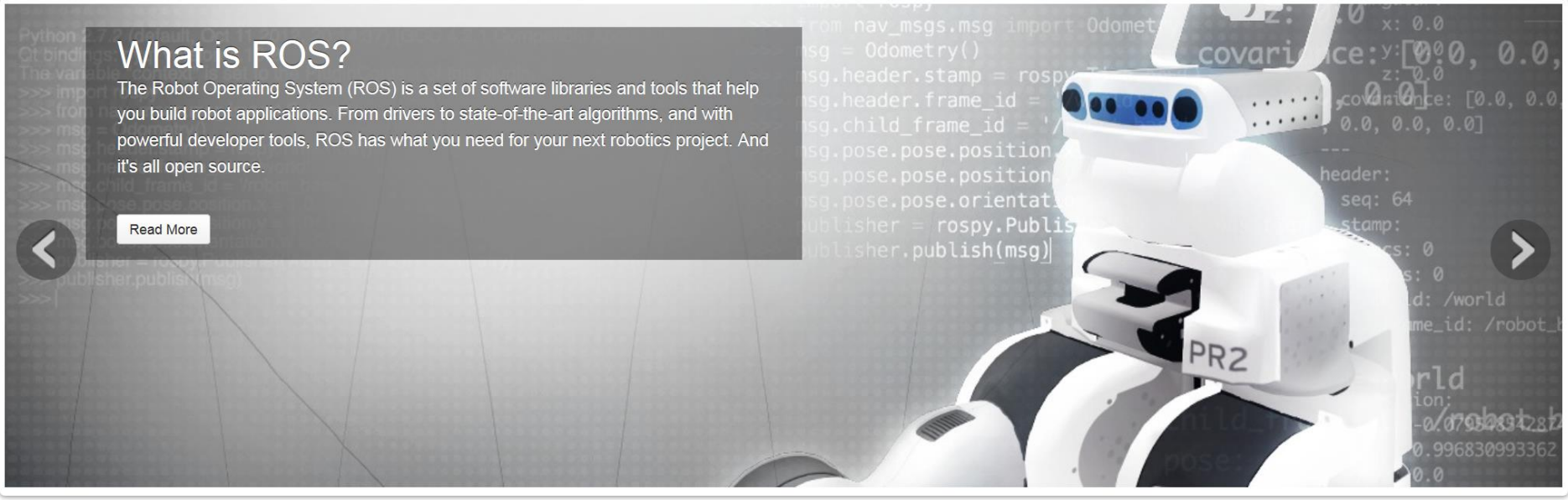
Aalto University
School of Engineering

ELEC-E8111: Autonomous mobile robots

Introduction to ROS for robot project

Mika Vainio
Arto Visala
(Andrei Sandru)

ROS.org



ROS Melodic Morenia

Melodic Morenia is the 12th official ROS release. It is supported on Ubuntu Artful and Bionic, along with Debian Stretch. Get Melodic Morenia now!

Download

ROS Lunar Loggerhead

Lunar Loggerhead is the 11th official ROS release. It is supported on Ubuntu Xenial, Yakkety and Zesty. Get Lunar Loggerhead now!

Download

ROS Kinetic Kame

Kinetic Kame is the 10th official ROS release. It is supported on Ubuntu Wily and Xenial. Get Kinetic Kame now!

Download

ROS?

ROS (Robot **O**perating **S**ystem) 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.

SORCE: http://wiki.ros.org/ROS/Introduction

ROS.org

Distributed computation

Many modern robot systems rely on software that spans many different processes and runs across several different computers:

- Some robots carry multiple computers, each of which controls a subset of the robot's sensors or actuators.
- Even within a single computer, it's often a good idea to divide the robot's software into small, stand-alone parts that cooperate to achieve the overall goal.
- When multiple robots attempt to cooperate on a shared task, they often need to communicate with one another to coordinate their efforts.
- Human users often send commands to a robot from a laptop, a desktop computer, or mobile device. We can think of this human interface as an extension of the robot's software.

SORCE: https://www.cse.sc.edu/~jokane/agitr/

Software reuse

The rapid progress of robotics research has resulted in a growing collection of good algorithms for common tasks such as navigation, motion planning, mapping, and many others. The existence of these algorithms is only truly useful if there is a way to apply them in new contexts, without the need to reimplement each algorithm for each new system.

ROS can help in at least two important ways:

1) ROS's standard packages provide stable, debugged implementations of many important robotics algorithms.

2) ROS's message passing interface is becoming a *de facto* standard for robot software interoperability, which means that ROS interfaces to both the latest hardware and to implementations of cutting edge algorithms are quite often available. For example, the ROS website lists hundreds of publicly-available ROS packages. This sort of uniform interface greatly reduces the need to write "glue" code to connect existing parts.

SORCE: https://www.cse.sc.edu/~jokane/agitr/

Rapid testing

One of the reasons that software development for robots is often more challenging than other kinds of development is that testing can be time consuming and error-prone. Physical robots may not always be available to work with, and when they are, the process is sometimes slow and finicky. Working with ROS provides two effective workarounds to this problem:

1) Well-designed ROS systems separate the low-level direct control of the hardware and high-level processing and decision making into separate programs. Because of this separation, we can temporarily replace those low-level programs (and their corresponding hardware) with a simulator, to test the behavior of the high-level part of the system.

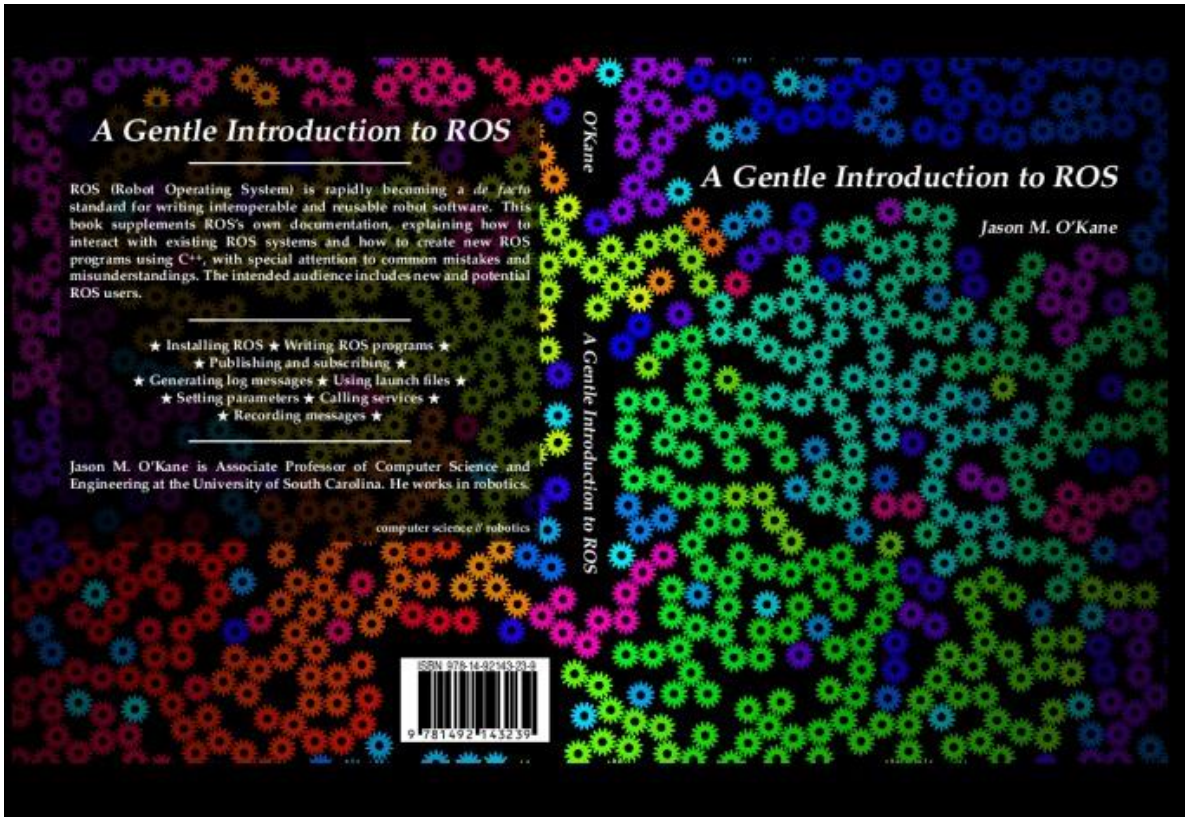
2) ROS also provides a simple way to record and play back sensor data and other kinds of messages. By recording the robot's sensor data, we can replay it many times to test different ways of processing that same data. In ROS, these recordings are called "bags" and a tool called rosbag is used to record and replay them.

SORCE: https://www.cse.sc.edu/~jokane/agitr/

A Gentle Introduction to ROS

Jason M. O'Kane

This book supplements ROS's own documentation, explaining how to interact with existing ROS systems and how to create new ROS programs using C++, with special attention to common mistakes and misunderstandings. The intended audience includes new and potential ROS users.



166 pages
ISBN 978-14-92143-23-9

Printed copies are available from Amazon.com

Electronic copies are free and available from the author's website:
https://www.cse.sc.edu/~jokane/agitr/

Contents

- Overview
- ROS Components
- How ROS communication works
- ROS Tools
- Helpful to know
- Examples and Demonstrations

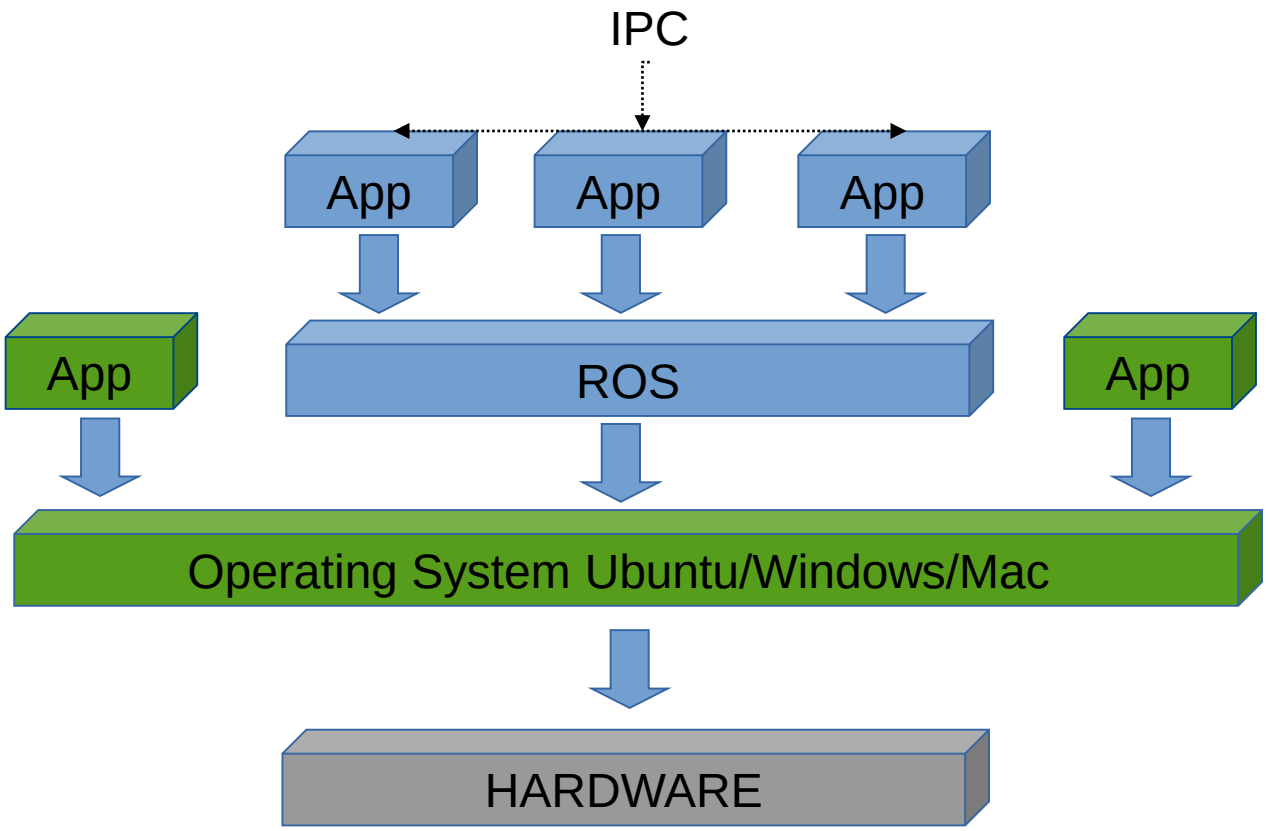
ROS.org

Overview: What is ROS? ...

- Robot Operating System
  - A “meta” operating system for robots
  - A collection
    - Packaging (over 3000 packages!)
    - Software building and data analysis tools
  - Distributed Communication architecture (inter-process / inter-machine)
  - Language independent architecture
  - Constantly increasing community

ROS.org

Overview: What is ROS?



ROS.org

Who made it ?

Stanford

Various efforts at Stanford University in the mid-2000s involving integrative, embodied AI, such as the STanford AI Robot (STAIR) and the Personal Robots (PR) program, created in-house prototypes of flexible, dynamic software systems intended for robotics use.

ROS.org

Willow Garage

Image taken from https://www.willowgarage.com/

- ROS released in January 2010
- Privately owned company
- Based in Menlo Park, California
- Strong open source commitment
- Shut down in 2014

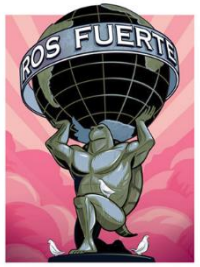
open robotics

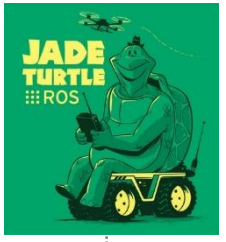
Open Source Robotics Foundation, Inc

At Open Robotics, we work with industry, academia, and government to create and support open software and hardware for use in robotics, from research and education to product development.

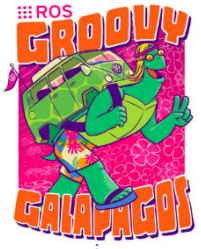
ROS Distribution Releases

January 2010
March 2010
August 2010
March 2011
August 2011
April 2012
December 2012
September 2013
July 2014
May 2015
May 2016
May 2017
May 2018





ROS 1.0
http://ros.org/

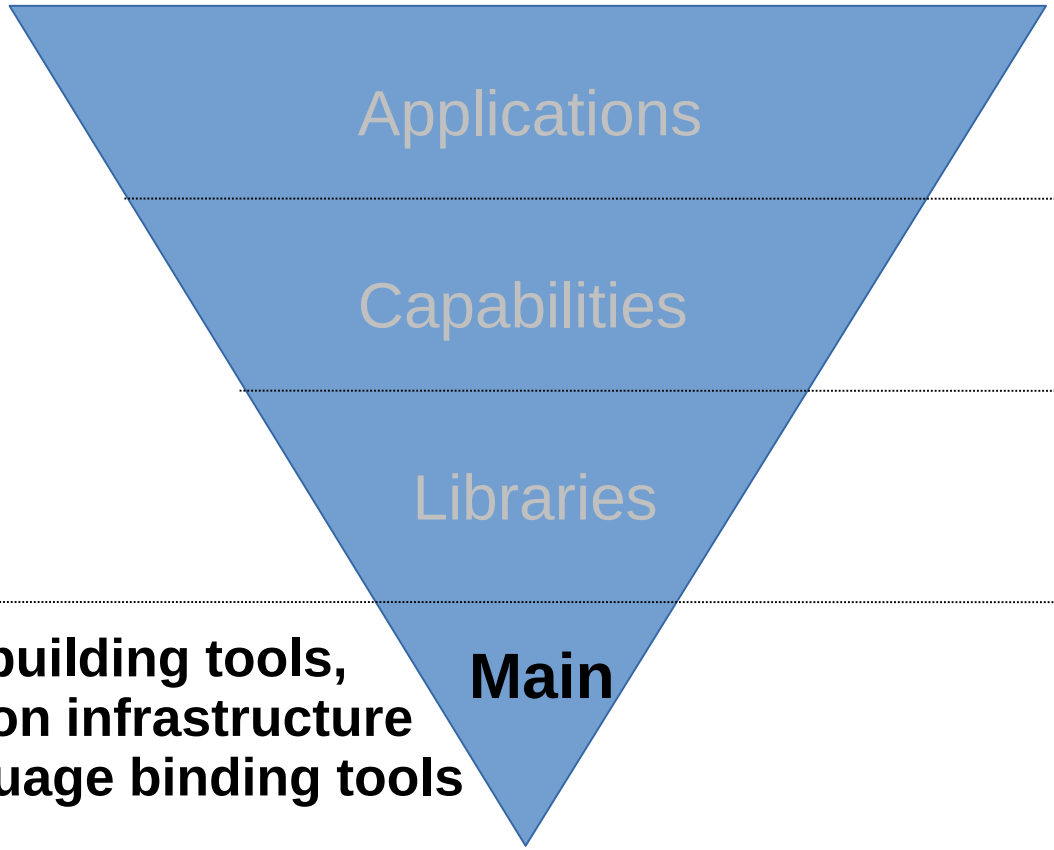


ROS
Melodic
Morenia

ROS.org

All ros logos taken from http://wiki.ros.org

What does ROS get you ?



Finding and fetching action figures

Grasping, control, execution, navigation ...

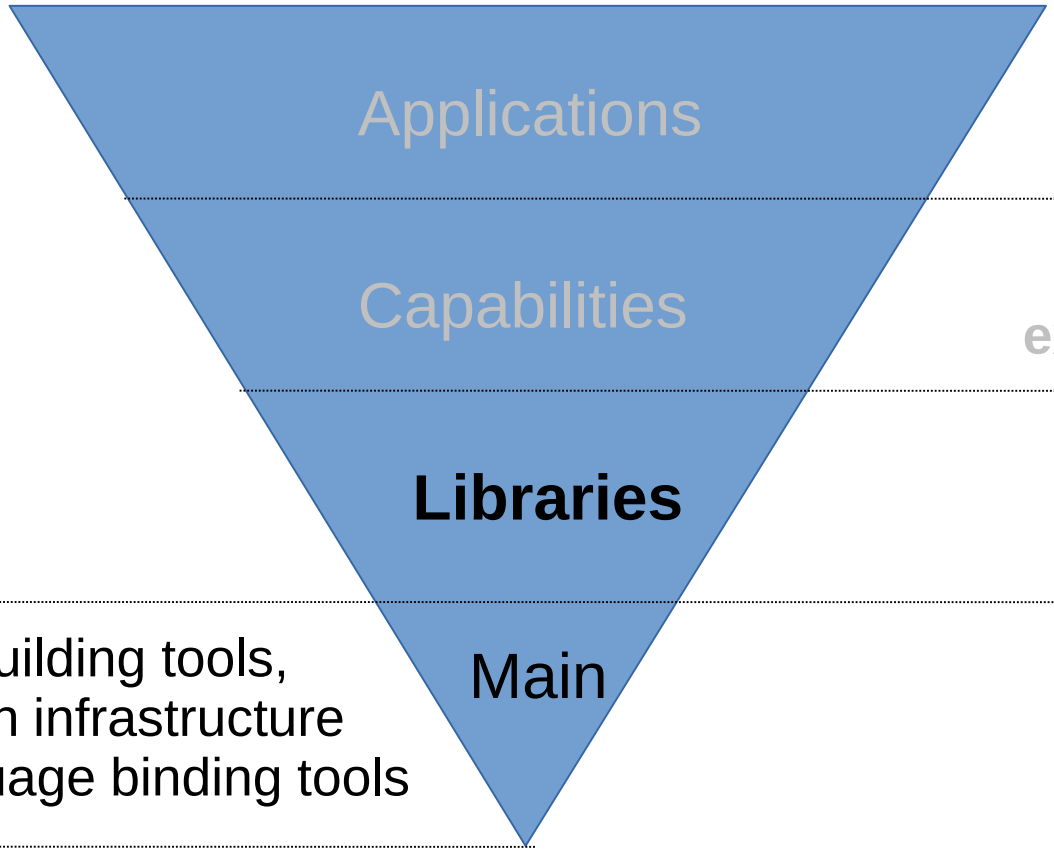
tf, OpenCV, PCL, KDL, Hardware drivers

Packaging & building tools,
Communication infrastructure
ROS API language binding tools

ROS

ROS.org

What does ROS get you ?



Finding and fetching action figures

Grasping, control, execution, navigation ...

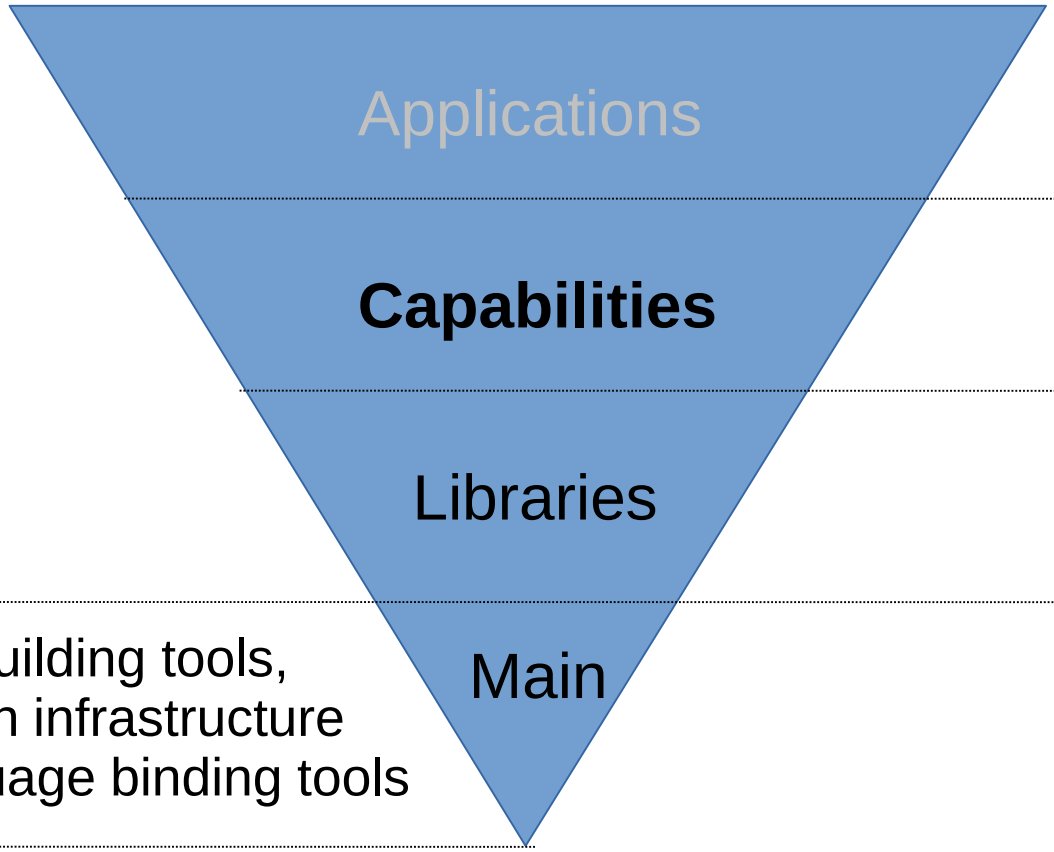
tf, OpenCV, PCL, KDL, Hardware drivers

Packaging & building tools,
Communication infrastructure
ROS API language binding tools

ROS

ROS.org

What does ROS get you ?



Finding and fetching action figures

Grasping, control, execution, navigation ...

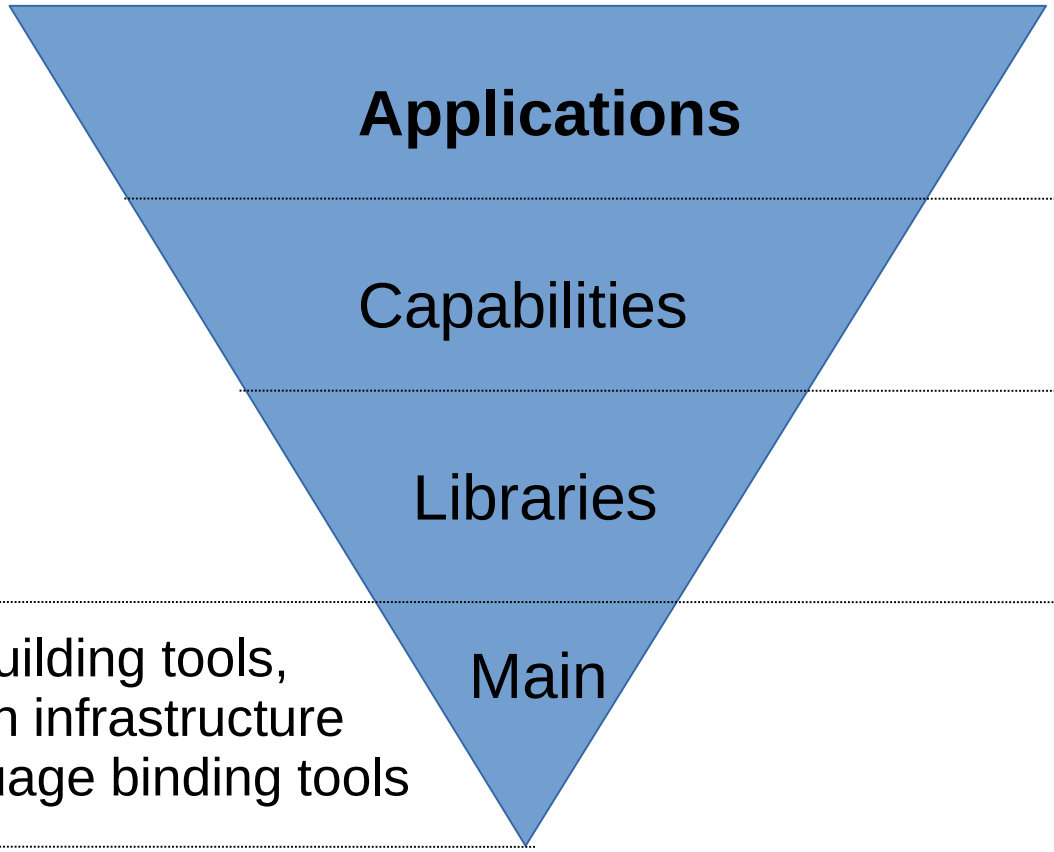
tf, OpenCV, PCL, KDL, Hardware drivers

Packaging & building tools,
Communication infrastructure
ROS API language binding tools

ROS

ROS.org

What does ROS get you ?



Finding and fetching action figures

Grasping, control, execution, navigation ...

tf, OpenCV, PCL, KDL, Hardware drivers

Packaging & building tools,
Communication infrastructure
ROS API language binding tools

ROS

ROS.org

ROS Components

- **Computational Graph**
  - **Master**
    - The ROS Master provides naming and registration services to the rest of the nodes.
  - **Nodes**
    - A node is a process that performs computation.
  - **Parameter Server**
    - A parameter server is a shared, multi-variate dictionary that is accessible via network APIs.
  - **Message**
    - Nodes communicate with each other by publishing messages to topics.
  - **Topic**
    - Topics are named buses over which nodes exchange messages.
  - **Services**
    - Request / reply blocking call.
  - **Bags**
    - A bag is a file format in ROS for storing ROS message data.

ROS.org

ROS Components

- **File System**
  - <root file system>
    - /opt/ros/<distro>/
      - bin
      - include
      - lib
      - share
      - etc...
  - <workspace>
    - build
      - Contains make and cmake generated files
    - devel
      - Contains same directory structure as root file system
    - src
      - <projects source and configurations> ...

ROS.org

ROS Components

- **Packages in workspace**

```
src/
    CMakeLists.txt          -- The “toplevel” CMake file
    package_1/              -- Package containing source code
        CMakeLists.txt      -- For CMake: Describes how to build the code and where
                            -- to install it.
        package.xml         -- Package description, including dependencies
        …
    package_2/
        CMakeLists.txt
        package.xml
        …
    …
```

ROS.org

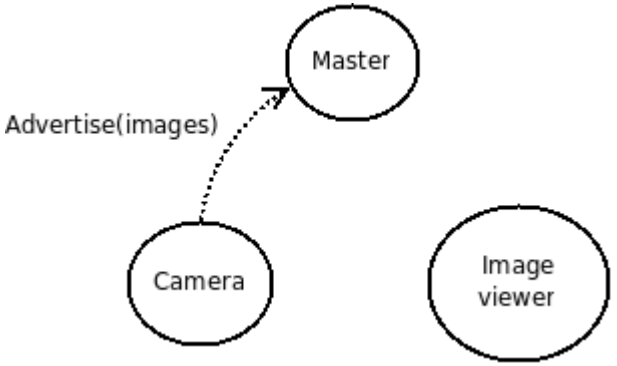
How ROS communication works

- Entities and terminology
- Communication models
  - Publisher/Subscriber model (Synchronous)
  - Service/Client model (Asynchronous)
- Pioneer 3DX case Example

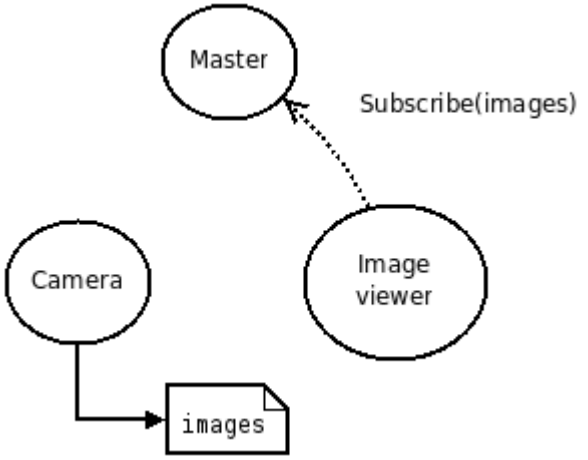
ROS.org

Master Name Service Example

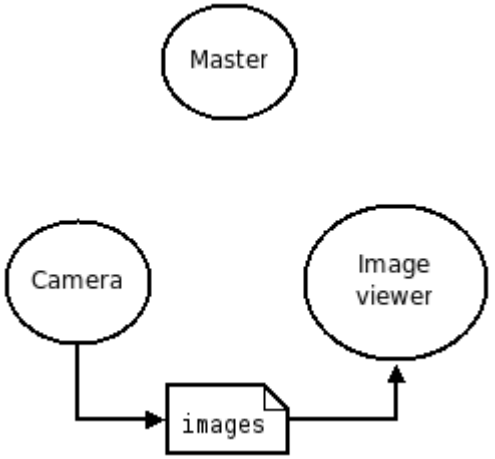
For instance, let's say we have two Nodes; a Camera node and an Image_viewer node. A typical sequence of events would start with Camera notifying the master that it wants to publish images on the topic "images":



Now, Camera publishes images to the "images" topic, but nobody is subscribing to that topic yet so no data is actually sent. Now, Image_viewer wants to subscribe to the topic "images" to see if there's maybe some images there:

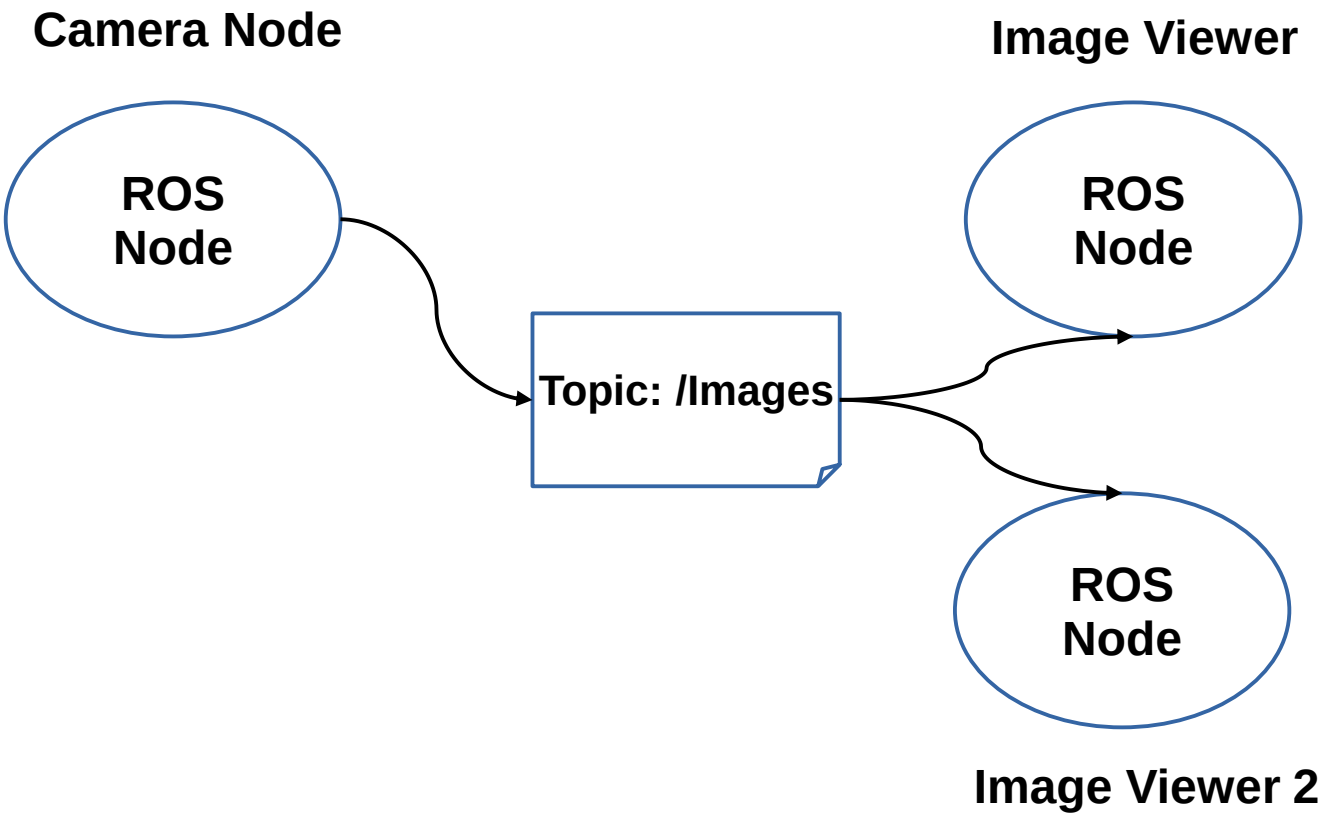


Now that the topic "images" has both a publisher and a subscriber, the master node notifies Camera and Image_viewer about each others existence so that they can start transferring images to one another:



ROS.org

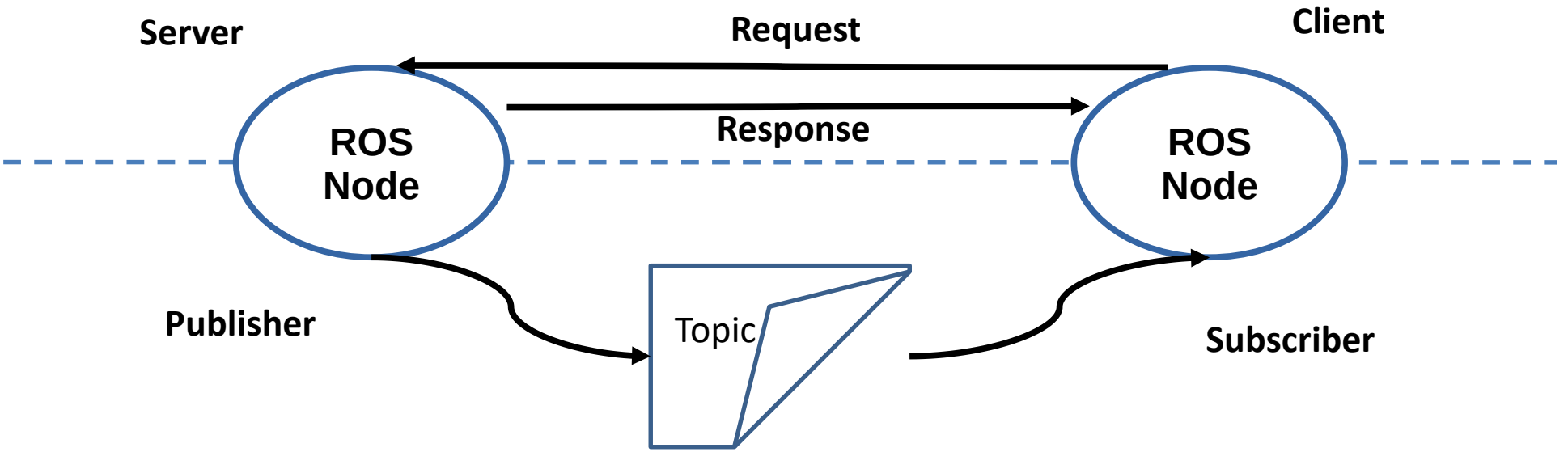
ROS communications: entities



ROS.org

ROS Communication

Synchronous



Asynchronous

ROS.org

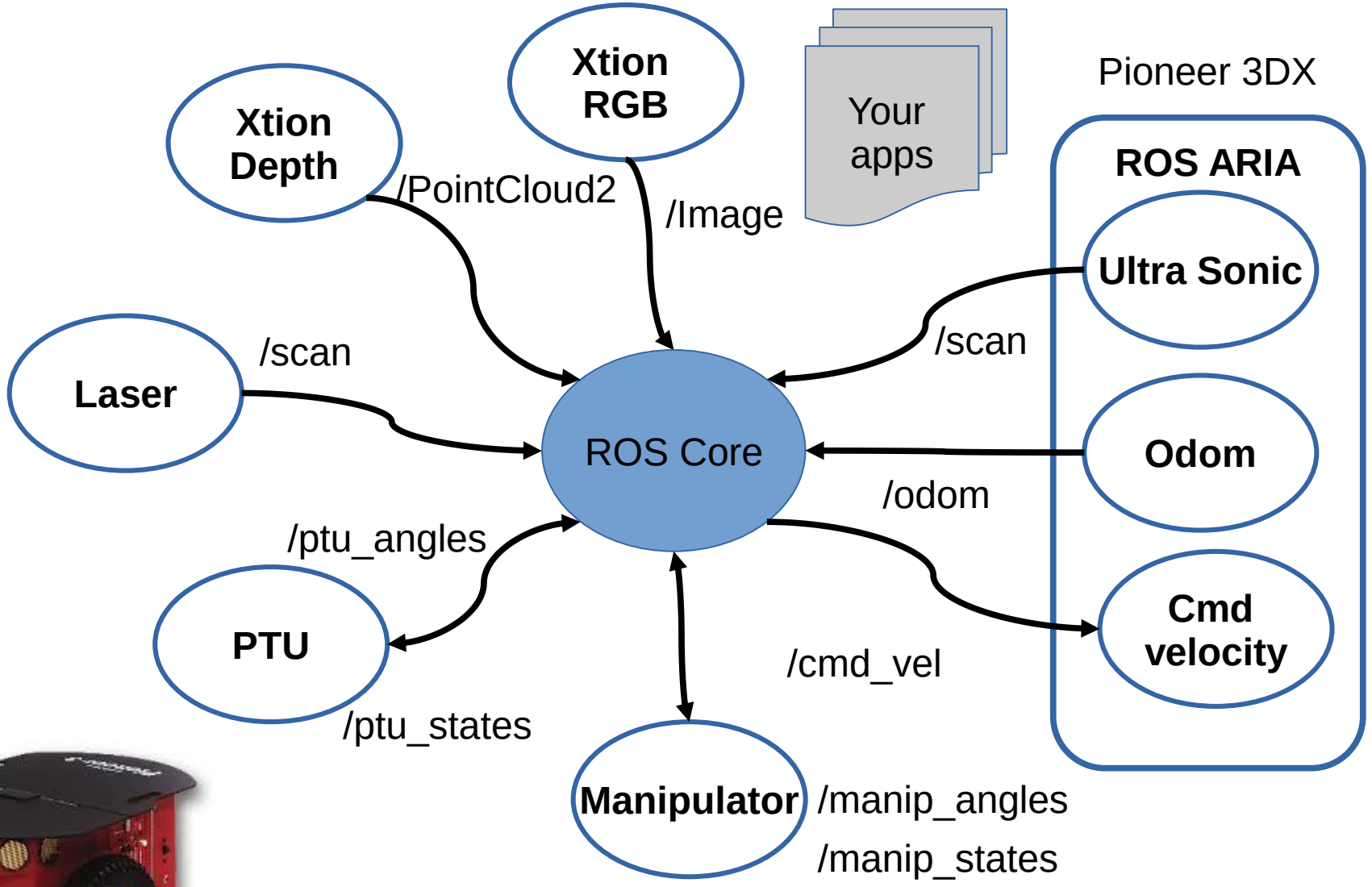
Pioneer 3DX case example

Adept MobileRobots was the manufacturer of Pioneer. It was formerly known as ActivMedia Robotics and MobileRobots Inc. It was active from the mid 1990s until 2018.

2D Sick Laser scanner

Asus Xtion camera (RGB+D)

Pioneer 3DX case example



ROS.org

ROS Tools

- roscore
- roscd
- rosmsg
- rostopic
- rosservice
- roswtf
- rosrun
- roslaunch
- rviz
- rosed
- rosparam
- rqt_logger_level
- rqt_console
- rqt_graph
- catkin_create_pkg
- catkin_make

ROS.org

rostopic

The rostopic command-line tool displays information about ROS topics. Currently, it can display a list of active topics, the publishers and subscribers of a specific topic, the publishing rate of a topic, the bandwidth of a topic, and messages published to a topic. The display of messages is configurable to output in a plotting-friendly format.

This is the current list of supported commands:

rostopic bw	display bandwidth used by topic
rostopic delay	display delay for topic which has header
rostopic echo	print messages to screen
rostopic find	find topics by type
rostopic hz	display publishing rate of topic
rostopic info	print information about active topic
rostopic list	print information about active topics
rostopic pub	publish data to topic
rostopic type	print topic type

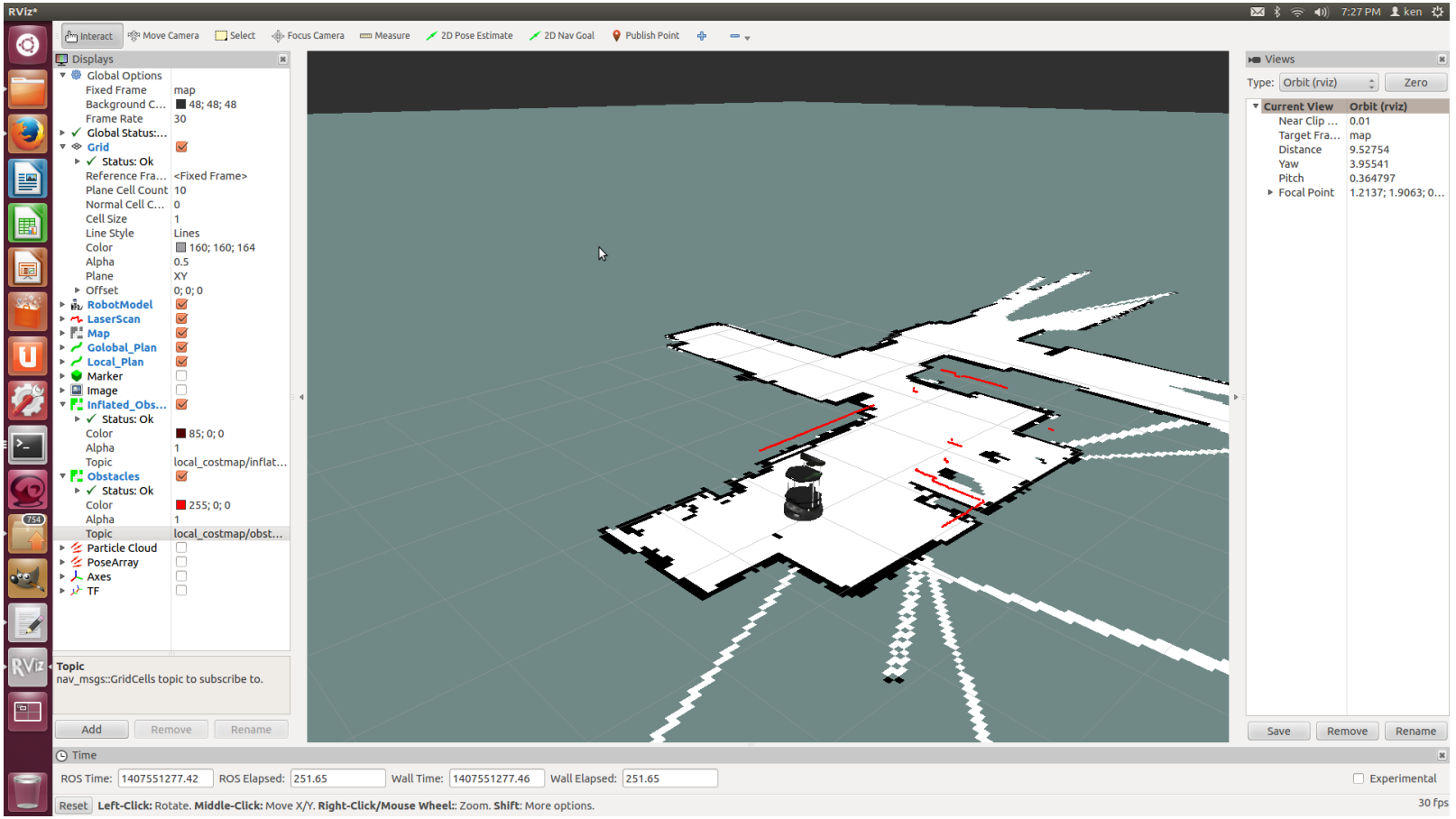
roswtf

roswtf is a tool for diagnosing issues with a running ROS system. Think of it as a FAQ implemented in code.

roswtf looks for many, many things, and the list is always growing. There are two categories of what it looks for: file-system issues and online/graph issues.

For file-system issues, roswtf looks at your environment variables, package configurations, stack configurations, and more. It can also take in a roslaunch file and attempt to find any potential configuration issues in it, such as packages that haven't been built properly.

For online issues, roswtf examines the state of your current graph and tries to find any potential issues. These issues might be unresponsive nodes, missing connections between nodes, or potential machine-configuration issues with roslaunch.

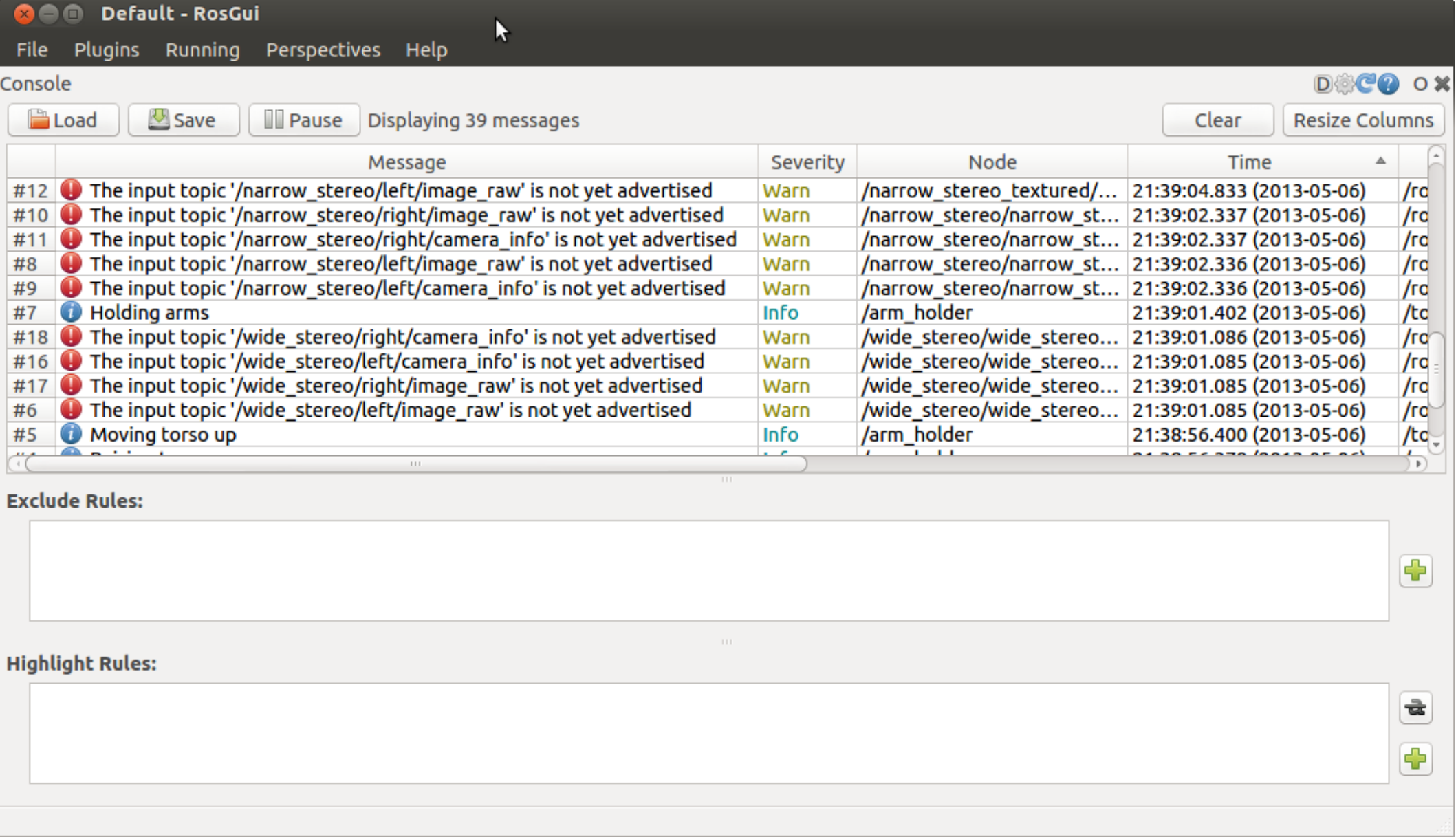


rviz

rviz is 3D visualizer for displaying sensor data and state information from ROS. Using rviz, you can visualize your robot's current configuration on a virtual model of the robot. You can also display live representations of sensor values coming over ROS Topics including camera data, infrared distance measurements, sonar data, and more.

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

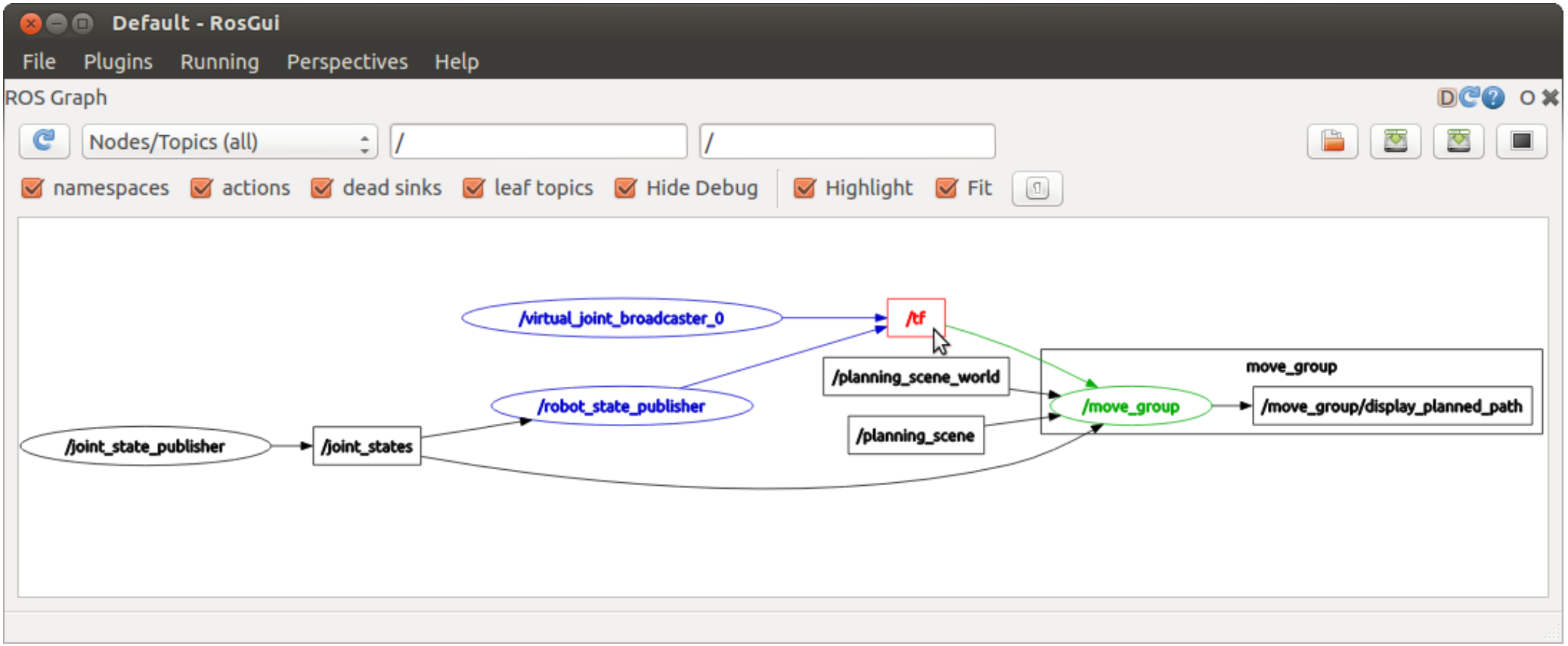
ROS.org



rqt_console

Provides a GUI plugin for displaying and filtering ROS messages.

ROS.org



rgt_graph

rqt_graph provides a GUI plugin for visualizing the ROS computation graph. It visualizes the publish subscribe relationships between ROS nodes

ROS.org

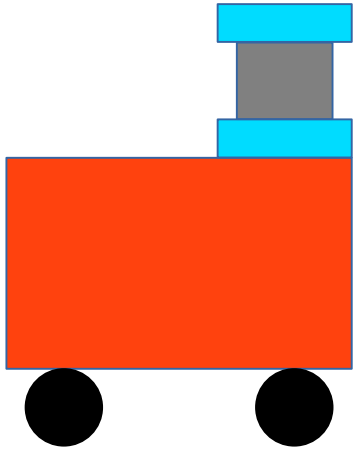
Helpful to know

- ROS On Distributed Machines
- Importance of TF Library
- ROS Launch files
- Gazebo

ROS.org

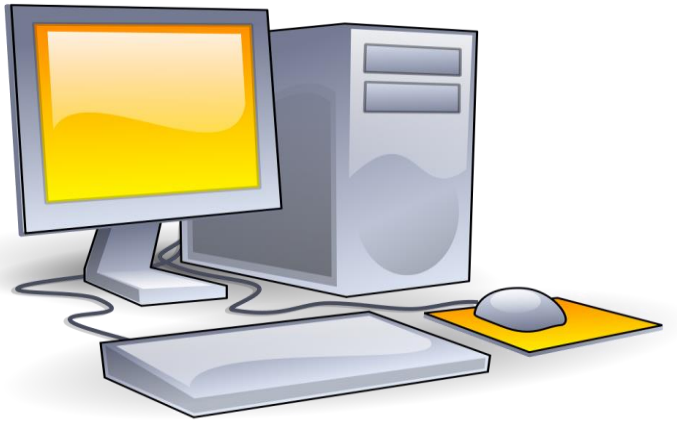
ROS On Distributed Machines

robot_IP = 192.168.1.110
ROS_MASTER_URI = 192.168.1.110
ROS_IP = 192.168.1.110



Mobile Robot

Computer_IP = 192.168.1.116
ROS_MASTER_URI = 192.168.1.110
ROS_IP = 192.168.1.116

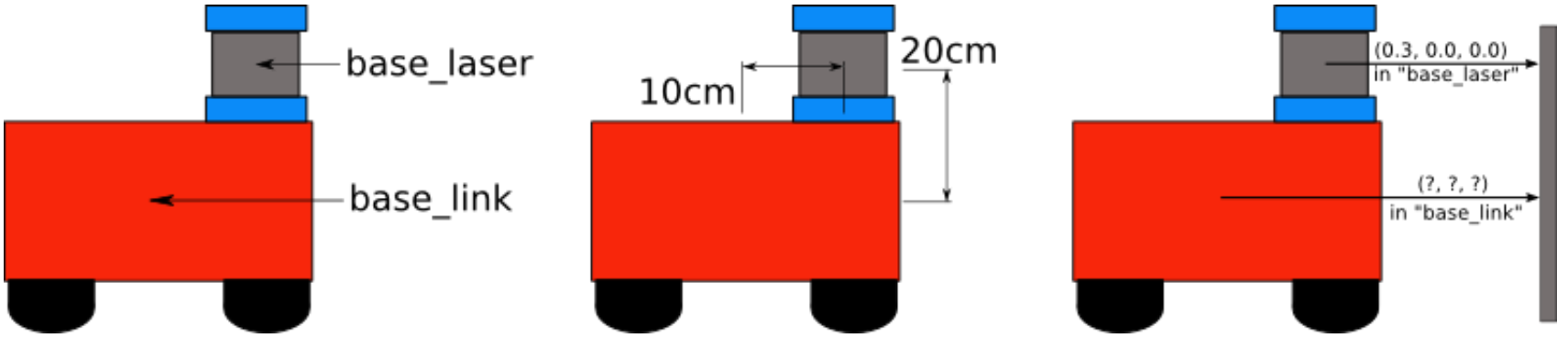


Teleop Computer

ROS.org

Importance of ros TF library

- TF
  - Transformation Frames
  - Must be connected to a global reference
    - /map or /world frames link with /odom ...



ROS.org

ROS Launch files

- Specify launch sequence
- Load required parameters and arguments
- Automatically launch rosmaster i.e. roscore (if it is not already running)
- Sample.xml

```
<launch>

<node ns="namespace" name="kinect_aux" pkg="kinect_aux"
    type="kinect_aux_node"/>

<node ns="namespace" name="ros_serial" pkg="rosserial_python"
    type="serial_node.py" args="/dev/ttyUSB0"/>

<node ns="namespace" name="kinect_ptu_node" pkg="kinect_ptu"
    type="kinect_ptu_node"/>

</launch>
```

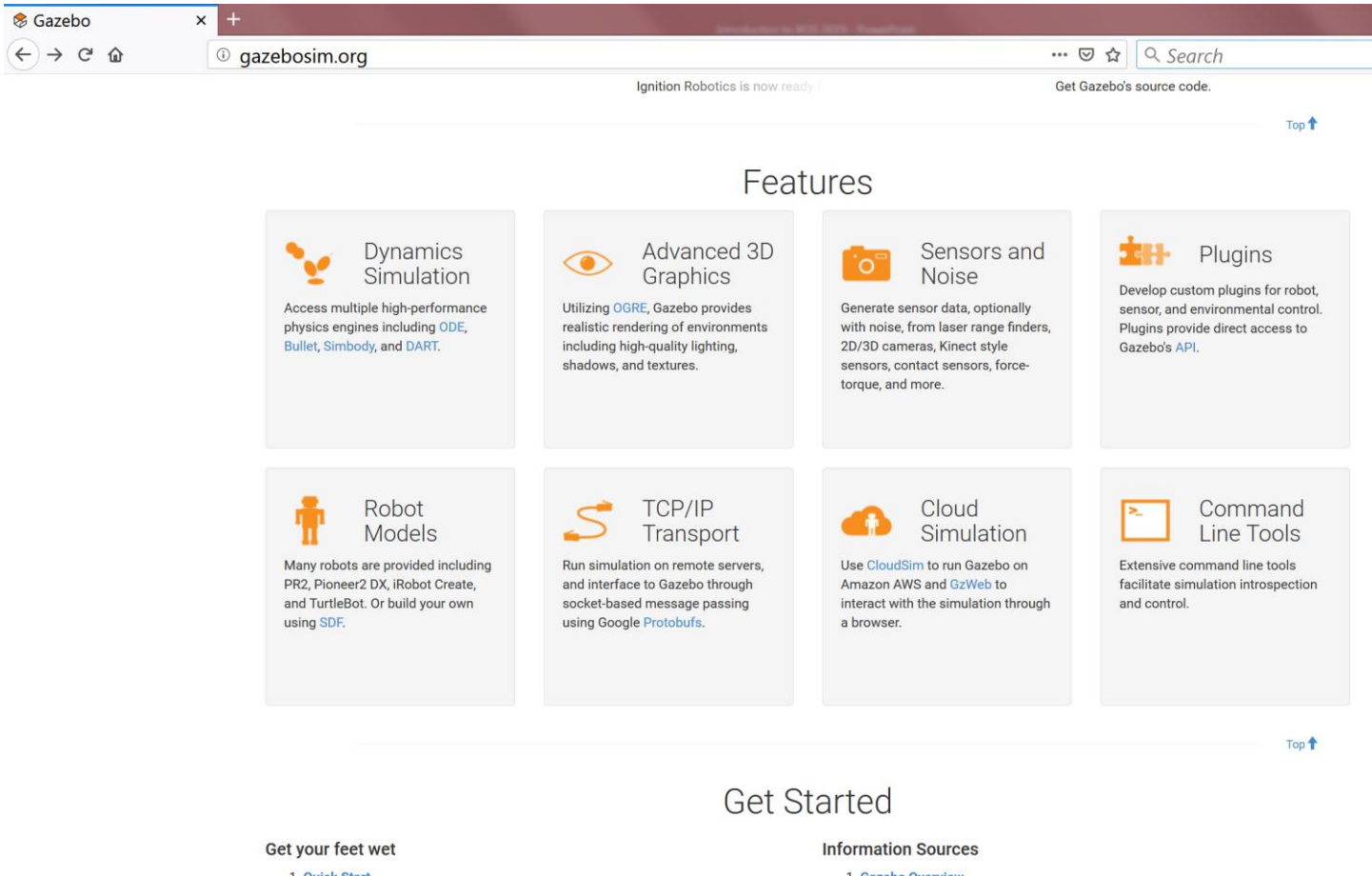
ROS.org

GAZEBO (1/2)

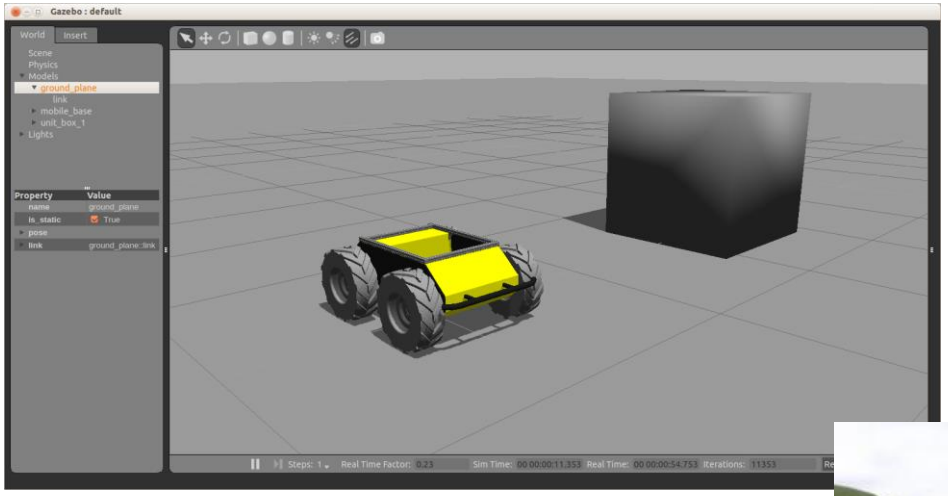
Why Gazebo?
Robot simulation is an essential tool in every roboticist's toolbox. A well-designed simulator makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. At your fingertips is a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. Best of all, Gazebo is free with a vibrant community.

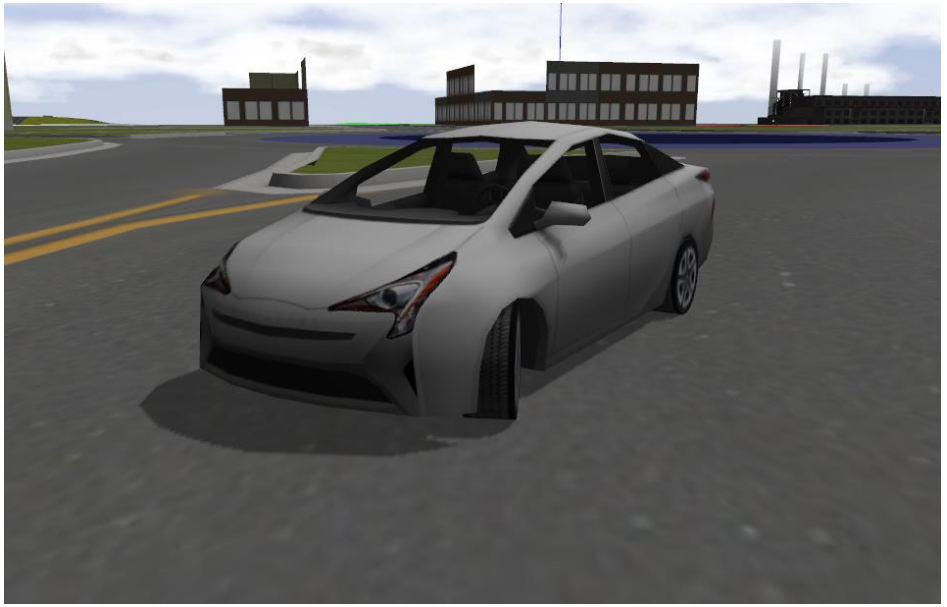
http://gazebosim.org/

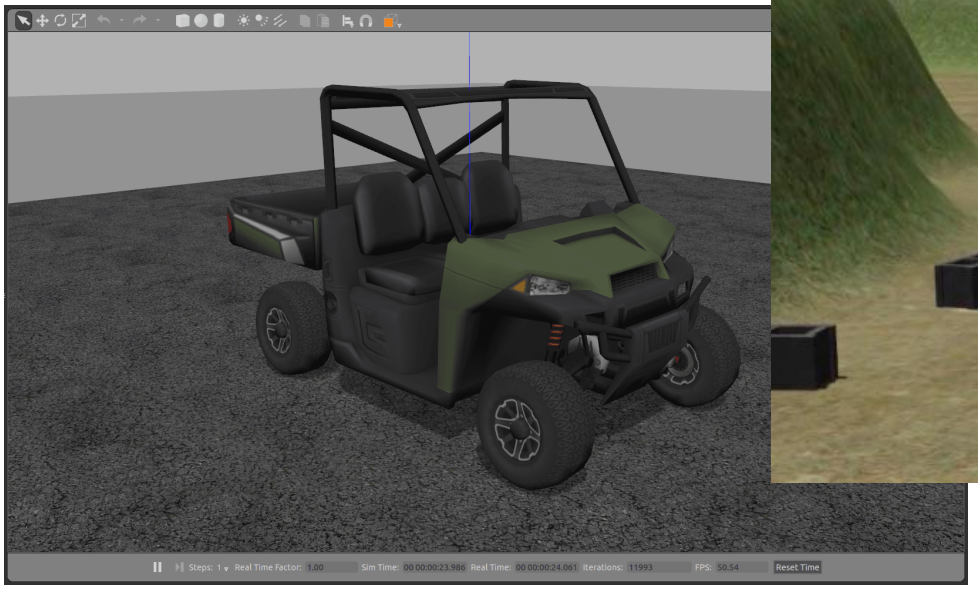
gazebo_ros_pkgs is a set of ROS packages that provide the necessary interfaces to simulate a robot in the Gazebo 3D rigid body simulator for robots. It integrates with ROS using ROS messages, services and dynamic reconfigure.

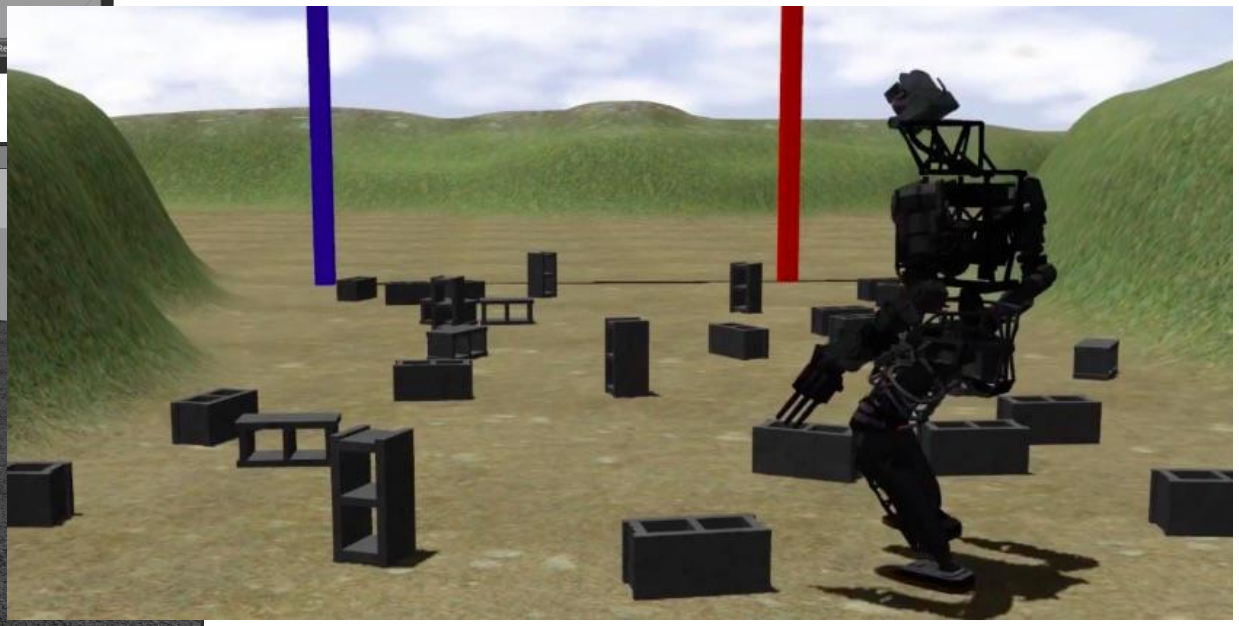


GAZEBO (2/2)


Gazebo : default
World
Insert
Scene
Physics
Models
ground_plane
link
mobile_base
unit_box_1
Lights
Property
Value
name
ground_plane
is_static
True
pose
link
ground_plane::link







Important links

- ROS home
  - http://www.ros.org
- ROS Indigo Installation
  - http://wiki.ros.org/indigo/Installation/Ubuntu
- ROS tutorials
  - http://wiki.ros.org/ROS/Tutorials
- Transformation frames library
  - http://wiki.ros.org/tf/Tutorials
- Gazebo
  - http://gazebosim.org/
- SLAM
  - http://wiki.ros.org/gmapping
- ROS OpenCV
  - http://wiki.ros.org/vision_opencv
- ROS PCL
  - http://wiki.ros.org/pcl
- ROS Navigation
  - http://wiki.ros.org/navigation
- ROSARIA
  - http://wiki.ros.org/ROSARIA
- URDF
  - http://wiki.ros.org/urdf/Tutorials

ROS.org

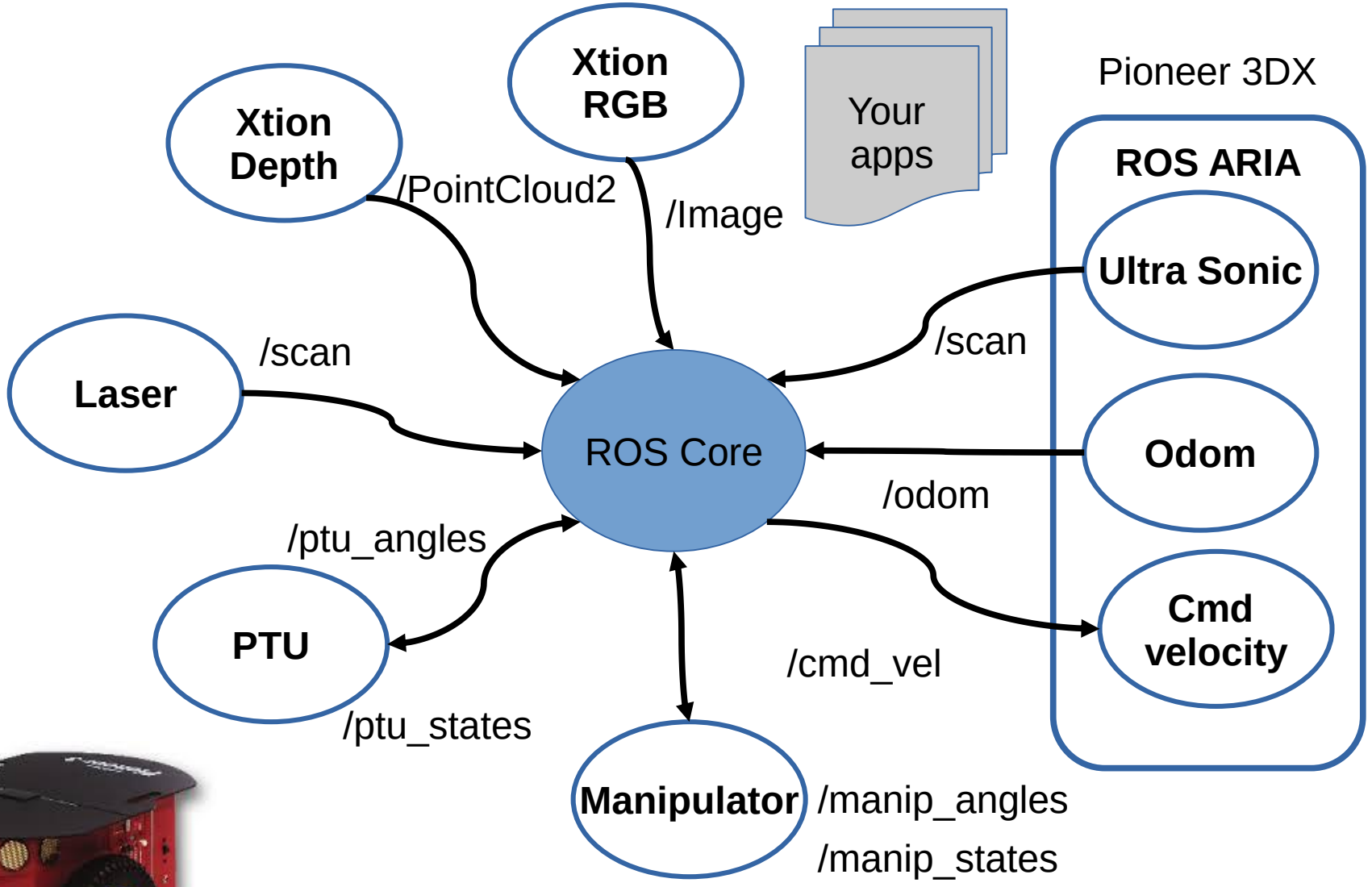
Pioneer 3DX PROJECT WORK

2D Sick Laser scanner

Asus Xtion camera (RGB+D)

ROS.org

Pioneer 3DX



ROS.org

Testing algorithms under ROS on Pioneer robot platform

1) Explore and map with SLAM an unknown area

- Easiest on the basis of Sick -laser scanner, but you can use Kinect as well.
- While mapping you can control the motion of the robot by manual controls
- Occupancy grid map is suitable for this case
- As base node Gmapping is used

2) Plan the collision free path of the robot from the current pose to the given pose using some of the algorithms of path planning.

- You should to use configuration space for path planning.
- As base planning node the ROS navigation stack is used.

ROS.org

Testing algorithms under ROS on Pioneer robot platform

3) Implement and test motion control with ROS.

4) Document the experiments and show the final operation of the robot.

You can get extra points by demonstrating other different SLAM and/or path planner algorithms.

ROS.org

Robot

- Pioneer P3-DX
- Motor controller, differential drive
- Encoders to calculate odometry
- 2D Sick Laser scanner
- Asus Xtion camera (Kinect type RGB+D)
  - Provides depth information
- Controlled by using ROS (Robot Operating System)

Practical Issues

- Robot situates in room 2552
- Reserve always time for your team in MyCourses
- First session two hours, extra time slots can be reserved.
- Power and recharging
  - Always keep the robot connected to the charger when possible

Timetable

Testing algorithms under ROS/Pioneer starts on March 18th

Design of a case robot system, DL Sunday April 14th, 21:00

Testing algorithms under ROS on Pioneer robot platform, DL for reports Sunday April 14th, 21:00

Important

- You are dealing with real, complicated mechanical device that can break. Please respect that!
  - If something happens, please contact staff immediately
- Some important things:
  - If you don't know how to do something – ask
  - If you don't understand the answer - ask again
  - DO NOT BREAK THE ROBOT

During the experiments, Mika can be reach:
Room: 2568 | Tel: 0505052156 | Email: mika.vainio@aalto.fi

FOR THIS AND NEXT WEEK:

Form your group (4-5 members)

Visit ros.org and go through some tutorials

Check out https://www.cse.sc.edu/~jokane/agitr/

From Friday 8th onwards book you time-slot with the robot