CHAPTER X

Web technologies in JdeRobot framework for robotics and computer vision

A.MARTÍNEZ[1] and J.M. CAÑAS[1]

[1]RoboticsLab, Universidad Rey Juan Carlos, josemaria.plaza@urjc.es

This paper presents a new development in the robotic software platform JdeRobot. Four web tools have been created to access to cameras, RBGD devices, wheeled robots and drones from any web browser. They use last generation web technologies like WebGL, WebWorkers and WebSockets to provide real time communication with robots and sensor devices. The sensors can be monitorized and the robot teleoperated from any computer, smartphone or laptop, regardless their operating system.

1 Introduction

In the last years, several robotic frameworks (also named SDKs or middlewares) have appeared that simplify and speed up the development of robot applications. They offer a simple and more abstract access to sensors and actuators than the operating systems of the robot. The SDK Hardware Abstraction Layer (HAL) deals with low level details of accessing to sensors and actuators, releasing the application programmer from that complexity.

They provide a particular software architecture for robot applications, a particular way to organize code, to handle code complexity when the robot functionality increases. In addition, robotic frameworks usually include simple libraries, tools and common functionality blocks, such as robust techniques for perception or control, localization, safe local navigation, global navigation, social abilities, map construction, etc.. This way they shorten the development time and reduce the programming effort needed to code a robotic application as long as the programmer can build it by reusing the common functionality included in the SDK, focusing in the specific aspects of the application.

Some tools of the JdeRobot robotic SDK (JdeRobot, 2016) let the human user to teleoperate robots (Kobuki wheeled robot, drone) and monitor the data from camera or RGBD sensors like Kinect. They are desktop applications in Linux that connect to the robot or sensor device and show a graphical user interface to the human user.

This paper presents the recent development of the web version of such tools. They provide the same functionality as the desktop equivalent but now from a web browser, so they are truly multiplatform. This way the robots can be teleoperated from any operating system (Windows, MacOs, Android, IOS...) and from any computer, even smartphones and tablets.

2 JdeRobot software platform

JdeRobot is a software development suite for *robotics* and *computer vision* applications. These domains include sensors (for instance, cameras), actuators, and intelligent software in between. It has been designed to help in programming such intelligent software. It is mostly written in C++ language and provides a *distributed component-based programming environment* where the application program is made up of a collection of several concurrent asynchronous components. Components may run in different computers and they are connected using ICE communication middleware (ZeroC, 2016). Components may be written in C++, Python, Java... and all of them interoperate through explicit ICE interfaces.

JdeRobot simplifies the access to hardware devices from the control program. Getting sensor measurements is as simple as calling a local function, and ordering motor commands as easy as calling another local function. The platform attaches those calls to remote invocations on the components really connected to the sensor or the actuator devices. Several *driver components* have been developed to support different physical sensors, actuators, both real and in simulation. They are ICE servers. Currently supported robots and devices include cameras, RGBD sensors (Kinect and Kinect2 from Microsoft, Asus Xtion), Kobuki (TurtleBot) from Yujin Robot and Pioneer from MobileRobotics Inc., ArDrone quadrotor from Parrot, laser scanners, Nao humanoid and Gazebo simulator.

JdeRobot includes several robot *programming tools and libraries*. For instance: viewers and teleoperators for several robots, its sensors and motors; a camera calibration component and a tunning tool for color filters; VisualHFSM tool for programming robot behavior using hierarchical Finite State Machines. In addition, it also provides a library to develop fuzzy controllers, a library for projective geometry and computer vision processing.

JdeRobot is *open-source software*, licensed as GPL and LGPL. It also uses third-party software like Gazebo simulator (OSRF, 2016), ROS, OpenGL, GTK, Qt, Player, Stage, GSL, OpenCV, PCL, Eigen, Ogre. It is ROS compatible.

3 Web tools

Four web tools have been developed using last generation web technologies like HTML5, JavaScript, CSS, WebGL, WebWorkers and WebSockets. All of them are ICE clients that run in the web browser and connect to the corresponding JdeRobot driver using ICE and Websockets.

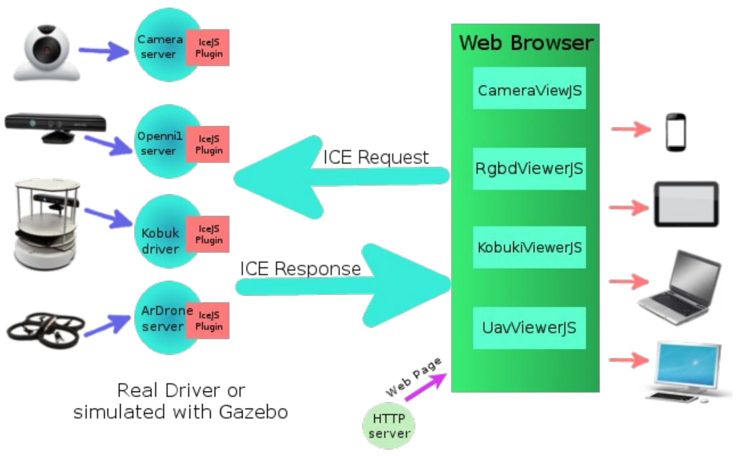


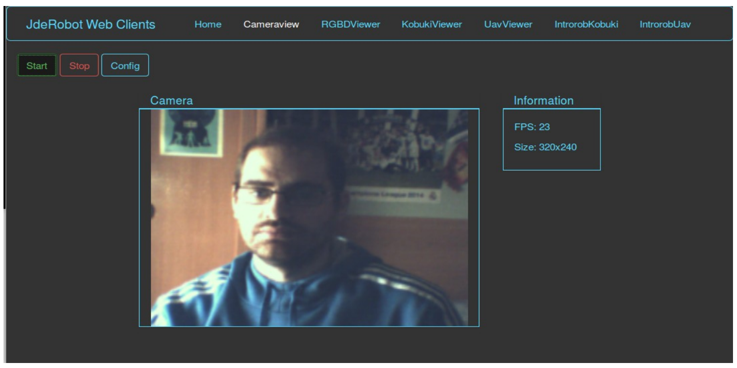
Fig. 1: Architecture of web tools in JdeRobot

As can be seen in Figure 1. The new release of ICE library supports WebSockets so the drivers have been used without source code modification, just a change in configuration to set WebSockets protocol instead of TCP or UDP.

Each client is divided into three distinct parts: (a) *Connectors*, where it internally creates a thread (WebWorker) that connects to the ICE server; (b) *Core*, that handles the inner workings of widgets; (c) *Graphic interface,* that organizes the appearance of the widgets within the webpage. In depth details and explanations can be found in (Martínez, 2016).

3.1 CameraViewJS

This client connects to the JdeRobot CameraServer which is the driver for real cameras. As can be seen in Figure 2, CameraViewJS shows the images in a HTML5 Canvas and the number of FPS (Frames per second) received.



FPS: 23
Size: 320x240

Fig. 2: Appearance of CameraViewJS webtool

It uses the API.Camera connector, which creates a WebWorker. The *connect* function establishes a JavaScript promise that is resolved when the WebWorker indicates that the connection has been established.

Like all the sensors, the WebWorker starts a stream with the JdeRobot ICE server, the driver. It constantly requests sensor data to the driver and passes them to the main thread. Figure 3 shows this dialogue.

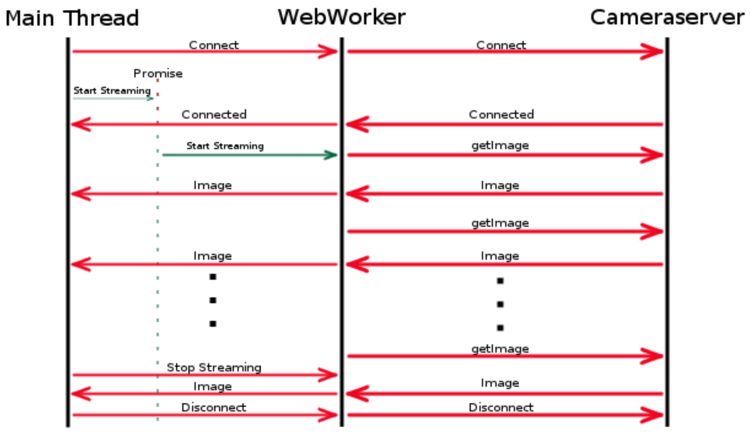


Fig. 3: Message exchange for CameraViewJS webtool

3.2 RGBDViewerJS

This web client connects to the JdeRobot server *Openni1Server* which provides information from RGBD sensors. As can be seen in Fig. 4, it shows the color and depth images in two HTML5 canvas and their FPS. A third canvas shows with WebGL the 3D scene, made of color points in 3D.

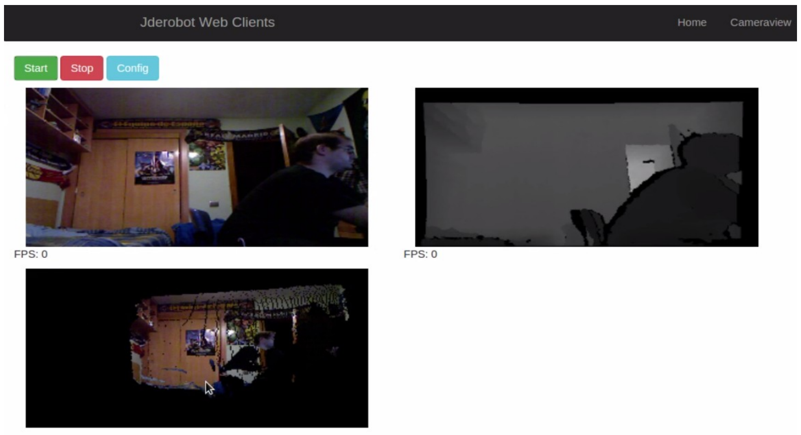


Fig. 4: Appearance of RGBDViewerJS webtool

RGBDViewerJS uses two API.Camera connectors, one for the RGB image and one for the distance.

To create the 3D scene, a 3D point is computed for each pixel in the depth image using the camera calibration parameters (pin hole model). The color is taken from the corresponding pixel in the RGB image, which is registered with the depth image.

3.3 KobukiViewerJS

This web client connects to JdeRobot *KobukiDriver* and is used to teleoperate a Kobuki wheeled robot from the browser. As seen in Figure 5, it shows the images from the robot stereo pair and the data from the onboard laser sensor. Another canvas displays the 3D scene using WebGL.

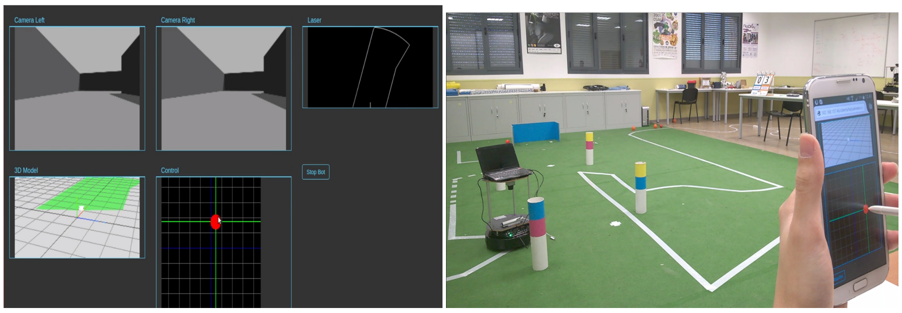


Fig 5: Appearance of KobukiViewerJS both with a simulated Kobuki robot and a real one

Regarding sensors, KobukiViewerJS uses two API.Camera connectors to connect with cameras, an API.Laser to get laser information and an API.Pose3D for robot position and orientation. Every time it receives laser data, the 2D-laser canvas and the 3D WebGl canvas are updated (green layer). When it receives a new Pose3D (encoders) data the orientation and position of the 3D model robot is updated too.

Regarding actuators, this webtool client has an API.Motors connector to send orders to the robot motors. When the control widget is moved then speed orders are sent to the robot. As seen in Figure 6 the exchange of messages for the actuators is in the opposite way to that of sensors.

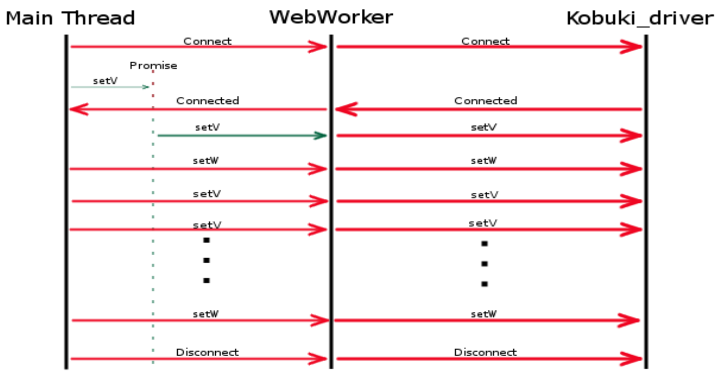


Fig 6: Message exchange for motors in KobukiViewer.JS

3.4 UAVViewJS

This client connects to JdeRobot *ArDroneServer* and allows to teleoperate a Parrot ArDrone quadrotor. As shown in Figure 5, it shows the drone on board camera images. 3D scene is shown in other canvas. In addition it provides two touch joysticks to control the drone movement. It also has flight indicators and buttons for takeoff, landing and camera change.

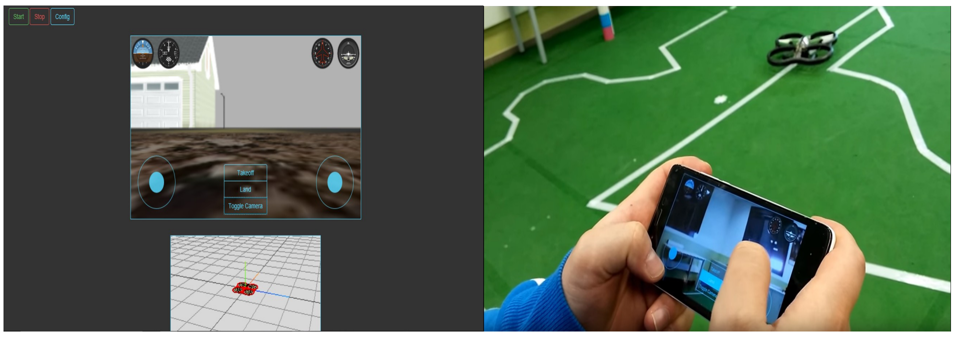


Fig 7: UAVViewerJS with a simulated drone a with a real one from a smartphone

UavViewerJS has API.Camera and API.Pose3D connectors to communicate with the drone driver for sensors. For actuators it uses API.ArdroneExtra and API.CMDVel to send motion commands.

5 Experiments

Several experiments have been performed to test the web tools functionality, as was shown in the previous images. In addition, we measured the framerate for the CameraViewJS web tool in different network scenarios. In good connectivity settings the web tools performs similar to the local desktop tool, about 8 fps. In poor connectivity scenarios the performance slows down to 4, 2 or 1 fps, depending on the bandwidth limitations.

6 Conclusions

Four webtools have been created in JdeRobot that allow multiplaform tools to teleoperate wheeled robots or drones and to monitor cameras and RGBD sensors from any browser, including those in smartphones. They use advanced web technologies like WebSockets, WebGL and WebWorkers. They have been included in the last official JdeRobot release.

Acknowledgements

This research has received funding from the RoboCity2030-III-CM project (Robótica aplicada a la mejora de la calidad de vida de los ciudadanos. Fase III; S2013/MIT-2748), funded by Programas de Actividades I+D en la Comunidad de Madrid and cofunded by Structural Funds of the EU.

References

OSRF, Gazebo simulator web page, 2016 http://gazebosim.org/

JdeRobot web page, 2016 http://jderobot.org

Martínez, A., Final Degree Project “Tecnologías web en la plataforma robótica JdeRobot”. Esc.Téc.Sup.Ing. Telecomunicación, URJC, 2016.

ZeroC ICE middleware, 2016 https://doc.zeroc.com/display/Ice35/Home