Development of AlphaBot2 Gazebo simulator for RPi camera

Ana Rafael
Porto, Portugal
up201405377@fe.up.pt

Cássio Santos
Feira de Santana, Brasil
sss.cassio@gmail.com

***Abstract*—In recent years robots have been used as a motivational tool for teaching notions in several areas of knowledge. Despite the accessible price of such robots, the fact that these have to be acquired for several students and college resources are limited, induce a shifting of the development towards a fully virtual environment reduces these physical demands, allows for a safer testing environment. A simulator such Gazebo allows the visualisation of a model, the reproduction of physical properties and the interfacing with the Robotics Operating System (ROS) for message passing between nodes. The proposed system architecture focuses in the physical behaviours of the robot, and the camera sensor. Both the real and simulation model were tested through a line following algorithm. It was verified that the implemented ROS structure was functional either in the simulation as well as in the real robot for the robot control and camera activation. Further work in the validation and testing is tested for the assessment of the correct model performance.**

***Index Terms*—AlphaBot2; Gazebo; Robotics Operating System (ROS);**

I. Introduction

Autonomous intelligent vehicles present an interesting development for the creation of self driving cars with capability to perform a number of tasks in different environments [1]. Through the perception of the environment, with cameras and sensors for the recognition of lines in the paving, signs and lights in the air, obstacle identification and avoidance, the robot is expected to perform accordingly to the road traffic regulations [2].

Besides that, the use of robots for teaching purposes has proven to be a motivational tool for students to learn science, technology, engineering and mathematics (STEM). However, due to low budget for resources which limit the number of available robots in a university and the necessity of repair in case of damage the limited access to these materials may be a constraint for the development and experimenting phases. Robot simulations allow for an inexpensive and fast debugging process, experimental repeatability and easy alteration in the environment and robot dynamics, giving the possibility to experiment changes in numerous parameters without compromising the real robot [3]. Such platforms provide a stand alone environment, resulting in no restrictions from lab hours or power management inherent to all physical systems. The AlphaBot 2 (represented in fig. 1) is a compact two wheeled robot with infrared (IR) sensors, line trackers, a RPi camera (oriented by two servo motors), a micro SC card, a 5V/2.5V USB adaptor, IR remote, and some additional components. Easily configured and with demo codes for its basic behaviours, described and available in [4]. The present work aims to produce a Gazebo simulation of the AlphaBot2, test and finely tune the virtual model for verification of similar performances between the real and simulated robot specifically for the RPi camera. The present paper starts by presenting a description of the developed robot and the implemented simplifications to it, in section II. Additionally, the used world for testing and the ROS message-passing architecture is represented in order to have a fare understanding of the developed tasks.The section III presents the results and tested parameters; And finally section IV outlines the main conclusions and future work.

II. Methodologies

This section yields a description of the developed robot's model in a 3D simulator, the implemented world, a simplification of the Portuguese autonomous driving competition and the ROS architecture.

A. Design of the Robot Model

For the design of the robot in the virtual world, first the main measurements for the several components were acquired and are detailed in table I.

TABLE I: AlphaBot 2 measurements.

Components	**Weight (g)**	**LWD[a] or DT[b] or D[c] (cm)**	**Height (cm)**
Base board	150	10.9x0.15	1.6
Pi Board	45		
Wheels		4.1x1.7	
Balance Wheel		1	1.5
Camera	55	3.7x3.3x7.5	
Pillars			3.8
Raspberry Pi	40	8.4x5.5x1.7	
Total			**15.5**

[a] Length Width Depth.
[b] Diameter Thickness.
[b] Sphere Diameter.

Hence, for the design of the robot in the Gazebo simulator a URDF (*.xacro*) file was written. In it the links define the inertial values, collisions and visuals for each of the AlphaBot2 components. For simplification reasons the robot

Fig. 1: AlphaBot2 front view.

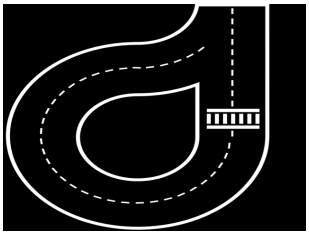

Fig. 2: Adaptation of Conde's track.

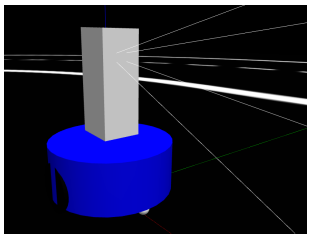

Fig. 3: Model of the AlphaBot2 in the Gazebo simulation.

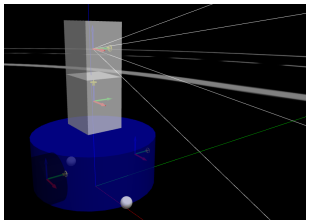

Fig. 4: Visualisation of the model joints in the Gazebo simulation.

was designed resorting to simple shapes line cylinders and boxes, the final model is represented in fig. 3. Thus, the inertial values are calculated for these shapes having in consideration each of the components' weight. The joints locate the points of articulation between elements and the plugins set the needed connections between the Gazebo simulation environment and the ROS. Since the camera orientation is controlled through two motors for pan-tilt action interfaced with the Gazebo plugin *libgazebo_ros_control.so* and the camera is activated with the *libgazebo_ros_camera.so*. As to the wheels these are controlled with the *libgazebo_ros_diff_drive.so*. A representation of the model joints is presented in fig. 4, the axis of rotation for each has a circular arrow indicating the allowed movement. The designed word is constituted by and adaptation of the Conde's track, as represented in fig. 2.

B. System architecture

The system architecture is based in the ROS meta-operating system. Through the implementation of nodes which publish and subscribe to specific topics message-passing between processes is accomplished. Two levels of packages (pkg) are responsible for the control of the robots behaviour in the Gazebo simulation and in the real robot. Hence, on the side of the Gazebo simulation two nodes are implemented for the readout of the camera sensor, which publishes a raw image, and a pkg for the control of the joints present in the camera support (motions of pan and tilt), this subscribes to the angle positions for the two motors. The connections between nodes using ROS for the real the robot are presented in fig. 5, and in fig. 6 is shown the simulator architecture. Using these packages, students and developers can program and test their own applications subscribing or publishing to the topics that will be described next.

1) Ros Topics:

- "/alphabot2/control": Publish a message of type *geometry_msgs/Twist* setting *linear velocity x* and *angular velocity z* to control the movement of the AlphaBot2;
- "/alphabot2/vertical": Publish a message of type *std_msgs/Float64* setting a value between -90 and 90 to control the tilt angle of the camera;
- "/alphabot2/horizonal": Publish a message of type *std_msgs/Float64* setting a value between -90 and 90 to control the pan angle of the camera;
- "/alphabot2/camera/image_raw": Subscribe to this topic to read a message of type *sensor_msgs/Image*.

2) System requirements:: For the reuse of the system, on the side of the simulator the minimum requirements are the Ubuntu 16.04, the ROS Kinetic and the Gazebo7. As to the real robot, the used system was Raspbian GNU/LINUX 9 (stretch).

III. Simulations and Experiments

In order to verify the correct implementation of the model and system architecture, nodes and topics, the robot was launched is an empty Gazebo's word (fig. 7), and messages were published to the topics /alphabot2/vertical and /alphabot2/horizontal through the *rostopic command-line tool*.

Using the command: *rostopic pub /alphabot2/vertical std_msgs/Float64 "data: -45"* the camera starts to face the ground, represented in fig. 8 where it is possible to see the new position of the Tilt and its vertical axis as the dashed red line which is tilted -45 degrees. To control the Pan a std_msgs/Float64 is published to the /alphabot2/horizontal topic using the command: *rostopic pub /alphabot2/horizontal std_msgs/Float64 "data: -45"* (Fig 9) and *rostopic pub /alphabot2/horizontal std_msgs/Float64 "data: 45"* (Fig 10). It is possible to see that the camera is positioned to the right of its initial position when a negative value is submitted to the horizontal topic and it is positioned in the left if a positive value is submitted.

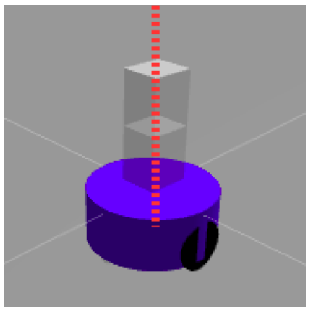

Fig. 7: Initial position

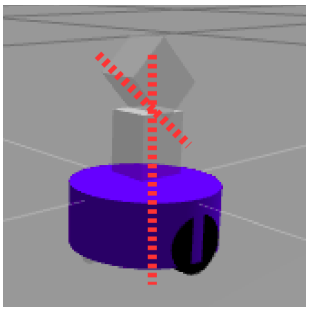

Fig. 8: $-45°$ at Tilt

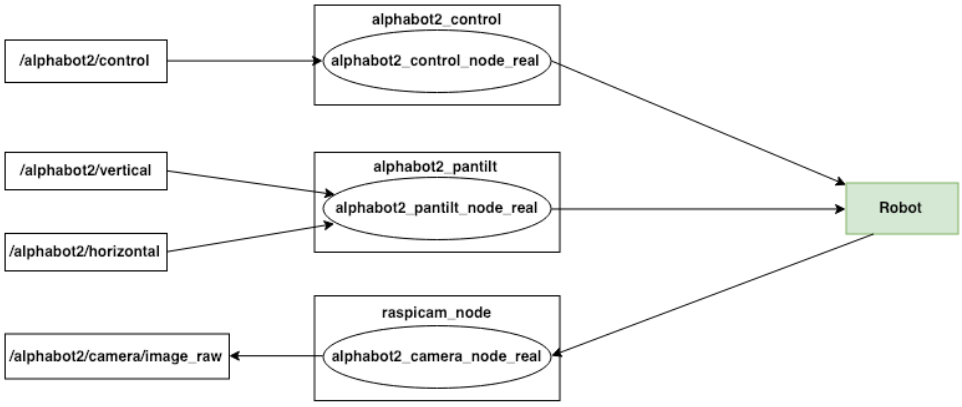

Fig. 5: ROS architecture for the real robot.

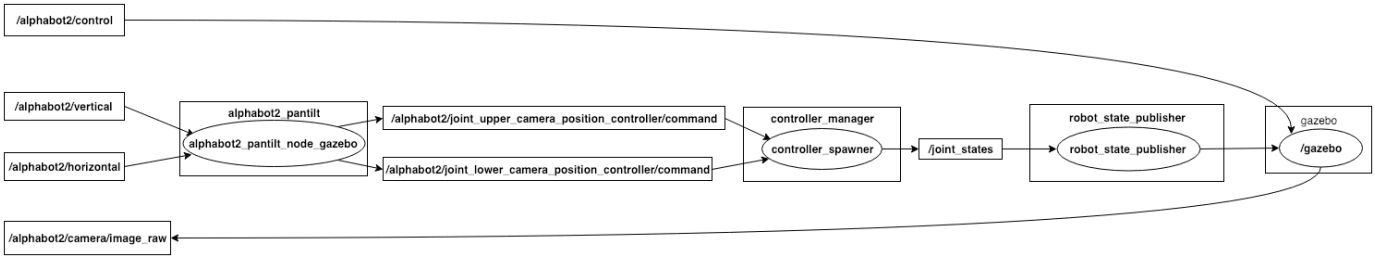

Fig. 6: ROS architecture for the simulator

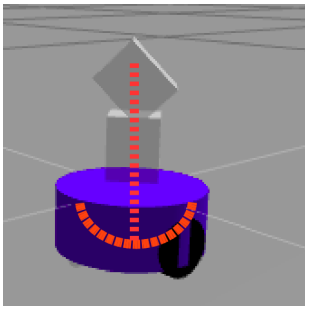

Fig. 9: $-45°$ at Pan

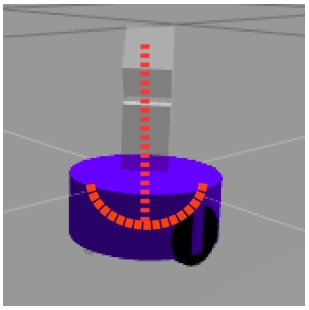

Fig. 10: $45°$ at Pan

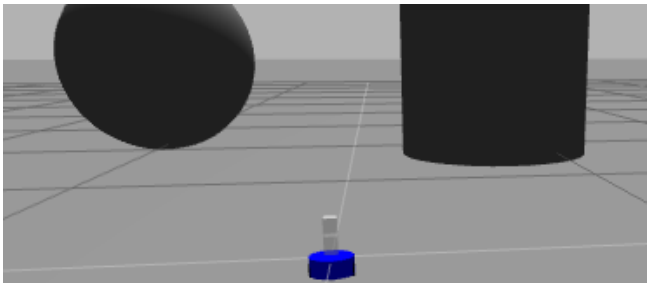

Fig. 11: Gazebo simulation with simple objects

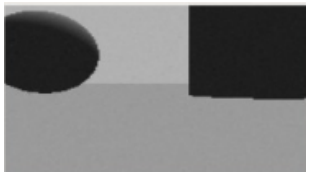

Fig. 12: RGB image captured from the camera

Fig. 13: Canny Edge Detection

For the verification of the camera functioning in the Gazebo simulation a package called alphabot2_tracking was created to subscribe to the /alphabot2/camera/image_raw topic. With the robot spawned on the a world with simple objects (fig. 11) it is possible to visualise both the RGB image captured from the camera (fig. 12) which shows that the gazebo library is working properly for the camera sensor. On top of this node the user as freedom to operate to what best fits its needs. fig. 13 exemplifies these transformations by applying a Canny Edge Detection for future line detection using Hough Transform. This information can latter be used for the control of the robots behaviour through, for example, a PID control for the adjustment of the robot's orientation with regards to that line [5].

IV. Discussion and Conclusion

In this paper it was presented a simulator and several ROS packages for the AlphaBot2 hardware abstraction and control of its low-level modules, thus allowing for a easy introduction of students to the field of robotics. The simulator was made using Gazebo as a 3D simulator and was tested by modelling the AlphaBot2 robot from Waveshare Eletronics. The simulator's components communicate with each other using the ROS framework which facilitates the user's work by suppressing concerns with low-level issues and hardware, making him only concerned with the logic behind the task. The user can then easily use the drive control of the AlphaBot2, its pantilt control and view information obtained by the camera.

The proposed objective of this paper was successfully achieved and a simulator regarding the AlphaBot2 was created. With relation to future developments improvements can be done in the *.xacro* model, more specifically, more accurate representations of the robot design in the Gazebo simulation could be achieved with, for example, *.stl* files for each of the components. However, these would yield superior computational requirements. Additionally, it is of paramount importance the evaluation of the developed model with the conduction of a comparison between the real robot and the simulated model. This can be conducted with a test setup for line following in the simulation and in the real world. Evaluating the travelled distance and the error between the relative line orientation and distance.

REFERENCES

[1] V. Costa, R. J. F. Rossetti, and A. Sousa. "Autonomous Driving Simulator for Educational Purposes". In: *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*. June 2016, pp. 1–5. DOI: 10.1109/CISTI.2016.7521461.

[2] C. Häne, T. Sattler, and M. Pollefeys. "Obstacle Detection for Self-Driving Cars Using Only Monocular Cameras and Wheel Odometry". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015, pp. 5101–5108. DOI: 10.1109/IROS.2015.7354095.

[3] Yuhanis Yusof, Mumin Abu Hassan, N. J. Mohd Saroni, et al. "Development of an Educational Virtual Mobile Robot Simulation". In: 2011.

[4] *AlphaBot2-Pi*. URL: %5Curl%7Bhttps://www.waveshare.com/wiki/AlphaBot2-Pi%7D.

[5] *Hough Transform in OpenCV*. URL: %5Curl%7Bhttps://docs.opencv.org/3.4.0/d6/d10/tutorial_py_houghlines.html?fbclid=IwAR2irZ4n7PMTvLVamUeq8iYV6f0cSSYDFkGyIsw%20B6buVffVSNJ39Cm1GgZg%7D.