Tekhnê
July - December 2021, Vol. 18, No. 2, pp. 25 – 36

ISSN 1692-8407

Swarm behavior simulator with bacterial Quorum Sensing

Simulador de comportamiento de enjambre con Quorum Sensing bacteriano

Eyder A. Rodríguez C.[1] and Daniel M. Romero S.[2]
[1]Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá, Colombia
earodriguezc@correo.udistrital.edu.co
[2]Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá, Colombia
dmromeros@correo.udistrital.edu.co

One of the most useful tools in the design of path-planning solutions is simulators. Thanks to them, it is possible to predict the performance of certain control strategies. In this paper, a simulator is presented that implements a swarm of automatons, which perform a wild motion in a user-selected environment. The robots will have the quality to avoid collisions with different obstacles that affect their mobility since they are equipped with proximity sensors. The interface of this simulator was designed entirely with the Qt Designer software. Successful configurations that replicate the performance of the real prototype are presented.

Keywords: Path-planning, quorum sensing, simulator, software, swarm

Una de las herramientas más útiles en el diseño de soluciones de planificación de trayectorias son los simuladores. Gracias a ellos, es posible predecir el rendimiento de determinadas estrategias de control. En este trabajo se presenta un simulador que implementa un enjambre de autómatas que realizan un movimiento salvaje en un entorno seleccionado por el usuario. Los robots tendrán la cualidad de evitar colisiones con diferentes obstáculos que afecten a su movilidad ya que están equipados con sensores de proximidad. La interfaz de este simulador se ha diseñado íntegramente con el software Qt Designer. Se presentan configuraciones exitosas que replican el desempeño del prototipo real.

Palabras clave: Enjambre, Planificación de rutas, quorum sensing, simulador, software

Article typology: Research

Received: June 25, 2021
Accepted: November 11, 2021

Research funded by: Universidad Distrital Francisco José de Caldas (Colombia).

How to cite: Rodríguez, E., and Romero, D. (2021). *Swarm behavior simulator with bacterial Quorum Sensing*. Tekhnê, 18(2), 25 -36.

Introduction

Motion planning is a critical aspect of programming an automaton, or a robot (Ichter et al., 2018; Mohanan & Salgoankar, 2018). It allows the robot to interact efficiently with its environment and enables it to move around dynamically, rather than being limited to its visible range. To address this need, motion planning simulation software can be developed. This software can be designed with certain basic functionalities and rules (Patle et al., 2019).

The motion planning simulation software in question implements a swarm of automatons that move around in a user-defined environment. These robots are equipped with proximity sensors, which allow them to avoid collisions with various obstacles that may affect their mobility. When the sensor detects the presence of an obstacle, the robot will randomly change direction to continue moving around in a wild, unpredictable manner (Bobadilla et al., 2012).

The user can choose the dimensions of the environment in which the automatons will be moving (Martínez et al., 2012). This environment is divided into a grid, which defines the regions where the automatons will be able to move. These regions are assigned weight values, which determine where the robots will tend to converge to reach the quorum, which is also defined by the user. The quorum is achieved when the automatons compare the weight values and the number of automatons present in each region.

The process of designing and programming this simulation software involves several steps, starting with the interface design and ending with the display of the quorum results (Khan et al., 2017). The interface is designed using the Qt Designer software, which makes it easy to position and size all the elements. The design is also focused on making it easy for the user to interact with the simulator. The final result of the interface design will be discussed in more detail later.

Literature review

(Huang et al., 1995) discuss the design and implementation of the simulator PowerMill, a novel transistor level simulator for the simulation of current and power behavior in vlsi circuits. They can form robust interspecies networks - bacterial communities - via sophisticated communication protocols. (Flikkema & Leid, 2005) assert that the improved understanding of these communities in the last decade provides a new model for swarm intelligence with distinct advantages, including ease of laboratory experimentation, explicit coupling of communication and behavior, and intergenerational dynamics. (Hsieh et al., 2008) present a biologically inspired approach to the dynamic assignment and reassignment of a homogeneous swarm of robots to multiple locations, which is relevant to applications like search and rescue, environmental monitoring, and task allocation. This paper studies self-organized flocking in a swarm of mobile robots. (Ferrante et al., 2012) present Kobot, a mobile robot platform developed specifically for swarm robotic studies, briefly describing its sensing and communication abilities. (Korani, 2008) present a new algorithm for PID controller tuning based on a combination of the foraging behavior of E coli bacteria foraging and Particle Swarm Optimization (PSO). This paper describes some of the results obtained after the design and implementation of a discrete cellular automata simulating the generation, degradation and diffusion of particles in a two dimensional grid where different colonies of bacteria coexist and interact. This lattice-based simulator use a random walk-based algorithm to diffuse particles in a 2D discrete lattice (Gómez & Rodríguez, 2011). To keep the level of quorum sensing molecules below the activation threshold (Lo et al., 2015) propose a biological controller that can generate different concentration levels of inhibitors under different environment conditions. An existing quorum sensing strategy enables a swarm to alter collective behavior after a decision, but has not been evaluated under local communication restrictions. (Cody & Adams, 2017) combine these strategies to enable a swarm to detect when a majority of agents support a site and cease deliberation in order to rapidly build a consensus. One of the subject of (Roozbahani & Handroos, 2019) is that the proposed control method combines a directed random search method and real-time simulation to develop an intelligent controller in which each generation of parameters is tested on-line by the real-time simulator before being applied to the real process. (Martínez et al., 2020) propose a strategy for the coordination of a swarm of robots in an unknown environment.

Problem statement

Designing and programming a motion planning simulation software that allows for the implementation of a swarm of automatons that can perform wild movement in a user-selected environment. These automatons should be equipped with proximity sensors that allow them to detect and avoid collisions with obstacles that may affect their mobility. Additionally, the environment should have a grid system that defines regions for the automatons to move in, with each region being assigned a weight value. The automatons should be able to compare the weight values and number of automatons present in each region in order to reach a quorum, which is also defined by the user. The software should have a user-friendly interface that allows for easy interaction with the simulator, and should provide a detailed explanation of the design and programming process.

The motion planning simulation software should be able to handle dynamic and constantly changing environments,

as this is an important aspect of modern automation. The automatons should be able to move randomly while avoiding obstacles, and should be able to converge on a specific region based on the weight values and number of automatons present in that region. The user should be able to adjust the dimensions of the environment, as well as the weight values and quorum requirements. The interface should be designed using the Qt Designer software, and should prioritize ease of use for the user.

The final result of the simulation software should include a detailed explanation of the design and programming process, including the steps involved and the methods used to achieve user interaction. The software should also display the quorum results, allowing the user to see how the automatons have converged on specific regions based on the weight values and number of automatons present. Overall, the motion planning simulation software should provide a flexible and user-friendly platform for programming and interacting with a swarm of automatons in dynamic environments.

Methods

In terms of the programming process, the first step is to define the functionalities and rules that the simulation software will follow. This may include the algorithms that the automatons will use to avoid collisions, the criteria for determining the quorum, and the rules for how the automatons will move around in the environment. Once these details have been ironed out, the next step is to implement the code that will bring the simulation to life. This may involve using programming languages such as C++ or Python to create the automatons, the environment, and the various algorithms and rules that will govern their behavior.

Once the code has been written, the next step is to test the simulation to ensure that it is working as intended. This may involve simulating different configurations, such as different numbers of automatons or different environments, to see how the automatons behave in each case. This testing process will help to identify any bugs or issues that need to be addressed before the simulation is ready for use.

Once the simulation is working as intended, the final step is to create the user interface that will allow the user to interact with the simulation. This may involve designing and building a graphical user interface (GUI) using a tool such as Qt Designer or using a command-line interface (CLI) that allows the user to input commands and see the simulation results in a text-based format.

Overall, the process of designing and programming a motion planning simulation software involves some steps, including defining the functionalities and rules, implementing the code, testing the simulation, and creating the user interface. By following these steps, it is possible to create a software tool that can be used to simulate the motion and behavior of automatons in a user-defined environment (Fig. 1).

The first step to performing the simulation is to load the environment in which the robots interact (Figs. 2 and 3). In this configuration the simulator will give the option to load an environment that can be any file in JPG or PNG format that has a white background and no borders. Click on load environment, and it will direct it to the D:/ directory of the computer to select the desired environment.

The loaded environments will be located in the simulator executable folder, more specifically in the maps folder. These loaded environments, found in the maps folder, will enable a user to experience their custom simulations within the simulator executable (Fig. 4).

When selecting the environment the software will give the option of a treatment to include in the simulation. Thanks to the OpenCV library any loaded environment will change to a composition of white and black pixels, creating a new explorable environment for the robots which is named "1", clarifying as an explorable environment the white pixels and as obstacles the black pixels. This means that the environment can be changed, adapted, and customized to the robot's needs, making it a very versatile tool in terms of simulation possibilities (Fig. 5).

Once the environment is loaded, the size of the environment is defined based on the size of the image, which is 800×600. The next step is to define the parameters of the robot, such as the number of robots and their speed (Fig. 6). The default size is given by the robot belonging to the ARMOS research group, which has dimensions of 0.61 cm $\times$ 0.61 cm. To ensure that the environment is accurate, it is important to properly set the parameters of the robot. Therefore, it is important to select the proper number of robots and adjust their speed to ensure that they move at a suitable rate.

The number of robots defining the Quorum Sensing must be adjusted in coherence with the total population size. For the number of robots, the maximum is defined as corresponding to 4% occupancy, as it is considered the right number for a relevant threshold performing a task; the minimum number is defined at the user's discretion (Fig. 7). The speed will depend on the physical construction of the robot we want to simulate, this will be reflected in the scan time and the percentage of occupation of our robot in the loaded environment. To define the quorum threshold, the user is free to decide the number of robots needed to meet it. Therefore, the number of robots within the Quorum Sensing needs to be adjusted according to population size, with a minimum and maximum number defined.

To define the place where the quorum will occur, the regions in the environment must be defined with a corresponding performance value, which is assigned using weight values or landmarks, which will give information to

Figure 1

Path planning simulator interface.

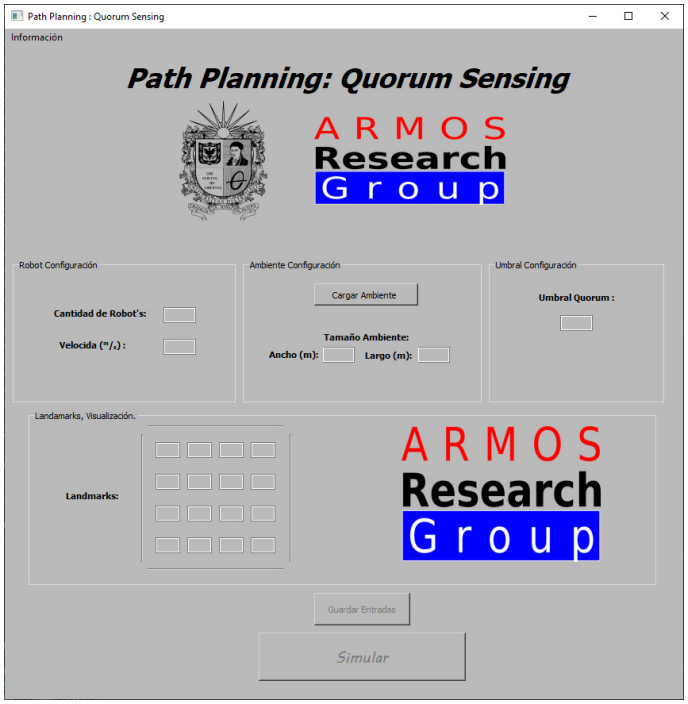
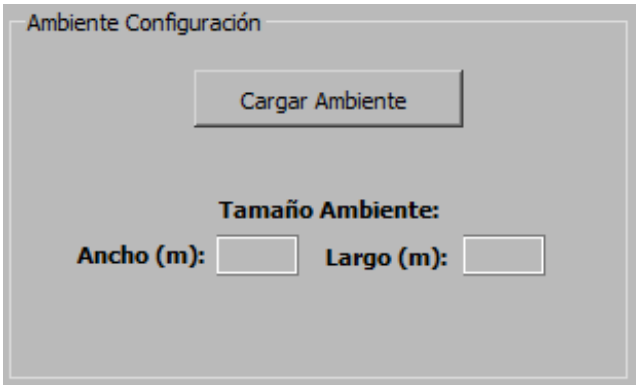


Figure 2

Environment configuration.



the robots about the importance of that region and the higher this value is, the more attractive it will be for the robots (Fig. 8). The value of each landmark will be a maximum of 10. Each region will be measured about the others, based on the criteria assigned to each of them so that a single region with a higher value is more likely to be chosen than several regions with lower values. Thus, the value assigned to each region will provide information about how attractive it is for the quorum.

Here the user will enter the weight values he wants for the regions of the environment and on the right side he will have a preview of the previously loaded environment. Once the basic values for the simulation are obtained the user will click on save, this will generate the values in the program base and then run the simulator in the Python Pygame platform. The simulator will process the user's weight values, which it reads from the program base, and then apply them to generate an environment on the Python Pygame platform.

Results

We will now review each of the functionalities offered by the program and its great versatility in representing different types of scenarios as required by the researcher. One of the possible variables when simulating is the size of the environment, to give context an environment of 100 m $\times$ 50 m and another of 75 m $\times$ 20 m will be simulated as shown in Fig. 9 and Fig. 10 respectively. These scenarios will be compared to one another, taking into account the difference in size and its effect on the results obtained.

As can be seen in the figures above, the environment did not change in size or shape, as it remains constant regardless of the size of the environment. However, the

Figure 3

Environment selection directory for the simulator.

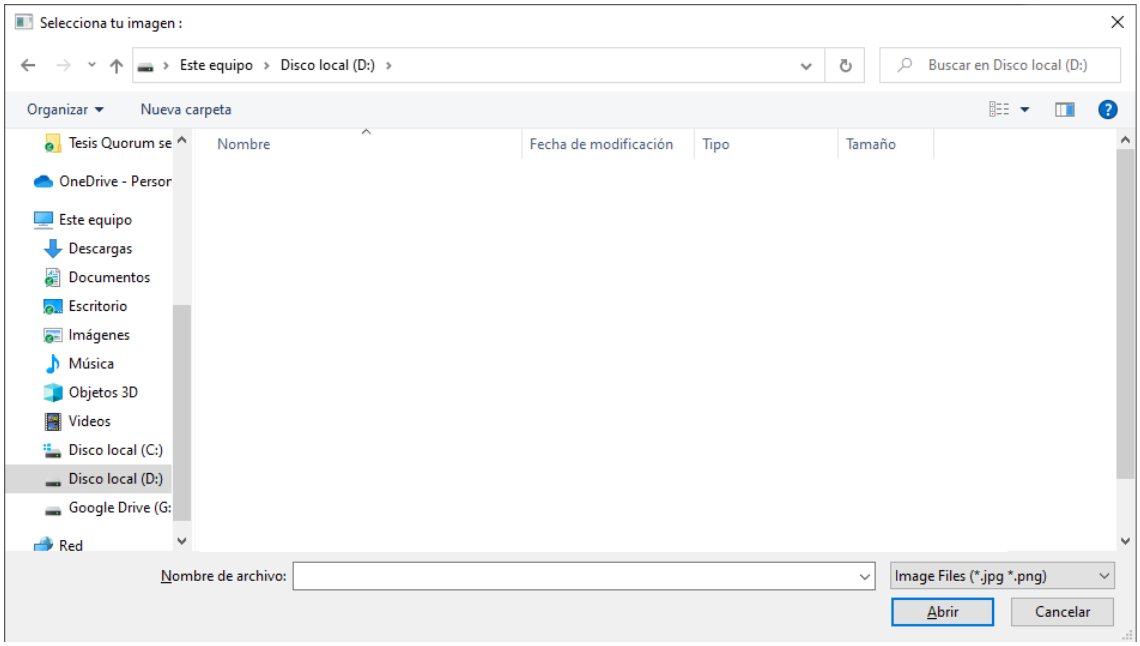


Figure 4

Pre-loaded environments for the simulator with their respective locations.

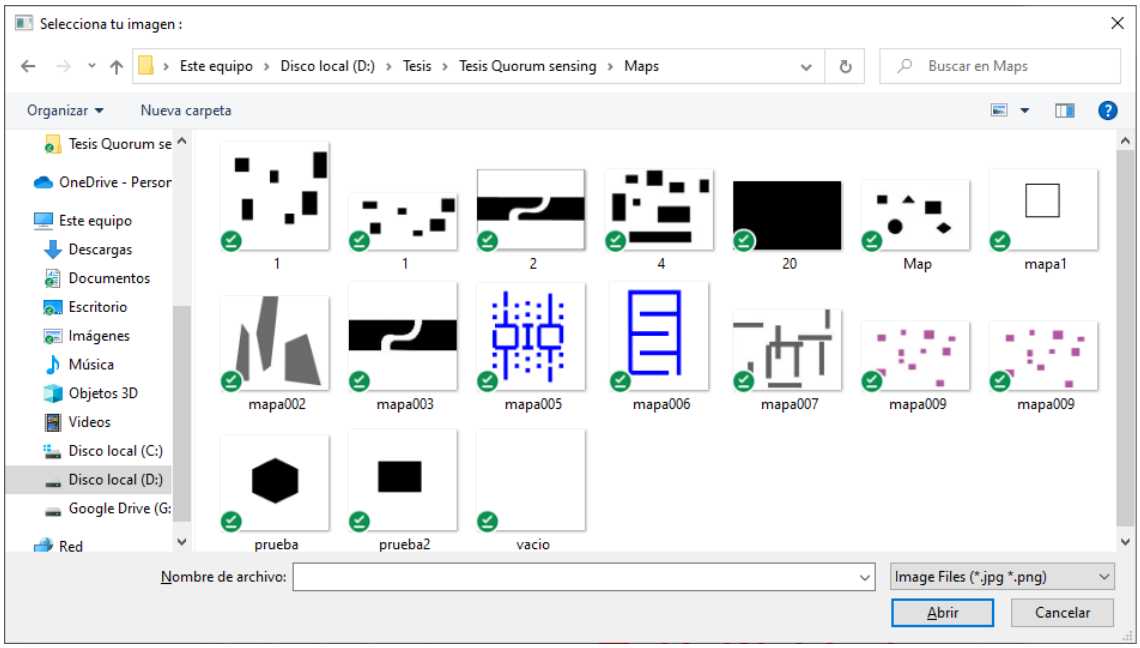


Figure 5

Environment modified and saved by the simulator.

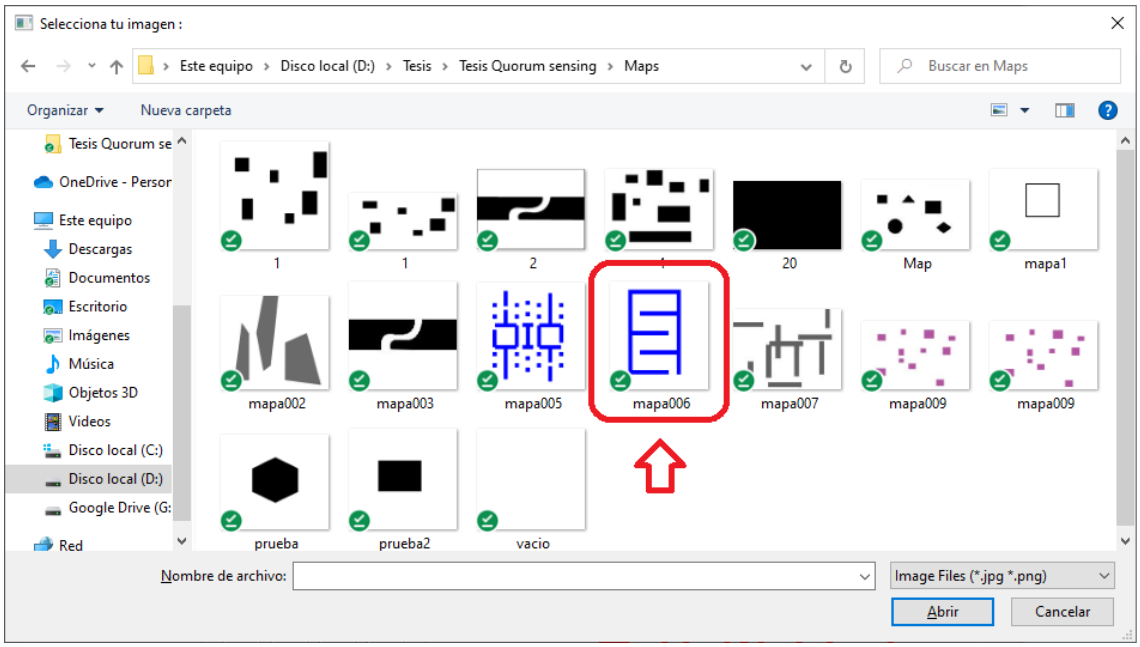


Figure 6

Configuration of robot parameters.

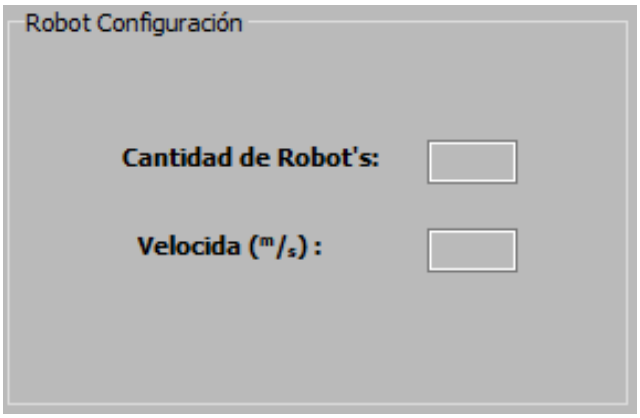
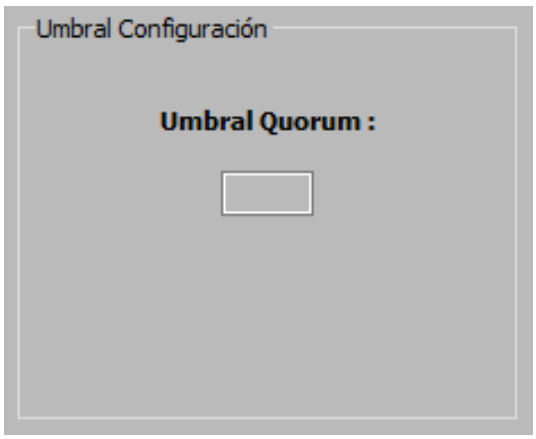


Figure 7

Setting the quorum threshold value.

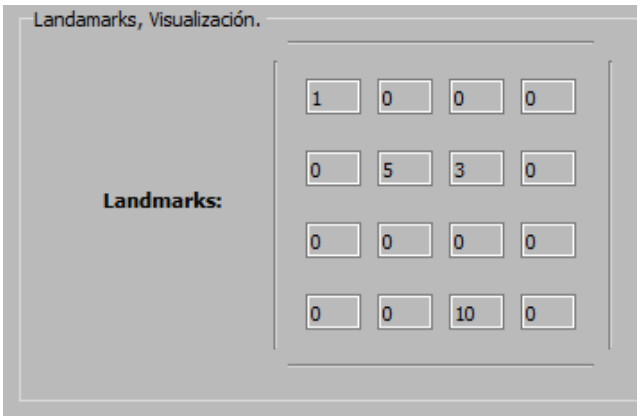


robots changed as the size of the environment changed, by having a larger area in the environment the robots look smaller in comparison, this means that the true size of the robots remains constant regardless of the size of the environment. The scale of the environment changes according to the simulation need, but the space used on the screen is optimized. The optimized size of the environment provides users with an improved visual representation that is easy to interact with.

To simulate various types of scenarios, our program allows loading different types of environments in .jpg or png format as shown in Fig. 11 and Fig. 12. The user can use the drop-down menu to select one of the many pre-existing environments or load a new environment of their own.

Figure 8

Assignment of landmarks to the simulator as a preview of the loaded environment.



In a navigation environment, it is important to define the number of robots that will perform the search task. With this objective in mind, the program allows the implementation of several robots that do not exceed 4% of the area of the environment. In this case, 35 robots were implemented in an obstacle-free environment of 75 m × 20 m as shown in Fig. 13.

To implement Quorum Sensing the robots will move using wild movement. Through the initial interface a certain quorum threshold is determined in a cell chosen by the user and weight is given to this zone, depending on this weight stipulated by the user, the robots that exceed the quorum threshold in a zone will wait a time directly proportional to the weight of the zone. Fig. 14 shows how several red robots are concentrated in different zones of interest, blue robots are waiting for more robots to complete the quorum and green robots are scanning the environment in search of a zone of interest. The robots' movements are chaotic, but the Quorum Sensing algorithm helps to control their overall behavior by determining the necessary quorum for a given region. In this way, the robots can efficiently explore a given area and detect regions of interest.

Once the user wishes to finish the simulation, he/she can click on the `Finish` button to enter the results screen, where he/she will find data of interest such as the number of robots used, the environment, and its divisions, the total simulation time, the time it took to reach quorum in any zone and the concentration of robots per zone of interest, this concentration of robots is given with a color palette explained in the upper left part of the results screen as shown in Fig. 15. On this results screen, the user can analyze the performance of the robots in the environment by checking how much time they needed to reach a quorum, or how many robots were used.

Despite the efforts made to create a simulation that matches real life, it was not possible to achieve accurate collisions between robots and environments due to the limitations of Pygame. Although the collisions work well most of the time, sometimes the robots can overlap with the objects and get trapped inside them, however, this problem could be solved by using environments included in the program by implementing fixed and well-determined hitboxes; when a user loads his environment it is not possible to accurately determine these hitboxes, so this kind of problems can occur. Thus, the only way to guarantee a more accurate collision between the robots and their environment is to create a simulation using different programming languages or libraries which are designed specifically for game development and animation, as these provide greater control over the physics of collisions. Therefore, while Pygame provided a useful framework to create a simulation and enable the robots to move realistically, the limitations of this program meant that it was not possible to guarantee an accurate collision between robots and their environment.

Conclusion

The use of Python as design software for the code used facilitated the process in the generation of both the robot and its interaction with the environment since thanks to its object-based programming it is possible to add qualities that in other software may take a longer process. This object-based programming made it easier to implement changes to the robot's behavior, without having to completely re-program it.

The versatility of the software to perform the simulation on environments established by the user generates endless possibilities; having only as a restriction that the environment is not with margin since it is one of the limitations of the software. This capability, along with the other features of the software, allows users to craft a wide range of simulations, limited only by their imaginations and the capabilities of the software.

The software gives positive results when performing the navigation simulation for a group of autonomous robots through the movement strategy formulated and applied by the research group. This strategy derives its working principle from a simplified model of bacterial interaction; these results are shown in the quorum values reached by the software and previously introduced by the user; these values have a direct interaction with the environment as well as the other values defined by the user. The software produces robust results, with the quorum values closely mirroring those which have been defined by the user. This is a testament to the effectiveness of the software, as it can accurately calculate the interactions between

Figure 9

100 m × 50 m navigation environment.

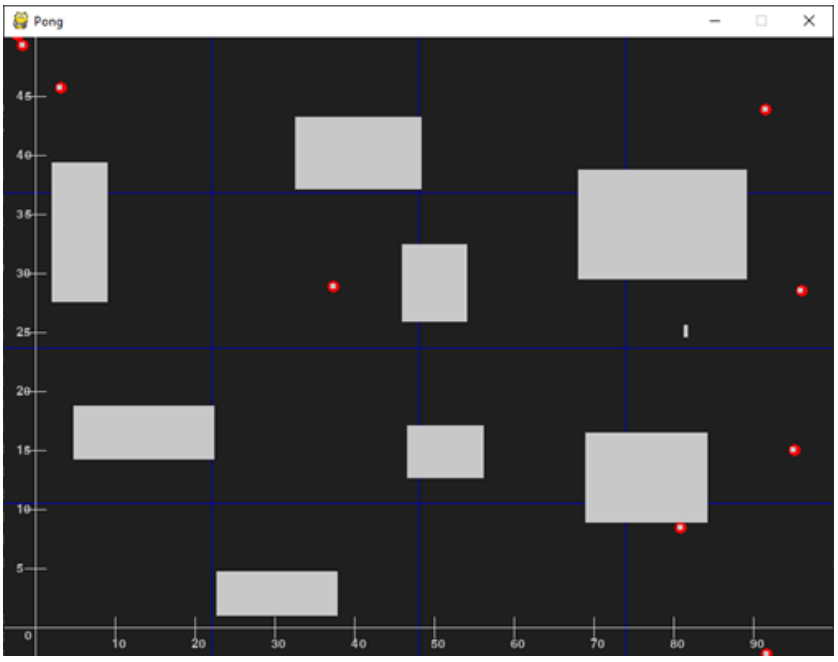


Figure 10

75 m × 20 m navigation environment.

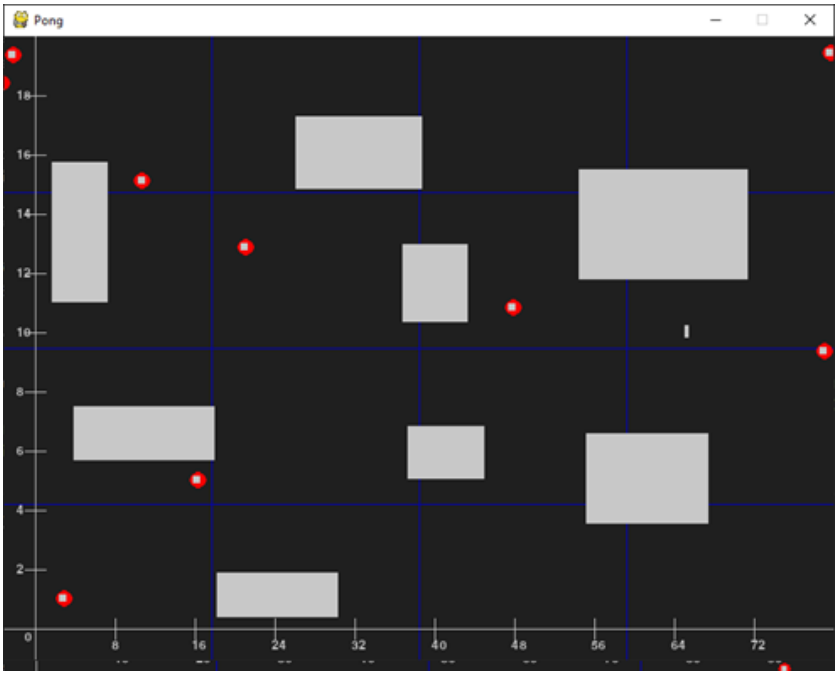


Figure 11

Loading of different environments (1).

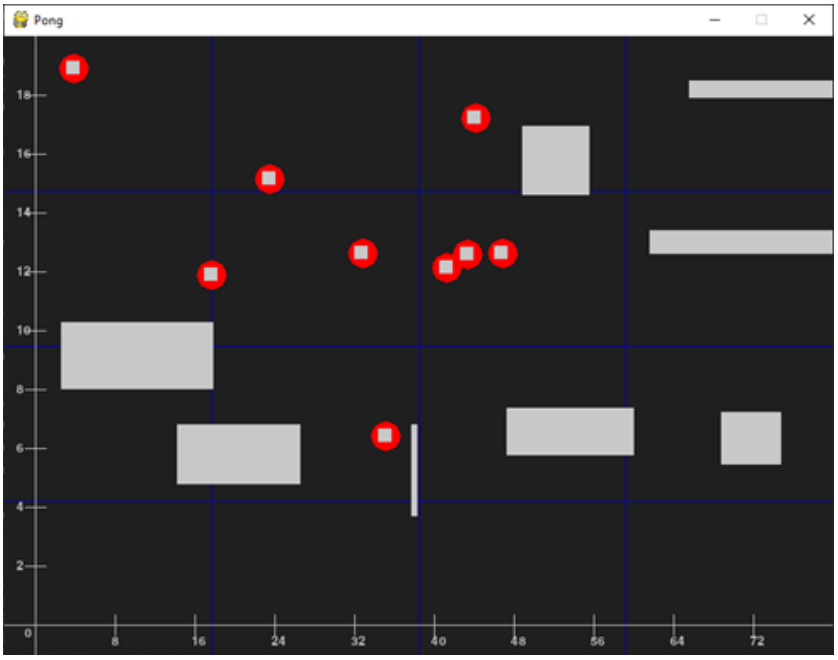


Figure 12

Loading of different environments (2).

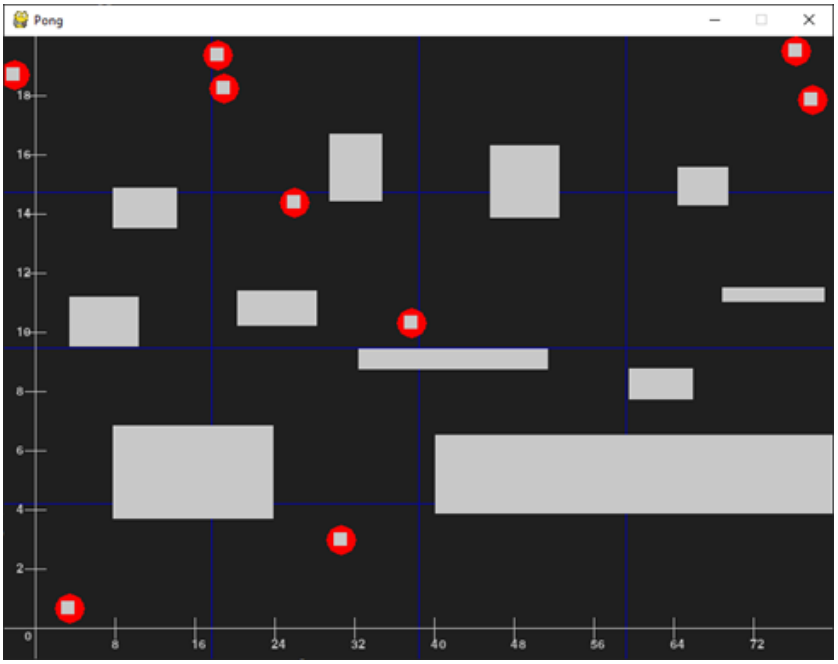
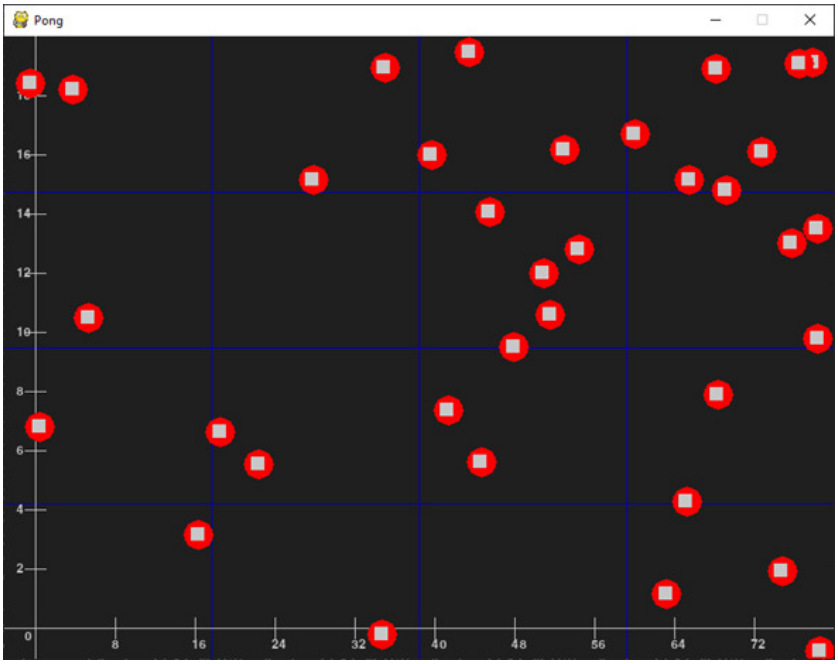


Figure 13

Loading 35 robots in a simulation.



various parameters, and generate robust results following the previously defined values.

References

Bobadilla, L., Martinez, F., Gobst, E., Gossman, K., & LaValle, S. M. (2012). Controlling wild mobile robots using virtual gates and discrete transitions. *2012 American Control Conference (ACC)*. https://doi.org/10.1109/acc.2012.6315569

Cody, J. R., & Adams, J. A. (2017). An evaluation of quorum sensing mechanisms in collective value-sensitive site selection. *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. https://doi.org/10.1109/mrs.2017.8250929

Ferrante, E., Turgut, A. E., Huepe, C., Stranieri, A., Pinciroli, C., & Dorigo, M. (2012). Self-organized flocking with a mobile robot swarm: A novel motion control method. *Adaptive Behavior*, *20*(6), 460–477. https://doi.org/10.1177/1059712312462248

Flikkema, P., & Leid, J. (2005). Bacterial communities: A microbiological model for swarm intelligence. *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.* https://doi.org/10.1109/sis.2005.1501655

Gómez, P., & Rodríguez, A. (2011). Simulating a rock-scissors-paper bacterial game with a discrete cellular automaton. In *New challenges on bioinspired applications* (pp. 363–370). Springer Berlin Heidelberg. https://doi.org/10.1007\/978-3-642-21326-7_39

Hsieh, M. A., Halász, Á., Berman, S., & Kumar, V. (2008). Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence*, *2*(2-4), 121–141. https://doi.org/10.1007/s11721-008-0019-z

Huang, C. X., Zhang, B., Deng, A.-C., & Swirski, B. (1995). The design and implementation of PowerMill. *Proceedings of the 1995 international symposium on Low power design - ISLPED '95*. https://doi.org/10.1145/224081.224100

Ichter, B., Harrison, J., & Pavone, M. (2018). Learning sampling distributions for robot motion planning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*. https://doi.org/10.1109/icra.2018.8460730

Khan, A., Noreen, I., & Habib, Z. (2017). On complete coverage path planning algorithms for non-holonomic mobile robots: Survey and challenges. *Journal of Information Science & Engineering*, *33*(1), 101–121.

Korani, W. M. (2008). Bacterial foraging oriented by particle swarm optimization strategy for

Figure 14

Robots exploring an environment.

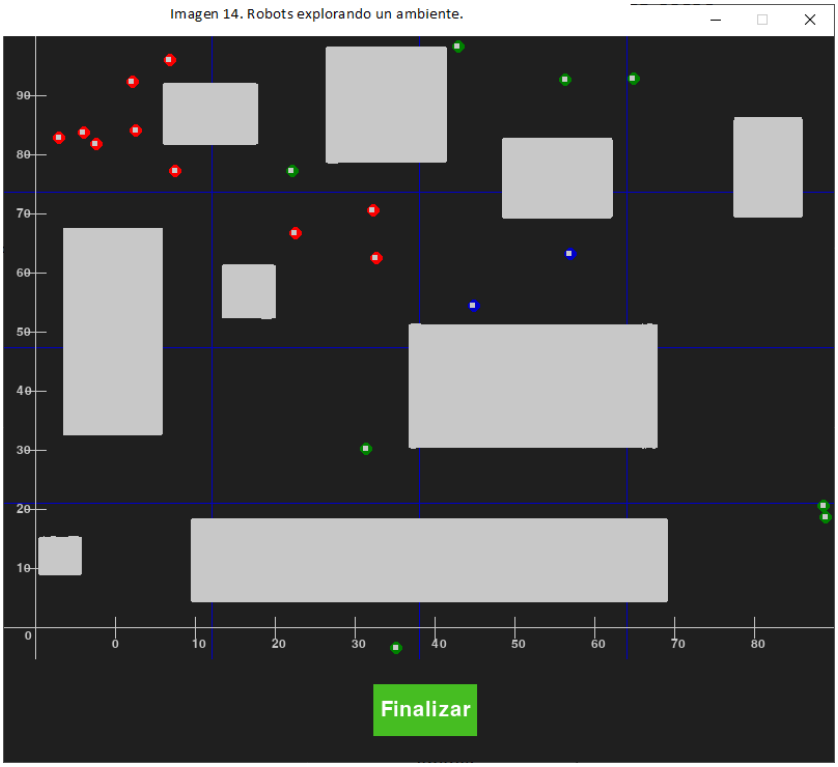


Figure 15

Results screen.

PID tuning. *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation - GECCO '08*. https://doi.org/10.1145/1388969.1388980

Lo, C., Wei, G., & Marculescu, R. (2015). Towards autonomous control of molecular communication in populations of bacteria. *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication*. https://doi.org/10.1145/2800795.2800822

Martínez, F., Jacinto, E., & Hernández, C. (2012). Particle diffusion model applied to the swarm robots navigation. *Tecnura*, *16*(2012), 34–43.

Martínez, F., Martínez, F., & Montiel, H. (2020). Bacterial quorum sensing applied to the coordination of autonomous robot swarms. *Bulletin of Electrical Engineering and Informatics*, *9*(1), 67–74. https://doi.org/10.11591/eei.v9i1.1538

Mohanan, M., & Salgoankar, A. (2018). A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, *100*(2018), 171–185. https://doi.org/10.1016/j.robot.2017.10.011

Patle, B., L, G. B., Pandey, A., Parhi, D., & Jagadeesh, A. (2019). A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, *15*(4), 582–606. https://doi.org/10.1016/j.dt.2019.04.011

Roozbahani, H., & Handroos, H. (2019). A novel haptic interface and universal control strategy for international thermonuclear experimental reactor (ITER) welding/machining assembly robot. *Robotics and Computer-Integrated Manufacturing*, *57*, 255–270. https://doi.org/10.1016/j.rcim.2018.12.011