A New Simulation Environment Based on Airsim, ROS, and PX4 for Quadcopter Aircrafts

Chenxiang Ma, You Zhou, Zhiqiang Li
Aircraft swarm intelligent sensing and cooperative control Key Laboratory of Sichuan Province
University of Electronic Science And Technology of China
Chengdu, Sichuan, China
e-mail: zhiqiangli@uestc.edu.cn

***Abstract*—In recent years, quadcopter aircrafts are widely used in various fields. Nowadays, the test methods for quadcopter can't meet the test requirements well. There are many shortcomings, such as it is inconducive for construction, not real enough, long development cycle, and so on. Aiming at these problems, this paper constructs a quadcopter aircrafts simulation environment based on AirSim, ROS, and PX4. Firstly, this paper introduces AirSim, ROS(Robot Operating System) and PX4 SITL (Software in the Loop). Secondly, this article describes the framework of this work and the communication way between different parts. Thirdly, two similar experiments were separately carried out in the simulation environment and the real environment. Finally, through the comparison of the test results, it proves that this simulation environment is real and efficient.**

Keywords-quadcopter aircrafts; simulation; AirSim; ROS(Robot Operating System); SITL(Software in the Loop); PX4

I. INTRODUCTION

With the continuous progress of technology, quadcopter aircrafts are widely used in security, rescue, plant protection, transportation, and other fields. However, the test methods for quadcopter can't meet the test requirements well, especially in the field of obstacle avoidance. Currently, there are two main test methods for quadcopter. The first way is to use a real quadcopter for the test. The other way is to use the current simulation environment for the test. Unfortunately, both methods have their shortcomings and can't meet the test requirements well.

Firstly, using an actual quadcopter for the test is a direct way to verify the aircraft. Undoubtedly, the effect is the most real. However, actual flights are inconvenient for the quadcopter test. The main problems are as follows. One is the safety issue, an unverified quadcopter is directly used for the test can easily injure the people. Secondly, untested aircrafts are extremely easy to crash, resulting in great economic losses. The third is an efficiency problem. A real flight needs to consider many issues, such as assembly, maintenance, battery and so on. These factors will greatly affect flight efficiency and extremely extend the development cycle.

Therefore, many researchers use simulation environments for the quadcopter test. At present, many simulation platforms can be used for tests such as Gazebo[1], jMavSim [2] and so on. They are separately shown in Fig. 1 and Fig. 2. Among them, Gazebo is a feature-rich simulation platform, which is widely used in the robots ,cars, and drone simulation. However, some limitations, like inconvenient construction, are not real enough. These two points above are extremely important for the quadcopter simulation tests. Usually, the real environment is extremely complex and has a wide variety of objects. If the simulation environment is inconducive for building large complex scenes, it will greatly extend the development time. Next, if the simulation environment scenes are not real enough, it will have much negative effect on the experiment, especially in some visual tests. As for jMavSim is an easy-to-use simulation platform, but its main function is to test the PX4 code, rather than do the specific application simulation.

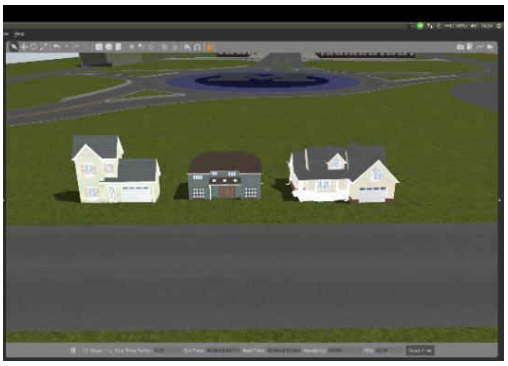

Figure 1. Gazebo simulation with Prius in MCity.

Figure 2. jMavSim simulation environment.

978-1-7281-6139-6/20/$31.00 ©2020 IEEE

Therefore, aiming at these shortcomings above, this paper proposes a new simulation environment based on the combination of AirSim, ROS, and PX4. Among them, PX4 and ROS can greatly improve the simulation efficiency, while AirSim provides us with a more realistic simulation environment and rich data interface.

II. Simulation Environment Construction

A. Introduction of AirSim,ROS and PX4 SITL

AirSim [3] is an open-source, cross-platform simulation tool based on Unreal Engine. The Unreal Engine is an excellent game engine, which makes the image extremely realistic as shown in Fig. 3. This greatly enhances the authenticity of the simulation. Compared to Fig. 1 and Fig. 2, AirSim's images are more realistic. At the same time, Unreal Engine has many ready-made scenes in application store that contain blocks, forest, snow mountains, and so on. The application store is shown is Fig. 4.These features provide great convenience for the construction of complex scenes. AirSim also provides users with rich interfaces. Users can directly retrieve sensor data through the interface and control the quadcopter. More importantly, it provides developers with rich kinds of sensors in the simulation environment, such as radar, ultrasound, binocular camera, etc.

Figure 3. AirSimNH environment.

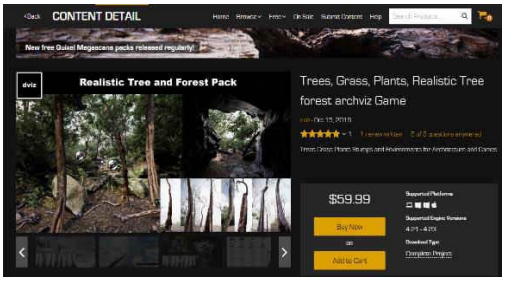

Figure 4. Unreal engine marketplace.

ROS (Robot Operating System), [4] is a set of the operating system specially developed for robots. But ROS is not a regular operating system like Windows or Mac OS. It only provides services similar to operating systems. In a nutshell, we can think ROS is a middleware that running in a Linux environment, which acts as a bridge between ROS applications. The running unit in ROS is Node. A Node is generated by a source code file, and Nodes can communicate with other Nodes by sending or receiving messages. The most important communication method in ROS is Topic, which is a publish/subscribe mode. The Publisher process may publish one or more Topics. The Subscribe process can subscribe to certain Topic and receive its content. The Master manages many nodes in the ROS program. The basic ROS architecture is shown in Fig. 5. At the same time, ROS has many advantages. Firstly, ROS supports many programming languages likes C++, Python, and so on. Secondly, ROS is compliant with the BSD protocol, which is completely free for personal and commercial applications. More importantly, ROS has plenty of open source packages. These packages include many sensor drivers, navigation tools, environment mapping tools, path planning tools, communication visualization tools, and many others.

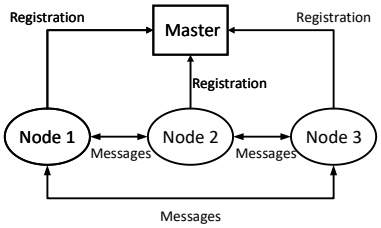

Figure 5. Basic frame for ROS.

PX4[5] is an open source flight control software, which is supported by an activate worldwide community. It provides a standard to deliver drone hardware support and software stack, allowing an ecosystem to build and maintain hardware and software in a scalable way. At the same time, it can be used for many kinds of vehicles from racing and cargo drones through to ground vehicles and submersibles.

SITL(Software in the Loop)[6] is a kind of simulation, where the flight stack runs on computer (either the same computer or another computer on the same network). At SITL operation, the sensor data comes from the flight dynamics model in the flight simulator. Currently, there are many flight stack can be used for SITL. Here we choose PX4 Autopilot for software-in-the-loop simulation. The basic SITL architecture diagram we use here is shown in Fig. 6.

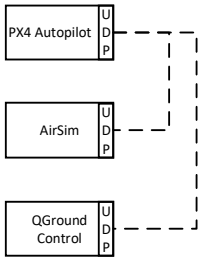

Figure 6. Basic frame for ROS.

B. Environment Construction

The overall architecture of the simulation environment is shown in the blue arrow section in Fig. 7. The green part in Fig. 7 is for real drones that will be discussed later. In the simulation, we obtain data from AirSim and publish a data Topic through a Node of ROS. Then the Process Data Node

subscribes to this data Topic, processes the data and generates the velocity and other information Topic. The MAVROS[7] subscribes to the velocity Topic and then transfers the data to the PX4 which now is in software in loop. Finally, the PX4 firmware generates the control signal to the simulation environment and controls the drone.

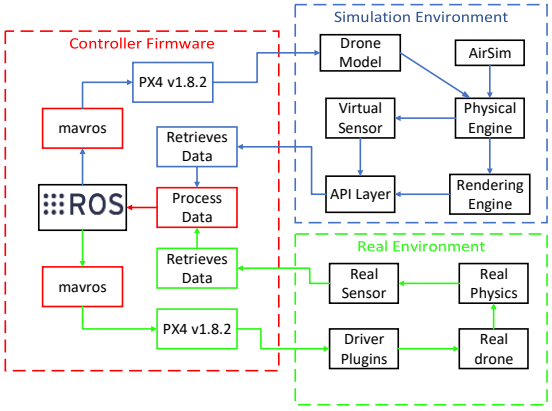

Figure 7. Process flow fiagram.

Fig. 8 shows the detail simulation environment. The upper left corner terminal is for PX4 which is in software in loop, the upper right corner terminal is for data processing terminal, MAVROS communication terminal, etc. And the lower part is the simulation scene. The PX4 v1.8.2 in the blue box in Fig.7 is used to control the quadcopter in the simulation environment.

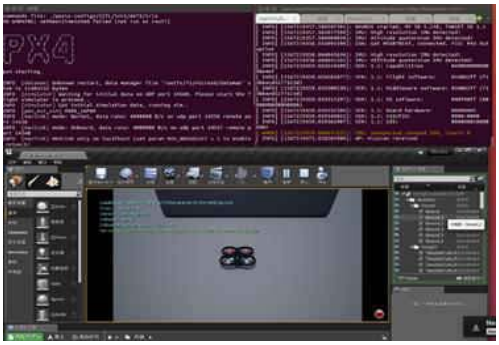

Figure 8. Simulation environment.

III. EXPERIMENT

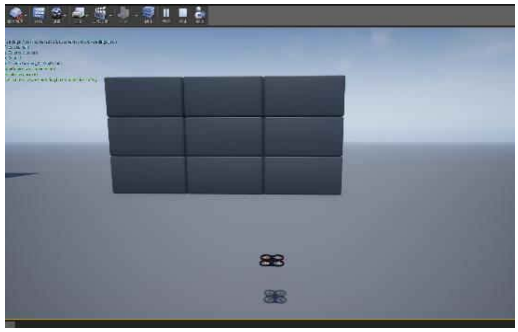

Figure 9. AirSim blocks environment.

Figure 10. Real flight test environment.

To verify the authenticity and portability of the platform, this paper conducts two similar obstacle avoidance tests in the simulation platform and the real environment. The scenes of the environment are similar. The specific scenes of the two tests are separately shown in Fig. 9, Fig. 10. And the aim of the drone is reaching the goal which is behind the obstacle.

A. Obstacle avoidance strategy

The flight environment was perceived by a depth camera in both experiments. Here we use the VFH algorithm to avoid the obstacle. It has been explained clearly in [8]. Therefore, we won't explain it here. The depth camera is set on the quadcopter. The basic idea is dividing each image captured by the depth camera into 5 columns which are shown in Fig. 11 and Fig. 12. This is equivalent to that there are five onboard range sensors in every part. Every part in Fig. 11 is 18 degree. If the front is safe, the quadcopter goes directly. Otherwise, it chooses a new direction. For simplifying the algorithm, we consider a setting with the following assumptions:

- The obstacle is in front of the drone. The obstacle distance is smaller than the detection distance in Fig. 11 but larger than the safe distance.
- The obstacle can be detected by the depth camera.
- The velocity of the drone is 1m/s.
- The quadrotor state can be estimated online.

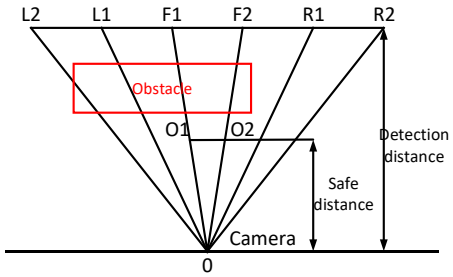

Figure 11. The view of the depth camera.

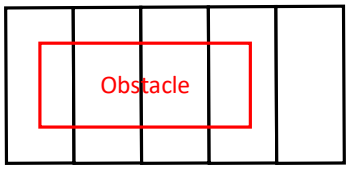

Figure 12. The method of image segmentation.

The detailed algorithm flow chart is shown in Fig. 13. Firstly, the drone judges whether it has reached the goal. If yes, the drone lands at the goal. Otherwise, it gets the sensor data and choose a safe direction. After that, it moves one step and judges whether it has reached the goal again.

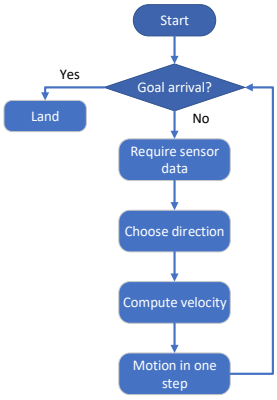

Figure 13. Obstacle avoidance algorithm flow chart.

B. Simulation Experiment

This test was carried out in the official AirSim scene Blocks which is shown in Fig. 9. The depth image type here is a special type in AirSim, which is called DepthPerspective. The pixel value of the depth image represents the actual distance from the obstacle. The angle and size of the depth image are separately set to 90°and 640*480. The purpose of the experiment is using the depth images to detect obstacles in front of the quadcopter, and controlling the drone to avoid obstacles and arrive at the goal.

As we can see in Fig. 9, a brick wall is in front of the drone. The depth image is shown in Fig. 14. From the image, we can see that the front right is safe. So the drone should flight to the right. The drone's path is shown in Fig. 15. In this figure, we can see the drone took off at point H ,flight to the right and arrived at the goal which is point 1. The result is as expected.

Figure 14. Simulation environment depth image.

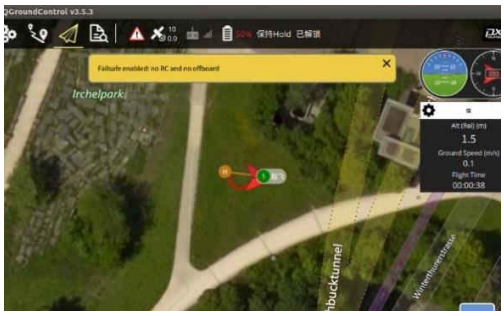

Figure 15. Drone's path in simulation environment.

C. Actual Experiment

In this experiment, we use the Amovlab's drone[9] which is shown in Fig. 16. In this test, we used two kinds of binocular cameras which are both designed by Intel. Among them, Intel RealSense T265 is used for indoor positioning. It is a stand-alone simultaneous localization and mapping(SLAM) [10] device for using in robotics, drones and more. The T265 includes two fisheye lens sensors, an IMU and an Intel Movidius. All of the V-SLAM algorithms run directly on the VPU, allowing for very low latency and extremely efficient power consumption. Intel RealSense Depth Camera D435i is used for capturing the depth image. It combines a robust depth sensing and an additional Inertial Measurement Unit (IMU). The quadcopter uses NVIDIA Jetson TX2 as an onboard computer. Jetson TX2 is a fast, powerful, efficient embedded AI computing device.

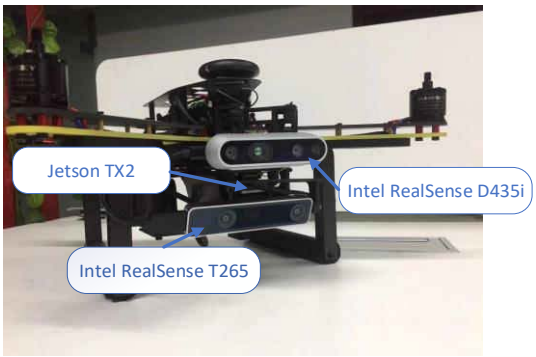

Figure 16. Amovlab P200.

To verify the simulation environment's authenticity and portability. We just replaced the image acquisition node. The specific block diagram is shown in the green part of Fig. 7. The depth image captured by D435i is shown in Fig. 17. In this figure, we can see the front left is safe.

Because the actual test was conducted indoor, there is no GPS signal indoor. We can't use QGC[11] to show the drone's path. Therefore, we recorded the drone's position data and showed it by Python. The result is shown in Fig. 18. We can see the drone fly to the left of the obstacle. The copter avoids the obstacle successfully and arrival at the goal. The result is as we expected.

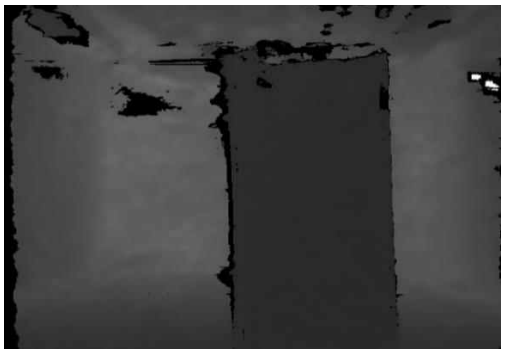

Figure 17. Actual test environment depth image.

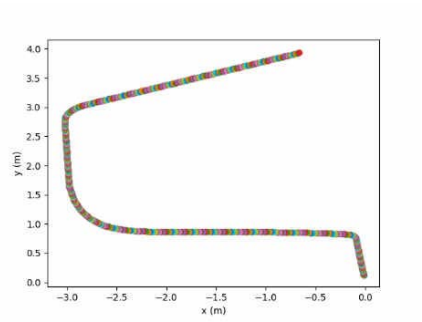

Figure 18. Drone's path in real environment.

IV. CONCLUSION

This paper introduces a SITL simulation environment based on AirSim, ROS, and PX4. The simulation environment's scene is realistic and easy to build. This environment is easier to debug than a separate AirSim simulation environment. At the same time, it has good portability and authenticity. In the end, the paper carries out two tests in the simulation environment and real scene respectively. In the case of only replacing the image acquisition Node in the ROS program, the quadcopter successfully avoided the obstacle in the real scene. This verifies the authenticity and reliability of the simulation environment. It is great for development.

ACKNOWLEDGMENT

The author would like to thanks Professor Zhiqiang Li for his valuable support during the experiments. At the same time, this paper's experiments were partially supported by Amovlab. Thanks for their help.

REFERENCES

[1] N. Koenig, and A. Howard, Design and use paradigms for Gazebo, an open-source multi-robot simulator, Institute of Electrical and Electronics Engineers Inc., 2004, pp. 2149-2154.

[2] [online] Available: https://pixhawk.org/dev/hil/jmavsim.

[3] S. Shah, D. Dey, C. Lovett, and A. Kapoor, AirSim: high-fidelity visual and physical simulation for autonomous vehicles [arXiv], arXiv, 2017-05-15 2017, pp. 14.

[4] Open Source Robotic Foundation. (2019) ROS/Introduction. [Online]. Available: http://wiki.ros.org/ROS/Introduction

[5] L. Meier, D. Honegger, and M. Pollefeys, PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms, IEEE, 2015

[6] K.D. Nguyen, and T. Nguyen, Vision-Based Software-in-the-Loop-Simulation for Unmanned Aerial Vehicles Using Gazebo and PX4 Open Source, Book Vision-Based Software-in-the-Loop-Simulation for Unmanned Aerial Vehicles Using Gazebo and PX4 Open Source, Series Vision-Based Software-in-the-Loop-Simulation for Unmanned Aerial Vehicles Using Gazebo and PX4 Open Source,ed., Editor ed., 2019, pp. 429-432

[7] Open Source Robotic Foundation. (2019) ROS/mavros. [Online]. Available: http://wiki.ros.org/mavros

[8] J. Borenstein, and Y. Koren, The vector field histogram-fast obstacle avoidance for mobile robots,IEEE Transactions on Robotics and Automation1991, pp. 278-288.

[9] [online] Available: https://amovauto.com/

[10] N. Karlsson, E. di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M.E. Munich, The vSLAM Algorithm for Robust Localization and Mapping, IEEE, 2005, pp. 24-29.

[11] [online] Available: http://qgroundcontrol.com/