Towards AGV Optimization using ROS and Stage Simulator

Bruno Carneiro da Silva
Universidade da Beira Interior
Instituto de Telecomunicações
Covilhã, Portugal
bruno.carneiro.silva@ubi.pt

Luís A. Alexandre
Universidade da Beira Interior
Instituto de Telecomunicações
Covilhã, Portugal
luis.alexandre@ubi.pt

***Abstract*—Autonomous Guided Vehicles (AGV) are currently being used in industry to move materials efficiently. Simulators may be used to help calculate the right number of AGVs needed for a particular task and also which types better suit the necessity of a company. This paper analyzes the characteristics of many of the most used simulators and focus on evaluating an environment using Stage and Robot Operating System (ROS), to find experimentally if one AGV may complete a specific task taking into account eventual path blockages by random events.**

***Index Terms*—Simulation, Autonomous Guided Vehicles.**

I. Introduction

There are numerous applications in which robots are used in the real world and several researches throughout distinct fields. The benefits from areas such as medicine [1], bionic movements [2], military applications [3], industrial operations [4], [5] and domestic use [6]. These robots may be used to meet the requirements of flexible production and improve the production efficiency [5].

Specifically in the industry field, Autonomous Guided Vehicles (AGVs) provide self-guided navigation system to find their routes to get to target workstations and move products and materials with no pre-defined routes [7]. They are currently being used in several areas. Many operational activities may benefit from AGVs such as the ones related to moving raw materials from and into warehouses or container terminals and other generic transportation activities [5], which are important tasks inside a factory. The interaction with delicate objects and human employees must be considered in this scenario. For this reason, there is a necessity of thoroughly evaluate these vehicles before they are deployed in the real world. However, using a real environment to evaluate robots may be a complicated assignment and also may present several risks. Therefore, a viable option to perform these evaluations is to use simulated environments.

The robot simulators are essential tools because they may complement traditional experimental approaches and enable tests in controlled and secure environments. Hence, choosing a suitable simulator may reduce the costs, possible damages to expensive equipment are avoided [7]. There are several types of simulators, some are used only for specific types of robots, other may be generic and open source or proprietary systems [5], [8], [9], In this work, several open source simulators were analyzed, based on the objectives they were built for, as well as their technical features. The Stage simulator working with Robot Operating System (ROS) was chosen to present a simulation of an AGV performing a task in an industrial environment. The main idea is to analyze the time in which the AGV could complete one task cycle, when obstacles appear.

This article is divided as follows: section II contains an overview of several simulators that are being used for different purposes, divided by fields of activity. Section III provides the tests and results of a preliminary experiment used to determine if a single AGV is sufficient to complete a task, when random obstacles may block the AGVs' path. The final section presents the conclusions and future work.

II. An Overview of Open Source Robotic Simulators

In this section, open source robot simulators are briefly described focusing on their main features, which vary from academic purposes, industrial usability and service operations and other goals. Also, several technical features of the simulators are disclosed and a comparison is presented in Table I. When choosing which simulator to use, the technical features such as supported programming languages, default resources provided by the simulator and the amount of robots inside its default library, must be considered. The physics engine and mesh manipulation should be considered when the main focus is the reality customization level of the constructed environment and robots. The output format refers to the type of data which will be analyzed after the simulation is completed.

A. Open Source Simulators focused on Academic Purposes

In principle, all the open source simulators may be used for academic purposes since their code is available online, but some of them evolved, becoming more robust to the point that companies started to use them to simulate usability at an industrial-scale. Others remained with its focus towards the academic field such as Morse, SimSpark, V-REP, Stage, Simbad and STDR, which are discussed below.

Modular Open Robots Simulation Engine (MORSE) [10] is used to develop and implement service robots and focuses on different environments. It allows the adjustment of the

Identify applicable funding agency here. If none, delete this.

realism level and the integration into several open source middlewares such as ROS and YARP. The SimSpark simulator targets multi-agents and supports the development of physical simulation for research related to artificial intelligence and robotics. These agents communicate between themselves with the simulation server through UDP and TCP protocols. Therefore, they allow implementations using any programming language that supports these sockets. SimSpark is the official simulator used in the Robocup Simulation League [11]. As for V-REP [12], the software offers support for a variety of programming languages and for the development of algorithms used in simulated environments where automation is applied. The software has a professional version which is used by engineers for monitor and security purposes. However, it also has an educational version, which is limited but it is free and targets researches goals.

Other simulators such as Stage, Simbad and Simple Two Dimensional Robot (STDR) focus on delivering a small variety of technical features compared to other simulators studied throughout this work. Their goal is to be lightweight and provide more automation through coding rather than realistic world accuracy and multiple detailed 3D features. For example, Simbad [13] targets Evolutionary Robots, offers support for artificial intelligence and machine learning algorithms and was built to create world features with some reality but it is intentionally maintained with simple technical characteristics such as own physics engine and small amount of default resources. As for STDR [14], it provides the option for experiments to be executed remotely and also, without a graphic environment. This simulator was created to be totally compatible with ROS, leaving the researchers free to use a traditional robot framework and take advantage of ROS features. This simulator provides distinct access to the STDR client graphical interface and server, allowing them to be executed in different computers. The Stage Project [15], provides open source tools that simplify controller development, particularly for multiple robot, distributed-robot, and sensor network systems. This project aims to be efficient and configurable rather than highly accurate.

B. Open Source Simulators used in Industry

Simulators are commonly used to evaluate robots in environments that may be considered sensitive due to product handling, interaction with people and factory equipment. As aforementioned, there are several types of simulators, a few open source ones are used in the industry field such as Webots and Gazebo. Webots [16] is used specially in industry and service segments, where any robot may be modeled, programmed and simulated using different programming languages. It has external libraries compatible with OpenCV and other platforms. On the other hand, Gazebo [9] is a multi-robot simulator which supports a large number of sensors and objects. The software is also compatible with ROS, such as many other platforms. It also provides realistic renderization of environments and allows the generation of different sensor data, such as lasers, contact sensors, lidars and 2D/3D cameras.

Another software that should be mentioned is Blender [17], which is a powerful tool created to design and manipulate service robots in complex environments. The platform offers support for 3D contents and may be used to simulate virtual worlds and robot behaviors. Blender also allows renderization, modelling, animation, simulation, video editing and scripting. Some simulators, such as MORSE, use physics engines which are contained in Blender.

C. Open Source Simulators for other Goals

Some simulators focus on delivering efficient reproduction of any type of swarm of robots, as the ARGoS platform [18], which uses multi-physics concept. The software is designed to help create applications for different types of contexts, such as hostile environments, disaster recovery and construction and large-scale medicine. ArGoS focuses on three components: high precision, flexibility and efficiency. This tool also allows the addition of new plugins to provide the users with better customization according to their necessities. Autonomous Navigation Virtual Environment Laboratory (ANVEL) [19] is a system that provides modelling and interactive simulation capabilities. It was built to assist in the design, testing and evaluation specifically for intelligent ground vehicles. This software allows the user to rapid and easy construct virtual models, insert sensors on board these models, use control algorithms, program plugins directly into the system and perform tests in a large number of virtual environments with scalable fidelity.

D. Technical Features of the Simulators

As aforementioned, each simulator is adapted to distinct purposes.Below, several technical features are presented, as well as a comparison table between the simulators based on these features.

Supported Languages: Each simulator allows the use of several different programming languages with C++ and Python being the most common ones.

Physics Engine: It is a technology which allows the simulation of physic systems as close to reality as possible [17]. Velocity, mass and air systems are some of the real conditions of which a physics engine simulates. This type of engine basically works with real-time and high precision physics. This is regarded as a crucial feature in robot simulation.

Mesh Manipulation: Mesh is a collection of vertices, borders and surfaces which define the shape of an object [20]. Specially used in computer graphics, 3D environment and modelling, the meshes may assume different formats. In the robot simulators, its function is the same and the simulators allow different use and manipulation of the mesh.

Output Formats: This feature provides different types of outputs such as video frames, text file, pictures and specific formats based on which simulator is being used and even its programming language options. The suitable output format is important to know what type of file will be further analyzed and used once the simulation finishes.

Robot Library: There are many options of robots on the market and the simulators allow the construction of customized robots or even have robot libraries with several pre-defined robot and world models. Also, some of these platforms allow that other models be imported.

Default Resources: The simulators also provide default resources besides the robot models such as random objects and sensors. Walls, doors, windows, trees and tracks are examples of these objects. Default resources may also depend on the framework that the simulator is able to use for integration, i. e, ROS.

III. Experimental use of a Simulator

In this section we illustrate the possible use of a simulator for AGV placement. We will use the Stage simulator because of its simplicity. Therefore, we focus on analyzing the results of the accomplishment of the task rather than on great realism.

The main idea of the use case, is to determine if only one AGV is sufficient to complete the exemplary task based on the probability of the obstacle occurrence, i.e., human workers walking through the robot path. For that, a simple simulation model was created inspired on AGV networks inside factories

A. Model Parameters

For the simulation we will need to define the following parameters: the path that the AGV would go through, the AGV velocity (in our case is a constant) and the total time in which the AGV must complete a cycle.

In practice, there will be some probability of the AGV finding obstacles in its path and hence, it will have to wait since it is following a magnetic band placed on the floor to define the path. These obstacles will be modelled in the simulation using a probability of obstacle occurrence (POO) p, and a fixed duration of the presence of the obstacle. The obstacles are commonly people that cross the path of the AGV and once they realize the AGV is blocked because of their presence, they move to allow the AGV to continue its work. So we modelled the duration of the obstacle presence with the fixed value of 10 seconds.

B. Environment Model

A snapshot of the simulation model can be seen in Fig. 1. It shows factory walls, one AGV, and the magnetic band in yellow. The total length of the path is 50 meters and the pre-established AGV velocity was 1 m/s. The ideal total time for the AGV to complete the cycle is 56 seconds (this is the cycle time).

C. Tests and Results

We made a set of experiments where we varied the value of p from 0.1 to 0.9. For each value we run 20 repetitions and store the total time the AGV needed to complete the path. The results are presented in Fig. 2. We see that, if the POO is more than around $p = 0.64$, the AGV exceeds the cycle time of 56 seconds and the further use of another AGV would be required in order to complete the task inside the defined cycle duration.

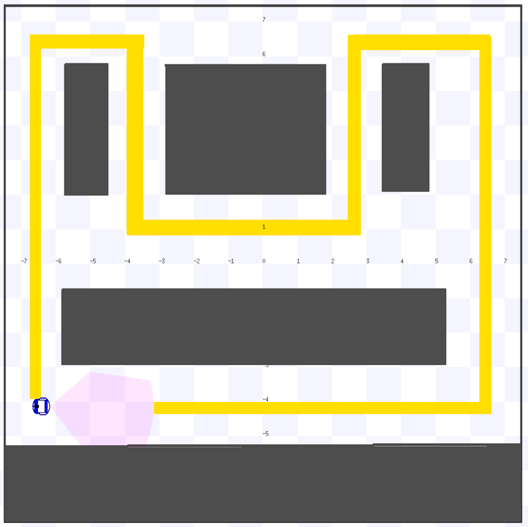

Fig. 1. A snapshot of the simulation model. The magnetic band is represented in yellow, walls are in dark grey, the AGV model is displayed in blue and its LiDAR scanning range is depicted in pink.

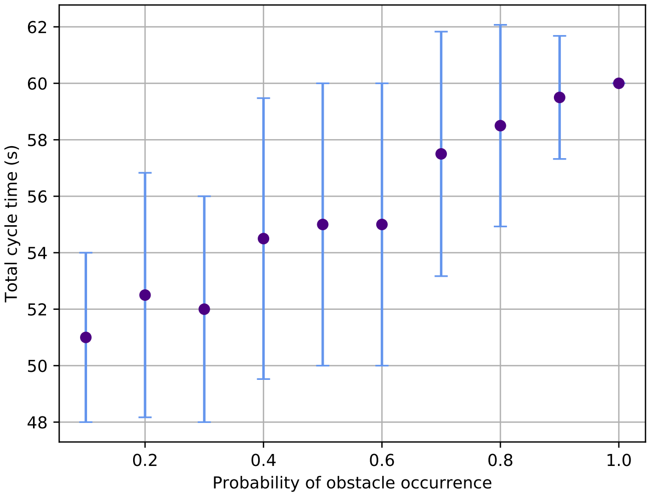

Fig. 2. Average and standard deviation, over the 20 repetitions, of the amount of time a single AGV required to perform the cycle, when objects were gradually appearing in its path, with probability of obstacle occurrence given in the X-axis.

IV. Conclusions

This paper presented a brief overview of several common robotic simulators and analyzed the possibility of a simulator helping in the decision regarding AGV placement based on the time per cycle when obstacles appear randomly along the path. The results show that when more interference emerged and the AGV stopped, the average time for one AGV to complete the task would increase, as expected, eventually passing the 56 seconds defined time per cycle threshold. Therefore, we concluded that, for the evaluated scenario, if the probability of obstacle occurrence is above 0.64, one AGV would not

TABLE I

COMPARISON BETWEEN OPEN SOURCE SIMULATION TOOLS REGARDING THEIR TECHNICAL FEATURES AND CUSTOMIZATION LEVEL

Simulator	Supported Languages	Physics Engine	Mesh Handling	Output Formats	Robot Library	Default Resources
Webots	C, Java, Python, C++, Matlab	ODE	Fixed Mesh	Video, 3D images	Small Diversity	Avg Amount
Simbad	Java	Own Customized Physics	Fixed Mesh	N/A	N/A	Small Amount
V-REP	C, Python, Matlab, Octave, Lua, Java	Bullet, Vortex, ODE, Newton	Allows Handling	Video, text & other data	Great Diversity	Great Amount
Stage	C, C++, Python	Own Customized Physics	Fixed Mesh	Specific Stage File & text	Small Diversity	Small Amount
MORSE	Python	Based in Blender	N/A	N/A	Moderate Diversity	Avg Amount
ArGos	Lua, C++	Own Customized Physics	Fixed Mesh	Video Frames and text	Small Variety	Small Amount
Anvel	C, Java, Python, C++, Matlab	Based in ODE	Multiple Meshes	N/A	Small Variety	Small Amount
STDR	C++	N/A	N/A	N/A	Small Variety	Small Amount
Simspark	Ruby	ODE	Multiple Meshes	N/A	Moderate Diversity	Avg Amount
Gazebo	Python	ODE	Fixed but allows import	Log file, text & video frames	Moderate Diversity, Allows addition	Avg Amount
Blender	Python	Various Physics Engine	Complete Handling	Great Variety	N/A	Great Quantity

be enough to perform the task in time. For future work, it is important to perform tests using the design of multiple alternative paths and cycles as well as variations in the amount of time which the AGV may stop in front of an obstacle. AGV velocity and time to perform a cycle vary accordingly to the type of vehicle used. Therefore, testing with an increased set of parameters may also be significant.

ACKNOWLEDGMENTS

This work was partially supported by Instituto de Telecomunicações under grant UID/EEA/50008/2019 and by project 026653 (POCI-01-0247-FEDER-026653) "INDTECH 4.0 New technologies for smart manufacturing", co-financed by the Portugal 2020 Program (PT 2020), Compete 2020 Program and the European Union through the European Regional Development Fund (ERDF).

REFERENCES

[1] P. Hamet and J. Tremblay, "Artificial Intelligence in Medicine," Metabolism, vol. 69, pp. S36–S40, April 2017.

[2] D. Liu, M. Sun and D. Qian, "Structural Design and Gait Simulation of Bionic Quadruped Robot," International Conference on Advanced Mechatronic Systems (ICAMechS), August 2018.

[3] P. Sapaty, "Military robotics: Latest Trends and Spatial Grasp Solutions," International Journal of Advanced Research in Artificial Intelligence, vol. 04, April 2015.

[4] H. Fazlollahtabar and S. Hassanli, "Hybrid cost and time path planning for multiple autonomous guided vehicles," Applied Intelligence, vol. 48, pp. 482-498, February 2018.

[5] Y. Pan, X. Ma, C. Mu, H. An and J. Chen. "Design of Industrial Robot Sorting System with Visual Guidance Based on Webots," International Conference on Computer and Communication Systems (ICCCS), vol. 3, pp. 516–521, April 2018.

[6] M. M. A. de Graaf, S. B. Allouch and J. A. G. M. van Dijk, "Why would i use this in my home? a model of domestic social robot acceptance," HumanComputer Interaction, vol. 34, pp. 115–173, 2019.

[7] Fengjia Yao et. al., "Optimizing the Scheduling of Autonomous Guided Vehicle in a Manufacturing Process," International Conference on Industrial Informatics (INDIN), pp. 264–269, July 2018.

[8] G. A. Camacho M., C. H. Rodriguez and D. Alvarez-Martinez, "Modelling the kinematic properties of an industrial manipulator in packing applications," 2018 IEEE 14th International Conference on Control and Automation (ICCA), pp. 1028–1033, June 2018.

[9] L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield. "Feature and Performance Comparison of the V-REP, Gazebo and ARGoS Robot Simulators," Towards Autonomous Robotic Systems, vol. 10965, pp. 357–368, July 2018.

[10] G. Echeverria, N. Lassabe, A. Degroote, S. Lemaignan. "Modular open robots simulation engine: Morse," IEEE International Conference on Robotics and Automation, May 2011.

[11] O. Obst and M. Rollmann, "Spark A Generic Simulator for Physical Multi-agent Simulations," MATES 2004: Multiagent System Technologies. Lecture Notes in Computer Science, vol. 3187, pp. 243–257, September 2004.

[12] E. Rohmer, S. P. N. Singh, M. Freese, "V-REP: A versatile and scalable robot simulation framework," International Conference on Intelligent Robots and Systems, pp. 1321–1326, November 2013.

[13] L. Hugues and N. Bredeche, "Simbad: An Autonomous Robot Simulation Package for Education and Research," SAB 2006: From Animals to Animats, vol. 9. Lecture Notes in Computer Science, vol 4095, pp. 831–842, September, 2006.

[14] "Simple two dimensional robot simulator." Online at http://stdr-simulator-ros-pkg.github.io/, Last accessed: April 2019.

[15] B. P. Gerkey, R. T. Vaughan and A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," In Proceedings of the International Conference on Advanced Robotics, pp. 317–323, June 2003.

[16] Olivier Michel, "Cyberbotics ltd. Webots$^{\text{TM}}$: Professional mobile robot simulation," International Journal of Advanced Robotic Systems, vol. 1 pp. 39-42, March 20014.

[17] "Blender Project: Free and Open 3D Creation Software." Online at http://www.blender.org, Last accessed: April 2019.

[18] Carlo Pinciroli et. al. "Argos: a modular, parallel, multi-engine simulator for multi-robot systems," Swarm Intelligence, vol. 6, pp 271–295, September 2012.

[19] Phillip J. Durst, "A Real-Time, Interactive Simulation Environment for Unmanned Ground Vehicles: The Autonomous Navigation Virtual Environment Laboratory (ANVEL)," International Conference on Information and Computing Science, pp. 7–10, July 2012.

[20] H. Kato, Y. Ushiku and T. Harada, "Neural 3D Mesh Renderer," Computing Research Repository (CoRR), pp. 3907–3916, August 2018.