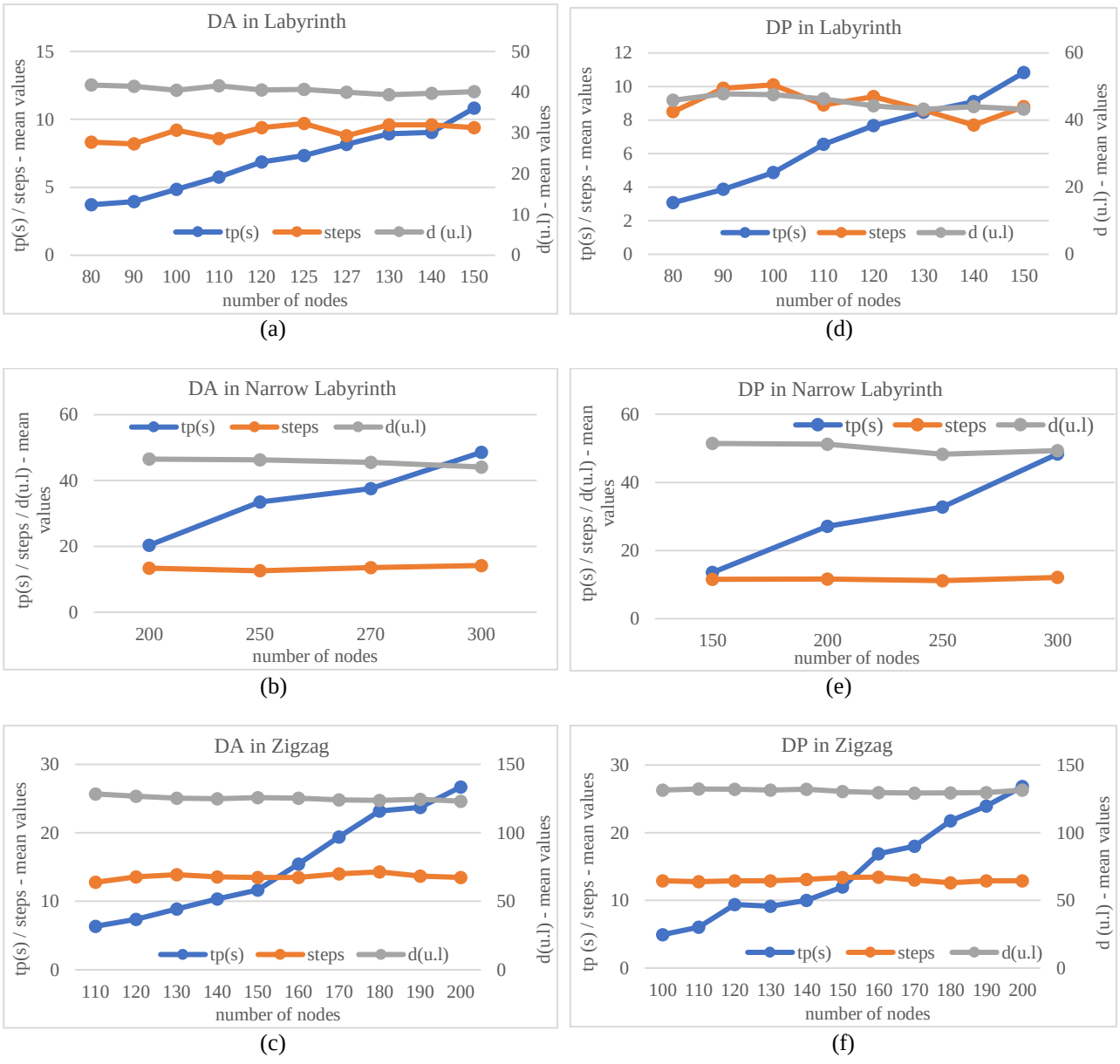


Figure 2 – Simulation results of DA and DP.

Figure 3 shows the simulation results for PRM (a, b and c) and RRT (d, e and f). The graph of RRT in the Zigzag map, figure 3-f, shows the values of delta "3*" and "3**" in the last two simulations. It means that the value of 3 for delta

was repeated two times. In the first, the maximum number of nodes and iterations were doubled and in the second time, these parameters were multiplied tenfold.

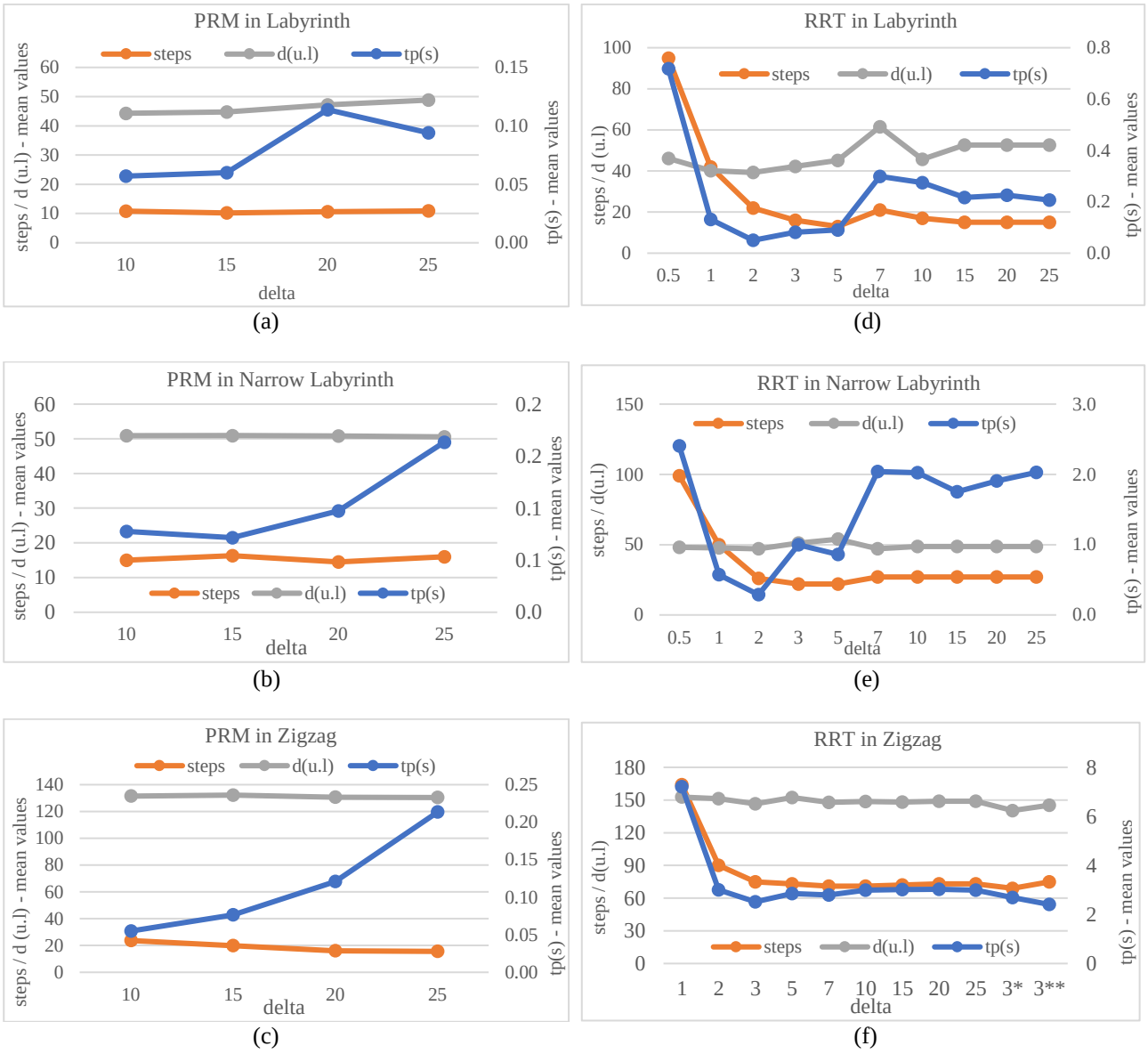


Figure 3 – Simulation Results for PRM and RRT.

Figure 4 presents the simulation results for Hybrid A* in Labyrinth (a) and Narrow Labyrinth (b) maps.

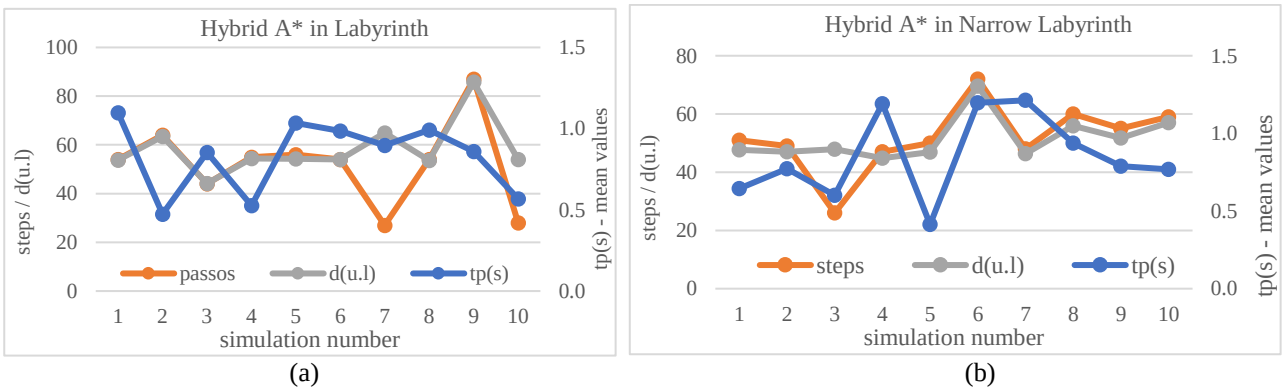


Figure 4 – Simulation results of Hybrid A* on Labyrinth and Narrow Labyrinth maps.

At last, Figure 5 shows the results of the simulation of Hybrid A* in the Zigzag map.

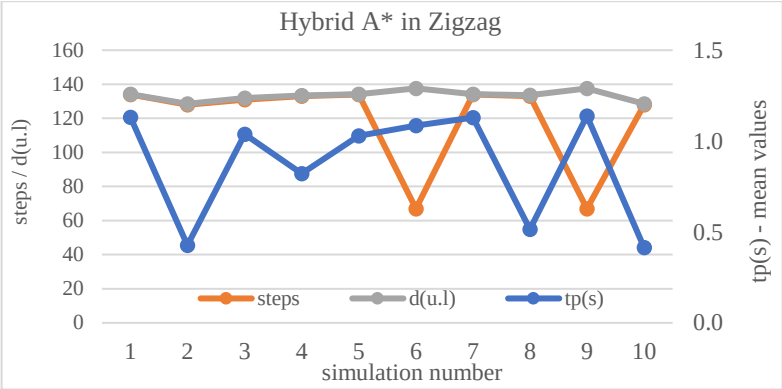


Figure 5 – Simulation results of Hybrid A* in Zigzag map.

Figure 6 shows examples of paths generated by the planners in the three maps: DA (column a), DP (column b), PRM (column c), RRT (column d) and Hybrid A* (column e).

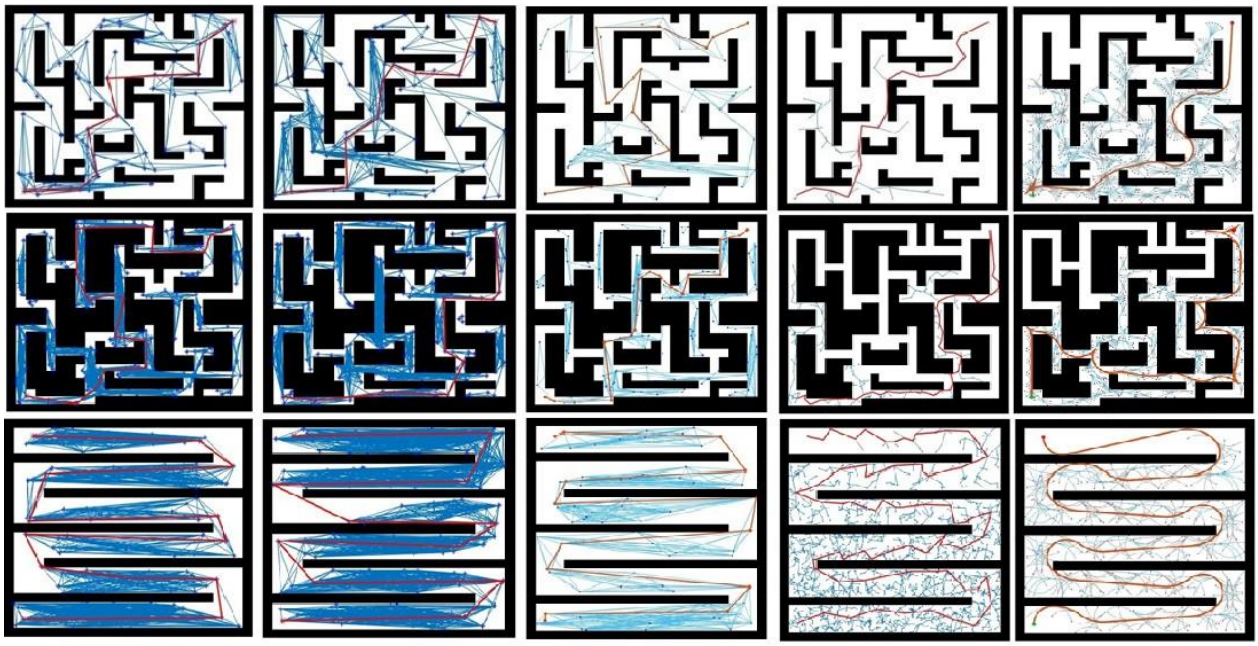
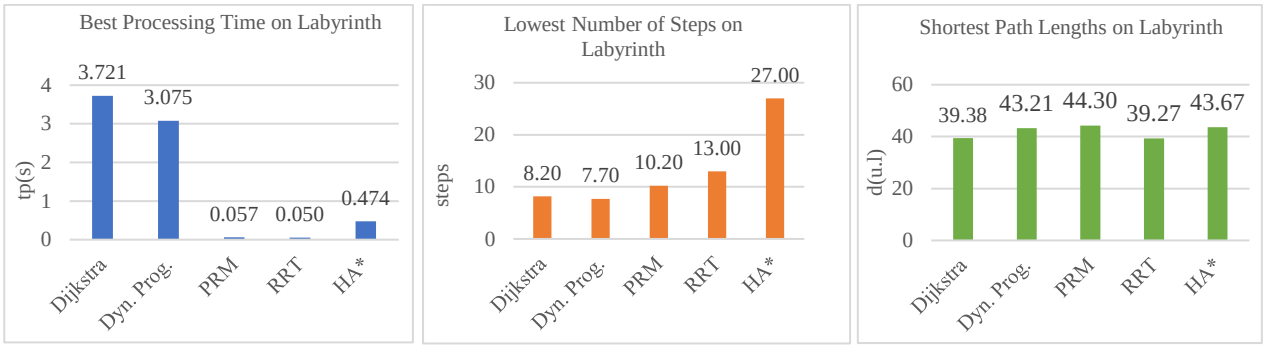


Figure 6 – Examples of paths generated by the planners.

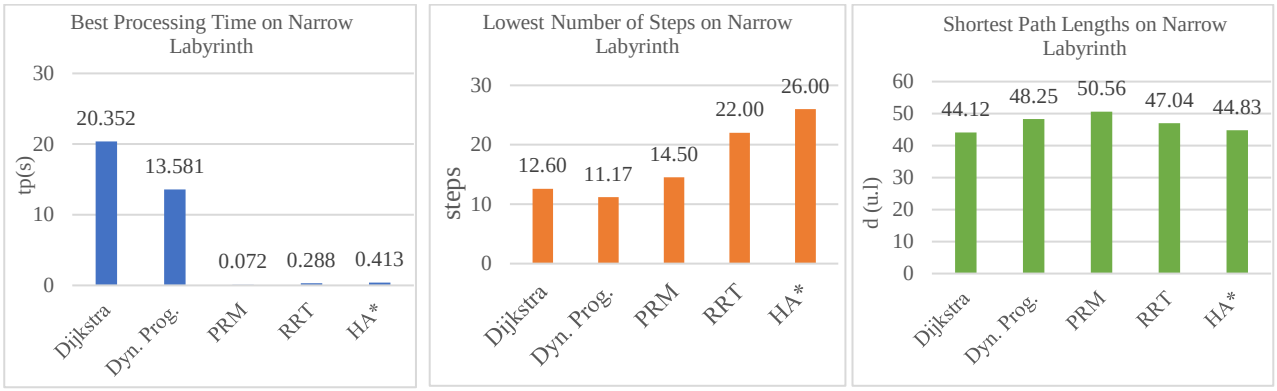
4.1 Comparison between planners

Figure 7 shows the best results of each planner regarding processing time (a), number of steps (b) and path length (c) for the simulations in the Labyrinth map. The shortest mean processing time in the Labyrinth map was obtained by RRT in its third series of simulations. Figure 7-a also shows the great disparity between the processing times of DA and DP relative to the others. Furthermore, they were not successful in all attempts to generate paths in the first series of both planners: DA generated nine paths in ten attempts, while DP generated eight paths. DP presented the path with the lowest number of steps, in its seventh series of simulations (Figure 7-b). The PRM planner presents a number of steps close to that of DP but with a much shorter processing time. The Hybrid A* obtained the largest number of steps due to its mainly curvilinear trajectory, since more steps are needed to draw a curve than to draw a straight line. RRT presented the shortest path length in its third series of simulations in the Labyrinth map, which was also the one that obtained the shortest average processing time. The shortest path lengths is the parameter that shows the smallest difference between the values of each planner. The greatest difference, for example, which is between the path lengths of RRT and PRM, is only 5.04 units long. This is because each algorithm was able to find its most suitable solution in the map. Figure 8 shows the best results for simulations in the Narrow Labyrinth map.



(a) (b) (c)

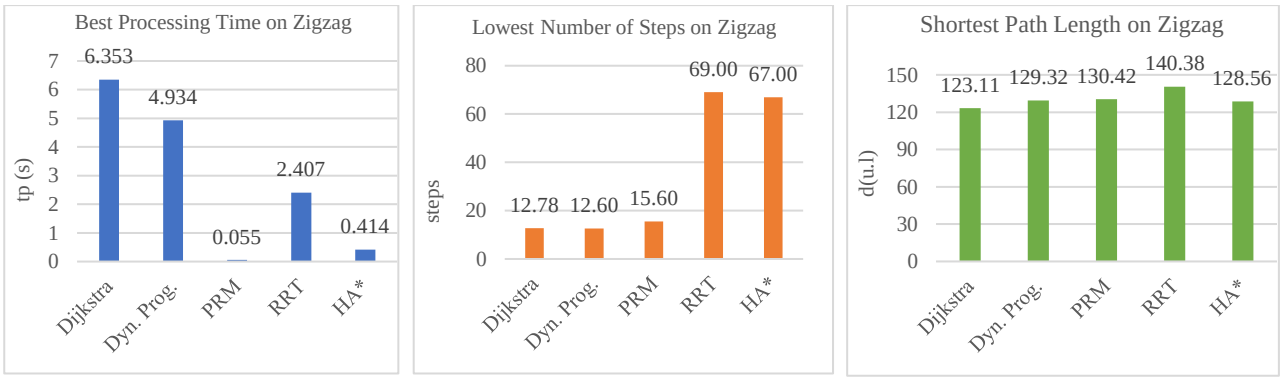
Figure 7 – Best results of each planner in the Labyrinth map.



(a) (b) (c)

Figure 8 - Best results of each planner in the Narrow Labyrinth map.

With remarkable superiority, PRM obtained the best processing time performance on Narrow Labyrinth, in its second series (Figure 8-a). The mean processing time presented by DA and DP, demonstrates the difficulty that these two planners had in generating trajectories in Narrow Labyrinth map. In addition, like in the simulations of the first map, DP presented the lowest mean number of steps (Figure 8-b). However, again, PRM is a better option, since its average number of steps is only 3.33 steps above of DP's, but with a processing time 347.99 times shorter. DA was the planner that obtained the shortest mean path length on Narrow Labyrinth, in its fourth series (Figure 8-c). However, the processing time for this series is the highest of the planner for this map. Regarding the path length, Hybrid A* is a good alternative, as its shortest length is only 0.71 units longer than DA's path length, and its processing time is 40.75 times shorter. Figure 9 presents the results with the best performance in the Zigzag map.



(a) (b) (c)

Figure 9 - Best results of each planner in the Zigzag map.

In the Zigzag map simulations, PRM obtained the lowest mean processing time, significantly better than the other ones (Figure 9-a). The planner also present good results regarding number of steps and path length. Again, DA and DP presented the highest values of processing time; followed by RRT, which best processing time is about 2 seconds less

than DP's time. Still regarding DP, the planner presented the lowest mean number of steps again, followed very close by DA, as shown on Figure 9-b. However, just like in the Narrow Labyrinth's simulations, PRM presented a difference of only three steps to DP value, with a much better processing time. Regarding path lengths (Figure 9-c), DA's performance was the best for the map with the longest way (Zigzag), as well as for the one with more restricted routes, the Narrow Labyrinth. Its disadvantage is the processing time, higher than the ones obtained by the other planners. The differences between path lengths on the Zigzag map are relatively small. Among the shortest paths, RRT presents the largest one with 17.27 units longer than DA path.

4.2 Discussion

It is clear that DA and DP are, among all five path planners, the ones that require more processing time. Differently from the other planners, these two algorithms promote as many connections as possible between the nodes. Therefore, the higher the number of nodes required, the longer the processing time will be. This behavior was a disadvantage especially in the Narrow Labyrinth map, where a greater number of nodes was necessary to create the paths. Despite that, DA was able to generate the shortest path in this challenging scenario, since the planner is designed for path optimization. This is the planner's tradeoff: an optimized path against high computational cost. DP, in turn, has the best mean number of steps in all three maps.

For PRM, the processing time is related to the value of delta, the maximum distance that a node can connect to another one. A higher value of delta implies more possibilities for a node to make connections, thus, increasing the processing time. Nevertheless, this planner presented the shortest processing time on two of three scenarios. PRM also showed the largest path length on two maps. This happens because the planner tries to find a viable path, in the shortest time possible. This is why in Narrow Labyrinth the planner showed the best processing time, but also the largest path.

The processing time for RRT is a function of how much the tree will have to spread its branches. A small value of delta, like 0.5, implies on the necessity of generating more nodes and connections. In Zigzag map, a greater nodular expansion was needed independently of the delta causing the processing time to increase highly. Simulations also revealed that a value of delta compatible to the map size permits the planner to generate paths in adequate time and length. According to Figure 3 delta equals 2 units long was the best for Labyrinth and Narrow Labyrinth and delta equals 3 units was the best for Zigzag. In order to verify this feature, these values were repeated two more times in series where the maximum number of nodes and iterations were doubled and multiplied tenfold.

Figures 4-b and 5 shows that for Hybrid A* the number of steps and the path length are similar. This is because the planner works in discretized space. When the planner property "interpolation distance" is set to "1", the step of the planner is equal to the size of the cell. For example, in the graph of Figure 4-a, Hybrid A* in Labyrinth, simulations 1 to 6, has "interpolation distance" set to "1", so that the values of step and length are very close. In simulation 7, the "interpolation distance is set to "2" and, in consequence, the number of steps is about half of the path length. In Narrow Labyrinth, the planner showed a good performance in path length, getting close to DA, because the primitive length had to be adjusted to the dimensions of the map. The "A*" part of the planner also grants a directed search for the objective. This is why the best processing time of this planner for all the three maps has such small difference. Table 1 compiles all the best results for each planner.

Table 1 – Best Simulation Results of Each Path Planner

	Labyrinth			*Narrow Labyrinth*			*Zigzag*		
Planner	$\overline{tp}(s)$	*steps*	*d(u.l)*	$\overline{tp}(s)$	*steps*	*d(u.l)*	$\overline{tp}(s)$	*steps*	*d(u.l)*
DA	3.721	8.200[1]	39.384[1]	20.352	12.600[1]	44.116[1]	6.353	12.778[1]	123.114[1]
DP	3.075	7.700[1]	43.206[1]	13.581	11.167[1]	48.251[1]	4.934	12.600[1]	129.320[1]
PRM	0.057	10.200[1]	44.302[1]	0.072	14.500[1]	50.556[1]	0.055	15.600[1]	130.418[1]
RRT	0.050	13.000	39.267	0.288	22.000	47.044	2.407	69.000	140.380
Hybrid A*	0.474	27.000	43.674	0.413	26.000	44.825	0.414	67.000	128.555

[1] Mean values

5. CONCLUSIONS

In this paper, a comparative study of five state-of-the-art path planner algorithms have been made through simulations performed in the MATLAB environment, in three different maps. PRM planner showed capacity of finding a path in the shortest time possible even in a challenging or an open environment. DA and DP presented good performances regarding path length and number of steps respectively, but with a high cost of computational resources. RRT obtained good results in more restricted environments with properly adjusted parameters. Hybrid A* presented a stable behavior in all three environments, especially regarding processing time.

6. ACKNOWLEDGEMENTS

This work is supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

7. REFERENCES

Balcilar, M., 2020. “RobotPathPlanning”. https://github.com/muhammetbalcilar/RobotPathPlanning. Accessed February 18, 2020

Baziyad, M., Nassif, A. B., Rabie, T., Fareh, R, 2019. “Comparative study on the performance of heuristic optimization techniques in robotic path planning”. In *Proceedings of the 3rd International Conference on Advances in Artificial Intelligence* – ICAAI 2019. Istambul, Turkey.

D’Andrea, R., 2020. “Dynamic programming and optimal control”. ETH Zurich. http://www.idsc.ethz.ch/education/lectures/optimal-control.html. Accessed February 2, 2021.

Gul, F., Rahiman, W., Nazli Alhady, S. S., 2019. “A comprehensive study for robot navigation techniques”. *Cogent Engineering*, Vol. 6.

Kavraki, L.E., Svestka, P., Latombe, J., Overmars, M.H., 1996. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE Transactions in Robotics and Automation*, Vol. 12, No. 4, pp. 566–580.

Khanmirza, E., Haghbeigi, M., Nazarahari, M., Doostie, S., 2017. “A comparative study of deterministic and probabilistic mobile robot path planning algorithms”. In *Proceedings of 5th RSI International Conference on Robotics and Mechatronics* – ICRoM 2017, pp. 534-539

LaValle, S. M., 1998. “Rapidly-exploring random trees: a new tool for path planning”. In *1998 Technical Report* – TR’98. Department of Computer Science - Iowa State University. No. 11.

LaValle, S. M., 2006. “Planning Algorithms”. Cambridge University Press. Available for downloading at http://planning.cs.uiuc.edu/.

Lynch, K.M., Park, F.C., 2017. “Modern robotics. Mechanics, planning, and control”. Cambridge University Press. 1st ed.

Mohamed, A., Ren, J., Huang, X., Ouda, A.N., Abdo, G.M., 2019. “Comparative study of dynamic programming and pontryagin's minimum principle for autonomous multi-wheeled combat vehicle path planning”. In *International Journal of Heavy Vehicle Systems*. Vol. 26, No. 3-4, pp.565-577.

Mohamed, E., Milan, S., 2014. “Sampling-based robot motion planning: a review”. In: *IEEE Access*, Vol. 2, pp. 56–77.

Mohd, N.C., Mohantab, J.C., 2018. “Methodology for path planning and optimization of mobile robots: a review”. In *Proceedings of 2018 International Conference on Robotics and Smart Manufacturing* – RoSMA 2018. Procedia Computer Science. No. 133, pp. 141-152.

Mokwele, D., Du, S., Benade, J., 2020. “Comparative study of autonomous path planning methods for mobile robot”. In *Proceedings of 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems* – icABCD 2020. Durban, KwaZulu Natal, South Africa

Patle, B.K., Babu, G., Pandey, A., Parhi, D.R.K., Jagadeesh, A., 2019. “A review: On path planning strategies for navigation of mobile robot”. *Defense Technology*. Elsevier, No. 15, pp.582-606.

Pattnaik, S. K., Mishra, D., Panda, S., 2021. “A comparative study of meta-heuristics for local path planning of a mobile robot”. *Engineering Optimization*. Vol. 1, No. 19.

Petereit, J., Emter, T., Frey, C. W., Kopfstedt, T., Beutel, A., 2012. “Application of hybrid a* to an autonomous mobile robot for path planning in unstructured outdoor environments”. In *Proceeding of 7th German Conference on Robotics* - ROBOTIK 2012, pp. 1-6.

Steffi, D.D.D., Mehta, S., Venkatesh, K.A., Dasari, S.K., 2018. “Robot path planning-prediction: a multidisciplinary platform: a survey”. In *Data Science and Security. Lecture Notes in Networks and Systems*, Vol.132, pp. 214-219.

Wahab, M. N. A., Nefti-Meziani, S., Aryabi, A., 2020. “A comparative review on mobile robot path planning: classical or meta-heuristic methods?” In *Annual Reviews in Control*, Vol. 10, No. 1.

Zaheer, S., Jayaraju, M., Gulrez, T., 2015. “Performance analysis of path planning techniques for autonomous mobile robots”. In *Proceedings of IEEE International Conference on Electrical, Computer and Communication Technologies* – ICECCT 2015, pp. 1-5.

Zhang, H., Lin, W., Chen, A., 2018. “Path planning for the mobile robot: a review”. *Symmetry*, Vol.10, No. 10, 450.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.