ATLAS

Introduction to ROS 2

NTA3

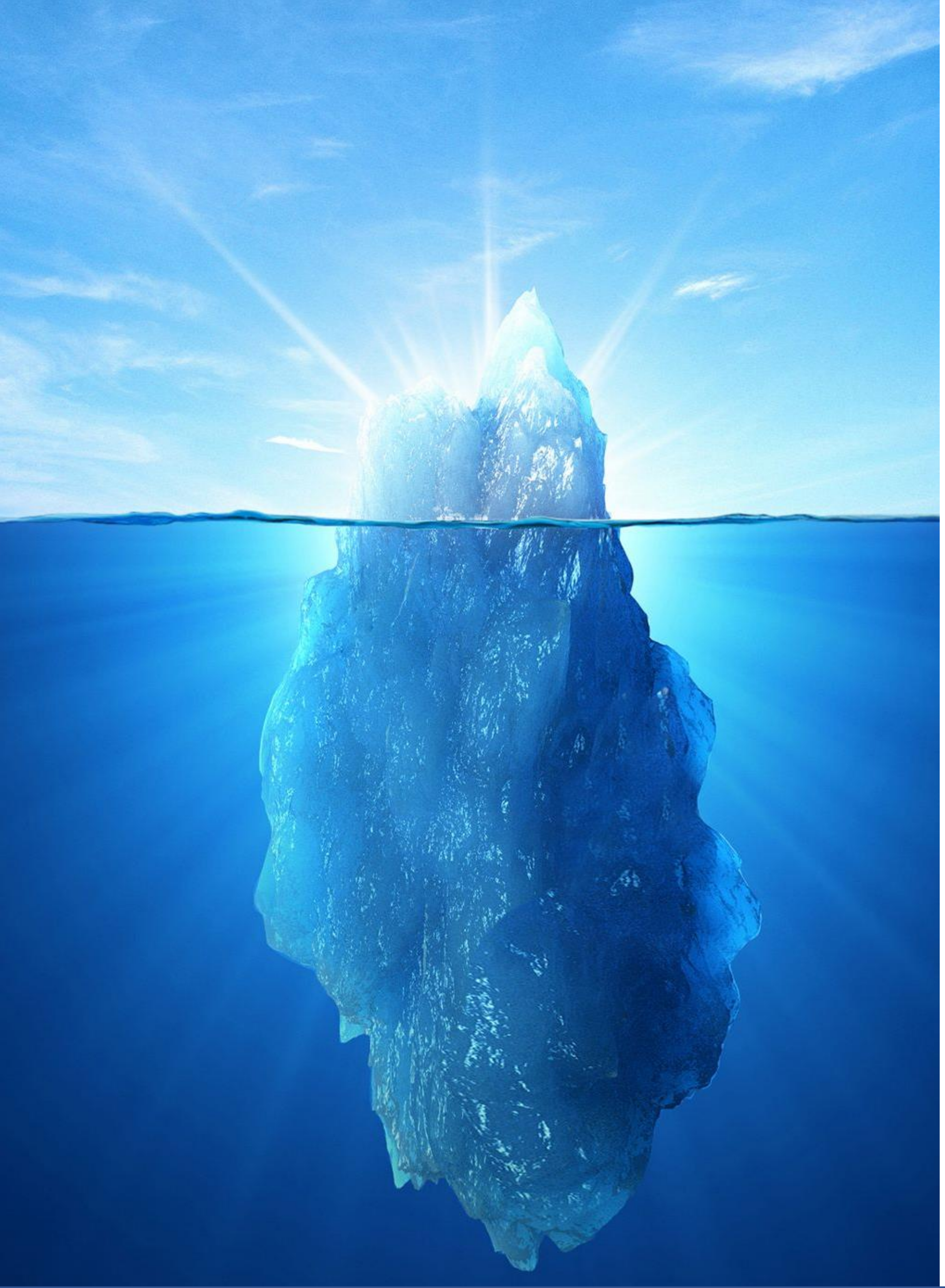
Diego Dall’Alba

UNIVR - Altair Robotics Lab

NTA3 @ KU Leuven 24 -28 February 2020

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

ALTAIR robotics lab



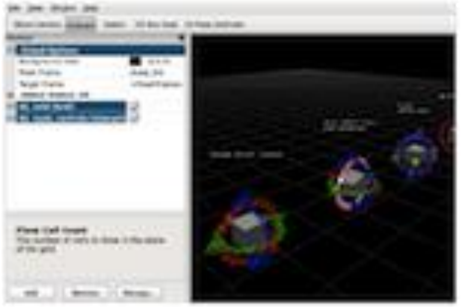
Overview

1. Visualization in ROS
2. Other ROS utils
   1. Tranformation
   2. URDF
   3. Ros time and ros bag
3. Simulation in ROS
4. Best practices in ROS

ROS Characteristics

Plumbing

- Process management
- Inter-process communication
- Device drivers



Tools

- Graphical user interface
- Simulation
- Visualization
- Data logging

Capabilities

- Control
- Planning
- Perception
- Mapping
- Manipulation

Ecosystem

- Package organization
- Software distribution
- Documentation
- Tutorials

Rqt visualizer & user interface (1)

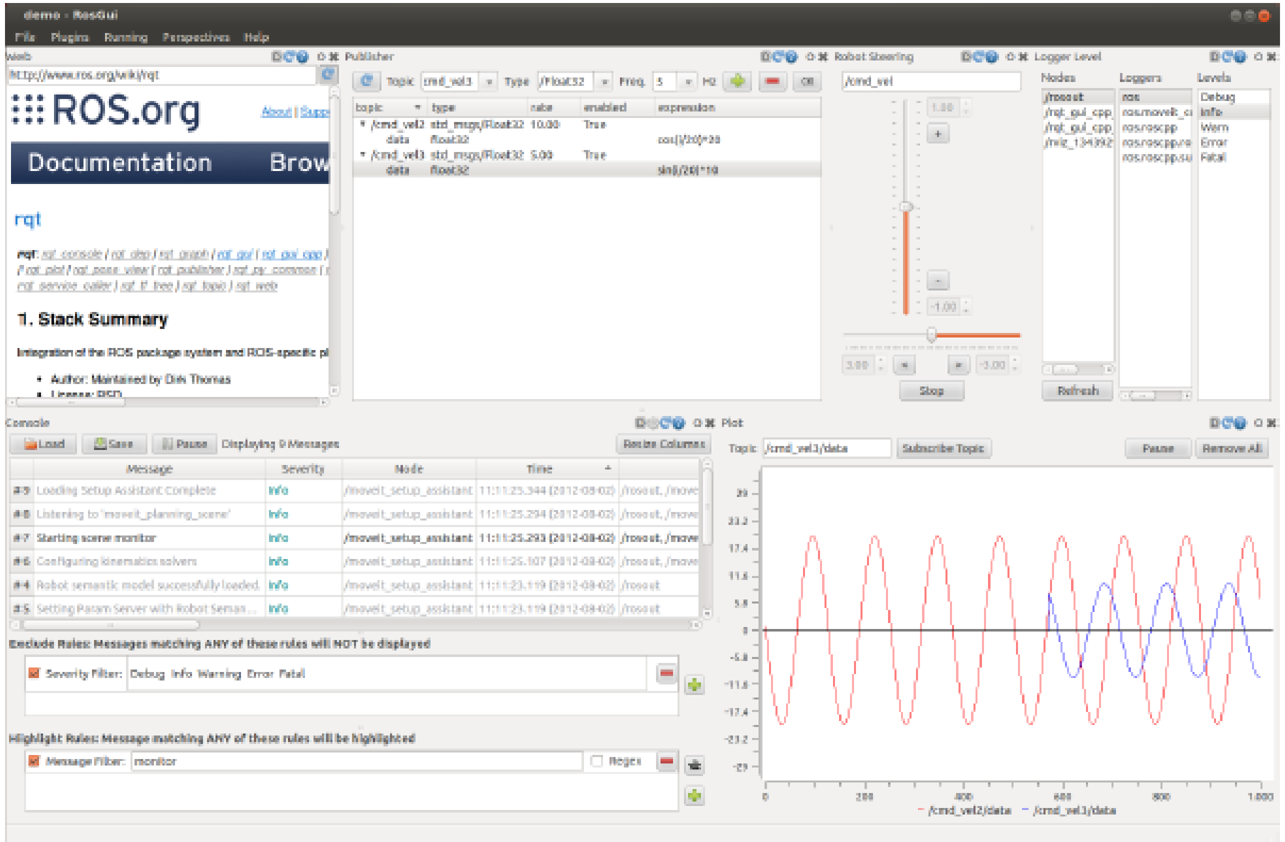
- User interface base on Qt
- Custom interfaces can be setup
- Lots of existing plugins exist
- Simple to write own plugins

Run RQT with

```
> rosrun rqt_gui rqt_gui
```

or

```
> rqt
```



http://www.ros.org/wiki/rqt

ROS.org About | Supp

Documentation Brow

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: cmd_vel3 Type: /Float32 Freq. 5 Hz

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

/cmd_vel

1.00 + − -1.00 3.00 -3.00 Stop

Nodes	Loggers	Levels
/rosout	ros	Debug
/rqt_gui_cpp_	ros.moveit_c	Info
/rqt_gui_cpp_	ros.roscpp	Warn
/rviz_134392	ros.roscpp.ro	Error
	ros.roscpp.su	Fatal

Refresh

Load Save Pause Displaying 9 Messages Resize Columns

	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

Topic: /cmd_vel3/data Subscribe Topic Pause Remove All

More info
http://wiki.ros.org/rqt/Plugins

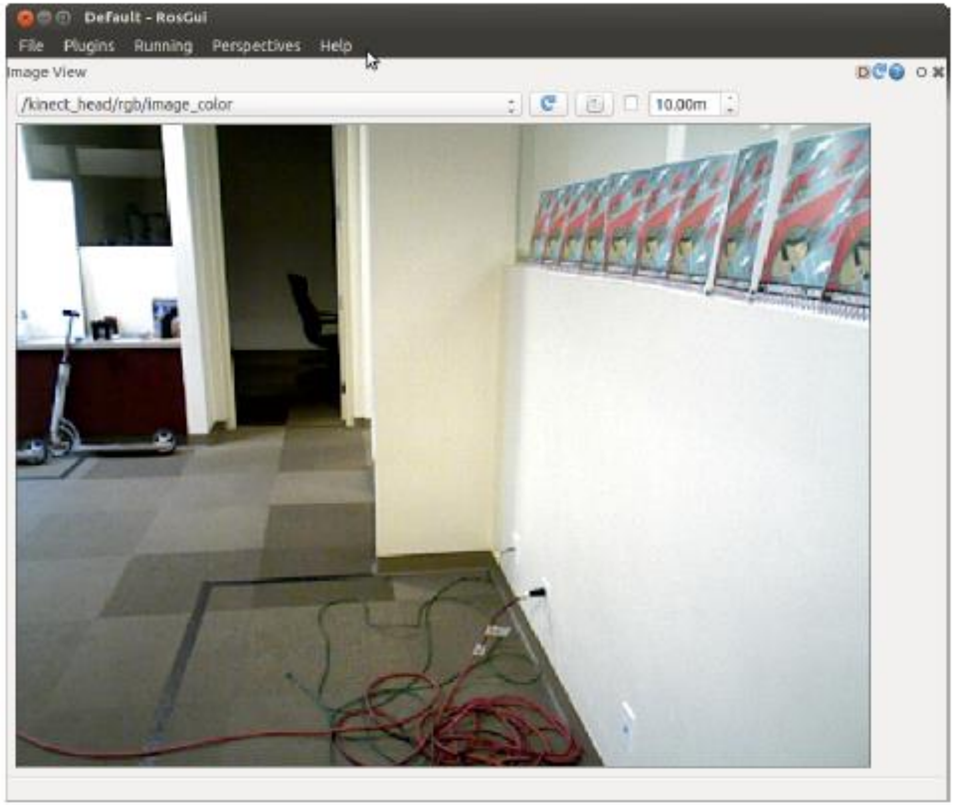
Rqt visualizer & user interface (2)

rqt_image_view

- Visualizing images

Run *rqt_graph* with

```
> rosrun rqt_image_view rqt_image_view
```



More info
http://wiki.ros.org/rqt_image_view

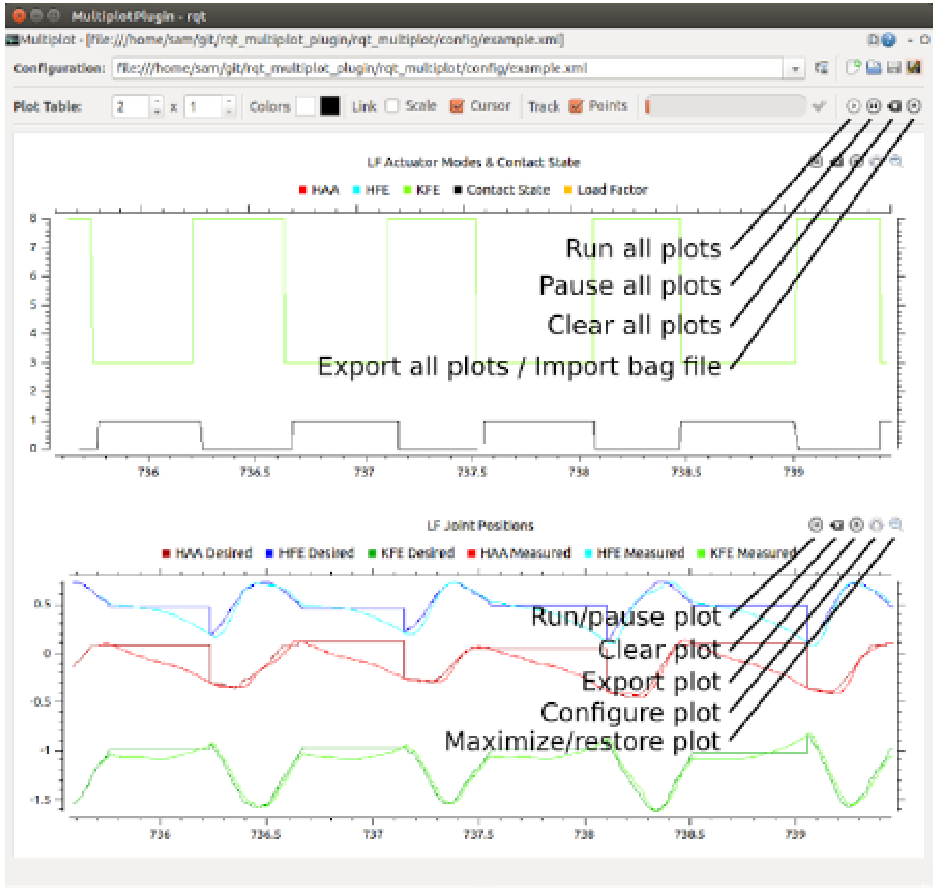
Rqt visualizer & user interface (3)

rqt_multiplot

- Visualizing numeric values in 2D plots

Run *rqt_multiplot* with

```
> 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

More info
http://wiki.ros.org/rqt_multiplot

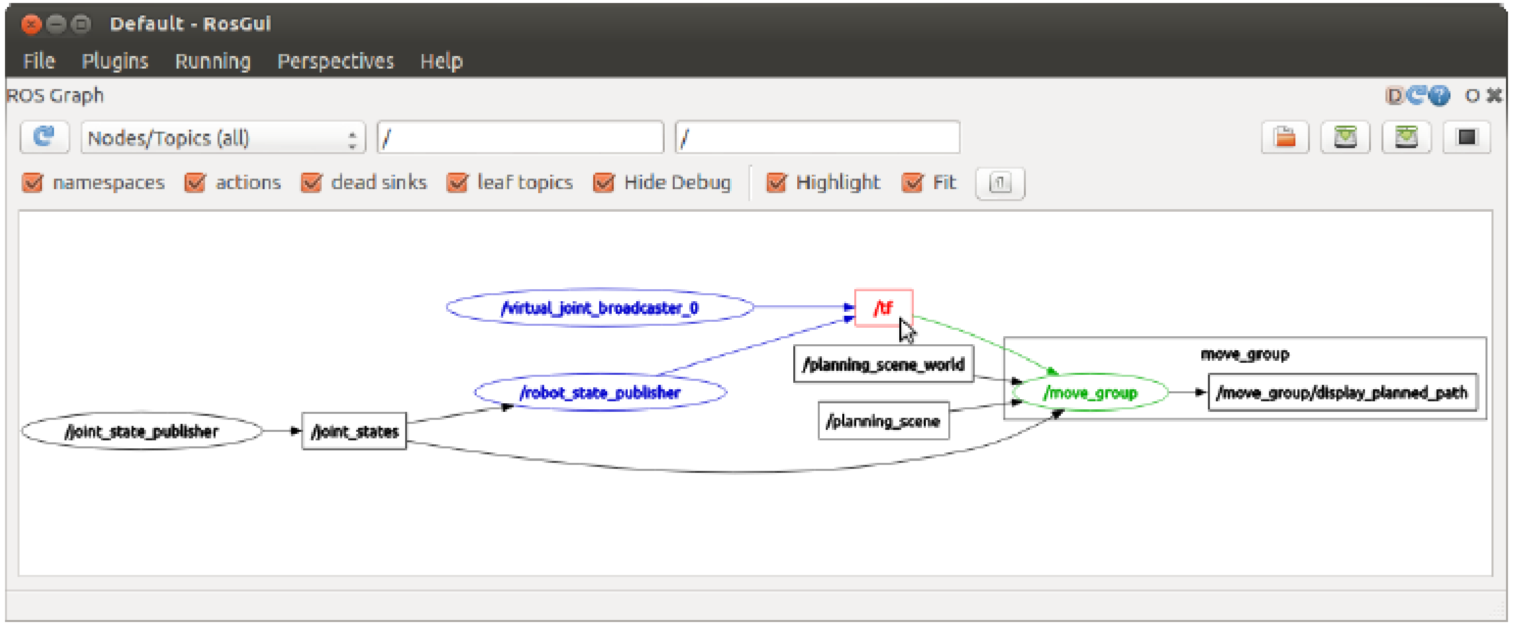
Rqt visualizer & user interface (4)

rqt_graph

- Visualizing the ROS computation graph

Run *rqt_graph* with

```
> rosrun rqt_graph rqt_graph
```



Nodes/Topics (all)

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

More info
http://wiki.ros.org/rqt_graph

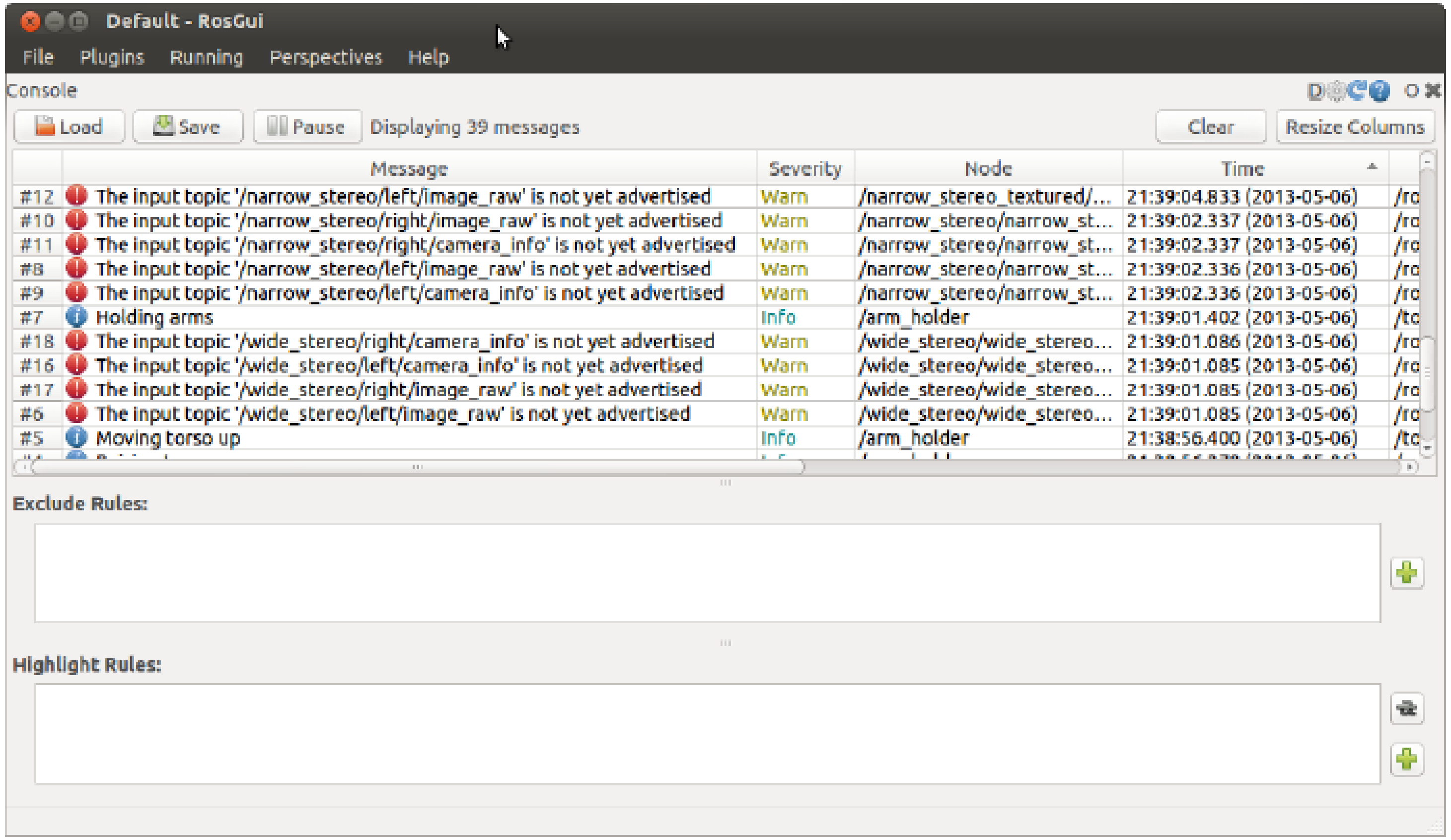
Rqt visualizer & user interface (5)

rqt_console

- Displaying and filtering ROS messages

Run *rqt_console* with

```
> 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

More info
http://wiki.ros.org/rqt_console

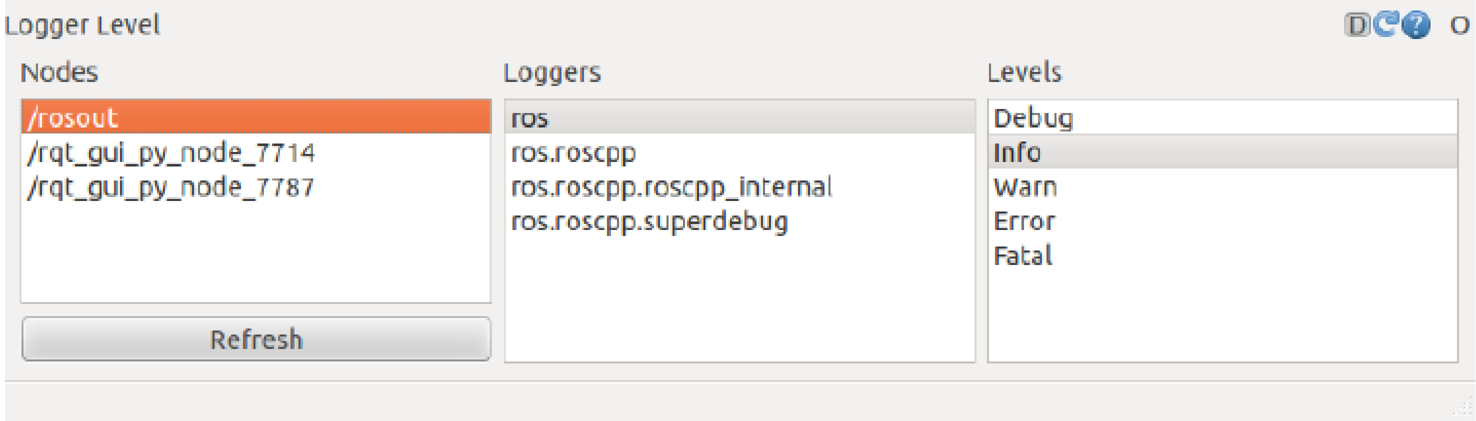
Rqt visualizer & user interface (6)

rqt_logger_level

- Configuring the logger level of ROS nodes

Run *rqt_logger_level* with

```
> rosrun rqt_logger_level
  rqt_logger_level
```



More info
http://wiki.ros.org/rqt_logger_level

URDF+Xacro

Unified Robot Description Format (**URDF**) is an XML format for representing a robot model.

It enable to describe kinematic, visual and dynamic properties of a manipulator.

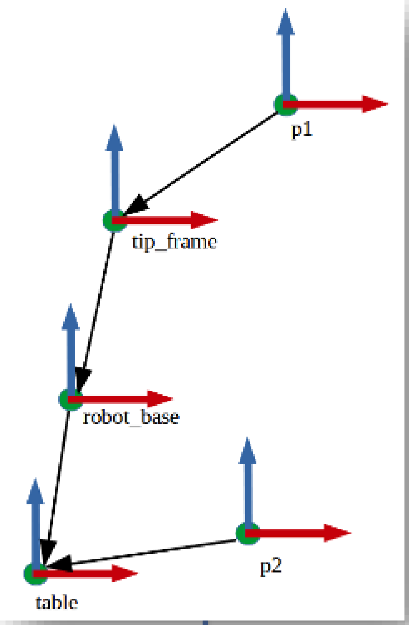
http://wiki.ros.org/urdf

Xacro is an XML macro language: enable construction of shorter and more readable XML files by using macros that expand to larger XML expressions.

http://wiki.ros.org/xacro

ROS provides parsing tools for reading and checking URDF files:

http://wiki.ros.org/urdf/Tutorials

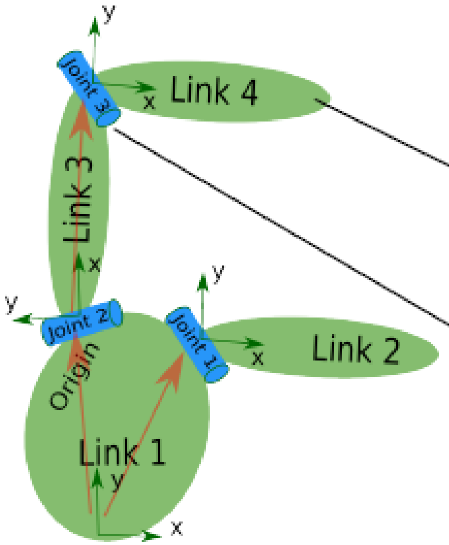


```
<robot name="test_robot">
 <link name="link1" />
 <link name="link2" />
 <link name="link3" />
 <link name="link4" />

 <joint name="joint1" type="continuous">
  <parent link="link1"/>
  <child link="link2"/>
 </joint>
 <joint name="joint2" type="continuous">
  <parent link="link1"/>
  <child link="link3"/>
 </joint>
 <joint name="joint3" type="continuous">
  <parent link="link3"/>
  <child link="link4"/>
 </joint>
</robot>
```

URDF Simple Example

- Description consists of a set of *link* elements and a set of *joint* elements
- Joints connect the links together



robot.urdf

```
<robot name="robot">
  <link> ... </link>
  <link> ... </link>
  <link> ... </link>

  <joint> .... </joint>
  <joint> .... </joint>
  <joint> .... </joint>
</robot>
```

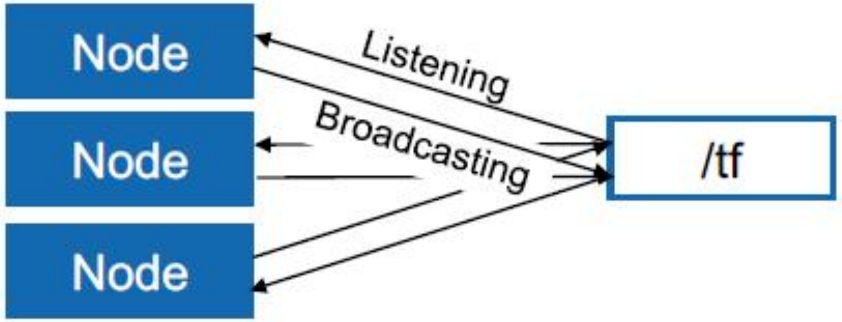
```
<link name="link_name">
  <visual>
    <geometry>
      <mesh filename="mesh.dae"/>
     </geometry>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="10"/>
    <inertia ixx="0.4" ixy="0.0" …/>
  </inertial>
</link>
```

```
<joint name="joint_name" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort="1000.0" upper="0.548" … />
  <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
  <parent link="parent_link_name"/>
  <child link="child_link_name"/>
</joint>
```

More info
http://wiki.ros.org/urdf/XML/model

TF Transformation System

- Tool for keeping track of coordinate frames over time
- Maintains relationship between coordinate frames in a tree structure buffered in time
- Lets the user transform points, vectors, etc. between coordinate frames at desired time
- Implemented as publisher/subscriber model on the topics `/tf` and `/tf_static`



Node

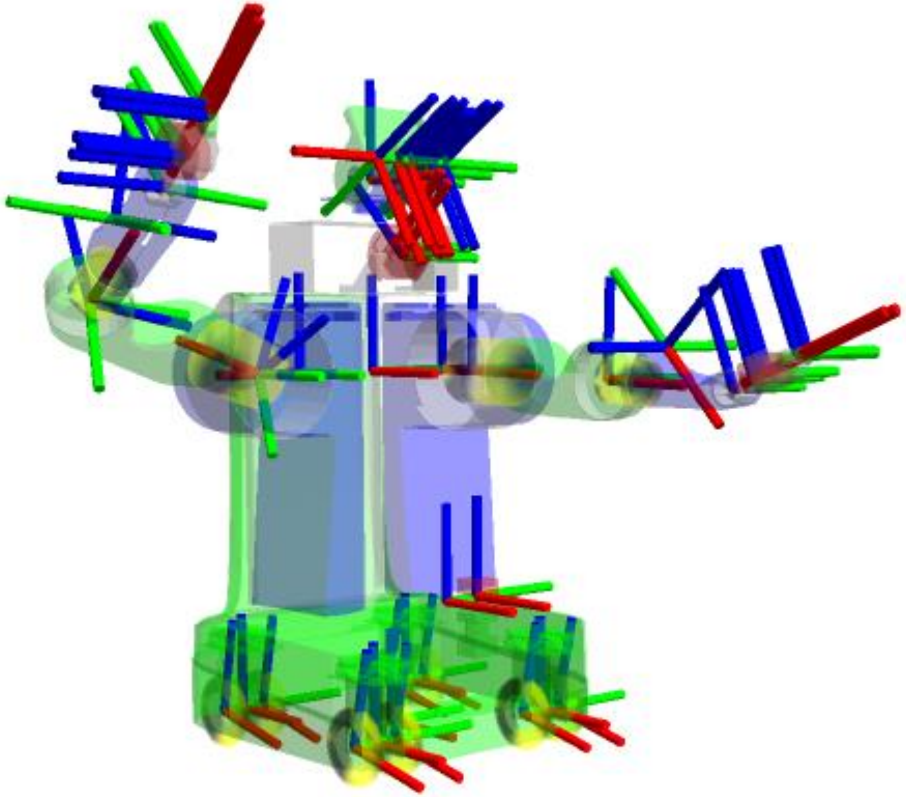
Node

Node

Listening

Broadcasting

/tf



More info
http://wiki.ros.org/tf2

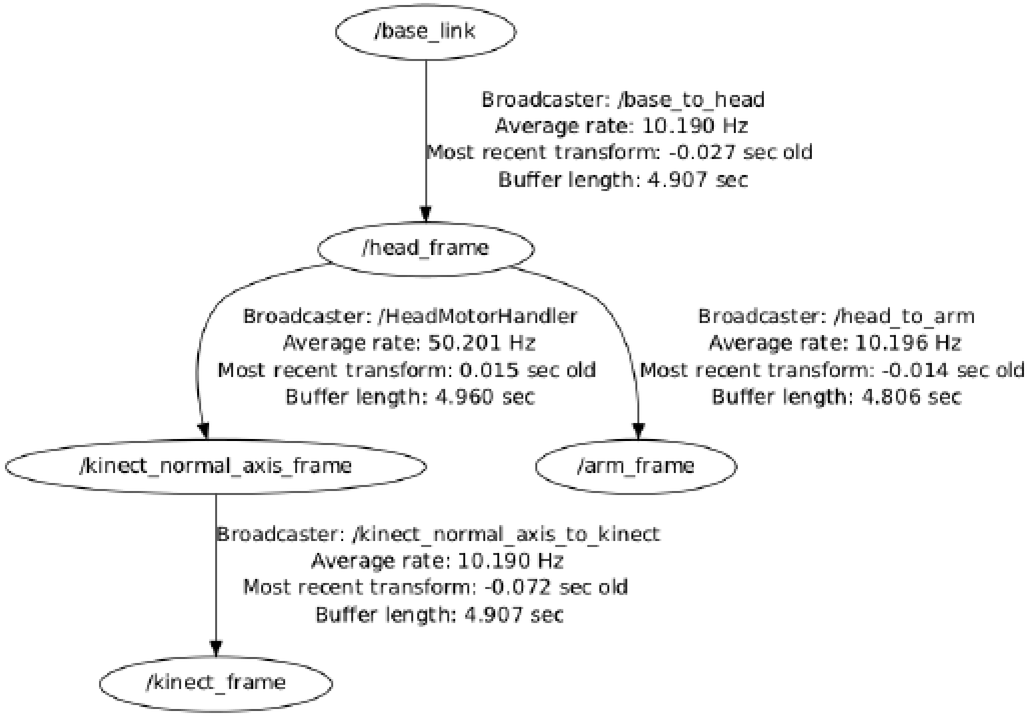
TF Transformation System

Transform Tree

- TF listeners use a buffer to listen to all broadcasted transforms
- Query for specific transforms from the transform tree

tf2_msgs/TFMessage.msg

```
geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seqtime stamp
    string frame_id
  string child_frame_id
  geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
    geometry_msgs/Quaternion rotation
```



Broadcaster: /base_to_head
Average rate: 10.190 Hz
Most recent transform: -0.027 sec old
Buffer length: 4.907 sec

Broadcaster: /HeadMotorHandler
Average rate: 50.201 Hz
Most recent transform: 0.015 sec old
Buffer length: 4.960 sec

Broadcaster: /head_to_arm
Average rate: 10.196 Hz
Most recent transform: -0.014 sec old
Buffer length: 4.806 sec

Broadcaster: /kinect_normal_axis_to_kinect
Average rate: 10.190 Hz
Most recent transform: -0.072 sec old
Buffer length: 4.907 sec

TF Transformation System

Transform Listener C++ API

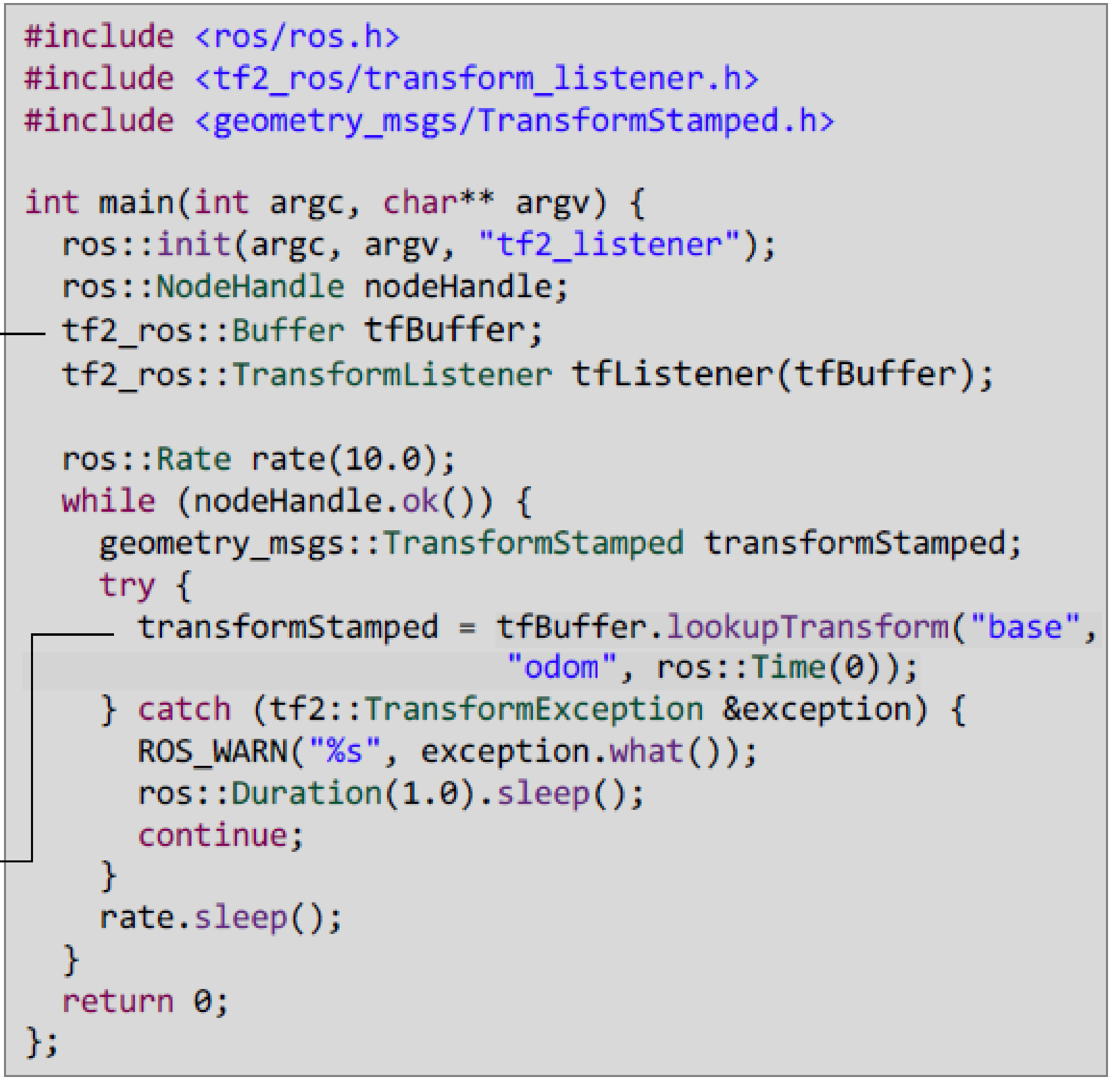
- Create a TF listener to fill up a buffer

```
tf2_ros::Buffer tfBuffer;
tf2_ros::TransformListener tfListener(tfBuffer);
```

- Make sure, that the `listener` does not run out of scope!
- To lookup transformations, use

```
geometry_msgs::TransformStamped transformStamped =
tfBuffer.lookupTransform(target_frame_id,
                         source_frame_id, time);
```

- For `time`, use `ros::Time(0)` to get the latest available transform



```
#include <ros/ros.h>
#include <tf2_ros/transform_listener.h>
#include <geometry_msgs/TransformStamped.h>

int main(int argc, char** argv) {
  ros::init(argc, argv, "tf2_listener");
  ros::NodeHandle nodeHandle;
  tf2_ros::Buffer tfBuffer;
  tf2_ros::TransformListener tfListener(tfBuffer);

  ros::Rate rate(10.0);
  while (nodeHandle.ok()) {
    geometry_msgs::TransformStamped transformStamped;
    try {
      transformStamped = tfBuffer.lookupTransform("base",
                                   "odom", ros::Time(0));
    } catch (tf2::TransformException &exception) {
      ROS_WARN("%s", exception.what());
      ros::Duration(1.0).sleep();
      continue;
    }
    rate.sleep();
  }
  return 0;
};
```

TF Transformation System

Tools

Command line

Print information about the current tranform tree

```
> rosrun tf tf_monitor
```

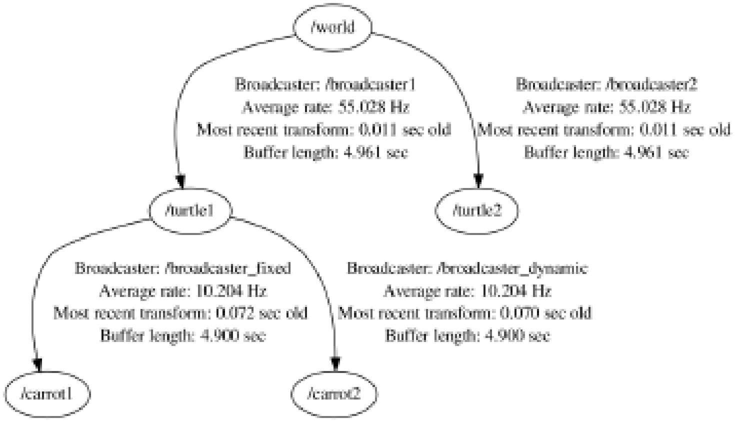
Print information about the transform between two frames

```
> rosrun tf tf_echo
    source_frame target_frame
```

View Frames

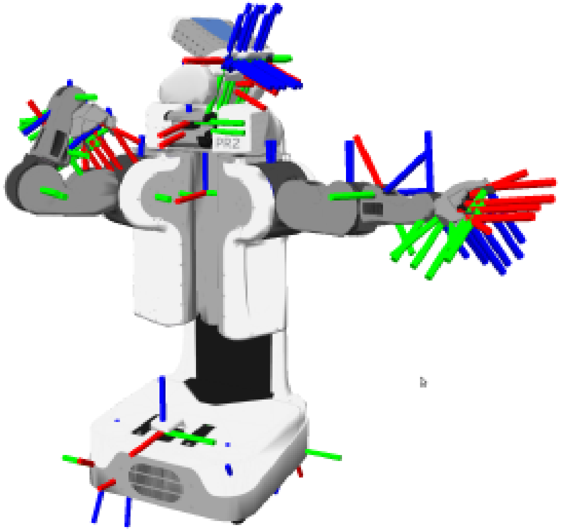
Creates a visual graph (PDF) of the transform tree

```
> rosrun tf view_frames
```


/world
Broadcaster: /broadcaster1
Average rate: 55.028 Hz
Most recent transform: 0.011 sec old
Buffer length: 4.961 sec
Broadcaster: /broadcaster2
Average rate: 55.028 Hz
Most recent transform: 0.011 sec old
Buffer length: 4.961 sec
/turtle1
/turtle2
Broadcaster: /broadcaster_fixed
Average rate: 10.204 Hz
Most recent transform: 0.072 sec old
Buffer length: 4.900 sec
Broadcaster: /broadcaster_dynamic
Average rate: 10.204 Hz
Most recent transform: 0.070 sec old
Buffer length: 4.900 sec
/carrot1
/carrot2

RViz

3D visualization of the transforms

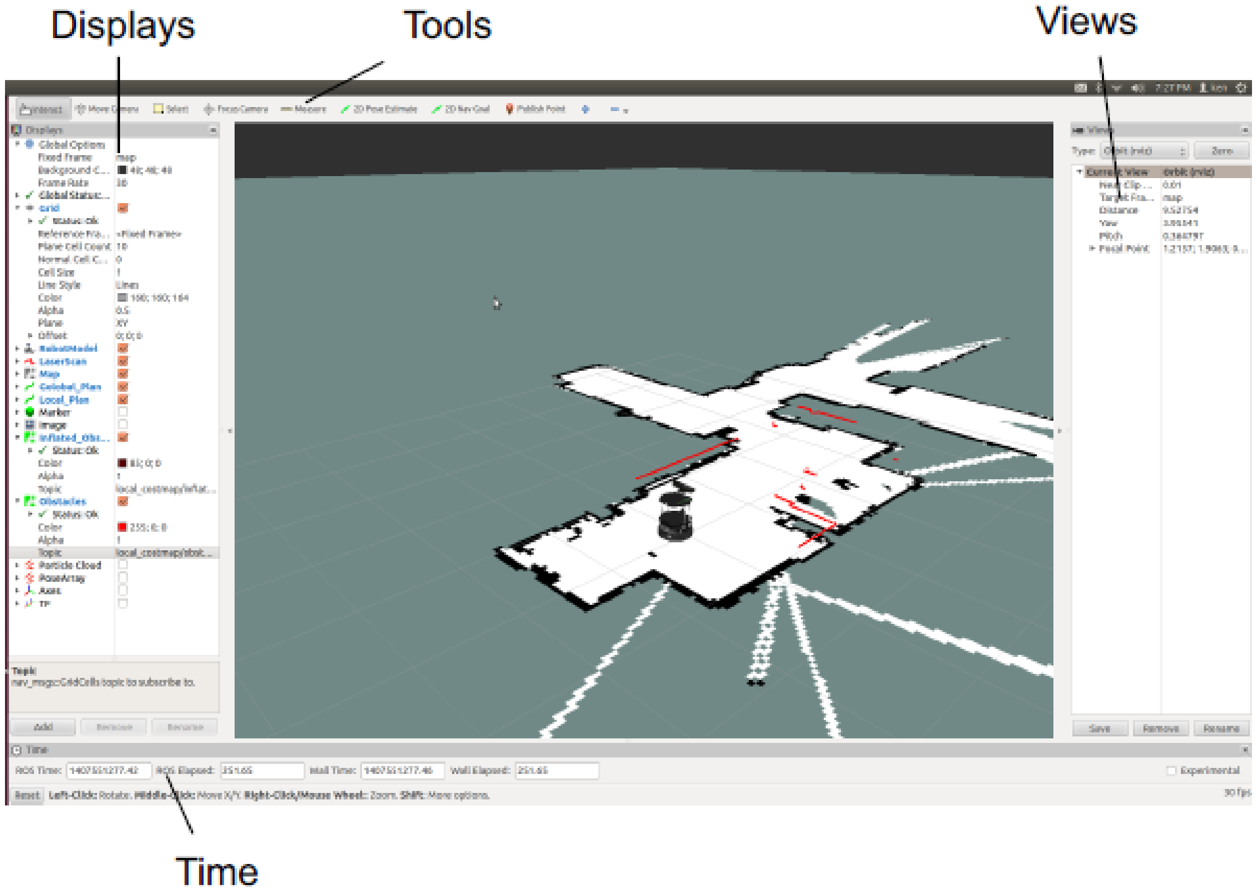


RViz

- 3D visualization tool for ROS
- Subscribes to topics and visualizes the message contents
- Different camera views (orthographic, top-down, etc.)
- Interactive tools to publish user information
- Save and load setup as RViz configuration
- Extensible with plugins

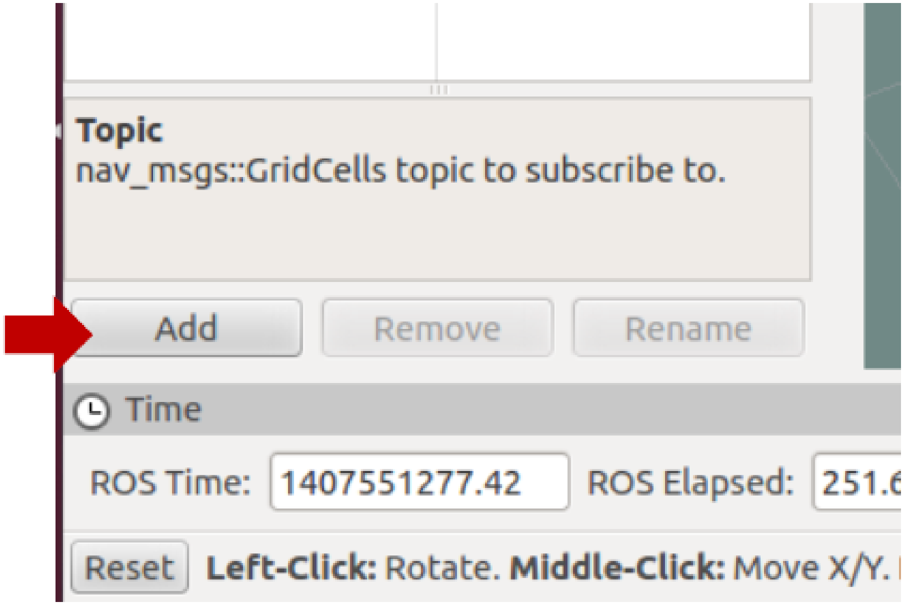
Run RViz with

```
> rosrun rviz rviz
```



More info
http://wiki.ros.org/rviz

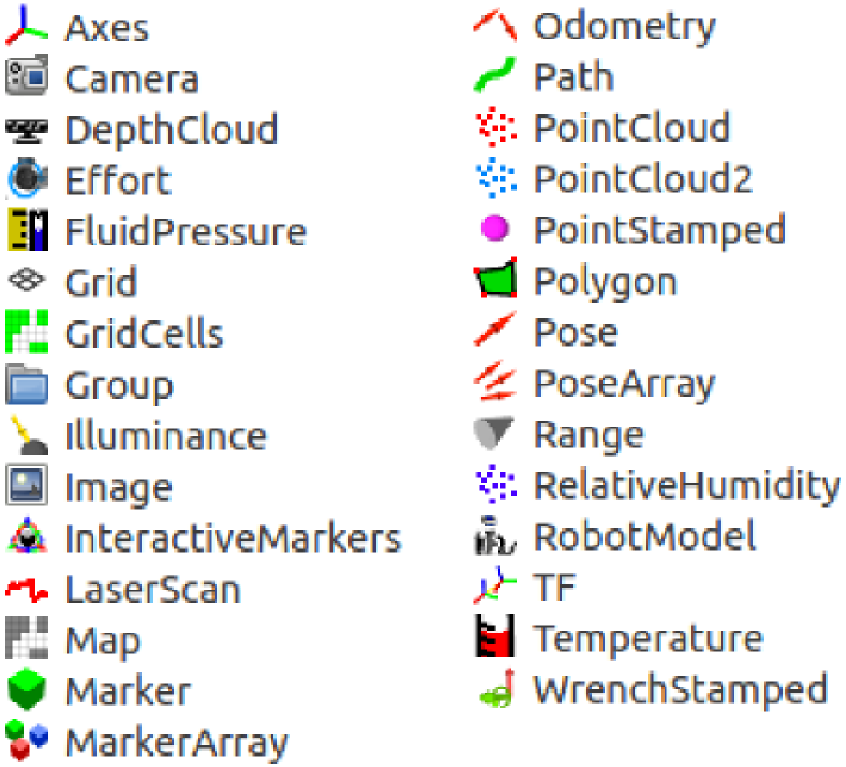
RViz Display plugin



Topic
nav_msgs::GridCells topic to subscribe to.

ROS Time: 1407551277.42 ROS Elapsed: 251.6

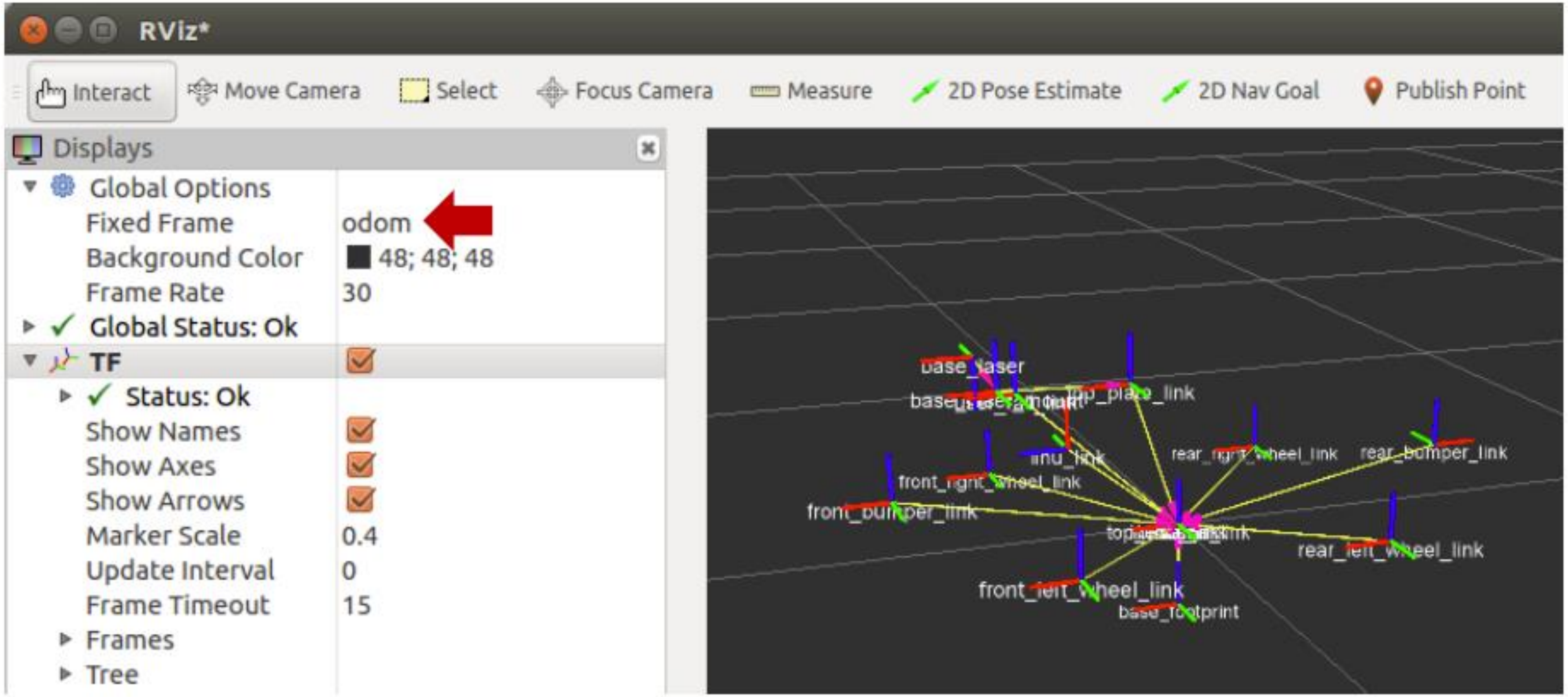
Left-Click: Rotate. **Middle-Click:** Move X/Y.



- Axes
- Camera
- DepthCloud
- Effort
- FluidPressure
- Grid
- GridCells
- Group
- Illuminance
- Image
- InteractiveMarkers
- LaserScan
- Map
- Marker
- MarkerArray
- Odometry
- Path
- PointCloud
- PointCloud2
- PointStamped
- Polygon
- Pose
- PoseArray
- Range
- RelativeHumidity
- RobotModel
- TF
- Temperature
- WrenchStamped

TF Transformation System

RViz Plugin



- Global Options
  - Fixed Frame: odom
  - Background Color: ■ 48; 48; 48
  - Frame Rate: 30
- ✓ Global Status: Ok
- TF ☑
  - ✓ Status: Ok
  - Show Names ☑
  - Show Axes ☑
  - Show Arrows ☑
  - Marker Scale: 0.4
  - Update Interval: 0
  - Frame Timeout: 15
  - Frames
  - Tree

Simulation environments in ROS

- Rviz a complex 3D visualizer, fundamental for debugging and better understanding
- It could also «animate» robotic kinematic chain (URDF models)
- Sometimes a more complete simulation is needed, including the behaviour of robots
- Gazebo is the default simulator used in ROS framework, maintained as a separate project from OSRF.
- V-REP is a robotic simulators developed by Coppelia Robotics
- It is a commercial software, that can be obtained for free in its educational version.

v-rep

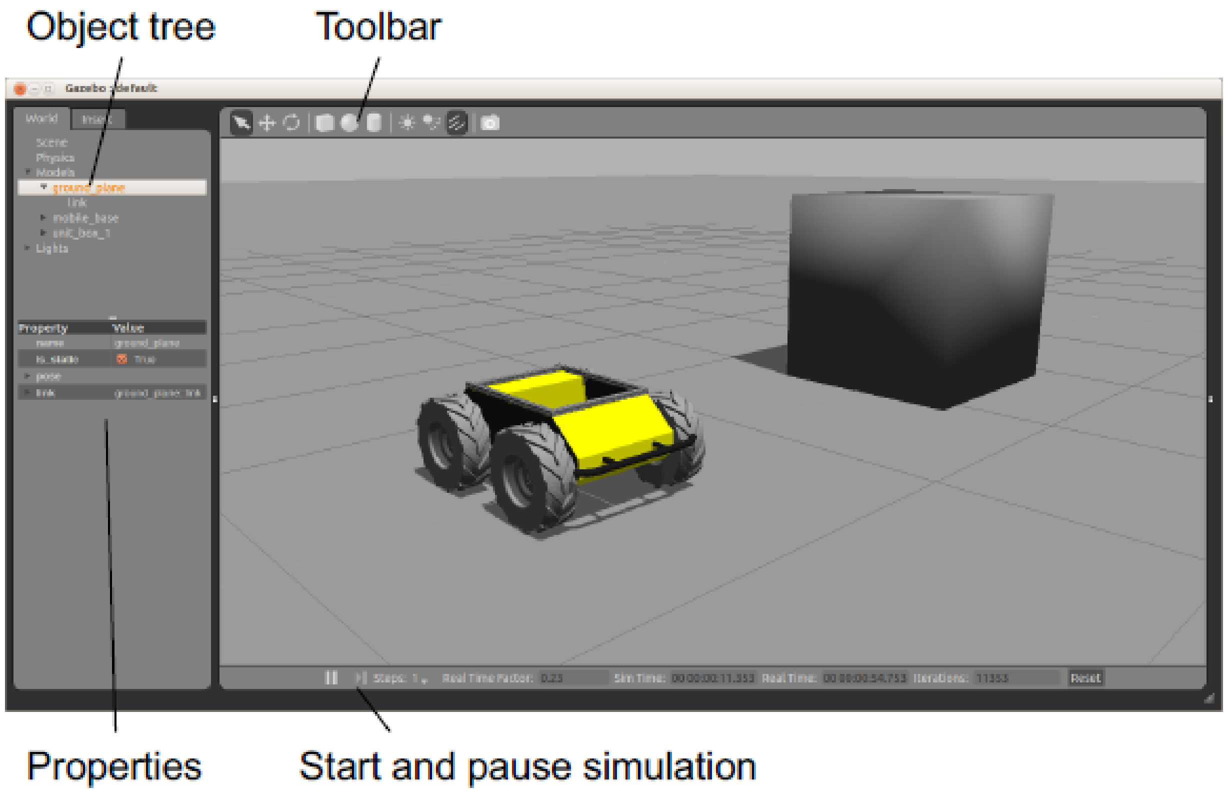
open robotics

GAZEBO

COPPELIA ROBOTICS

Gazebo Simulator

- Simulate 3d rigid-body dynamics
- Simulate a variety of sensors including noise
- 3d visualization and user interaction
- Includes a database of many robots and environments (*Gazebo worlds*)
- Provides a ROS interface
- Extensible with plugins



Property	Value
name	ground_plane
is_static	True
pose	
link	ground_plane::link

Run Gazebo with

```
> rosrun gazebo_ros gazebo
```

More info
http://gazebosim.org/
http://gazebosim.org/tutorials

~~V-rep Simulator~~
CoppeliaSim

- V-REP has support for Windows, Linux and Mac operating systems.
- It is possible to use 7 different programming languages with V-REP, the default language being Lua.
- V-REP doesn't have a native ROS node for it.
- his means that it is not yet possible to run it as a part of a ROS system in a single launchfile, but instead alongside it, in another Linux terminal.
- On the other hand, V-REP does offer a default ROS plugin that can be used in VREP Lua scripts for creating ROS publishers and subscribers..

Comparison of (main) Simulation Environments for ROS

	V-REP	GAZEBO	ARGoS Large-scale robot simulations
Physics Engines	BULLET PHYSICS LIBRARY OPEN DYNAMICS ENGINE™ newton DYNAMICS vortex by CM LABS	OPEN DYNAMICS ENGINE™	Custom 2D and 3D engines
Languages	Lua, C++, ROS, RemoteAPI	C++, ROS	Lua, C++, ROS
Threads	Spawned automatically	Two (simulator + interface)	Set by user
3D meshes	Importing, manipulation, materials	Importing, but no editing	No importing, OpenGL only
Object library	A lot of robots and other objects	A fair number of robots and other objects	A limited number of robots
Documentation	Extensive, a lot of code examples	Fairly comprehensive, some non-working code examples	Good quality but rather limited

🟩 Rich □ Neutral 🟥 Poor simulator characteristics

Simulation Scene description example: Simulation Description format (SDF)

- Defines an XML format to describe
  - Environments (lighting, gravity etc.)
  - Objects (static and dynamic)
  - Sensors
  - Robots
- SDF is the standard format for Gazebo
- Gazebo converts a URDF to SDF automatically

More info
http://sdformat.org

ROS Time

- Normally, ROS uses the PC's system clock as time source (*wall time*)
- For simulations or playback of logged data, it is convenient to work with a simulated time (pause, slow-down etc.)
- To work with a simulated clock:
  - Set the `/use_sim_time` parameter

    ```
    > rosparam set use_sim_time true
    ```

  - Publish the time on the topic `/clock` from
    - Gazebo (enabled by default)
    - ROS bag (use option `--clock`)

- To take advantage of the simulated time, you should always use the ROS Time APIs:
  - **`ros::Time`**

    ```
    ros::Time begin = ros::Time::now();
    double secs = begin.toSec();
    ```

  - **`ros::Duration`**

    ```
    ros::Duration duration(0.5); // 0.5s
    ```

  - **`ros::Rate`**

    ```
    ros::Rate rate(10); // 10Hz
    ```

- If wall time is required, use `ros::WallTime`, `ros::WallDuration`, and `ros::WallRate`

More info
http://wiki.ros.org/Clock
http://wiki.ros.org/roscpp/Overview/Time

ROS Bags

- A *bag* is a format for storing message data
- Binary format with file extension *.bag
- Suited for logging and recording datasets for later visualization and analysis

Record all topics in a bag

```
> rosbag record --all
```

Record given topics

```
> rosbag record topic_1 topic_2 topic_3
```

Stop recording with `Ctrl` + `C`

Bags are saved with start date and time as file name in the current folder (e.g. `2017-02-07-01-27-13.bag`)

Show information about a bag

```
> rosbag info bag_name.bag
```

Read a bag and publish its contents

```
> rosbag play bag_name.bag
```

Playback options can be defined e.g.

```
> rosbag play --rate=0.5 bag_name.bag
```

`--rate=`*`factor`*	Publish rate factor
`--clock`	Publish the clock time (set param `use_sim_time` to `true`)
`--loop`	Loop playback
	etc.

More info
http://wiki.ros.org/rosbag/Commandline

Debugging strategies

Debug with the tools you have learned

- Compile and run code often to catch bugs early
- Understand compilation and runtime error messages
- Use analysis tools to check data flow (`rosnode info`, `rostopic echo`, `roswtf`, `rqt_graph` etc.)
- Visualize and plot data (RViz, RQT Multiplot etc.)
- Divide program into smaller steps and check intermediate results (`ROS_INFO`, `ROS_DEBUG` etc.)
- Make your code robust with argument and return value checks and catch exceptions
- If things don't make sense, clean your workspace

```
> catkin clean --all
```

Learn new tools

- Build in *debug* mode and use GDB or Valgrind

```
> catkin config --cmake-args
                    -DCMAKE_BUILD_TYPE=Debug
```

- Use Eclipse breakpoints
- Maintain code with unit tests and integration tests

More info
http://wiki.ros.org/UnitTesting
http://wiki.ros.org/gtest
http://wiki.ros.org/rostest
http://wiki.ros.org/roslaunch/Tutorials/Roslaunch%20Nodes%20in%20Valgrind%20or%20GDB

ROS Best practice (1)

You should follow all the rules/recommendations described in the previous presentations!

For example:

- Check for available solutions (packages, nodes, …),
- Understand design pattern underneath successful and working packages/stack
- Define common units, please refer to Standard Units of Measure and Coordinate Conventions.
- Test your code (push only tested code on the shared repository)

Albert is watching you!

ROS Best practice (2)

Messages

- Check if common messages are already available: https://github.com/ros/common_msgs
- Create separate packages that contain only messages, services and actions (separation of interface and implementation).
- Do not define a new msg/srv/action definition for each topic/service/action!
- Complex messages are built through composition (e.g. geometry_msgs/PoseWithCovarianceStamped).
- Try to avoid building messages that tend to not get completely filled out.

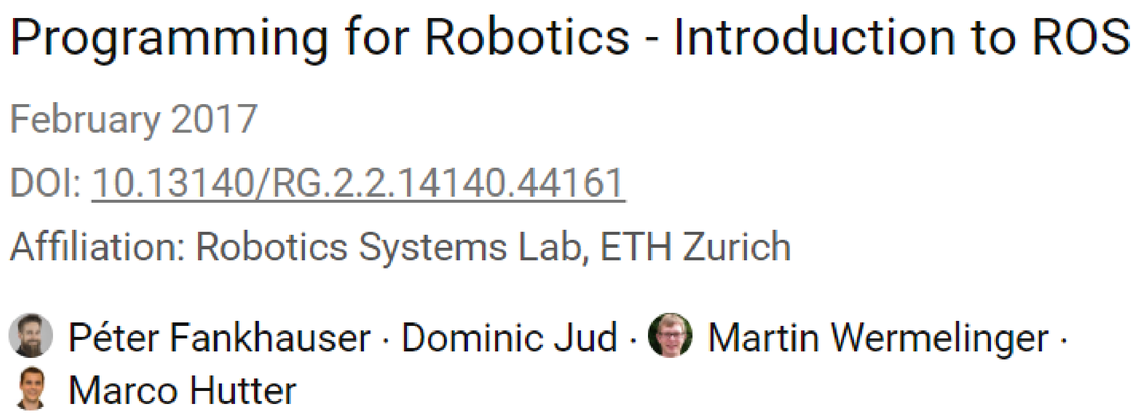
ROS Best practice (3)

Package Organization

- The overhead of a ROS package is not large. Define separate packages wherever they make sense. Pay attention to avoid over
- The package dependency graph must be acyclic, i.e. no package may depend on another that directly or indirectly depends on it. → Avoid combining nodes that pull in mutually unneeded dependencies and are often used separately (to eliminate unnecessary build overhead).
- **Create separate packages that contain only messages, services and actions**
- Group related packages in stacks.
- Package Names → Choose the name carefully:
    - They are messy to change later.
    - Package names are global to the entire ROS ecosystem.
    - Try to pick names that will make sense to others who may wish to use your code.

Questions?

The contents of these slides are partially based on:



Programming for Robotics - Introduction to ROS

February 2017

DOI: 10.13140/RG.2.2.14140.44161

Affiliation: Robotics Systems Lab, ETH Zurich

Péter Fankhauser · Dominic Jud · Martin Wermelinger · Marco Hutter

Please check also: https://github.com/leggedrobotics/ros_best_practices/wiki