Development of a web application to support validation of indoor robots using AR

Kento Hirose[1] and Masanobu Koga[2*]

[1]Creative Informatics, Kyushu Institute of Technology,
680-4 Kawazu, Iizuka, Fukuoka, Japan (koga@ics.kyutech.ac.jp)
[2]Creative Informatics, Kyushu Institute of Technology,
680-4 Kawazu, Iizuka, Fukuoka, Japan (hirose.kento646@mail.kyutech.jp) * Corresponding author

Abstract: In this paper we report on the web application which was developed to help users considering the purchase of an indoor robot to verify the effectiveness of the robot in their environment of use. The application uses a robot model in RDTF format that works in a web browser. For the simulation calculations, it use a physical calculation engine compiled with WebAssembly or other tools to run at high speed, so that the robot can run at high speed even in a web browser. Similarly, for animation, it use a library compatible with WebGL, which enables high-speed display on a Web browser. In order to enable users to check the results of the indicated task and confirm the validity of the robot's purchase by checking the 3D animation in the operating environment, the trajectory of the task is generated in conjunction with ROS.

Keywords: RDTF, URDF, glTF, ROS, babylon.js, ode.js

1. INTRODUCTION

In recent years, indoor robots have been developed that can be made to perform tasks of daily life. In order to properly select and use these robots, it is necessary to understand and consider task performance (service), size, appearance, price, and power consumption. In a paper by Tsusaka, Okazaki, Komatsu, and Yokokoji [1], they state the following: "Major industrial robots currently in operation operate in an environment adapted to the robot's environment, making it difficult to operate in an environment similar to that of a home. Therefore, it is necessary to establish a robot motion generation method that can flexibly respond to environmental changes in order for robots to perform various tasks in the home." However, even if the robot can respond flexibly, it is difficult to respond perfectly, as there are situations in which the robot will not work depending on the environment in which it is placed. In this regard, it is important to confirm that the tasks required of the robot and the size of the robot are appropriate in the environment in which the robot will operate. However, unlike the pre-introduction verification of industrial robots, the purchasers of indoor robots are general households. Therefore, it is difficult to prepare and verify an actual robot for general users (hereinafter, purchasers of indoor robots in general households are referred to as "general users").

Since it is difficult to prepare and test a real machine, we next consider simulation. Gazebo[2] is software for robot simulation. Gazebo has a built-in physics engine that processes the robot's algorithms and sensor data, allowing the physical behavior of the robot to be verified in real-time simulations. It is also possible to visualize the robot's movements during the simulation, allowing the user to intuitively verify the robot's behavior. Similarly, Unity[3] can be used to simulate robots. Unity is an integrated development environment used to create 3D applications. Unity's editor is GUI-based, allowing the average user to manipulate the robot intuitively. However, the operating environment of gazebo and Unity depends on the user's environment. This creates an obstacle for the general user to verify the validity of the robot. Therefore, a web application that runs on a browser and does not require the user to build an environment is desirable for verifying indoor robots.

An example of a web application that can verify robot behavior is robotbenchmark.net[4]. Because the web application runs in the browser, the general user can verify the robot's behavior on any device. Robotbenchmark uses Webots instances running in the cloud to run physical simulations in real time. This allows step-by-step learning of robot control without depending on the execution environment. However, since learning is the main objective, the robot can only be controlled by code written in the Python language. It is also not possible to display AR tailored to the user's environment, taking advantage of the robot's ability to run on any device.

Therefore, we propose a web application that allows the user to check the validity of the operating environment of an indoor robot on a web browser. And, We propose to connect it with ROS to enable it to perform tasks.

2. WEB APPLICATION DEVELOPMENT

2.1 Requirement

Consider the functions required for the application. First, the application needs to be a web application that can be used easily by general users, so it needs to be a web application that runs on a browser without the need to build an environment. Therefore, as a simulation function that operates in a Web application, it needs to be fast enough that users do not feel stressed. Next, it is necessary to have the robot perform the tasks instructed by the general user in order to check the validity of the indoor robot. In order to verify the validity of the system in the operating environment, it is desirable to be able to verify

the system while being aware of the real space in which the system is operated by general users. Therefore, the application requirements can be defined as follows.

- Fast enough to be used by general users without stress
- Can make the robot perform tasks for validation
- Has a function to validate the robot while checking the real space

2.2 Methodology for Simulation and Animation

Generally, Web applications are slower than desktop applications, so simulation and animation in Web applications require consideration of speeding up. Therefore, a physical computation engine compiled with WebAssembly[5] etc. is used for the computation for simulation. For animation, a library compatible with WebGL, which enables high-speed display on a Web browser, is used. Figure.1 shows an overview of the method of simulation on the browser. After the simulator calculates the robot's state, the robot's 3D model is animated by updating its state. This allows the simulation to be performed in the browser.

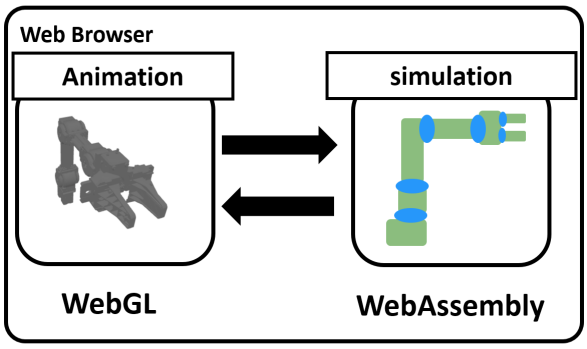

Fig. 1. Web Application Development Methodology

The developed application g2rViewer uses ODE.js[6] for physics calculation Babylon.js[7] for display and to enable physics simulation and realistic 3D animation using only a web browser. Babylon.js is a WebGL framework that makes it easy to develop 3D content and applications with WebGL. ODE.js is a library of the Open Dynamics Engine (ODE), originally a C/C++ language library, compiled into WebAssembly, a binary code format that runs fast on the browser. Figure.2 shows the process of simulation and animation.

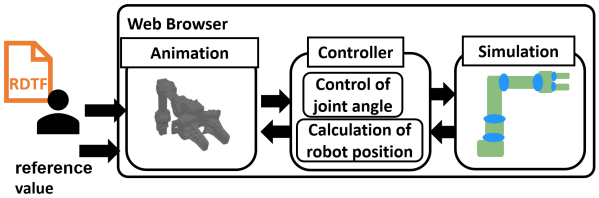

Fig. 2. Web Application Overview

1. By loading a model in RDTFs using the glTF loader in Babylon.js, a robot model for display and a robot model for physics calculation are created in Babylon space and in ODE space, respectively.
2. Perform Control simulation in ODE.js make the robot to follow the reference value.
3. Reflect the simulation results done in ODE space to the model on Babylon space.

g2rViewer uses RDTF[8] as its robot model format. Although glTF[9] is a 3D model format often used for display in a Web browser using WebGL, it does not have the physical parameters necessary for simulation. Therefore, we use the model format RDTF, which adds physical parameters to glTF. RDTF is a format with physical parameters like URDF[10] that can be easily used by anyone like glTF by adding mass, inertial properties, friction coefficient of joints, etc. to glTF as extended properties.

RDTF can be generated from URDF. A 3D model in RDTF format generated from a 3D model in URDF format used in ROS can be used as a glTF model for animation with reference to physical parameters when performing simulations. Also, since the model is generated from URDF, it has the necessary parameters to be used in conjunction with ROS.

2.3 Cooperation with ROS

It is necessary to have the robot perform the task directed by the general user to enable the general user to perform the validation. However, in order to accomplish the work directed by the general user, it is necessary to generate a trajectory of how the robot will move to accomplish the task. Therefore, we considered linking with ROS in terms of being able to control the robot on the actual machine. ROS[11] is a meta-operating system for robot system development. In ROS, the robot's sensors, actuators, and control algorithms exchange information to control the robot. The moveit[12] library can be used to plan the robot's trajectory and control the robot. Using roslibjs[13] and rosbridge[14], web applications can communicate with ROS nodes. When connecting ROS and the web application and exchanging information, the ServiceServer, which is capable of bidirectional communication of information, is used to issue commands and use ROS functions. Figure.3 shows a schematic of the communication method with ROS.

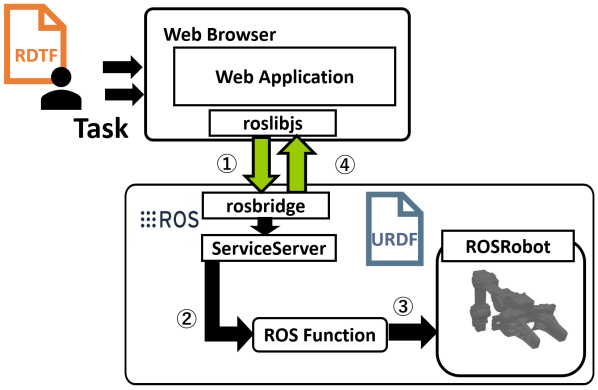

Fig. 3. Outline of communication method with ROS

1. Using roslibjs and rosbridge, connect to ServiceServer by service communication from the web application and give instructions.

2. Use ROS functions from ServiceServer.
3. Send joint angle information to the ROS robot. (Confirmation is also made on the ROS side)
4. Send the information on the ROS side as reference value to the web application by topic communication using roslibjs and rosbridge.

In ROS, there are two main communication methods: topic communication and service communication. In topic communication, a circuit called a ROS topic is used to communicate from the Publisher, who sends information, to the SubScriber, who receives the information. In service communication, the ServiceClient first sends a command to the ServiceServer. Then The ServiceServer processes the commands received from the ServiceClient and returns the results. This communication method is used to link the web application and ROS. An overview of the developed web application's communication with ROS is shown in Fig.4.

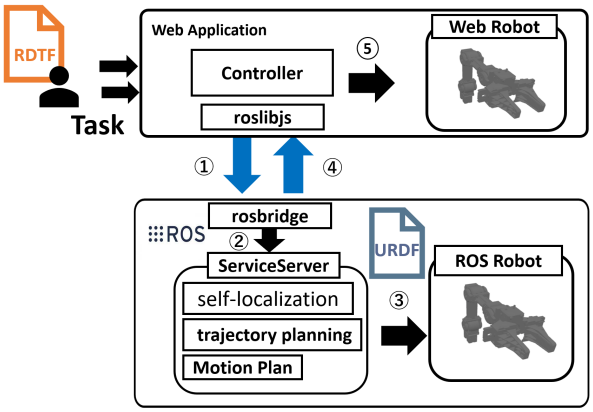

Fig. 4. Communication with ROS

1. roslibjs and rosbridge are used to connect from the web application to the ServiceServer via service communication and issue instructions.
2. Update the robot status and plan the trajectory using moveit from ServiceServer.
3. Send joint angle information to the robot controller. (Operate the robot on the ROS side)
4. Send information of joints as reference value to g2rViewer by topic communication using roslibjs and rosbridge.
5. g2rViewer's controller controls the robot on g2rViewer side to become the reference value

The information sent to and received from the ServiceServer at this time is as follows.

contents of transmission to ROS

- Current joint angle
- Obstacle size
- Obstacle position
- Obstacle posture
- Object size
- Object position
- Object posture
- goal of EndEffector position

Content received from ROS

- goal joint angle

First, g2rViewer's "Current joint angle" is sent to synchronize the state of the robot on the g2rViewer side and the robot on the ROS side. The "Current joint angle" is the angle of each joint of the robot on the g2rViewer side. After that, the following items are sent for the work to be performed by the robot. EndEffector position is the position the robot is expected to reach. Obstacle size, position, and posture are environmental information that must be taken into account when the robot operates. Object size, position, and posture are information on objects to be carried by the robot. After that, the ROS side receives the goal joint angle, which is the joint angle when the robot performs the indicated work.

2.4 Robot model superimposed on AR display

In addition to simulation in the virtual space, g2rViewer enables users to check the operation in the user's operating environment by using the AR function. The mechanism of superimposing the robot model on the real space when using the AR function is shown below.

Fig. 5. Placement of virtual objects

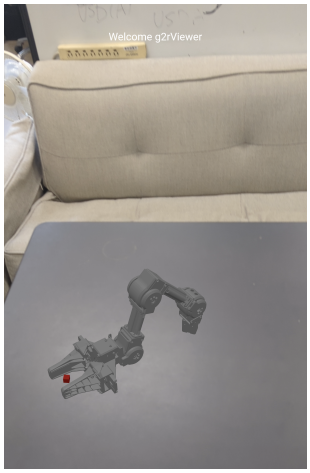

Fig. 6. Placement of Robot

1. When the camera is started, plane detection is performed and the plane directly under the camera the coordinate origin is located at.

2. Virtual object is placed on the plane as shown in Fig.5.
3. Users clicks on the object at the location where the robot model is to be superimposed.
4. Record the coordinates of the object when clicked and place the robot model at those coordinates as shown in Fig.6.

2.5 Obstacle Placement

In order to simulate a robot in its operating environment, it is necessary to make the environment in which the simulation is performed closer to the actual operating environment. Therefore, multiple obstacles are placed in the operating range of the robot during the simulation. This allows us to confirm the validity of the approximate environment in which the robot will operate. When placing obstacles, a rectangular object should be placed to cover the obstacles. Figure.7 shows the placement of the rectangle to cover the obstacle. Obstacle information is transmitted together with the coordinates of the goal when instructing the ROS side to plan a trajectory to avoid the obstacle. In addition, by dragging and dropping a rectangle that approximates an obstacle on the screen rectangle can be placed in any desired position and posture.

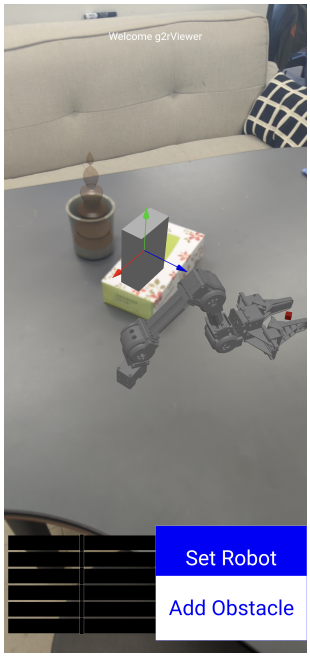

Fig. 7. Obstacle setting

2.6 User Interface

Figure.8 shows a web application we developed to read and display the RDTF of the robot arm. By communicating with the ROS, the user can set a starting point and a goal point to plan the trajectory and move the robot. It is also possible to change each joint of the robot arm by input from the user.

The buttons lined up on the left side can be used to perform as shown in Table.1.

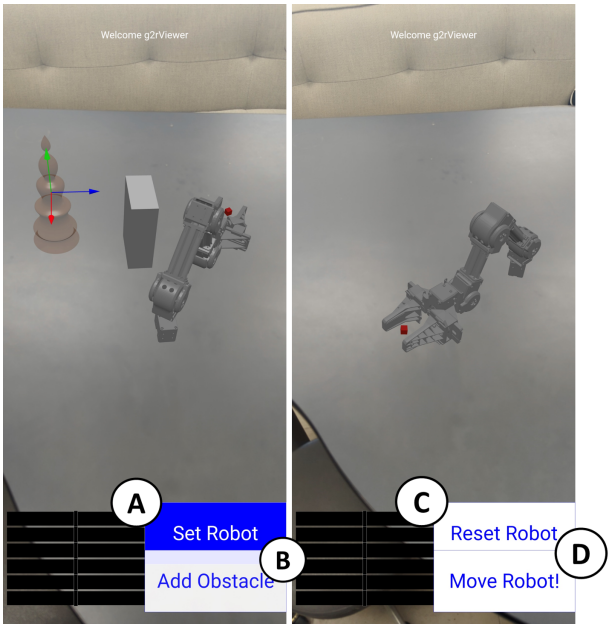

Fig. 8. Simulation of robot arm motion

Table 1. Contents of buttons

name of button	Description
Set Robot	Set the start posture and goal object coordinates of the robot.
Add obstacles	Place obstacles.
Reset Robot	Enables manipulation of the of the robot and the moving object posture.
Move Robot!	On the robot posture and goal, perform trajectory planning and move the robot.

3. EXAMPLES

The screen of g2rViewer during the operation of experiment is shown in Fig.9. The task of having robot A

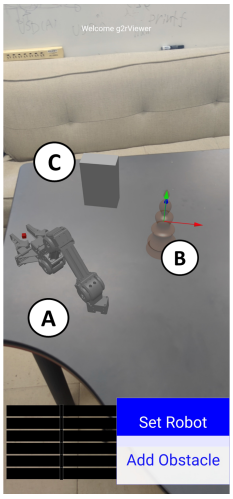

Fig. 9. g2rViewer screen during operation experiment

avoid obstacle C and retrieve the target object to be carried to hand is used as an example to verify whether the

robot is validated to be able to perform the operation. The procedure of the motion experiment is as follows.

1. As shown in Fig.7, set the position and size of rectangle C to cover the obstacle in the actual operating environment.
2. As shown in Fig.10, move object B, which is assumed to be a goal object, to the goal.
3. As shown in Fig.11, set the current robot posture and goal.
4. Use a motion command.

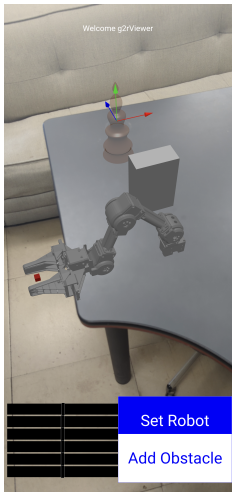

Fig. 10. Set goal coordinates

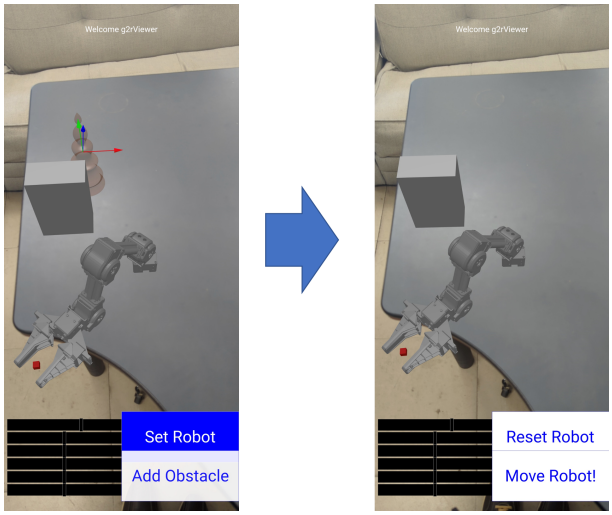

Fig. 11. Robot and object fixation

Figure. 12 and Fig.13 show the transition of the end-effector positions of the robot on the g2rViewer side and the robot on the ROS side during the execution of the robot's task when there are no obstacles and when there are obstacles. The base position of the robot is the xyz=0,0,0 origin. The starting point is near xyz=0.15,-0.15,0.02, and the robot is moving to the goal near xyz=0.18,0.2,0.02. The size of the obstacle is wide,height,depth=0.1,0.2,0.1 and its position is around xyz=0.2,0,0.15. This confirms that the position of the end-effector of the robot on the g2rViewer side follows the position of the end-effector of the robot on the ROS side. The robot is able to perform the task of avoiding obstacles and reaching the goal.

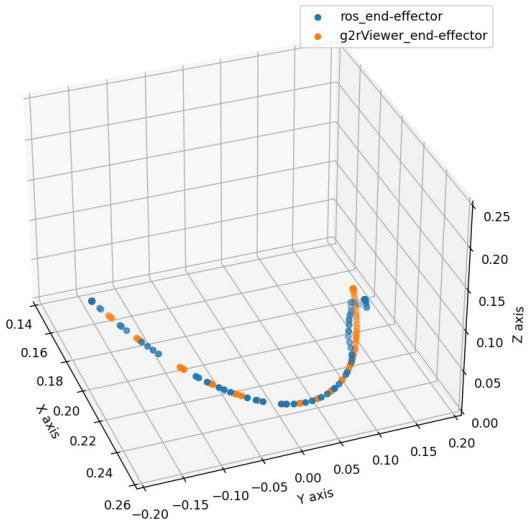

Fig. 12. Movement of the robot during task execution

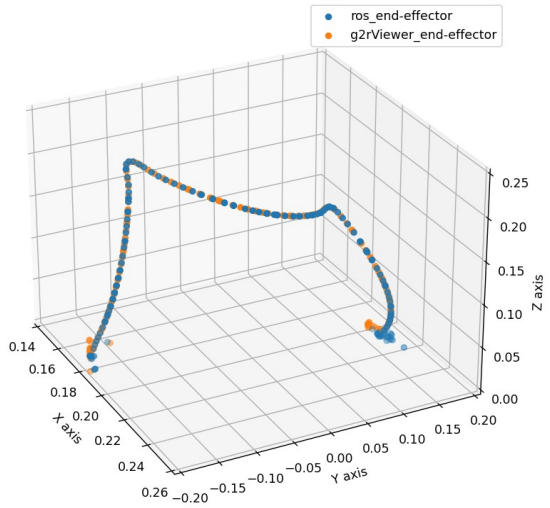

Fig. 13. Movement of the robot during task execution with obstacle

4. CONCLUSION

In this study, we developed a web application that can perform simulations on a web browser to check the validity of indoor robots in their operating environment. In addition, we proposed to link with ROS in order to be able to execute tasks. As a result, we found that we can develop a web application that can support validation in the operating environment of an indoor robot. We have not been able to verify the errors in the superimposition of the robot on the AR display. Therefore, in the future, we will consider how much error to allow for when displaying the robot in AR. In addition, it is necessary to study the superimposition method to reduce the error. By developing a user interface that facilitates validation for users, it is expected that more accurate operation validation will be possible, which will help users find and solve problems when purchasing a robot.

REFERENCES

[1] Yuko Tsusaka, Yasunao Okazaki, Mayumi Komatsu and Yasuyoshi Yokokohji, Proposal of Robot Motion Sequence Generation Method for Domestic Housekeeping Tasks based on the Robotizing Strategy Considering Required Adaptability and Difficulty of Task Transactions of the Institute of Systems, Control and Information Engineers Vol.28, No.6, pp.237-248, 2015

[2] gazebo, https://gazebosim.org/home

[3] unity, https://unity.com/ja

[4] robotbenchmark, https://robotbenchmark.net.

[5] webassembly, https://webassembly.org/.

[6] Open Dynamics Engine, https://www.ode.org/.

[7] Babylon.js, https://www.babylonjs.com.

[8] Ryoya Izawa , Masanobu Koga. An Extension of glTF for Robot Description. Proceedings of 19th international Conference on Control, Automation and Systems,pp.1010-1014, 2019.

[9] glTF, https://www.khronos.org/gltf/.

[10] URDF, https://wiki.ros.org/urdf/XML.

[11] ROS, http://wiki.ros.org/ja.

[12] moveit, https://moveit.ros.org/

[13] roslibjs, http://wiki.ros.org/roslibjs.

[14] rosbridge, http://wiki.ros.org/ja/rosbridge.