From Gazebo to Gear VR: simulating robot view into VR view communicate by ROS system

Ni-Ching Lin[1] and Che-Ming Lin [1]

Abstract— **Virtual reality system is often proposed as an appropriate technology. In this paper, we will introduce our developed 3D simulation system for autonomous vehicles. As we know, simulation technologies can verify the algorithms and identify potential problems before the actual test and avoiding accident. To enhance drivers driving skill, we develop a 3D simulator Gazebo[1] which is based on robot operation system (ROS)[2] and a game engine, Unity3D. Unity3D has powerful graphics and can support high-fidelity 3D environments. On the other hand, Gazebo and ROS are well-designed simulator for testing algoritnms, peform regression testing[3]. This integrated simulator can handle real-time autonomous vehicle control algorithms.**

I. INTRODUCTION

In this paper, we are going to show our system which combine with four part, simulation, Unity3D , mobile device, and communication between them. We use Gazebo as a simulater which create user driving environment which is shown in Figure 1, such as Duckietown[4], [5] and other Gazebo building editor builds. To make user feel like they are driving in a real environment with safty. We found out that VR is a good way to do so. With all this ideal we work on Unity3D which we can build an app and install it on an Android device with VR view which is shown in Figure 2. We also work on wireless communication between Gazebo and Android device.

Simulation of robotic systems is an essential tool in teaching, research, and development.[6]. Simulator makes it possible to test algorithms rapidly, design robots, and train AI system using realistic scenarios. Gazebo offers the ability to simulate populations of robots in complex indoor and outdoor environments accurately and efciently. At your fingertips is a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. Best of all, Gazebo is free with a vibrant community.

The Robot Operating System (ROS) is a flexible framework for writing robot software. ROS is the set of software libraries and tools that help us build robot application. ROS is the most popular robot operation system for robot application. Unity3D is across-platformgame engine which is primarily used to developvideo gamesandsimulationsforcomputers,consolesandmobile devices. Unity3D is also widely utilized as user-studied platform.

We proposed a system combined ROS and Unity3D to perform real scene visualization from Gazebo simulation by

*This work was not supported by any organization

[1]Institute of Electrical and Control Engineering, National Chiao Tung University, Taiwan. `niching.ece05, gpapa830621.ece05g@g2.nctu.edu.tw`

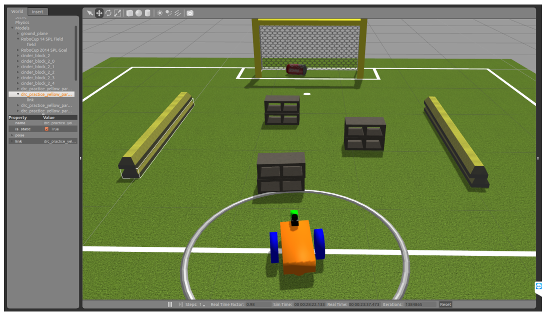

Fig. 1. The experimental environment we designed for user to control the car playing obstacle race

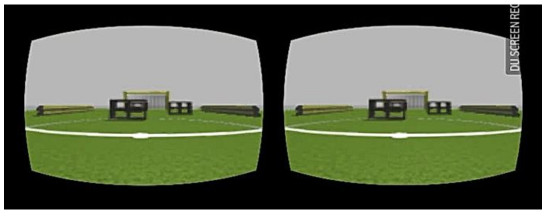

Fig. 2. Human view in VR device, the image type of Unity3D which is sent by the camera image from Gazebo

VR. Also, we expect to promote the application of our system in human-robot interaction. We design an experiment which is shown in Figure 4. User could see the view from the mobile car in Gazebo by VR device and control the mobile car moving by the joystick. We hope to get some feedbacks by the experiment so that we can improve our system.

II. MOTIVATION

Human-robot interaction[7] is important for practical robot application. A safe and realistic human-studied system is necessary to evaluate presented ideas such as self-driving car. Consider the experimental part, testing in real world will consume a lot of resources including the consumption of hardware and environmental restrictions, so we choose to build up an experimental environment in virtual world. In this case, we wish to realize the system by the combination of our own research. We choose to use self-driving car model contorl in real world simulating in Gazebo and ROS and transfer the images to Uniy3D to visualize in VR decive.

III. TECHNICAL REALIZATION

In our system viewing is an important part which is shown in Figure 3. One is Gazebo environment view and the other is VR view. For the VR part we use Uinty3D. Unity3D allows C++ control logic to access same memory for data exchange

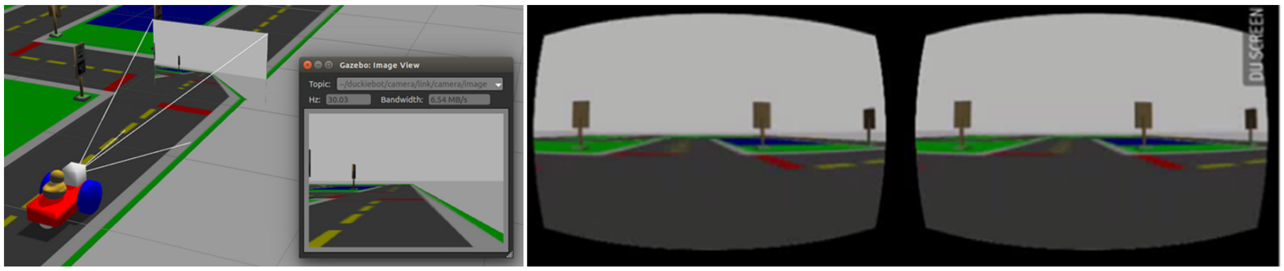

Fig. 3. The Original image and processed image. (a)The camera raw image which is sent to the ROS in Gazebo, (b)Visualization in VR device, we transform the raw image in ROS to Unity3D image type which match the Gear VR app.

was implemented. However, this interface can only be used in Windows platform, which means it cant communicate with ROS that usually runs in Ubuntu.[8] Using ROS.NET as a Unity3D plugin not only solve this issue, but also realize communication between remote server and client. In our case, Unity3D is considered as the server, and ROS part is the client. For the simulation part, we design a mobile car and build up the environment for the experiment in Gazebo. The communication between car model and joystick control in real world is done by ROS, we transfer the wheel joint_state messages in ROS.

A. *Mobile device*

Unity3D has introduced built-in support for certain VR devices. This will focus on the Oculus[9] family of VR devices, the consumer edition of the Gear VR. We are going to use Gear VR development, with Samsung Galaxy S7. With this device we are going to write an app to receive the image from gazebo and change the view into the VR image type.

B. *Gazebo*

Gazebo simulator becomes a powerful tool when its worked as a node in ROS environments. It is a relatively simple way to simulate our robot to complete ordinary tasks in daily life. Whenever we want to simulate our robot in a specific environment, we could build the world ourselves. There are lots of way to build up our own world which included importing some models we build in Sketchup and design the world structure by Gazebo building editor. In our project, we build the architecture of the experimental environment by the building editor and add some obstacle models which can see at Figure 1.

By the connection between Gazebo and ROS, we could send some specific messages between two systems. In our project, we use ROS to control our car model in Gazebo by changing the joint_state. First, we set up the joint_state of each wheel in Gazebo and use ROS message to change the joint_state of the wheel which allowed us to control the car by changing the direction of the wheels. Second, we connect the joy messages and the supported joystick by joy_node and make sure the publication of each button has a corresponding to its message. At last, we write a joy_teleop node to receive the messages from joy_node and change the joint_state while pushing a button on joystick.

C. *Robot Operating System(ROS)*

There have been a number of attempting to develop a standard robot middleware. Over the last decade ROS has become the standard middleware for the development of autonomous systems. Within ROS, robot control is modelled as a collection of asynchronous processes (known as nodes) that communicate with messages passing. Although ROS has been ported to a number of different hardware platforms and bindings exist for a number of different languages, support is primarily targeted towards Ubuntu, with software libraries targeted at C++ and Python.

D. *Unity3D*

Unity3D is a widely utilized as user-studied platform which is a powerful engine ecosystem and free software. With Unity3D we can use a series of C# projects that allow a managed .NET application to communicate traditional ROS nodes. This Unity3D.Net can coummunicate Gazebo and Unity3D by ROS messages. In our experiment Unity3D is the best platform which we can create our system. Because, Unity3D can work on Windows platform and Android device. With both platform, we are able to run our system on any Oculus device, such as, Oculus Rift Development Kit 2 and Gear VR. There is some issue why we use Unity3D below.

- It has 45% of Global Game Engine market share.
- It is preferred by 47% Game developers as primary development tool.
- It is Top Grossing 3D Mobile Game engine in all territories.
- It provides flexibility to deploy projects on multi-platforms like iOS, Windows and Android.
- Easy and quick import process of resource subsystem leading to an optimized unified Assets pipeline and supports most image, audio, video and text formats band 3D packages.
- Excellent Integrated Level editor with supporting JavaScript and C# for scripting.

IV. EXPERIMENT

This project is about making user feel like driving in the real word in a safe virtual environment which can enhance drivers skills of driving. In this paper, we test our system with some user and get feedback from them. First, we ask them

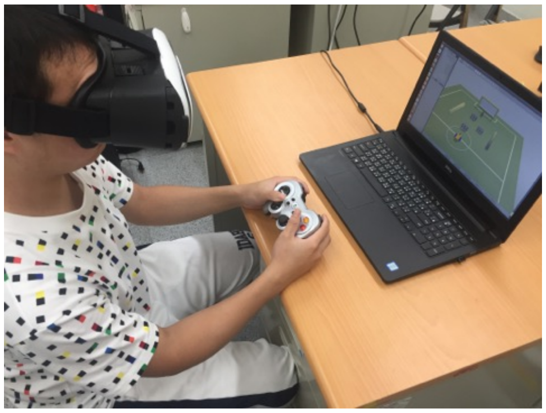

Fig. 4. User conducting a practical experiment on our system

to wear our Gear VR device which is shown in Figure 4 and teach them how the joystick works. Then we open different environment to let the user finish our goal, such as kicking the bottle into the gate or driving on the road. After that, we get some suggestion back.

They all told us that our system is significant and interesting but they would like to try this at more different environment. Also, some of the users think that the view inside the VR device seems not clear at all. We've tried to upgrade the framerate from Gazebo and it works, but Unity3D app would shut down in 1 minute. Therefore, we find the highest framerate which won't cause the app shuting down.

V. CONCLUSION

In this paper, we gave a brief introduction to our newly developed Gazebo to Unity3D autonomous vehicle simulation system. With our system people can drive in a virtual word which can protect user and improve there driving skills. We start the experiment by the operation in real world and realize the motion in virtual world, it's a great integration of the actual and virual situation. However, the visualization in VR device isn't as clear as the image we recieve in Gazebo. We hope to improve the framerate and make sure it could be succesfully sent to app. In future development, we want to add a system which is a combination between the self-driving car in real world and the VR device. We expect that user can control the car by VR device and joystick not only in the virtual reality simulation world but also in the real world.

REFERENCES

[1] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.

[2] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[3] R. Codd-Downey, P. M. Forooshani, A. Speers, H. Wang, and M. Jenkin, "From ros to unity: Leveraging robot and virtual environment middleware for immersive teleoperation," in *Information and Automation (ICIA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 932–936.

[4] J. Tani, L. Paull, M. T. Zuber, D. Rus, J. How, J. Leonard, and A. Censi, "Duckietown: an innovative way to teach autonomy," in *International Conference EduRobotics 2016*. Springer, 2016, pp. 104–121.

[5] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, *et al.*, "Duckietown: an open, inexpensive and flexible platform for autonomy education and research," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 20–25.

[6] P. Castillo-Pizarro, T. V. Arredondo, and M. Torres-Torriti, "Introductory survey to open-source mobile robot simulation software," in *Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American*. IEEE, 2010, pp. 150–155.

[7] C. Bartneck, M. Soucy, K. Fleuret, and E. B. Sandoval, "The robot enginemaking the unity 3d game engine work for hri," in *Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on*. IEEE, 2015, pp. 431–437.

[8] Y. Hu and W. Meng, "Rosunitysim: Development and experimentation of a real-time simulator for multi-unmanned aerial vehicle local planning," *Simulation*, vol. 92, no. 10, pp. 931–944, 2016.

[9] V. Oculus, "Oculus rift," *Available from WWW:¡ http://www. oculusvr. com/rift*, 2015.