Submitted on May 04, 2018 to the CIS 693 Autonomous Intelligent Robotics Course
Volume 1??,Number 1??,

Web-based Mobile Robot Control and Monitoring

Danye Luo Henglong Ma and Nuo Zhou

Abstract In this paper we describe our project of creating interactive web pages to remotely control multiple TurtleBots using the web browser instead of command line. We created web pages which allow the user to use keyboard, button click, and speech commands to remotely operate a TurtleBot to move forward, move backward, turn left, and turn right. Our web pages can display the camera view of the TurtleBot which is being tele-operated in the web browser. Therefore, users can see what the TurtleBot sees in the real time world. The web page also provides navigation using a map where the user can select a goal position and the TurtleBot can navigate itself to reach the destination autonomously. Also, our web page enables a "One Click Go Home" feature that the user can click one button to have the TurtleBot auto docking. As a result, we can provide the end user with opportunities to control TurtleBots remotely and virtually experience the environment of the Autonomous Intelligent Robotics Lab in Cleveland State University. One application of our project could be providing virtual demonstration of robots and our lab for local middle school and high school students. Compared with traditional lab tour, it might provide young students with a more intuitive and interactive way to learn about the TurtleBots.

1 Introduction

With the development in the area of robotics, there are increasing number of robots that can perform more and more complicated tasks. Correspondingly, the interaction between human and robots will increase, which calls for more intuitive methods of or systems for interacting with robots (Allard, 2003).

Having remote access to robots through other user friendly interface such as a web page or a graphical user interface would enable simpler interaction and control of robots for non-expert users (Allard, 2003; Gomez et al., 2016). Also, teleoperation of robots make it possible for using the robots to perform tasks in conditions that are too dangerous for humans such as transporting hazardous materials, operating search and rescue, and military and law enforcement applications (Allard, 2003).

Keywords: teleoperation, telepresence, TurtleBot, ROS

Furthermore, with the growing trend in incorporating intelligence technologies in new buildings, applications of robots teleoperation and telepresence significantly improve efficiency and reduce manpower in situations such as performing maintenance and surveillance routine tasks with minimum human intervention (Lopez et al., 2013). The technology of robots teleoperation also can contribute to promote the well being of people such as conducting in-home environment screening for preventing fall risks for seniors (Du et al., 2014).Therefore, robots teleoperation and telepresence applications are beneficial in many situations.

2 Related Work

In the research area of robots teleoperation and telepresence, there is increasing number of applications that have been developed and tested. Gomez et al. (2016) developed a graphical user interface for remotely operating robots for teaching purpose. The robots that they used in their project run on the Robot Operating System, namely, ROS (Quigley et al., 2009). They created a qt-based graphical user interface to provide two modes for users: Teleoperation mode and Follow me mode. In teleoperation mode, users can set speed and directions for robot's movement through slider and button on the GUI. In the follow me mode, robot will detect movements of humans in its current environment and start following a person with a fixed distance. In order to simplify the user interaction, graphical widget are associated with ROS nodes for controlling the robot. Users also can dynamically change the IP and hostname of the connection through the GUI. The result of their project have demonstrated that the GUI has helped students effectively learn to control the robot and deepen their understanding about how the robots work.

The Department of Computer Science of The University of Texas at Austin has developed a telepresence system for remotely operated building tours called Virtour (Lankenau, 2016), which runs on the BWI (Building Wide Intelligence) segbot robot platform (Khandelwal et al., 2017). It provides the users with remote access to the robots in computer science building through a website directly. It allows multiple users to login in the system at the same time, but only one user can be the tour leader to control the robot while other users can spectate the tour. The end-users can see real-time video feedback instantly on the website while teleoperating the robot. The robots can also take spoken messages delivered by the user and perform scavenger hunt interaction tasks. The web-based interface serves as a web client to send requests to the robot through the ROS bridge node. The server runs on the physical robot receives the requests and delegate services providers to perform the requested operations by the user. They created smallDNS to manage the dynamically changing IP addresses, and robots will constantly update its IP address to the server. The Virtour system is devoted to guarantee security and safety from both client side and server side to prevent unauthorized users and incorrect or invalid actions.

Du et al. (2014) developed a robotic system of in-home environment screening to assess fall risks for seniors. The robotic system is implemented on TurtleBot platform. It has a patient side who has the robot running inside of his or her home, and a provider side who remotely operates the robot to assess the patient's home environment through the visual feedback streamed from the patient side. ROS manages different nodes running on the robot in a form of a local network for robot operations.

It communicates with the provider side through a web interface and a standalone applications. Between the provider side and the patient side, there is an intermediate layer called service management layer for hosting the web page and conducting access control to ensure secured access from authorized users. The robot will make both 2D and 3D map with the cost in order to avoid obstacles when generates a trajectory. With the assistance of the camera and microphone, patients will be able to interact with the doctor or operator for taking care of patients' health concerns besides the assessment of the home environment.

These existing work of robot teleoperation and telepresence all provide intuitive user interface for easy use to make it more accessible for people who don't have experience with robots. They all can provide end-users with real time video streamed from the camera on the robot for more realistic and engaging user experience. At the same time, it also increases the safety of teleoperation and telepresence, since it will be dangerous to instruct the robot to move without knowing robot's current environment. However, among these previous work, robots have relatively limited autonomy since their operations were mostly controlled by the remote operator. More work can be added to increase robots' autonomy while balancing the safety concerns during teleoperation and telepresence. Also, it is worthwhile to explore more ways that allow end-users to interact with the environment where the robot is currently at. Du et al. (2014) suggested to incorporate object recognition features, and therefore robots can detect environment autonomously to help end-users to collect more information from the robot's environment.

3 Approach

Our project is implemented on the TurtleBot platform. The packages for web connection that we use in our project are developed and maintained by the ROS web tools community (Toris et al., 2015). We use HTML, CSS, and JavaScript to create web pages to serve as the web client. According to Joseph (2017) and Lankenau (2016), the web client includes ROS clients which can send the user's commands as JSON command. We installed and set up ROS web packages that include *rosbridge_server* to receive JSON commands which can be converted to ROS commands to publish and subscribe ROS topics and request ROS services for operations of the TurtleBot. The feature of controlling the TurtleBot through speech commands is enabled by using the CMU Sphinx speech recognition system (Lamere et al., 2003), which has a simple wrapper for ROS users called pocketsphinx. We installed and set up *web_video_server* for video streaming to display the camera view of the TurtleBot in the browser. We have tested in both Turtlebot Gazebo simulator and the TurtleBots (Koenig & Howard, 2004), and most of our testing is conducted on the TurtleBots. We utilized the 2D navigation widget to implement the TurtleBot's autonomous navigation using the map of the Autonomous Intelligent Robotics Lab in Cleveland State University. We adjusted our expectations from previous proposal. We replaced the feature of TurtleBot asking humans for help to open the door and the feature that the TurtleBot will display questions that the end-user types in on the website and requests answers from humans in its environment, with the feature of TurtleBot auto docking, which we think would be more helpful to enable the user to control the TurtleBot to autonomously charge itself from anywhere.

4 System Design

In this system, people and robots are terminals across two sides of the system, Figure1. How to build an effective connection between them is the core work this system should do. Here, we use the ROS built-in distributed architecture and *rosbridge_suite* package, which is provided by ROS as well, to set up the system fundamental architecture, Figure2. ROS distributed architecture provides the necessary framework for remote multiple robots control while *rosbridge_suite* creates a communication channel between a web page and ROS.

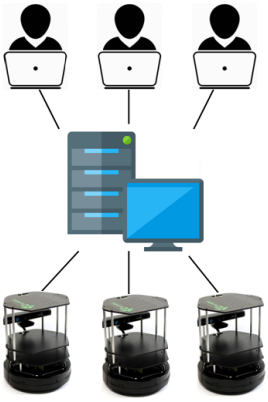


Figure 1 System Design

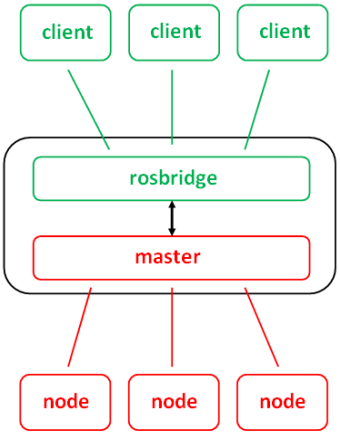


Figure 2 System Architecture

4.1 ROS Distributed Environment

Essentially, ROS is a type of middleware. It works as an information center. The Master is the administrator in charge of gathering data, messages, service requests and other information sent by numerous nodes, and switches them to the requested nodes. The identification between the Master and its nodes is through Master IP. Once nodes get Master's IP and report their own ROS IP to the Master, Master doesn't care where its nodes are physically set but still can communicate with them.This is the unique characteristic of ROS for distributed environment. In this system, we use ROS distributed attribute to setup the system architecture for multi-robot remote control. A server is set as the home of ROS Master. Each remote robot is treated as a node to register to the Master. For instance, the Master IP is 10.219.10.10 and one robot node IP is 10.219.10.11. Export IP information in the Master and node *bashrc* files individually to establish their connection.

For Master *bashrc* file

export ROS_MASTER_URI=10.219.10.10
export ROS_IP=10.219.10.10

For node *bashrc* file

export ROS_MASTER_URI=10.219.10.10
export ROS_IP=10.219.10.11

When the Master and nodes are launched, they can automatically build connection by Master IP with each other so that it implements Master remotely controls nodes' behavior.

4.2 rosbridge_suite Package

As an encapsulated system, information flows and exchange by Master within ROS. However, if it needs to interact with the surround environment outside ROS, there must be an interface for this kind of communication. ROS provides a package named *rosbridge_suite* especially for interaction between ROS and web page JavaScript requests.This is the only interface used in this system to interact with the web page requests and ROS.

Figure3 gives an example for the connection establishment between ROS and a web page through *rosbridge_suite*. 10.219.10.10 is the Master IP, which the web page tries to connect. 9090 is the defaulted port of *rosbridge_suite* for ROS-web page communication.

```
var ros = new ROSLIB.Ros({
    url : 'ws://10.219.10.10:9090'
});
```

Figure 3 ROS and Web Page Connection

Figure4 gives an example of definition for ROS-webpage interface. *robot_3_home* is an interface variable which is bound to a ROS topic *gohome*. Its message type is *std_msgs/String*.

Figure5 gives an example of the communication method between ROS and a web page. Function *home_3* is a button triggered event function. After triggered by a user click, function *home_3* is invoked and variable *robot_3_home* publishes a

```
var robot_3_home = new ROSLIB.Topic({
    ros : ros,
    name : 'gohome',
    messageType : 'std_msgs/String'
});
```

Figure 4 rosbridge Interface Definition

std_msgs/String message "*home*" to ROS to implement the communication between ROS and a web request.

```
function home_3() {
    var cmd = new ROSLIB.Message({
        data : 'home'
    });
    robot_3_home.publish(cmd);
}
```

Figure 5 rosbridge Communication

5 Features

Our final project achieved features including controlling multiple TurtleBots with buttons, keyboard, and voice control, adjusting Turtlebot's moving speed, receiving video images from the TurtleBot's view, locating TurtleBot's position on the map of the lab room as well as autonomous navigation, and TurtleBot auto docking (one click go home).

5.1 Controlling multiple TurtleBots simultaneously

We use a PC as ROS master to connect to two TurtleBots through ssh and TurtleBots' IP addresses. Rosbridge_server can be set up on both the PC side or the TurtleBot side. We run rosbridge_server on the PC to facilitate the communication between the web page and TurtleBots. The rosbridge_server is the intermediate layer between the web page and the TurtleBots' ROS system. The rosbridge_server receives the JSON commands that sent from the web page at the user's end, and convert JSON commands to ROS topics or services and send to ROS system.

For preparation work, we first need to include standard CSS style files and import all of the required JavaScript files of different modules. In our HTML file, we created an object of ROSLIB.Ros to communicate with rosbridge_server, and specify the IP address of the TurtleBot that is running ROS. We created ROSLIB.Topic and ROSLIB.Message to define ROS topic and messages in the HTML file. We create JavaScript functions including 'forward', 'back', 'left', 'right', and 'stop' that correspond to the buttons on the web page. In each of those function, we send ROS message to a specific ROS topic, which is converted by rosbridge_server and is received by the TurtleBot. On the TurbleBot, we have setup node which is subscribing to the corresponding ROS topic. After receiving ROS message, the callback functions on the TurtleBot are invoked, and the Turtlebot

executes movement that requested from the user end (Source code is available at: https://github.com/nzhoucsu/remote_control_multi-turtlebot).

In addition to the user's button click, the web page can also accept the user's keyboard input. The keyboard teleoperation is achieved by a JavaScript module keyboardteleopjs that can handle keyboard teleoperation. In the HTML file, we need to create a handler of this JavaScript module class with the previously created ROS node object and the ROS topic about teleoperation.

For adjusting the moving speed of the TurtleBot, we have an UI slider for the user to specify the desired speed. Each time when the user changes the value of the speed on the slider, it will change both the text label on the web page and the scale variable of the keyboard teleoperation object, which will adjust the moving speed of the TurtleBot.

Also, we tested to control the TurtleBot via the web page using speech command, while this feature needs more test to guarantee its robustness.We utilized the CMU speech recognition tool to convert human voice to text. We created a commands dictionary for controlling the TurtleBot to perform desired movement including moving forward and back, turning left and right, and stop. The speech commands given by the user will be converted to text information, and also will be published as ROS message to voice recognition related ROS topic. The TurtleBot that is subscribing to this topic reacts to the user' speech input through callback function, which will make the TurtleBot physically move by generating local control signals to accomplish navigation tasks.

Figure 6 Remote Multi-Robot Control

5.2 Obtaining video stream of the TurtleBot's view

Our web page can display the video images from the TurtleBot's view, which helps us to see the dynamic video view through the TurtleBot's camera. On the TurtleBot's side, we launch file to setup the TurtleBot's camera. On the PC workstation, we run web_video_server to obtain the video images captured by the TurtleBot. The web_video_server can convert ROS image transport topic to a video stream, and make it accessible via HTTP. In our web page, we upload an image and specify its source to point to the TurtleBot's IP address, port and its corresponding ROS topic about image view output. It can display the images obtained from the TurbleBot as a video stream on the web page (Source code is available at: https://github.com/danyeluo1114/Control-TurtleBot-From-Web).

```
<img src="http://10.219.67.51:8080/stream?topic=/image_view/output">
```

Figure 7 Display Image view as video stream on the web page

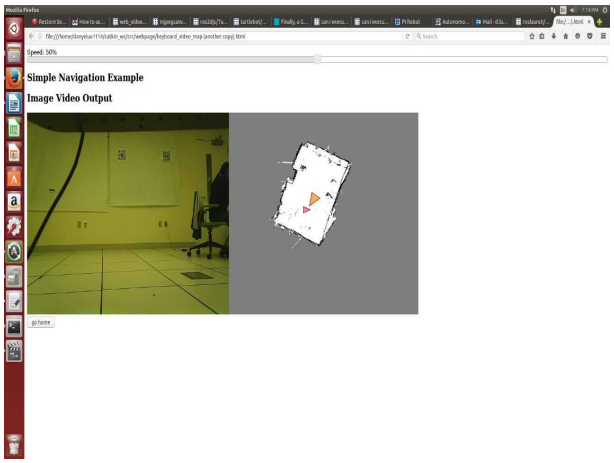


Figure 8 Web page has video stream and map

5.3 Displaying the TurtleBot's location on the map

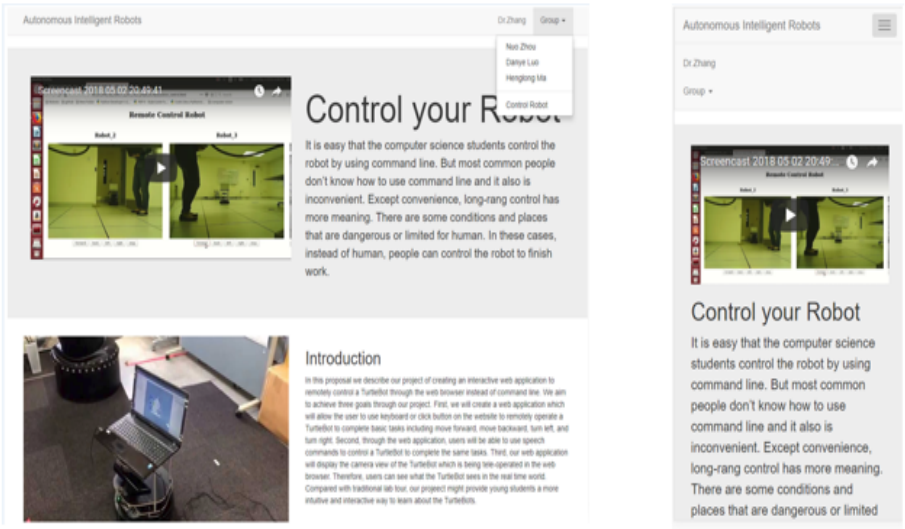
On our web page, right next to the video stream of TurtleBot's camera, we can display the map of the lab, therefore the user can see the static environment setting of the lab. It has an orange triangle on the map to indicate the current location of the TurtleBot. When the user double clicks a random point on the map, the destination point that selected by the user will be indicated as a pink triangle. Then the TurtleBot will reach the destination via autonomous navigation. It is achieved by using the 2D navigation widget: nav2djs. Nav2djs is a tool that can display map and allow interaction with a robot's autonomous navigation capabilities via the map. The map has been loaded when we bring up the TurtleBot. The main widget renders an image of the TurtleBot's internal map and stream it as a nav_msgs/OccupancyGrid message to display on HTML canvas element. Along with the application of nav2djs, we also run the robot_pose_publisher node, which is a simple node to publish the TurtleBot's dynamic position relative to the map. Figure8 demonstrates how the feature 5.2 and feature 5.3 looks in the web browser.

5.4 One click go home

When the user clicks the 'go home' button on the web page, the TurtleBot will navigate itself autonomously to a 2 * 5 rectangle area in front of the docking station, from there it can autonomously dock and go charge itself. We study the launch file in the kobuki_auto_docking package, and incorporate needed node and launch file in our code. When the user clicks the 'go home' button, a ROS message has been sent to a topic defined by us through the rosbridge_server, and is received by the TurtleBot that is listening on the specified topic. The TurtleBot's reacts to this message through

its callback function. It will autonomously navigate itself to the goal position which we defined near to the docking station, and the TurtleBot will detect the position of the station and complete autonomous docking.

5.5 Responsive website



(a) Desktop view (b) Mobile view

Figure 9 Home page of responsive website

We developed the responsive website which can adjust to the screens of desktops, laptops, and the mobile phones. On the home page, it provides the user with three different choices for control:Button, voice, and view video. After clicking on a choice, it will take the user to the subpage of the selected choice (Source code is available at: https://github.com/PatrickMaH/rosweb).

The first line displays the name of class and my instructor. And if the user clicks 'my group', it will list the group members. The second part shows the background and the test videos of this project. The user can watch the videos on 'my page'. The third part shows the introduction and the robots that we need to use. The last part demonstrates the ways of controlling robot that we achieved. When the user clicks the button, they can start to control the robot at the new subpage.

6 Summary

In this project, we achieved to use web pages on the remote PC to control single or multiple TurtleBots through buttons, keyboard, and voice command. We also achieved to obtain the video stream from the images output of the TurtleBot's camera, and display them on the web page. We also can load the map of the lab onto the web page, and are able to see the TurtleBot's current location relative to the map and

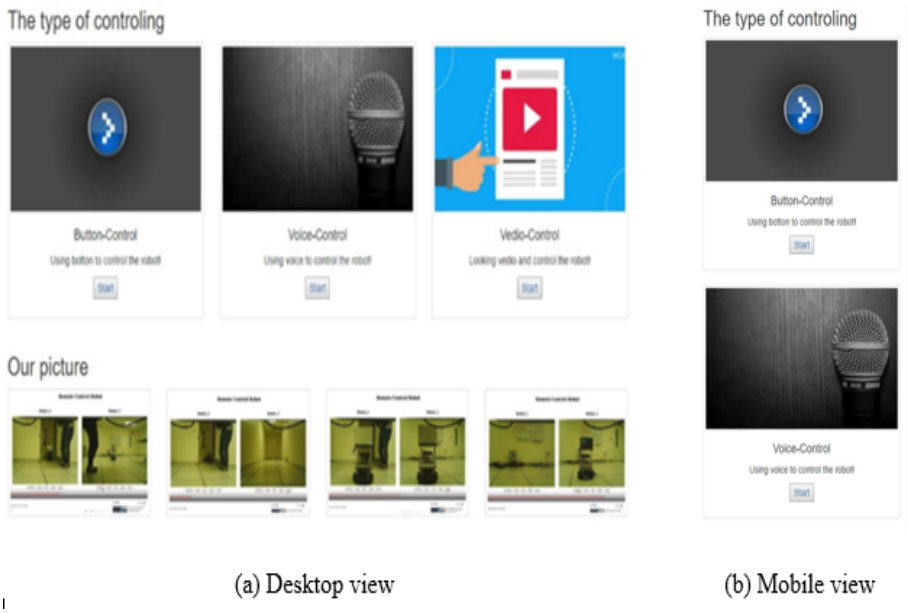


Figure 10 Control and monitoring options

make the TurtleBot do autonomous navigation via double click positions on the map. Our project also have the feature of sending the TurtleBot to complete auto docking via button click on the web page from the user end.

During the course of completing this project, we have deepened our understanding about TurtleBot networks configuration about connecting the remote PC workstation and TurtleBots. We also have learned about tools for facilitating communications and information exchange over the web including rosbridge_server, web_video_server, and so on, and how they help the web browser communicate with the ROS system. In the details about function implementation in the HTML file, we utilize Publisher and Subscriber to send and receive ROS message in ROS topic, and invoke callback function to respond to the request from the web page. We also incorporate knowledges learned from previous class assignment including autonomous navigation to a goal location and voice recognition. We also learned that teamwork is significant for the completion of this project. When we exchanged approaches to achieve a specific goal with each other, it broadens our horizon on diverse solutions and deepen our understanding of certain concepts and ways of implementation.

7 Future Work

In this project, we view and test our web pages through local computer. We would like to expend our work to put our web pages and website on a server, and make it available to the public in order to allow the public user to remotely control the TurtleBots. While at the same time, we need to take the safety and privacy into consideration, such as how to avoid the conflict that multiple users try to control one TurtleBot, and how to balance the privacy of teaching and other activities in the lab

room or even in the department building when the public user can remotely connect to the TurtleBot and watch the video stream of its view. Also, another area that would be interesting to expend is to increase the interaction between the remote user and TurtleBot's current environment.

8 Work Contribution

Table 1 Work Contribution

Function	People
Architecture Design	Nuo Zhou
ROS Distributed Environment Configuration	Nuo Zhou
Remote Control	Nuo Zhou
Multiple Robot Control	Nuo Zhou
Video Transmission	Danye Luo
Map Navigation	Danye Luo
Speed Adjustment	Danye Luo
Go Home	Danye Luo
Auto Charge	Danye Luo Nuo Zhou
Voice Control	Henglong Ma
Webpage Development	Henglong Ma
Moble App Development	Henglong Ma
System Testing	Henglong Ma

References

[1] Allard, J.R., "Method and system for remote control of mobile robot,"(2003).

[2] Du, R., Jagtap, V., Long, Y., Onwuka, O., and Padir, T, "Robotics enabled in-home environment screening for fall risks." Proceedings of the 2014 workshop on Mobile augmented reality and robotic technology-based systems, pp. 9–12.

[3] Gomez, C., Hernandez, A.C., Crespo, J., and Barber, R., "Learning robotics through a friendly graphical user interface, " *ICERI2016 Proceedings*, (2016), pp. 483–492.

[4] Joseph, L., *ROS robotics projects: Build a variety of awesome robots that can see, sense, move, and do a lot more using the powerful Robot Operating System.*, Packt Publishing, Birmingham, UK, 2017.

[5] Khandelwal, P., Zhang, S., Sinapov, J., Leonetti, M., Thomason, J., Yang, F., Gori, I., Svetlik, M., Khante, P., Lifschitz, V. and Aggarwal, J.K., Mooney, R., and Stone, P., 2017. Bwibots: A platform for bridging the gap between ai and human-robot interaction research. The International Journal of Robotics Research, 36(5-7), pp.635-659.

[6] Koenig, N. and Howard, A., 2004, September. 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, pp. 2149-2154). IEEE.

[7] Lamere, P., Kwok, P., Gouvea, E., Raj, B., Singh, R., Walker, W., Warmuth, M. and Wolf, P., 2003, April. The CMU SPHINX-4 speech recognition system. In IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong (Vol. 1, pp. 2-5).

[8] Lankenau, P., "Virtour: Telepresence system for remotely operated building tours", 2016.

[9] Lopez, J., Perez, D., Paz, E., and Santana, A., "WatchBot: A building maintenance and surveillance system based on autonomous robots, " *Robotics and Autonomous Systems*, vol. 61 (2013), pp. 1559–1571.

[10] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., 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).

[11] Toris, R., Kammerl, J., Lu, D.V., Lee, J., Jenkins, O.C., Osentoski, S., Wills, M. and Chernova, S., "Robot web tools: Efficient messaging for cloud robotics. In Intelligent Robots and Systems (IROS)", 2015 IEEE/RSJ International Conference on (pp. 4530-4537). IEEE.

Luo
Washkewicz College of Engineering
Cleveland State University
2121 Euclid Ave
Cleveland OH 44115
United States
d.luo@vikes.csuohio.edu
http://ndjfl.nd.edu/

Ma
Washkewicz College of Engineering
Cleveland State University
2121 Euclid Ave
USA
??

Zhou
Washkewicz College of Engineering
Cleveland State University
2121 Euclid Ave
USA
??