2018 18th International Conference on Control, Automation and Systems (ICCAS 2018)
Oct. 17–20, 2018; YongPyong Resort, PyeongChang, GangWon, Korea

Mapping System with Virtual Reality for Mobile Robot Teleoperation

Seungwoon Kim[1], Yihan Kim[1], Jeesoo Ha[1], and Sungho Jo[2*]

[1]School of Computing, Korea Advanced Institute of Science and Technology,
Daejeon, 34141, Korea (swkim17@kaist.ac.kr, kabi@kaist.ac.kr, jsha2913@kaist.ac.kr)
[2]School of Computing, Korea Advanced Institute of Science and Technology,
Daejeon, 34141, Korea (shjo@kaist.ac.kr) * Corresponding author

Abstract: Robots using teleoperation, especially linking robots to users in real-time have been studied. For a convenient and immersive operation to a robot, it should be able to use sensors which are mounted on the robot, and effectively visualize it to increase the immersive feeling of the user. We propose a system that creates a map using a camera and a sensor installed in the robot, recognizes the surrounding environment. It visualizes PointCloud data in real time on the user's virtual reality device. To demonstrate our proposed system, we used TurtleBot and Gear VR as robot and user device, respectively. Also, a laptop connected with TurtleBot processes the PointCloud data obtained from Depth Camera to Octomap, sends it to the VR device, and the VR device visualizes received data in real time. We experimented in a real environment by creating a VR-based teleoperation system that remotely manipulates TurtleBot and visualizes the data. The transmitted PointCloud data were reflected in the VR device in real time so that the user can remotely recognize the situation of the site.

Keywords: Teleoperation, Virtual Reality (VR), Remote Control, Point Cloud

1. INTRODUCTION

Robots using teleoperation have been used for disaster recovery and exploration in stand-off environments. In particular, there are several studies linking robots to users in situations or environments where people cannot access[1][2].

Tang et al.[3] proposed a system in which the user perceives the environment of the robot through the screen and manipulates the construction robot remotely. The virtual reality device of the system displays the front view of the robot on the screen, which limits the user's working immersion. Benaoumeur et al.[4] introduced robot control using a virtual reality device. Since position sensor and camera are outside the robot, its application is limited to indoor environments. Both approaches use the external fixed camera to localize the position and the surrounding environment of the robot.

To solve the usability problem of the robot, it is conceivable to use only the sensor mounted on the robot. This can increase immersiveness by visualizing the head-mounted *virtual reality device*(hereinafter referred to as *VR device*) while simultaneously exploring the surrounding environment in real time. Even if the camera does not see the direction immediately, it will allow the user to recognize the area previously scanned. In addition, the environment that robot is located can be temporarily stored, therefore it can be visualized in the user's device.

For this reason, we propose a system that creates a map using cameras and sensors installed in the robot, recognizes the surrounding environment and visualizes it on a VR device in real time. Especially, by visualizing Octomap using PointCloud data, the user can quickly recognize the surrounding environment. The system consists of two parts: user side, including the user and VR device, robot side consisting of the robot, PC, and camera sensor. The robot handles the surrounding environment and information collected through the sensors and transmits them to the user side. The VR device on the user side receives and visualizes the data on the robot side. Also, the user watches the VR screen and transmits an operation command to the robot. Then, we conducted the actual test through the experiment and the performance evaluation.

We introduce methods and tools used to implement the functionality in Section 2. In Section 3, we introduce experiments based on the above system and analyze results.

2. METHODOLOGY

To demonstrate that real-time environmental visualization is possible, we used a library that links ROS and Unity[5]. The system largely consists of VR visualization device, a remotely controllable robot, and communication between two devices. The hardware and software architecture is described in Figure 1.

We used ROS[6] as the programming language for controlling the TurtleBot on a laptop. It scans the surrounding environments through the Depth camera connected to the laptop and transmits the obtained PointCloud data to the user through the laptop. The PointCloud data consists of X, Y, and Z coordinates, and each point can be combined to construct the surrounding environment. Then we used ROSBridge to send PointCloud messages from the sensor to Unity. ROSBridge provides a simple, socket-based programmatic access to robot in-

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2017-0-00432).

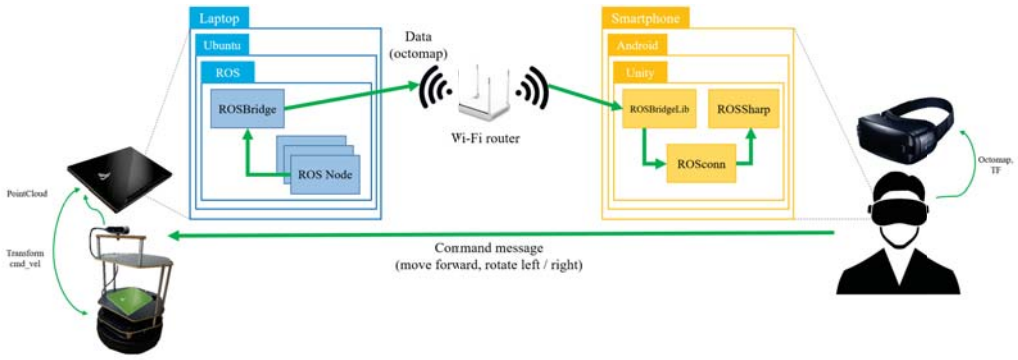

Fig. 1 Hardware and software architecture of VR mapping system for mobile teleopertion

terfaces and it facilitates the use of web technologies such as Javascript[7]. Through ROSBridge, the laptop opens a socket connection to communicate with the user's VR device.

Unity is a game engine that supports various platforms such as mobile, VR as well as PC. Siemens' ROS Sharp[1] was used on Unity for communication with the robot. ROS Sharp is an open source software library that allows ROS to communicate with .NET applications, especially Unity. TurtleBot's remote control and real-time camera information reception are implemented in the library. We used open sockets to receive PointCloud messages over the wirless network and visualized them in Unity so that they can be applied to VR devices.

3. EXPERIMENTS AND RESULTS

3.1 Experimental setup

For the experiments, we used Gear VR as the user's device for environmental visualization. Gear VR is a Virtual Reality headset that can display VR screen by attaching Samsung's Galaxy series mobile phone. Galaxy S8 Plus was used as a mobile phone equipped with Gear VR. We used the Unity development platform to build applications that work with Gear VR.

We used TurtleBot 2, a mobile robot made by YUJIN ROBOT, for stable teleoperation of the remote robot. For remote control of TurtleBot, we connected TurtleBot and a laptop (Ubuntu 16.04, Intel i7-7700HQ, 16GB RAM) with an operating system in the middle. The laptop connected to TurtleBot acts as a communication intermediary to transmit the data obtained from the connected sensor camera back to Unity while parsing the request received from the user's Unity program to determine the TurtleBot's operation.

Experiments were conducted in two cases: a classroom and corridor. In the classroom, the enclosing loop was tested in a narrow space by creating a square obstacle in the middle. The actual size of the classroom is 6.55m in width and 6.75m in length. In the corridor, on the other hand, a long path (unclosed) was moved to search for the

[1]https://github.com/siemens/ros-sharp

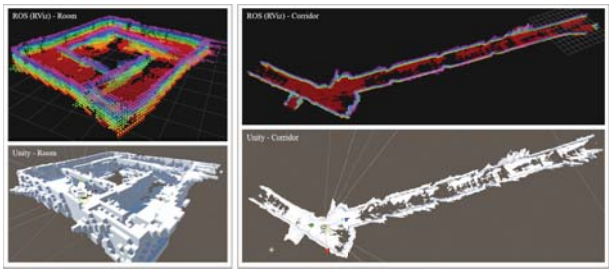

Fig. 2 Comparison between the results displayed on the robot side (top) and the results displayed on the VR (bottom) device of two experiments.

shape of the surrounding area.

3.2 Experiments

We experimented with teleoperation remotely by visualizing the PointCloud data that TurtleBot had obtained from its exploration to VR.

TurtleBot collects the PointCloud data while exploring the classroom and corridor in real time, converts it to Octomap, and transmits it through the wireless communication network. The user views the visualized map on the VR device and controls the TurtleBot to avoid obstacles to go in the right direction.

3.3 Results

As results of the experiments, Figure 2 is comparing PointCloud map generated by TurtleBot with Octomap visualized by VR device. As shown in Figure 2, we were able to visualize Octomap on the VR device using PointCloud data, and we found that it is similar to the actual map.

Also, the length between the actual octomap and the four corners of the map was measured to be 6.48 m, 6.66 m, 6.53 m, and 6.76 m, respectively. It can be seen that the difference between the measured value and actual value is less than 0.1m. A comparison of the measured and actual maps is shown in Figure 3.

Figure 4 compares the real image from the Turtlebot with the grid view through the VR device. In addition, we found that there was no difference between the actual

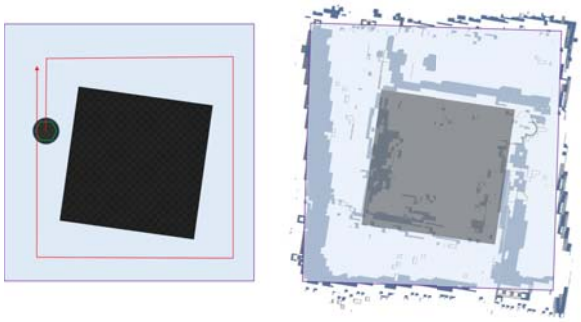

Fig. 3 Configuration of the first experiment (left), and comparison of expected with measured map (right). Central obstacle was marked with a black square.

Fig. 4 Comparison of real image (top) and transformed grid view (bottom).

position of the Turtlebot and the position of generated octomap. Therefore, using the PointCloud data generated by TurtleBot, remote users can perform teleoperation while recognizing the situation of the site through VR device.

4. CONCLUSIONS

We have created an system to visualize a map on the VR device using PointCloud data collected from Turtlebot. The transmitted PointCloud data are reflected in the VR device in real time so that the user can remotely recognize the situation of the site and teleoperate the Turtlebot. Through experiments, we found that there was no significant difference between the real and virtual environments.

We used only the depth camera sensor in this study. However, we can expand the scope of our study by installing other sensors based on our proposed system. For example, if a fire scene needs to be explored, data can be mapped to a VR device with a temperature sensor to aid in the determination. You can also get on-site communication information to determine if TurtleBot's communication can be continued or not. In addition, it can respond promptly to the field by sharing information with TurtleBot remotely by utilizing air pressure sensor and power information.

In the future, we will study human-robot interaction by attaching various sensors to our proposed system. This will allow the user to see the information received from various sensors in an immersive manner, thereby activating interaction between the user and robot.

REFERENCES

[1] Casper, J., and Murphy, R. R. (2003). "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center." *In IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(3), 367-385.

[2] Murphy, R. R. (2004). "Human-robot interaction in rescue robotics." *In IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2), 138-153.

[3] Tang, X., Zhao, D., Yamada, H., and Ni, T. (2009, August). "Haptic interaction in tele-operation control system of construction robot based on virtual reality." *In Mechatronics and Automation*, 2009. ICMA 2009. International Conference on (pp. 78-83). IEEE.

[4] Benaoumeur, I., Zoubir, A. F., and Reda, H. E. A. (2015). "Remote control of mobile robot using the virtual reality." *International Journal of Electrical and Computer Engineering (IJECE)*, 5(5), 1062-1074.

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

[6] Quigley, M.,Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. Y. (2009, May). "ROS: an open-source Robot Operating System." *In ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

[7] Crick, C., Jay, G., Osentoski, S., Pitzer, B., and Jenkins, O. C. (2017). "Rosbridge: Ros for non-ros users." *In Robotics Research* (pp. 493-504). Springer, Cham.