

Chapter 6

ROS Tools

Apart from the commands introduced in Chapter 5, there are various tools that can help us when using the ROS. We should note these GUI tools as complementary to the command line tools. There are quite a number of ROS tools, including the tools that ROS users have personally released as well. Among these tools, the ones we will discuss in this chapter do not directly process a function in the ROS, but they are greatly useful supplementary tools for programming with ROS.

The tools that we will cover in this chapter are as follows.

- **RViz** 3D visualization tool
- **rqt** Qt-based ROS GUI development tool
- **rqt_image_view** Image display tool (a type of rqt)
- **rqt_graph** A tool that visualizes the correlation between nodes and messages as a graph (a type of rqt)
- **rqt_plot** 2D data plot tool (a type of rqt)
- **rqt_bag** GUI-based bag data analysis tool (a type of rqt)

6.1. 3D Visualization Tool (RViz)

RViz¹ is the 3D visualization tool of ROS. The main purpose is to show ROS messages in 3D, allowing us to visually verify data. For example, it can visualize the distance from the sensor of a Laser Distance Sensor (LDS) to an obstacle, the Point Cloud Data (PCD) of the 3D distance sensor such as RealSense, Kinect, or Xtion, the image value obtained from a camera, and many more without having to separately develop the software.



FIGURE 6-1 Loading screen of RViz, the 3D visualization tool of ROS

¹ <http://wiki.ros.org/rviz>

It also supports various visualization using user specified polygons, and Interactive Markers² allow users to perform interactive movements with commands and data received from the user node. In addition, ROS describes robots in Unified Robot Description Format (URDF)³, which is expressed as a 3D model for which each model can be moved or operated according to their corresponding degree of freedom, so they can be used for simulation or control. The mobile robot model can be displayed, and received distance data from the Laser Distance Sensor (LDS) can be used for navigation as shown in Figure 6-2. RViz can also display the image from the camera mounted on the robot as shown in the lower-left corner of Figure 6-2. In addition to this, it can receive data from various sensors such as Kinect, LDS, RealSense and visualize them in 3D as shown in Images 6-3, 6-4, and 6-5.

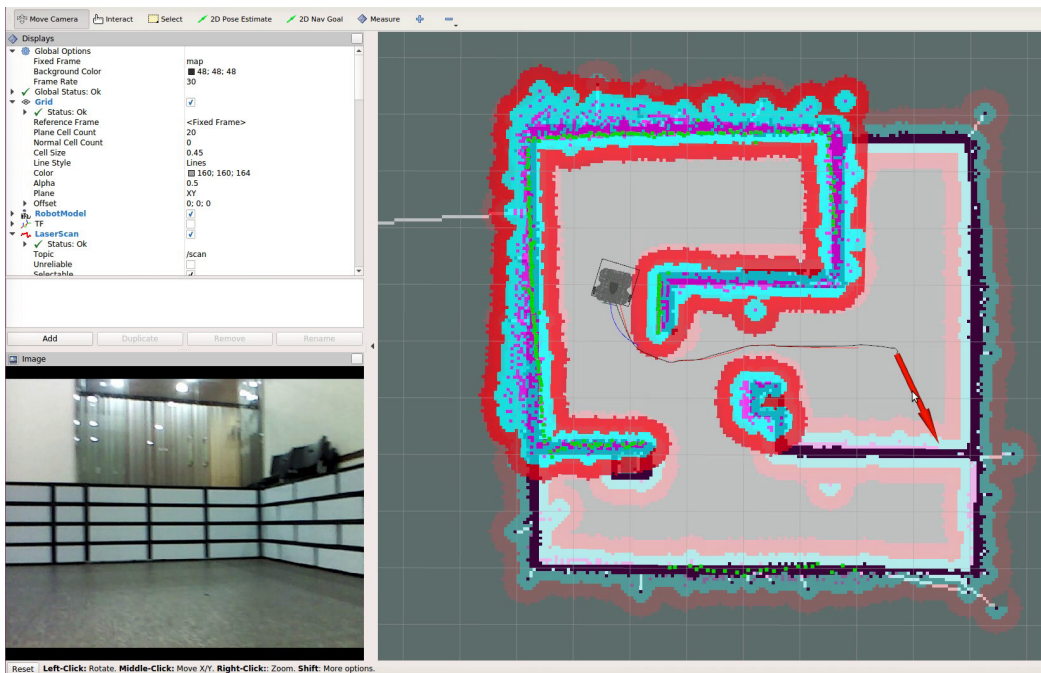


FIGURE 6-2 RViz example 1: Navigation using TurtleBot3 and LDS sensor

² <http://wiki.ros.org/rviz/Tutorials/Interactive%20Markers%3A%20Getting%20Started>

³ <http://wiki.ros.org/urdf>

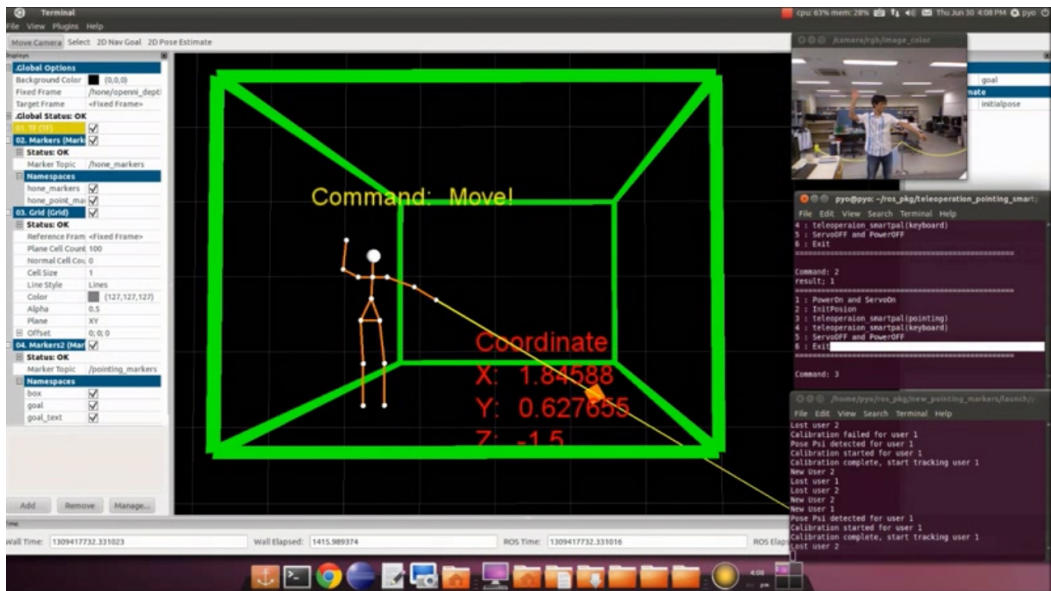


FIGURE 6-3 RViz example 2: Obtain the skeleton of a person using Kinect and Command with a Motion

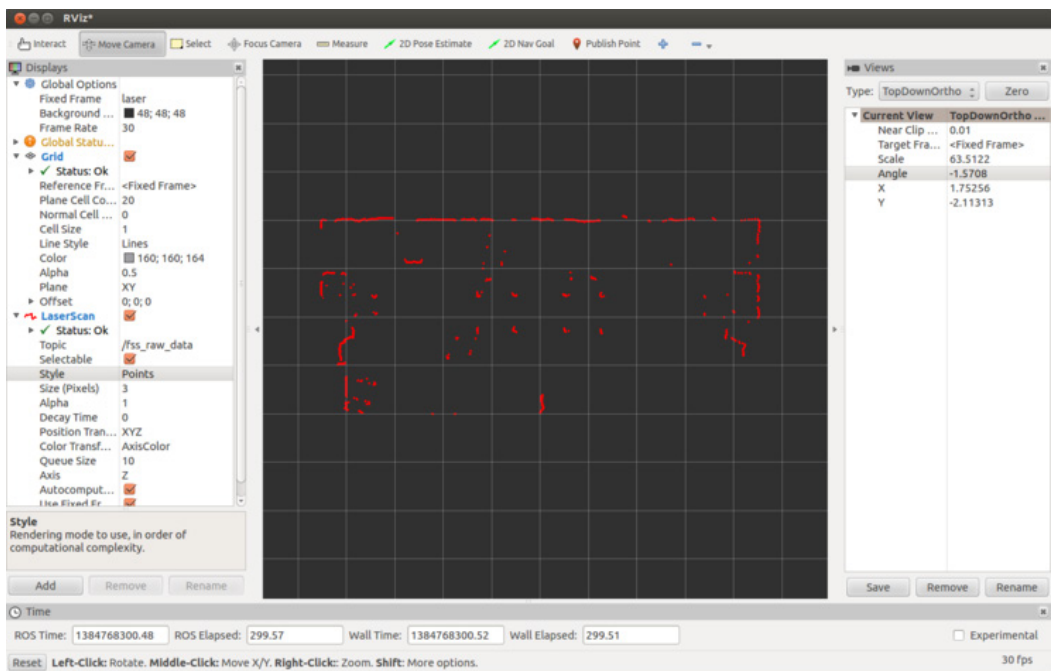


FIGURE 6-4 RViz example 3: Measuring distance using LDS

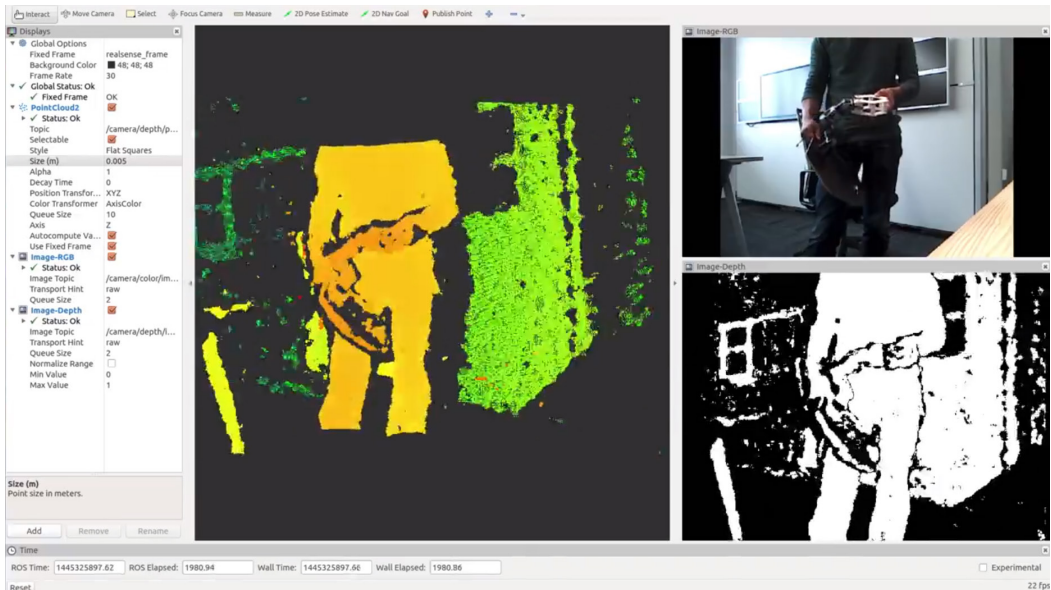


FIGURE 6-5 RViz example 4: distance, infrared, color image value obtained from Intel RealSense

6.1.1. Installing and Running RViz

If you have installed ROS with ‘ros-[ROS_DISTRO]-desktop-full’ command, RViz should be installed by default. If you did not install the ‘desktop-full’ version ROS or for some reason, RViz is not installed, use the following command to install RViz.

```
$ sudo apt-get install ros-kinetic-rviz
```

The execution command of RViz is as follows. However, just as for any other ROS tool, roscore must be running. For your reference, you can also run it with the node running command ‘roslaunch rviz rviz’.

```
$ rviz
```

6.1.2. RViz Screen Components

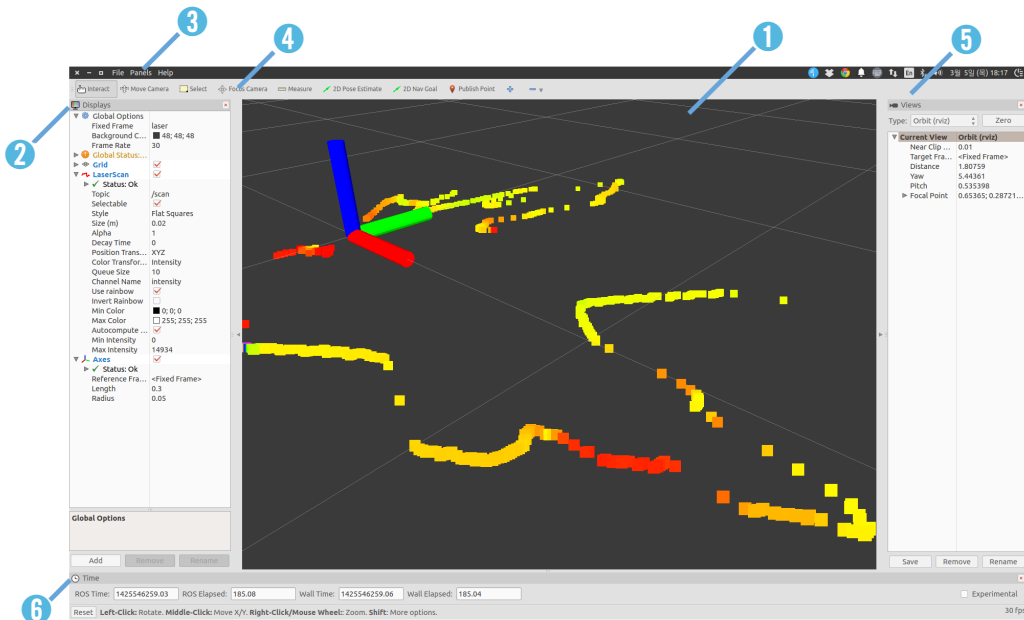


FIGURE 6-6 Composition of the RViz screen

- 1 **3D View:** This black area is located in the middle of the screen. It is the main screen which allows us to see various data in 3D. Options such as background color of the 3D view, fixed frame, and grid can be configured in the Global Options and Grid settings on the left column of the screen.
- 2 **Displays:** The Displays panel on the left column is for selecting the data that we want to display from the various topics. If we click [Add] button on the lower left corner of the panel, the display⁴ selection screen will appear as shown in Figure 6-7. Currently, there are about 30 different types of displays we can choose from, which we will explore more in the following section.
- 3 **Menu:** The Menu bar is located on the top of the screen. We can select commands to save or load the current display settings, and also can select various panels.
- 4 **Tools:** Tools are located below the menu bar, where we can select buttons for various functions such as interact, camera movement, selection, camera focus change, distance measurement, 2D position estimation, 2D navigation target-point, publish point.
- 5 **Views:** The Views panel configures the viewpoint of the 3D view.

⁴ <http://wiki.ros.org/rviz/DisplayTypes>

- **Orbit:** The specified point is called the focus and the orbit rotates around this point. This is the default value and the most commonly used view.
 - **FPS(first-person):** This displays in a first-person viewpoint.
 - **ThirdPersonFollower:** This displays in a third-person viewpoint that follows a specific target.
 - **TopDownOrtho:** This uses the Z-axis as the basis, and displays an orthographic projection of objects on the XY plane.
 - **XYOrbit:** This is similar to the default setting which is the Orbit, but the focus is fixed on the XY plane, where the value of Z-axis coordinate is fixed to zero.
- 6 **Time:** Time shows the current time (wall time), ROS Time, and the elapsed time for each of them. This is mainly used in simulations, and if there is a need to restart it, simply click the [Reset] button at the very bottom.

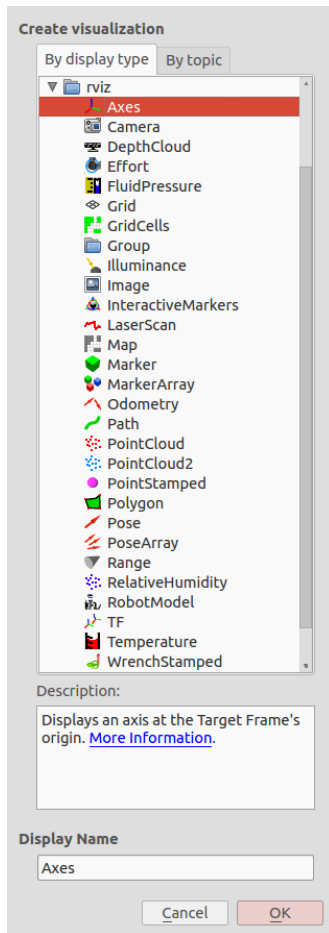

















FIGURE 6-7 RViz display select screen

6.1.3. RViz Displays

The most frequently used menu when using RViz will probably be the Displays⁵ menu. This Displays menu is used to select the message to display on the 3D View panel, and descriptions on each item are explained in Table 6-1.

Icon	Name	Description
	Axes	Displays the xyz axes.
	Camera	Creates a new rendering window from the camera perspective and overlays an image on top of it.
	DepthCloud	Displays a point cloud based on the Depth Map. It displays distance values acquired from sensors such as Kinect and Xtion with DepthMap and ColorImage topics as points with overlaid color obtained from the camera.
	Effort	Displays the force applied to each rotary joint of the robot.
	FluidPressure	Displays the pressure of fluids, such as air or water.
	Grid	Displays 2D or 3D grids.
	Grid Cells	Displays each cells of the grid. It is mainly used to display obstacles in the costmap of the navigation
	Group	This is a container for grouping displays. This allows us to manage the displays being used as one group.
	Illuminance	Displays the illuminance.
	Image	Displays the image in a new rendering window. Unlike the Camera display, it does not overlay the camera
	InteractiveMarkers	Displays Interactive Markers. We can change the position (x, y, z) and rotation (roll, pitch, yaw) with the mouse.
	LaserScan	Displays the laser scan value.
	Map	Displays the occupancy map, used in navigation, on top of the ground plane.
	Marker	Displays markers such as arrows, circles, triangles, rectangles, and cylinders provided by RViz.
	MarkerArray	Displays multiple markers.

⁵ <http://wiki.ros.org/rviz/DisplayTypes>















Icon	Name	Description
	Odometry	Displays the odometry information in relation to the passage of time in the form of arrows. For example, as the robot moves, arrow markers are displayed showing the traveled path in a connected form according to the time intervals.
	Path	Displays the path of the robot used in navigation.
	Point Cloud	Displays point cloud data. This is used to display sensor data from depth cameras such as RealSense, Kinect, Xtion, etc. Since PointCloud2 is compatible with the latest Point Cloud Library (PCL), we can generally use PointCloud2.
	Point Cloud2	
	PointStamped	Displays a rounded point.
	Polygon	Displays a polygon outline. It is mainly used to simply display the outline of a robot on the 2D plane.
	Pose	Displays the pose (location + orientation) on 3D. The pose is represented in the shape of an arrow where the origin of the arrow is the position(x, y, z,) and the direction of the arrow is the orientation (roll, pitch, yaw). For instance, pose can be represented with the position and orientation of the 3D robot model, while it can be represented with the goal point.
	Pose Array	Displays multiple poses.
	Range	This is used to visualize the measured range of a distance sensor such as an ultrasonic sensor or an infrared sensor in the form of a cone.
	RelativeHumidity	Displays the relative humidity.
	RobotModel	Displays the robot model.
	TF	Displays the coordinate transformation TF used in ROS. It is displayed with the xyz axes much like the previously mentioned axes, but each axis expresses the hierarchy with an arrow according to the relative coordinates.
	Temperature	Displays the temperature.
	WrenchStamped	Displays the wrench, which is the torsion movement, in the form of 'arrow' (force) and 'arrow+circle' (torque).

TABLE 6-1 Rviz Displays Panel

6.2. ROS GUI Development Tool (rqt)

Besides the 3D visualization tool RViz, ROS provides various GUI tools for robot development. For example, there is a graphical tool that shows the hierarchy of each node as a diagram thereby showing the status of the current node and topic, and a plot tool that schematizes a message as a 2D graph. Starting from the ROS Fuerte version, more than 30 GUI development tools have been integrated as the tool called rqt⁶ which can be used as a comprehensive GUI tool. Furthermore, RViz has also been integrated as a plugin of rqt, making rqt an essential GUI tool for ROS.

Not only that, but as the name suggests, rqt was developed based on Qt, which is a cross-platform framework widely used for GUI programming, making it very convenient for users to freely develop and add plugins. In this section we will learn about the ‘rqt’ plugins ‘rqt_image_view’, ‘rqt_graph’, ‘rqt_plot’ and ‘rqt_bag’.

6.2.1. Installing and Running rqt

If you have installed ROS with ‘ros-[ROS_DISTRO]-desktop-full’ command rqt will be installed by default. If you did not install the ‘desktop-full’ version of ROS or for some reason, ‘rqt’ is not installed, then the following command will install ‘rqt’.

```
$ sudo apt-get install ros-kinetic-rqt*
```

The command to run rqt is as follows. You can simply type in ‘rqt’ on the terminal. For your reference, we can also run it with the node execution command ‘roslaunch rqt_gui rqt_gui’.

```
$ rqt
```

If we run ‘rqt’ then the GUI screen of rqt will appear as shown in Figure 6-8. If it is the first time being launched, it will display only the menu without any content below. This is because the plugin, which is the program that is directly run by ‘rqt’, has not been specified.

⁶ <http://wiki.ros.org/rqt>

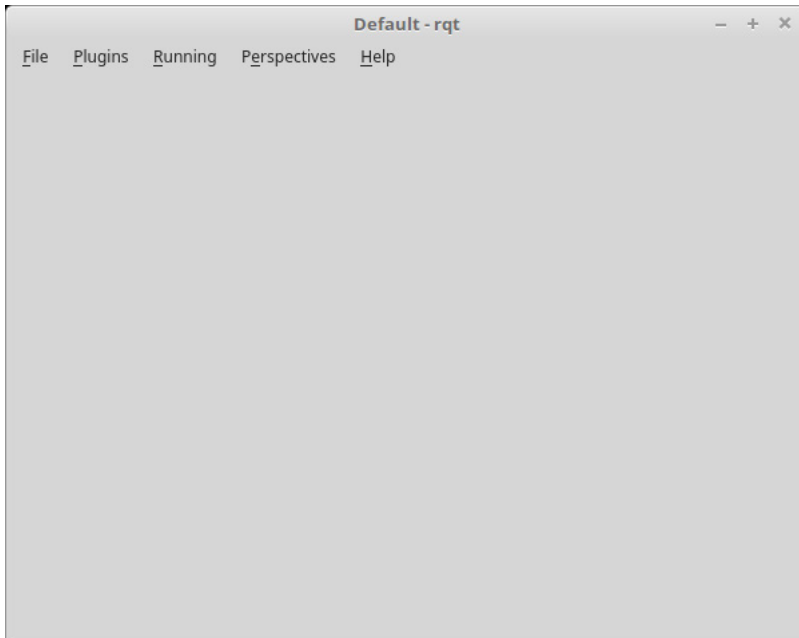


FIGURE 6-8 Initial screen of rqt

The rqt menus are as follows.

- **File** The File menu only contains the sub-menu to close ‘rqt’.
- **Plugins** There are over 30 plugins. Select the plugin to use.
- **Running** The currently running plugins are shown and they can be stopped when they are not needed.
- **Perspectives** This menu saves operating plugins as a set and uses them later to run the same plugins.

6.2.2. rqt Plugins

From the ‘rqt’ menu on the top, if we select [Plugins⁷ ⁸] we can see about 30 plugins. These plugins have the following roles. Most of them are default plugins of ‘rqt’ that have very useful features. Unofficial plugins can also be added, and if necessary, we can add custom ‘rqt’ plugins that we developed for ourselves as well.

⁷ <http://wiki.ros.org/rqt/Plugins>

⁸ http://wiki.ros.org/rqt_common_plugins

Action

- **Action Type Browser:** This is a plugin to check the data structure of an action type.

Configuration

- **Dynamic Reconfigure:** This is a plugin to modify the parameter value of a node.
- **Launch** This is a GUI plugin of roslaunch, which is useful when we cannot remember the name or composition of roslaunch.

Introspection

- **Node Graph:** This is a plugin for the graphical view that allows us to check the relationship diagram of the currently running nodes or message flows.
- **Package Graph:** This is a plugin for the graphical view that displays the dependencies of the packages.
- **Process Monitor:** We can check the PID (Processor ID), CPU usage, memory usage, and number of threads of the currently running nodes.

Logging

- **Bag:** This is a plugin regarding the ROS data logging.
- **Console:** This is a plugin to check the warning and error messages occurring in the nodes in one screen.
- **Logger Level:** This is a tool to select a logger, which is responsible for publishing the logs, and set the logger level⁹ to publish a specific log such as 'Debug', 'Info', 'Warn', 'Error', and 'Fatal'. It is very convenient if 'Debug' is selected while debugging process.

Miscellaneous Tools

- **Python Console:** This is a plugin for the Python console screen.
- **Shell:** This is a plugin that launches a shell.
- **Web:** This is a plugin that launches a web browser.

⁹ <http://wiki.ros.org/roscpp/Overview/Logging>

Robot Tools

- **Controller Manager:** This is a plugin to check the status, type, hardware interface information of the robot controller.
- **Diagnostic Viewer:** This is a plugin to check the robot status and error.
- **MoveIt! Monitor:** This is a plugin to check the MoveIt! data that is used for motion planning.
- **Robot Steering:** This is a GUI tool for robot manual control, and this GUI tool is useful for controlling a robot in remote.
- **Runtime Monitor:** This is a plugin to check the warnings or errors of the nodes in real-time.

Services

- **Service Caller:** This is a GUI plugin that connects to a running service server and requests a service. This is useful for testing service.
- **Service Type Browser:** This is a plugin to check the data structure of a service type.

Topics

- **Easy Message Publisher:** This is a plugin that can publish a topic in a GUI environment.
- **Topic Publisher:** This is a GUI plugin that can publish a topic. This is useful for topic testing.
- **Topic Type Browser:** This is a plugin that can check the data structure of a topic. This is useful for checking the topic type.
- **Topic Monitor:** This is a plugin that lists the currently used topics, and checks the information of the selected topic from the list.

Visualization

- **Image View:** This is a plugin that can check the image data from a camera. This is useful for simple camera data testing.
- **Navigation Viewer:** This is a plugin to check the position or goal point of the robot in the navigation.
- **Plot:** This is a GUI plugin for plotting 2D data. This is useful for schematizing 2D data.
- **Pose View:** This is a plugin for displaying the pose (position+orientation) of a robot model or TF.
- **RViz:** This is the RViz plugin which is a tool for 3D visualization.
- **TF Tree:** This is a plugin of a graphical view type that shows the relationship of each coordinates acquired from the TF in the tree structure.

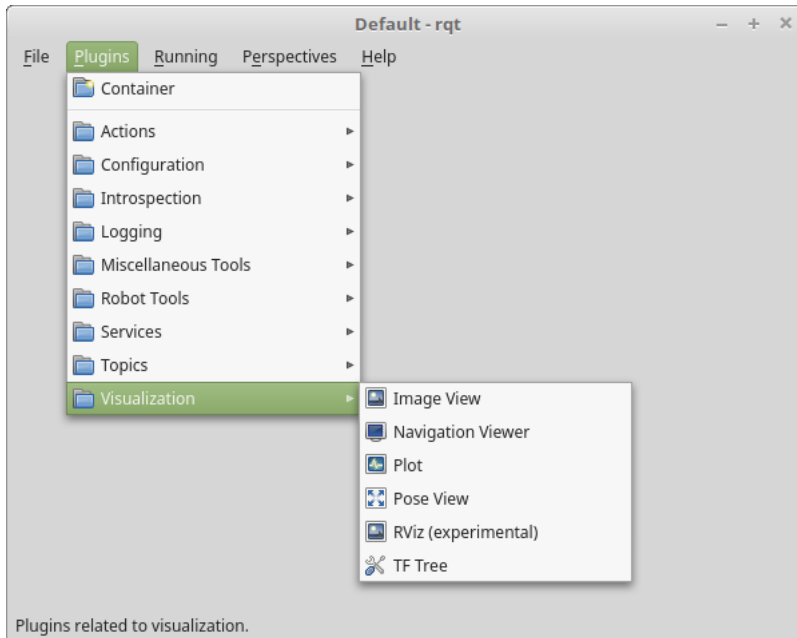


FIGURE 6-9 rqt Plugin

Since it is difficult to introduce all of the plugins, in this chapter we will learn about the ones that are most frequently used, which are 'rqt_image_view', 'rqt_bag', 'rqt_graph' and 'rqt_plot'.

6.2.3. rqt_image_view

This is a plugin¹⁰ to display the image data of a camera. Although it is not an image processing process, it is still quite useful for simply checking an image. A USB camera generally supports UVC, so we can use the 'uvc_camera' package of ROS. First, install the 'uvc_camera' package using the following command.

```
$ sudo apt-get install ros-kinetic-uvc-camera
```

Connect the USB camera to the USB port of the PC, and launch the 'uvc_camera_node' in the 'uvc_camera' package using the following command.

```
$ roslaunch uvc_camera uvc_camera_node
```

¹⁰ http://wiki.ros.org/rqt_image_view

Once installation is completed, run 'rqt' with the 'rqt' command, and go to the menu and select [Plugins] → [Image View]. In the message selection field located on the upper left side, select '/image_raw' to see the image as shown in the Figure 6-10. More information about the camera sensor will be provided in Section 8.3.

```
$ rqt
```

Apart from selecting the plugin from the rqt menu, we can also use dedicated execution command as below.

```
$ rqt_image_view
```

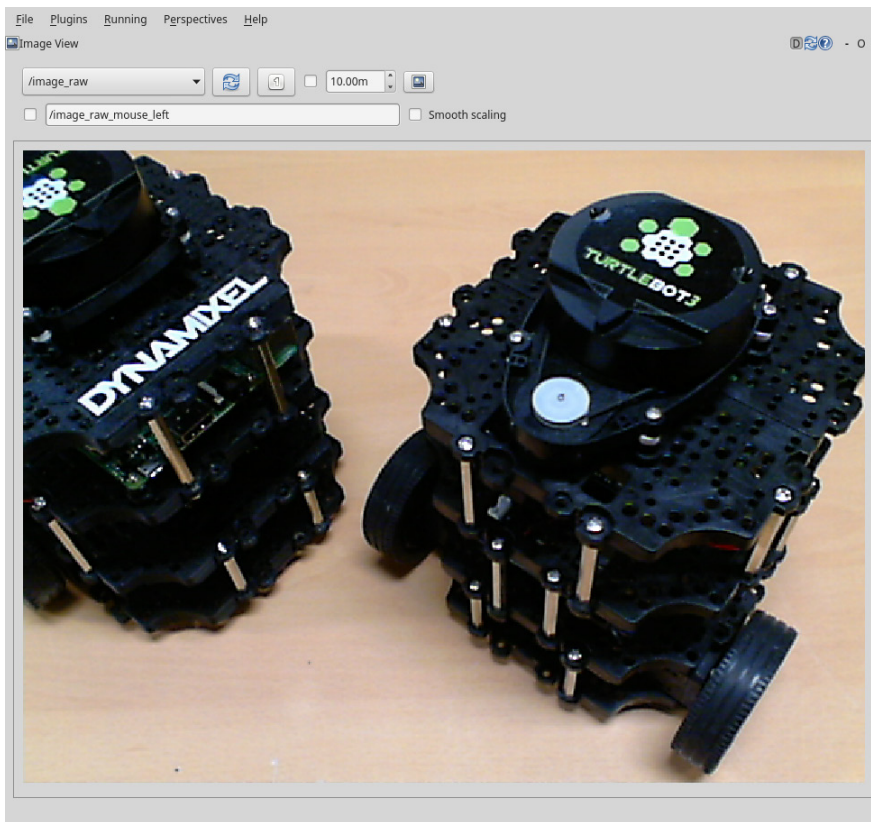


FIGURE 6-10 Checking the image data of the USB camera as an image view

6.2.4. rqt_graph

The ‘rqt_graph’¹¹ is a tool that shows the correlation among active nodes and messages being transmitted on the ROS network as a diagram. This is very useful for understanding the current structure of the ROS network. The instruction is very simple. As an example, for the purpose of checking the nodes, let’s run ‘turtlesim_node’ and ‘turtle_teleop_key’ in the ‘turtlesim’ package described in Section 3.3, and the ‘uvc_camera_node’ in the ‘uvc_camera’ package described in Section 6.2.3. Each node should be executed in a separate terminal.

```
$ roslaunch turtlesim turtlesim_node
$ roslaunch turtlesim turtle_teleop_key
$ roslaunch uvc_camera uvc_camera_node
$ roslaunch image_view image_view image:=image_raw
```

After executing all nodes, launch ‘rqt’ with the ‘rqt’ command, and go to the menu to select [Plugins] → [Node Graph]. For your information, we can also run it with ‘rqt_graph’ without having to manually select the plugin from the menu.

The correlation among nodes and topics when ‘rqt_graph’ is running is shown as Figure 6-11.

```
$ rqt
```

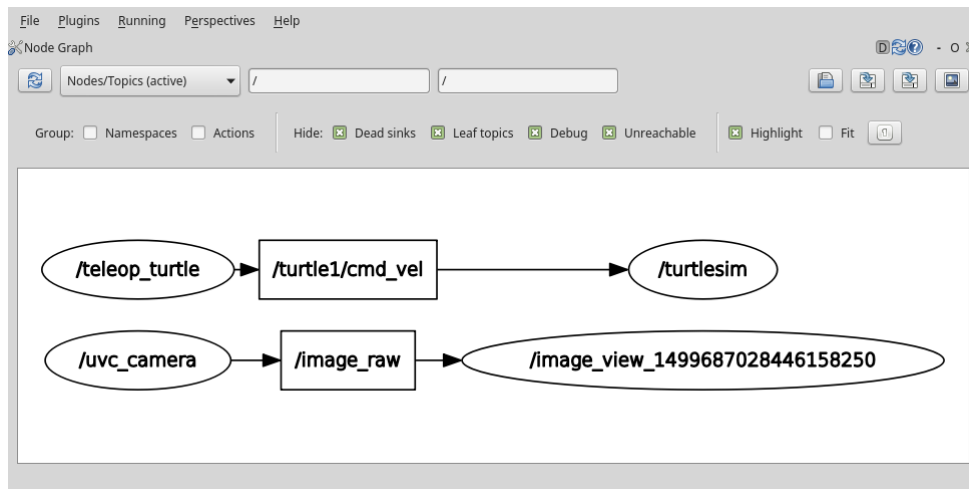


FIGURE 6-11 Example of rqt_graph

¹¹ http://wiki.ros.org/rqt_graph

In Figure 6-11, circles represent nodes (/teleop_turtle, /turtlesim) and squares (/turtle1/cmd_vel, /image_raw) represent topic messages. The arrow indicates the transmission of the message. In the previous example when we executed 'turtle_teleop_key' and 'turtlesim_node', the 'teleop_turtle' node and the 'turtlesim' node were running respectively. We can verify that these two nodes are transmitting data with the arrow key values of the keyboard in the form of translational speed and rotational speed message (topic name: /turtle1/cmd_vel).

We can also verify that the 'uvc_camera' node in the 'uvc_camera' package is publishing the '/image_raw' topic message and the 'image_view_XXX' node is subscribing it. Unlike this simple example, the actual ROS programming consists of tens of nodes that transmit various topic messages. In this situation, 'rqt_graph' becomes very useful for checking the correlation of nodes on the ROS network.

6.2.5. rqt_plot

This time let's run 'rqt_plot' with the following command instead of selecting the plugin from the rqt menu. For your reference, we can run it with the node execution command 'roslaunch rqt_plot rqt_plot'.

```
$ rqt_plot
```

Once 'rqt_plot' is up and running, click the gear shaped icon on the top right corner of the program. We can select the option as shown in Figure 6-12, where the default setting is 'MatPlot'. Apart from MatPlot we can also use PyQtGraph and QwtPlot, so refer to the corresponding installation method and use the graph library of your choice.

For example, in order to use PyQtGraph as the default plot instead of MatPlot, download and install the latest 'python-pyqtgraph_0.9.xx-x_all.deb' file from the download address below. If installation is completed, the PyQtGraph item will be enabled and you will be able to use PyQtGraph.

- <http://www.pyqtgraph.org/downloads/>

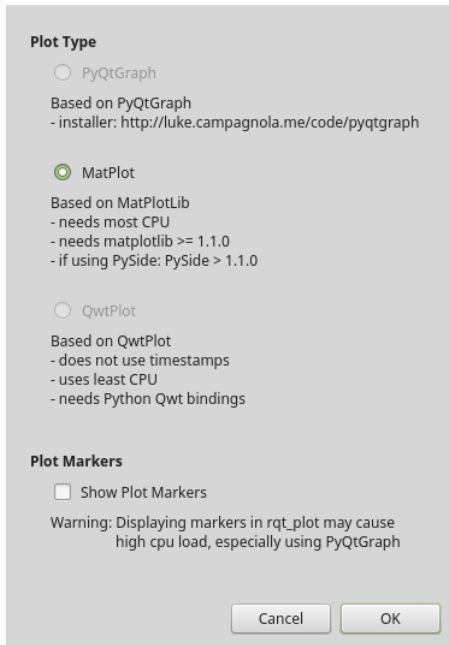


FIGURE 6-12 Install option of `rqt_plot`

The ‘`rqt_plot`’¹² is a tool for plotting 2D data. Plot tool receives ROS messages and displays them on the 2D coordinates. As an example, let us plot the x and y coordinates of the ‘`turtlesim`’ node pose message. First we need to launch ‘`turtlesim_node`’ of the `turtlesim` package.

```
$ rosrun turtlesim turtlesim_node
```

Enter ‘`/turtle1/pose/`’ in the Topic field on the top of ‘`rqt_plot`’ tool, and it will draw the ‘`/turtle1/pose/`’ topic on the 2D (x-axis: time, y-axis: data value) plane. Alternatively, we can run it with the following command by specifying the topic to be schematized.

```
$ rqt_plot /turtle1/pose/
```

Then launch ‘`turtle_teleop_key`’ in the ‘`turtlesim`’ package so that we can move around the turtle on the screen.

```
$ rosrun turtlesim turtle_teleop_key
```

¹² http://wiki.ros.org/rqt_plot

As shown in Figure 6-13, we can check that the x, y position, direction in theta, translational speed, and rotational speed of the turtle are plotted. As we can see, this is a useful tool for displaying the coordinates of 2D data. In this example, we have used turtlesim, but it is also useful for displaying 2D data of nodes developed by users as well. It is particularly suitable for displaying the sensor value over a period of time, such as speed and acceleration.

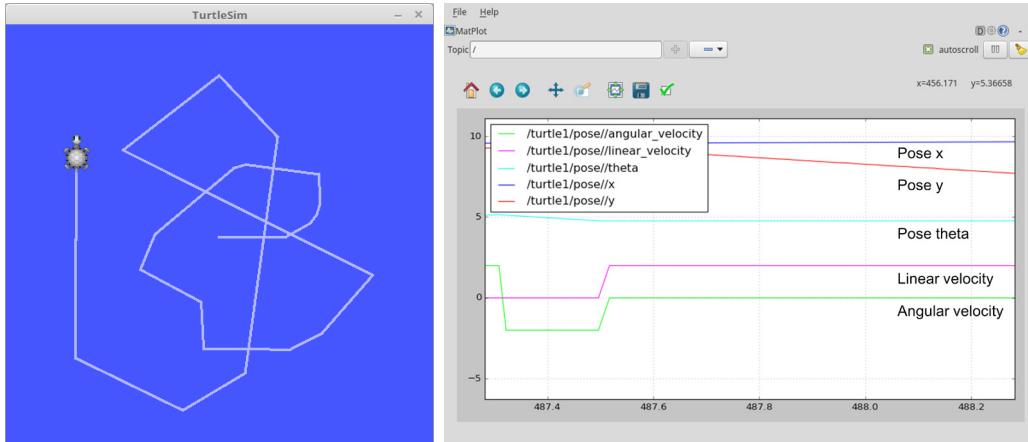


FIGURE 6-13 Example of rqt_plot

6.2.6. rqt_bag

The 'rqt_bag' is a GUI tool for visualizing a message. The 'rosbag' that we covered in 'Section 5.4.8 rosbag: ROS Log Information' was text-based tool, but 'rqt_bag' has a visualization function added which allows us to see the image of the camera right away, making it very useful for managing image data messages. Before we begin practice, we must run all of the 'turtlesim' and 'uvc_camera' related nodes covered in the 'rqt_image_view' and 'rqt_graph' tool. Then we create a bag file with the '/image_raw' message of the camera and the '/turtle1/cmd_vel' message of 'turtlesim' using the following command.

In Section 5.4 we have used the 'rosbag' program to record, play, and compress various topic messages of ROS as a bag file. The 'rqt_bag' is a GUI version of the previously introduced 'rosbag', and just like rosbag it can also record, play, and compress topic messages. In addition, since it is a GUI program, all commands are provided as buttons so they are easy to use, and we can also watch the camera images according to the change in time like the video editor.

As in the following example, in order to take advantage of the feature of 'rqt_bag', let us save the USB camera image as a bag file and then play it with 'rqt_bag'.

```
$ rosrunc uvc_camera uvc_camera_node
$ rosbag record /image_raw
$ rqt
```

Launch 'rqt' with the 'rqt' command, and go to the menu and select [Plugins] → [Logging] → [Bag]. Then select the folder-shaped Load Bag icon on the top left side and load the '*.bag' file that we just recorded. Then we will be able to check the camera image according to the change in time as shown in Figure 6-14. We can also zoom in, play, and check the number of data over time, and with the right-click, 'Publish' option will appear which allows us to publish the message again.

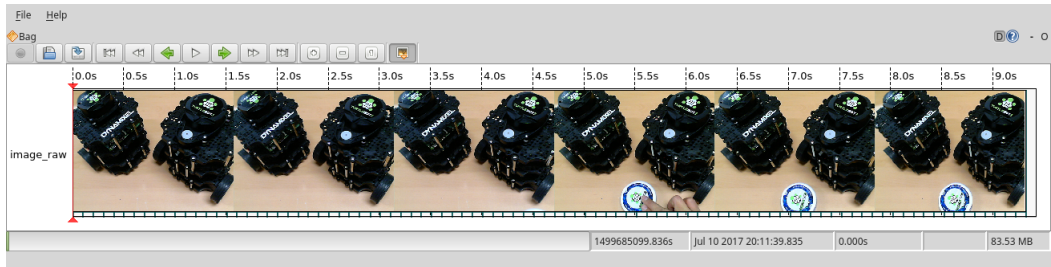


FIGURE 6-14 Example of rqt_bag

We have now completed the installations and instructions of the rqt tools. As we could not explain all of the plugins in this section, we encourage you to try using these tools for yourself referring to the few of the examples we have seen until now. Although these tools may not directly involved with robots or sensors as ROS nodes do, when we are performing these tasks they can be used as helpful supplementary tools for saving, preserving, modifying, and analyzing data.