2023 20th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE) | 979-8-3503-0676-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCE60043.2023.10332846

A Software Tool for Teaching the Homotopy-Based Path Planning Method for Mobile Robot Applications

G. Diaz-Arango, H. Vazquez-Leal, Z. Ortiz-Viveros
Facultad de Instrumentacion Electronica,
Universidad Veracruzana,
Cto. Gonzalo Aguirre Beltran S/N,
91000 Xalapa, Veracruz, Mexico
Email: gu.diaz.gd@gmail.com

L. Hernandez-Martinez, A. Jaramillo Alvarado
Electronics Department,
National Institute for Astrophysics, Optics and Electronics (INAOE)
Luis Enrique Erro 1,
72840 Puebla, Mexico
Email:luish@inaoep.mx

Y. Preciado-Quintero, E. Duran-Toyo, D. Sanchez-Casarrubias
Escuela de Ingenierias,
Universidad Interamericana A. C.,
Lateral Sur de la Atlixcáyotl 7007,
72828 Puebla, Mexico

***Abstract*—In this work, a software tool is presented, developed for easy implementation in academic projects and mobile robotics teaching using the collision-free path planner based on continuation homotopy (HPPM). A concise explanation of the HPPM formulation is offered, along with a discussion on the key parameters essential for its implementation. A primary objective of this study is to ensure that the HPPM is accessible to students and researchers, thereby encouraging its adoption in academic and research contexts. An entire section is devoted to introducing the precompiled planning software tool, detailing how it can be integrated as a module in Python scripts. Following this, three case studies are showcased, where the tool is employed to solve real environment maps containing circular obstacles. In the final case study, the practical application of the planner, resulting in collision-free paths for a Parallax S2 robot, is demonstrated. The study concludes by addressing several crucial considerations for implementation, followed by concluding remarks and directions for future research.**

***Index Terms*—Mobile robot, Path planning, Homotopy continuation methods, Real time planner, Image processing.**

I. Introduction

Mobile robotics is identified as a field of robotics that focuses on the design, development, and operation of autonomously moving agents in static and dynamic environments [1]. In recent years, a significant rise in the popularity of this branch of robotics has been observed, driven by technological advancements in areas such as perception, data processing, and trajectory planning [2]. Within a mobile robot, the navigation module is regarded as one of the most crucial components, allowing the robot to be moved autonomously and efficiently in its environment. Trajectory planners, which are supported by information from external sensors such as cameras, infrared sensors, and ultrasonic sensors, are seen as vital in the navigation module. This information is processed to create a manageable abstract representation for onboard computers [3], [4]. Multiple trajectory planning algorithms are utilized in mobile robotics, including Rapidly-exploring Random Tree (RRT), Probabilistic Roadmap (PRM), artificial potential fields, and homotopy-based planning, among others. Each algorithm possesses its own advantages and disadvantages, and the selection is determined by the robot's environment and objectives [5], [6]. In the last decade, the Homotopic Path Planning Method (HPPM) has been proven to be effective in solving planning problems that are not addressed by other approaches [7]–[9]. Furthermore, this method can be implemented on resource-limited devices such as microcontrollers and FPGAs [10]. While the principle of operation for HPPM is seen as relatively simple, its implementation can be considered challenging due to the involved mathematical complexity. Despite these challenges, homotopy-based planning is regarded as a valuable tool for teaching mobile robotics at various educational levels, ranging from secondary school to university. In this work, a trajectory planning software tool for terrestrial mobile robots is introduced. The tool has been precompiled in C++ to facilitate its use, especially in educational robotics projects, ensuring seamless integration. In Section II, the fundamental principles of homotopy-based trajectory planning are outlined. Section III introduces the precompiled HPPM planner and discusses the adjustable parameters that dictate its performance. Furthermore, the integration of this tool into Python scripts for a user-friendly application is demonstrated. Several case studies, showcasing the tool's utility in both simulated and real-world environments, are presented in Section IV. Finally, in Section V, the capabilities of the tool are discussed, and potential future directions for this project are proposed.

II. Homotopic Path planning Method

The HPPM is a single-query planner capable of obtaining a collision-free path by utilizing a mathematical model derived from the robot model and its configuration space [10], [11]. This planner is founded on homotopy continuation methods (HCM), typically employed to find multiple solutions to Nonlinear Algebraic Equation Systems (NAES) in the following form:

$$F(X) = 0: \quad \mathbb{R}^n \longrightarrow \mathbb{R}^n. \tag{1}$$

The homotopy operates by deforming $F(X)$ through the addition of a function $G(X)$ and a homotopy parameter λ, resulting in the following expression:

$$H(X, \lambda) = \lambda F(X) + (1 - \lambda)G(X) = 0, \tag{2}$$

where $H(X, \lambda) : R^{n+1} \longrightarrow R^n)$, with $(X \in R^n)$ and $(\lambda \in [0, 1])$. Additionally, $(G(X))$ is a function with a known solution [9], [11]. The homotopy formulation satisfies the following constraints:

- If $H(X, 0) = 0$, the trivial or known solution is obtained.
- If $H(X, 1) = F(X)$, a solution of the original system is found.
- The homotopy curve consists of the set of intersection points between the equations in the homotopy system (2). That is, $H(X, \lambda) = 0$ for $(X, \lambda) \in H^{-1}(0)$, where $H^{-1}(0)$ represents the set of intersections. This set is commonly denoted as γ [11], [12]. Moreover, all solutions of the original system $F(X) = 0$ are contained in $H^{-1}(0)$, and these solutions are found during the continuous deformation process until $\lambda = 1$ is reached.

In homotopy applications, the primary objective is typically to determine the maximum number of system solutions. However, in path planning, the main task is centered on tracking a continuous homotopic curve from an initial to a final point, which correspond to the initial and final robot configurations, respectively [11]. In this context, the HPPM process is based on a system of equations that models the configuration space, and Newton's homotopy formulation is utilized to derive a collision-free path [11]. For the Newton's homotopy (employed in this work), the auxiliary function is defined as $G(X) = F(X) - F(X_0)$, where X_0 is the known starting point (initial configuration of the robot in free space). In this system, the curve is represented as a sequence of points that describe continuous motion from a starting point to a goal point while avoiding collisions with obstacles [8], [10], [11]. When the mobile robot with 2-D displacement is assumed to be holonomic and its representation can be simplified to a point robot, the configuration space is then reduced in the following manner:

$$f_1(x, y) = L_1(x, y) = 0, \tag{3}$$

$$f_2(x, y) = L_2(x, y) + g(x, y) = 0, \tag{4}$$

where, $g(x, y)$ models the obstacles on the map through singular mathematical regions around and inside them. $L_1(x, y)$ and $L_2(x, y)$ are two auxiliary straight lines that intersect at the goal point. They are used solely to generate a simple solution for the system at the goal point. These lines can be represented as follows:

$$L_i(x, y) = -y + m_i x + b_i = 0; \ i = 1, 2, \tag{5}$$

After applying Newton's homotopy to (3) and (4), the system is transformed as follows:

$$H = \begin{cases} H_1(f_1(x, y), \lambda) = f_1(x, y) - (1 - \lambda)f_1(x_0, y_0) = 0, \\ H_2(f_2(x, y), \lambda) = f_2(x, y) - (1 - \lambda)f_2(x_0, y_0) = 0, \end{cases} \tag{6}$$

The initial point is represented by (x_0, y_0). Obstacles in the configuration space are represented using circumferences, ellipsoids, or other semi-algebraic forms [8], [10], [11]. These obstacles are integrated into the HPPM formulation using the following expression:

$$W_O(x, y) = \sum_{i=1}^{i=k} \frac{p_{O,i}}{O_i(x, y)}, \tag{7}$$

In this expression, $W_O(x, y)$ is the term that models the effect of all obstacles in the environment, $O_i(x, y)$ describes the shape of each obstacle, and $p_{O,i}$ is the repulsion parameter for the i-th obstacle [10]. Consequently, the term $g(x, y)$ in 4 is defined as:

$$g(x, y) = W_O(x, y)N(x, y), \tag{8}$$

where $N(x, y) = W_O(x_{goal}, y_{goal})$ is an expression that nullifies the effect of obstacles at the goal point (x_{goal}, y_{goal}). A more detailed explanation of the obstacle formulation and the effect of repulsion parameters in the HPPM can be found in [10], [11].

A. Spherical Tracking Algorithm

After the system of nonlinear algebraic equations modeling the environment has been created and the homotopy established, a method is required to obtain all intersection points between the two surfaces described by $H_1(x, y, \lambda)$ and $H_2(x, y, \lambda)$. For this task, the sphere tracking algorithm (SA) proposed by Yamamura [13] is utilized. The SA is described as a technique with a straightforward operating principle; it is an iterative process used to identify an intersection point between the tracking curve H^{-1} and an n-dimensional sphere. In this process, the initial sphere or circumference is placed at a point along the desired trajectory, ensuring that the current sphere will intersect the curve at least at two points. Subsequently, the next sphere is placed at the intersection point of the previous sphere in the forward direction. This process is repeated a number of times, determined by the user, or until reaching the point where the curve intersects with $\lambda = 1$ [12]. In Figure 1, the schematic procedure of the spherical algorithm is depicted. Consequently, the homotopic system is augmented with the SA:

$$H_S = \begin{cases} H_1(x, y, \lambda) = 0, \\ H_2(x, y, \lambda) = 0, \\ S_i(x, y, \lambda) = 0. \end{cases} \tag{9}$$

In this work, three dimensions are considered, representing two spatial dimensions (x, y) of the configuration space and the homotopic parameter λ. The sphere is represented as:

$$S_i(x, y, \lambda) = (x - c_x)^2 + (y - c_y)^2 + (\lambda - c_\lambda)^2 - r^2 = 0, \tag{10}$$

where, r is the radius and (c_x, c_y, c_λ) is the center of the sphere in each spherical tracking step.

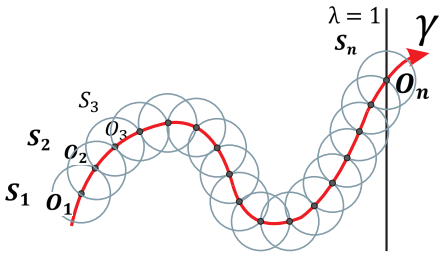


Fig. 1: Spherical tracking for a homotopy curve.

A proper predictor-corrector scheme is employed to allow the SA to accurately trace the solution homotopy curve. For this work, the scheme reported in [10], [14] is utilized. The predictor point is determined using Euler's expression:

$$(x_p, y_p, \lambda_p) = (c_x, c_y, c_\lambda) + r\|\vec{v}_p\| \tag{11}$$

where, (c_x, c_y, c_λ) is the center of the sphere, r is the sphere radius and $\vec{v_p}$ is the tangent vector [14]. Once an initial approximation of the point on the path is obtained using the predictor scheme, the Newton-Raphson (N-R) method is used to adjust the approximation value and compute the system's solution at each step of the spherical tracking. The solution point of the homotopic trajectory is calculated using the N-R method expression as follows:

$$(x_{j+1}, y_{j+1}, \lambda_{j+1}) = (x_j, y_j, \lambda_j) - [J(H_S))]^{-1} H_S, \tag{12}$$

where, j denotes the current iteration, (x_j, y_j, λ_j) is the point obtained during each iteration, and $[J(x_j, y_j, \lambda_j)]^{-1}$ is the inverse matrix of the Jacobian $H_S\,(x_j, y_j, \lambda_j)$. The tolerance criterion for the N-R method in this work is established as:

$$\| H_S\,(x_{j+1}, y_{j+1}, \lambda_{j+1}) \| < 1 \times 10^{-6}, \tag{13}$$

For this work, the maximum number of iterations is set to $j_{max} = 20$. Although it can be observed that the implementation of the HPPM method involves a series of mathematical tools, the precompiled tool presented in this work does not require knowledge of these concepts. It has been designed to simplify the implementation of planning problems for research and teaching purposes.

III. HPPM-TOOL FOR 2-D MOBILE ROBOTS

The HPPM is a planner in which reliance is placed on meticulously configured mathematical tools and methods. Due to its operating principle, it is susceptible to numerical noise, which may arise from data resolution, truncation, or rounding errors. To foster its use among developers, students, and educators in robotics, a precompiled tool named **"HPPM_v1.exe"** has been developed. This tool can be accessed at the repository https://github.com/Guda108/HPPM-tool, which hosts various implementation examples that will be examined in the case study section. Furthermore, the HPPM-tool has been specifically crafted for 2-D environments with circular obstacles. It is compatible with omnidirectional robots and necessitates a top-down perspective of the environment. Such features are indicative of the standard configuration commonly employed to elucidate the functioning of such vehicles in robotics courses. Figure 2 depicts the conventional setup for a path planning issue, where an overhead view of the environment is provided by a camera positioned above. "A" and "B" denote the initial and goal points of the trajectory that the robot is tasked to determine and execute.

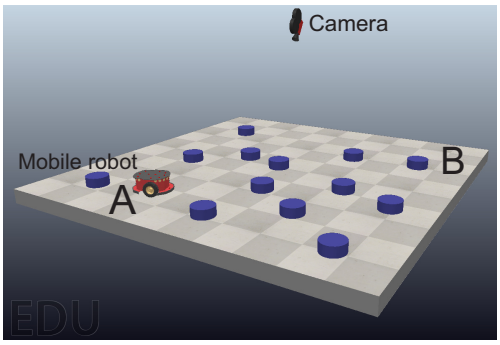


Fig. 2: Simple setup used for capturing top-view images of the environment in 2-D mobile robotics applications. Image generated using CoppeliaSim V-REP EDU.

The **HPPM_v1** tool is configurable using two text files found in the same repository: **slopes.txt** and **obstacles.txt**. The **slopes.txt** file contains all the essential parameters for running HPPM, while **obstacles.txt** specifies the position and size of the circular obstacles. Furthermore, upon execution, the planning tool produces a **tray.txt** file, detailing the collision-free trajectory the robot should follow. The format for the **obstacles.txt** file includes size, position, and repulsion parameter information for each obstacle.

```
x_pos_1  y_pos_1  r_obsta_1  p_obsta_1
x_pos_2  y_pos_2  r_obsta_2  p_obsta_2
.
.
.
x_pos_n  y_pos_n  r_obsta_n  p_obsta_n
```

let (x_pos_i, y_pos_i) for $i = 1, 2, ..., n$ denote the center of the i-th obstacle in 2-D space. The term r_obsta_i signifies the radius of the obstacle, while p_obsta_i indicates its repulsion parameter.

The essential parameters needed for the simulation, such as the starting point, goal point, maximum number of steps, and step size, are set in the **slopes.txt** file. In this file, (x_start,y_start) and (x_goal,y_goal) denote the starting and goal points, respectively. The terms L1_slope and L2_slope

refer to the slopes of the straight lines that define the robot's environmental model. For practical experiments, values of L1_slope=-1 and L2_slope=-4 are recommended (A comprehensive study of these values can be found in [10]). The parameter n_obstacles represents the count of circular obstacles present in the environment. The sphere_size describes the radius of each sphere used in the homotopic path tracking, and max_steps indicates the maximum count of spheres utilized in the SA procedure.

The configuration files have been designed for simplicity, facilitating their use in more advanced implementations for automating the planning problem-solving process. This ease of use will be further illustrated in the case study section. finally, the output file lists the ordered pairs of x-y coordinates, denoting the robot's trajectory.

A. Integration of the HPPM tool into a Python script

The `HPPM_v1.exe` tool can be seamlessly integrated using a high-level programming language such as Python. Python was chosen due to its widespread popularity and its intuitive learning curve. To incorporate the HPPM tool within a Python script, the following line of code can be used:

```
os.system('path_to_HPPM/HPPM_v1.exe')
```

Here, "path_to_HPPM/HPPM_v1.exe" should be replaced with the correct path to the executable. Depending on the operating system, the filename should be `HPPM_v1.exe` (for Windows systems) or `HPPM_v1.o` (for Linux systems). For a comprehensive understanding of how the tool is implemented in Python, one can refer to the provided case studies in the repository https://github.com/Guda108/HPPM-tool.

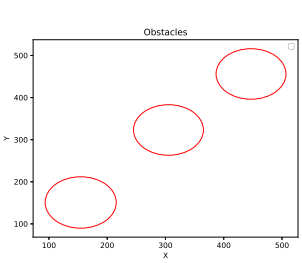
IV. Case Studies

Three case studies are provided below, showcasing the versatility of the `HPPM` tool for both manually-configured simple tasks and intricate tasks involving environments processed through image techniques. All cases were implemented using `Python 3.8` on a personal computer equipped with an `Intel i5-8350U` processor and `8GB` of RAM. Nonetheless, they can be executed on low-resource systems like a `Raspberry Pi`. It should be emphasized that these case studies serve as examples of the tool's intended use. That said, users are encouraged to design their own experiments or case studies to assess the `HPPM` tool's capabilities. In the case studies where the Parallax S2 robot is used, a movement configuration will be employed in which the robot performs rotations and displacements in sequence. The differential movement capability of the robot is not exploited, allowing the robot to be modeled as a holonomic vehicle for purely demonstrative implementations.

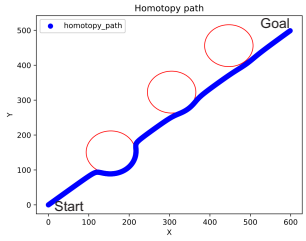
A. Case 1: A simple map with three obstacles

In this first case study, the focus is on a simple map containing three circular obstacles. The objective is to plan a collision-free path for a robot navigating through this environment. The positions and sizes of the obstacles are predefined in **"obstacles.txt"** file. The HPPM tool is utilized to demonstrate the successful generation of a feasible trajectory that avoids collisions with the obstacles. This case study serves as an initial showcase, highlighting the effectiveness and simplicity of the HPPM tool in solving path planning problems within environments involving multiple obstacles. The figure 3 illustrates two different solution paths for the same environment map. In the first path (subfigure 3b), all obstacle repulsion parameters are set with positive values, while in the second path (subfigure 3c), a negative value is used for the central obstacle's repulsion parameter.

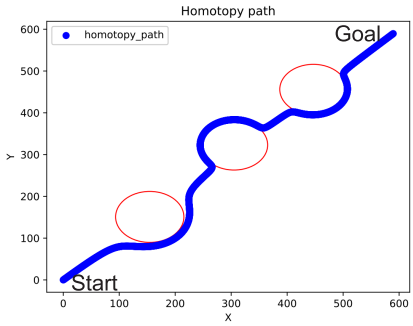
It is worth noting that, for this and the subsequent case studies, the repulsion parameter values for each obstacle are established within the range of $(\pm r_{obs}, \pm {r_{obs}}^2)$. This allows the repulsion parameter to be either positive or negative, based on the user's criteria. The allowable values are bounded by the radius of the obstacle and the square of its radius. This consideration is crucial to ensure the success of tracing a collision-free trajectory. However, it is essential to emphasize that the user should not be restricted by these values, as the proper utilization of these parameters remains an open problem.



(a) Obstacles map



(b) Solution path using all positive repulsion parameters.



(c) Solution path using negative repulsion parameter value of the central obstacle.

Fig. 3: Solution path for the simple environment map of Case 1.

More information about the Python scripts and configuration files is available in the repository at https://github.com/Guda108/HPPM-tool/tree/main/Caso_1. Additionally, a video illustrating the tool's operation with basic case studies *has been made available* at https://youtu.be/jFTSeMHDVj4.

B. Case 2: Environment with 24 obstacles

In this study, an environment comprising 24 obstacles is analyzed. These obstacles are derived from a real environment through image processing techniques. The aim is to chart a collision-free trajectory for a robot maneuvering within this setting using the HPPM tool. The obstacle positions and dimensions are extracted from the image processing outcomes. This study showcases the HPPM tool's ability to accommodate settings with a large count of obstacles, underscoring its utility in tangible scenarios. In real applications, the robot is often portrayed as a singular point within the model, usually signified by its center of mass [1]. This necessitates the inclusion of an extra safety distance, which is factored in by enlarging the obstacle dimensions. For the studies presented here, the robot is conceptualized as a circular entity. Given the circular character of the obstacles, the robot's radius is incorporated into the measurements of each obstacle, as depicted in Figure 4.

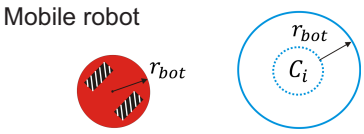
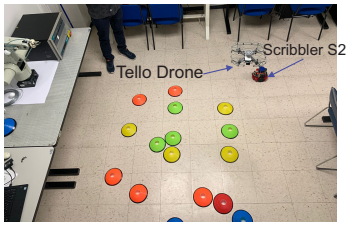
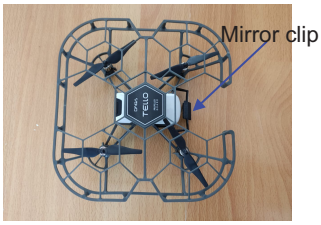


Fig. 4: The robot's radius added as a safety distance to each circular obstacle.

An environment has been set up in the laboratory for this case study and case 3, using circular obstacles as depicted in Figure 5a. To capture a top-down view of the experimental setup, a mirror with a holder has been attached to a Tello drone. This mirror clip allows reflecting the image below the drone to its front camera, as can be seen in Figure 5b.



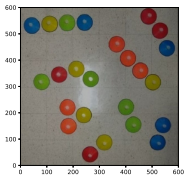
(a) Configuration of the environment map used for case studies 2 and 3.



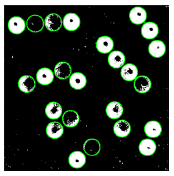
(b) Drone and mirror clip used to capture top-view images of the environments.

Fig. 5: Environment setting and configuration used to capture top-view images.

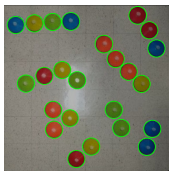
The obstacle positions and sizes to be used by the HPPM tool were determined through image processing, within which a specific region of interest was identified. In case studies 2 and 3, this designated region spanned a physical area of 2x2 meters and was subsequently transformed into a 600x600 pixel image. Figure 6a showcases the region of interest featuring 24 circular obstacles. Concurrently, figures 6b and 6c depict the obstacle edge detection and their accurate classification as circles within the environment, respectively.



(a) Environment map of Case 2.



(b) Edge detection of the environment map.



(c) Correct assignment of circumferences.

Fig. 6: Edge detection and assignment of circumferences to each obstacle in the environment map of Case 2.

In Figure 7, the solution path for the planning problem is illustrated. It is evident that a safety distance of 10 cm, which corresponds to the radius of the Parallax Scribbler S2 robot used in case 3, has been added to the contour of each obstacle.

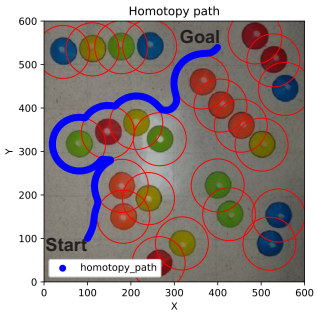


Fig. 7: Solution path for case study 2. The robot's radius added as a safety distance to each circular obstacle.

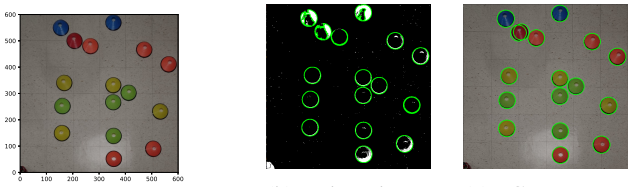
The data produced by the edge detection and circumference assignment script has been employed to update the configuration file **"obstacles.txt"**, which is subsequently utilized by the HPPM tool. Detailed information regarding the Python scripts and configuration files can be found in the repository at https://github.com/Guda108/HPPM-tool/tree/main/Caso_2.

C. Case 3: Path planning for Parallax Scribbler S2

In this case study, the focus is on path planning and execution for the Parallax Scribbler S2 robot. The integration of the HPPM tool with this physical robot platform is showcased. A collision-free trajectory for the Scribbler S2 robot in a specified environment is planned using the HPPM tool. The case underscores the HPPM tool's practical application in real-world scenarios involving a physical robot, emphasizing its ease of use and effectiveness in mobile robotics.

Figure 8a illustrates a region containing 16 circular obstacles. Meanwhile, figures 8b and 8c depict the edge detection of the obstacles and their accurate representation as circumferences in the environment, respectively.

Figure 9a shows the solution path of the planning problem has been illustrated, where the safety distance corresponding to the radius of the Parallax Scribbler S2 robot. In this figure, it can be noticed that one of the physical obstacles is represented with two circumferences in the environment due to an error in



(a) Environment map of case 3.

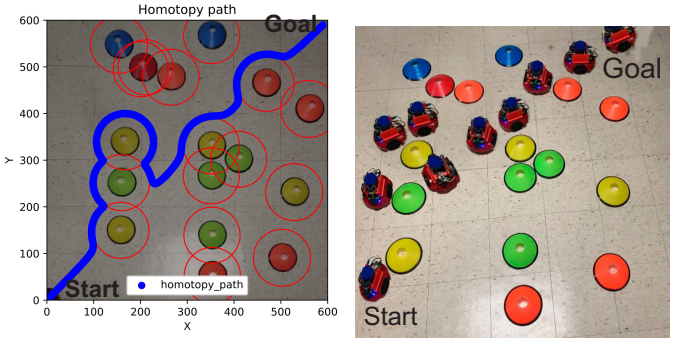
(b) Edge detection of the environment map.

(c) Correct assignment of circumferences.

Fig. 8: Edge detection and assignment of circumferences to each obstacle in the environment map of Case 3.

edge detection. While this increases the number of obstacles in the environment, it does not affect the execution of the tool, as the problem can still be successfully solved, and the execution time is only a few milliseconds. It is worth noting that the execution time is not reported in this work, as it is not the focus of this study. However, the execution time of the HPPM tool is negligible compared to the path execution time and image processing time.

For case studies 2 and 3, positive repulsion parameters were used, resulting in a suboptimal trajectory. Nevertheless, the trajectory successfully reaches the goal from the starting point. Finally, in Figure 9b, a sequence of movements made by the Parallax Scribbler S2 robot is displayed. It can be observed that the solution trajectory, from the starting point to the goal, is followed by the robot.



(a) Solution path for Case 3. The map considers 16 physical obstacles and a safety distance of 10cm.

(b) Execution of the solution path using the HPPM tool with the Parallax Scribbler S2 robot.

Fig. 9: Edge detection and assignment of circumferences to each obstacle in the environment map of Case 3.

Information regarding the Python scripts, images, and configuration files can be found in the repository at https://github.com/Guda108/HPPM-tool/tree/main/Caso_3.

V. Conclusions and Future Work

This paper introduces the HPPM-tool, a user-friendly path planning tool tailored for a broad audience, from elementary education to advanced research in universities. Designed primarily to show the utility of HPPM, this tool also supports integration with Python scripts, catering to research needs. Through various case studies, the simplicity and adaptability of the tool in different situations have been emphasized, with its ability to quickly compute collision-free paths standing out. For easier access in educational settings, a collection of scripts, examples, and case studies is available on a dedicated GitHub repository. Regarding future developments, the integration of real-time image processing modules is being considered to identify both the robot's position and that of any moving obstacles, thereby aiding operations in dynamic environments. Moreover, the inclusion of multiple robots in the environment is feasible, with each robot being treated as an obstacle relative to the others. This opens the door to developing strategies for collaborative robotics.

Acknowledgment

Acknowledgments to the National Council of Humanities, Science and Technology (CONAHCyT) for their support of this project under grant 480421, and to the National Institute for Astrophysics, Optics and Electronics for providing the facilities and resources to implement the case studies.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006. Available at http://planning.cs.uiuc.edu/.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, pp. 846–894, jun 2011.

[3] N. A. K. Zghair and A. S. Al-Araji, "A one decade survey of autonomous mobile robot systems," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 6, p. 4891, 2021.

[4] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, 2019.

[5] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[6] I. Al-Bluwi, T. Siméon, and J. Cortés, "Motion planning algorithms for molecular simulations: A survey," *Computer Science Review*, vol. 6, no. 4, pp. 125–143, 2012.

[7] S. Liu and M. A. Belabbas, "A homotopy method for motion planning," *arXiv preprint arXiv:1901.10094*, 2019.

[8] G. Diaz-Arango, H. Vazquez-Leal, L. Hernandez-Martinez, V. M. Jimenez-Fernandez, A. Heredia-Jimenez, R. C. Ambrosio, J. Huerta-Chua, H. De Cos-Cholula, and S. Hernandez-Mendez, "Multiple-target homotopic quasi-complete path planning method for mobile robot using a piecewise linear approach," *Sensors*, vol. 20, no. 11, 2020.

[9] H. J. Islas, M. L. Quemada-Villagómez, M. de la Luz López-González, and J. M. Oliveros-Muñoz, "Enseñanza del método de continuación homotópica con seguimiento hiperesférico para estudiantes de ingeniería," *Acta Universitaria*, vol. 32, pp. 1–43, 2022.

[10] G. Diaz-Arango, H. Vázquez-Leal, L. Hernandez-Martinez, M. T. S. Pascual, and M. Sandoval-Hernandez, "Homotopy path planning for terrestrial robots using spherical algorithm," *IEEE Transactions on Automation Science and Engineering*, vol. PP, no. 99, pp. 1–19, 2017.

[11] H. Vazquez-Leal, A. Marin-Hernandez, Y. Khan, A. Yildirim, U. Filobello-Nino, R. Castaneda-Sheissa, and V. Jimenez-Fernandez, "Exploring collision-free path planning by using homotopy continuation methods," *Applied Mathematics and Computation*, vol. 219, pp. 7514–7532, 2013.

[12] M. d. l. L. López-González, M. L. Quemada-Villagómez, G. M. Martinez-González, J. M. Oliveros-Muñoz, and H. Jiménez-Islas, "A novel predictive homotopic path tracking algorithm to solve non-linear algebraic equations," *The Canadian Journal of Chemical Engineering*, vol. 101, no. 6, pp. 3382–3408, 2023.

[13] Y. K., "Simple algorithms for tracing solution curves," *IEEE International Symposium on Circuits and Systems*, vol. 6, pp. 2801–2804, 1992.

[14] J. M. Oliveros-Munoz and H. Jiménez-Islas, "Hyperspherical path tracking methodology as correction step in homotopic continuation methods," *Chemical Engineering Science*, vol. 97, pp. 413–429, 2013.