Erle-copter Simulation using ROS and Gazebo

Krishneel Kumar, Sheikh Izzal Azid, Adriano Fagiolini, and Maurizio Cirrincione

Krishneel Kumar
School of Engineering and Physics
The University of the South Pacific
Suva, Fiji
krishneel.kumar@usp.ac.fj

Sheikh Izzal Azid
School of Engineering and Physics
The University of the South Pacific
Suva, Fiji
sheikh.azid@usp.ac.fj

Adriano Fagiolini
Department of Engineering
University of Palermo
Palermo, Italy
adriano.fagiolini@unipa.it

Maurizio Cirrincione
School of Engineering and Physics
The University of the South Pacific
Suva, Fiji
maurizio.cirrincione@usp.ac.fj

***Abstract*— The recent decrease in the price as well as size of semi-conductor logic and due to significant advancements in technologies such as microcontrollers, motors and sensors, the application of quadcopters in several fields has been achieved. However, testing of quadcopter prototypes still has a risk of damage due to faults and unexpected behavior. Hence, a method of testing of quadcopters in simulation mimicking the actual conditions of the real environment in an actual hardware test has been proposed. For this purpose, Gazebo simulator integrated with ROS has been chosen for the simulation of the path of the quadcopter. Moreover, the software Matlab/Simulink has been interfaced with Gazebo in order for the simulation of the quadcopter to be achieved.**

Keywords- ROS, Gazebo, UAV, GCS, autonomous vehicle, simulation, Erle-copter.

I. INTRODUCTION

A quadcopter is a flying object which changes its attitude and altitude by using four rotating blades [1]. The four rotors of the quadcopter are directed upwards and are placed in a square formation, at an equal distance from the center of mass of the quadcopter [2, 3]. Two major configurations of the quadcopter are possible: the plus (+) and the cross (x) configurations. According to [4, 5], an x configured quadcopter is considered to have more stability when compared to the + configuration which is more suited for acrobatic maneuvers.

Due to the recent advancements in technologies like microcontrollers, motors and sensors, the use of quadcopters in several fields has widely spread. These applications include natural disaster management, weather monitoring, forest fire detection, traffic control, cargo transport, emergency search and rescue, communication relaying, security, surveillance, agriculture and academic teaching such as control engineering.

However, due to the naturally unstable nature of quadcopters, hardware testing of the quadcopters still poses a risk of damage because of sudden faults and unexpected. Thus, a method for testing of quadcopters in simulation, mimicking the conditions of the real environment in an actual hardware test has been proposed. This practice is common also in other fields, such as the industry [6] and automotive ones [7], were testing on real experimental setups is only advisable when simulation results show that the proposed control technique is robustly working. For this purpose, Gazebo simulator integrated with ROS has been chosen for the simulation of the path of the quadcopter. The software Matlab/Simulink has been interfaced with Gazebo in order to achieve the simulation of the quadcopter.

This paper is organized as follows. ROS and Gazebo are introduced in Section II. Section III discussed the Gazebo simulation environment setup. Section IV presents the simulation steps of the Erle-copter in Gazebo. Finally, in Section V presents the simulation results obtained from Gazebo. The paper is concluded in Section VI.

II. ROS AND GAZEBO

Robot Operating System (ROS) is a meta-operating system used for robots. The services provided by ROS are similar to any operating system, and may include abstraction of hardware, control of low level devices, ability to pass messages between processes and the management of packages. Moreover, it provides users with libraries and tools to help build, write and run codes across multiple computers. With the use of ROS, users are provided with a variety of communication styles like synchronous RPC style communication over services, asynchronous streaming of data over topics and storage of data on a parameter server [8]. ROS currently works either on systems based on UNIX platforms or Mac OS systems [8].

Gazebo is a robotics simulator, which can be utilized for creating applications for real robots in a virtual environment. This software can be used to simulate robots with actual world parameters to visualize the behavior of the

978-1-7281-5200-4/20/$31.00 ©2020 IEEE

robot in the actual hardware tests environment. Gazebo-simulated hardware is designed to reflect the behavior of its equivalent in reality. Because of this, the client software uses an interface identical to the real robot [9]. This would therefore save time and money in carrying out tests directly on hardware without the knowledge of how the robot would actually behave in the physical world scenario. An advantage of using Gazebo software is the capability to simulate various types of position sensors such as laser scanning, sonar and Global Positioning System (GPS). Moreover, in its library it contains robots which are commonly used and enables realistic simulation of rigid body physics [9].

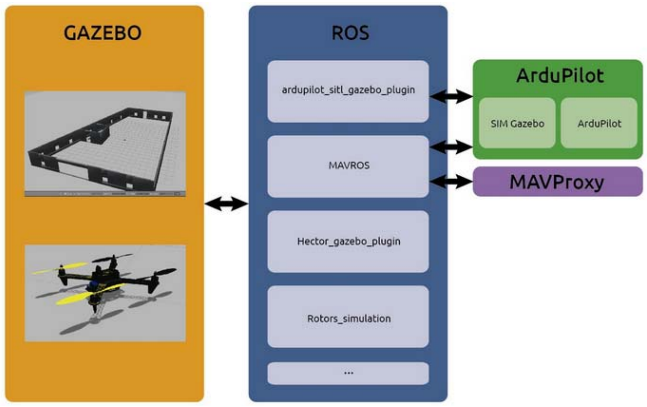

Figure 1: Link Between ROS and Gazebo

Fig. 1 shows how Gazebo is interfaced with the autopilot of the Robot with the help of ROS. In order for the simulation of the robot to take place in Gazebo, a communication link must be achieved between the robot in Gazebo and its autopilot. This is made possible through the use of ROS.

To simulate the quadcopter in Gazebo, the Erle-copter model has been utilised which is available in the Gazebo library. The Erle-copter is the first Linux based smart drone that utilize robotics frameworks for example the Robot Operating System (ROS) [10]. The Erlecopter is equipped with the Erle brain 3, which is the latest edition of the robot brain from Erle robotics. Table 1 presents the parameters of the Erlecopter.

Table 1: Erle-copter Parameters

Parameter Name	Value
Mass m	1.30 kg
Gravitation constant, g	9.80 m /s^2
Length l	0.18 m
Inertia I_x	0.03 kg.m^2
Inertia I_y	0.05 kg.m^2
Inertia I_z	0.0977 kg.m^2
Motor constant k_f	8.55x10^{-6} Nm/rad^2
Moment constant k_m	0.016 Nm/rad^2

In addition, the controller used for the simulation of the Erle-copter model in Gazebo is the autopilot of the Erle-copter is called the Ardupilot. Ardupilot is an open source Autopilot Software Suite for unmanned vehicles and offers:

- Capability of manual to fully autonomous flight modes including missions that can be programmable.
- Robot simulation on different simulators.
- Capability of faculty conditions like low battery and loss of GPS.
- Establishment of connection in real time with the Ground Control Station (GCS)
- Log mission data which would be useful in the analysis after tests.

III. Gazebo Simulation Environment Setup

Firstly, a workspace is created and initialized, and inside the workspace, the Erle-copter model and resources are downloaded. The steps listed in the Erle robotics documentation page for the configuration of the environment were followed to achieve this. Once the workspace has been created and initialized, the Erle-copter can then be launched. In order to launch the Erle-copter in Gazebo, the following commands are entered into the terminals in an Ubuntu machine:

In Terminal 1:

```
source ~/simulation/ros_catkin_ws/devel/setup.bash
cd ~/simulation/ardupilot/ArduCopter
../Tools/autotest/sim  vehicle.sh -j 4 -f Gazebo
```

In Terminal 2:

```
source ~/simulation/ros_catkin_ws/devel/setup.bash
roslaunch ardupilot_sitl_gazebo_plugin erlecopter_spawn.launch
```

This enables the Erle-copter model to be launched in Gazebo. Fig. 2 shows the Erle-copter model spawned in Gazebo simulator.

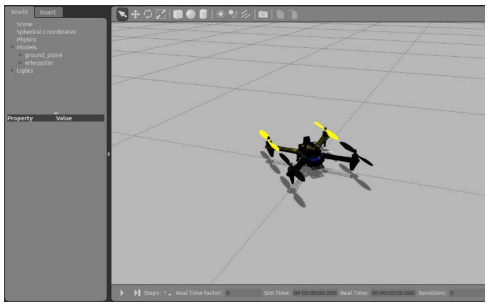

Figure 2: Erle copter spawned in Gazebo Simulator

Once the Erle copter model has been launched in Gazebo, the next step is to load its parameters by using the following command:

In Terminal 1

```
param load /home/simulation/ardupilot/Tools/Frame_params/Erle-Copter.param
```

IV. ERLE-COPTER SIMULATION IN GAZEBO

To simulate the Erle-copter, Matlab/Simulink should be interfaced with Gazebo. This is achieved through the ROS toolbox in the Simulink library. Two main types of functions, specifically "Publish and Subscribe" functions are used in Matlab/Simulink and Gazebo to communicate with each other. This communication is achieved by choosing appropriate topics and its corresponding message types of the ROS blocks in the ROS toolbox. The "publish" ROS block is used to send data from Matlab/Simulink to Gazebo while the "subscribe" ROS block is used to receive information from Gazebo. For the purpose of simulating the Erle-copter in Gazebo, position data generated in Matlab/Simulink is published to the Erlecopter model in Gazebo, while the path followed by the Erle-copter takes in Gazebo is obtained by Matlab/Simulink through the use of the "subscribe" ROS block.

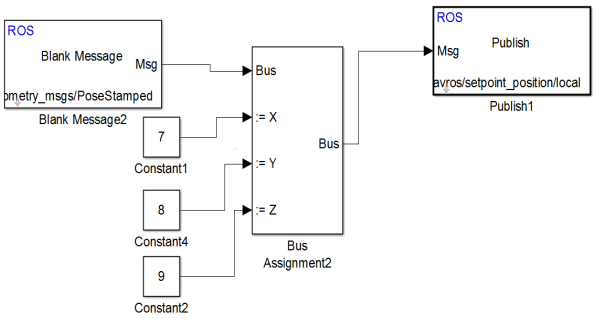

Figure 3: Publish ROS block for Position data

Fig. 3 shows how the position data is published to the Gazebo simulation environment via the publish ROS block found in the ROS toolbox in Simulink. To achieve this, the desired position data is published to a ROS topic relating to the position of the Erle-copter. Moreover, each ROS topic is accompanied by a particular message type depending on the ROS topic. Thus to publish the position data to the Gazebo simulation environment, a particular ROS topic is chosen and the appropriate message type for the ROS topic is selected. The ROS topic and the message type for the publish block used for publishing the position data is shown below:

Topic: /mavros/setpoint_position/local
Message Type: geometry_msgs/PoseStamped

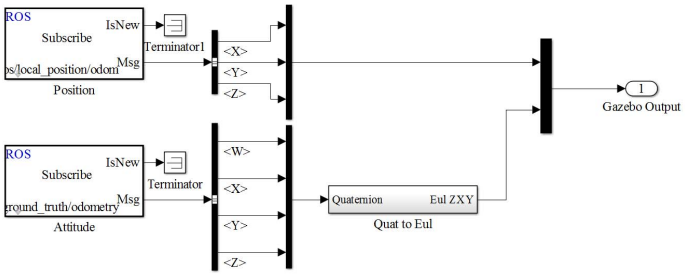

Figure 4: Subscribe ROS Blocks for Position and Attitude

Fig. 4 shows the Subscribe ROS blocks to obtain the linear and angular position data of the Erlecopter from Gazebo. The linear and angular position data is obtained directly from Gazebo by subscribing to the below ROS topic together with its message type:

Topic: /erlecopter/ground_truth/odometry
Message Type: nav_msgs/Odometry

Note: The angular position of the Erle-copter subscribed from Gazebo is in the form of quaternions. Hence, the conversion from quaternions to Euler angles is utilized to obtain the orientation of the Erle-copter in terms of Euler angles. This is shown in Fig. 4 where a quaternion to Euler conversion has been used after obtaining the orientation data of the Erle-copter from Gazebo. The function to achieve this is as follows:

$$eul=quat2eul(quat) \quad (1)$$

V. SIMULATION RESULTS

Two test cases were chosen for the simulation of the path of the Erle-copter in Gazebo. Firstly, a square path was defined and the Erle-copter was commanded to move in the square path of dimension 5 m x 5 m and at a height of 2 m. In the second test case, the Erle-copter was commanded to move in circular path of radius 3 m and a height of 5 m. The results are obtained from the simulation in Gazebo are

presented in Fig. 5. to Fig 8.

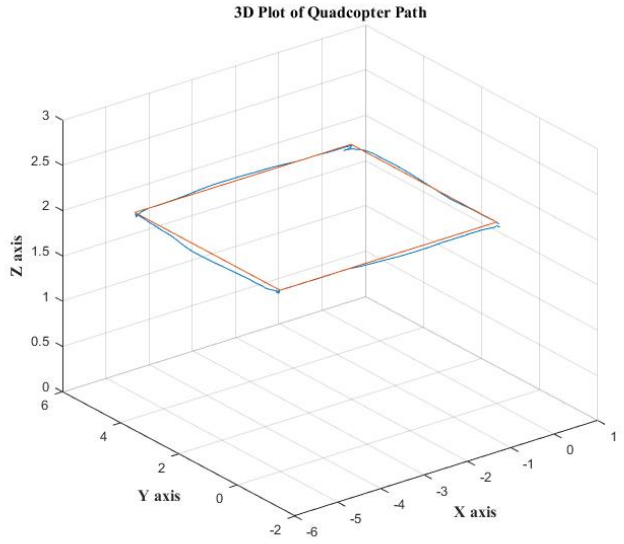

Figure 5: 3D Plot of Erle-copter moving in Square Path

Fig. 5 shows the 3D plot of the path of the Erle-copter subscribed from Gazebo for the square path. The graph in orange above represents the reference square path while the graph represented in blue above shows the actual path the Erle-copter took in Gazebo.

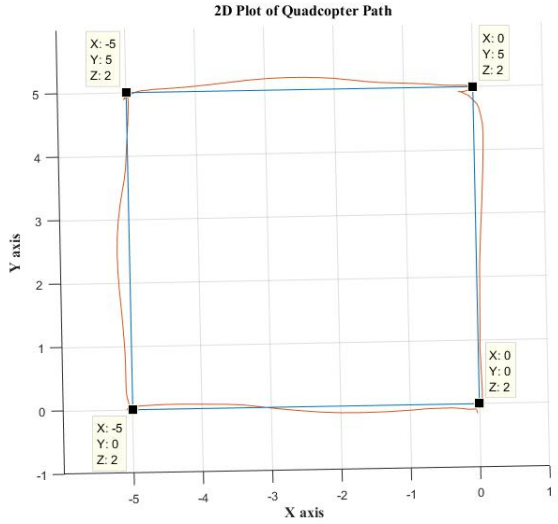

Figure 6: Graph of Erle-copter Path in 2D for Square Path

Fig. 6 shows the 2D plot of the square path the Erle-copter was commanded to move in. The dimensions of the square path were 5 m x 5 m. The graph in blue in Fig.7 represents the reference square path the Erle-copter had to follow while the graph in red represents the actual path of the Erle-copter. From the comparison of the reference and the actual path, it can be said that the Erle-copter was successfully able to move in the square path initially defined in the simulation.

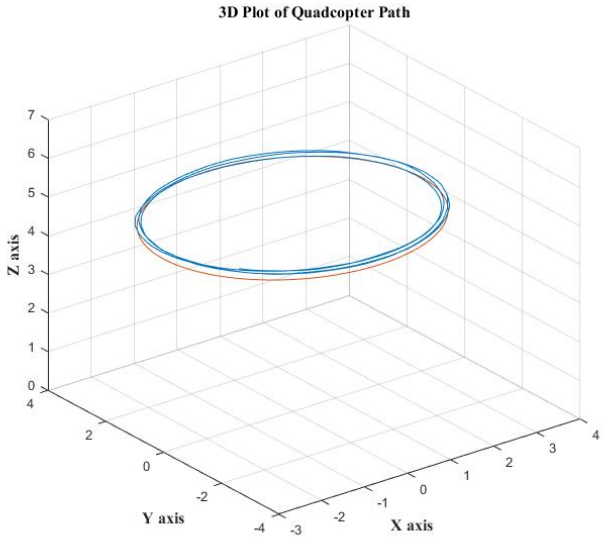

Figure 7: 3D Plot of Erle-copter moving in Circular Path

The 3D plot of the circular path which the Erle-copter was commanded to move in is represented in Fig. 7. Morover, the graph in orange in Fig. 7 represents the reference circular path the Erle-copter was commanded to move whereas the graph represented in blue in the Fig 7 is the graph of the actual path the Erlecopter moved in in Gazebo during simulation

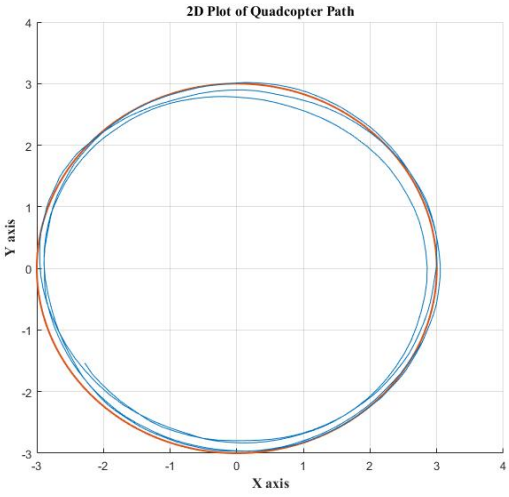

Figure 8: Graph of Erle-copter in 2D for Circular Path

In Fig 8 shows the 2D plot of the circular path for the second test case. The radius of the circular path was set to 3 m at a height of 5 m. Furthermore, in Fig 8, the graph represented in orange is the reference circular path commanded to the Erle-copter while the actual path of the Erle-copter in Gazebo is represented by the blue graph in the Fig 8. The commanded circular path of 3 m for simulation in gazebo has been achieved as the path taken by the Erle-copter during the simulation is similar to commanded path.

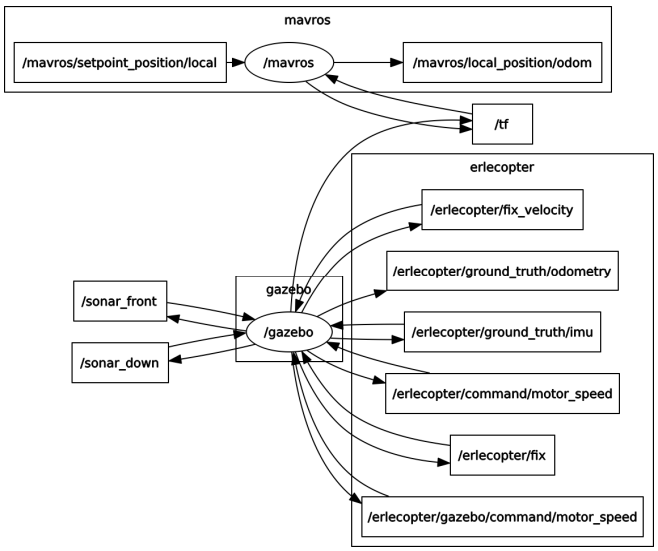

Figure 9: ROS Computation Graph

Fig. 9 shows the ROS visualization graph (rqt_graph) which provides a Graphical User Interface plugin useful in visualizing the ROS computational graph. Fig. 9 illustrates how the simulation is executed and how each component is linked to the other. In the mavros block, the two topics used for publishing and subscribing of data are shown. Moreover, the tf block in Fig. 9 is a package that allows the user to keep track of multiple coordinate frames over some time period. Moreover, it also allows users to transform points and vectors between any two coordinate frames at any desired point in time.

VI. CONCLUSION

The simulation of the Erle-copter in Gazebo via ROS was presented and the software Matlab/Simulink was utilized to carry out the simulation of the Erle-copter in Gazebo. Furthermore, ROS toolbox in Simulink was used to achieve communication between Gazebo and Matlab/Simulink. Two simulation test cases were chosen: which were firstly, commanding the Erle-copter to follow a square path of dimensions 5 m by 5 m at a height of 2 m while the second test case commanded the Erle-copter to follow a circular path of radius 3 m at a height of 5 m. The simulation results of these test cases have been presented the results can be concluded that the Erle-copter was able to follow the commanded paths in both the test cases.

REFERENCES

1. A. Lebedev, "Design and Implementation of a 6DOF Control System for an Autonomous Quadrocopter", 2013.
2. A. Gibiansky, "Quadcopter Dynamics, Simulation, and Control. 2012," URL: http://andrew.gibiansky.com/blog/physics/quadcopter-dynamics, (12.5. 2016.).
3. T. Luukkonen, "Modelling and control of quadcopter", 2011.
4. S. Ghazbi, Y. Aghli, M. Alimohammadi and A. Akbari, "Quadrotors Unmanned Aerial Vehicles: A Review", International Journal on Smart Sensing and Intelligent Systems, vol. 9, no. 1, pp. 309-333, 2016.
5. S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A Survey Of Quadrotor Unmanned Aerial Vehicles," in Proceedings of IEEE Southeastcon, pp. 1-6, Orlando, Florida, March 15-18, 2012.
6. A. Fagiolini, G. Dini, A. Bicchi, "Distributed intrusion detection for the security of industrial cooperative robotic systems" (2014) IFAC Proceedings Volumes (IFAC-PapersOnline), 19, pp. 7610-7615.
7. D. Caporale, A. Fagiolini, L. Pallottino, A. Settimi, A. Biondo, F. Amerotti, F. Massa, S. De Caro, A. Corti, A. Venturini, L. "A Planning and Control System for Self-Driving Racing Vehicles", IEEE 4th International Forum on Research and Technologies for Society and Industry, 2018,
8. S. Cousins, "Welcome to ROS Topics [ROS Topics]," in IEEE Robotics & Automation Magazine, vol. 17, no. 1, pp. 13-14, March 2010.
9. I Alonso, M. Fernandez, J. Maestre, M. Fuente "Service Robotics within the Digital Home", *Springer*, 2018.
10. "Erle-Copte Erle Robotics Docs", *Docs.erlerobotics.com*, 2018. [Online]. Available: http://docs.erlerobotics.com/erle_robots/erle_copter. [Accessed: 13- May- 2018].