Development of Simulation Environment for Socially Assistive Robots Testing Using ROS 2 and Gazebo

1st Muhammad Alfi Maulana Fikri
Department of Computer Engineering
Faculty of Intelligent Electrical
and Informatics Technology
Insititut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
maulana.17072@mhs.its.ac.id

2nd Mauridhi Hery Purnomo
Department of Computer Engineering
Faculty of Intelligent Electrical
and Informatics Technology
Insititut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
hery@ee.its.ac.id

3rd I Ketut Eddy Purnama
Department of Computer Engineering
Faculty of Intelligent Electrical
and Informatics Technology
Insititut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
ketut@te.its.ac.id

4th Muhtadin
Department of Computer Engineering
Faculty of Intelligent Electrical
and Informatics Technology
Insititut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
muhtadin@te.its.ac.id

***Abstract*—Over the past few years, robots have undergone significant developments. One of this development is socially assistive robots (SARs) which are able to assist users in the form of social interaction. However, due to their nature which involves direct interaction with the user, testing SARs could be difficult and risky. For this reason, in this study we propose a simulation environment for testing SARs created using the Gazebo simulator. In this simulation environment, the robot model will be tested virtually with a user model and other object models. In order for the test performed in the simulation could be applied to real robots, the controller system in the robot will be abstracted by separating each component into nodes using ROS 2. As a result, the user model and virtual room in the simulation could be used in the pose detection experiment and the SLAM experiment. The system created can produce the same action in moving the robot model and the real robot with an error difference of 2.6% in the simulation and 12.5% in the real world. Furthermore, With the performance of ROS 2, images delivery with a resolution of up to 640 x 480 can produce delays below 50 ms and frequencies above 90% on the same device and between devices both in the simulation and the real world.**

***Index Terms*—Simulation, Assistive Robotics, ROS 2, Gazebo.**

I. Introduction

Over the past few years, robots have undergone a significant development from mobile robots for education [1] to manipulator robots for industries [2]. One of that development is socially assistive robots (SARs). SARs are robots in the field of socially assistive robotics which combine aspects in assistive robotics and socially interactive robotics. Because of that aspects, SARs could provide assistance to users in the form of social interaction [3].

However, due to the nature of SARs that involve direct interaction with the user, testing of the robot could be difficult and risky for the user involved in the test [4]. One of the solutions to overcome this problem is to do virtual testing through robot simulation. In addition to minimizing risk, the use of robot simulations as a medium for robot testing could also reduce the required costs and save testing time during the development of the robot [5].

Until now, there are several simulators that could be used to run a robot simulation such as Webots [6], Gazebo [7], V-REP [8], OpenAI Gym [9], etc. However, these simulators are just platforms that are generally used to help robot development through a virtual simulation. While the development of the simulation environment and the robot controller for the simulation must be made by the robot developers themselves.

For that reason, in this study, we propose research related to the development of a simulation environment for SARs testing using ROS 2 and Gazebo. ROS 2 and Gazebo were chosen because of the availability of many libraries that could help the development and tests of robots, especially for simulations. In addition, with the presence of the hardware abstraction scheme in ROS 2, the robot controller which is tested through simulation could be transferred to a physical robot to be tested directly on the user [5].

II. Related Works

Several previous studies have been successful in developing a simulated environment for robots using ROS (the predecessor of ROS 2) and Gazebo. As done by Qian et al. [10] who developed simulations for manipulator robots, Zhang et al.

[11] who developed simulations for UAV quadrotor robots, and Takaya et al. [5] who developed simulation environments for mobile robots testing. However, in contrast to previous research, the research that we have done chooses to use ROS 2 so that the robot controller made for simulation has better performance and can work in real-time [12].

In addition, other research has also been carried out by Erickson et al. [4] who developed Assistive Gym, a simulation framework for assistive robotics based on OpenAI Gym. That simulation framework was then used by Clegg et al. [13] to develop learning methods through simulations for collaboration between robots and humans in helping humans wear clothes. However, because it does not use ROS, the robot controller created for simulation using that framework needs to be redeveloped when it will be tested directly on users using real robots. Nevertheless, research conducted by Zamora et al. [14] shows that the simulation in OpenAI Gym can also be integrated into ROS and Gazebo, so it is possible that Assistive Gym can also be used in conjunction with ROS 2 and Gazebo.

III. ROS 2 and Gazebo

A. Robot Operating System 2 (ROS 2)

Robot Operating System (ROS) [15] is a collection of libraries, drivers, and tools that facilitate system development on robots. ROS has Linux like command tools, interprocess communication systems, and various packages related to system development on robots.

Processes that are executed on ROS are referred to as nodes, communication between each processes uses the publish/-subscribe model, and the communication data sent between processes is referred to as a topic. A publisher process can submit one or more topics, then other processes that subscribe to that topic could get the contents of that topic. In addition, there is also a service that has a function like a topic, but it's done in two-ways direction. This service works by using a client/server model where a service client could send request data in the form of a service request and then a service server will send reply data in the form of a service response.

The second generation of Robot Operating System, ROS 2, is a continuation of ROS that brings reliability and performance for real-time use while still supporting the advantages of the previous ROS [12]. To meet the reliability and performance requirements for real-time use, ROS 2 uses Data Distribution Service (DDS) [16][17], the industry standard for real-time communication systems and end-to-end middleware, which replaces the previous ROS's interprocess communication system.

B. Gazebo

Gazebo [7] is a part of the Player Project [18] that allows a robot simulation and sensor application to work in a three-dimensional indoor or outdoor simulation environment. Gazebo has a client/server architecture and a publish/subscribe model for the communication system between processes. Each simulation object in the Gazebo can be associated with one or more controllers that will process commands to set and determine the state of an object. Data generated by a controller will be sent to shared memory using Gazebo interfaces (ifaces). Later, ifaces from other processes can read the data in shared memory, thus enabling inter-process communication between the program that controls the robot and the Gazebo simulator, regardless of the programming language used.

IV. System Design

A. Robot Design

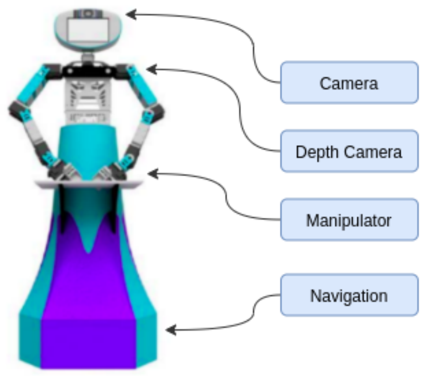


Figure 1. Diagram of the components in the *Dienen* robot.

The robot that will be used in this research is the Dienen robot which is a continuation of the IRIS robot [19][20] with the addition of the ICHIRO robot [21] design at the top of the robot. A design like this is generally known as a mobile humanoid robot design [22], which is a combined design between a mobile robot and a humanoid robot. As shown in figure 1, the lower part of the robot resembles a mobile robot with three omnidirectional wheels driving which allows the robot to move in a holonomic way [23], while the upper part of the robot resembles a humanoid robot consisting of a body, head, and arms. With the use of this mobile humanoid robot design, it is expected that users can experience better social interaction with the robot because it has a human-like appearance [24] while making it easier for the robot to navigate in various places.

The Dienen robot is equipped with several sensors such as an IMU (inertial measurement unit) sensor to determine the orientation of the robot, rotary encoder sensors to perform odometry calculations from the robot, distance sensors to detect the presence of other objects around the robot, a camera sensor in the head to capture images, and a depth camera sensor that can be used to do mapping of a room. In addition, this robot is also equipped with two manipulator arms that can be adjusted in various positions and orientations [25]. With the presence of these sensors and actuators, it is expected that the robot will be able to carry out socially assistive actions in accordance with the data obtained from existing sensors.

B. Robot Controller Design

The robot controller used for this simulation will be developed using ROS 2. The controller will be separated into

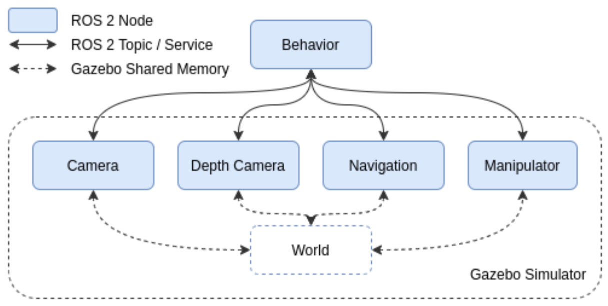


Figure 2. Diagram of the robot controller system in the simulation with the separation of behavior from the virtual sensors and actuators.

several parts in the form of ROS 2 nodes as shown in figure 2. Each existing node will be connected to each other using the ROS 2's interprocess communication system in the form of topics and services.

The main part of the robot controller is the behavior node which contains a program that regulates all robot actions based on data obtained from the sensors in the simulation. Then the behavior node will be connected to four other nodes that represent the sensors and actuators in the robot. Those four nodes will be attached in the scope of the Gazebo simulator as Gazebo plugins, so that they can be used to access and manipulate data in the simulation using the shared memory system in the Gazebo [26].

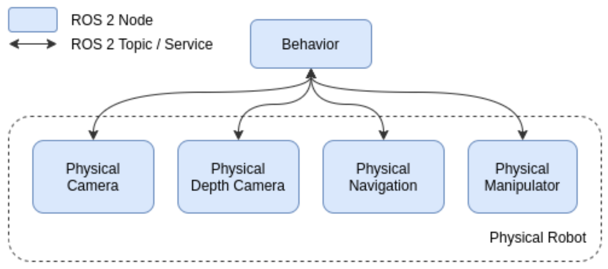


Figure 3. Diagram of the real robot controller system with the separation of behavior from the sensor and actuator devices in the robot.

This system is designed separately so that the behavior nodes tested in the simulation environment could be used directly on the real robot. As shown in figure 3, the transfer of controllers to the real robot can be done by changing the entire scope of the Gazebo simulator, which consists of the four nodes mentioned earlier, into nodes that process sensors and actuators on the real robot. With this, the tests carried out in the simulation can be directly applied when tested on the real robot because there is no need to redevelop the controller that adapts to the existing system on the real robot.

V. Experiments

A. Movement Testing

Movement testing is done by running the `move_for` node as a behavior node that will instruct the robot to move at a certain speed for a certain period of time. As shown in figure 4, in the simulation, the `move_for` node will be connected to the `navigation_plugin` node to control the speed of the robot using the `/navigation/maneuver_input` topic. As for testing in the real world, as shown in figure 5, the role of the `navigation_plugin` node which manages the navigation on the virtual robot will be replaced by the `navigation` node which manages the navigation on the real robot.

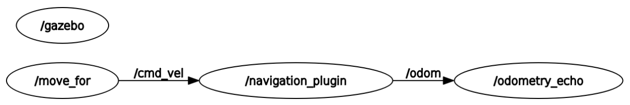


Figure 4. Node scheme of the movement testing in the simulation.

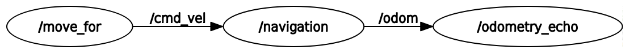


Figure 5. Node scheme of the movement testing on the real robot.

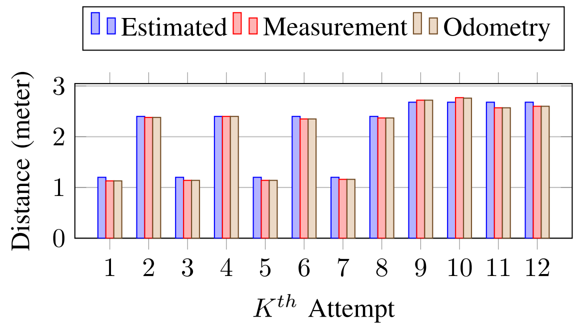


Figure 6. Comparison of last distance results from the expected and measured values in the simulation.

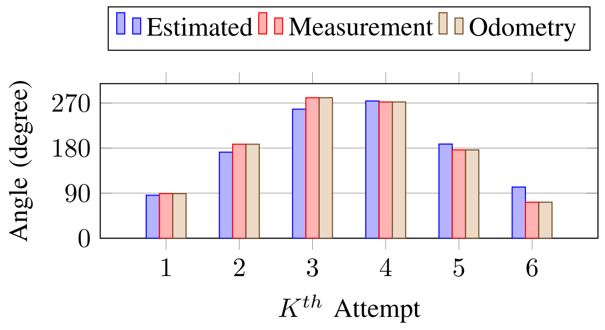


Figure 7. Comparison of last orientation results from the expected and measured values in the simulation.

The movement test is divided into two parts, linear movement testing and angular movement testing, each is tested with various combinations of speed values for 3 seconds on a virtual robot in a simulation environment and on a real robot in the real world. As shown in figure 6 and figure 7, the last distance and orientation of the odometry received by the robot relatively has the same value as the last distance and orientation of the estimated value from the given speed and the measurement from the starting point of the robot model in the simulation. From these results, the error percentage of the odometry value compared to the average of estimated and the measurement value in the simulation is 2.6%.

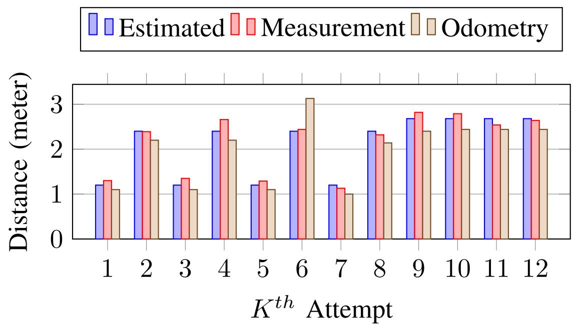


Figure 8. Comparison of last distance results from the expected and measured values on the real robot.

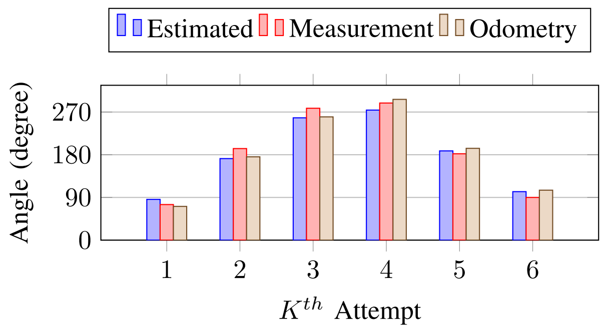


Figure 9. Comparison of last orientation results from the expected and measured values on the real robot.

In contrast to the testing results in the simulation environment, movement testing results in the real world have a little difference between the estimated and measurement value and the odometry values received by the robot. As shown in figure 8 and figure 9, there is a difference between the two value due to combinations of several factors like inaccuracies in the sensors and the occurance of slip when the robot move with high acceleration. Because of these factors, the error percentage of the odometry value compared to the average of estimated and the measurement value on the real robot is 12.5%. However, despite the differences in the level of accuracy, by using the same node behavior, the robot is capable of carrying out appropriate movement commands when tested in the simulation and the real world.

B. Image Transfer Testing

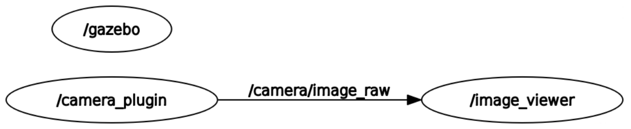


Figure 10. Node scheme of the image transfer testing in the simulation.

Image Transfer Testing is done by running the `image_viewer` node that will display the received image in the form of GUI.

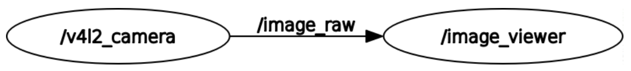


Figure 11. Node scheme of the image transfer testing on the real robot.

As shown in figure 10, in the simulation, the `image_viewer` node will be connected to the `camera_plugin` node that will send image data using the `/camera/image_raw` topic. As for the testing in the real world, as shown in figure 11, the role of the `camera_plugin` node will be replaced by the `v4l2_camera` node that will send image data which came from camera device on the real robot.

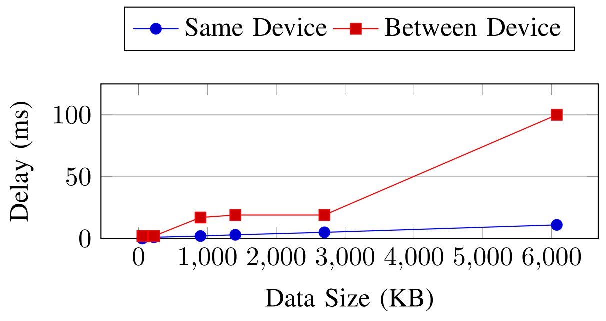


Figure 12. Delay results compared to the data size of image transfer in the simulation.

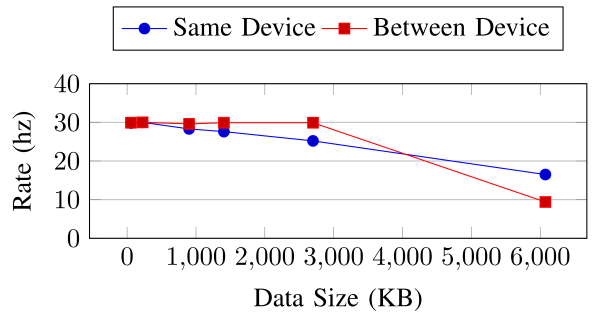


Figure 13. Rate results compared to the data size of image transfer in the simulation.

The image transfer test is carried out under conditions of transmission on the same device and on transmission between two different devices. Each of these tests is carried out with combinations of resolution values from 160x120 to 1920x1080 on a virtual robot in a simulation environment and on a real robot in the real world. As shown in figure 12 and figure 13, in the simulation, the resulting delay values tends to increase while the resulting rate values tends to decrease with the increase of the sent data. From these results, Up to 800x600 resolution ($\pm$2500 KB), image transfer in the simulation could produce low delays under 50 ms and stable rates above 90% of the normal value.

Meanwhile, on the real robot, as shown in figure 14 and figure 15, the obtained results are also relatively the same in which the delay values tends to increase while the rate values tends to decrease. However, different from the test results in

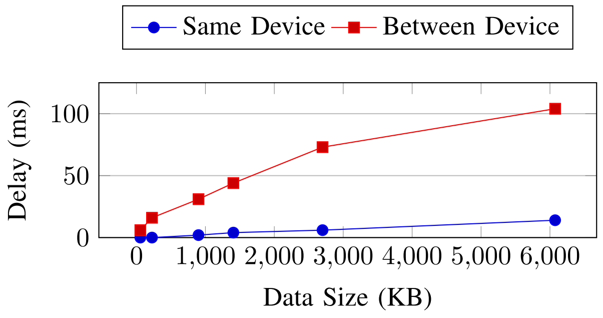


Figure 14. Delay results compared to the data size of image transfer on the real robot.

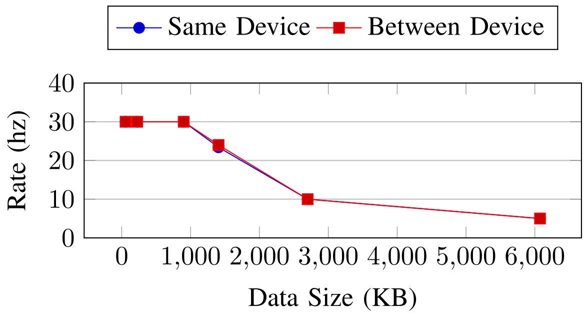


Figure 15. Rate results compared to the data size of image transfer on the real robot.

the simulation environment, from the results on the real robot, it is obtained that up to 640x480 resolution (±1500 KB), image transfer on the real robot could produce low delays under 50 ms and stable rates above 90% of the normal value.

C. Depth Camera Testing

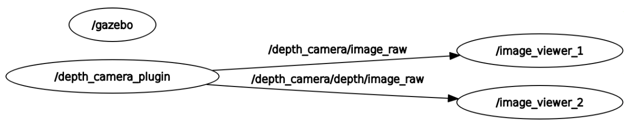


Figure 16. Node scheme of the depth camera testing in the simulation.

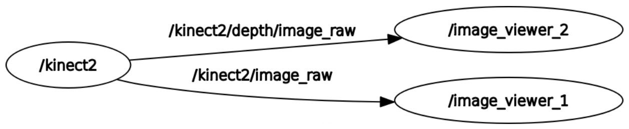
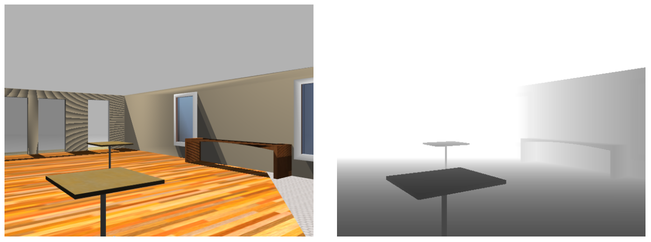


Figure 17. Node scheme of the depth camera testing on the real robot.

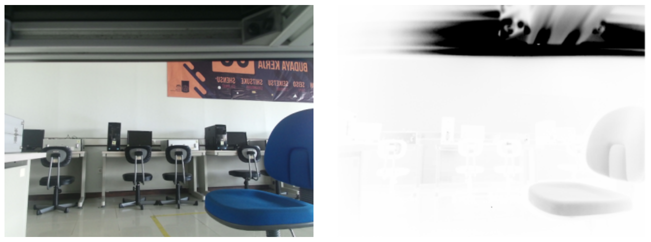
Depth camera testing is done to test the ability of depth camera in the simulation to simulate the depth camera device, while at the same time to test the hardware abstraction in the ROS 2. This test is done by running two `image_viewer` nodes that each of which will receive color images and depth images. As shown in figure 16, in the simulation, each `image_viewer` nodes will receive color image data using the `/depth_camera/image_raw` topic and depth image data using the `/depth_camera/depth/image_raw` topic. As for the testing in the real world, as shown in figure 17, the role of the `depth_camera_plugin` node will be replaced by `v4l2_camera` node that will send data which came from the Kinect V2 device on the real robot.



(a) Color image (b) Depth image

Figure 18. Comparison of color image and depth image capture results from depth camera in the simulation.

The result, as shown in figure 18, `image_viewer` node will display the data received from the simulation in which the left image (18a) shows a colored image, while the right image (18b) shows a depth image with black and white display. In the depth image, the brighter a point in the image indicates a position that is far from the center point of the camera.



(a) Color image (b) Depth image

Figure 19. Comparison of color image and depth image capture results from depth camera on the real robot.

Meanwhile, on the real robot the obtained results are also relatively the same. As shown in figure 19, the left image (19a) indicates a colored image while the right image (19b) indicates a depth image. However, in those images, the depth image has a brighter result, this happens because the range of the Kinect V2 is shorter (4.5 meter) than the range that was set on the depth camera in the simulation.

D. Pose Detection Testing

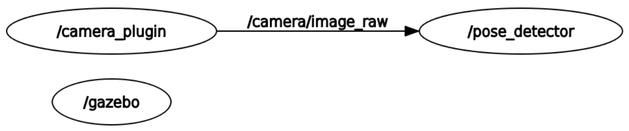


Figure 20. Node scheme of the pose detection testing.

Pose detection testing is done in the simulation with purpose to test the user model ability in simulating the real user. In this test, `pose_detector` node will be run to do a computer vision process in detecting user's pose. As shown in the figure 20, `pose_detector` node will be connected to the `camera_plugin` node that will send user image data which will then be processed by the `pose detector` node to produce the detected pose.

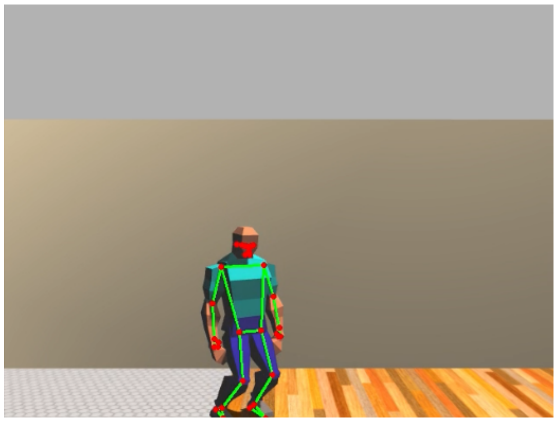


Figure 21. User pose detection results.

The result, as shown in the figure 21, from the received image, `pose_detector` node will detect the user's pose and display the result as lines and dots on the received image. Besides that, the pose detection process could also display the corresponded result when the user's posture is changed from standing to sitting as in the result of figure 21.

E. SLAM Testing

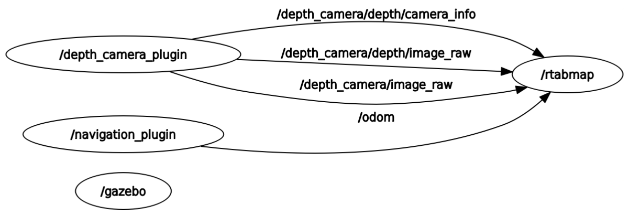


Figure 22. Node scheme of the SLAM testing.

SLAM (simultaneous localization and mapping) Testing is done in the simulation with purpose to test the virtual room ability in simulating the real room. In this test, the robot model will move around the room while doing the mapping using the SLAM method. As shown in the figure 22, `rtabmap` node will be connected to the `depth_camera_plugin` node and the `navigation_plugin` node that will send color image, depth image, and odometry data which will be used in the room mapping process.

The result, as shown in the figure 23, the room mapping process produce a three-dimensional map that composed of point clouds which shape the room that is tested in the simulation. That map is displayed on the GUI that owned by the `rtabmap` node. Besides that, The map also shows a visualization of the path that the robot has traversed as a line with a blue dot.

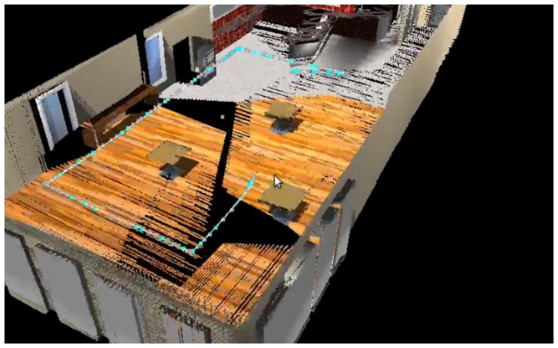


Figure 23. Room mapping results.

VI. Conclusion

From the discussion described in the previous section, it was found that by using the proposed system, a robot, especially SARs, can be developed and tested both in a simulation environment using a virtual robot and in the real world using a real robot. With this, robot development will be easier to do, more cost-effective, and safer to be tested on users because it can be developed virtually in a simulation environment, while not leaving the ability to also be tested on a real robot.

References

[1] P. Gonçalves, P. Torres, C. Alves, F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, vol. 1, 2009.

[2] M. Blatnický, J. Dižo, J. Gerlici, M. Sága, T. Lack, and E. Kuba, "Design of a robotic manipulator for handling products of automotive industry," *International Journal of Advanced Robotic Systems*, vol. 17, no. 1, 2020.

[3] D. Feil-Seifer and M. J. Mataric, "Defining socially assistive robotics," in *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, 2005, pp. 465–468.

[4] Z. Erickson, V. Gangaram, A. Kapusta, C. K. Liu, and C. C. Kemp, "Assistive gym: A physics simulation framework for assistive robotics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 169–10 176.

[5] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, "Simulation environment for mobile robots testing using ros and gazebo," in *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, 2016, pp. 96–101.

[6] O. Michel, "Cyberbotics ltd. webots™: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[7] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[8] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326.

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[10] W. Qian, Z. Xia, J. Xiong, Y. Gan, Y. Guo, S. Weng, H. Deng, Y. Hu, and J. Zhang, "Manipulation task simulation using ros and gazebo," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, 2014, pp. 2594–2598.

[11] M. Zhang, M. Lan, J. Lin, S. Wang, K. Liu, F. Lin, and B. M. Chen, "A high fidelity simulator for a quadrotor uav using ros and gazebo," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, 2015, pp. 002 846–002 851.

[12] Y. Maruyama, S. Kato, and T. Azumi, “Exploring the performance of ros2,” in *2016 International Conference on Embedded Software (EMSOFT)*, 2016, pp. 1–10.

[13] A. Clegg, Z. Erickson, P. Grady, G. Turk, C. C. Kemp, and C. K. Liu, “Learning to collaborate from simulation for robot-assisted dressing,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2746–2753, 2020.

[14] I. Zamora, N. Lopez, V. Vilches, and A. Cordero, “Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo,” *arXiv preprint arXiv:1608.05742*, 2016.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, “Ros: an open-source robot operating system,” *ICRA Workshop on Open Source Software*, vol. 3, 2009.

[16] G. Pardo-Castellote, “Omg data-distribution service: architectural overview,” in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, 2003, pp. 200–206.

[17] J. M. Schlesselman, G. Pardo-Castellote, and B. Farabaugh, “Omg data-distribution service (dds): architectural update,” in *IEEE MILCOM 2004. Military Communications Conference, 2004.*, vol. 2, 2004, pp. 961–967 Vol. 2.

[18] B. Gerkey, R. Vaughan, and A. Howard, “The player/stage project: Tools for multi-robot and distributed sensor systems,” *Proceedings of the International Conference on Advanced Robotics*, 2003.

[19] R. Dikairono, Setiawardhana, D. Purwanto, and T. A. Sardjono, “Cnn-based self localization using visual modelling of a gyrocompass line mark and omni-vision image for a wheeled soccer robot application,” *International Journal of Intelligent Engineering and Systems*, 2020.

[20] Muhtadin, R. M. Zanuar, I. K. E. Purnama, and M. H. Purnomo, “Autonomous navigation and obstacle avoidance for service robot,” in *2019 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, 2019, pp. 1–8.

[21] M. Muhtadin, M. Arrazi, S. Ali, T. Pratama, D. Wicaksono, A. Putra, I. Ari, A. Maulana, O. Bramastyo, S. Asshakina, M. Attamimi, M. Arifin, M. Hery Purnomo, and D. Purwanto, “Ichiro robots winning robocup 2018 humanoid teensize soccer competitions,” in *RoboCup 2018: Robot World Cup XXII*, 2019, pp. 425–435.

[22] Z. Mohamed and G. Capi, “Development of a new mobile humanoid robot for assisting elderly people,” *Procedia Engineering*, vol. 41, pp. 345 – 351, 2012.

[23] H. Oliveira, A. Sousa, A. Moreira, and P. Costa, “Dynamical models for omni-directional robots with 3 and 4 wheels,” *ICINCO 2008 - Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics*, vol. 1, 2008.

[24] S. Rossi, M. Staffa, and A. Tamburro, “Socially assistive robot for providing recommendations: Comparing a humanoid robot with a mobile application,” *International Journal of Social Robotics*, 2018.

[25] J. Iqbal, M. Ul Islam, and H. Khan, “Modeling and analysis of a 6 dof robotic arm manipulator,” *Canadian Journal on Electrical and Electronics Engineering*, vol. 3, pp. 300–306, 2012.

[26] “Gazebo plugins,” 2021. [Online]. Available: http://gazebosim.org/tutorials?tut=ros_gzplugins