Introduction to ROS

Marco Bernardi

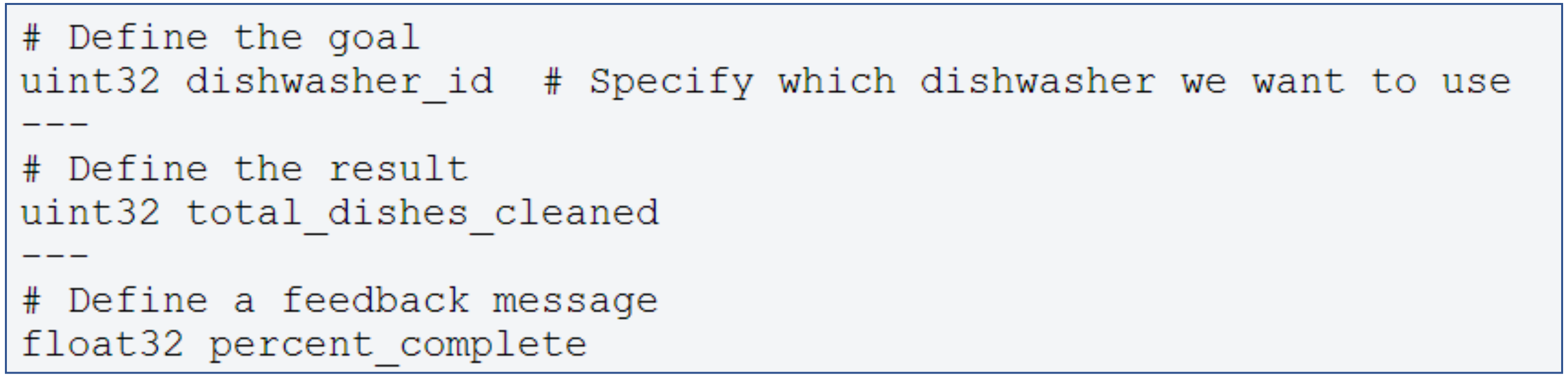
Internet of Things 2020/2021

Actions

- ROS actions are the best way to implement interfaces to time-extended, goal-oriented behaviors
- Similar to the request and response of a service, an action uses a goal to initiate a behavior and sends a result when the behavior is complete
- But the action further uses feedback to provide updates on the behavior’s progress toward the goal and also allows for goals to be canceled
- Actions are asynchronous (in contrast to services)

Actions 2

- The action specification is defined in an .action file
- These files are placed in the package’s ./action directory
- Example for an action file:



```
# Define the goal
uint32 dishwasher_id  # Specify which dishwasher we want to use
---
# Define the result
uint32 total_dishes_cleaned
---
# Define a feedback message
float32 percent_complete
```

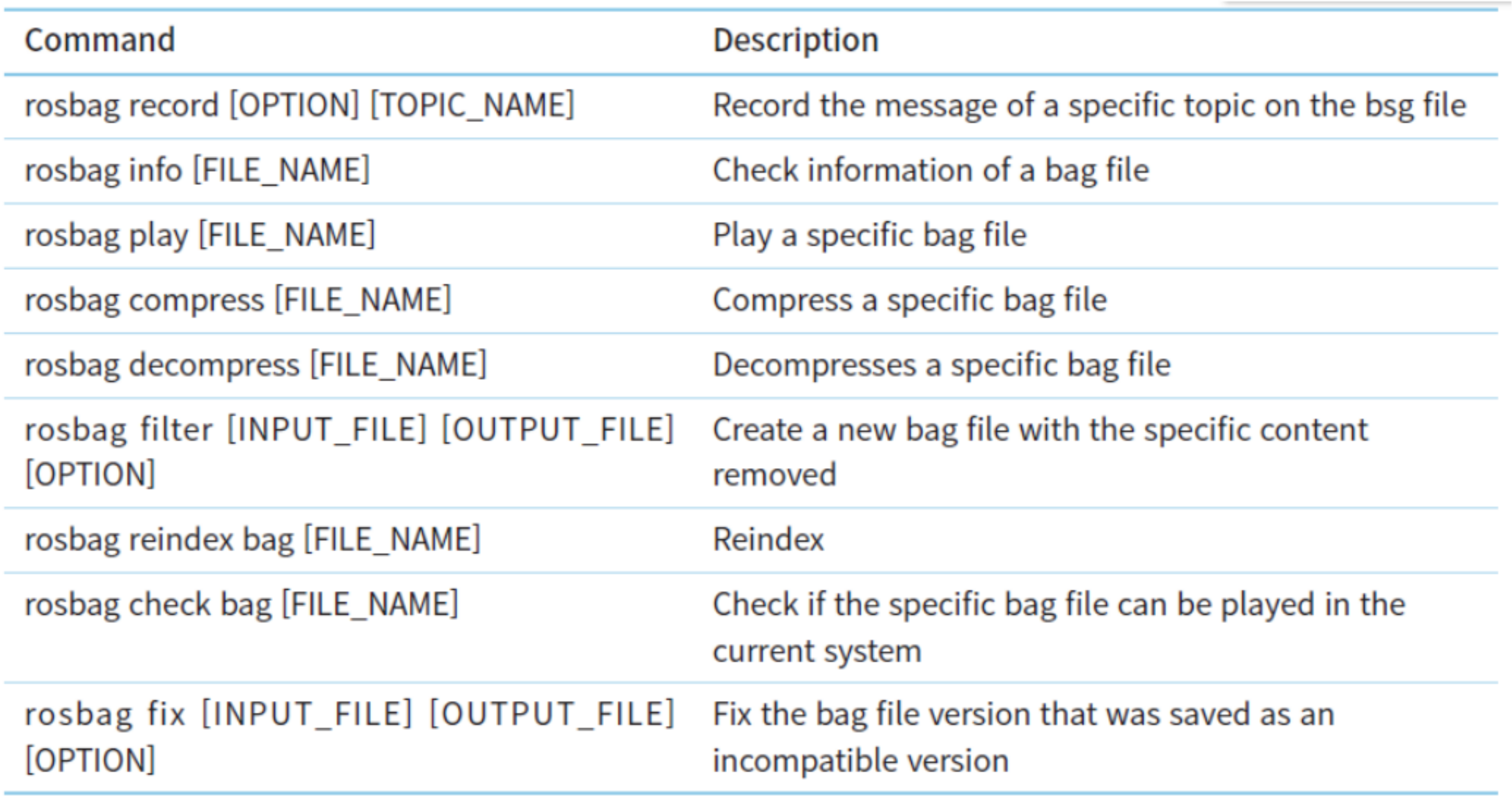
rosbag

- Data contained in ROS messages can be recorded in .bag files
- To have one recording that can be used repeatedly by playing back each time the exact operational scenario in which the bag was registered

Example: sensors data

- An example of the usefulness of bag files is given by registration messages containing the data produced by the robot sensors
- During experiments with the real robot, sensor data can be registered in a bag
- Recorded messages can then be loaded without the need to repeat the experiment, thus allowing more easily develop algorithms that require frequent parameter changes

Rosbag tool

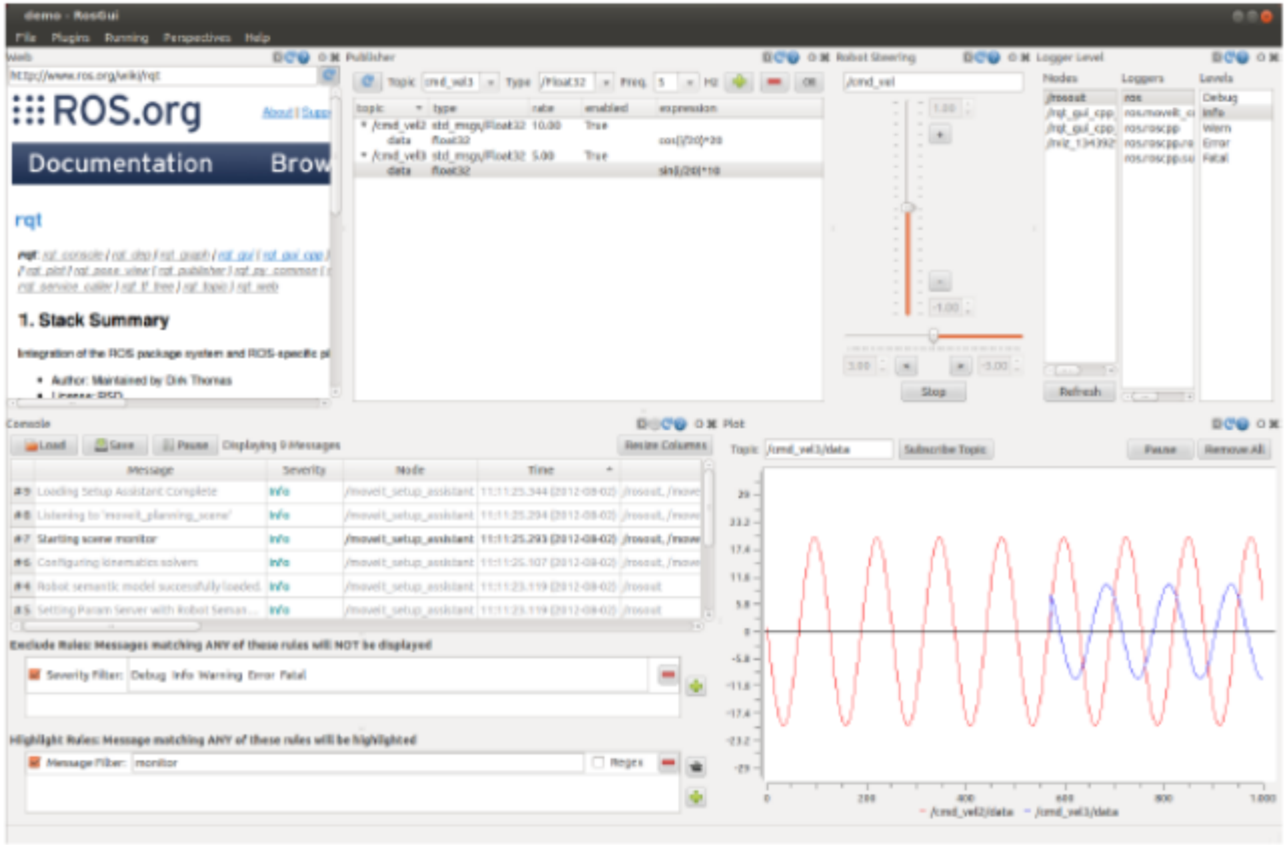


Command	Description
rosbag record [OPTION] [TOPIC_NAME]	Record the message of a specific topic on the bsg file
rosbag info [FILE_NAME]	Check information of a bag file
rosbag play [FILE_NAME]	Play a specific bag file
rosbag compress [FILE_NAME]	Compress a specific bag file
rosbag decompress [FILE_NAME]	Decompresses a specific bag file
rosbag filter [INPUT_FILE] [OUTPUT_FILE] [OPTION]	Create a new bag file with the specific content removed
rosbag reindex bag [FILE_NAME]	Reindex
rosbag check bag [FILE_NAME]	Check if the specific bag file can be played in the current system
rosbag fix [INPUT_FILE] [OUTPUT_FILE] [OPTION]	Fix the bag file version that was saved as an incompatible version

Rqt visualizer & user interface (1)

- User interface developed in Qt
- Custom interfaces can be setup
- Lots of existing plugins
- Simple to write own plugins

rqt



rqt

rqt: rqt_console | rqt_dep | rqt_graph | rqt_gui | rqt_gui_cpp | rqt_plot | rqt_pose_view | rqt_publisher | rqt_py_common | rqt_service_caller | rqt_tf_tree | rqt_topic | rqt_web

1. Stack Summary

Integration of the ROS package system and ROS-specific pl

- Author: Maintained by Dirk Thomas
- License: BSD

topic	type	rate	enabled	expression
/cmd_vel2	std_msgs/Float32	10.00	True	
data	float32			cos(i/20)*20
/cmd_vel3	std_msgs/Float32	5.00	True	
data	float32			sin(i/20)*10

Displaying 9 Messages

	Message	Severity	Node	Time	
#9	Loading Setup Assistant Complete	Info	/moveit_setup_assistant	11:11:25.344 (2012-08-02)	/rosout, /move
#8	Listening to 'moveit_planning_scene'	Info	/moveit_setup_assistant	11:11:25.294 (2012-08-02)	/rosout, /move
#7	Starting scene monitor	Info	/moveit_setup_assistant	11:11:25.293 (2012-08-02)	/rosout, /move
#6	Configuring kinematics solvers	Info	/moveit_setup_assistant	11:11:25.107 (2012-08-02)	/rosout, /move
#4	Robot semantic model successfully loaded.	Info	/moveit_setup_assistant	11:11:23.119 (2012-08-02)	/rosout
#5	Setting Param Server with Robot Seman...	Info	/moveit_setup_assistant	11:11:23.119 (2012-08-02)	/rosout

Exclude Rules: Messages matching ANY of these rules will NOT be displayed

☑ Severity Filter: Debug Info Warning Error Fatal

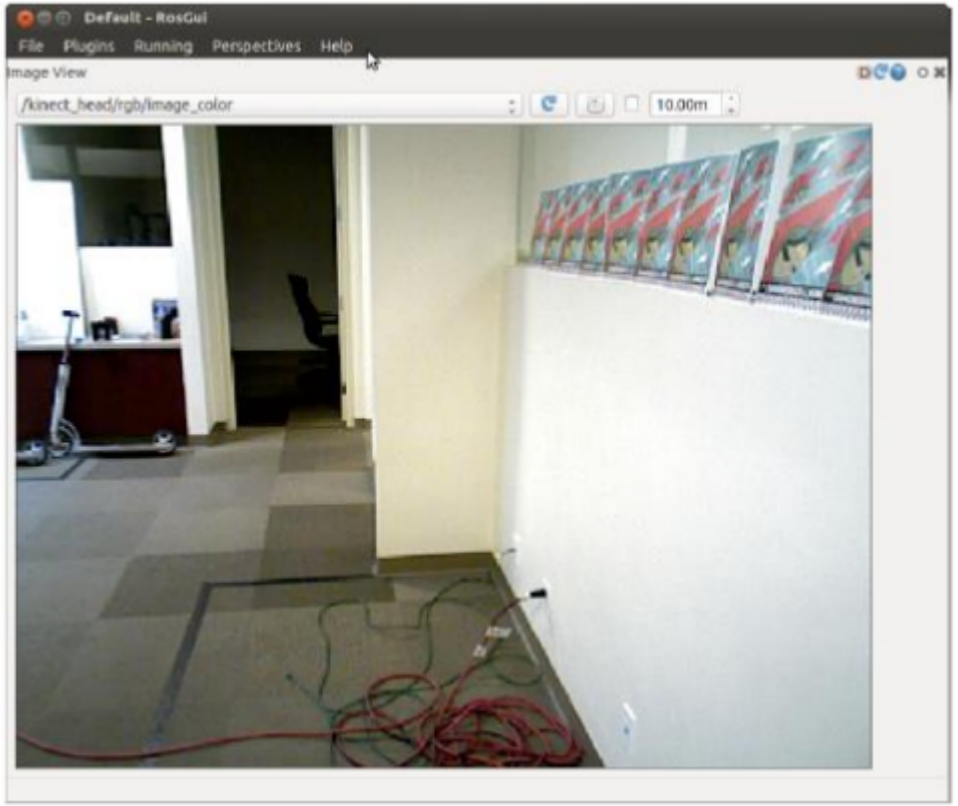
Highlight Rules: Message matching ANY of these rules will be highlighted

☑ Message Filter: monitor □ Regex

Rqt visualizer & user interface (2)

- Visualizing images

```
rosrun rqt_image_view rqt_image_view
```



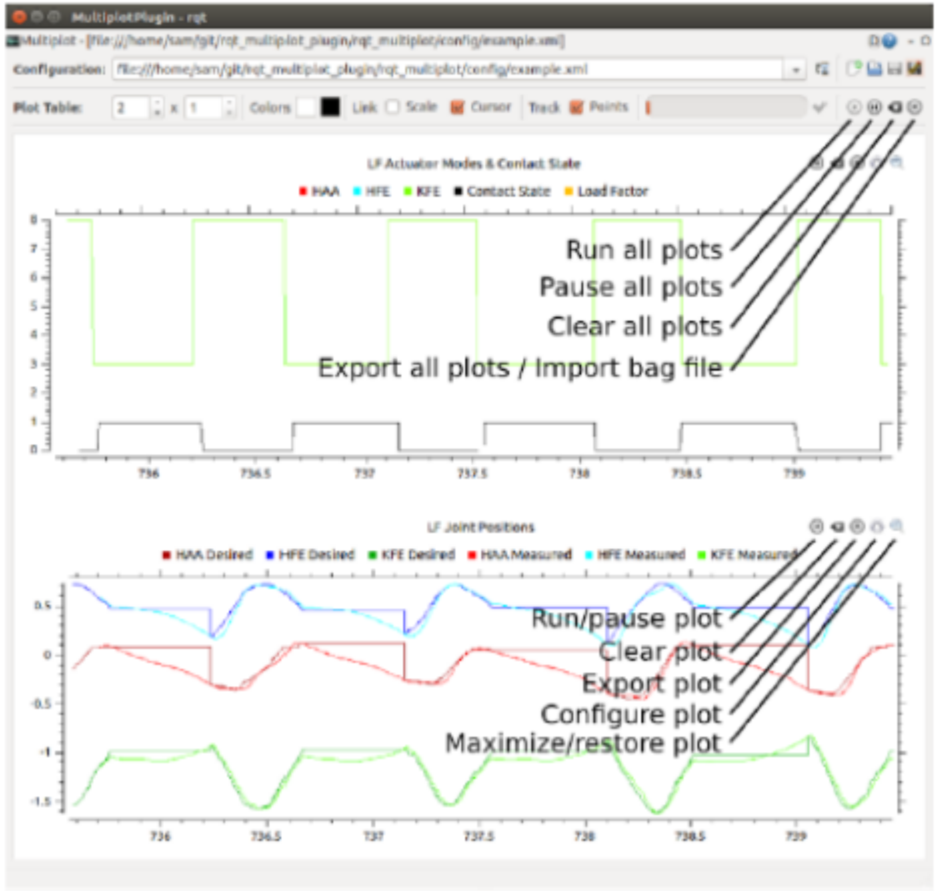
/kinect_head/rgb/image_color

10.00m

Rqt visualizer & user interface (3)

- Visualizing numeric plots

```
rosrun rqt_multiplot rqt_multiplot
```



MultiplotPlugin - rqt

Multiplot - [file:///home/sam/git/rqt_multiplot_plugin/rqt_multiplot/config/example.xml]

Configuration: file:///home/sam/git/rqt_multiplot_plugin/rqt_multiplot/config/example.xml

Plot Table: 2 x 1 Colors Link □ Scale ☑ Cursor Track ☑ Points

Run all plots

Pause all plots

Clear all plots

Export all plots / Import bag file

Run/pause plot

Clear plot

Export plot

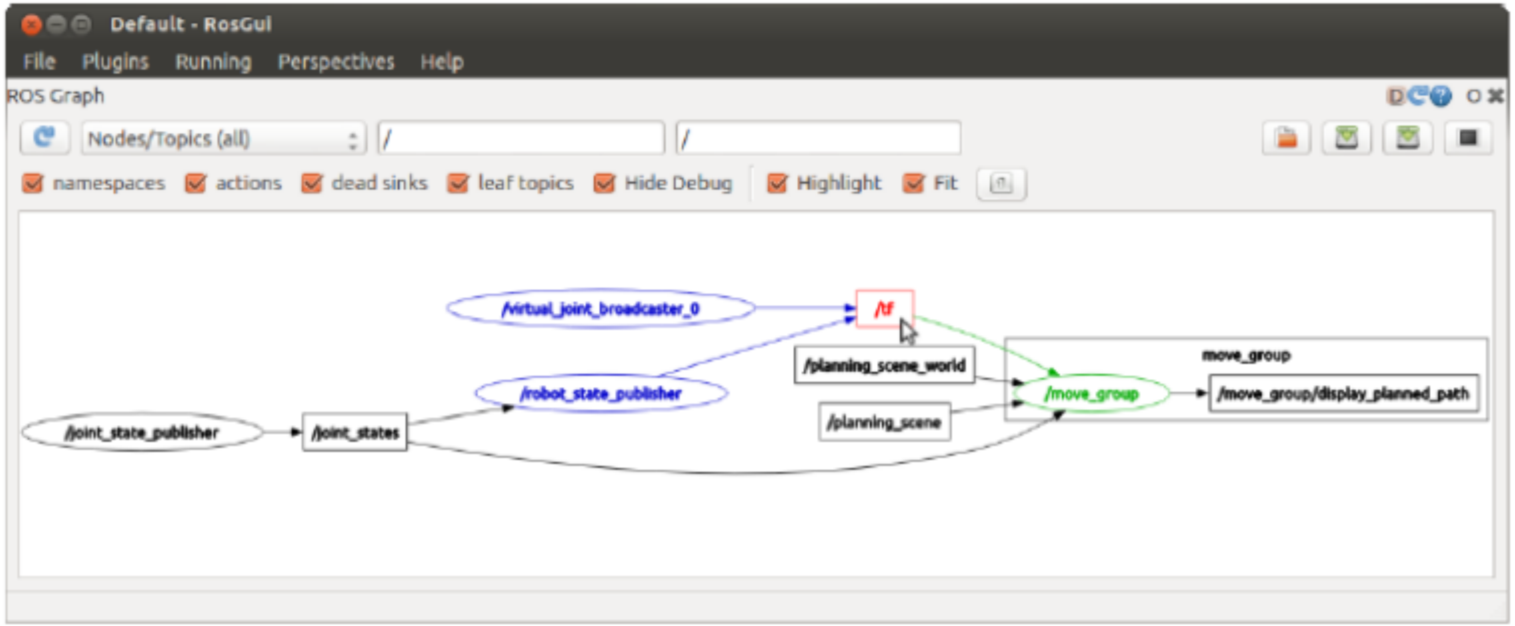
Configure plot

Maximize/restore plot

Rqt visualizer & user interface (4)

- Visualizing ros computational graph

```
rosrun rqt_graph rqt_graph
```



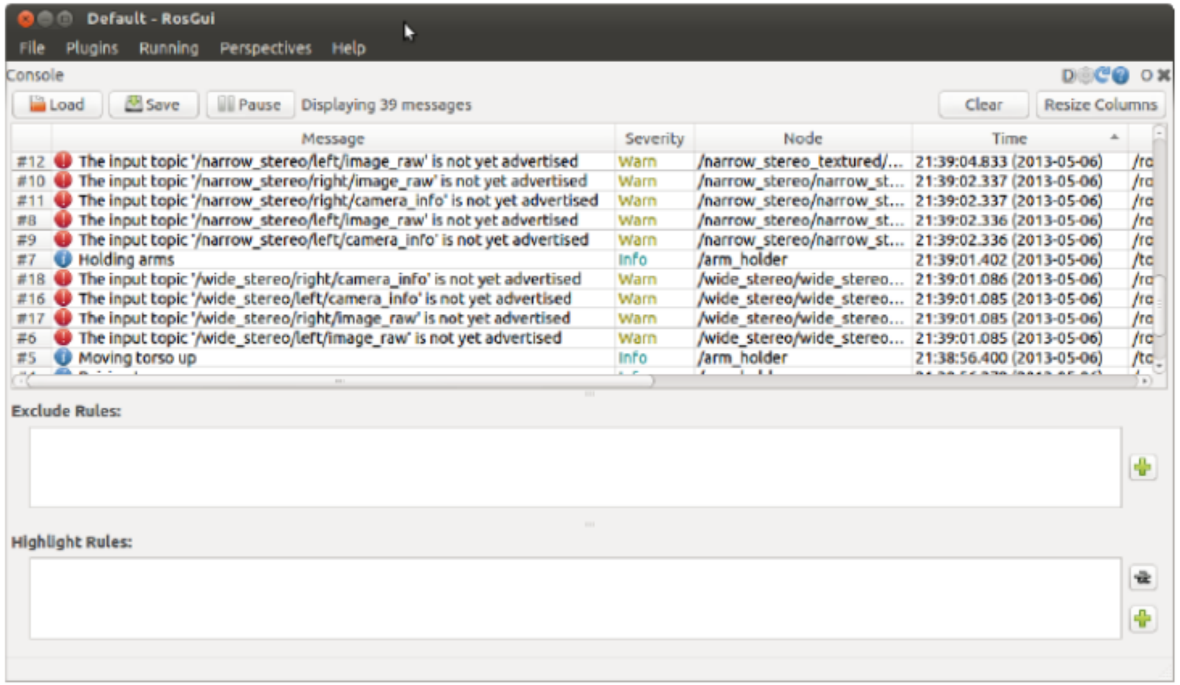
Nodes/Topics (all) | / | /

☑ namespaces ☑ actions ☑ dead sinks ☑ leaf topics ☑ Hide Debug ☑ Highlight ☑ Fit

Rqt visualizer & user interface (5)

- Displaying and filtering ROS Messages

```
rosrun rqt_console rqt_console
```



Displaying 39 messages

	Message	Severity	Node	Time	
#12	The input topic '/narrow_stereo/left/image_raw' is not yet advertised	Warn	/narrow_stereo_textured/...	21:39:04.833 (2013-05-06)	/ro
#10	The input topic '/narrow_stereo/right/image_raw' is not yet advertised	Warn	/narrow_stereo/narrow_st...	21:39:02.337 (2013-05-06)	/ro
#11	The input topic '/narrow_stereo/right/camera_info' is not yet advertised	Warn	/narrow_stereo/narrow_st...	21:39:02.337 (2013-05-06)	/ro
#8	The input topic '/narrow_stereo/left/image_raw' is not yet advertised	Warn	/narrow_stereo/narrow_st...	21:39:02.336 (2013-05-06)	/ro
#9	The input topic '/narrow_stereo/left/camera_info' is not yet advertised	Warn	/narrow_stereo/narrow_st...	21:39:02.336 (2013-05-06)	/ro
#7	Holding arms	Info	/arm_holder	21:39:01.402 (2013-05-06)	/to
#18	The input topic '/wide_stereo/right/camera_info' is not yet advertised	Warn	/wide_stereo/wide_stereo...	21:39:01.086 (2013-05-06)	/ro
#16	The input topic '/wide_stereo/left/camera_info' is not yet advertised	Warn	/wide_stereo/wide_stereo...	21:39:01.085 (2013-05-06)	/ro
#17	The input topic '/wide_stereo/right/image_raw' is not yet advertised	Warn	/wide_stereo/wide_stereo...	21:39:01.085 (2013-05-06)	/ro
#6	The input topic '/wide_stereo/left/image_raw' is not yet advertised	Warn	/wide_stereo/wide_stereo...	21:39:01.085 (2013-05-06)	/ro
#5	Moving torso up	Info	/arm_holder	21:38:56.400 (2013-05-06)	/to

Simulation environments in ROS

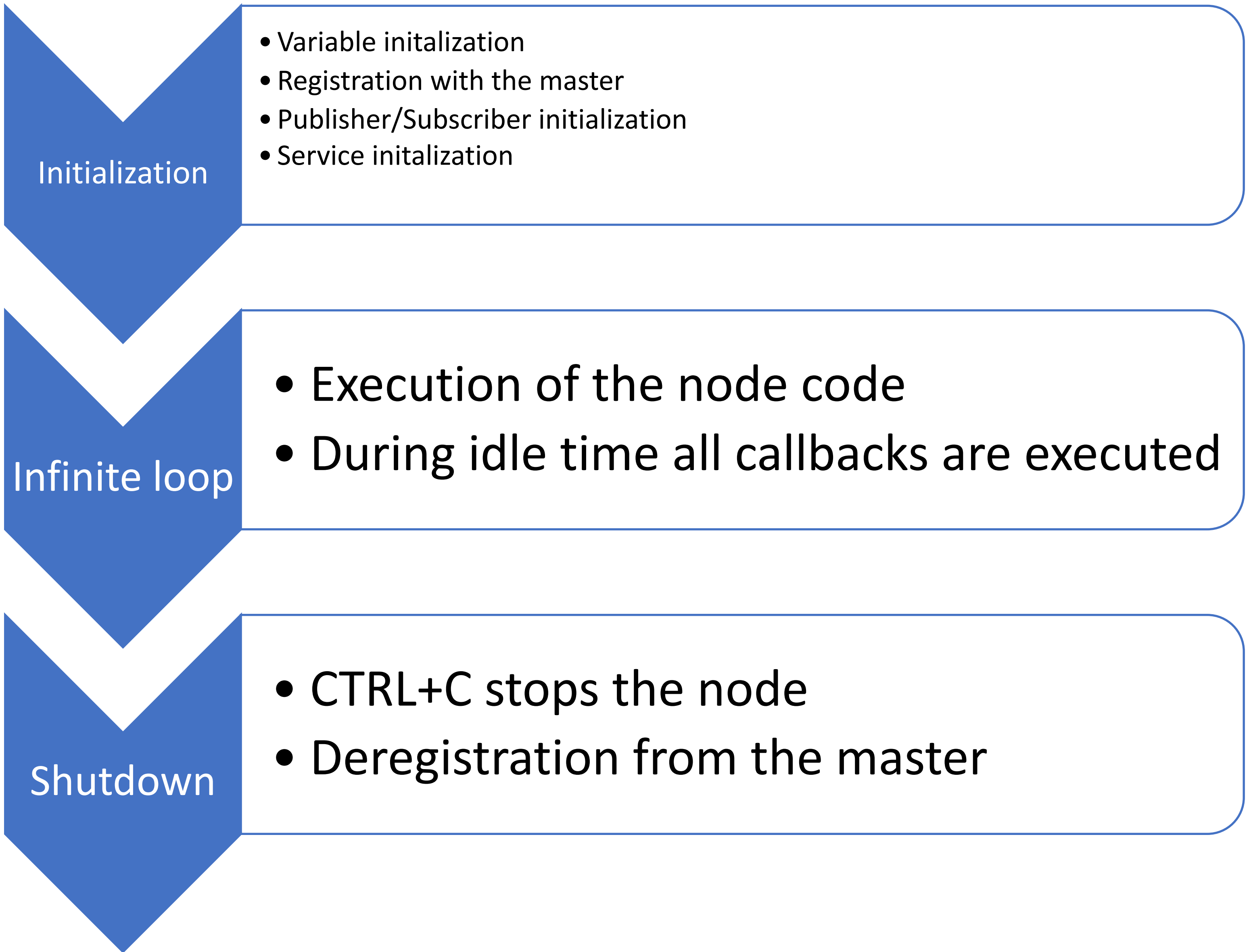
- Gazebo is the default simulator used in ROS framework, maintained as a separate project from OSRF.
- CoppeliaSim is a robotic simulators developed by Coppelia Robotics
  - It is a commercial software, that can be obtained for free in its educational version.

GAZEBO

CoppeliaSim

from the creators of V-REP

How does a node work?



Initialization

- Variable initalization
- Registration with the master
- Publisher/Subscriber initialization
- Service initalization

Infinite loop

- Execution of the node code
- During idle time all callbacks are executed

Shutdown

- CTRL+C stops the node
- Deregistration from the master

Hello word in ROS

```
#include <ros/ros.h>
int main(int argc, char* argv[])
{
                ros::init(argc, argv, "hello_world");
                ros::NodeHandle nodeHandle;
                ros::Rate loopRate(10);
                unsigned int count = 0;
                while (ros::ok())
                {
                    ROS_INFO_STREAM("Hello World " << count);
                    ros::spinOnce();
                    loopRate.sleep();
                    count++;
                }
return 0;
}
```

- ROS main header file include
- ros::init(…) must be called before other ROS functions
- The node handle is the access point for communications with the ROS system (topics, services, parameters)
- ros::Rate is a helper class to run loops at a desired frequency
- ros::ok() checks if a node should continue running
- ROS_INFO() logs messages to the filesystem
- ros::spinOnce() processes incoming messages via callbacks
  - ros::spin() processes callbacks and will not return until the node has been shutdown

Logging in ROS

- Mechanism for logging human readable text from nodes in the console and to log files
- Different severity levels (INFO, WARN, etc.)
- Instead of std::cout, use e.g. ROS_INFO
- Supports both printf- and stream-style formatting

```
ROS_INFO("Result: %d", result); // printf
ROS_INFO_STREAM("Result: " << result);
```

Subscriber

- Start listening to a topic by calling the method subscribe() of the node handle

```
ros::Subscriber subscriber = nodeHandle.subscribe(topic, queue_size, callback_function);
```

- When a message is received, callback function is called with the contents of the message as argument

Publisher

- Create a publisher with help of the node handle

```
ros::Publisher publisher = nodeHandle.advertise(topic, queue_size);
```

- Create the message contents
- Publish the contents with

```
publisher.publish(message);
```

Object Oriented Programming

```
#include #include <ros/ros.h>
#include "my_package/MyPackage.hpp"
int main(int argc, char* argv[])
{
		ros::init(argc, argv, "my_package");
		ros::NodeHandle nodeHandle("~");
		my_package::MyPackage myPackage(nodeHandle);
		ros::spin();
		return 0;
}
```

- Main node class providing ROS interface (subscribers, parameters, timers etc.)
- Class implementing the algorithmic part of the node
  - Note: The algorithmic part of the code could be separated in a (ROS-independent) library
- Specify a function handler to a method from within the class as

```
subscriber_ = nodeHandle_.subscribe(topic, queue_size, &ClassName::methodName,
this);
```

Additional Resources

- Site: http://www.ros.org/
- Blog: http://www.ros.org/news/
- Documentation: http://wiki.ros.org/