



Introduction to Robot Operating System

Michal Reinštein

Czech Technical University in Prague, Faculty of Electrical Engineering, Dept. Cybernetics, Center for Machine Perception

[http://cmp.felk.cvut.cz/~reinsmic/
reinstein.michal@fel.cvut.cz](http://cmp.felk.cvut.cz/~reinsmic/reinstein.michal@fel.cvut.cz)



RECOMMENDED LITERATURE & LINKS

- ROS <http://wiki.ros.org/>
- ROS Answers <http://answers.ros.org/questions/>
- Willow Garage <https://www.willowgarage.com/>
- Very good book <http://www.cse.sc.edu/~jokane/agitr/>

What is ROS?



*ROS is an **open-source, meta-operating system** for your robot. It provides the services you would expect from an operating system, including **hardware abstraction**, low-level **device control**, implementation of commonly-used functionality, **message-passing** between processes, and **package management**. It also provides tools and libraries for obtaining, **building, writing, and running** code across multiple computers.*

--- J. M. O' Kane, USC

>> ROS is a middleware ...

What is middleware?



- Middleware is software providing **services to applications** beyond those available from the operating system.
- Middleware **makes it easier** for software developers to perform communication and input/output.
- Most commonly used in the context of **distributed applications**.
- More specifically: **dash in “client-server”**.
- Also used in a sense of: a **software driver**, an abstraction layer hiding details of hardware and software from an application.

What is ROS?



- Stands for **Robot Operating System**, but runs on top of existing OS like (mostly) Linux (Ubuntu officially supported)
- Originates from **SAIL/Willow Garage**, now moved to **OSRF**
- Combines several features into a consistent project:
 - A **software distribution** with a dependency mechanism
 - An **integrated build system**
 - A **communication** middleware
 - Helper **tools**: data visualization, record/replay, etc.

Why ROS?



- Excellent **documentation** and **tutorials**
- Reliable software tools for **data management**
- Accepted **standardization** for releasing codes & datasets
- Reliable **communication** system
- Link with **OpenCV** (2D computer vision), **PCL** (3D computer vision), and many others ...

What is ROS good for?



Distributed computation

- robot systems rely on software that **spans many different processes** and runs **across several different computers**
- need for **communication and data exchange** between multiple processes that may or may not live on the same computer

What is ROS good for?



Software reuse

progress of robotics research >> growing collection of good algorithms for common tasks: *perception, navigation, motion planning, mapping, etc.*

- ROS's **standard packages** provide stable, debugged implementations of many **important robotics algorithms**.
- ROS's **message passing interface** is becoming a **standard** for robot software interoperability

>> **less time reinventing wheels**

What is ROS good for?

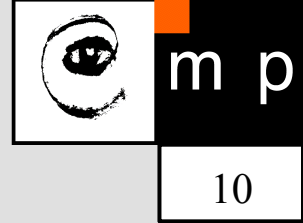


Rapid testing

Testing in robotics >> slow, time consuming, error-prone, availability of the physical robots ...

- **Separation** of the low-level direct control of the hardware and high-level processing and decision making
- Low-level programs (and their corresponding hardware) can easily be **replaced with a simulator** to test the behavior of the high-level part of the system
- Simple way to **record and play back** sensor data
- Less **time spent operating a physical robot**

What is ROS not?



- *ROS is not a programming language* - but works best with C++/Python, interface to Java (Android)
- *ROS is not (only) a library* - but includes client libraries (OpenCV, PCL, ActionLib, SMACH, MoveIt, GAZEBO, etc.)
- *ROS is not an integrated development environment* - but can be used with most popular IDEs (e.g. *Eclipse + pydev*)
- *ROS is not a real-time architecture*

ROS basics ...



- **Node:** A single process / computation unit (component of the system)
- **Message:** Data structure used by ROS nodes to communicate
- **Topic:** Broadcast communication between nodes (TCP/UDP)
- **Service:** Synchronous communication between nodes (RPC)
- **Roscore:** ROS (name) server for the whole system management

ROS node & ROS message



Node

- **Single process** that performs particular computation
- Communicate with each other by **passing messages**, through topic (**TCP, UDP**) or service (**RPC**).
- Connection between two nodes is accomplished using **topics** through the **roscore**, which acts as a name server.
- Each ROS node can be **written in different language**

Message

- Strictly typed **data structure**, support standard primitive types (*integer, float, Boolean, etc.*) and *arrays*.
- Messages can be composed of **arrays of messages**.
- Message type is defined in plain text using **Message Description Language** and code is generated for each supported language.

Topic

- A **named broadcast stream of messages**.
- Generally there can be more publishers of the same topic.
- Publishers are aware if someone is subscribed.
- Topic is defined by **name** and **message type**.

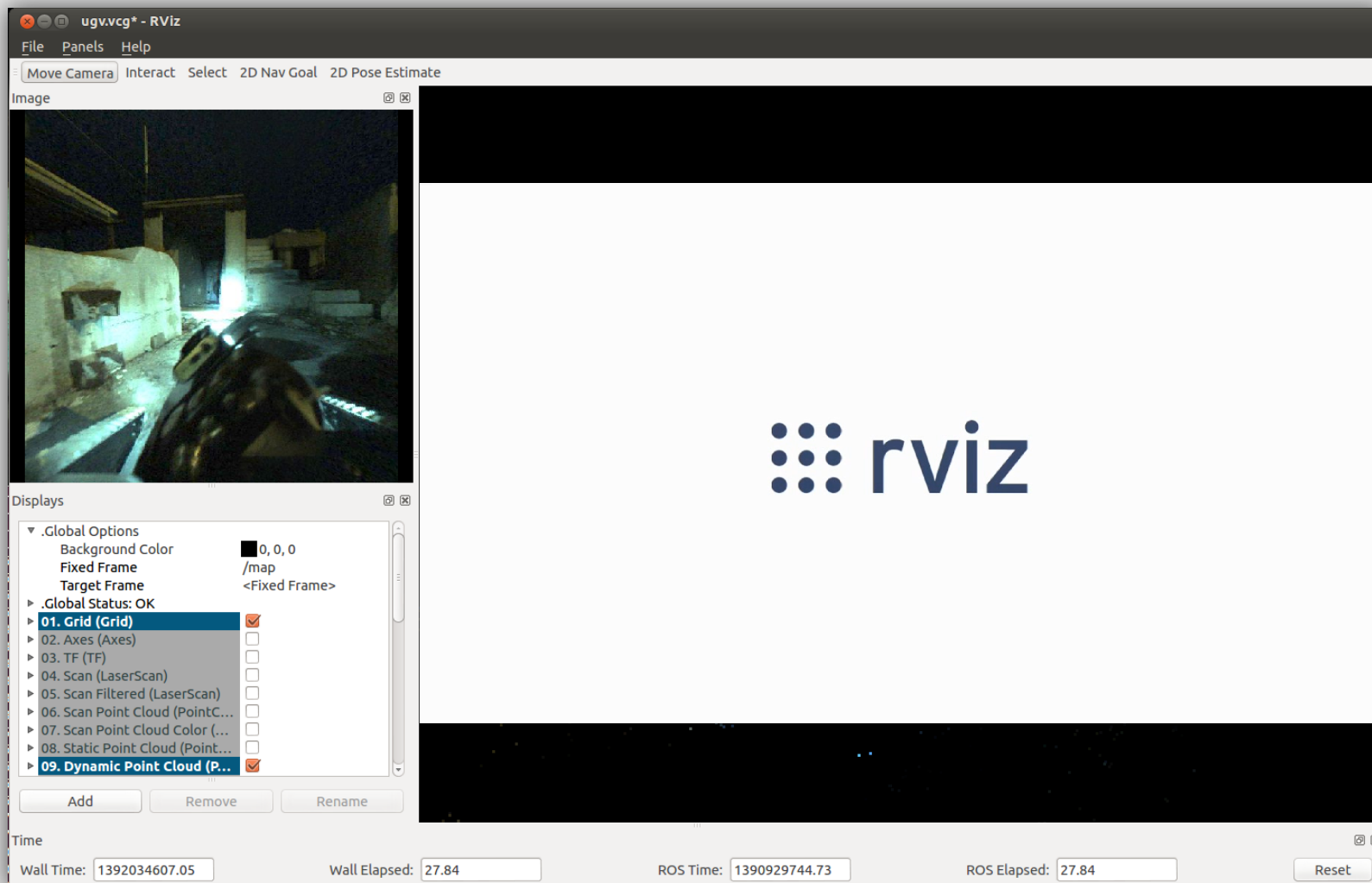
Service

- A **named synchronous communication**.
- There can be **just one node providing a service** of some name.
- Service is defined by name and two message types – *request* and *reply*.

ROS Tools: RVIZ



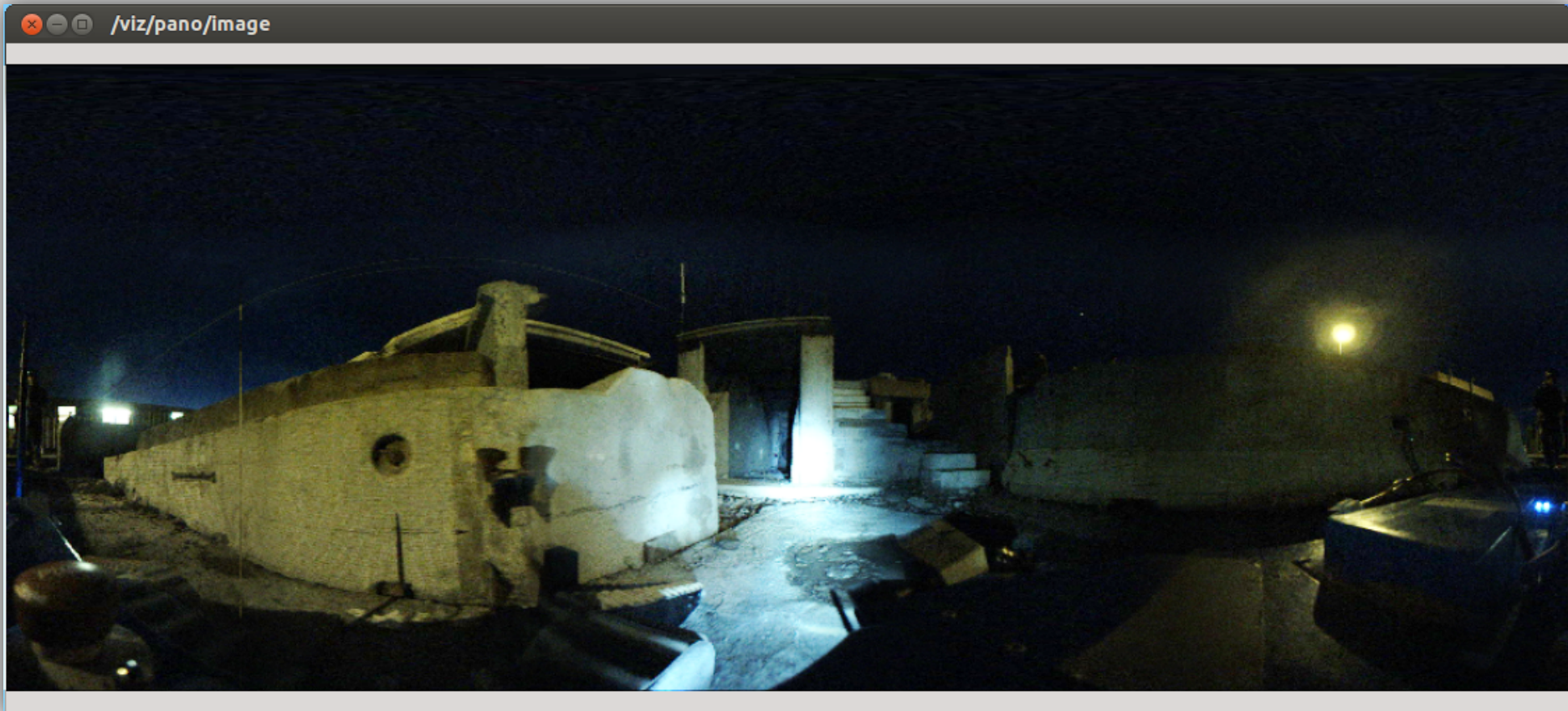
14



>> `roslaunch rviz rviz`

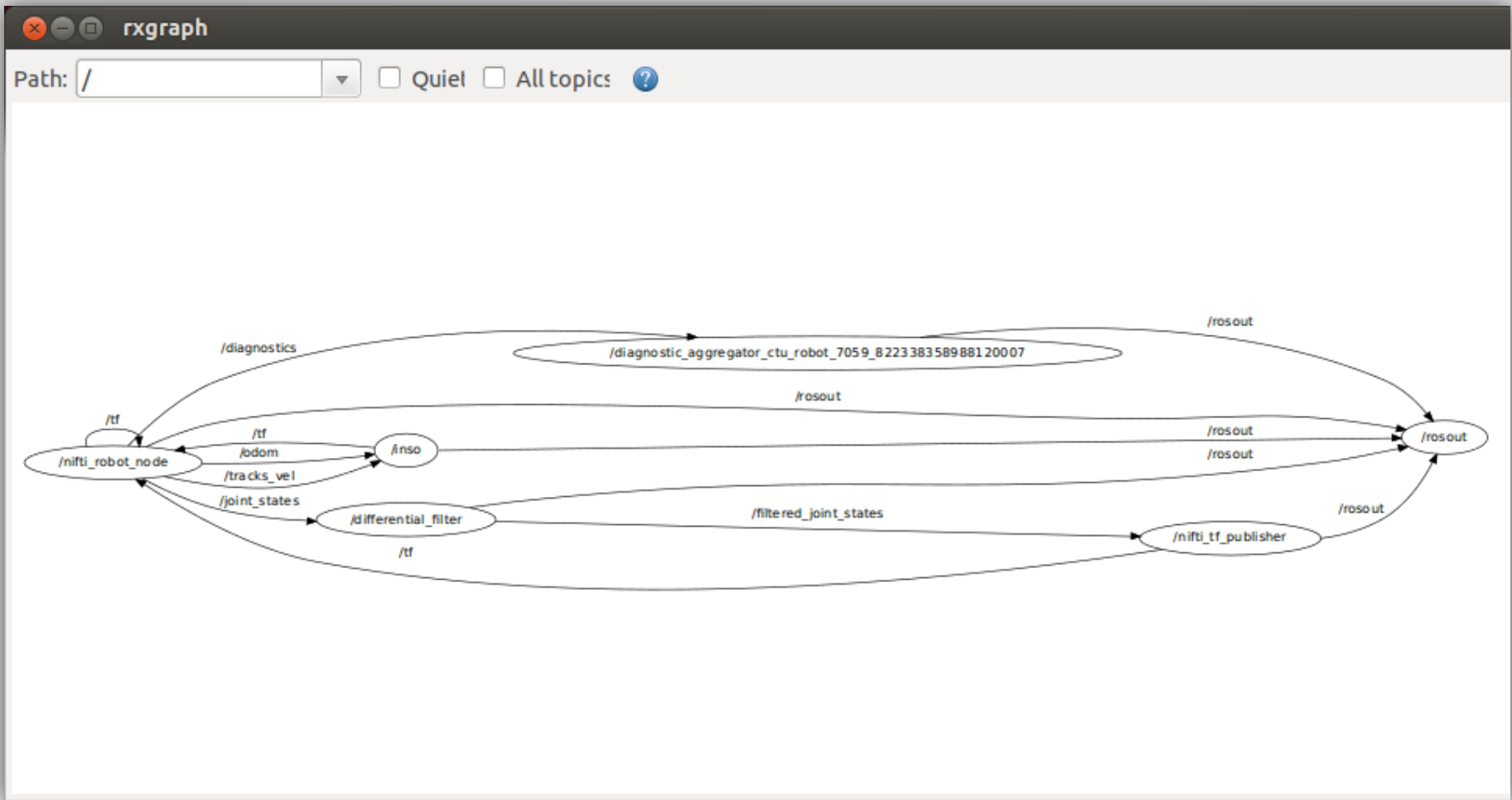
<http://wiki.ros.org/rviz/UserGuide>

ROS Tools: IMAGEVIEW



```
>> rosrun image_view image_view image:=<image topic> [image transport type]  
http://wiki.ros.org/image\_view
```

ROS Tools: RXGRAPH



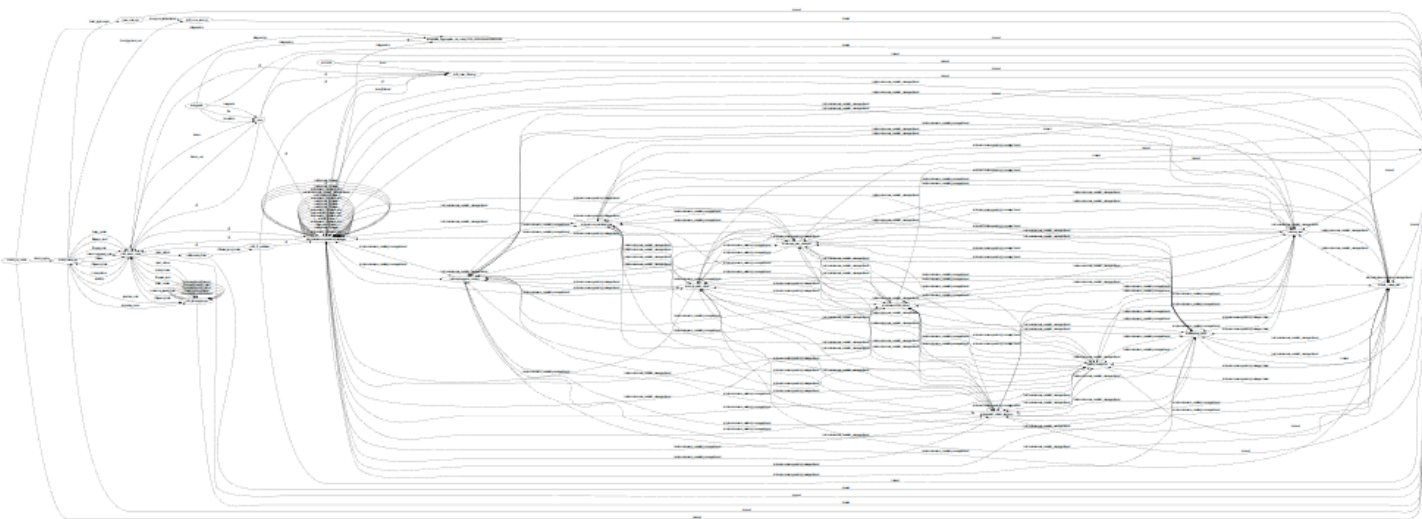
>> `rxgraph [options]` // in newer version of ROS groovy+ is replaced by `rqt_graph`
<http://wiki.ros.org/rxgraph>

ROS Tools: RXGRAPH



rxgraph

Path: / Quiet All topics ?

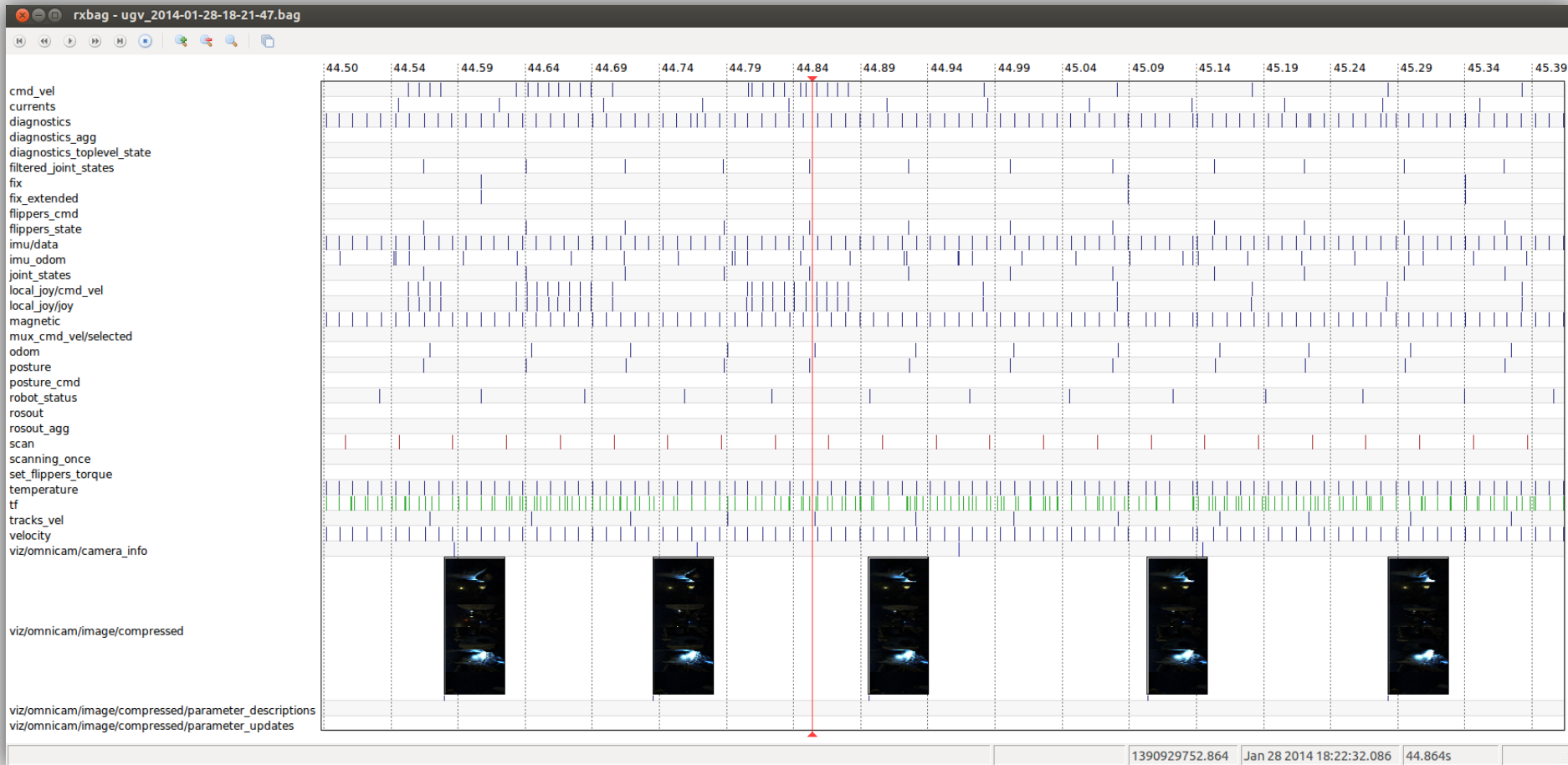


Info:

- Node [/viz/omnicamera_nodelet_manager]
- Publications:
 - * /viz/ptz/camera_info [sensor_msgs/CameraInfo]
 - * /viz/pano_vodom/image/theora/parameter_updates [dynamic_reconfigure/Config]
 - * /viz/omnicam/image_raw/theora [theora_image_transport/Packet]
 - * /viz/image_proc_debayer/parameter_updates [dynamic_reconfigure/Config]
 - * /viz/camera_5/image/compressedDepth/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
 - * /viz/pano_vodom/image/compressed/parameter_updates [dynamic_reconfigure/Config]
 - * /viz/camera_1/image/compressedDepth/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
 - * /viz/omnicam/image_raw/compressed/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
 - * /viz/camera_2/image/compressed/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
 - * /viz/ptz/image/theora/parameter_updates [dynamic_reconfigure/Config]

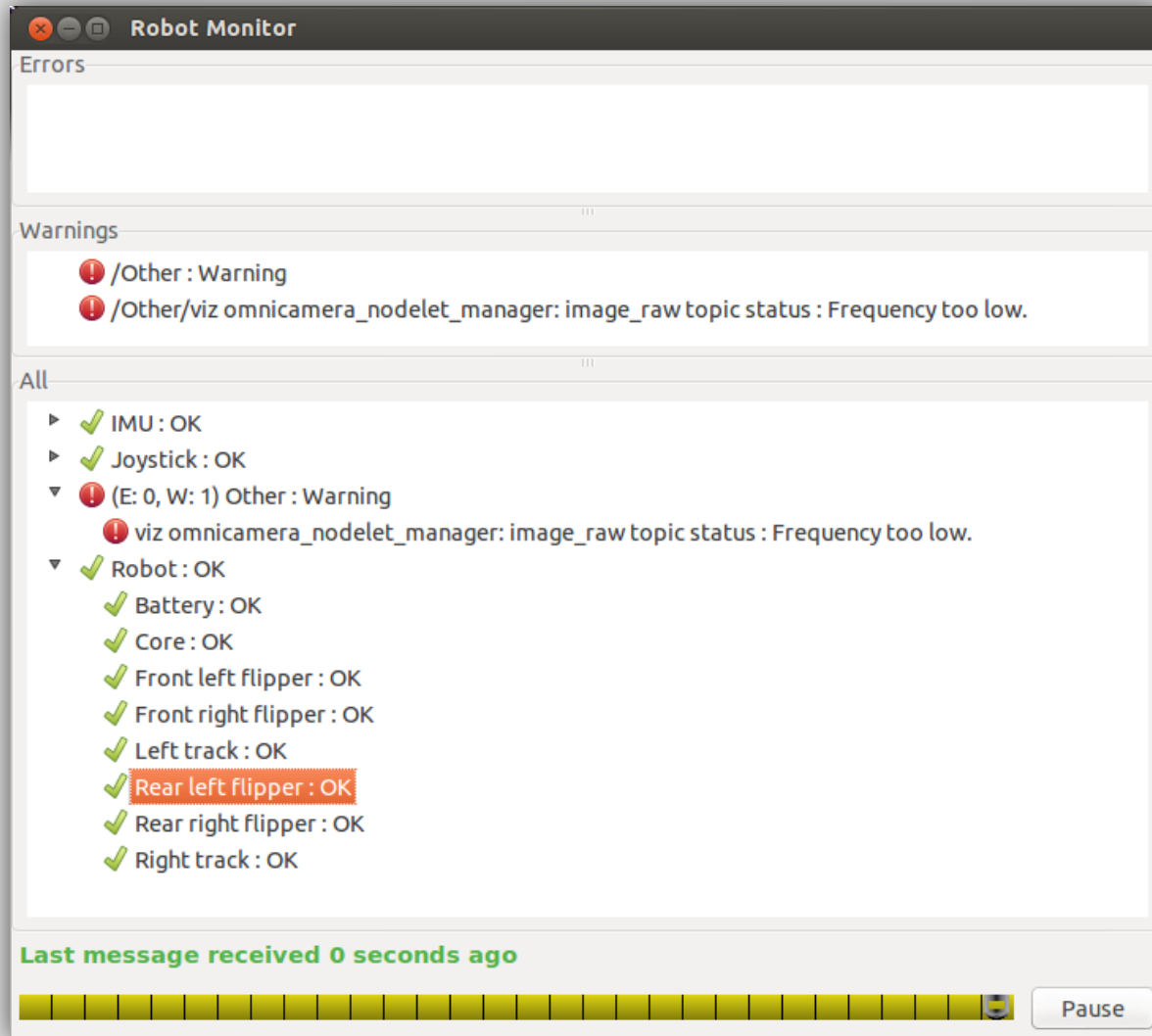
How does it look in the real life?

ROS Tools: RXBAG



>> `rxbag name_of_bag.bag` // in newer version of ROS groovy+ replaced by `rqt_bag`
<http://wiki.ros.org/rxbag>

ROS Tools: ROBOT MONITOR



>> `roslaunch robot_monitor robot_monitor`

http://wiki.ros.org/robot_monitor

ROS Tools: RXCONSOLE



20

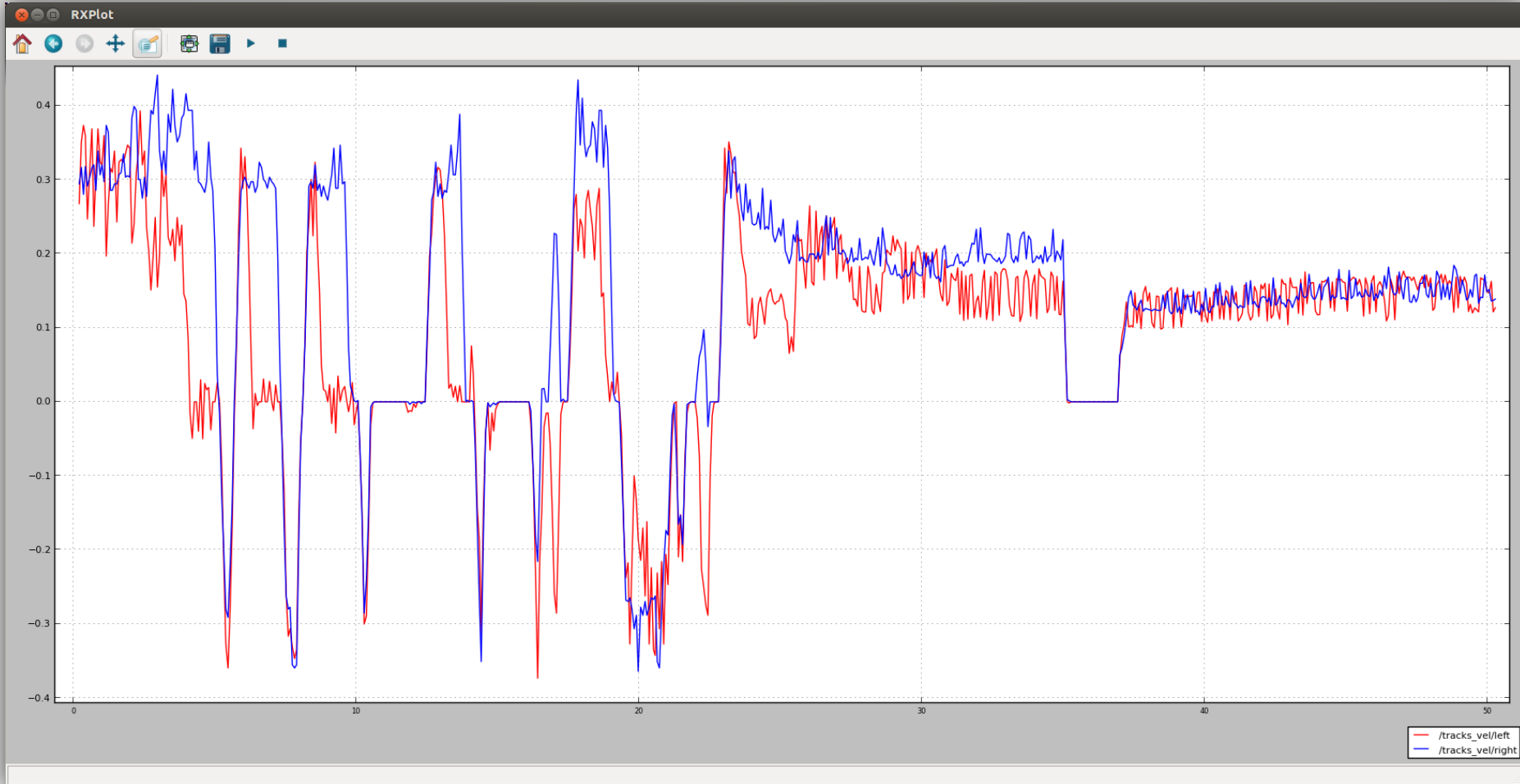
The screenshot shows the rxconsole application interface. The main window displays a log of messages with the following columns: Message, Severity, Node, Time, Topics, and Location. The messages are filtered to show only 'Error' and 'Warn' levels. The messages include information about point cloud processing, filter application, and image size calibration. The interface also features a 'Severity' filter bar at the bottom with checkboxes for Fatal, Error, Warn, Info, and Debug. There are also buttons for 'Pause', 'Clear', 'Setup', 'Levels...', and 'New Window...'. A search bar is visible at the bottom with the text 'Enabled' and a search button labeled 'Include'.

Message	Severity	Node	Time	Topics	Location
* ObservationDirectionDataPointsFilter - 1456 points out (-0%)	Info	/mapper	1390929844.895612952	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* OrientNormalsDataPointsFilter - 1456 points out (-0%)	Info	/mapper	1390929844.895612952	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Applied 5 filters - 1456 points out (-97.6087%)	Info	/mapper	1390929844.895612952	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Input filters took 0.0335683 [s]	Info	/mapper	1390929844.895612952	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Not enough points in newPointCloud: only 1456 pts.	Error	/mapper	1390929844.959889524	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Processing new point cloud	Info	/mapper	1390929847.906570732	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Applying 5 DataPoints filters - 60987 points in	Info	/mapper	1390929847.906570732	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* SimpleSensorNoiseDataPointsFilter - 60987 points out (-0%)	Info	/mapper	1390929847.916627639	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* SamplingSurfaceNormalDataPointsFilter - 48736 points out (-20.0879%)	Info	/mapper	1390929847.946802268	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* MaxDensityDataPointsFilter - 1458 points out (-97.0084%)	Info	/mapper	1390929847.946802268	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* ObservationDirectionDataPointsFilter - 1458 points out (-0%)	Info	/mapper	1390929847.946802268	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* OrientNormalsDataPointsFilter - 1458 points out (-0%)	Info	/mapper	1390929847.946802268	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Applied 5 filters - 1458 points out (-97.6093%)	Info	/mapper	1390929847.946802268	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Input filters took 0.0415154 [s]	Info	/mapper	1390929847.946802268	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Not enough points in newPointCloud: only 1458 pts.	Error	/mapper	1390929847.946802268	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Calibrated image size (0x0) matches neither full Format7 size (1616x3696) nor ROI ...	Warn	/viz/omnicamera_nodelet_...	1390929848.115466718	/rosout,/viz/camera1394_d...	/home/robot/bleed...
Calibrated image size (0x0) matches neither full Format7 size (1616x3696) nor ROI ...	Warn	/viz/omnicamera_nodelet_...	1390929848.115466718	/rosout,/viz/camera1394_d...	/home/robot/bleed...
Processing new point cloud	Info	/mapper	1390929851.041001531	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Applying 5 DataPoints filters - 59992 points in	Info	/mapper	1390929851.041001531	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* SimpleSensorNoiseDataPointsFilter - 59992 points out (-0%)	Info	/mapper	1390929851.051084068	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* SamplingSurfaceNormalDataPointsFilter - 47961 points out (-20.0543%)	Info	/mapper	1390929851.071201510	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* MaxDensityDataPointsFilter - 1474 points out (-96.9267%)	Info	/mapper	1390929851.081255279	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* ObservationDirectionDataPointsFilter - 1474 points out (-0%)	Info	/mapper	1390929851.081255279	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
* OrientNormalsDataPointsFilter - 1474 points out (-0%)	Info	/mapper	1390929851.081255279	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Applied 5 filters - 1474 points out (-97.543%)	Info	/mapper	1390929851.081255279	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Input filters took 0.0394042 [s]	Info	/mapper	1390929851.081255279	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...
Not enough points in newPointCloud: only 1474 pts.	Error	/mapper	1390929851.091307582	/rosout,/tf,/point_map,/i...	/local/nifti_codes/b...

>> rxconsole // in newer version of ROS groovy+ is replaced by rqt_console

<http://wiki.ros.org/rxconsole>

ROS Tools: RXPLOTT



```
>> rxplot /topic1/field1:field2:field3
```

<http://wiki.ros.org/rxplot>

ROS Tools: CHEATSHEET



ROS Cheat Sheet

Filesystem Command-line Tools

rospack/rostack A tool inspecting `packages/stacks`.
roscd Changes directories to a package or stack.
rosls Lists package or stack information.
roscrcat-pkg Creates a new ROS package.
roscrcat-stack Creates a new ROS stack.
roscdep Installs ROS package system dependencies.
rosmake Builds a ROS package.
roswtf Displays a errors and warnings about a running ROS system or launch file.
rxdeps Displays package structure and dependencies.

Usage:
\$ rospack find [package]
\$ roscd [package[/subdir]]
\$ rosls [package[/subdir]]
\$ roscrcat-pkg [package.name]
\$ rosmake [package]
\$ roscdep install [package]
\$ roswtf or roswtf [file]
\$ rxdeps [options]

Common Command-line Tools

roscore
A collection of `nodes` and programs that are pre-requisites of a ROS-based system. You must have a roscore running in order for ROS nodes to communicate.

roscore is currently defined as:

```
master  
parameter server  
rosout
```

Usage:
\$ roscore

rosmmsg/rossrv

rosmmsg/rossrv displays Message/Service (msg/srv) data structure definitions.

Commands:
rosmmsg show Display the fields in the msg.
rosmmsg users Search for code using the msg.
rosmmsg md5 Display the msg md5 sum.
rosmmsg package List all the messages in a package.
rosnode packages List all the packages with messages.

Examples:
Display the Pose msg:
\$ rosmmsg show Pose
List the messages in nav_msgs:
\$ rosmmsg package nav_msgs
List the files using sensor_msgs/CameraInfo:
\$ rosmmsg users sensor_msgs/CameraInfo

roslaunch

roslaunch allows you to run an executable in an arbitrary package without having to cd (or roscd) there first.

Usage:
\$ roslaunch package executable

Example:
Run turtlesim:
\$ roslaunch turtlesim turtlesim.launch

rostopic

Displays debugging information about ROS nodes, including publications, subscriptions and connections.

Commands:
rostopic ping Test connectivity to node.
rostopic list List active nodes.
rostopic info Print information about a node.
rostopic machine List nodes running on a particular machine.
rostopic kill Kills a running node.

Examples:
Kill all nodes:
\$ rostopic kill -a
List nodes on a machine:
\$ rostopic machine aqy.local
Ping all nodes:
\$ rostopic ping --all

roscat

Starts ROS nodes locally and remotely via SSH, as well as setting parameters on the parameter server.

Examples:
Launch on a different port:
\$ roscat -p 1234 package filename.launch
Launch a file in a package:
\$ roscat package filename.launch
Launch on the local nodes:
\$ roscat --local package filename.launch

rostopic

A tool for displaying debug information about ROS topics, including publishers, subscribers, publishing rate, and messages.

Commands:
rostopic bw Display bandwidth used by topic.
rostopic echo Print messages to screen.
rostopic hz Display publishing rate of topic.
rostopic list Print information about active topics.
rostopic pub Publish data to topic.
rostopic type Print topic type.
rostopic find Find topics by type.

rostopic

Examples:
Publish hello at 10 Hz:
\$ rostopic pub -r 10 /topic.name std_msgs/String hello
Clear the screen after each message is published:
\$ rostopic echo -c /topic.name
Display messages that match a given Python expression:
\$ rostopic echo --filter "m.data=='foo'" /topic.name
Pipe the output of rostopic to rosmmsg to view the msg type:
\$ rostopic type /topic.name | rosmmsg show

roscparam

A tool for getting and setting ROS parameters on the parameter server using YAML-encoded files.

Commands:
roscparam set Set a parameter.
roscparam get Get a parameter.
roscparam load Load parameters from a file.
roscparam dump Dump parameters to a file.
roscparam delete Delete a parameter.
roscparam list List parameter names.

Examples:
List all the parameters in a namespace:
\$ roscparam list /namespace
Setting a list with one as a string, integer, and float:
\$ roscparam set /foo "[1', 1, 1.0]"
Dump only the parameters in a specific namespace to file:
\$ roscparam dump dump.yaml /namespace

rosservice

A tool for listing and querying ROS services.

Commands:
rosservice list Print information about active services.
rosservice node Print the name of the node providing a service.
rosservice call Call the service with the given args.
rosservice args List the arguments of a service.
rosservice type Print the service type.
rosservice uri Print the service ROSRPC uri.
rosservice find Find services by service type.

Examples:
Call a service from the command-line:
\$ rosservice call /add_two.ints 1 2
Pipe the output of rosservice to rostopic to view the srv type:
\$ rosservice type add_two.ints | rostopic show
Display all services of a particular type:
\$ rosservice find rospy.tutorials/AddTwoInts

>> rxplot /topic1/field1:field2:field3

<http://wiki.ros.org/rxplot>