https://doi.org/10.31256/Lx2Pn5L

Web based ROS interface for telepresence

Sean Katagiri*, Joshua Roe, Matthew O'Hara, Jonatan Scharff Willners, Sen Wang and Yvan Petillot
Institute of Sensors, Signals and Systems
Heriot-Watt University
Edinburgh, UK
{s.katagiri, joshua.roe, m.ohara, j.scharff_willners, s.wang, y.r.petillot}@hw.ac.uk

***Abstract*—This paper describes the approach of using roslibjs in combination with rosbridge to create a custom web interface for remote telepresence. Remote connection from the web server to the robot is established through Husarnet to allow access to multiple robots simultaneously with low latency without the need to be in the same network. The interface enables viewing a 360 image that can be scrolled through, multiple views of the virtual visualisation of data, sending waypoints to the autonomous pilot, as well as basic controls of a manipulator attached to the Autonomous Underwater Vehicle (AUV). This approach has been tested with multiple users from around the world successfully operating an AUV remotely.**

***Index Terms*—Marine robotics, telepresence, navigation, human-robot interaction**

I. Introduction

With the increase in demand for autonomous inspection systems in remote locations such as offshore structures, there has also been an increase in demand for interfaces for enabling human operators to monitor and operate these systems remotely. Such an interface is required to have low latency on long-range remote operation, the capability to convey the relevant data comprehensively to the operator, and take user input from the operator to send commands directly or adjust the behaviour of the autonomy. A web-based solution can be used to allow access regardless of the type of smart device used to access the interface while being very easy to customise the User Interface (UI) elements, and supporting the integration of databases for user authentication as well as supporting multiple endpoints. Telepresence for marine robots has previously been used for observation of a subsea robot during environmental assessment [5]. In this paper, we will demonstrate a proof of concept design for a platform-agnostic web-based remote operation interface that is capable of handling multiple users simultaneously. The interface is also equipped with a full web stack with user authentication allowing control over access levels for each functionality, enabling distinction between observers with no influence over the autonomous system and operators with full control.

II. Visualisation

A key component of the system for enhancing telepresence is the 360 image that can be scrolled around in to view the surrounding image, shown in the bottom left of Fig. 1. This is achieved by converting the fisheye image feed from the 360 camera to an equirectangular image feed, which is then broadcast through a User Datagram Protocol (UDP) stream that the client-side browser can render into a spherical mesh through javascript. The use of UDP over Transmission Control Protocol (TCP) is preferable as it allows multiple clients to access the image feed simultaneously without significantly increasing the processing load on the server-side computer. The equirectangular image being rendered on a per client basis is also intentional to allow each client to retain their own individual view of the 360 image without having to share a perspective with other clients.

Visualisation of sensor data and the robot state is also crucial for a telepresence system, which an implementation of can be seen in the top of Fig. 1. The two views are generated using a virtual camera within RViz through the use of the rviz_virtual_camera plugin. The plugin only requires the desired intrinsic camera parameters and a reference transform in Robot Operating System (ROS) [2], allowing easy configuration for different robotic platforms and vehicles. These images are mainly used for displaying the waypoint markers, pointclouds, or octomaps [1] in our particular implementation. However any visualisation available in RViz is also available to be displayed if required. The generated image feed is then passed through to the web interface by having rosbridge republish the images to a websocket using TCP, and subscribed to by the client browser using roslibjs.

III. Remote Control of Robot

Rosbridge and roslibjs discussed in the previous section is also used in the opposite direction of communication for sending commands from the client to the AUV. The top down view in the top left of Fig. 1 also acts as an interface for sending waypoints. The location of a click on the html element is converted into coordinates in the base_link frame in a node running on the server computer, which is then sent to the AUV pilot for conversion into the navigation frame and execution of the waypoint. The autonomous pilot checks the waypoint and plans a safe route with collision avoidance enabled, while also choosing the best orientation for keeping trackable features in view for the visual Simultaneous Localisation And Mapping (SLAM) [4] that is used in the autonomy system [3]. The

This work was supported by the EPSRC funded ORCA-HUB (EP/R026173/1).

* Corresponding author.

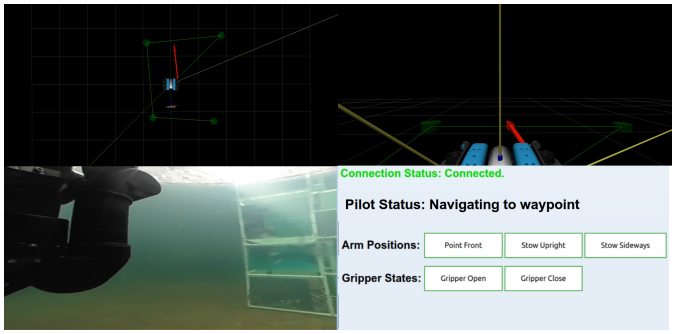

Fig. 1. The web interface as seen by the operator.

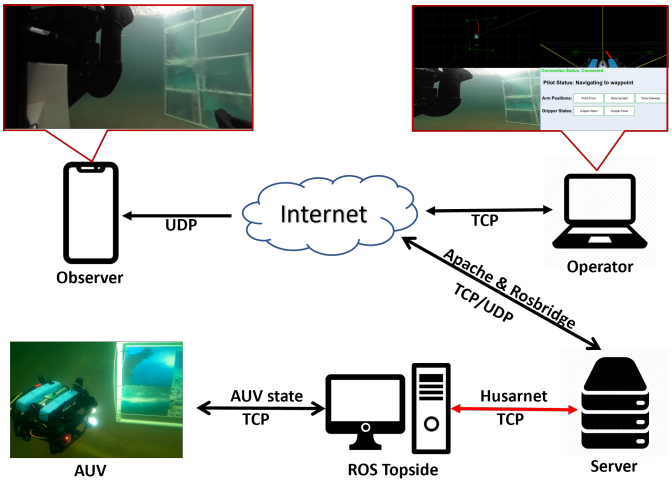

Fig. 2. Network structure used for the overall system.

Fig. 3. AUV used for demonstration of system, a BlueRov2 heavy with an additional skid for the SLAM system and another for the 360 camera plus reach alpha manipulator.

AUV is equipped with a 5 degrees of freedom manipulator[1] that can be controlled using labelled preset joint positions and gripper states which are selected using the buttons in the web interface, once again through rosbridge to communicate with the ROS driver. Although high latency connections are relatively harmless due to the autonomous system ensuring collision avoidance and station keeping, Husarnet is used to simplify and secure the connection between the server and the AUV. In addition, this allows the server to be located in a central location with good connectivity while the AUV is in a remote location without sacrificing low latency connections (approximately 100-200ms delay in most situations). A full web stack is implemented with SQL database and php serverside operations to authenticate and control access rights to the ability to control the AUV on a per client basis. With multiple access levels it is possible to give the majority of clients access to components such as 360 camera view, while a limited number of select clients get a TCP connections to the AUV for monitoring and control. By restricting the number of TCP connections to the system the bandwidth and computing power usage is kept to a minimum. The final network structure of the overall system can be seen in Fig. 2.

IV. System Evaluation

The web interface was demonstrated during the Robot Lablive 2021 & 2022 under Roboquarium with a BlueROV[2] in a wave tank (approximately $12 \times 10 \times 2.5$ metres) at Heriot-Watt University. The BlueROV is equipped with a custom payload containing stereo cameras and computational power to enable SLAM and autonomous operations, hence enabling it to operate as an AUV [3]. An additional skid is attached to house the 360 camera and the manipulator as seen in Fig. 3 During the demonstration, there were up to 20 clients connected to our system simultaneously, and a select few were given the access rights to control the AUV directly from their smart devices remotely. The demo lasted for 4 sessions each for up to 25 minutes per session, with a total of 8 unique operators over the course of the demonstration. Each operator was tasked with and was successful in finding the hidden object around the structure in the wave tank by controlling the AUV using the web interface. For future work we are looking into deploying and testing the system on different robots, such as the Boston Dynamics' SPOT robot, as well as integrating additional functionalities such as predefined waypoint control to turn the system into a truly platform-agnostic solution.

[1]https://blueprintlab.com/products/manipulators/reach-alpha/
[2]https://bluerobotics.com/store/rov/bluerov2/

References

[1] A. Hornung, K. Wurm, M. Bennewitzm, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," Autonomous Robots, 2013.
[2] M. Quigley, et al., "ROS : an open-source Robot Operating System," ICRA Workshop on Open Source Software 2009.
[3] J. Scharff Willners, et al., "From market-ready ROVs to low-cost AUVs," IEEE Oceans 2021.
[4] S. Xu, , et al., "Underwater Visual Acoustic SLAM with Extrinsic Calibration," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2021.
[5] I. Rekleitis, G. Dudek, Y. Schoueri, P. Giguere, and J. Sattar, "Telepresence across the ocean," CRV 2010 - 7th Canadian Conference on Computer and Robot Vision, 2010.