Journal of Mechatronics Engineering

Comparison of Algorithms for Path Planning Collision Avoidance

PEDRO WILSON FELIX MAGALHÃES NETO[1], JHONES WENDEL SILVA DE LIMA [1],
DAIANE FABRICIO DOS SANTOS [1], ADRIANO JOSE XAVIER SANTIAGO[1],
JOSIAS GUIMARAES BATISTA[1,2], REGIS CRISTIANO PINHEIRO MARQUES[1]

[1]Graduate Program in Telecommunications Engineering (PPGET),
Federal Institute of Ceará - IFCE, Treze de Maio Av., 2081, 60040-531, Ceará, Brazil
[2]Department of Industry, Federal Institute of Education,
Science and Technology of Ceará - IFCE, Campus Fortaleza, Fortaleza, CE, Brazil

<pedro.neto00@aluno.ifce.edu.br>, <jhones.wendel60@aluno.ifce.edu.br>,
<daiane.fabricio03@aluno.ifce.edu.br>, <adriano.jose.xavier79@aluno.ifce.edu.br>,
<josiasbatista@ifce.edu.br>, <regismarques@ifce.edu.br>

DOI: 10.21439/jme.v8i1.123

Abstract. Collision-free path planning is a critical aspect of mobile robotic navigation, with significant applications in autonomous systems and intelligent transportation. The choice of an algorithm directly influences computational efficiency and trajectory quality. This study evaluates three widely used paradigms: Artificial Potential Fields (APF), Probabilistic Roadmap Method (PRM), and Rapidly-Exploring Random Tree (RRT). Experiments were conducted in a two-dimensional environment with two and three stationary obstacles, assessing each method based on execution time and path length. The results indicate that APF is simple and fast but prone to local minima. PRM is effective for complex environments but comes with higher computational costs. RRT efficiently explores space, but often generates non-optimal trajectories. The best approach depends on environmental constraints and computational requirements. For future work, increasing scenario complexity or validating the results through real-world robotic experiments is recommended.

Keywords: path planning, collision avoidance, mobile robotics, navigation algorithms, obstacles.

1 Introduction

One of the challenges of autonomous robotics is path planning, which is responsible for defining the sequence of actions to realize the robot's travel from one origin to the endpoint, without hitting obstacles present in the environment. In this activity, a list of rules is created in order to control and guide the robot in the defined environment, that it is following the trajectory in a safe and efficient way. This problem can be solved by different computational methods; hence, it can be used in various kinds of robots like humanoids, autonomous vehicles, mobile robots, and industrial manipulators. (NASCIMENTO et al., 2020).

The use of robotic manipulators in industrial applications has expanded, significantly enhancing production system efficiency. However, certain challenges remain that can disrupt operations and lead to losses. Unexpected stoppages may occur due to various factors, including accidents and collisions between robotic manipulators, operators, and other machinery (BATISTA et al., 2023).

The use of mobile robots has numerous commercial applications in any area of industry, services and medicine. In the medical field, for example, they are used in diagnostic imaging procedures, robot-assisted surgeries and remote patient monitoring, ensuring greater precision and safety (TURNIP et al., 2025; DARIO et al., 1996; SANTOS et al., 2024). In engineering, mobile robots perform inspections and maintenance in difficult to access environments, power plants and infrastructure in inhospitable places like a bridges and underwater platforms, increasing efficiency and reducing risks for professionals. In the industrial sector, these systems

are essential for automating assembly lines, internal logistics and storage, optimizing production capacity and often reducing operating costs (ZHAO, 2023; LI; NA; GAO, 2020).

The present work aims to conduct a comparison between different collision avoidance algorithms, analyzing their potentials and limitations in real application scenarios. The objective is to identify strategies that enable the implementation of more robust and efficient systems, contributing to the advancement of autonomous robotics technologies and the expansion of their commercial applications across various sectors.

This study contributes to collision-free path planning by comparing the performance of three widely used algorithms — Artificial Potential Fields (APF), Probabilistic Roadmap Method (PRM), and Rapidly-Exploring Random Tree (RRT) — analyzing their execution time, path length, strengths, and weaknesses in different obstacle scenarios. The results highlight the trade-offs between computational efficiency, trajectory quality, and adaptability to complex environments. APF is fast and easy to implement but struggles with local minima, PRM excels in complex environments but has high preprocessing costs, and RRT efficiently explores space but produces irregular paths.

Experimental validation in a controlled 2D environment demonstrates that the best algorithm choice depends on computational constraints and scenario complexity. These insights help optimize path-planning strategies for mobile robotics, with future work suggested to include more complex environments and real-world applications.

This paper is organized as follows. Section 2 presents the theoretical foundation. Section 3 presents the materials and methods for developing the research. Section 4 presents the results obtained with the implementation of methods applied to scenarios with two and three obstacles for collision prevention. Section 5 presents discussions and comparisons of the implemented methods. Finally, Section 6 presents the research conclusions as well as suggestions for future work.

2 Theoretical Foundation

In this section, the theoretical principles of three route planning algorithms will be presented, including their advantages, disadvantages, and applications.

2.1 Artificial Potential Fields (APF)

The Artificial Potential Fields (APF) method was originally proposed by Khatib (1986) as an efficient approach for trajectory planning with obstacle avoidance.This approach maps the environment as a field of forces, where the desired target position attracts the robot and obstacles generate a repulsion. Thus, the gradient of these forces defines the trajectory of the robot, allowing it to reach its target point without collisions (JR.; CARVALHO; CONCEIçãO, 2019).

2.1.1 APF Mathematics

The mathematical formulation of the APF defines a total potential field $U(q)$, composed of an attractive field $U_{\text{att}}(q)$ and a repulsive field $U_{\text{rep}}(q)$:

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q) \tag{1}$$

The attractive field guides the robot toward the goal and is modeled as:

$$U_{\text{att}}(q) = \frac{1}{2}\zeta\rho_f^2(q) \tag{2}$$

Where $\rho_f(q)$ is the Euclidean distance between the robot's current position and the target position. The repulsive field prevents collisions and is defined as:

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2, & \text{if } \rho(q) \leq \rho_0 \\ 0, & \text{if } \rho(q) > \rho_0 \end{cases} \tag{3}$$

Where $\rho(q)$ is the distance to the nearest obstacle, ρ_0 is the obstacle's influence radius, and η defines the intensity of the repulsive force.

Although intuitive and efficient for simple environments, the APF method has some limitations, such as the occurrence of local minima, where the robot may get stuck without reaching the goal. Some proposed solutions to this problem include the introduction of rotational fields or strategies to escape local minima.

2.2 Probabilistic Roadmap Method (PRM)

The PRM (Probabilistic Roadmap) is usually a sampling-based method to manage motion planning for problems without nontrivial robot dynamics.PRM consists of two key phases: Initially, random samples (or milestones) are drawn in the free space, C_{free}, and used to form a graph (also referred to as a "roadmap"). Node collaboration is modeled by collision-free straight-line

segments, which result in a graph that represents valid paths through the state space (BERA; BHAT; GHOSE, 2019).

2.2.1 PRM Mathematics

PRM constructs a connectivity graph in the robot's free space, enabling efficient path searches (HSU; LATOMBE; KURNIAWATI, 2007).

2.2.2 Configuration Space

The configuration space C is defined as the set of all possible robot configurations. This space is partitioned into:

- **Free space** C_{free}, where the robot can move without collisions.
- **Occupied space** C_{obs}, where the robot collides with obstacles.

The goal of PRM is to find a continuous path P that connects an initial configuration q_{start} to a goal configuration q_{goal}, ensuring that $P \subseteq C_{free}$.

2.2.3 Random Sampling

PRM nodes are randomly generated in the free configuration space C_{free}, following a uniform distribution:

$$q_i \sim U(C_{free}) \tag{4}$$

where $U(C_{free})$ represents a uniform distribution over C_{free}.

2.2.4 Roadmap Graph Construction

After generating the nodes $\{q_1, q_2, \ldots, q_n\}$, we construct a graph $G = (V, E)$, where:

- V is the set of nodes (configurations).
- E is the set of edges between connectable nodes.

A connection between two nodes q_i and q_j is established if the direct path between them is collision-free:

$$(q_i, q_j) \in E \quad \text{if} \quad \text{StraightLine}(q_i, q_j) \subseteq C_{free} \tag{5}$$

In some PRM variants, connections may be based on the k-nearest neighbors:

$$E = \{(q_i, q_j) \mid d(q_i, q_j) \leq r\} \tag{6}$$

Where $d(q_i, q_j)$ is the Euclidean distance between nodes and r is a connection radius.

2.2.5 Path Search in the Graph

Once the graph is constructed, search algorithms are used to find a path between q_{start} and q_{goal}. The cost of each edge is given by:

$$C(q_i, q_j) = \|q_i - q_j\| \tag{7}$$

Commonly used search algorithms include:

- **Dijkstra**, which finds the shortest path in the graph (KARAMAN; FRAZZOLI, 2011).
- **A***, which uses heuristics to optimize the search.

2.2.6 Probabilistic Completeness

PRM is probabilistically complete, meaning that the probability of finding a feasible path, if one exists, approaches 1 as the number of samples n increases:

$$\lim_{n \to \infty} P(\text{PRM finds a path}) = 1 \tag{8}$$

2.3 Rapidly-Exploring Random Trees (RRT)

RRTs are trees that grow incrementally and randomly within a configuration space, expanding preferentially into unexplored regions. The algorithm starts from an initial point and, at each iteration, generates a random point q_{rand} in the configuration space. It then identifies the nearest node q_{near} within the tree and creates a new connection q_{new} with a limited step size Δq towards q_{rand}, ensuring that the new position remains in a collision-free space (SOUZA; CHAIMOWICZ, 2007).

2.3.1 Algorithm Mathematics

Mathematically, the tree expansion can be described as:

$$q_{new} = q_{near} + \Delta q \cdot \frac{q_{rand} - q_{near}}{|q_{rand} - q_{near}|} \tag{9}$$

Where:

- q_{new} - Newly generated node in the tree.
- q_{near} - Nearest node to q_{rand} in the tree.
- q_{rand} - Randomly sampled point in the configuration space.
- Δq - Maximum step size allowed for tree expansion.
- $q_{\text{rand}} - q_{\text{near}}$ - Vector pointing from q_{near} to q_{rand}.

- $|q_{\text{rand}} - q_{\text{near}}|$ - Norm (Euclidean distance) between q_{rand} and q_{near}.
- $\frac{q_{\text{rand}} - q_{\text{near}}}{|q_{\text{rand}} - q_{\text{near}}|}$ - Unit vector in the direction of q_{rand}, ensuring movement always occurs in that direction.

According to LaValle and Kuffner (2000), RRTs possess several important properties:

- They rapidly explore high-dimensional spaces;
- They are probabilistically complete, meaning that if a solution exists, the RRT will find it with probability 1 as execution time increases;
- They are efficient for complex path-planning problems.

2.3.2 Applications

Working in robotics, RRTs are often used to guide mobile robots, allowing new safe and optimal trajectories to be generated on the fly. It is especially beneficial in an environment that is dynamic, in which the occurrence of obstacles may vary over time (LAVALLE, 1998).

In the field of video games, the authors Souza e Chaimowicz (2007) showed that RRTs are quite efficient to carry out NPC (Non-Player Character) navigation. The technique was tested using the game Quake II to compare traditional graph-based methods against their approach, yielding smoother and more spontaneous trajectories for NPCs. Such behaviors make NPCs more life-like and challenging, providing a better player experience.

3 Materials and Methods

In this section, the most relevant topics for understanding the work will be addressed, such as the material, which will be referred to as the Development Environment, and the methods of the models discussed. For the implementation of the algorithms, the Python programming language was used along with the following libraries:

- *NumPy* for mathematical calculations and vector manipulation.
- *Matplotlib* for visualizing environments and trajectories.
- *Random* for generating random numbers.
- *Itertools* for iterating over all possible parameter combinations.
- *Deque from collections* Deque is a data structure that allows adding and removing elements from both ends (front and back) efficiently.
- *Time* for measuring the execution time of each algorithm.

3.1 Development Environment - Colab

Python was chosen as the programming language due to its extensive use in the Machine Learning industry, in addition to being the default language of Google Colaboratory, which served as the Integrated Development Environment (IDE) for this scientific production. An IDE is a software application used for developing programs through coding, providing specialized tools for this purpose.

Also known as Google Colab, Google Colaboratory is a free web-based development environment that allows users to write and execute Python code directly in the browser. It is extremely useful for machine learning, data analysis, and programming education, offering access to powerful computing resources without requiring local setup. Google Colab allows Python code execution in notebooks, which are documents that can contain executable code, text, equations, visualizations, and narrative explanations. These notebooks follow a format similar to Jupyter Notebook, ensuring full compatibility between both IDEs.

Google Colab is integrated with Google Drive, enabling users to save and share their notebooks easily. Notebooks can be stored directly in Google Drive, facilitating collaboration and remote access.

Just like other Google products, Colab allows real-time collaboration. Multiple users can edit and execute the same notebook simultaneously, facilitating teamwork.

Google Colab has several advantages:

- **Accessibility**: Available for free to anyone with a Google account.
- **Portability**: Being entirely web-based, users can access it from anywhere and on any device, regardless of the operating system, requiring only an internet connection.
- **Computational Resources**: Free access to GPUs and TPUs, which is advantageous for tasks requi-

ring high processing power. While free GPU access is limited, it provides computational resources through chipsets such as Nvidia Tesla K80, T4, P4, and P100, which can significantly accelerate machine learning and deep learning tasks.

3.2 Methods

Three path-planning algorithms and one parameter optimization algorithm were implemented and compared:

- **Artificial Potential Fields - APF** : Based on artificial potential fields, where attractors and repulsors guide the agent toward the destination while avoiding obstacles.
- **Probabilistic Roadmap - PRM**: Constructs a graph based on random sampling of the configuration space, connecting feasible points to generate paths.
- **Rapidly-exploring Random Tree - RRT**: Expands a tree randomly until a valid path from the initial point to the goal is found, avoiding obstacles.
- **Grid Search**: A systematic parameter optimization technique.

3.2.1 Artificial Potential Fields (APF)

In the APF method, the robot is influenced by two forces:

- **Attractive Force:** Pulls the robot toward the goal, proportional to the distance.
- **Repulsive Force:** Repels the robot from obstacles to prevent collisions.

The robot's position is iteratively updated based on the vector sum of these forces. Adjustments to parameters such as the repulsive constant and repulsion area are essential to ensure the robot efficiently avoids obstacles.

3.2.2 Grid Search

Grid Search is a hyperparameter optimization technique that performs an exhaustive search over a predefined space of parameter values. In other words, it constructs a grid with all possible combinations of hyperparameters and evaluates the model's performance for each combination, aiming to identify the configuration that maximizes efficiency. Despite its simplicity and systematic approach, Grid Search can become computationally expensive when the number of hyperparameters and their candidate values is large.

According to Bergstra e Bengio (2012), although random search methods can sometimes be more computationally efficient in certain scenarios, Grid Search remains widely used as a benchmark for comparing other optimization techniques. Furthermore, James et al. (2013) highlight that this approach is fundamental for model validation, serving as a reference point to assess performance across different parameter configurations.

All obstacles in the environment were modeled as static square regions. Each obstacle was defined by its boundary coordinates in the format `(x_min, y_min, x_max, y_max)`, following the coordinate system used in Python.

To fine-tune the parameters of the APF method, we performed an exhaustive *Grid Search* over all possible combinations of k_{att}, k_{rep}, repulsion area, step size, and maximum number of iterations. For each set of parameters, the algorithm attempted to generate a collision-free path from the start point to the goal. If a valid path was found, its length was calculated and compared to the best result so far. The configuration yielding the shortest successful path was saved as the optimal one. Finally, this best parameter set was used to run a final simulation, which included visualizing the resulting path and the corresponding potential field in 3D.

3.2.3 Probabilistic Roadmap Method (PRM)

PRM uses a probabilistic sampling approach to construct a graph representing the free space. The main steps are:

- **Node Sampling:** Generating random points in the environment while avoiding collision areas.
- **Node Connection:** Connecting nearby nodes (within a predefined distance) while ensuring no collisions along the connecting segments.
- **Path Search:** Using a search algorithm (such as BFS) to find a route from the starting point to the goal.

The PRM methodology employed a bounded space (limit_space), considering obstacles (obstacles). A total of 300 nodes (num_ nos=300) were generated for graph

construction, with a maximum connection distance of 20 (distancia_conexao=20) arbitrary units. The execution start time was recorded using the python function time.time(), allowing measurement of the algorithm's efficiency.

3.2.4 Rapidly-Exploring Random Tree (RRT)

The RRT method builds an exploratory tree that expands through the space:

- **Random Point Generation:** In each iteration, a random point is generated in the environment.
- **Tree Expansion:** The tree expands from the nearest node to the generated point using a predefined step size.
- **Connection to Goal:** When a new node approaches the goal, the path is reconstructed by connecting the initial point to the destination.

The RRT methodology was used for path planning, considering a bounded space (limit_space) and the presence of obstacles (obstacles). The tree grows in steps of size 5 (step_size=5), with a maximum limit of 1000 iterations (iterations=1000). The execution start time was recorded using time.time(), allowing performance evaluation of the algorithm.

3.3 Simulation Scenario

The workspace for the three algorithms is a two-dimensional environment defined within a rectangular area ranging from coordinates $(0, 0)$ to $(100, 100)$.The environment includes three square obstacles, each defined by their lower-left and upper-right corner coordinates using the Python format `(x_min, y_min, x_max, y_max)`. These rectangular regions represent impassable areas that the planning algorithms must navigate around. The environment boundaries range from 0 to 100 in both the x and y directions. All spatial measurements—including coordinates and path lengths expressed in arbitrary units, as defined by the simulation grid.

Within this space we performed two scenarios, where in the first two fixed obstacles were placed, and in the second three fixed obstacles:

(i) Scenario 1

- **Obstacle 1**: Defined by the area from $(30, 30)$ to $(60, 60)$.
- **Obstacle 2**: Defined by the area from $(70, 10)$ to $(80, 30)$.

(ii) Scenario 2

- **Obstacle 1**: Defined by the area from $(30, 30)$ to $(60, 60)$.
- **Obstacle 2**: Defined by the area from $(70, 10)$ to $(80, 30)$.
- **Obstacle 3**: Defined by the area from $(50, 70)$ to $(55, 85)$.

The choice of sampling parameters for both PRM and RRT algorithms was guided by recommendations from their original formulations. For the PRM, the number of sampled nodes and the maximum connection distance were selected based on the insights provided by Kavraki et al. (1996), who demonstrated that increasing the number of samples enhances roadmap connectivity and the probability of finding a feasible path, though at the cost of higher computational complexity. For the RRT, the expansion step size was chosen in accordance with LaValle (1998), who emphasized that smaller step sizes lead to finer resolution and better obstacle avoidance in cluttered environments, while larger step sizes can speed up exploration but reduce path precision. These parameters were further refined through empirical sensitivity analysis to balance performance and computation time.

All methods were executed under the same conditions to ensure result comparability, which will be presented in the following section.

4 Results

This section presents the results of the collision avoidance methods evaluated in this study—APF, PRM, and RRT—comparing their performance based on execution time, path length, and trajectory efficiency in different obstacle scenarios. Visual representations of the generated paths illustrate how each algorithm navigates the environment, highlighting their strengths and limitations. A comparative table summarizes key trade-offs, providing insights into the suitability of each method for different applications.

Before initializing the route planning code, we searched for the best parameters for the APF algorithm by performing a Grid Search with all its routing variables.

The search was conducted with the following values:

- **k_att:** [0.5, 1.0, 1.5]
- **k_rep:** [50.0, 100.0, 150.0, 200.0, 500.0, 750.0]
- **repulsion_area:** [25.0, 100.0, 150.0, 250.0, 500.0, 1000.0]
- **step_size:** [0.5, 1.0, 1.5]
- **iterations:** [1000, 2000, 3000]

After running the Grid Search algorithm, the best route was found with the following variable values:

- **k_att:** 1.0
- **k_rep:** 500.0
- **repulsion_area:** 150.0
- **step_size:** 0.5
- **iterations:** 1000

For this combination, the generated path had a total length of 128.85 arbitrary units. This configuration proved to be the most effective at avoiding collisions and minimizing the effects of local minima.

4.1 Comparison of Collision-Free Route Planning Methods Results

The figures presented show the performance of three different route planning methods for mobile robots: Artificial Potential Fields (APF), Probabilistic Roadmap Method (PRM), and Rapidly-Exploring Random Tree (RRT). Each image illustrates how the algorithm plots a path from the starting point (green) to the goal (red), avoiding obstacles (red areas). Next, we discuss the main characteristics of each method and present a table summarizing their strengths, weaknesses, advantages, and disadvantages.

4.1.1 Path Found by the APF

Figure 1 and 2 illustrates a relatively smooth path, although it is observed that the robot approaches the largest obstacle significantly. To mitigate this behavior, we introduced a variable called "margin"in the algorithm, which defines a safety distance relative to objects, thus reducing the probability of local minima occurrence.This parameter was empirically determined through practical testing and direct observation of the system's behavior. In the absence of a formal mathematical formulation, the margin value was manually tuned to ensure the agents avoided collisions while maintaining the smallest value that still guaranteed safe navigation. In more complex scenarios, the APF method may converge to local minima or struggle in navigating tight spaces near obstacles. Despite being a straightforward and simple method to implement, its effectiveness requires fine-tuning of parameters (e.g., repulsion strength, repulsion area) to avoid collisions and local minima. For this reason, we developed code to perform a Grid Search and locate the best parameters.

4.1.2 Path Found by the PRM

Figures 3 and 4 presents the paths generated by the PRM method, where it projects several nodes (in blue) and connections (cyan edges). The path found (in green) safely circumvents obstacles and provides a clear route from the start to the goal. This method is advantageous in complex spaces, as generating a dense graph increases the chances of finding viable paths. Conversely, it may require a longer pre-processing time (to sample nodes and check for collisions) and more memory when there are many nodes.

4.1.3 Path Found by the RRT

Figures 5 and 6, we see a "tree" with blue branches spreading throughout the environment. The final path (in green) follows a relatively efficient route, safely circumventing obstacles. RRT is simple to implement and quickly explores large regions of space, but it can generate more irregular paths and, in some cases, require post-processing(for example, path smoothing).

4.1.4 Path Measurement of the Three Algorithms

The results at Table 1 and Table 2 indicates that each method has specific characteristics and is better suited to certain types of environments or application requirements.

While APF offers simplicity and speed, PRM is robust in complex scenarios and RRT stands out for its ability to quickly explore high dimensions. The ideal choice depends on factors such as the complexity of the environment, available computational resources and the need for smoother or faster paths to be found.

5 Discussions

The comparative analysis of the three path-planning algorithms—Artificial Potential Fields (APF), Probabilistic Roadmap Method (PRM), and Rapidly-Exploring

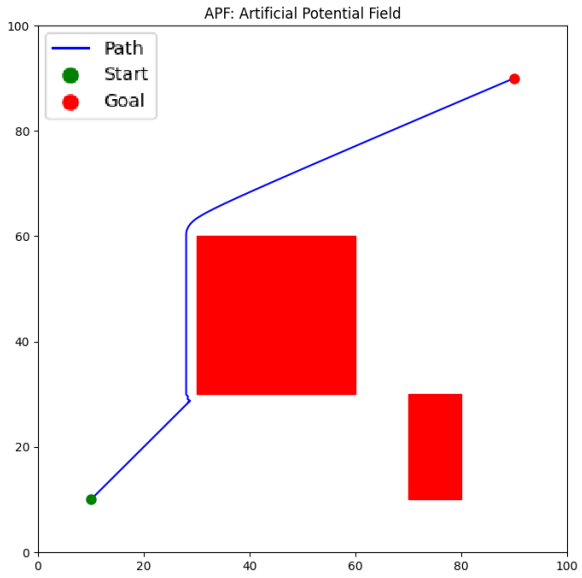

Figure 1: Path APF with 2 obstacles.

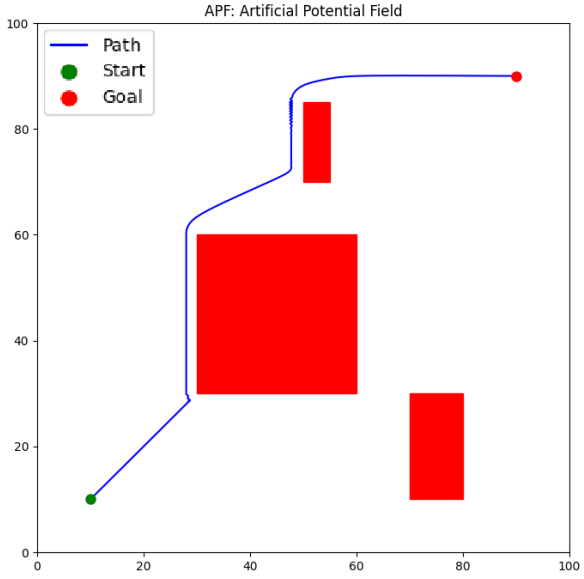

Figure 2: Path APF with 3 obstacles.

Model	Time (s)	Length (arbitrary units)
APF	0.01	128.85
PRM	0.63	132.45
RRT	0.04	155.00

Table 1: Comparison of route planning models for 2-obstacle space in terms of execution time and path length.

Model	Time (s)	Length (arbitrary units)
APF	0.01	144.89
PRM	0.66	128.55
RRT	0.03	143.60

Table 2: Comparison of route planning models for 3-obstacle space in terms of execution time and path length.

Random Tree (RRT)—reveals distinct advantages and limitations based on execution time, path length, and computational efficiency.

5.1 Performance in Terms of Execution Time

The results show that APF is the fastest method, with an execution time of 0.01 seconds in both 2-obstacle and 3-obstacle environments. This makes APF highly efficient for real-time applications where speed is a priority. However, its primary limitation is the tendency to get stuck in local minima, which can prevent the robot from reaching its target in more complex environments.

PRM, on the other hand, has the longest execution time (0.66s for 3 obstacles and 0.63s for 2 obstacles) due to its preprocessing phase, where it constructs a probabilistic roadmap by sampling free-space nodes and checking for collisions. Although computationally expensive, PRM is well-suited for complex environments as it provides multiple valid paths, making it useful for scenarios where precomputed graphs can be reused.

RRT achieves a balance between execution time and adaptability, taking slightly longer than APF (0.03s for 3 obstacles and 0.04s for 2 obstacles) but still significantly faster than PRM. This makes it a viable choice for environments requiring quick path generation, especially when working with high-dimensional spaces.

Quantitative comparisons were made based on execution time and path length, as shown in Tables 1 and 2.

When comparing the 2-obstacle and 3-obstacle scenarios:

- **APF** maintained a constant execution time of 0.01 seconds. However, its path length increased from 128.85 to 144.89 units, representing a **12.46% increase**, indicating sensitivity to obstacle density.
- **PRM** showed a modest **4.76% increase** in execution time (from 0.63 to 0.66 seconds), while the path length decreased from 132.45 to 128.55 units

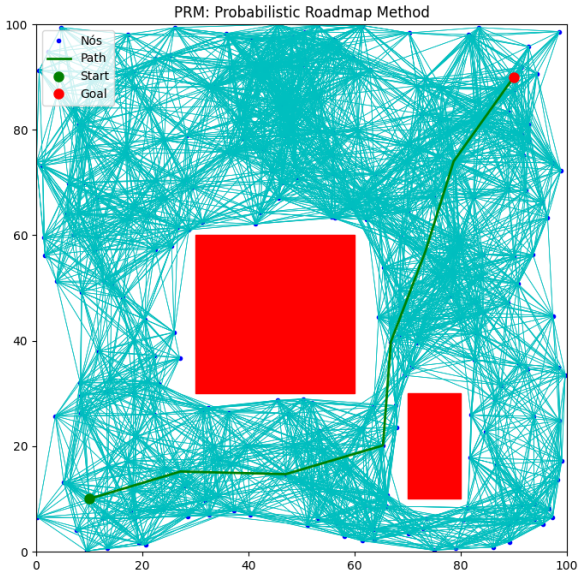

Figure 3: Path PRM with 2 obstacles.

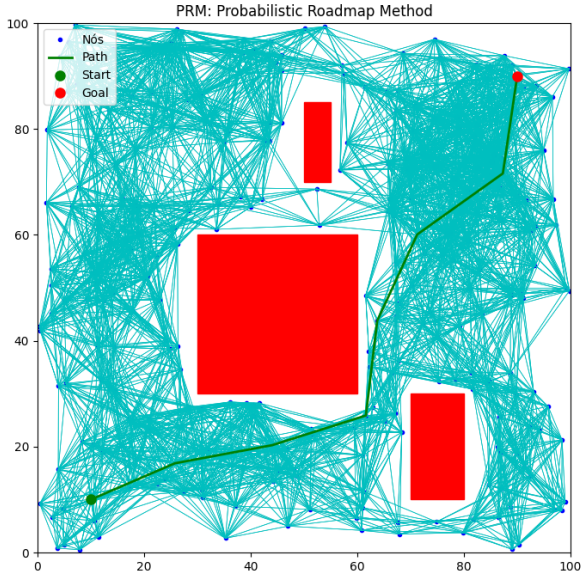

Figure 4: Path PRM with 3 obstacles.

(a **2.95% reduction**), reflecting its robustness and adaptability.

- **RRT** exhibited a **25% decrease** in execution time (from 0.04 to 0.03 seconds) and a **7.35% reduction** in path length (from 155.00 to 143.60 units), possibly benefiting from improved sampling in denser obstacle fields.

These quantitative differences highlight the diverse behavior and efficiency trade-offs of each algorithm under varying levels of environmental complexity.

5.2 Path Length and Trajectory Efficiency

The path length is another crucial factor in evaluating the efficiency of each algorithm. PRM consistently produced the shortest paths (128.55 arbitrary units for 3 obstacles and 132.45 arbitrary units for 2 obstacles), indicating that it finds more optimized routes when given enough preprocessing time. This makes PRM preferable in applications where path efficiency is more critical than immediate response time.

APF generated longer paths compared to PRM (144.89 arbitrary units for 3 obstacles and 128.85 arbitrary units for 2 obstacles), suggesting that while it is computationally efficient, it does not always produce the most direct trajectory. Additionally, the presence of local minima can force the robot to take unnecessary detours, further increasing the total travel distance.

RRT, while being an effective exploration-based approach, produced the longest paths (143.60 arbitrary units for 3 obstacles and 155.00 arbitrary units for 2 obstacles). This is due to the randomized nature of its tree expansion, which does not inherently optimize the trajectory and often results in non-smooth, irregular paths. In practical applications, this necessitates post-processing techniques such as path smoothing or additional optimization to improve trajectory efficiency.

5.3 Suitability of Each Algorithm for Different Environments

Each algorithm demonstrates strengths and weaknesses depending on the application: APF is best suited for simple, well-structured environments with sparse obstacles, where its speed and ease of implementation outweigh its limitations. However, it may require fine-tuning of parameters to avoid convergence to local minima. PRM is highly effective for complex environments where multiple paths need to be considered, making it valuable for robotic applications that allow for precomputed roadmaps. However, its high computational cost makes it less ideal for real-time navigation. RRT is advantageous for exploring high-dimensional spaces and large free regions, where its ability to rapidly generate feasible paths is beneficial. However, its non-optimal paths often require post-processing to improve efficiency.

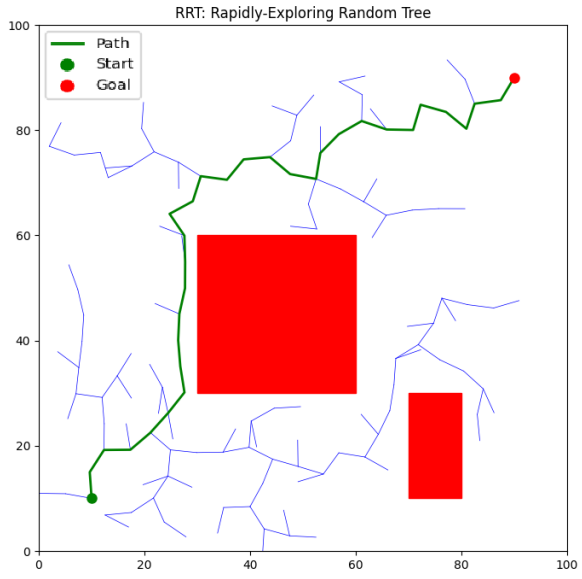

Figure 5: Path RRT with 2 obstacles.

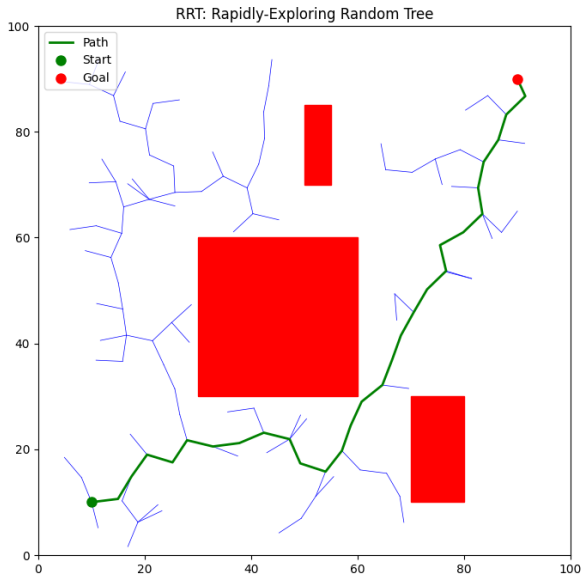

Figure 6: Path RRT with 3 obstacles.

Table 3 presents a resume qualitative comparison of collision-free route planning methods.

6 Conclusion

The evaluated algorithms demonstrated good performance in the proposed task. It was observed that the execution times of the APF and RRT methods were similar, while PRM required more time due to the computational cost of its preprocessing phase. Regarding the length of the generated paths, APF and PRM produced similar values in the scenario with two objects, whilst RRT resulted in a longer trajectory, suggesting the need for post-processing refinement. However, in the scenario with three objects, APF and RRT generated similar paths, both longer than PRM, showing a variation in performance depending on the complexity of the environment.

APF was an efficient solution for thin-dense environments because APF is a simple and fast method. But some extra care has to be taken in order to avoid local minima, which in turn may affect the quality of the solution. On the other hand, PRM demonstrated advantages in more complex scenarios, as its node sampling and roadmap construction allow for diverse path possibilities. However, this benefit comes at the cost of increased computational overhead in the preprocessing stage. The RRT algorithm, in turn, stood out as a suitable choice for high-dimensional spaces or large free regions, as its random tree expansion effectively explores the environment.

The objective of this work was achieved by comparing path planning algorithms and analyze their characteristics, pros and cons in different scenarios. Quantitative results demonstrate that:

- **APF**, while extremely fast, generates longer paths in denser environments, with a **12.46% increase** in path length when moving from 2 to 3 obstacles.
- **PRM** provides a balance between time and path efficiency, showing a **4.76% increase in execution time** but a **2.95% decrease** in path length.
- **RRT** improved in both metrics with additional obstacles, with a **25% reduction in execution time** and a **7.35% shorter path**, indicating high adaptability in complex scenarios.

These insights guide the choice of algorithm depending on application requirements such as speed, smoothness, or obstacle complexity.

An interesting direction for the future research, would be expand the complexity or even to perform experiments in practice to see if the results apply to real applications.

Future research should focus on developing hybrid path-planning approaches that combine the strengths of different algorithms, optimizing trajectory efficiency

Table 3: Comparison of collision-free route planning methods.

Model	Strengths	Weaknesses	Advantages	Disadvantages
APF	Simple implementation; Fast execution; Good for sparsely obstructed environments	Gets stuck in local minima; Sensitive to parameter calibration	Smooth trajectories when well-tuned; Easy to understand and debug	Requires fine-tuning (k_rep, repulsion_area); May fail in complex scenarios
PRM	Efficient in complex environments; Can reuse the graph for multiple queries	High pre-processing cost; High memory consumption if many nodes are sampled	Captures multiple paths in a single graph; Versatile in different scenarios	Performance depends on the number of sampled nodes; Requires collision checking between node pairs
RRT	Quickly explores large areas; Easy to implement	Irregular paths; May require post-processing for smoothing	Well-suited for high-dimensional spaces; Capable of finding solutions even in complex scenarios	Routes may be very "tortuous"; Sensitive to step size

and adaptability. Real-world validation on physical robotic platforms is essential to assess performance under dynamic conditions, while metaheuristic optimization and machine learning integration can enhance adaptability and path efficiency. Expanding the study to multi-robot systems and high-dimensional environments, such as autonomous vehicles and drones, would improve scalability. Additionally, exploring energy-efficient path planning can optimize robotic navigation for battery-powered systems. These advancements will contribute to more robust, efficient, and intelligent autonomous navigation systems across diverse applications.

References

BATISTA, J. G.; RAMALHO, G. L. B.; TORRES, M. A.; OLIVEIRA, A. L.; FERREIRA, D. S. Collision avoidance for a selective compliance assembly robot arm manipulator using topological path planning. **Applied Sciences**, v. 13, n. 21, 2023. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/13/21/11642>.

BERA, T.; BHAT, M. S.; GHOSE, D. **Analysis of Obstacle based Probabilistic RoadMap Method using Geometric Probability**. 2019. Disponível em: <https://arxiv.org/abs/1906.00136>.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **Journal of machine learning research**, v. 13, n. 2, 2012.

DARIO, P.; GUGLIELMELLI, E.; ALLOTTA, B.; CARROZZA, M. C. Robotics for medical applications. **IEEE Robotics & Automation Magazine**, IEEE, v. 3, n. 3, p. 44–56, 1996.

HSU, D.; LATOMBE, J.-C.; KURNIAWATI, H. On the probabilistic foundations of probabilistic roadmap planning. In: THRUN, S.; BROOKS, R.; DURRANT-WHYTE, H. (Ed.). **Robotics Research**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 83–97. ISBN 978-3-540-48113-3.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. et al. **An introduction to statistical learning**. [S.l.]: Springer, 2013. v. 112.

JR., U. d. M. P.; CARVALHO, M. P.; CONCEIçãO, A. G. S. Campos potenciais artificiais aplicado ao planejamento de trajetórias do braço robótico jaco. In: **Congresso Brasileiro de Automática - CBA**. [S.l.: s.n.], 2019.

KARAMAN, S.; FRAZZOLI, E. **Sampling-based Algorithms for Optimal Motion Planning**. 2011. Disponível em: <https://arxiv.org/abs/1105.1186>.

KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. **The international journal of robotics research**, Sage Publications Sage CA: Thousand Oaks, CA, v. 5, n. 1, p. 90–98, 1986.

LAVALLE, S. M. Rapidly-exploring random trees : a new tool for path planning. **The annual research report**, 1998. Disponível em: <https://api.semanticscholar.org/CorpusID:14744621>.

LI, Y.; NA, J.; GAO, G. Dynamic modeling and analysis for 6-dof industrial robots. In: IEEE. **2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)**. [S.l.], 2020. p. 247–252.

NASCIMENTO, L.; SANTOS, V. Gaboardi dos; PEREIRA, D.; FERNANDES, D.; ALSINA, P. Planejamento de caminho para sistemas robóticos autônomos. In: ____. [S.l.: s.n.], 2020. p. 69–89. ISBN 978-85-7669-489-2.

SANTOS, L. A.; MAIA, S. M.; BATISTA, J. G.; SOEIRO, R. R.; MACEDO, A. R. L. de; ALBUQUERQUE, V. H. C. de. Recent advances in swarm mobile robotics. **Journal of Mechatronics Engineering**, v. 7, n. 1, p. 1–24, 2024.

SOUZA, S. A. de; CHAIMOWICZ, L. Utilizando rapidly-exploring random trees (rrts) para o planejamento de caminhos em jogos. In: **VI Brazilian Symposium on Computer Games and Digital Entertainment**. [S.l.: s.n.], 2007. p. 170.

TURNIP, A.; FARIDHAN, M. A.; WIBAWA, B. M.; ANGGRIANI, N. Autonomous medical robot trajectory planning with local planner time elastic band algorithm. **Electronics**, v. 14, n. 1, 2025. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/14/1/183>.

ZHAO, T. C. X. Autonomous mobile robots in manufacturing operations. **19th International Conference on Automation Science and Engineering**, IEEE, 2023. Disponível em: <https://ieeexplore.ieee.org/document/10260631>.