2022 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
Santa Maria da Feira, Portugal – April 29-30, 2022

2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) | 978-1-6654-8217-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICARSC55462.2022.9784794

RobotAtFactory 4.0: a ROS framework for the SimTwo simulator

João Braun*‡, Alexandre Oliveira Júnior*, Guido S. Berger*§, José Lima*†, Ana I. Pereira*, Paulo Costa†¶
* Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal, Email: {jbneto, alexandrejunior, guido.berger, jllima, apereira}@ipb.pt
† INESC Technology and Science, Porto, Portugal
‡ Mountains of Research Collaborative Laboratory, Portugal
§ Universidade de Trás-os-Montes e Alto Douro, Vila Real, Portugal
¶ Faculty of Engineering of University of Porto, Portugal, Email: paco@fe.up.pt

***Abstract*—Robotics competitions encourage the development of solutions to new challenges that emerge in sync with the rise of Industry 4.0. In this context, robotic simulators are employed to facilitate the development of these solutions by disseminating knowledge in robotics, Education 4.0, and STEM. The RobotAtFactory 4.0 competition arises to promote improvements in industrial challenges related to autonomous robots. The official organization provides the simulation scene of the competition through the open-source SimTwo simulator. This paper aims to integrate the SiwTwo simulator with the Robot Operating System (ROS) middleware by developing a framework. This integration facilitates the design of robotic systems since ROS has a vast repository of packages that address common problems in robotics. Thus, competitors can use this framework to develop their solutions through ROS, allowing the simulated and real systems to be integrated.**

***Index Terms*—Industry 4.0, Education 4.0, Autonomous Mobile Robots, Robotics Simulators, ROS, Fiducial Markers, Indoor Localization System, Pose Estimation**

I. Introduction

Robotics competitions and challenges are essential in the Education 4.0 scope, stimulating creative thinking and improving the skills of young students and professionals [1]. Simulators are also essential educational and development tools in robotics helping reduce the costs of these activities since they dispense the developers' need to possess a real robotic system [2]. Thus, robotics applications in simulators make it possible to observe the results of interactions beforehand in the simulated environment to be applied to solve challenges in the physical world [3].

The RobotAtFactory 4.0 competition aims to foster the development of solutions to challenges inherent to operations of autonomous mobile robots (AMRs) on the factory floor, covering aspects of simulators and real robotic systems. The competition scenario happens in real life and consists of a supply warehouse with two processing machines. Its primary goal is to encourage the development of AMRs by the attendees, capable of transporting boxes between the warehouse's input and output and its machines in the least possible time [4].

To facilitate the design of robotic systems by the competitors, a simulation model of the real competition scenario is provided through the SimTwo robotic simulator, which enables adaptations in the robot's structure and its sensory system for the development of the localization system needed for the robot's navigation in the simulated and real environment [4]. SimTwo is an open-source simulator based on the Pascal programming language. It offers tools for developing different robotic platforms, enabling the dynamic simulation of rigid bodies in a 3D visualization interface. SimTwo provides three distinct types of communication, UDP and ZMQ, which can be used to send data remotely, and serial communication that can be applied to Hardware-in-the-loop approaches [1], [5].

Some of the leading robotics simulators used in the educational and industrial environment, such as Gazebo [6] and CoppeliaSim [7], have integration with the Robot Operating System (ROS). It is a widely used middleware that provides an abstraction layer for hardware and software configurations in robotics, generalizing its access to external hardware, either with sensors or actuators embedded in different robotics platforms [8]. Thus, the contribution of this work is to provide integration through a framework for SimTwo with ROS.

The communication established through the framework allows the ROS available functionalities to be applied in solving the challenges proposed by the competition and developing other robotic applications that can be validated through the integrated systems.

The structure of the paper is as follows: Section II surveys the state of the art of functionalities of the ROS middleware and its integration with robotic simulators. Section III describes the RobotAtFactory 4.0. Moreover, the competition scenario in the SimTwo simulator is presented in Section IV. In addition, Section V describes the proposed framework that integrates the simulator with ROS. Also, the main results obtained through tests to validate the framework is detailed in Section VI. Finally, Section VII presents the conclusions and future work.

II. State of the art

With the evolution of computational devices, simulators have become powerful tools to support research and development, enabling the analysis and validation of virtual models of potential solutions to problems encountered in the

978-1-6654-8217-2/22/$31.00©2022 IEEE

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on February 27,2023 at 11:12:50 UTC from IEEE Xplore. Restrictions apply.

real world [3]. By designing new simulation elements and investigating their performance by adopting different methods and techniques related to the object of research, simulation allows evaluating the structures, characteristics, and functions of a robotic system [9].

As the complexity of the developed solutions increases, it becomes necessary to use resources that facilitate the development process of the hardware and software of the robotic devices. In this instance, ROS is an essential open-source tool in robotics, widely used in industrial and academic research, being integrated with several robotic simulators such as Gazebo, CoppeliaSim, Webots, MORSE, among others [3].

ROS provides low-level hardware abstraction, device control layer, packet management system, and communication-based abstraction. It is based on a graph architecture where the process takes place on nodes that can receive and send messages to sensors, actuators, robotic devices, and others [3].

Several works found in the literature use ROS features to simplify a mobile robot's controlling process in simulation and transfer it to the physical system [10], [11], [6]. For example, the Gazebo simulator does not have its motion planning functionality for its robotic devices. However, due to its compatibility with ROS middleware, integrated features and packages can be applied to Gazebo for this and other functionalities [9].

Due to the expressive adherence of ROS in different types of simulators, a vast amount of development libraries are constantly provided by the ROS community. Thus, it accelerates the development of new applications in the context of Industry 4.0 (I4.0) [11], mobile robotics [6], digital twins [12], precision agriculture [13], among others.

Other work involving ROS in simulators is based on the use of computer vision ROS packages for fiducial markers identification, making it possible to perform the reading of information packets related to positioning and map location in the virtual world [14].

Other research makes use of the ROS interface to access the readout of different types of sensors commonly used in robotic architectures for autonomous navigation, such as LIDAR, RGB-D camera, Intertia Measurement Units [15], [16]. In this way, it becomes much easier to track the performance metrics of the robot and analyze its behavior before its deployment in the real world.

III. The Competition

As previously mentioned in Section I, the competition tries to mimic a fully automated industry-warehouse environment with boxes organized by AMRs. Thus, the robots must self-localize and navigate whilst avoiding collisions. In addition, they should identify the types of boxes and collect and transport them to their respective spots in the least time possible. The official competition floor is displayed in Figure 1.

The boxes that the AMRs must organize are displaced in the four blank rectangles represented in Figure 1 where it is indicated "incoming warehouse". There are three types of boxes, the first one, called the raw box, it must be transported to machine type A, soon after machine type B, and then finally to the outgoing warehouse. The second type, the intermediate box, must be moved to machine type B and later the exit. Lastly, the third one, called the final box, must be displaced directly to the outgoing warehouse. The RobotAtFactory 4.0 competition was updated from the classic competition Robot@Factory. [17] to address the I4.0 concept. Thus, the competition floor and the walls of the input and output warehouses are filled with fiducial markers that the competitors can use for localization [18], [19]. Figure 1 displays the markers' IDs and their respective positions alongside the world's coordinate frame.

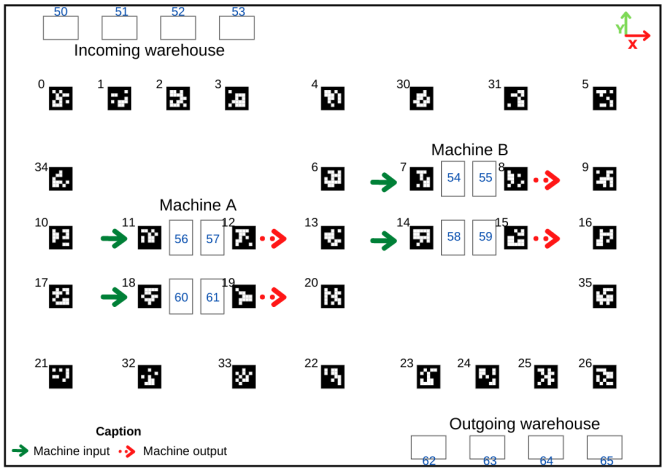

Fig. 1. RobotAtFactory 4.0 official competition floor's illustration (adapted from [4]).

The artificial markers displaced in the competition environment displayed in Figure 1, use the ArUco 5x5 marker pattern of codification. There is a restriction to the size of the AMR, and it must fit into a cube of $30 \times 30 \times 30$ cm. There is neither a restriction on the number of robots that a team can use simultaneously nor which type of localization system the robots can have. The robots must be completely autonomous and cannot establish any communication with an external system despite the one provided by the organization. This system is a task assignment server that the robot can communicate with to request info about which boxes are in the incoming warehouse and if the slots in the machines are occupied. The competition is divided into three rounds with increasing difficulty by including raw and intermediate boxes. To profoundly understand the competition characteristics and rules, the reader is referred to [4].

IV. Simulation Environment

As previously mentioned in Section I, the simulation environment used by the RobotAtFactory 4.0 competition is the SimTwo simulator, where it works with rigid-body dynamics interactions and constraints [20]. Moreover, it has realistic dynamics and sensory errors compared to the real scenario [21]. It achieves these features using the Open Dynamics Engine (ODE) for the physics simulation, a high-performance physics

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on February 27,2023 at 11:12:50 UTC from IEEE Xplore. Restrictions apply.

engine that provides quality rigid-body dynamics simulation [22]. The simulator implements its graphical aspect using GLScene components, a simple implementation of OpenGL [20]. Figure 2 displays the virtual representation of the official RobotAtFactory 4.0 competition scene inside the simulator.

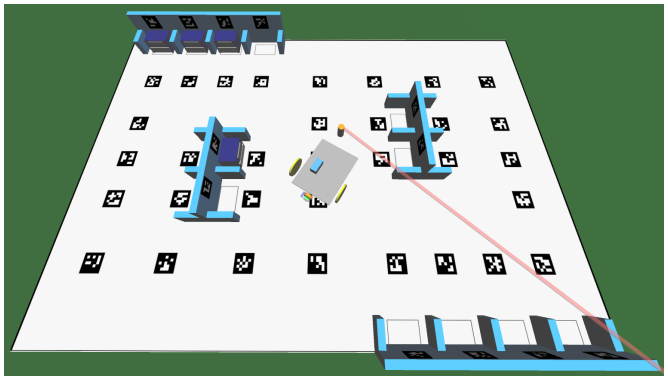

Fig. 2. ArUco markers' and world's coordinate frame position of RobotAtFactory 4.0 competition [4].

As one can see in Figure 2, the simulated scenario follows the description of the official competition rules. The user can interact with it by several windows, one of them being the XML editor. The simulator reads the main XML file whenever a new scene is loaded. Inside this file, the user can describe objects and their respective characteristics inside the world, such as solids, motors, robots, shells, sensors, controllers, meshes, and so forth. For example, if a user wants to describe a DC engine inside a robot, first, it is needed to define a robot object in the world. Afterward, inside the robot's XML description, the user can design the robot with its dimensions, sensors, driving architecture, etc. Also, in this XML, the programmer can define the motors that drive the wheels. In this reasoning, it is necessary to express several entries representing the DC motor characteristics such as weight, gear ratio, armature's resistance and inductance, static and viscous friction, type of controller, etc. The simulator has other features, such as the code editor, which allows the developer to program it. In addition, there is the sheets window which is essentially a matrix of cells that the simulator and the user can read and write data in it. Moreover, it is possible to interact with the code editor by creating buttons in the cells. There are other features, namely the charts' window, which allows for the visualization of the states of every defined robot in the scene. Furthermore, it supports the creation of custom real-time charts in the code editor. Since describing SimTwo in detail is not the scope of this work, the reader is referred to [20] for more information.

A simulated robot is required to develop this framework. Therefore, this simulation model was based on a real differential-drive AMR designed and assembled for the competition. The robot picture, alongside its high-level architecture block diagram, is presented in Figure 3. As one can see, the Raspberry Pi acts as the robot's brain where it commands the high-level control of the robot, i.e., the LIDAR, RGB camera, sensor fusion, localization, navigation, and decision making. In turn, the Arduino Uno commands the low-level control of the robot, that is to say, the motors, encoders, the contact switch, and the electromagnet. The AMR features a 2D LIDAR sensor, an RGB camera, and encoders for localization purposes. A contact switch was coupled on the robot to identify if it could take a box. Finally, an electromagnet was embedded so that the AMR could pick the box.

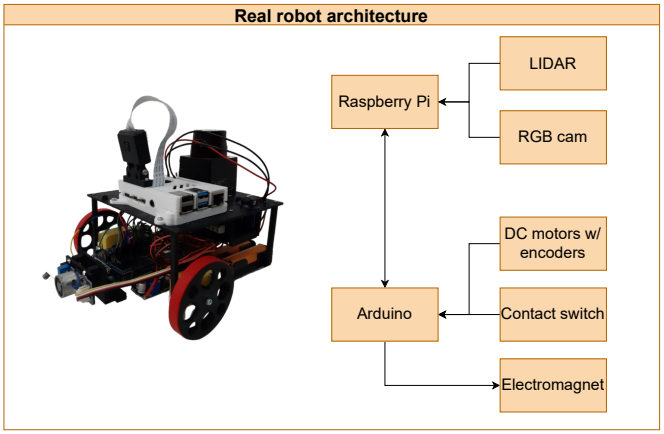

Fig. 3. The AMR that the simulated model was built from, "totta".

V. Developed Framework

This section will address the proposed framework for the RobotAtFactory 4.0 competition environment in SiwTwo simulator with the ROS middleware, mainly discussing the implemented communication structure and the preliminary localization system developed for the mobile robot.

A. Communication Structure

The integration between the SiwTwo simulator and the ROS middleware demands communication between the two systems. That is, data needs to be sent across these platforms. For this purpose, a communication structure was developed using UDP and ZMQ protocols native to SiwTwo.

The simulator only supports sending bitmaps of the camera sensor through ZMQ communication. The other data is sent through datagrams with UDP communication, as shown in Figure 4. ZMQ protocol is a messaging library that provides several types of communication patterns such as the publisher and subscriber and task distribution in an asynchronous I/O model [23].

The image sensor in SimTwo is an RGB camera with a resolution of 640×480 pixels, and the type of bitmap that it sends through ZMQ communication is the 32 bits RGBA codification. It is important to note that it is possible to configure the frequency of the communication in the simulator, and it is set by default to $f = \frac{FPS}{4}$. Where the f is the messaging frequency, FPS is the frame rate of the simulator that can be adjusted, and the divisor it is a constant that can also be modified.

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on February 27,2023 at 11:12:50 UTC from IEEE Xplore. Restrictions apply.

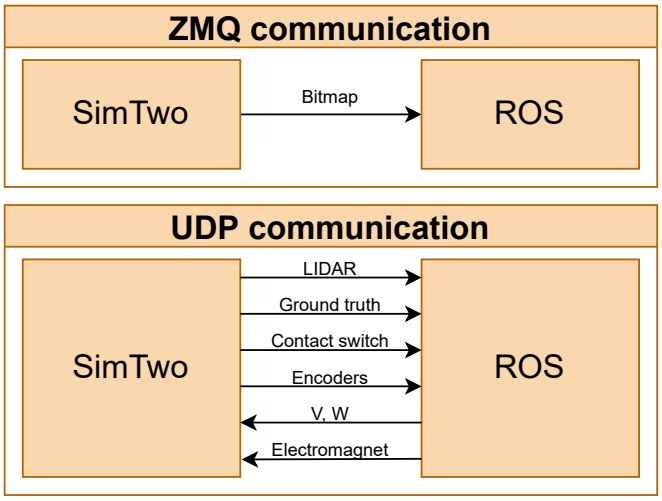

Fig. 4. The types of communication between SimTwo and ROS framework, which are exchanged in each SimTwo's control cycle.

As previously mentioned, the rest of the data is sent through UDP datagrams. The information is encoded in a string with labels for each value and then posteriorly decoded on the ROS side. SimTwo in each control cycle sends the values of LIDAR sweeps, ground truth (actual pose for testing), contact switch, and the encoders of the motors, as displayed in Figure 4. Soon after, the ROS framework makes the necessary computations and sends back via UDP datagram the robot's speed states V and W, and the command to turn on/off the electromagnet.

B. Preliminary localization system

Despite the simulator having a ground truth for localization, the objective is that the developers use the simulator to validate their approaches in the simulation scene before performing the transition to the real scenario since realistic simulators provide several advantages during the development phase [5]. Thus, a preliminary localization system was made so that the users that will take advantage of this framework could also have a starting point for their respective approaches to the localization system of their AMRs. In this way, the odometry is computed for the robot's simulated model. If the user modifies the simulated model, the odometry must be changed accordingly, especially if the driving architecture changes. In addition, the robot's pose, which means position and orientation relative to the world's frame, are also estimated using the RGB camera and ArUco markers. Finally, with the LIDAR sweeps available, the user can implement an approach for localization such as Perfect Match [24], or using reflectors [25], since the competition allows the placing in the competition's floor reflectors, beacons, etc [4]. Afterward, a position filter such as an Extended Kalman Filter (EKF) or a Particle Filter can be implemented to fuse all this data to get a reliable pose estimation.

For convenience, the optical center of the RGB camera was considered the robot's center. In this idea, the pose estimations from the odometry and LIDAR must pass through the necessary homogeneous transformations to reach the robot's center. In the same reasoning, if the developer wants to consider another position on the robot as the robot's center, the necessary homogeneous transformations should be computed for every sensor's pose estimation.

The pose estimation using the RGB camera and the ArUco markers was performed using the ArUco module from OpenCV library [26]. As mentioned earlier, the localization system is preliminary so that the programmers can implement their approaches. Thus, just for validation purposes, the pose estimation of the camera's optical center can be computed from every visible ArUco marker in a given camera frame. The way it works is that every tag has its ID. Moreover, the robot has stored every marker's absolute pose associated with each respective ID in its database, i.e., the position and orientation relative to the world's frame. Thus, with the aid of OpenCV, every time the algorithm detects a marker, its ID is decoded, and its position and orientation relative to the camera's frame are computed. The algorithm cannot detect when the features of the tag are not apparent by it being too distant or too rotated relative to the camera. In other words, the algorithm derives the translational and rotational vectors of the marker relative to the camera's frame. After that, some homogeneous transformations must be performed to estimate the pose of the camera's frame in reference to the global frame. In this way, consider that T_N^{N-1} an homogeneous transformation of frame N relative to frame $N-1$. Equation 1 displays the homogeneous transformation:

$$T_{camera}^{world} = T_{aruco}^{world} \times (T_{aruco}^{camera})^{-1} \tag{1}$$

where T_{aruco}^{camera} represents the homogeneous transformation of the ArUco frame in reference to the camera frame, the parameters of this matrix are estimated through the ArUco module from the OpenCV library. By taking the inverse of this transformation matrix, we have the transformation of the camera frame in relation to the ArUco frame. Finally, T_{aruco}^{world} represents the transformation of the marker frame to the world's frame, where the parameters of this matrix are given by the absolute pose of the ArUco marker. Finally, by performing the multiplication of equation 1, the absolute pose of the optical center of the camera is obtained.

One may note that several markers can be detected in a given camera frame sent to ROS. Therefore, several estimations can be computed in just one camera frame. In this context, one approach to address this situation is by implementing an EKF to consider all the pose estimations from each ArUco detection. Moreover, by fusing with other exteroceptive and proprioceptive sensors' data, it is possible to obtain a reliable pose estimate for the robot. Furthermore, the Kalman gain can be adjusted to depend on the distance and orientation of each marker relative to the camera's frame, giving more weight to more reliable detections. That is, markers near or with a low angular disparity relative to the camera's frame, or both. This framework considers the closest tag to the camera to compute the pose estimate of the camera's frame.

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on February 27,2023 at 11:12:50 UTC from IEEE Xplore. Restrictions apply.

VI. RESULTS

Two tests were made to validate the framework. The first test was to validate the UDP communication alongside the low-level control of the AMR. Thus, a cubic hermit spline for linear speed was generated to reach 0.3 m/s in one second. After that, this reference signal was sent to the simulator so that the robot could follow it. The simulator, in turn, sent back the estimated speeds that were derived through encoders and the dynamic model of the AMR. Figure 5 displays the test result.

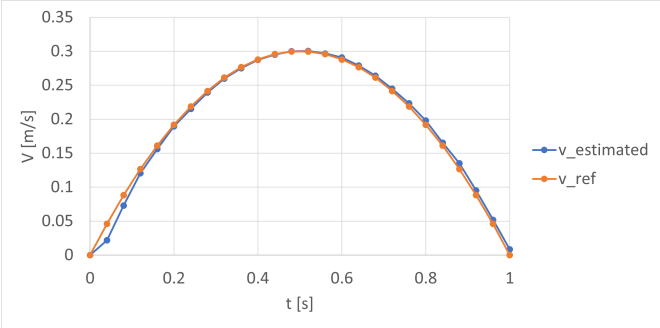

Fig. 5. Linear speed test between the SimTwo simulator and the ROS framework.

As one can see in Figure 5, the robot was able to follow the reference, validating the communication.

The second test was to validate the preliminary localization system made with the fiducial markers. In this way, Figure 6 displays a given frame received by the ROS framework already processed with the algorithm to detect the ArUco markers.

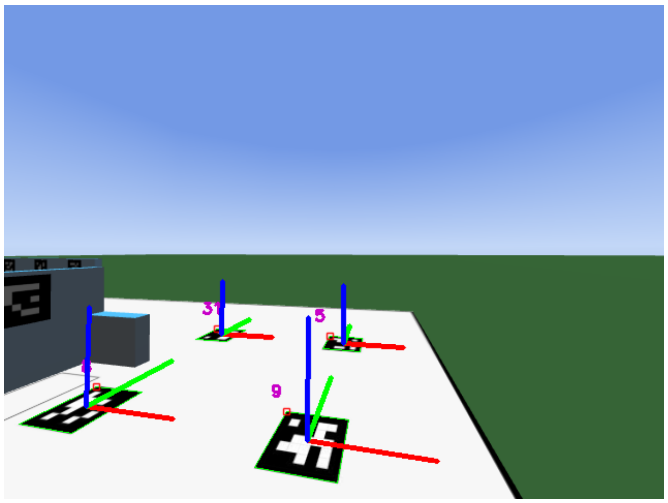

Fig. 6. Camera frame with the detected ArUco markers visualized in the ROS framework.

The blue axis in Figure 6 represents the Z axis, whereas the green axis represents the Y axis, and the red one the X axis. This feature from the framework is just for visualization and debugging. The important part is the pose estimate of the camera's optical center relative to the world's frame. In this way, Figure 7 displays the approach explained in Subsection V-B.

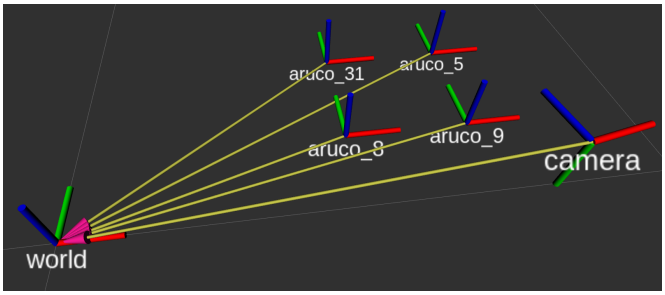

Fig. 7. Absolute pose of all the detected markers and the camera visualized in Rviz.

Figure 7 displays the coordinate frames of the world, each detected marker displayed in Figure 6, and the camera. It is important to note that the frames of the markers and the camera are relative to the world's frame since they were generated by the homogeneous transformations explained in the Subsection V-B. The yellow arrows denote the relation between the frames. Finally, the color codification of the coordinate system follows the same pattern as the one in Figure 6. Another way to observe the transformations is through Figure 8. This illustration was generated by the ROS package "rqt_tf_tree" which displays the system's coordinates transformations in a tree graph. The vertex "world" in the illustration represents the global reference frame, whereas "aruco_8" and "camera" vertices represent, respectively, relative to the world frame, the ArUco Tag 8, and the camera frames.

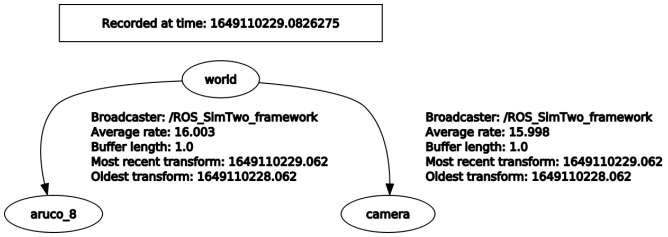

Fig. 8. Tree graph of the system's coordinates transformations.

VII. CONCLUSION AND FUTURE WORK

The simulators aid the development of robotics systems, saving costs, improving the management and planning of projects, and adding safety when dealing with complex environments. This work presents a SimTwo and ROS integration through a framework to facilitate hardware and software implementation and simplify the tests in simulation and the real scenario. This framework is focused on the RobotAtFactory 4.0 competition, but it can also be used as a starting point for other robotics applications in the simulator.

The adequation of the ROS interface in the simulator brings advantages in development terms of software and hardware when making the transition to the real scenario by providing a set of new development tools that helps in robotic systems.

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on February 27,2023 at 11:12:50 UTC from IEEE Xplore. Restrictions apply.

Thus, supporting the learning of robotics in the academic environment and fostering the development of new solutions to the industry.

This framework will make it possible to extend to new autonomous navigation applications by using the ROS middleware. Therefore, new control, localization, and navigation strategies of robots in SimTwo will be possible in the scope of academy, industry, and research.

This framework was validated by performing two tests. The first test successfully validated the UDP communication and the low-level control architecture of the AMR. The second test, in turn, validated the ZMQ communication alongside the preliminary localization system developed for the competition using the fiducial markers provided in the simulation scene.

For future work, a localization system will be implemented where the pose estimates of the camera from the fiducial markers will be inserted in the innovation phase of an EKF. In addition, pose estimates using the LIDAR sensor will also be considered in the innovation phase. The pose estimate from the LIDAR sensor will use reflectors in the allowed competition area. The EKF will use odometry as the state transition model in the prediction phase. The Kalman gain from the EKF of the ArUco markers will depend on several factors, such as the marker-camera distance and their relative orientation, since they impact the reliability of the pose estimate. This system will be validated in the simulation and the real scenario. Finally, since ROS Noetic has a vast quantity of packages and content, and the objective of this framework is to provide what ROS has to offer, the framework will be supported until ROS 2 has more content available.

ACKNOWLEDGMENT

This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the Project Scope: UIDB/05757/2020. The project that gave rise to these results received the support of a fellowship from "la Caixa" Foundation (ID 100010434). The fellowship code is LCF/BQ/DI20/11780028.

REFERENCES

[1] João Braun, Lucas A Fernandes, Thiago Moya, Vitor Oliveira, Thadeu Brito, José Lima, and Paulo Costa. Robot@Factory Lite: An Educational Approach for the Competition with Simulated and Real Environment. In *Iberian Robotics conference*, pages 478–489. Springer, 2019.

[2] Sokratis Tselegkaridis and Theodosios Sapounidis. Simulators in Educational Robotics: A Review. *Education Sciences*, 11(1):11, 2021.

[3] Caio Camargo, José Gonçalves, Miguel Á Conde, Francisco J Rodríguez-Sedano, Paulo Costa, and Francisco J García-Peñalvo. Systematic Literature Review of Realistic Simulators Applied in Educational Robotics Context. *Sensors*, 21(12):4031, 2021.

[4] Paulo Costa, José Lima, and Victor Pinto. Rules for RobotAtFactory 4.0 2022. https://github.com/P33a/RobotAtFactory/blob/main/RobotAtFactory_4_0_2022_Rules.pdf, 2022. Accessed: 2022-02-03.

[5] João Braun, José Lima, Paulo Gomes da Costa, and António Paulo GM Moreira. Robot@Factory Lite Competition: a Digital Twin Approach for the AGV. In *11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH 2021*, pages 311–318, 2021.

[6] Denis Chikurtev. Mobile Robot Simulation and Navigation in ROS and Gazebo. In *2020 International Conference Automatics and Informatics (ICAI)*, pages 1–6, 2020.

[7] Carlos Fernando Joventino, André Schneider de Oliveira, João Alberto Fabro, and Jonathas H. M. Pereira. Application of a ROS / CoppeliaSim Integration in a Practical "OBR" Competition Scenario. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, pages 1–6, 2020.

[8] Tatiana M. Pinho, A. Moreira, and J. Cunha. Framework Using ROS and SimTwo Simulator for Realistic Test of Mobile Robot Controllers. volume 321, 07 2014.

[9] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. A Review of Physics Simulators for Robotic Applications. *IEEE Access*, 9:51416–51431, 2021.

[10] Zandra B. Rivera, Marco C. De Simone, and Domenico Guida. Unmanned Ground Vehicle Modelling in Gazebo/ROS-Based Environments. *Machines*, 7(2), 2019.

[11] Laura Romeo, Antonio Petitti, Roberto Marani, and Annalisa Milella. Internet of Robotic Things in Smart Domains: Applications and Challenges. *Sensors*, 20(12), 2020.

[12] Alberto Martínez-Gutiérrez, Javier Díez-González, Rubén Ferrero-Guillén, Paula Verde, Rubén Álvarez, and Hilde Perez. Digital Twin for Automatic Transportation in Industry 4.0. *Sensors*, 21(10), 2021.

[13] Luiz F. P. Oliveira, António P. Moreira, and Manuel F. Silva. Advances in Agriculture Robotics: A State-of-the-Art Review and Challenges Ahead. *Robotics*, 10(2), 2021.

[14] Alexandre de Oliveira Júnior, Luis Piardi, Eduardo Giometti Bertogna, and Paulo Leitão. Improving the Mobile Robots Indoor Localization System by Combining SLAM with Fiducial Markers. In *2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*, pages 234–239, 2021.

[15] Nick Teslya, Alexander Smirnov, Artem Ionov, and Alexander Kudrov. *Multi-robot Coalition Formation for Precision Agriculture Scenario Based on Gazebo Simulator*, pages 329–341. 09 2020.

[16] Rafael Casado and Aurelio Bermúdez. A Simulation Framework for Developing Autonomous Drone Navigation Systems. *Electronics*, 10(1), 2021.

[17] Paulo José Costa, Nuno Moreira, Daniel Campos, José Gonçalves, José Lima, and Pedro Luís Costa. Localization and Navigation of an Omnidirectional Mobile Robot: The Robot@Factory Case Study. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 11(1):1–9, 2016.

[18] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand Ambrosi, Camden Whitehead, and Nikolaos Vitzilaios. Fiducial Markers for Pose Estimation. *Journal of Intelligent & Robotic Systems*, 101(4):1–26, 2021.

[19] Sara Roos-Hoefgeest, Ignacio Alvarez Garcia, and Rafael C. Gonzalez. Mobile Robot Localization in Industrial Environments Using a Ring of Cameras and ArUco Markers. In *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6, 2021.

[20] Paulo Costa, José Gonçalves, José Lima, and Paulo Malheiros. Simtwo Realistic Simulator: A Tool for the Development and Validation of Robot Software. *Theory and Applications of Mathematics & Computer Science*, 1(1):17–33, 2011.

[21] A Paulo Moreira, José Lima, and Paulo Costa. Improving a Position Controller for a Robotic Joint. In *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 97–103. IEEE, 2021.

[22] Russell Smith et al. Open Dynamics engine. 2005.

[23] Pieter Hintjens. *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, 2013.

[24] Martin Lauer, Sascha Lange, and Martin Riedmiller. Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-Localization. In *Robot Soccer World Cup*, pages 142–153. Springer, 2005.

[25] Sen Wang, Xiaohe Chen, Guanyu Ding, Yongyao Li, Wenchang Xu, Qinglei Zhao, Yan Gong, and Qi Song. A Lightweight Localization Strategy for LiDAR-Guided Autonomous Robots with Artificial Landmarks. *Sensors*, 21(13):4479, 2021.

[26] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Authorized licensed use limited to: b-on: Instituto Politecnico de Braganca. Downloaded on February 27,2023 at 11:12:50 UTC from IEEE Xplore. Restrictions apply.