Journal of Theoretical and Applied Information Technology
15th March 2021. Vol.99. No 5

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

APPLICATION OF CUSTOM-MADE SIMULATOR FOR TRAINING IN ROBOTIC NAVIGATION STRATEGIES

[1]JORGE DUARTE, [2]YEISSON TRIVIÑO, [3]FREDY MARTÍNEZ

[1,2,3]Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá D.C, Colombia

E-mail: [1]jaduartea@correo.udistrital.edu.co, [2]yftrivinor@correo.udistrital.edu.co, [3]fhmartinezs@udistrital.edu.co

ABSTRACT

The social needs that require the support of robotics are increasingly evident. Tasks such as caring for children and the elderly in isolated, unsupervised homes have become evident following the demands of social isolation caused by the COVID-19 pandemic. Service robotics is presented as a current and readily available solution for the remote monitoring of people, particularly those with medical restrictions or in education processes. However, the development of these applications requires a critical mass of professional personnel trained in the design of these robots and their navigation and manipulation schemes. Current robotic systems tend to be expensive, so their availability at the university level is limited. Its use aims to be dedicated to research, leaving aside the processes of training at the undergraduate level. A very efficient way to develop a specialized training in robotics for young researchers is through the use of software tools that simulate existing robotic models in the laboratory. This article presents the formulation and development of a software tool designed to train young researchers within the research group. The tool is the first of a knowledge management system that the research group has proposed to encourage and accelerate research in service robotics. This first tool focuses on the problem of autonomous planning of movement in globally observable environments, similar to those normally foreseen for service robots. Specifically, it implements the algorithm of visibility graphs to define the navigation route of a robot from a point of origin to a point of destination. As result, it is presented the final operation of the tool, its advantages in terms of configuration and manipulation, and its capacity to evaluate the performance in front of different variables of the problem. This article is developed as part of the group's research in mobile robotics and image processing, specifically related to the training of junior researchers.

Keywords: *Research Process, Robotics, Self-Directed Learning, Simulation Model, Specialized Training*

1. INTRODUCTION

It is widely accepted that learning activities correspond to the most critical elements within an educational process [1, 2, 3]. When designing a training scheme, the first step of the process is to define the learning objectives, which are intimately linked to the learning process to be generated in the students [4, 5, 6]. This is particularly important today when many of the training needs are linked to technological developments [7, 8]. Technology is fundamental not only as a training concept and strategy for the construction of teaching material but as a general strategy in the design of an effective and significant learning process.

Specific purpose simulators are a powerful training tool. This software can be designed to acquire and perfect specific skills that are difficult or complex to develop through other specialized laboratory strategies [9, 10, 11]. This is especially true when the training purposes are not strictly framed within a formal curriculum, as is the case of advanced work within the research seedbeds of a research group. They also turn out to be key tools for the validation of schemes and algorithms before their implementation on real prototypes [12, 13]. Also, they propose a training scheme that strengthens the self-training and critical skills of the student, since they stimulate a process in which direct interaction between teacher and student is not necessary [14].

One of the great strengths of the use of simulators for training is the great adaptability that is achieved in training, which allows consolidating learning gaps specific to each of the students [15, 16, 17]. These gaps are difficult to identify in the initial assessment of students but can be autonomously solved by them when they have at their disposal a tool configurable to their needs and training desires. This also provides an additional

advantage, which makes the training process more attractive to the student, given the lack of direct linkage to training schedules and locations, in addition to the fact of interacting with a friendly technological tool [18]. These characteristics make the software tools a strong and positive modifier of study habits [19, 20]. The approach to the tool is given by a strong influence of the student's interest in the subject, which ends up affecting their daily habits, and therefore promotes the construction of a culture of self-training.

Each student has his or her structure in terms of how he or she learns [21, 22]. Habits, schedules, skills, tools, and different ways to approach bodies of knowledge. Skills such as self-management of material, time, resources, task development, as well as critical analysis of available information, are aspects and skills that are encouraged and strengthened by the targeted use of these software tools. Students can develop their training process according to the strategies proposed by the teachers, but at the same time investigate functional variants according to their interest. This undoubtedly impacts both the student's habits and the quality of their learning process when building new training strategies [23, 24, 25].

This project was developed to support the specific training of young researchers within the research group. The central tool of the project corresponds to a set of simulators for specific use in service robotics. As a prototype, the implementation of the geometric strategy of path planning in observable environments using the algorithm of visibility graphs was selected. The challenge to obtain an optimal solution in motion planning is contemplated in two crucial parameters, path planning (geometric) and trajectory planning (control). The objective is to indicate the minimum path between the initial point and the arrival point and to calculate the point that the robot must reach at each instant of time, respectively, achieving a correct trajectory. Correct means that the trajectory does not collide with the environment and that it complies with the kinematic (and ideally dynamic) constraints of mobile robots.

Even though the use of the software requires dedication and supplementary guidance from expert teachers, the continuous use of the tool not only attacks specific deficiencies in the student but also creates study habits because the continuous repetition of certain practices modifies the individual's behavior and self-motivated autonomous work encourages critical thinking and self-regulation.

2. RELATED WORK

The traditional strategy used in most of the courses related to robotics is centered on lectures, master classes, and practical laboratory activities [26]. Under this scheme, and according to the limitations of the laboratory, it is created a learning gap in the student when he tries to bring down the theoretical concepts in his laboratory practices. In the laboratory, the student has access to real robots on which it is expected to implement strategies and algorithms, but in principle, the implementation of such algorithms on the real platforms is not direct or easy, which makes it difficult to derive the theoretical conclusions on a functional prototype. The tool more powerful and versatile commonly used to close this gap is the simulators software, tools of special-purpose capable of facilitating the implementation of the algorithms and to project in a virtual environment the scenario of performance of the robot under the specific programmed conditions [27]. These tools also reduce the time spent on applied research, since they allow a rapid validation of algorithms before programming in real platforms [28]. Masterclasses and lectures do not bring the student closer to the real problems they face when working a given algorithm on the robot, making the laboratory practices focus much of the activity on the knowledge and handling of equipment, rather than on the implementation and evaluation of algorithms. A simulator can greatly reduce these problems, in addition to allowing the student to work in a safe environment that leads to personal experimentation.

The existing simulators in robotics for use in education and research can be separated into two categories [28]. On the one hand, there are the commercial simulators, characterized by a closed code development oriented to commercial use of the tool. On the other hand, there are open-source projects, which include a large number of initiatives from research centers and groups. The problem with commercial tools lies in their high cost and the impossibility of modifying their internal code to suit the training and/or research process. Many higher education institutions cannot afford the high cost of commercial software, even more, when it is necessary to think about the number of licenses according to the student population, and annual updates of the same. In many cases, the choice is to invest in hardware and software tools that benefit the majority of the student population, so special-purpose simulators are not usually viable options. Furthermore, in applied research, it is a requirement to be able to access the routines of a tool to adapt

them to the particular conditions of the experiment, which is impossible in a commercial tool.

In the commercial category, three platforms can be easily identified: Webots [29, 30], MATLAB [31, 32], and Easy-Rob [33, 34]. Webots is a simulator from Cyberbotics Ltd. developed for three-dimensional environments. Although it was born as a closed-source commercial platform, today it is an open-source tool, the commercial business was oriented towards industrial and academic support. A robot in Webots can be equipped with physical characteristics similar to those of the real robot, including a wide variety of sensors and actuators. The control code can be written in C, C++, Java, Python, and Matlab. The tool is compatible with ROS (Robot Operating System), as well as commercial robotic platforms such as Epuck, Pioneer, Lego Mindstorm, Sony AIBO, and Fujitsu HOAP-2. One of the most outstanding features of Webots is that it allows the user to interact with his robot throughout the simulation.

MATLAB is perhaps the most widespread and well-known academic tool. It is developed and marketed by MathWorks and is structured around the concept of MATrix LABoratory and a proprietary programming language. It has a large number of add-ons called Toolboxes that extend the capabilities of the software in specific domains. Some of these Toolboxes are Simulink, Robotics, and SimMechanics which allow the kinematic and dynamic simulation of any robotic structure. However, the problem of this tool is precisely its commercial nature, this feature makes it complex to use outside the restrictions of its license, which prevents the free dissemination of code developed with it, even for the documentation of research articles. Finally, Easy-Rob is developed and marketed by EASY-ROB Software GmbH and is intended to be a complete low-cost simulation solution. This tool is oriented towards the planning and verification of work cells, which does not exclude that other types of platforms can be simulated within their three-dimensional environment.

Within the open-source options the most important, and open distribution are Player/Stage [35, 36, 37], Gazebo [32, 27, 38], Robot Operating System (ROS) [27, 38, 39], Simbad [40, 41], Carnegie Mellon Robot Navigation Toolkit (CARMEN) [42, 43], Unified System for Automation and Robot Simulation (USARSim) [44, 45], Microsoft Robotics Developer Studio (MRDS) [46, 47], and MissionLab [48, 49].

Player/Stage is a tool designed to replicate the behavior of a robot according to the interaction of its sensors and actuators. It was developed by the International Team of Robotics Researchers and has a server/client structure that allows configuring multiple capacities to the robot using the communication with the server. The Player module is the interface for the definition of the robots (Hardware Abstraction Layer, HAL), while Stage provides the two-dimensional simulation environment (Robot OS). Gazebo is a simulator of three-dimensional environments originally developed as a component of Player/Stage, but currently is a standalone tool supported by Willow Garage. The simulator is capable of performing realistic renderings of environments, as well as modeling sensors that identify the simulated environment. The objects simulated in Gazebo have mass and other physical attributes such as friction or contact.

Robot Operating System (ROS) is a framework developed by the Stanford Artificial Intelligence Laboratory, with current development by Willow Garage, which works as an operating system to the robot to reduce the direct control of low-level devices of sensors and actuators. It has a large community of support and allows the development in C, C++, Python, Octave, and LISP. Simbad is another open-source tool that can simulate the interaction of individual robots or groups of robots in a three-dimensional environment. It is developed in Java, but it also supports Python, and as in the previous cases it also supports a wide variety of sensors. This simulator was developed to investigate the use of tools based on artificial intelligence, so it has libraries for neural networks and evolutionary algorithms.

Carnegie Mellon Robot Navigation Toolkit (CARMEN) is a collection of tools designed for robot control. With this tool, it is easy to implement navigation strategies, path planning, and mapping. Although its simulation environment is two-dimensional, its simplicity and support for multiple platforms and sensors make it a high-performance tool when evaluating custom strategies. It supports C and Java and is limited to use on Linux. Unified System for Automation and Robot Simulation (USARSim) is a three-dimensional environment simulator based on the Unreal Tournaments game engine. Its objective was the search and rescue applications at the urban level, as well as the study of the human-machine interaction. The control code is developed using the GameBot interface, the Mobility Open Architecture Simulation and Tools

System (MOAST), Player interface, or MATLAB through the Toolbox USARSim.

Microsoft Robotics Developer Studio (MRDS) is another three-dimensional environment simulator with extensive support for robots, sensors, and actuators. While this tool does not have its open-source but is available to the public for free use. The great characteristic of this simulator is that it uses a language of programming visual (VPL) for the control of the robot. Finally, MissionLab is a three-dimensional tool developed by the Mobile Robot Laboratory at Georgia Tech for Linux systems. It is a multi-agent simulator with individual reaction capability, and support for multiple commercial robots. Other interesting projects worth mentioning include the Robocup soccer simulator, EyeSim for the EyeBot Robot, SimBot and WoB.

3. METHODS

The training of researchers is a complex task that is part of the activities of a research group. In our particular case, we have detected our problems in the understanding and application of different schemes and algorithms in robotics. Among the fields with major problems are those of path planning, model construction, application of inverse kinematics, and the study of our algorithms proposed by the research group such as Quorum Sensing (QS). The research group has implemented some strategies to reduce this gap, among these strategies is the development of customized simulators that cover specific aspects of the implementation of algorithms, path planning with visibility graphs is one of the first tools under development. This research has opted for a configurable graphical tool since it was identified in the students the impossibility of interacting with the algorithms simply and reliably and under the conditions of our laboratory and our robots. Among the variables of interest was the student's ability to identify the performance of a given strategy under specific conditions, and the restrictions that each strategy imposes on the robotic platforms available in the laboratory.

Visibility graphs provide a geometric approach to solving the problem of path planning. This method operates with polygonal models of the environment. It is a widespread method, and some algorithms build this kind of graph. It is limited to models of environments defined as polygons and can work both in the plane and in space. In our research, we use the simulation in two dimensions (2D - plane). The plane guarantees to generate graphs that contain the optimal path, although this does not happen in space. They have the disadvantage of fitting too close to obstacles, so they require a safety expansion system. In a set of different polygonal obstacles located in a plane, the visibility graph will be formed by the unions of the pairs of vertices that are seen mutually, without any intersection with the obstacles (Figure 1) [50, 51].

Formally, a G graph consists of two finite sets $\{N\}$ and $\{A\}$. $\{N\}$ is the set of graph elements, also called vertices or nodes. $\{A\}$ is a set of arcs, which are the connections that relate the nodes to form the graph. Arcs are also called edges, lines, or paths. Nodes are used to represent objects, and arcs are used to represent the relationship between them. For example, nodes can represent countries and arcs can represent the existence of roads that communicate them. It is said that two nodes are adjacent or neighbors if there is an arc that connects them [51].

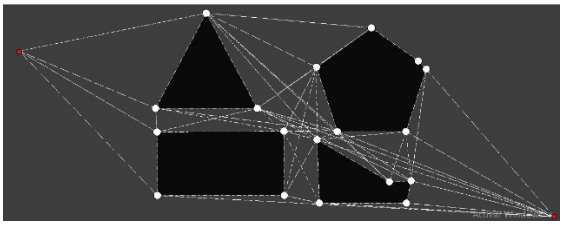

Figure 1: Example of visibility graph modeling

The selection of the path among the multiple options defined by the visibility network requires the use of an optimization algorithm. In our first version of the software, we have chosen to implement Dijkstra's algorithm [52, 53]. The algorithm is also known as the minimum path algorithm and determines the shortest path from a vertex or origin node to the rest of the nodes in a graph with weight values in each path [53]. The idea is to explore all the shortest paths from the origin vertex to all other vertices. When the shortest path from the origin vertex to the rest of the vertices that make up the graph is obtained, the algorithm stops.

Figure 2 shows the methodological sequence applied in the research for the generation of visibility graphs of an unknown environment with the aim of building the optimal path. There can be identified six fundamental stages: Determination of the environment or workspace, establishment of the scale of the workspace and dimensions to the robot, indication of the initial point and the final point, identification of the vertices of the objects in the environment, projection of the possible solution

trajectories, and selection of the optimal path from the Dijkstra algorithm.

All the formulation, design, and implementation of our tool followed the principles of minimalism, open access, and the possibility of modification and adjustment of routines at the user's discretion, this includes not only the programming language but also the distribution and documentation tools. Our simulator was developed in Python version 3.8, and the default options are adjusted to the features of our robotic platform ARMOS TurtleBot [54]. However, the environment is fully configurable, which means that other platforms can be implemented, as well as different navigation environments can be configured. The user selects the workspace as a graphic file in JPG and PNG format. In this file, the two-dimensional design of the environment, including obstacles, must be provided. The length restriction of the obstacles of the workspace must be higher than five pixels, that is, the program only accepts that the obstacles will be polygonal, it is not allowed that the workspace contains a line thickness lower than five pixels. The user must also indicate the scale of the workspace. With this information, the navigation environment within the simulator is built. The graphic manipulation (morphological operations, labeling, filters, and edge detection) is done through OpenCV. We make a simplification of vertices using the Douglas-Peucker algorithm.

After establishing the simulation environment, the user can set in the environment the starting point and the finishing point for the robot navigation. The sequential form to establish the points is the initial to the end. Due to the configuration of the program, in the case that the user omits to incorporate these points, the program will not enable the route search option. The user can define the number of nodes according to the need and complexity of the environment, i.e., increasing or decreasing the local minimums.

The grouping of the vertices of the obstacles is defined within a vector. This simplifies the handling of coordinates, particularly for the initial and target points (Figure 3). The design of the user interface (GUI) was developed with Tkinter. The vectorization of information from the navigation environment also facilitated the graphic design in Tkinter. From the width and length of the workspace, an image matrix is established where each of the coefficients is determined by the value of each pixel in grayscale to 8 bits.

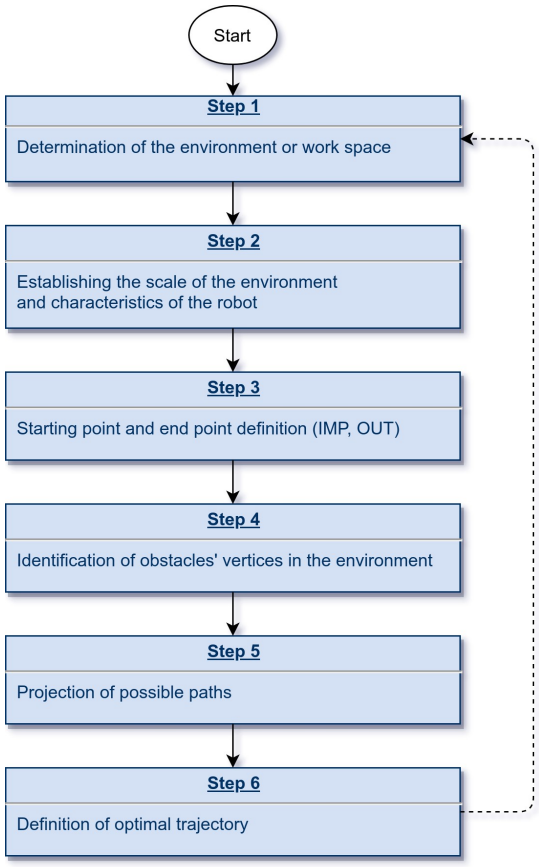

Figure 2: Chart of implemented methodology

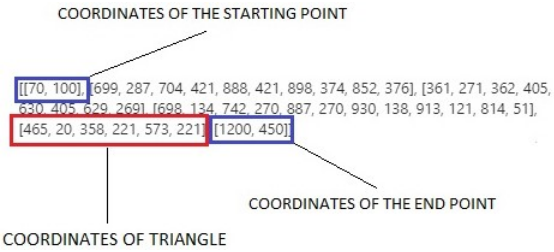

Figure 3: Encoding of navigation environment information, this example includes an obstacle, and the start and target points

According to the theory of visibility graphs, each of the vertices of the obstacles in the environment is obtained, and the lines of the graph are projected. These lines are divided into three groups:

- Obstacle action lines: These are the lines that join the vertices of the same obstacle.
- Lines of action between obstacles: These are the lines that join the vertices of obstacle *i* with obstacle *j*.

- Lines of action between points and obstacles: Are the lines that join the obstacles of the workspace with the initial and target points.

These lines of action overlap the obstacles, so it is necessary to segment them according to the location of the obstacles, this is done with the Bresenham algorithm.

Finally, with the corresponding grouping, the assignment of the weight matrix, and facilitating the verification of adjacent nodes in the system, we use the obstacle matrix to obtain the optimal trajectory through Dijkstra's algorithm. In Tkinter's interface, the optimal trajectory is highlighted, joining the starting point with the arrival point.

4. RESULTS

The results of the tool were measured and evaluated from two perspectives, firstly from its ability to meet the design profile defined by the research group, and secondly, in the medium term, according to its capacity to effectively shape a specific training tool that helps young researchers to reduce conceptual problems while facilitating the transition to real platforms and proposing an evaluation scheme for novel strategies. Most of the results shown in this section correspond to the first perspective, however, we already have some results from its use with students that derive into possible future adjustments of the tool.

The simulator has a clean and simple interface with the basic options at the file level: *Open*, *Save*, *Delete*, and *Exit* (Figure 4). The simulator has a set of basic environments in graphic format, but the user can load his designs. These designs are opened from the Open option, the Save option allows to store the information of the optimal path calculated in a simulation.

The graphic file corresponding to the selected navigation environment is shown in the central part of the simulator (Figure 5). To the image of the environment is added the dimensional information of the robot, real physical constraints of the robot, which are represented by circles. The circular geometric models of the robot are placed bordered the obstacles of the environment to establish the real free space, and to guarantee the not collision of the robot. After loading the environment, it is proceeded to adjust the parameters of the simulation, this is carried out in the section *Settings*. This stage is fundamental for the simulation and its analysis. The conditions of the scale of the space of work and dimensions of the robot are introduced.

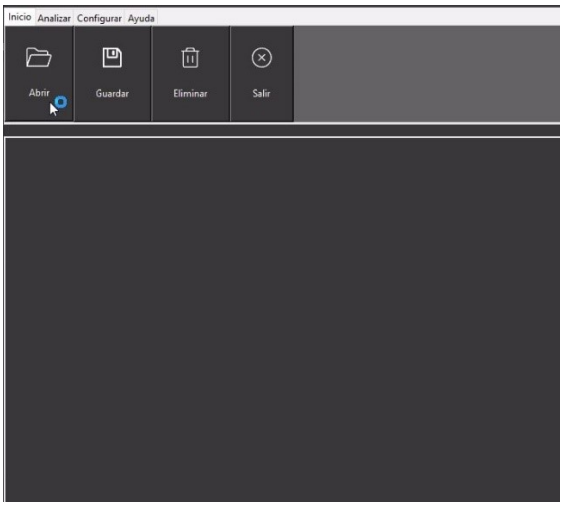

Figure 4: Simulator graphic interface

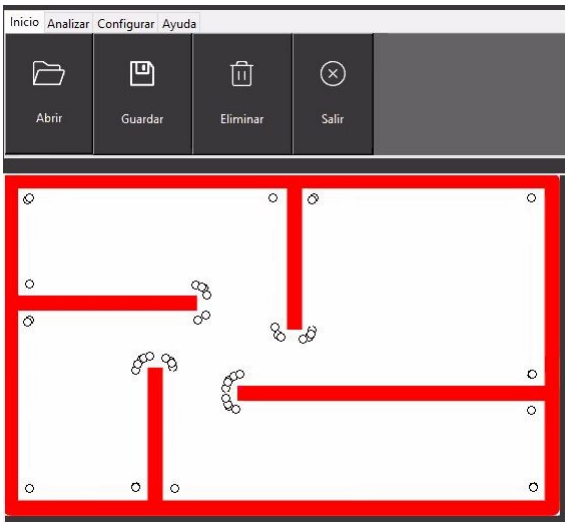

Figure 5: Workspace treatment

In the *Analyze* section, there are options to define the initial navigation point, the target point, the option to start the route search, and the selection of the strategy to define the shortest route. The program gives the freedom to drag the initial and final point, to be user friendly, without the need to insert the coordinates of each point. Once the points are indicated, the user must start the calculation of the possible solution routes.

When the option to search for routes is selected, the program draws a dotted line border over the navigation environment (Figure 6). When the software finds the optimal route, the path is displayed in the environment by linking the start and end points (Figure 7).

Journal of Theoretical and Applied Information Technology
15th March 2021. Vol.99. No 5

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

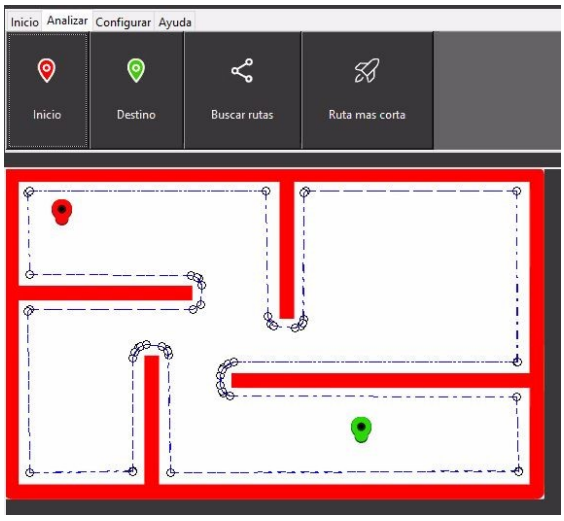

Figure 6: Simulator looking for possible navigation routes

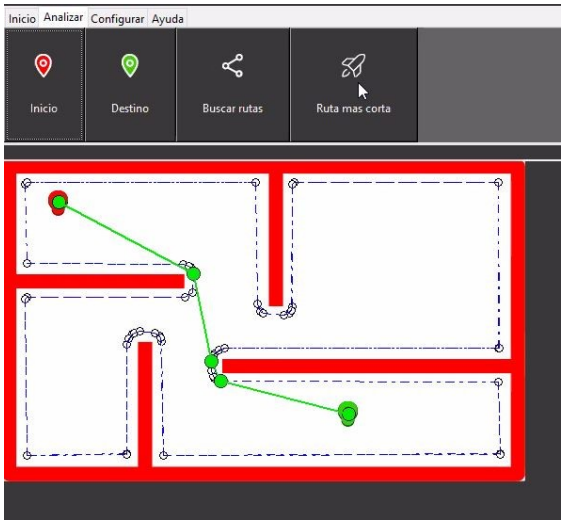

Figure 7: Simulator indicating the optimal route after applying visibility graphs and Dijkstra

For performance evaluation, we use different navigation environments and different robotic platforms (different sizes and travel speeds). In the first case we used a robot of 0.2 m by 0.2 m, the starting point was placed in the upper left, behind the two large obstacles, and the end point in the center right, again behind a third obstacle (Figure 8). The optimal route found by the simulator is shown in Figure 9. The figure shows how, in this case, it manages to define a path along with the narrow space between the two obstacles.

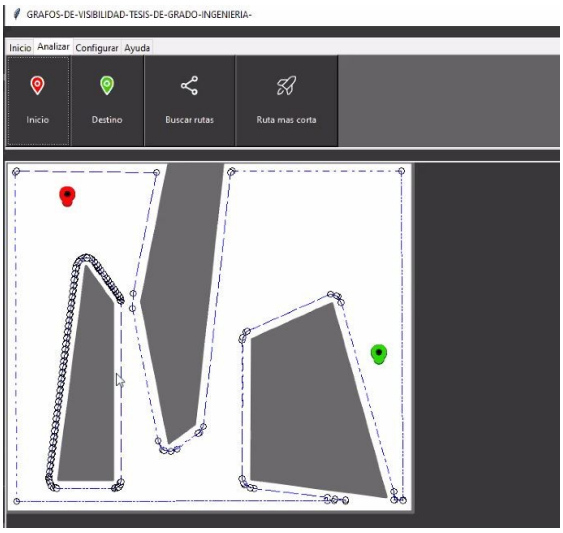

Figure 8: Environment setting for the first performance test

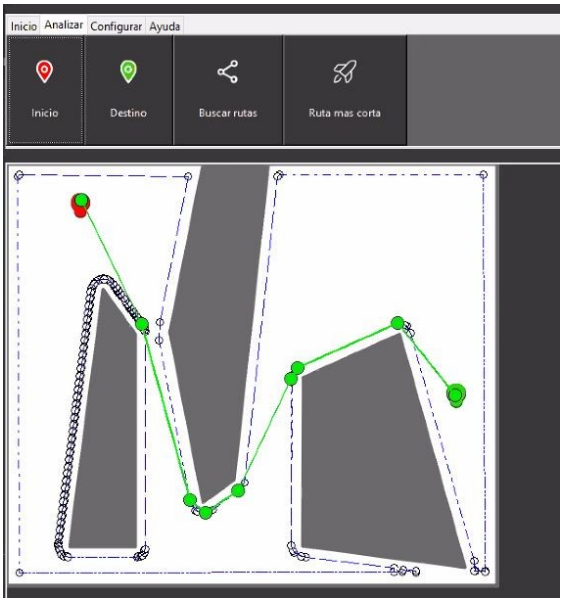

Figure 9: First performance test result

For the second test we used the same environment, and the same initial and final points, but we changed the robot utilized. In this second case, we used a 50% larger robot, for size of 0.3 m by 0.3 m. This new robot cannot pass between the two big obstacles on the left like the first robot did, so a new route is expected for it. In Figure 10 it is observed the new behavior of the solution. Although the software establishes the same routes from the visibility graph, the dimensional constraints applied to the strategy make the two responses different, and in both cases correct.

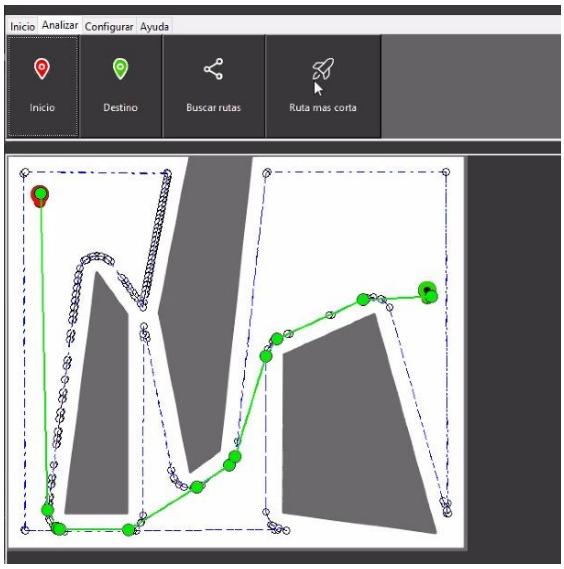

Figure 10: Second performance test result

The third performance test consisted of increasing the size of the robot again while keeping the other parameters constant. In this new case, the robot was dimensioned to 0.4 m by 0.4 m, a size with which we expect to be impossible to reach from the initial point to the final point. The new size of the robot is superior to the spaces between obstacles, and between obstacles and boundaries of the environment, for it is not waiting for a result optimum. Figure 11 shows the result thrown by our simulator. This event of not tracing the dotted lines in the option of searching routes makes understand to the user that it is not possible to obtain a route optimal because the dimensions of the robot exceed the free space, for it, not project any route optimal.

Consequently, unlike the basic scheme of visibility graphs, our simulator considers the real dimensions of the robot and adjusts its search according to the real behavior of the robot in the environment. The most important contribution of our tool is the facility it has to incorporate new robots, considering in the simulation strategy the real dimensions of the machine. By default, the simulator incorporates the parameters of our ARMOS TurtleBot, which allows an easy approach to our robotic platform, but also allows us to define the characteristics of the robots developed by the students, turning it into an important tool for verification and development.

Some aspects were not considered as part of the strategy of navigation, as the fact of defining angles of safe turn for the robots. These parameters were not incorporated in our simulator because they are difficult to establish for the robots. In any case, our research raises this (a criterion that is already being incorporated into the code is the definition of a minimum angle of rotation of the robot, which will be a restriction in the simulator) and other improvements to the software in a later design stage. Even so, the software can calculate the actual total time the robot takes to develop the route calculated by the strategy from the nominal speed of the machine, even considering the time of the turns. This information can be saved for comparison both graphically and in a plain text file from which it is possible to reconstruct all the simulation information (map, robot characteristics, starting point, target point, and coordinates of the whole path). These files facilitate the analysis of different strategies for the same robot, which in the development of navigation strategies corresponds to a key performance comparison.

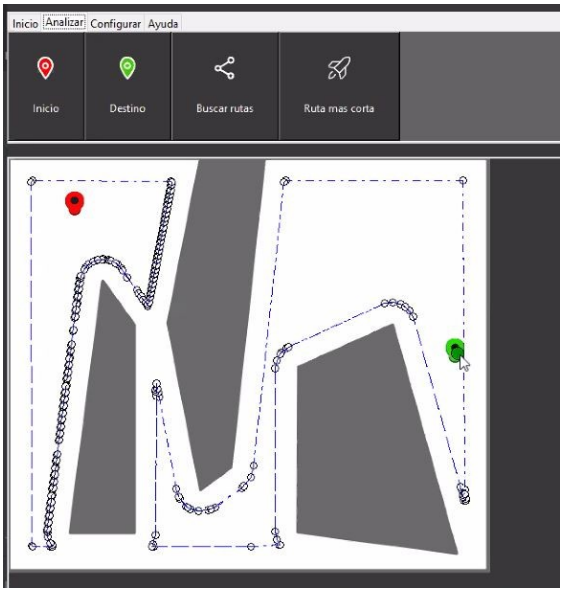

Figure 11: Third performance test result

An important feature of our simulator is that it allows, during image processing, to adjust the sensitivity of the number of nodes to be analyzed for each obstacle in the workspace. Obstacles are abstractions of real objects, and according to the desired sensitivity of the analysis or the desired simplification of the problem, different solutions can be obtained for the same problem. The greater the number of nodes, the greater the number of possibilities with a higher computational cost. Figures 13 and 14 show the effects on the nodes of two different sensitivity values for the same problem.

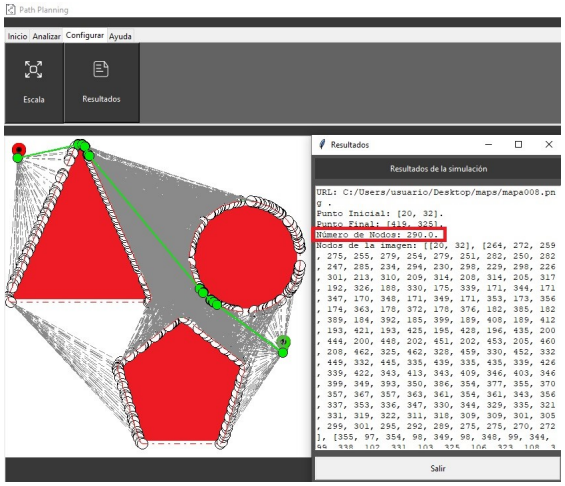

Figure 12: Identification of highly sensitive nodes in the workspace

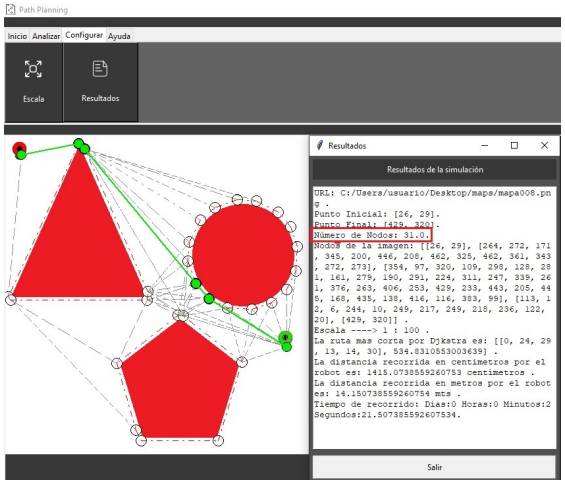

Figure 13: Identification of nodes with low sensitivity in the working space

5. DIFFERENCES FROM PREVIOUS WORK

From previously published work, we can identify the following pros of our tool:

- Our tool is an open-source simulator with a high capacity for user integration and modification. The basic code can be modified to incorporate functionalities, algorithms, optimization schemes, and even add new modules. This is a limitation of commercial tools since they generally do not allow internal code modification.
- It does not require a commercial license, so it can be freely distributed as source code, and the information collected from it can be documented without problems in scientific articles without infringing copyright and facilitating the duplication of the experiments by the readers.
- It is intended for any user interested in the subject. It requires no special training or specialized knowledge to use, and the initial learning curve is very flat.
- The configuration of the working environment can be easily done from an image. Similarly, the configuration of the robot is very simple, and only requires some physical and functional parameters of the machine.

In the same way, compared to other equivalent tools, there are some disadvantages of our simulator:

- It is in the development stage, so its results cannot yet be considered for applications in the production stage, only as a training tool and primary evaluation of schemes.
- Due to its simplicity of use and training approach, the tool does not consider precise odometry parameters, which can lead to errors in the final performance of the robots. The relevance of this limitation is low since the purpose of the tool is training and research, not production.
- A comprehensive evaluation of the simulator that would allow proposing adjustments and improvements has not yet been developed. It is expected that the use of the tool in the medium term will provide functional metrics and approaches for future development.
- Limited distribution. Although the tool is open source and freely distributed, it is currently only available to our research group.

6. CONCLUSION

This article documents the development of a custom-made simulator for the training of young researchers in robotic navigation strategies. Initial tests with members of the research group point to the fact that the software supports the processes of cognitive appropriation of basic strategies.

In its first stage, the software implements the visibility graph algorithm for a robot in conjunction with Dijkstra's algorithm as support for the definition of the optimal route. The algorithm implements the traditional strategy but allows the user to define some real-world constraints such as the actual size of the robot, and the design of the navigation environment and its obstacles. In their development, it was considered so many technical

aspects related to the characteristics of the algorithms, and the robots of the group of investigation, as pedagogical related to the formation of intrinsic and extrinsic motivators in the students to develop a support tool in the specialized self-training. The article presents details of its development both in terms of structure and graphic interface level. It is also shown how through the graphic interface the user can specify the simulation environment, the characteristics of the robot, and the navigation requirements. It is also shown how this information is digitized and processed by the simulator to define the possible routes and finally establish the optimal route.

From the performance tests performed by our students, we can show the ability of the simulator to both replicate the behavior of the algorithm and to develop active learning in students. The students were able, even those with low knowledge in robotics, to correctly handle the simulator, and understand the behavior of the planning strategy. In the final stage of the project, improvements to the tool are proposed that include the incorporation of a greater number of navigation strategies and search algorithms.

6. LIMITATIONS AND FUTURE RESEARCH

The previous results of this research allow us to propose future directions of the research, both in the improvement of the simulator and in the strengthening of the knowledge management system in which its use is framed. Below we detail the most important branches.

There are functions in the tool that can and should be complemented with new algorithms. In particular, the scheme used for the weighting and selection of the optimal route between the feasible paths should be complemented with a greater number of strategies. Classic informed and uninformed search algorithms should be implemented, such as A* style heuristic searches, graph searches as first in width, or first in-depth, gradient searches as simulated annealing, as well as randomized search algorithms such as genetic algorithms, or ant colony.

During the design of the simulator, an image processing training was used, in which a total of 100 images were randomly taken, and its performance/efficiency was 96%. It was identified that the images containing figures, and objects (obstacles), when their color composition was clear (beige, pink), the OpenCV module, cannot identify the obstacle, causing trajectories on itself. The recommendation to the user is to adjust the color of the obstacle in such a way that it generates a high contrast with the workspace so that the OpenCV module processes the recognition of the image properly.

Greater restrictions must be incorporated in the movement of the robot according to the real capacities of the robots. The more important of these restrictions are related to the safe movement of the robot, for a real robot it is impossible to make any type of turn, since some of these or are physically impossible, or they put at risk the stability of the machine.

It should be included in the tool other traditional strategies of movement planning. These new strategies should allow a form of work similar in terms of the configuration of the environment of navigation and the robots to facilitate the comparisons of performance. As well as the design of this simulator, the new modules must allow the continuous integration of features and functions.

It is necessary to use the initial results of the use of the tool related to the learning habits of the users, as well as their working method to re-design the interface and the usability of the software. The simulator can and should be improved to improve the learning process of the student population to which the tool is directed. This improvement must be continuous since study habits tend to be dynamic.

7. ACKNOWLEDGMENT

This study was granted by the Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Colombia. The trials and evaluations were developed by researchers from the ARMOS research group.

REFRENCES

[1] P. Ibañez, C. Villalonga, and L. Nuere. Exploring Student Activity with Learning Analytics in the Digital Environments of the Nebrija University. Technology, Knowledge and Learning, 25(4):769–787, 2020. doi: 10.1007/s10758-019-09419-4.

[2] N. Nordin, N. Majid, and N. Zainal. Mobile augmented reality using 3d ruler in a robotic educational module to promote stem learning. Bulletin of Electrical Engineering and Informatics, 9(6):2499–2506, 2020. doi: 10.11591/eei.v9i6.2235.

[3] S. Djabbarova, M. Tadjieva, R. Mardonova, and G. Turaeva. Problem based learning and its efficiency in teaching process. European Journal of Molecular and Clinical Medicine, 7(2):291–296, 2020.

[4] H. Tinmaz and J. Lee. An analysis of users' preferences on learning management systems: A case on German versus Spanish students. Smart Learning Environments, 7(1), 2020. doi: 10.1186/s40561-020-00141-8.

[5] K. Chrysafiadi, C. Troussas, and M. Virvou. Combination of fuzzy and cognitive theories for adaptive e-assessment. Expert Systems with Applications, 161, 2020. doi: 10.1016/j.eswa.2020.113614.

[6] K. Zuza, M. De Cock, P. Van Kampen, T. Kelly, and J. Guisasola. Guiding students towards an understanding of the electromotive force concept in electromagnetic phenomena through a teaching-learning sequence. Physical Review Physics Education Research, 16(2), 2020. doi: 10.1103/PhysRevPhysEducRes.16.020110.

[7] D. Jiménez-Hernández, V. González-Calatayud, A. Torres-Soto, A. Mayoral, and J. Morales. Digital competence of future secondary school teachers: Differences according to gender, age, and branch of knowledge. Sustainability (Switzerland), 12(22):1–16, 2020. doi: 10.3390/su12229473.

[8] A. Mishina, G. Batyrshina, Z. Yavgildina, and I. Avramkova. Personalization of Art Students' Training in the Context of the Transition to the Digital Economy. International Journal of Criminology and Sociology, 9:931–935, 2020. doi: 10.6000/1929-4409.2020.09.97.

[9] S. Scott, T. Dalsgaard, J. Jepsen, C. von Buchwald, and S. Andersen. Design and validation of a cross-specialty simulation-based training course in basic robotic surgical skills. International Journal of Medical Robotics and Computer Assisted Surgery, 16(5):1–10, 2020. doi: 10.1002/rcs.2138.

[10] J. Blevins, K. Felix, D. Ling, P. Sculco, M. McCarthy, C.A. Demetracopoulos, A.S. Ranawat, and D.T. Fufa. Surgical Games: A Simulation-Based Structured Assessment of Orthopedic Surgery Resident Technical Skill. Journal of Surgical Education, 77(6):1605–1614, 2020. doi: 10.1016/j.jsurg.2020.05.009.

[11] E. Jokinen, T. Mikkola, and P. Härkki. Simulator training and residents' first laparoscopic hysterectomy: A randomized controlled trial. Surgical Endoscopy, 34(11):4874–4882, 2020. doi: 10.1007/s00464-019-07270-3.

[12] Q. Deng, J. Wang, K. Hillebrand, C. Benjamin, and D. Soffker. Prediction Performance of Lane Changing Behaviors: A Study of Combining Environmental and Eye-Tracking Data in a Driving Simulator. IEEE Transactions on Intelligent Transportation Systems, 21(8):3561–3570, 2020. doi: 10.1109/TITS.2019.2937287.

[13] C. De Oliveira, M. De Aguiar, A. De Castro, P. Guazzelli, W. De Andrade Pereira, and J. De Almeida Monteiro. High-Accuracy Dynamic Load Emulation Method for Electrical Drives. IEEE Transactions on Industrial Electronics, 67(9):7239–7249, 2020. doi: 10.1109/TIE.2019.2942566.

[14] A. Bernard, P. Chemaly, F. Dion, S. Laribi, F. Remerand, D. Angoulvant, and F. Ivanes. Evaluation of the efficacy of a self-training programme in focus cardiac ultrasound with simulator. Archives of Cardiovascular Diseases, 112(10):576–584, 2019. doi: 10.1016/j.acvd.2019.06.001.

[15] A. Akalin and S. Sahin. The impact of high-fidelity simulation on knowledge, critical thinking, and clinical decision-making for the management of pre-eclampsia. International Journal of Gynecology and Obstetrics, 150(3):354–360, 2020. doi: 10.1002/ijgo.13243.

[16] A. Wahl. Expanding the concept of simulator fidelity: The use of technology and collaborative activities in training maritime officers. Cognition, Technology and Work, 22(1):209–222, 2020. doi: 10.1007/s10111-019-00549-4.

[17] S. Guraya, S. Guraya, and M. Al-Qahtani. Developing a framework of simulation-based medical education curriculum for effective learning. European Journal of Anatomy, 24(4):323–331, 2020.

[18] Á. Arnaiz-González, J. Díez-Pastor, I. Ramos-Pérez, and C. García-Osorio. Seshat - a web-based educational resource for teaching the most common algorithms of lexical analysis. Computer Applications in Engineering Education, 26(6):2255–2265, 2018. doi: 10.1002/cae.22036.

[19] A. Megías, A. Cortes, A. Maldonado, and A. Cándido. Using negative emotional feedback to modify risky behavior of young moped riders. Traffic Injury Prevention, 18(4):351–356, 2017. doi: 10.1080/15389588.2016.1205189.

[20] H. Chen, H. Park, and C. Breazeal. Teaching and learning with children: Impact of reciprocal

peer learning with a social robot on children's learning and emotive engagement. Computers and Education, 150, 2020. doi: 10.1016/j.compedu.2020.103836.

[21] B. Adewale, F. Jegede, P. Aderonmu, O. Fulani, E. Erebor, and O. Joshua. The Relationship between teacher's/students' characteristics and their learning styles in a visual communication class. International Journal of Civil Engineering and Technology, 9(9):782–791, 2018.

[22] N. Harihara Sudhan, S. Shriram, M. Anand, and R. Sujeetha. Game environment exploration using curiosity-driven learning. International Journal of Recent Technology and Engineering, 7(6):715–718, 2019.

[23] A. Townsend-Nicholson. Educating and engaging new communities of practice with high performance computing through the integration of teaching and research: Using technology to transform learning. Interface Focus, 10(6), 2020. doi: 10.1098/rsfs.2020.0003rsfs20200003.

[24] C. Schuster, F. Stebner, D. Leutner, and J. Wirth. Transfer of metacognitive skills in self-regulated learning: An experimental training study. Metacognition and Learning, 15(3):455–477, 2020. doi: 10.1007/s11409-020-09237-5.

[25] M. Macías García, A. Izaguirre Alegría. and A. Cortés Pérez. Cyber-Physical Labs to enhance engineering training and education. International Journal on Interactive Design and Manufacturing, 14(4):1253–1269, 2020. doi: 10.1007/s12008-020-00704-6.

[26] I. Gaudiello and E. Zibetti. Learning Robotics, with Robotics, by Robotics: Educational Robotics. Wiley, 2016.

[27] J. Cañas, E. Perdices, L. García-Pérez, and J. Fernández-Conde. A ROS-based open tool for intelligent robotics education. Applied Sciences (Switzerland), 10(21):1–20, 2020. doi: 10.3390/app10217419.

[28] A. Staranowicz and G. Mariottini. A survey and comparison of commercial and open-source robotic simulator software. In Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments, PETRA '11, pages 1–8, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 978-1-4503-0772-7. doi: 10.1145/2141622.2141689.

[29] V. Carbonell and R. Estepa. Simulation of a quadrupedal bioinspired modular robot using webots. International Review on Modelling and Simulations, 12 (2):94–102, 2019. doi: 10.15866/iremos.v12i2.16349.

[30] Ó. Avilés, O. Rubiano, M. Mauledoux, A. Valencia, and R. Moreno. Simulation of a mobile manipulator on webots. International Journal of Online Engineering, 14(2):90–102, 2018. doi: 10.3991/ijoe.v14i02.7789.

[31] E. Neha, M. Suhaib, S. Asthana, and S. Mukherjee. Grasp analysis of a four-fingered robotic hand based on matlab simmechanics. Journal of Computational and Applied Research in Mechanical Engineering, 9 (2):169–182, 2020. doi: 10.22061/jcarme.2019.3427. 1390.

[32] M. Sarkar, A. Homaifar, B. Erol, M. Behniapoor, and E. Tunstel. PIE: A Tool for Data-Driven Autonomous UAV Flight Testing. Journal of Intelligent and Robotic Systems: Theory and Applications, 98(2):421–438, 2020. doi: 10.1007/s10846-019-01078-y.

[33] A. Brunete, E. Gambao, J. Koskinen, T. Heikkilä, K. Kaldestad, I. Tyapin, G. Hovland, D. Surdilovic, M. Hernando, A. Bottero, and S. Anton. Hard material small-batch industrial machining robot. Robotics and Computer-Integrated Manufacturing, 54:185–199, 2018. ISSN 0736-5845. doi: 10.1016/j.rcim.2017.11.004.

[34] W. Dong, F. Palmquist, and S. Lidholm. A Simple and Effective Emulation Tool Interface Development for Tricept Application. page 5, 2002.

[35] A. Hentout, A. Maoudj, N. Kaid-Youcef, D. Hebib, and B. Bouzouia. Distributed Multi-agent Bidding-Based Approach for the Collaborative Mapping of Unknown Indoor Environments by a Homogeneous Mobile Robot Team. Journal of Intelligent Systems, 29(1):84–99, 2020. doi: 10.1515/jisys-2017-0255.

[36] J. Song and S. Gupta. E ∗ : An Online Coverage Path Planning Algorithm. IEEE Transactions on Robotics, 34(2):526–533, 2018. doi: 10.1109/TRO.2017.2780259.

[37] L. Peng, F. Guan, L. Perneel, H. Fayyad-Kazan, and M. Timmerman. Decentralized Multi-Robot Formation Control with Communication Delay and Asynchronous Clock. Journal of Intelligent and Robotic Systems: Theory and Applications, 89(3-4):465–484, 2018. doi: 10.1007/s10846-017-0557-y.

[38] C. Liang, V. Kamat, and C. Menassa. Teaching robots to perform quasi-repetitive construction tasks through human demonstration. Automation in Construction, 120, 2020. doi: 10.1016/j.autcon.2020.103370.

[39] S. Abdulredah and D. Kadhim. Developing a real time navigation for the mobile robots at unknown environments. Indonesian Journal of Electrical Engineering and Computer Science, 20(1):500–509, 2020. doi: 10.11591/ijeecs.v20.i1.pp500-509.

[40] L. Hugues and N. Bredeche. Simbad: An Autonomous Robot Simulation Package for Education and Research. In S. Nolfi, G. Baldassarre, R. Calabretta, J. Hallam, D. Marocco, J. Meyer, O. Miglino, and D. Parisi, editors, From Animals to Animats 9, Lecture Notes in Computer Science, pages 831–842, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-38615-5. doi: 10.1007/11840541_68.

[41] B. Rachid and M. Benmohamed. Global Localization and Concurrent Mapping for Mobile Robot on the robotic simulator "SIMBAD". AIP Conference Proceedings, 1107(1):368–372, 2009. ISSN 0094-243X. doi: 10.1063/1.3106505.

[42] M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: The Carnegie Mellon Navigation (CARMEN) Toolkit. In Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453), volume 3, pages 2436–2441 vol.3, 2003. doi: 10.1109/IROS.2003.1249235.

[43] J. Pineau and A. Atrash. SmartWheeler: A Robotic Wheelchair Test-Bed for Investigating New Models of Human-Robot Interaction. In AAAI Spring Symposium, pages 1–6, 2007.

[44] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. USARSim: A robot simulator for research and education. In IEEE International Conference on Robotics and Automation, page 1405, 2007. doi: 10.1109/ROBOT.2007.363180.

[45] S. Balakirsky and Z. Kootbally. USARSim/ROS: A Combined Framework for Robotic Control and Simulation. In ASME/ISCIE 2012 International Symposium on Flexible Automation, pages 101–108. American Society of Mechanical Engineers Digital Collection, 2013. doi: 10.1115/ISFA2012-7179.

[46] D. Michal and L. Etzkorn. A comparison of Player/Stage/Gazebo and Microsoft Robotics Developer Studio. In Proceedings of the 49th Annual Southeast Regional Conference, ACM-SE '11, pages 60–66, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 978-1-4503-0686-7. doi: 10.1145/2016039.2016062.

[47] Antonio Matta-Gómez, J. Del Cerro, and A. Barrientos. Multi-robot data mapping simulation by using microsoft robotics developer studio. Simulation Modelling Practice and Theory, 49:305–319, December 2014. ISSN 1569-190X. doi: 10.1016/j.simpat.2014.10.003.

[48] F. Serrano, B. Diego, V. Rodilla, J. Rodriguez-Aragon, R. Santos, and C. Fernandez-Carames. The complete integration of MissionLab and CARMEN. International Journal of Advanced Robotic Systems, 14(3): 1729881417703565, 2017. ISSN 1729-8814. doi: 10.1177/1729881417703565.

[49] A. Chella, R. Sorbello, S. Siniscalchi, and S. Vitabile. MIP: A New Hybrid Multi-Agent Architecture for the Coordination of a Robot Colony Activities. In 15th Eureopean Conference on Artificial Intelligence ECAI'2002, pages 1–5, 2002.

[50] F. Martínez, F. Martínez, and H. Montiel. Hybrid Free-Obstacle Path Planning Algorithm using Image Processing and Geometric Techniques. ARPN Journal of Engineering and Applied Sciences, 14(18):3135-3139, 2019.

[51] A. Montes. Planificación de Caminos basada en Modelo Combinando Algoritmos de Búsqueda en Grafo, derivados de RRT y RRT*. Escuela Técnica Superior de Ingeniería. Universidad de Sevilla, 2017.

[52] E. Jacinto, F. Martínez, and F. Martínez. A comparative study of geometric path planning methods for a mobile robot: Potential field and Voronoi diagrams. A Comparative Study of Geometric Path Planning Methods for a Mobile Robot: Potential Field and Voronoi Diagrams, pages 1-6, 2013.

[53] W. Huijuan, Y. Yuan, and Y. Quanbo. Application of Dijkstra algorithm in robot path-planning. 2011 Second International Conference on Mechanic Automation and Control Engineering, 2011. doi: 10.1109/MACE.2011.5987118.

[54] C. Penagos, L. Pacheco, and F. Martínez. ARMOS TurtleBot 1 Robotic Platform: Description, Kinematics and Odometric Navigation. International Journal of Engineering and Technology (IJET), 10(5): 1402-1409, 2018, doi: 10.21817/ijet/2018/v10i5/181005043.