Online Monitoring and Visualization with ROS and ReactJS

Artem Ivanov
Laboratory of Intelligent Robotic Systems (LIRS)
Intelligent Robotics Department
Institute of Information Technology and Intelligent Systems
Kazan, Russia
artevivanov@stud.kpfu.ru

Aufar Zakiev
Laboratory of Intelligent Robotic Systems (LIRS)
Intelligent Robotics Department
Institute of Information Technology and Intelligent Systems
Kazan, Russia
zaufar@it.kfu.ru

Tatyana Tsoy
Laboratory of Intelligent Robotic Systems (LIRS)
Intelligent Robotics Department
Institute of Information Technology and Intelligent Systems
Kazan, Russia
tt@it.kfu.ru

Kuo-Hsien Hsia
Department of Electrical Engineering
National Yunlin University of Science and Technology
Douliou, Taiwan
khhsia@yuntech.edu.tw

2021 International Siberian Conference on Control and Communications (SIBCON) | 978-1-7281-8504-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/SIBCON50419.2021.9438890

***Abstract*—This paper presents a tool for monitoring and visualization of Robot Operating System (ROS) data in a standard browser. It is compatible with ROS and uses ReactJS to visualize topic data. The browser-based solution is software-agnostic and does not require ROS installation to view data, which are sent via ROS topics. It is compatible with most browsers and could be deployed on a remote hosting. The deployed solution could be accessed by a remote user with no need to connect the robot's local network. The developed software was tested on Turtlebot 3 robot in the Gazebo simulator. It has a modular structure and could be integrated into any robotic system based on ROS.**

***Index Terms*—visualization, monitoring, ReactJS, GUI, TurtleBot3, mobile robot, sensors**

I. Introduction

Every software and hardware system developer requires a stable way to get internal data of the system. Internal data could be used both during a development process and at a production stage. These data are retrieved using specially developed monitoring systems that could include different features such as real-time monitoring, logging, stats collection, anomaly detection, etc [1]. A monitoring system integrated into a robot should create minimum load on a CPU [2] and should be available remotely since robots are distinct devices with no physical connection to a developer's machine [3], [4].

Robot Operating System (ROS) is a common framework for robotic software development. It has a wide variety of algorithms implemented and available as independent *packages*. The framework connects a set of binaries using network connections and provides a very high flexibility [5], [6]. This means that ROS-based systems often have complex data flow schemes and multiple jobs running in parallel. To cope with this complexity, we require a flexible monitoring tool to integrated into the ROS system easily.

This paper presents a ROS monitoring system based on React web-framework. It could be integrated into any ROS-based robotic system and acts as a ROS node. This monitoring system could reserve the external domain and be accessed from anywhere. Since it is a web application, the monitoring system does require only the web browser to be installed. We have tested the developed tool using Turtlebot 3 robot simulation. The tool is entirely written in JavaScript.

JavaScript [7] has been used for web development for several decades. ReactJS is a relatively new framework with a wide variety of features to create powerful and effective tools for data visualization. ReactJS [8] and ROS integration is the way to create a cross-platform tool for monitoring the robots. This paper proposes a web-based monitoring tool for ROS-controlled robots [9]. It fetches the data from ROS topics and visualizes them in a remotely available web application.

This paper is organized as follows: Section 2 is about related work, Section 3 describes main concepts connected with hard and software and basic principles of the provided solution. Section 4 digs deeper into the architecture, final sections are dedicated to results and the conclusion.

II. Related Work

There are several tools existing for the visualization of ROS. The most commonly used is the built-in graphical tool is RViz [10]. It provides a robot model visualization, capturing data from the robot sensors and deep customization features. It

This work was supported by the Russian Foundation for Basic Research (RFBR), project ID 19-58-70002.

can display data obtained from a camera, laser range finders, inertial measurement units, etc. However, RViz needs direct access to the robot's network and requires ROS installation on the operator's machine. These requirements limit the use of the RViz on non-preconfigured machines.

Another example of a visualization tool is a web-based application called Webviz [11]. This tool could be used to playback and visualize ROS bag files or robots in real-time. It solves the problem of remote monitoring, however, it is facing several compatibility problems with different browsers. Moreover, it requires a huge amount of installed libraries, therefore this software is very recourse consuming. Also, considering it is a universal monitoring tool and not oriented to a certain model of the robot.

Another ROS-connected web-project is Jviz [12]. It is a non-official tool building the graph of the relation between the ROS nodes. It does not fetch the data from the ROS topics but reflects the dependencies between ROS nodes and data streams between them. Thus, it is still should be considered a convenient tool for visualization in ROS.

Robot Management System (RMS) [13] is a web-management system written in PHP that is backed by a MySQL database. It uses roslibjs to communicate with ROS servers and control a variety of robots. RMS is a remote management tool designed for use with controlling ROS-enabled robots from the web. This software is intended to be installed on UNIX-based web servers. Thus, it is not considered to be light-weight solution and requires complex setting up process.

Ros Control Center is a universal tool for controlling robots running ROS. It runs in the browser using a WebSocket connection and roslibjs from RobotWebTools [26]. Provided application has the following features: nodes, topics and service monitoring, publishing messages, controlling ROS parameters. This solution is Angular-based and focused on control rather than monitoring.

III. Robot hardware and software

The mobile robot TurtleBot3 Waffle Pi (Fig. 1) is a differential drive robot designed as a collaboration project by Open Robotics and ROBOTIS It is a relatively small robot for education and research purposes. TurtleBot3 Waffle is equipped with the following hardware:

- 360° LiDAR (used to emit laser signal, which reflects of nearby obstacles)
- Raspberry Pi (single-board computer, commonly used for applications that do not require high processing power)
- optical camera
- OpenCR (Connects to Raspberry Pi and allows it to control the motors and sensors)
- 11.1V Lithium Polymer battery
- Bluetooth module
- Dynamixel XL430-W250 servo motors

Software of the Turtlebot3 is unified: on-board computer (Raspberry Pi) runs lightweight Linux distribution Ubuntu 14.04. ROS is installed on the SD memory card, providing the robot several abilities for publishing and subscribing via ROS Topics.

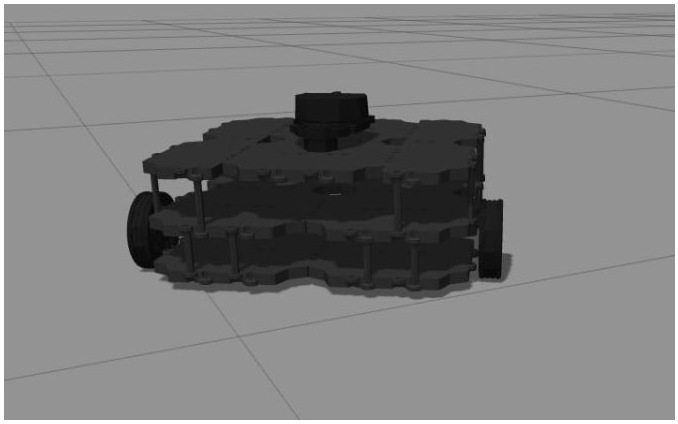

Fig. 1. TurtleBot Waffle Pi

The robot is integrated into the ROS framework and uses the ROS navigation stack for localization [14], path planning [15], [16], and simultaneous localization and mapping (SLAM) [17]. The manufacturer also provides ROS packages to create fully operational simulations. Several virtual experiments in Gazebo were designed to properly test the data receiving from the sensors and check its applicability to our solution.

IV. ReactJS integration architecture

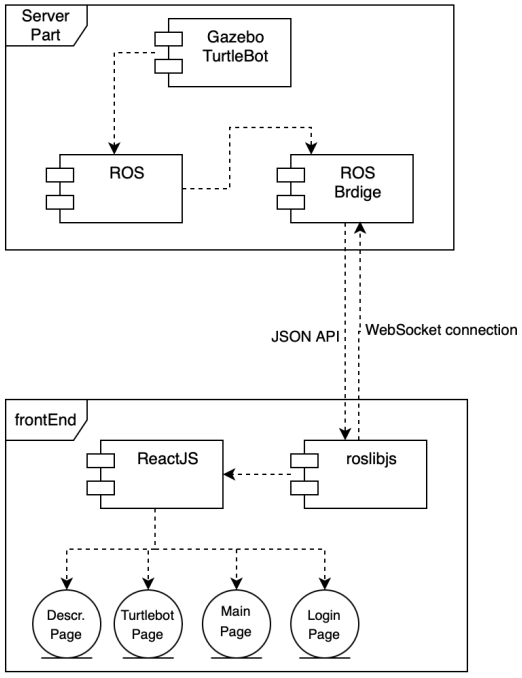

Fig. 2. Architecture of the application

The following technologies were used for back-end to implement the tool:

- roslibjs library [18]
- ROS Kinetic [19]
- ROS Bridge [20]
- surge (Hosting)

And the following technologies are used for front-end part of the tool:

- react-ros library
- npm package manager [21]
- yarn package manager [22]
- React Router [23]
- Material-ui (graphical component library that speeds up the development of the visual part) [24]
- React-hot-loader (for accelerated rendering from the server side parts)
- web_video_server (camera raw data visualization) [25]

The tool operates as a Representational state transfer (REST) application (Fig. 2). ROS Bridge fetches ROS messages and converts them to JSON to be processed by ReactJS. In order for the web app to connect to ROS, React-ros library was used. It uses WebSockets to connect with rosbridge and provides publishing, subscribing, service calls, actionlib, TF, URDF parsing, and other essential ROS functionality. REST architecture allows several ROS Masters to exist and connect to the application.

The Router library is used to emulate web links (dynamic routing). This is the core of the navigation of almost any ReactJS Application. Its main concept is to provide rendering of certain components depending on URLs. Moreover, all the routes are stored in the BrowserHistory list, which is used to navigate back and forth using the browser's navigation arrows. Also, this library can be used to open web_video_server in a separate window of the browser. Redux library allows data to pass through bypassing Document Object Model (DOM) tree by using special stores, that are located outside of the tree. Redux's actions are used to trigger any event. After that, functions called reducers combine the state of the store and Redux actions to change data inside the stores and return a new state of the app. Without Redux, data of the app can only move through the DOM tree from the root to the leaves, reducing the webpage updated rate. With Redux, data stores are accessible on any level of the tree.

React-ros is a non-official library, which creates a more convenient way to connect roslibjs and ReactJS, taking into account that initially roslibjs was designed for pure JavaScript, rather than for its frameworks. It provides ROS hook for functional components and simplifies the process of connecting to the core JavaScript library for interacting with ROS from the browser (roslibjs). It uses WebSockets to connect with rosbridge and provides essential ROS functionality.

V. Results

A. Connecting to /ODOM

The application consists of several pages: login page, main page, description page and robot data page.

Login page is a page allows user to enter the website. After authorization, app saves token to the LocalStorage. Thus, the user data is saved in a browser and persists until the user manually clears the cache or until the web app clears the data.

Main page x has modular architecture and consists of a set of tiles. Each tile represents a single connection to the specific ROS Master. Successful connection automatically detects the list of active ROS topics and shows the connection status (Fig. 4).

Description page is a short instruction for the newcomers. It only shows the list of tools required to be installed on the robot (note: not on the operator's machine) to use the features of the proposed visualization tool.

The robot's page (Fig. 5) is UI that visualizes general info about the connected robot. It uses data fetched from */odom* topic and shows them with the user-defined approximation. The UI provides data about its position in the simulator in two-dimensional coordinates relative to the starting point as well as its orientation. Additionally, the robot's speed (linear velocity and angular) is included in the UI.

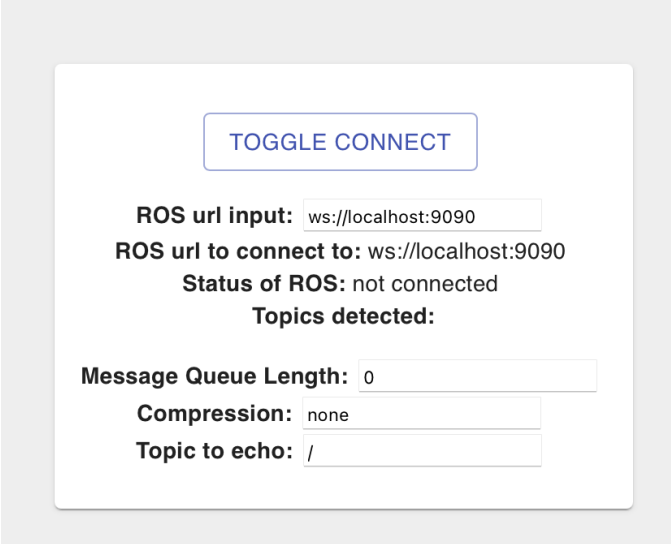

Fig. 3. Main page (Disconnected)

B. Receiving data from camera

In order to work with the camera, the aforementioned web_video_server tool is required, because the camera sends raw data, that cannot be displayed directly. Web_video_server is a ROS package for HTTP Streaming of ROS Image Topics in Multiple Formats. Therefore, the extra tool allows information from the robot's camera to be visualized in the separate browser tab.

VI. Conclusion

This article presented a tool for monitoring and visualization of Robot Operating System (ROS) data in a browser. The described solution has demonstrated the effectiveness of using ReactJS with ROS. Most browsers support this software, its

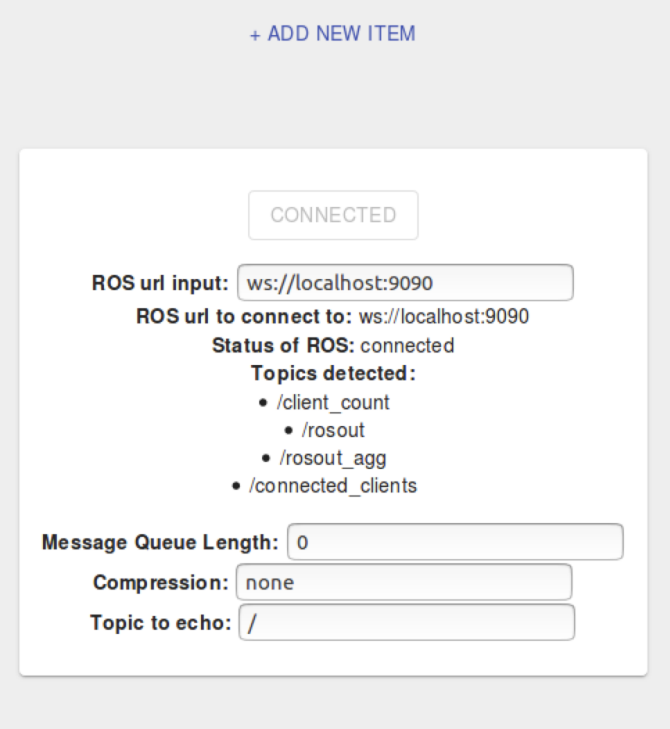

Fig. 4. Main page (Connected)

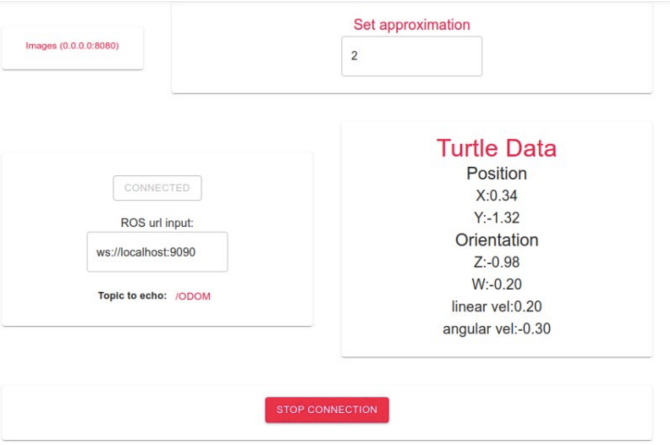

Fig. 5. Turtlebot page

architecture let the application to be deployed on a remote hosting and, moreover, it's modular structure allows to be embedded into other developing ReactJS application. Thus, the provided solution can be applied to several other projects and broaden the use of JavaScript with ROS.

References

[1] Zakiev, A., Tsoy, T., & Magid, E. (2018, September). Swarm robotics: remarks on terminology and classification. In International Conference on Interactive Collaborative Robotics (pp. 291-300). Springer, Cham.

[2] Moskvin, I., & Lavrenov, R. (2020). Modeling tracks and controller for servosila engineer robot. In Proceedings of 14th International Conference on Electromechanics and Robotics "Zavalishin's Readings" (pp. 411-422). Springer, Singapore.

[3] Pashkin, A., Lavrenov, R., Zakiev, A., & Svinin, M. (2019, October). Pilot communication protocols for group of mobile robots in USAR scenarios. In 2019 12th International Conference on Developments in eSystems Engineering (DeSE) (pp. 37-41). IEEE.

[4] Amirabdollahian, F., Dautenhahn, K., Dixon, C., Eder, K., Fisher, M., Koay, K. L., ... Webster, M. (2013). Can you trust your robotic assistant?. Social Robotics.

[5] Abbyasov, B., Lavrenov, R., Zakiev, A., Yakovlev, K., Svinin, M., & Magid, E. Automatic tool for Gazebo world construction: from a grayscale image to a 3D solid model. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 7226-7232). IEEE.

[6] Lavrenov R., Zakiev A., Magid E. Automatic mapping and filtering tool: From a sensor-based occupancy grid to a 3D Gazebo octomap //2017 International Conference on Mechanical, System and Control Engineering (ICMSC). – IEEE, 2017. – C. 190-195.

[7] Crockford D. JavaScript: The Good Parts: The Good Parts. – " O'Reilly Media, Inc.", 2008.

[8] Fedosejev A. React. js essentials. – Packt Publishing Ltd, 2015.

[9] Bai, Y., Wang, Y., Svinin, M., Magid, E., & Sun, R. (2020). Function approximation technique based immersion and invariance control for unknown nonlinear systems. IEEE Control Systems Letters, 4(4), 934-939.

[10] Kam, H. R., Lee, S. H., Park, T., & Kim, C. H. "Rviz: a toolkit for real domain data visualization." Telecommunication Systems 60.2: 337-345, 2015.

[11] Pitkow, James Edward, and Krishna A. Bharat. Webviz: A tool for world wide web access log analysis. Georgia Institute of Technology, 1994.

[12] Wiese, Kay C., Edward Glen, and Anna Vasudevan. "jViz. Rna-A Java tool for RNA secondary structure visualization." IEEE transactions on nanobioscience 4.3: 212-218, 2005.

[13] Toris, R., Kent, D., & Chernova, S. (2014). The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. Journal of Human-Robot Interaction, 3(2), 25-49.

[14] Guimarães, Rodrigo Longhi, et al. "ROS navigation: Concepts and tutorial." Robot Operating System (ROS). Springer, Cham. 121-160, 2016.

[15] Raja, Purushothaman, and Sivagurunathan Pugazhenthi. "Optimal path planning of mobile robots: A review." International journal of physical sciences 7.9 (2012): 1314-1320.

[16] Lavrenov, R., & Magid, E. (2017). Towards heterogeneous robot team path planning: acquisition of multiple routes with a modified spline-based algorithm. In MATEC Web of Conferences (Vol. 113, p. 02015). EDP Sciences.

[17] Durrant-Whyte, Hugh, and Tim Bailey. "Simultaneous localization and mapping: part I." IEEE robotics & automation magazine 13.2 (2006): 99-110.

[18] Alisher, K., Alexander, K., & Alexandr, B. (2015). Control of the mobile robots with ROS in robotics courses. Procedia Engineering, 100, 1475-1484.

[19] Fairchild, C., & Harman, T. L. (2017). ROS Robotics By Example: Learning to control wheeled, limbed, and flying robots using ROS Kinetic Kame. Packt Publishing Ltd.

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

[21] Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to build high-performance network programs. IEEE Internet Computing, 14(6), 80-83.

[22] McKenzie, S., Nakazawa, C., & Kyle, J. (2016). Yarn: A new package manager for javascript. Luettavissa: https://code. facebook. com/posts/1840075619545360. Luettu, 1, 2017.

[23] Subramanian, V. (2019). React Router. In Pro MERN Stack (pp. 233-265). Apress, Berkeley, CA.

[24] Boduch, A. (2019). React Material-UI Cookbook.

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

[26] Alexander, B., Hsiao, K., Jenkins, C., Suay, B., & Toris, R. (2012). Robot web tools [ros topics]. IEEE Robotics & Automation Magazine, 19(4), 20-23.