A comparative analysis of the Probabilistic Roadmap Method and the A* algorithm in autonomous mobile robot navigation

Sandra Alejandra Rodríguez Hernández, Ulises Orozco-Rosas, Jesús Antonio Camacho González*

CETYS Universidad, Av. CETYS Universidad No. 4. El Lago, C.P. 22210, Tijuana B.C., México
[sandra.hernandez]@cetys.edu.mx, [ulises.orozco, jesus.camacho]@cetys.mx

Abstract

Path planning is fundamental in autonomous navigation, especially in industrial environments with priorities for time efficiency and optimal resource management. This work examines the Probabilistic Roadmap Method (PRM) and the A* algorithm, comparing their path generation time, trajectory distance, and tracking time to determine which performs better under these metrics. For the evaluation, both algorithms were implemented in MATLAB, and the CoppeliaSim simulator was used to model the movement of a differential robot in different environments inspired by industrial work cells. Both algorithms stand out for their versatility, as path generation is independent of the type of robot, allowing their application in differential, Ackermann, and omnidirectional robots with minimal adjustments to navigation. Choosing the right path-planning algorithm can significantly enhance the performance of robotic systems by reducing operational delays and optimizing efficiency in industrial environments. Understanding the differences between these algorithms is crucial for improving navigation in structured spaces, such as industrial work cells, where safety and efficiency are critical factors. By providing a detailed analysis of their performance, this research contributes to developing more efficient robotic systems, particularly in scenarios where precise and reliable navigation is essential.

Key Words— A* Algorithm, Probabilistic Roadmap Method, Path Planning, Autonomous Navigation, Mobile Robots

Resumen

En la navegación autónoma, la planificación de rutas es un aspecto fundamental, especialmente en entornos industriales donde la eficiencia en el tiempo y el uso óptimo de los recursos son prioritarios. En este trabajo, se comparan el método Probabilistic Roadmap (PRM) y el algoritmo A en términos de tiempo de generación de ruta, distancia de la trayectoria y tiempo de seguimiento. El objetivo principal es determinar cuál de los dos algoritmos ofrece un mejor desempeño bajo estas métricas. Para llevar a cabo la evaluación, se implementaron ambos algoritmos en MATLAB y se utilizó el simulador CoppeliaSim para modelar el movimiento de un robot diferencial en distintos entornos inspirados en celdas de trabajo industriales. Ambos algoritmos destacan por su versatilidad, ya que la generación de rutas es independiente del tipo de robot, lo que permite su aplicación en robots diferenciales, Ackermann y omnidireccionales con ajustes mínimos en la navegación. Seleccionar el algoritmo de planificación de rutas adecuado puede mejorar significativamente el desempeño de los sistemas robóticos, reduciendo retrasos operativos y optimizando la eficiencia en entornos industriales. Comprender las diferencias entre estos algoritmos es fundamental para mejorar la navegación en espacios estructurados, como celdas de trabajo industriales, donde la seguridad y la eficiencia son aspectos críticos. Al proporcionar un análisis detallado de su desempeño, esta investigación contribuye al desarrollo de sistemas robóticos más eficientes, especialmente en escenarios donde la navegación precisa y confiable es esencial.*

Palabras Clave— Algoritmo A*, Probabilistic Roadmap, Planificación de Trayectoria, Navegación Autónoma, Robots Móviles

1. INTRODUCTION

There is a constant pursuit of faster and more efficient path-planning algorithms, especially in an industrial environment where time directly impacts production and the limited available resources. This leads to the implementation of navigation algorithms that ensure reaching the goal as quickly and safely as possible [1].

Among the classical autonomous navigation algorithms are the Probabilistic Roadmap Method (PRM) and the A* algorithm. These algorithms have proven effective, successfully generating feasible paths in known environments. However, in industrial applications, such as material transportation, it is important to ensure successful transport and complete it in the shortest possible time and with maximum safety [2].

Therefore, it is essential to evaluate the performance of these algorithms in terms of path generation time, path length, and tracking time [3]. This is because the shortest path does not always guarantee the lowest tracking time.

This study presents a comparison between the PRM and the A* algorithm to assess their performance in known environments. The path planning is implemented on a differential robot within different scenarios inspired by the layout of industrial work cells, using the CoppeliaSim simulator.

2. THEORETICAL FRAMEWORK

Path planning is the process by which a robot or autonomous system determines the optimal route to move from a starting

* Corresponding author.

point to a specific destination [4]. This process involves using algorithms to evaluate possible trajectories, considering factors such as distance, the presence of obstacles, and environmental constraints [5]. Path planning is essential for autonomous navigation, enabling the robot to move efficiently and safely within its operational environment [6].

Focusing on path planning, this work addresses two of the most employed alternatives, the A* algorithm, and the Probabilistic Roadmap Method. The **A* (A-star) algorithm** is a search and path-planning algorithm. It is an extension of Dijkstra's algorithm that incorporates a heuristic to improve search efficiency.

The algorithm evaluates each node based on two costs: the accumulated cost from the start to the current node, $g(n)$, and a heuristic estimate of the cost from the current node to the goal, $h(n)$. The sum of these costs guides the search, allowing the selection of the node with the lowest value, $f(n)$, as can be seen in Eq. 1.

$$f(n) = g(n) + h(n) \quad (1)$$

The process starts with the initial node and expands neighboring nodes, moving each evaluated node from the open set (nodes yet to be explored) to the closed set (nodes already explored). If a node provides a shorter path, its costs are updated, and it is added to the open set. This process repeats until the goal node is found or the open set is exhausted. Once the goal is reached, the optimal path is reconstructed by tracing back from the goal node to the start node [7].

The **Probabilistic Roadmap Method** (PRM) is a path-planning technique used in robotics. It consists of two main phases: the construction phase and the query phase.

In the construction phase, a series of nodes are randomly generated within the free configuration space (the space where the robot can move without colliding). Nearby nodes are connected by direct paths if these paths do not intersect obstacles, forming a graph of possible routes. In the query phase, the start and goal points are added to the graph and connected to the nearest existing nodes [8].

Now, once the path has been planned, we require the controller to track the resultant path. The **Pure Pursuit controller** is available in MATLAB. Pure Pursuit is a path-tracking algorithm that calculates the angular velocity command to steer the robot, assuming a constant linear velocity, from its current position toward a look-ahead point in front of the robot. The algorithm then moves the look-ahead point along the path based on the robot's current position until reaching the final point of the trajectory [9].

The `controllerPurePursuit` object is not a traditional controller but rather a tracking algorithm for path-following purposes. Desired linear velocities and maximum angular velocities can be specified, and these properties are defined based on the vehicle's specifications.

Metrics for evaluating the planning algorithms. Initially, multiple experiments were conducted with different parameter values for each algorithm, defining these parameters empirically.

The performance of the PRM depends on the number of nodes provided to the algorithm. Therefore, experiments were conducted with three different values: 100, 300, and 500 nodes. The goal was to determine the optimal number of nodes that provide the best balance between the route generation time, the robot's tracking time within the simulation, and the shortest path length.

To determine the required number of repetitions for reliable data, preliminary test runs were performed. Using a 20% margin of error and a 90% confidence level, it was determined that 67-68 samples were needed for each node count and each map.

Similarly, for the A* algorithm, the number of repetitions was calculated to ensure reliable data. With a 20% margin of error based on the standard deviation and a 90% confidence level, a sample size of 67.5 was obtained, which was rounded to 68 samples per map.

These tests were used to select the optimal number of nodes for the PRM and the heuristic for the A* algorithm. For PRM, 100, 300, and 500 nodes were tested, while for A*, a heuristic value of $\sqrt{2}$ was chosen.

3. PROPOSAL

It is essential to define the research objectives, which in this case focus on determining which algorithm offers a shorter time for both path planning and the robot's path-following process. Time and distance play a crucial role in this work, as they are the key parameters used to measure the efficiency of the algorithms [10].

The programming will be carried out in MATLAB, where the map and algorithms will be implemented to generate both the paths and the robot's velocities, allowing simulations to be performed later in the CoppeliaSim environment. A black-and-white image is used, where black regions represent obstacles. The image is converted to PGM format, where each pixel is assigned a value normalized to either 1 or 0. Values of 1 represent obstacles, while 0 represents free space within the map. Once the map is represented as a matrix, the start and goal coordinates are defined within MATLAB. It is important to ensure that these locations are within free space; otherwise, the algorithm will indicate that generating a path is not possible.

Once the start and goal points are defined, the mapping process begins using the selected algorithm. For the PRM, a specific number of nodes is required. This algorithm generates nodes in random positions outside the obstacles and establishes connections between nearby nodes, provided a direct line can be drawn without intersecting obstacles. Three different node counts will be tested: 100, 300, and 500, to find a balance between computation time and the quality of the generated path.

On the other hand, the A* algorithm requires an initial heuristic value. A higher heuristic value can result in longer paths but reduces the computational cost, decreasing the processing time. In this case, a constant heuristic value of $\sqrt{2}$ will be used since the grid created by the algorithm has unit distances of 1 per side. Once the heuristic is defined, a cost

map is generated, containing the cost values between nodes and the accessible points for the path (i.e., those not occupied by obstacles).

Experimental Environment. To validate the efficiency of the algorithms, a realistic simulation environment is created in CoppeliaSim, due to its versatility in designing scenarios and its ability to consider factors such as gravity and collisions. The environment is constructed using prismatic objects that simulate the arrangement of tables in industrial work cells. The PRM and A* algorithm will be tested on five different maps—four of them based on typical work cell layouts and one designed to challenge the algorithms by incorporating an obstacle that forces the search for alternative routes. The map images are shown in Fig. 1.

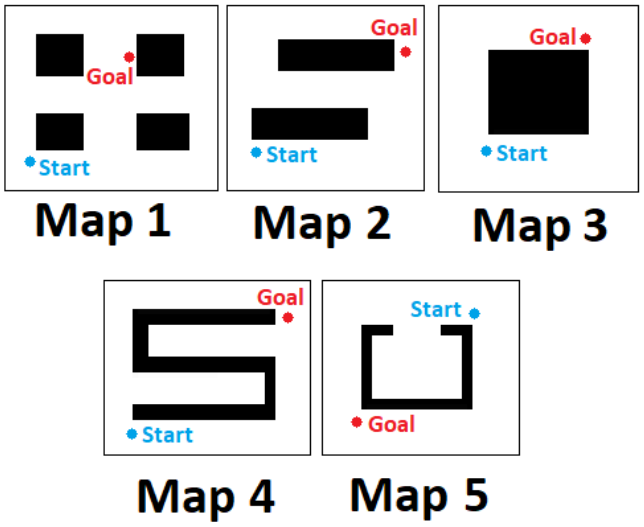

Fig.1. Maps used with their corresponding start and goal points

Variables and configurations. The variables to be evaluated are the route generation time, the time taken by the robot to follow the route, and the length of the route. To measure the generation time, the MATLAB "`cputime`" function is used, which records the time in seconds spent by the processor executing the program instructions. The time is measured by placing the function before and after the corresponding code and calculating the difference, as shown in Eq. 2.

$$GenerationTime = FinalHour - InitialHour \quad (2)$$

Subsequently, the Pure Pursuit controller from MATLAB is implemented for path-following. This controller uses a look-ahead point with a distance of 0.3 units and the points from the generated path to guide the robot. Since it is a differential drive robot, the wheel speeds are calculated using specific equations for its movement. The velocities of each wheel of the robot need to be obtained. For this, the equations shown in Eq. 3 and Eq. 4 are used.

$$Vl = \frac{v - \omega * \frac{0.3309}{2}}{0.195} \quad (3)$$

$$Vr = \frac{v + \omega * \frac{0.3309}{2}}{0.195} \quad (4)$$

The connection between MATLAB and CoppeliaSim is made through the simulator's Remote API, ensuring that the necessary files (`remApi.m`, `remoteApi.dll`, and `remoteApiProto.m`) are present in the working directory. During the path-following process, the "`plotTransforms`" function in MATLAB is used to visualize the robot's position and orientation in real time, verifying how the robot follows the planned trajectory. Once the path is completed, the simulation process is finished and the connection between MATLAB and CoppeliaSim is terminated.

To measure the time, it takes for the robot to follow the path, the "`cputime`" function is used again. In this case, it is placed at the beginning and end of the path, before terminating the connection between the programs. The same procedure used for calculating the path generation time is applied, and the difference is calculated as shown in Eq. 5.

$$FollowTime = FinalHour - InitialHour \quad (5)$$

To obtain the total distance, the "`dist`" function is used, which measures the distance between two points. Therefore, to obtain the total distance, the function is applied to the list of coordinates output by the algorithms when generating the path.

4. RESULTS

A total of 68 samples were taken for each algorithm configuration on each map, ensuring a 90% confidence level in the results obtained.

4.1. Results of the PRM

For the PRM, configurations with 100, 300, and 500 nodes were tested. As expected, when using 100 nodes, the path generation time was lower compared to the case with 500 nodes, as shown in Table 1.

However, it was observed that the results in terms of distance and path-following time were significantly better when using 500 nodes, as shown in Table 2 and Table 3. This is because a higher number of nodes provides better resultant paths with shorter lengths, at the cost of slightly increased computation time.

For this reason, the comparison between the PRM and A* algorithm was carried out with the 500-node configuration, as the additional time required for path generation with 500 nodes is compensated by the improvement in trajectory quality and efficiency in path-following, as shown in Table 4.

Generation time				
Map	Nodes	Average time	Best time	Worst time
Map 1	100	1.6026	1.3438	2.1094
Map 1	300	2.0802	1.3750	3.2500
Map 1	500	5.7313	5.1875	8.4219
Map 2	100	1.0833	0.9375	1.3281
Map 2	300	2.8365	2.5156	3.1250
Map 2	500	5.4557	5.0312	6.1406
Map 3	100	1.0539	0.9688	1.2969
Map 3	300	2.8979	2.7344	3.1406
Map 3	500	5.4190	5.0625	5.9844
Map 4	100	1.0333	0.9375	1.1719
Map 4	300	2.9156	2.6094	3.4844
Map 4	500	5.2938	4.9688	5.7500
Map 5	100	1.1208	0.9688	1.6719
Map 5	300	3.0020	2.6875	3.3906
Map 5	500	4.7676	4.3594	5.8906

Table 1. Results showing the time in seconds for the PRM to generate the path

Path length				
Map	Nodes	Average length	Best length	Worst length
Map 1	100	157.8630	125.8899	207.7826
Map 1	300	157.8626	128.0059	201.8299
Map 1	500	165.0253	129.1917	191.9098
Map 2	100	355.3018	114.7656	692.1785
Map 2	300	396.8069	295.8282	542.4253
Map 2	500	399.0446	296.0836	529.9278
Map 3	100	237.0556	184.1975	267.2522
Map 3	300	261.6338	251.6303	340.0315
Map 3	500	258.8238	182.5344	433.1237
Map 4	100	247.3331	223.7597	335.6227
Map 4	300	278.4244	223.6645	319.8824
Map 4	500	272.7864	222.9630	318.2069
Map 5	100	292.7153	180.5289	450.3051
Map 5	300	296.0954	249.5646	349.7893
Map 5	500	306.0244	247.9657	426.7358

Table 2. Results for the path length in meters of the PRM

Tracking time				
Map	Nodes	Average time	Best time	Worst time
Map 1	100	52.4859	37.1094	81.2969
Map 1	300	52.2677	35.8438	64.6562
Map 1	500	47.8250	36.6094	63.6094
Map 2	100	100.1974	75.8125	124.0312
Map 2	300	92.1401	79.7656	111.6562
Map 2	500	87.6182	78.6562	99.8438
Map 3	100	91.4242	78.0312	104.5625
Map 3	300	84.2688	74.2344	96.8750
Map 3	500	79.6611	64.5625	98.8281
Map 4	100	105.3875	80.3125	135.0938
Map 4	300	93.7198	76.6094	124.7344
Map 4	500	102.8958	95.1406	109.9062
Map 5	100	92.3167	82.0781	103.6562
Map 5	300	91.7129	76.4062	122.7969
Map 5	500	71.6553	59.6719	88.0938

Table 3. Results for the time in seconds of the PRM for path-following

Generation and tracking time		
Map	Nodes	Average time
Map 1	100	54.0886
Map 1	300	54.3479
Map 1	500	53.5563
Map 2	100	101.2807
Map 2	300	94.9766
Map 2	500	93.0740
Map 3	100	92.4781
Map 3	300	87.1667
Map 3	500	85.0801
Map 4	100	106.4208
Map 4	300	96.6354
Map 4	500	108.1896
Map 5	100	93.4375
Map 5	300	94.7148
Map 5	500	76.4229

Table 4. Results for the time in seconds of the PRM for both path generation and path-following

4.2. Results of the A* algorithm

The A* algorithm showed a deterministic characteristic, meaning that each time it is executed under the same initial conditions, it generates the same path. However, to compare it with the PRM algorithm, multiple tests were conducted to measure the path-generation time and the path-following time, thus recording a reliable database for comparison.

In Tables 5, 6, 7, and 8, the results for the times and distances of the paths are shown, allowing for the following comparisons to be made.

Generation time

Map	Average time	Best time	Worst time
Map 1	3.9351	3.7969	4.1562
Map 2	5.3469	5.1562	5.6094
Map 3	12.3187	12.0312	12.6562
Map 4	8.6344	8.2969	9.1250
Map 5	32.5448	31.5625	33.8125

Table 5. Results for the time in seconds of the A* algorithm to generate the path

Path length

Map	Length
Map 1	151.8823
Map 2	193.4558
Map 3	223.4558
Map 4	274.9706
Map 5	241.4558

Table 6. Results for the path length in meters of the A* algorithm

Tracking time

Map	Average time	Best time	Worst time
Map 1	62.3365	57.0469	71.3125
Map 2	93.5167	80.9375	110.9844
Map 3	75.7281	67.4219	89.8438
Map 4	73.8364	69.9688	81.0156
Map 5	79.9698	75.8281	84.5781

Table 7. Results for the time in seconds of the A* algorithm for path-following

Generation and tracking time

Map	Average time
Map 1	66.2716
Map 2	98.8635
Map 3	88.0469
Map 4	82.4708
Map 5	112.5146

Table 8. Results for the time in seconds of the A* algorithm for both path generation and path-following

4.3. Comparison between algorithms

Analyzing the results obtained, significant differences can be observed depending on the map:

- On simple maps (such as Map 4), the A* algorithm proved superior, generating faster and more efficient paths.
- On maps with a moderate number of obstacles (such as Maps 1, 2, and 3), both algorithms yielded similar results in terms of distance and path-following time.
- On complex maps (such as Map 5), the PRM outperformed the A* algorithm in terms of path generation time.

The superior performance of the PRM on complex maps is attributed to its ability to generate paths more quickly, especially in situations where the search space is large, and obstacles are more restrictive. On the other hand, when comparing only the path-following times of the paths generated by both algorithms, no significant differences were identified, indicating that PRM's advantage lies solely in path generation time.

As mentioned, a comparison was made between the results of the PRM with 500 nodes and the A* algorithm, as shown in Table 9.

Generation and tracking time

Map	Average time A*	Average time PRM
Map 1	66.2716	53.5563
Map 2	98.8635	93.0740
Map 3	88.0469	85.0801
Map 4	82.4708	108.1896
Map 5	112.5146	76.4229

Table 9. Generation and tracking time comparative between the A* algorithm and the PRM

The A* algorithm stood out for having the least variation in its results due to its deterministic nature. However, it was the slowest in path generation, making it less ideal for applications where planning time is critical.

5. CONCLUSIONS

The comparative analysis carried out between the PRM and A* algorithm allows for the identification of several strengths and areas for improvement. The PRM proved to be more efficient in terms of path generation time, especially in complex maps, where it significantly outperformed the A* algorithm. On the other hand, the A* algorithm stood out for its stability and consistency, as it generated more predictable results in both path distance and generation times. Despite the A* algorithm tending to generate shorter paths, the results showed that the time it takes for the robot to follow those paths is lower when using PRM with 500 nodes. This demonstrates that a shorter distance does not always result in shorter path-following time, as factors like the arrangement of the points on the path also influence the robot's performance.

Both algorithms proved to be efficient and share great versatility as one of their key strengths. Path generation is independent of the type of robot that will carry out the navigation, allowing their application on different platforms, from differential robots to Ackerman robots or omnidirectional drive robots, with only minimal adjustments required for navigation and obstacle safety margins.

In future work, it would be relevant to take the comparison of both algorithms to physical robots, evaluating their performance under real-world conditions and considering limitations imposed by hardware, such as sensors and motors. Additionally, exploring how different heuristic values affect the performance of the A* algorithm could provide valuable insights into the balance between generation time and path quality. Another interesting research avenue would be to delve into the computational and energy costs of both algorithms, which would provide significant value, especially in systems with limited resources such as small autonomous robots or drones. These explorations would help expand the applications and optimization of both algorithms in more complex and practical scenarios.

6. REFERENCES

[1] U. Orozco-Rosas, K. Picos and O. Montiel, "Hybrid Path Planning Algorithm Based on Membrane Pseudo-Bacterial Potential Field for Autonomous Mobile Robots," *IEEE Access,* vol. 7, pp. 156787 - 156803, 2019.

[2] U. Orozco-Rosas, O. Montiel and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Applied Soft Computing,* vol. 77, pp. 236-251, 2019.

[3] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor and A. Cuesta-Infante, "Mobile Robot Path Planning Using a QAPF Learning Algorithm for Known and Unknown Environments," *IEEE Access,* vol. 10, pp. 84648-84663, 2022.

[4] R. Siegwart, I. R. Nourbakhsh and D. Scaramuzza, Introduction to Autonomous Mobile Robots, Cambridge, MA: The MIT Press, 2004.

[5] U. Orozco-Rosas, K. Picos and O. Montiel, "Acceleration of Path Planning Computation Based on Evolutionary Artificial Potential Field for Non-static Environments," in *Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications*, Studies in Computational Intelligence, volume 862, Springer, 2020, pp. 271-297.

[6] S. Shekhar, R. Vaishnav, A. Kumari, S. Gautam and S. Banerjee, "Quantitative Comparison of Path Planning Algorithms in Simulated Environment," in *International Conference on Artificial Intelligence and Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA)*, 2024.

[7] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Pearson Series in Artificial Intelligence, 2021.

[8] S. M. Lavalle, Planning Algorithms, Cambridge University Press, 2006.

[9] Y. Chen, Y. Shan, L. Chen, K. Huang and D. Cao, "Optimization of Pure Pursuit Controller based on PID Controller and Low-pass Filter," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[10] U. Orozco-Rosas, K. Picos, O. Montiel and O. Castillo, "Environment Recognition for Path Generation in Autonomous Mobile Robots," in *Hybrid Intelligent Systems in Control, Pattern Recognition and Medicine*, Studies in Computational Intelligence, volume 827, Springer, 2020, pp. 273-288.