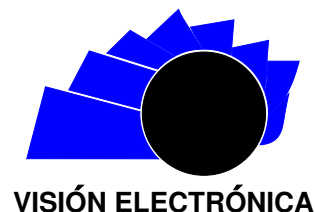




Visión Electrónica

Más que un estado sólido



<https://doi.org/10.14483/issn.2248-4728>



A RESEARCH VISION

PRM navigation in trading drone and Gazebo simulation

Navegación PRM en dron comercial y simulación Gazebo

Franklin Pineda-Torres ¹, Luis Alejandro Arias-Barragán ²

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:

Enviado: 27/01/2020

Recibido: 20/02/2020

Aceptado: 15/05/2020

Keywords:

Drone
Gazebo
Matlab
Navigation
PRM
ROS



Palabras clave:

Dron
Gazebo
Matlab
Navegación
PRM
ROS

ABSTRACT

Starting from a commercial drone AR Dron Parrot 2.0, an autonomous navigation process is developed with a PRM probabilistic route planner in real time, through a ROS network between the drone and the Gazebo simulation software. Using the robotics system toolbox from software Matlab that interacts with Gazebo, it is possible to study the desired trajectory planner, in addition, the creation and connection of the ROS network on the Linux operating system, where the navigation algorithm is analyzed from the practical vs., simulation points of views. The errors that are presented are minimal, taking into account the propagation delays and the control algorithm; this is in charge of receiving location information in order to correct and minimized the mean square error.

RESUMEN

Partiendo de un dron comercial AR Dron Parrot 2.0, se desarrolla un proceso de navegación autónoma con un planificador de ruta probabilístico PRM en tiempo real, por medio de una red ROS entre el dron y el software de simulación Gazebo. Utilizando el paquete de robótica - Robotics System Toolbox del software Matlab que interactúa con Gazebo, es posible el estudio del planificador de trayectoria deseado, anexando la creación y conexión de la red ROS sobre sistema operativo Linux, donde se analiza el algoritmo de navegación desde el punto de vista práctico vs., simulación. Los errores que se presentan son mínimos, teniendo en cuenta los retardos de propagación y el algoritmo de control para que en la medida que se recibe información de ubicación se pueda corregir y minimizar el error medio cuadrático.

¹BSc. In Electronic engineering, MSc Electronic and Computers Universidad de los Andes, Specialist in Automation and Industrial Computing. Professor and director of the group research seedbed S&C at the Fundación Universidad Autónoma de Colombia, Colombia. E-mail: franklin.pineda@fuac.edu.co

²BSc. In Electromechanic engineering, MSc Information sciences and telecommunications. Ph.D. in engineering. Current position: Professor and researcher at Fundación Universidad Autónoma de Colombia, Colombia. E- mail: larias.barragan@fuac.edu.co

1. Introduction

The planning and generation track plays a priority role in all the developments that are being made for applications with autonomous mobile robots. The ability of a mobile robot to determine its location in space is an essential task to be able to navigate completely autonomously [1]. The knowledge of the robot position, as well as other characteristics of interest in the robot's environment like an objects and obstacles, are the basic foundations on which the highest level navigation operations are based. For example, the location allows the robot be possible to plan paths to reach a specific destination while avoiding obstacles [2], and it is essential for more complex tasks such as building maps of unknown environments. Without the ability to locate, robots would be doomed to interact with the environment through reactive behaviors, and would be unable to plan actions beyond their local field of perception [3].

The robots, in this case the UAVs, must calculate their trajectory autonomously in order to carry out their mission in a safe and efficient way from the information obtained from the environment through the on-board sensors.

There are several methods of geometric origin to solve the basic problem of road planning; three methods have been highlighted [2]: artificial potential fields or *potential functions*, roadmaps and *cell decompositions*. In the last decade a new methodology called *probabilistic* road maps has been developed [1,4,5]. That methodology was created because the geometric methods may have limitations in terms of the necessary computation time and success in the generation of trajectories, this is especially due for complex environment, high number of degrees of freedom and kinematic restrictions [6].

1.1. General Description

The general idea is: first generate on a previously established environment in both simulation softwares: Matlab and Gazebo, these must be the same morphologically but not necessarily in their scale. Second, it is necessary to locate the obstacles, the limits of movement, the initial coordinate and the final the drone must start and finish its movement over a coordinate system. The probabilistic route algorithm takes the environment and indicates the form and nodes where the drone should move; The ROS robotics operating system, interacting on a Wi-Fi network, it is created by the drone using its own wireless module, it communicates to gazebo software with the settings

that the drone must have for its operation. The figure 1 represent graphically and generally represents the aspects involved in the project.

Figure 1: General Description.



Source: own

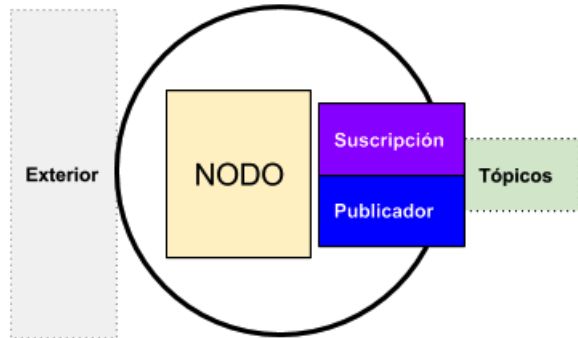
2. ROS

Robot Operating System - ROS provides standard services of an operating system such as hardware abstraction, control of low-level devices, implementation of common use functionality, message passing between processes and package maintenance. It is based on a graph architecture where processing takes place at the nodes that can receive, send and multiplex sensor messages, control messages, states messages, planning and actuators messages, among others [7, 8]. Since ROS standardizes the way of handling services, in order to work with ROS you must be sure that the software and / or hardware manufacturer (nodes) provides the appropriate drives and drivers for the operating system OS; the platform where you are working on.

2.1. Communication with ROS

We can very briefly represent a node as a piece that has a mechanism for communicating with ROS through messages and that can have an interconnection mechanism with external elements [9]. ROS essentially incorporates two communication mechanisms subscription/publication and services –see figure 2. The request and response is made through the services and the subscription/publication, they can be seen as sending and receiving information based on topics (types of messages) in the ROS. The topics of subscription and publication are a huge list because they are the most used and it is also possible to create new topics.

Figure 2: Types of nodes for the communication with ROS. [9].



2.2. ROS in Matlab

Before using ROS with Matlab it must be installed [10]. The robotics toolbox provides creation, connection and communication with a ROS network. It must start with the *rosinit* function and finish with *rosshutdown* function. The input parameters for initialization are two the host name, which can be an IP address, and the access port. The list of nodes can be observed with *rosclear* function and the different topics with *rostopic* function. As you noted, the functions are very intuitive and practical in their use.

3. PRM

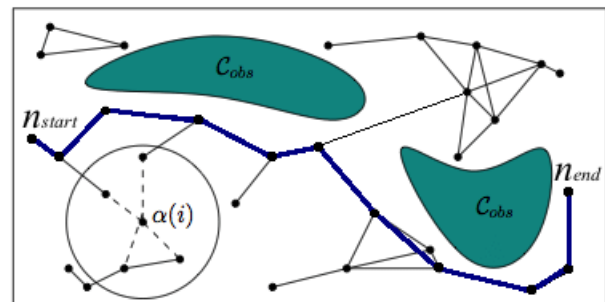
The probabilistic road map is created using a random configuration space; The basic idea for two dimensions is to choose random points that will be joined according to a function of distance over the free space they have every two points. Each union creates a new edge within a graph, this will be solved using some search algorithm such as A*, Dijisktra etc., [6]. El pseudo-code is presented in table 1.

Table 1: Pseudocode PRM. Adapted from [6].

Pseudo-code PRM (x):	
1:	for 1:N
2:	$x, y \leftarrow \text{random}$
3:	if $x_k \in C_{free}$
4:	$f(\text{distancia})$
5:	conectar nodos
6:	end if
7:	end for
return	

Once the PRM algorithm has connected the nodes, annexing that the characteristic obstacle space must be taken into account C_{obs} in the figure 3, it starts on the node n_{start} and trace the path to the final node n_{end} . To calculate the minimum trajectory, the Euclidean distance is usually used with the A* algorithm. It is observed that among larger values for N in the pseudocode, the path to be found is more optimal, in fact, there may be situations where N is small to find a solution from n_{start} to n_{end} . As an important assumption, a large N will cost computation time which means that a relationship of nodes vs. solution vs. time must be established.

Figure 3: Creation of probabilistic route map. Adapted from [6].



Source: own

3.1. PRM in Matlab

Within the robotics package with Matlab R2018 and superiors, it is possible to create a PRM object that is specified with the map Map, the number of nodes *NumNodes* and the connection distance between nodes *ConnectionDistance* [10]. Exemplifying in table 2 a short script of how the object is created by calling the class *robotics* and access to their methods: *BinaryOccupancyGrid* design the respective map, the variable *sMap* is an array where the obstacles are included with ones over an environment of zeros; it follows that curved or polygonal obstacles should not approximate obstacles with right angles. The variable *sc* is the scaling with respect to the real environment. The figure 4 shows the result for 20 nodes.

3.2. Matlab-ROS-ArDrone

The connection between Matlab and the Ardrone 2.0 is made through the package *ardrone_autonomy* [11] that manually or command control of the Ardrone 2.0 is possible. The node of the Matlab network in figure 5 allows connection to the ROS.

Table 2: PRM Script of example.

```

script-example_PRM:
1: map = robotics.BinaryOccupancyGrid(sMap,sc);
2: prmSimple = robotics.PRM(map,NumNodes);
3: nstart = [2 6];
4: nend = [13 4];
5: path = findpath(prmSimple,nstart,nend);
6: show(prmSimple)
return

```

Source: own

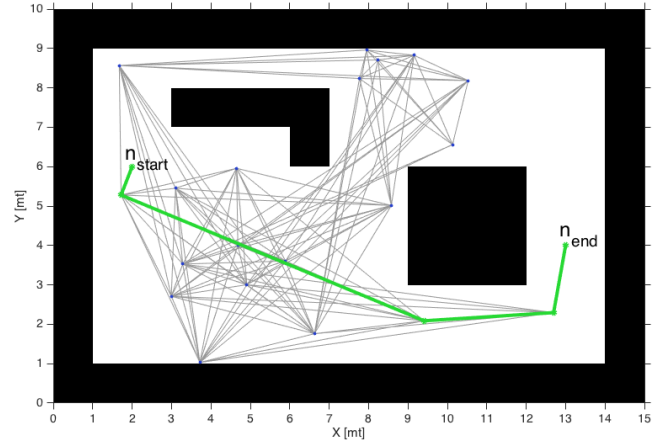
4. Gazebo

Gazebo is a dynamic 3D simulator for free use [12-13], with the ability to accurately and efficiently simulate different types of robots in complex indoor and outdoor environments. Although it is similar to game engines, Gazebo offers physical simulation with a much higher degree of fidelity, where it is possible to use a set of sensors and interfaces for users, also with the possibility of connecting to other programs. Although Gazebo can work on Windows³, it is not recommended because the CPU loads a lot, it is better to use it on another operating system.

4.1. Matlab-Gazebo integration

The link or connection for the communication and movement of the Virtual Robot (Gazebo) with the PRM algorithms of Matlab is done through a Matlab-ROS-Gazebo integration, see figure 6, this receives the Matlab commands or orders and transmits them to the Gazebo simulator to be interpreted as instructions; These instructions allow to visualize

the movement of the Ardrone 2.0 through simulation environment, Figure 7.

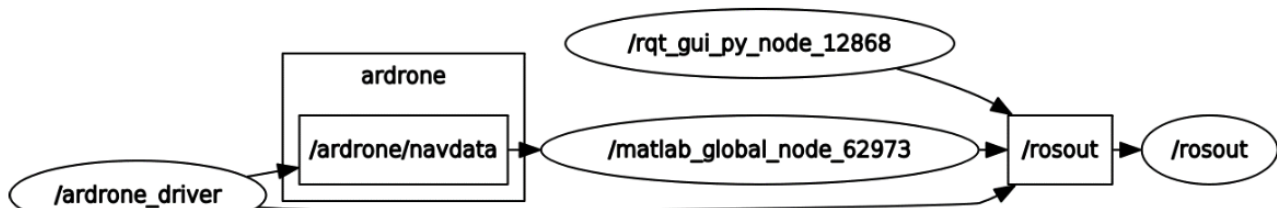
Figure 4: Script result.

Source: own

5. Tests and results

Having Matlab-ROS-ArDrone2.0 and Matlab-ROS-Gazebo networks complete, three routes are established as evidence, one per quadrant –see table 3, starting in the coordinate (2,0,0); the coordinate (0,0,0) is a wall on the chosen route map –see figure 7. Matlab generates the PRM and the solution is sent to both: the physical drone and the drone simulated in Gazebo, this is done node by node over the entire trajectory. The results are based on the relative error equation (1) of the simulated drone target coordinate. $Dron_S$ and the physical Dron $Dron_F$.

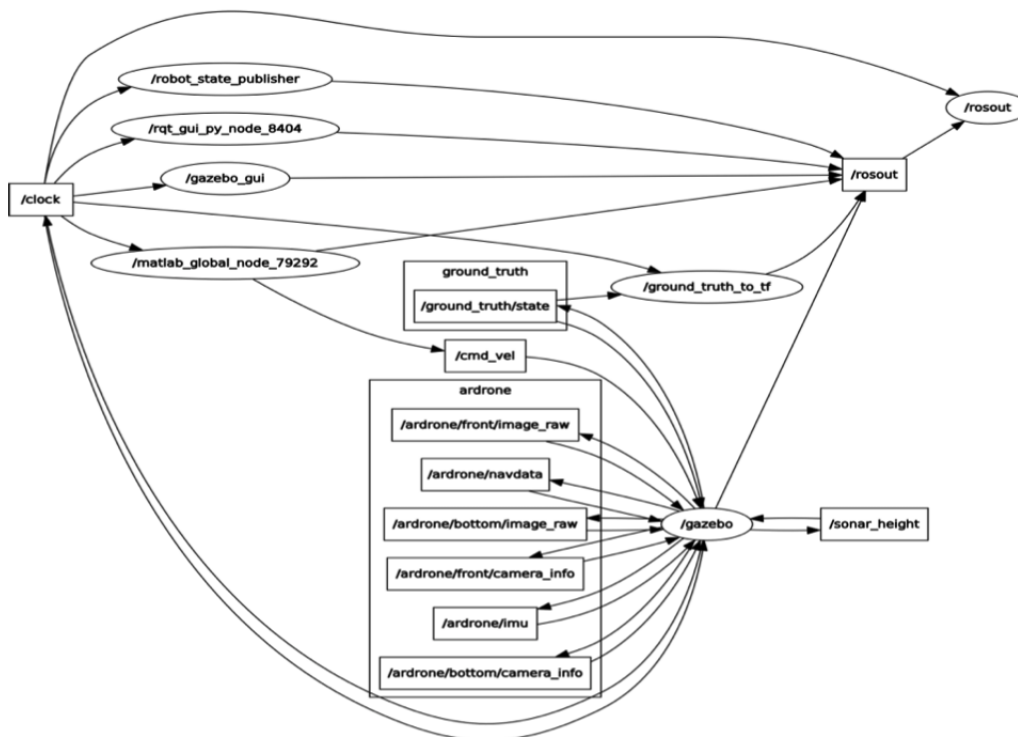
$$e_r = \frac{Dron_S - Dron_F}{Dron_S} * 100\% \quad (1)$$

Figure 5: Matlab node for ArDrone control.

Source: own

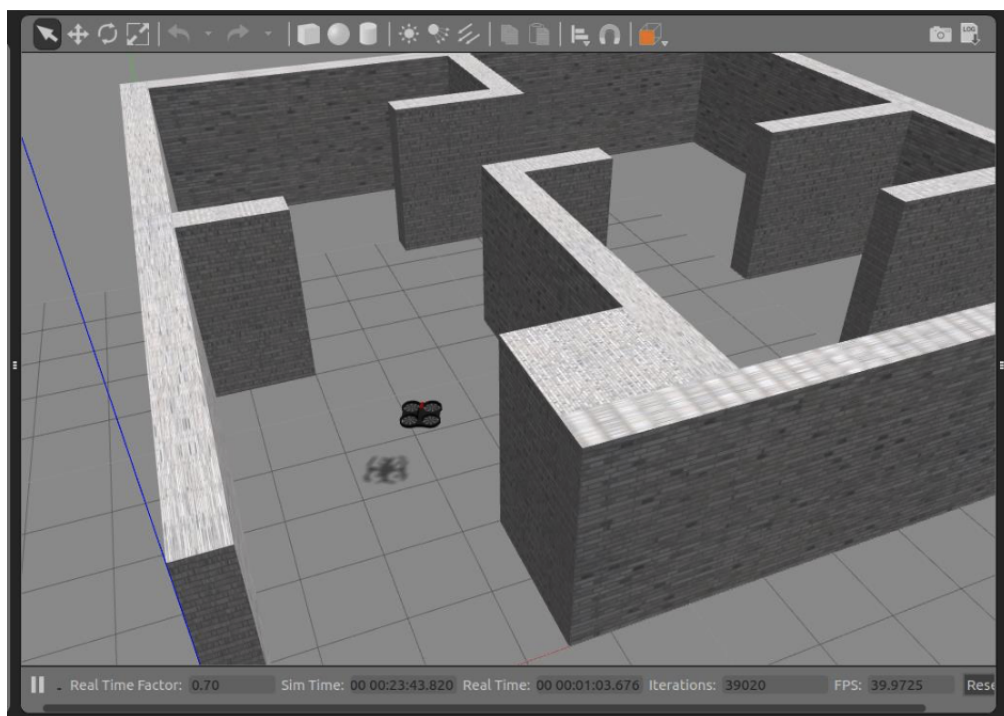
³To run Gazebo on Windows, it is recommended to use Gazebo Web [13]

Figure 6: Nodes and topics to Matlab-Gazebo integration.



Source: own

Figure 7: Simulation environment Gazebo.



Source: own

Table 3: Test Routes.

<i>Route 1:</i>	<i>nodo_fin = [9 5];</i>
<i>Route 2:</i>	<i>nodo_fin = [10 10];</i>
<i>Route 3:</i>	<i>nodo_fin = [4 10]</i>

Source: own

5.1. Results Route 1

The nodes generated by the PRM for the first route are shown in table 4. There it is observed that both the second and the penultimate node are very close to the beginning and end of the trajectory respectively.

Table 4: Nodes: Route 1.

<i>Path Route 1</i>						
x [mt] :	2	1.8531	4.4551	7.8059	9.0608	9
y [mt] :	0	0.1558	7.2647	7.0649	5.1210	5

Source: own

The relative errors developed with respect to the final location are shown in Table 5. Although the relative error is smaller on the x-axis for route 1, the mean square error rms of the entire trajectory on this axis is much greater.

Table 5: Errors: Route 1.

	<i>Dron_F</i>	<i>Dron_S</i>	<i>e_r</i>	<i>e_{rms}</i>
x [mt] :	8.9097	9.0523	1.57%	18.99%
y [mt] :	5.4243	5.0812	6.75%	13.26%

Source: own

5.2. Results Route 2

The nodes generated by the PRM for the second route are shown in Table 6. As can be seen in Figure 9, the errors for this route are greater with respect to the first one. The mean squared error of the trajectory in the real drone on the x-axis exceeds the other errors by high values, - see Table 7.

Table 6: Nodes: Route 2.

<i>Path Route 2</i>							
x [mt] :	2	2.1671	3.4904	5.2556	7.5362	9.5969	10
y [mt] :	0	0.8686	5.8767	8.1023	8.9617	10.2894	10

Source: own

Table 7: Errors: Route 2.

	<i>Dron_F</i>	<i>Dron_S</i>	<i>e_r</i>	<i>e_{rms}</i>
x [mt] :	9.8991	9.5644	3.38%	36.04%
y [mt] :	10.0481	10.2278	1.78%	9.72%

Source: own

5.3. Results Route 3

Route 3 is the shortest route with only two nodes - see table 8. In table 9, similar to route 2, the mean square error on the coordinate is high, 40%, this is due to the impulse that the drone carries on the axis and its respective balances. Figure 10 shows part of the stage with the drone landing on the end coordinate, there each tile is approximately 50cm.

Table 8: Nodes: Route 3.

<i>Path Route 3</i>				
x [mt] :	2	1.8531	3.9124	4
y [mt] :	0	0.1558	9.8943	10

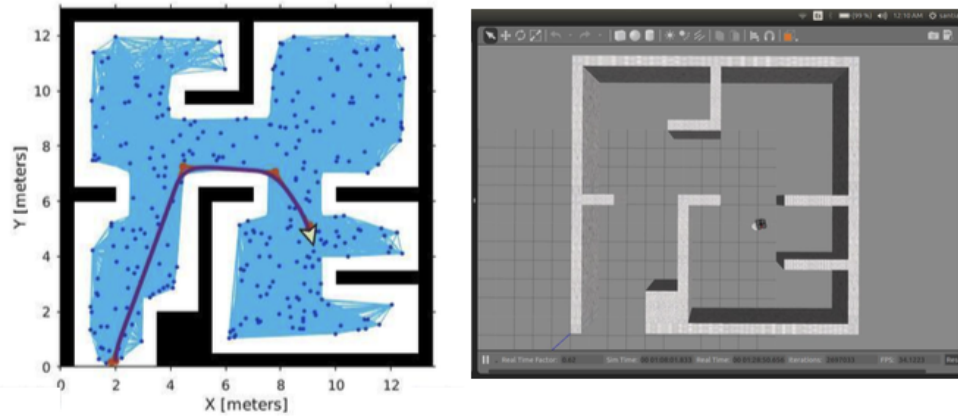
Source: own

Table 9: Errors: Route 3.

	<i>Dron_F</i>	<i>Dron_S</i>	<i>e_r</i>	<i>e_{rms}</i>
x [mt] :	3.9118	3.9464	1.86%	18.49%
y [mt] :	9.9176	9.5488	3.79%	40.11%

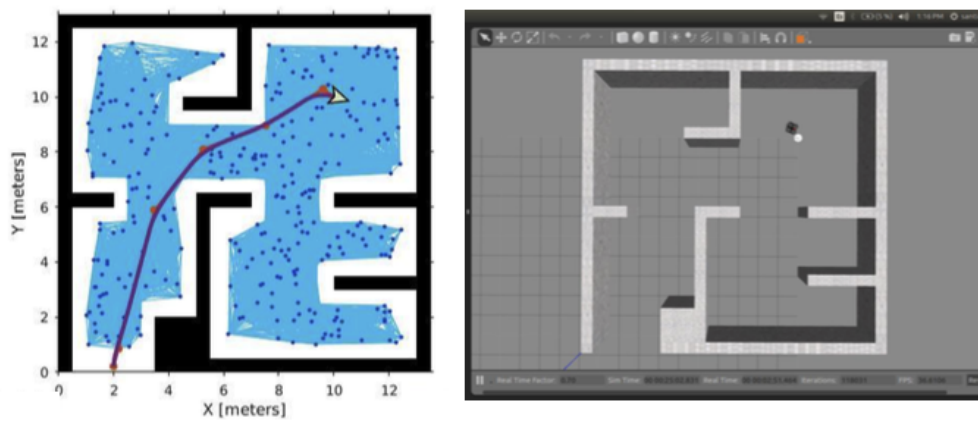
Source: own

Figure 8: PRM and Gazebo Simulation for the Route 1.



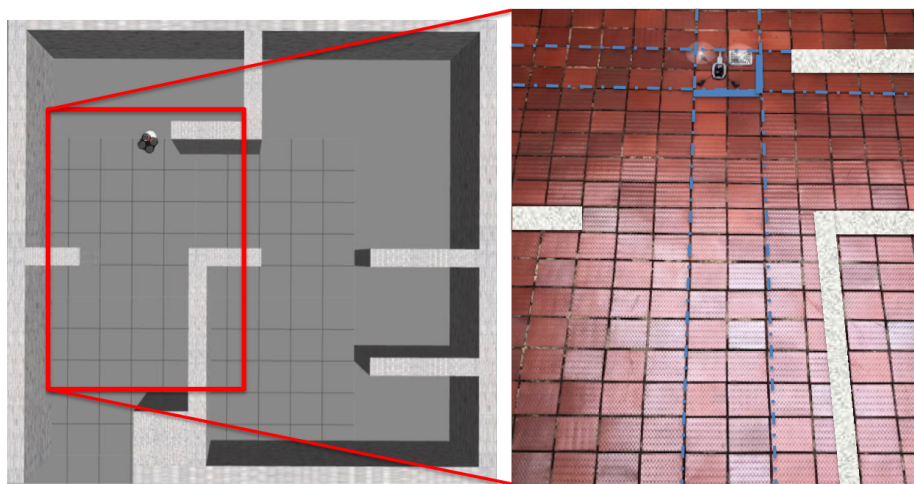
Source: own

Figure 9: PRM and Gazebo Simulation for the Route 2.



Source: own

Figure 10: Gazebo Simulation and a part of the real scenario for Route 3.



Source: own

6. Conclusions

This article shows the PRM as an algorithm to implement navigation paths with a commercial drone; There, two types of communication networks are identified, the first one where Matlab simulation software and the physical drone are involved and the second one between Matlab and the simulation and emulation software Gazebo. The upper layer of both networks called ROS; Open access robotics operating system, creates nodes by means of topics, to interact the different components of both hardware and software. The networks worked with good response times and low CPU load on Linux. From the trajectories analyzed and generated by the PRM, it was found that the imbalances that the Ardrone 2.0 has, it makes the average square errors of the trajectory vary greatly, tending to be high, almost always greater than 10%. The relative errors referred to the goal are minor, tending to be less than 5%.

References

- [1] Mathworks: Robotics System Toolbox, “Design and test algorithms for robotics applications”, 2019. [Online] Available at: <https://la.mathworks.com/products/robotics.html>
- [2] J. C. Latombe, “Robot Motion Planning”, Boston, MA: Springer US, 1991. <http://doi.org/10.1007/978-1-4615-4022-9>
- [3] R. Siegwart, I. Nourbakhsh, D. Scaramuzza, “Introduction to Autonomous Mobile Robots”, Second Edition, MIT, 2011.
- [4] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki and S. Thrun, “Principles of Robot Motion: Theory, Algorithms, and Implementation”, 2007. [Online] Available at: http://biorobotics.ri.cmu.edu/book/booboo_book.pdf
- [5] D. C. Hanselman and B. Littlefield, “The student edition of MATLAB”, user’s guide. Prentice Hall, 1997.
- [6] S. M. LaValle, “Planning algorithms”, Cambridge University Press, 2006. <https://doi.org/10.1017/CB09780511546877>
- [7] A. Ángel, R. Gandul and F. C. Rojo, “Integración de ROS con Arduino y Raspberry Pi”. Universidad de Sevilla, 2014. [Online] Available at: <http://bibing.us.es/proyectos/abreproy/12186/fichero/PFC+Alvaro+Romero+Gandul+Revisa%20do.pdf>
- [8] ROS, “ROS.org — About ROS”, 2015. [Online] Available at: <http://www.ros.org/about-ros/>
- [9] Programación ROS, “¿Cómo funciona ROS?”, 2019. [Online] Available at: <http://programacionextrema.es/2017/12/08/ros/>
- [10] Mathworks & ROS, “Connect a ROS network”, 2019. [Online] Available at: <https://la.mathworks.com/help/robotics/examples/connect-to-a-ros-network.html>
- [11] Ar.Drone, “Volar AR.Drone con Teclado”, 2019. [Online] Available at: <https://www.tamps.cinvestav.mx/~arodas/keyboard.html>
- [12] Gazebo, “Why Gazebo?”, 2019. [Online] Available at: <http://gazebosim.org>
- [13] Web Client for Gazebo, “Gzweb”, 2019. [Online] Available at: <http://gazebosim.org/gzweb.htmlb>