Introduction to ROS

Slides adapted from: http://courses.csail.mit.edu/6.141/spring2014/pub/lectures/Lec05-ROS-Lecture.pptm

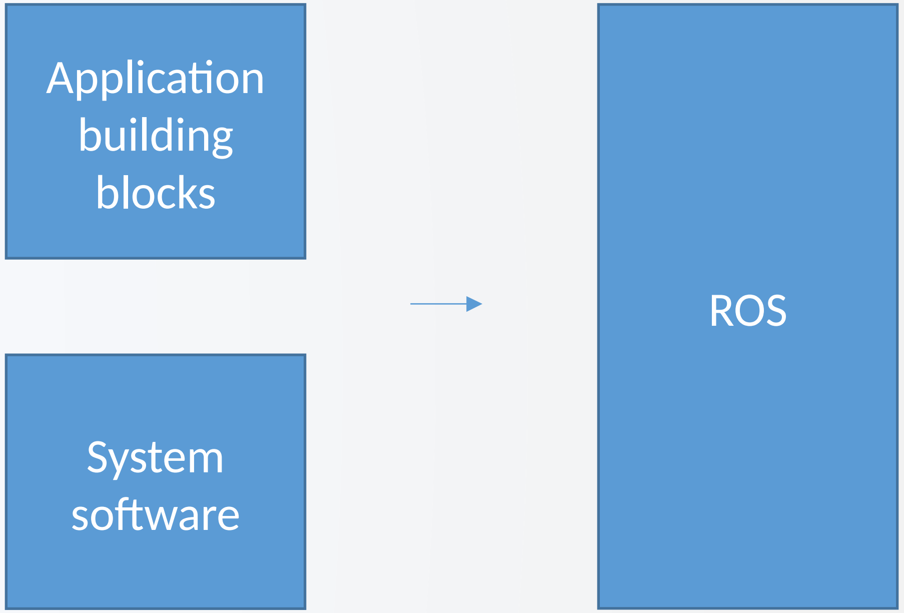
A meta-operating system for robots

What is ROS?

- A “Meta” Operating System.
    - Open source
    - Runs in Linux (esp. Ubuntu)
    - OS X support
    - Ongoing Windows implementation
- Nodes
- Message passing
    - Publish
    - Subscribe
    - Services via remote invocation
- Supports numerous programming languages (C++, Python, Lisp, Java)

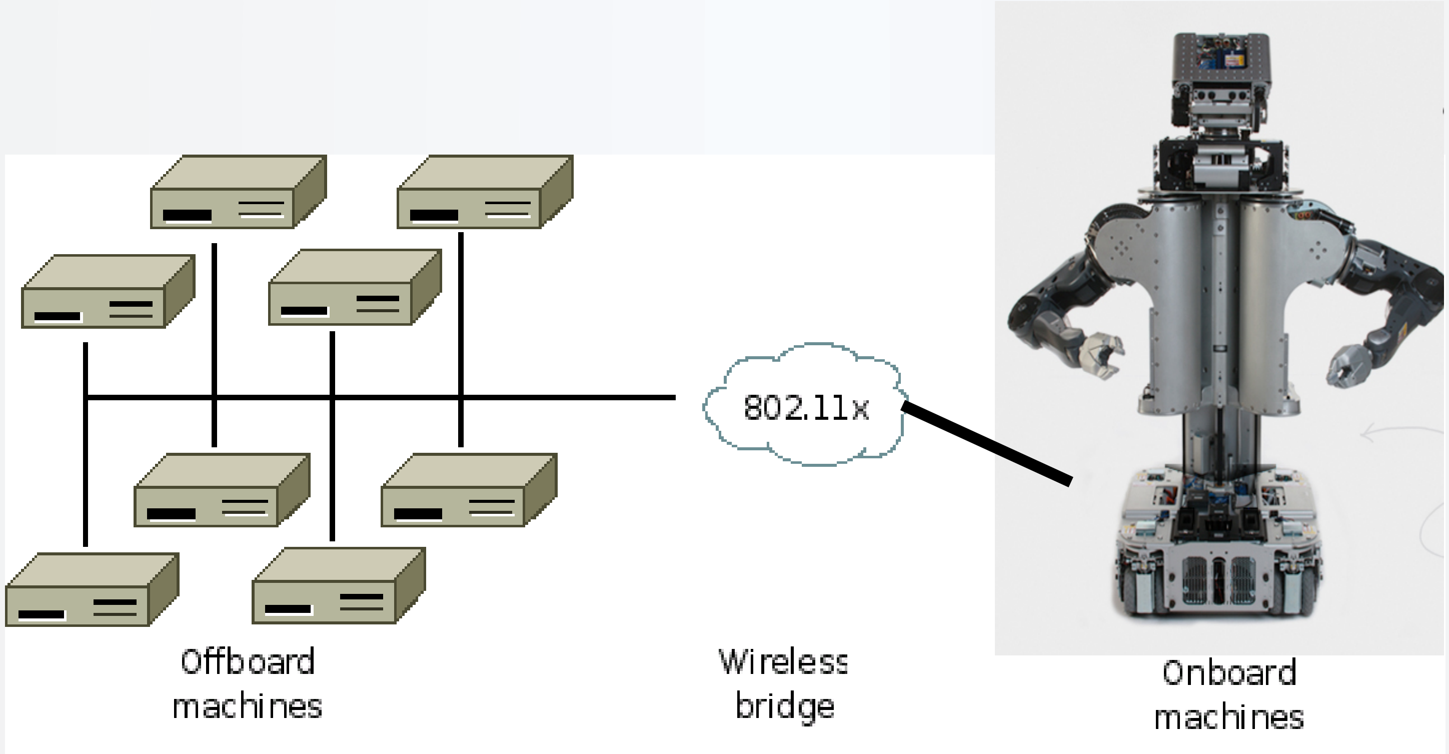
What is ROS?

- Low level device abstraction
  - Joystick
  - GPS
  - Camera
  - Controllers
  - Laser Scanners
  - …
- Application building blocks
  - Coordinate system transforms
  - Visualization tools
  - Debugging tools
  - Robust navigation stack (SLAM with loop closure)
  - Arm path planning
  - Object recognition
  - …



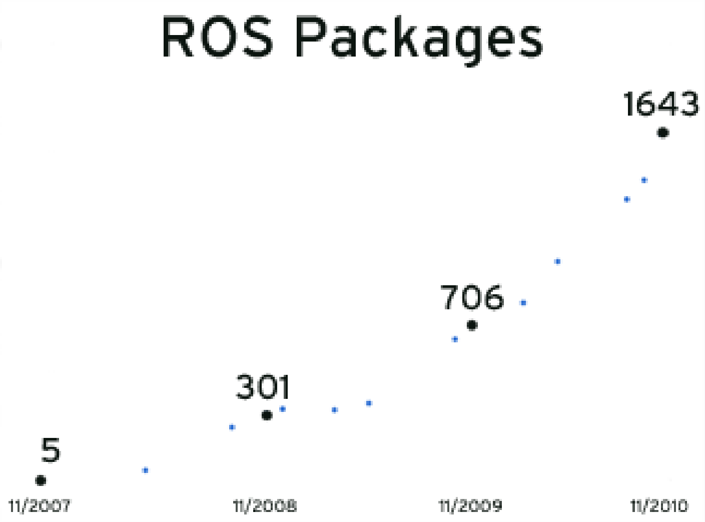
What is ROS?

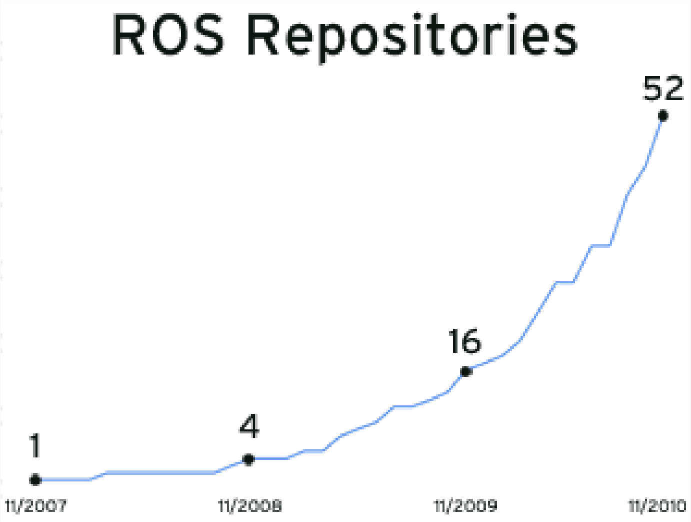
- Software management (compiling, packaging)
- Remote communication and control



What is ROS?

- Founded by Willow Garage
- Exponential adoption
- Countless commercial, hobby, and academic robots use ROS (http://wiki.ros.org/Robots)





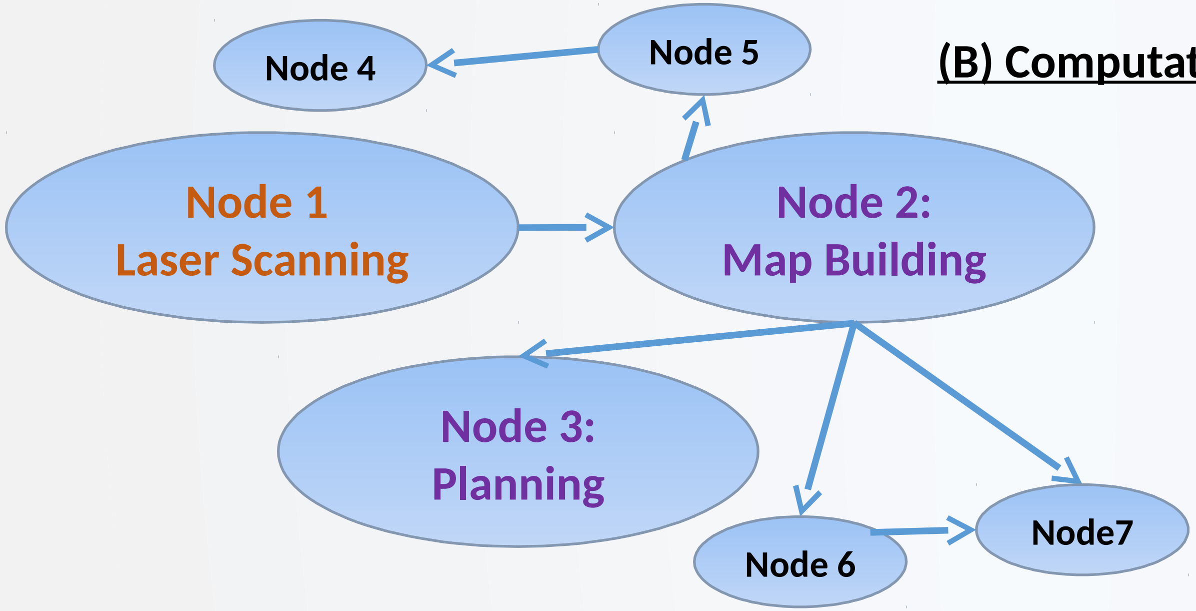
ROS Philosophical goals

- “Hardware agnosticism”
- Peer to peer
- Tools based software design
- Multiple language support (C++/Java/Python)
- Lightweight: runs only at the edge of your modules
- Free
- Open source
- Suitable for large scale research and industry

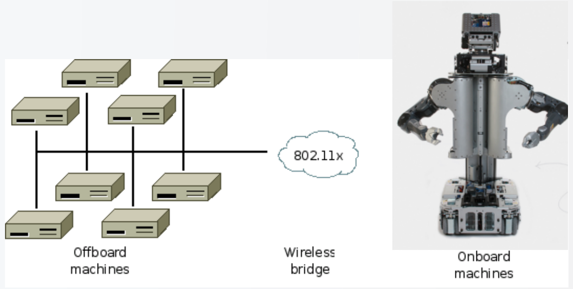
ROS software development

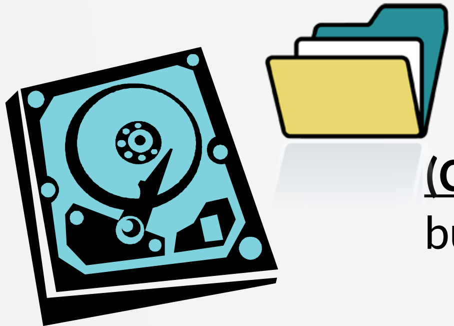
Conceptual levels of design

(A) ROS Community: ROS Distributions, Repositories



(B) Computation Graph: Peer-to-Peer Network of ROS nodes (processes).





(C) File-system level: ROS Tools for managing source code, build instructions, and message definitions.

Tools-based software design

Tools for:

- Building ROS nodes (catkin_make)
- Running ROS nodes (rosrun, roslaunch)
- Viewing network topology (rqt_graph)
- Monitoring network traffic (rostopic)

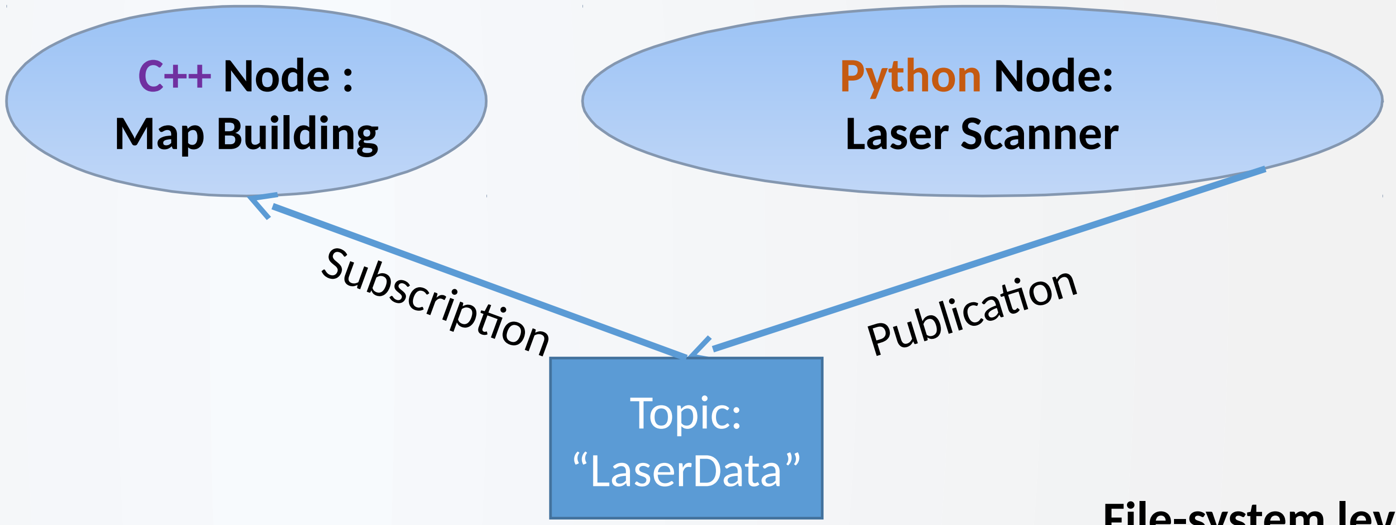
Many cooperating processes, instead of a single monolithic program.

Multiple language support

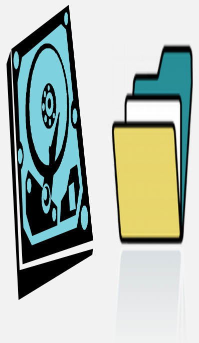
- ROS is implemented natively in each language.
- Quickly define messages in language-independent format.

File: PointCloud.msg

```
Header header
Points32[] pointsXYZ
int32 numPoints
```



File-system level



Lightweight

- Encourages standalone libraries with no ROS dependencies:

  Don’t put ROS dependencies in the core of your algorithm!

- Use ROS only at the *edges* of your interconnected software modules: Downstream/Upstream interface

- ROS re-uses code from a variety of projects:
  - OpenCV : Computer Vision Library
  - Point Cloud Library (PCL) : 3D Data Processing
  - MoveIt : Motion Planning

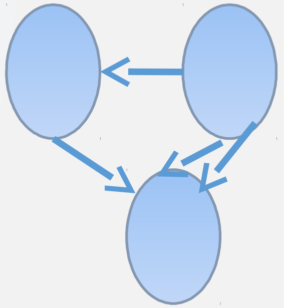
ROS Community

Carnegie
Mellon

Peer to Peer Messaging

- No Central Server through which all messages are routed.
- “Master” service run on 1 machine for name registration + lookup
- Messaging Types:
  - Topics : *Asynchronous* data streaming
  - Parameter Server

Computation Graph

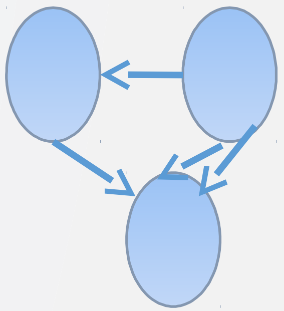


Peer to Peer Messaging

- *Master:* Lookup information, think DNS

  roscore command ✉ starts master, parameter server, logging
- Publish: Will not block until receipt, messages get queued.
- Delivery Guarantees: Specify a queue size for publishers: If publishing too quickly, will buffer a maximum of X messages before throwing away old ones
- Transport Mechanism: TCPROS, uses TCP/IP
- Bandwidth: Consider where your data's going, and how

Computation Graph



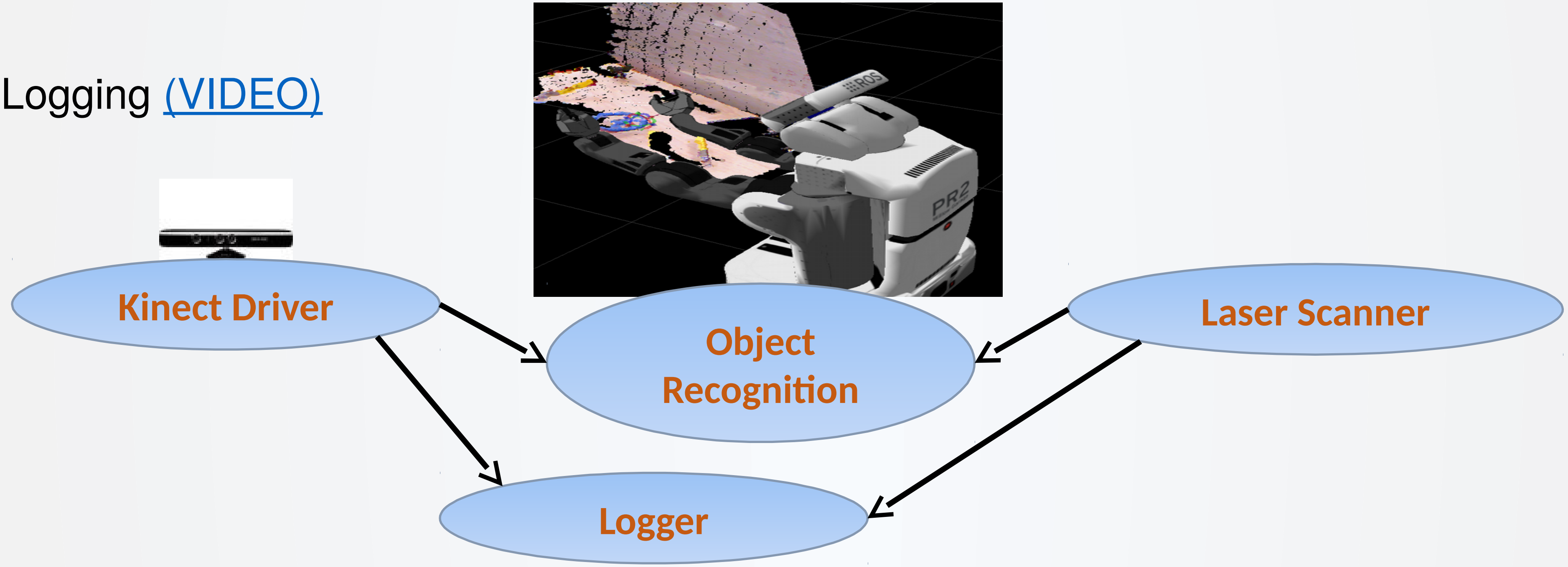
Free & Open Source

- BSD License : Can develop commercial applications
- Drivers (Kinect, Joystick, Lasers, and others)
- Perception, Planning, Control libraries
- Interfaces to other libraries: OpenCV, PCL, etc.

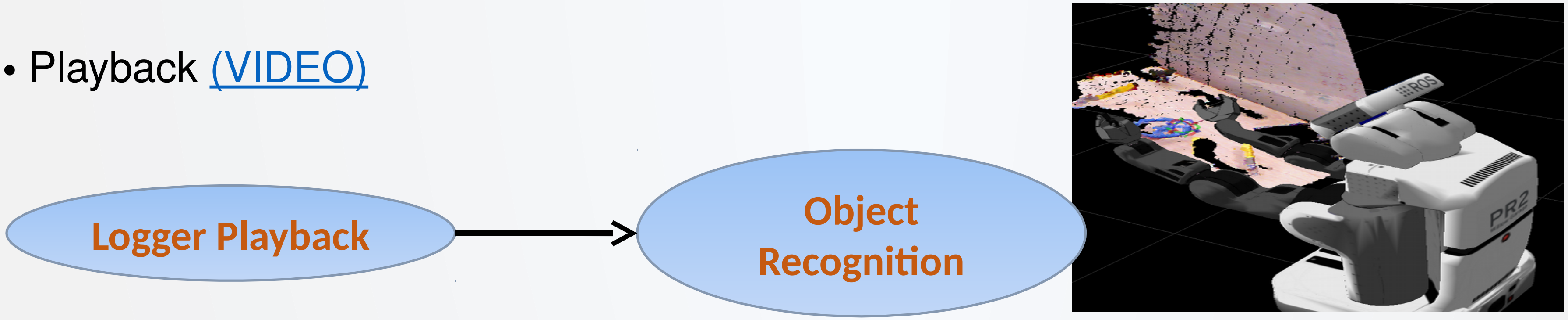
ROS Debugging

- Shutdown “Object” node ✉ re-compile ✉ restart : won’t disturb system

- Logging (VIDEO)



- Playback (VIDEO)



Useful ROS Debugging Tools

- `rostopic`: Display debug information about ROS topics: publishers, subscribers, publishing rate, and message content.

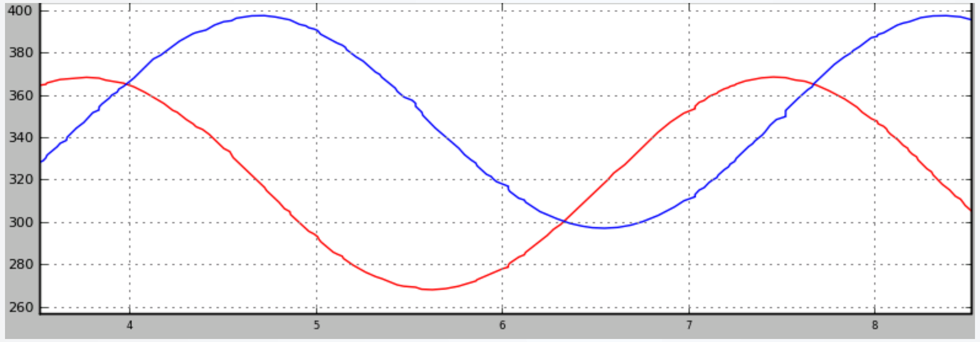
  `rostopic echo [topic name]` ✉ *prints messages to console*

  `rostopic list` ✉ *prints active topics*

  … *(several more commands)*

- `rqt_plot` : Plot data from one or more ROS topic fields using matplotlib.

  `rqt_plot /turtle1/pose/x,/turtle1/pose/y` ✉ graph data from 2 topics in 1 plot



Record data from published to topics

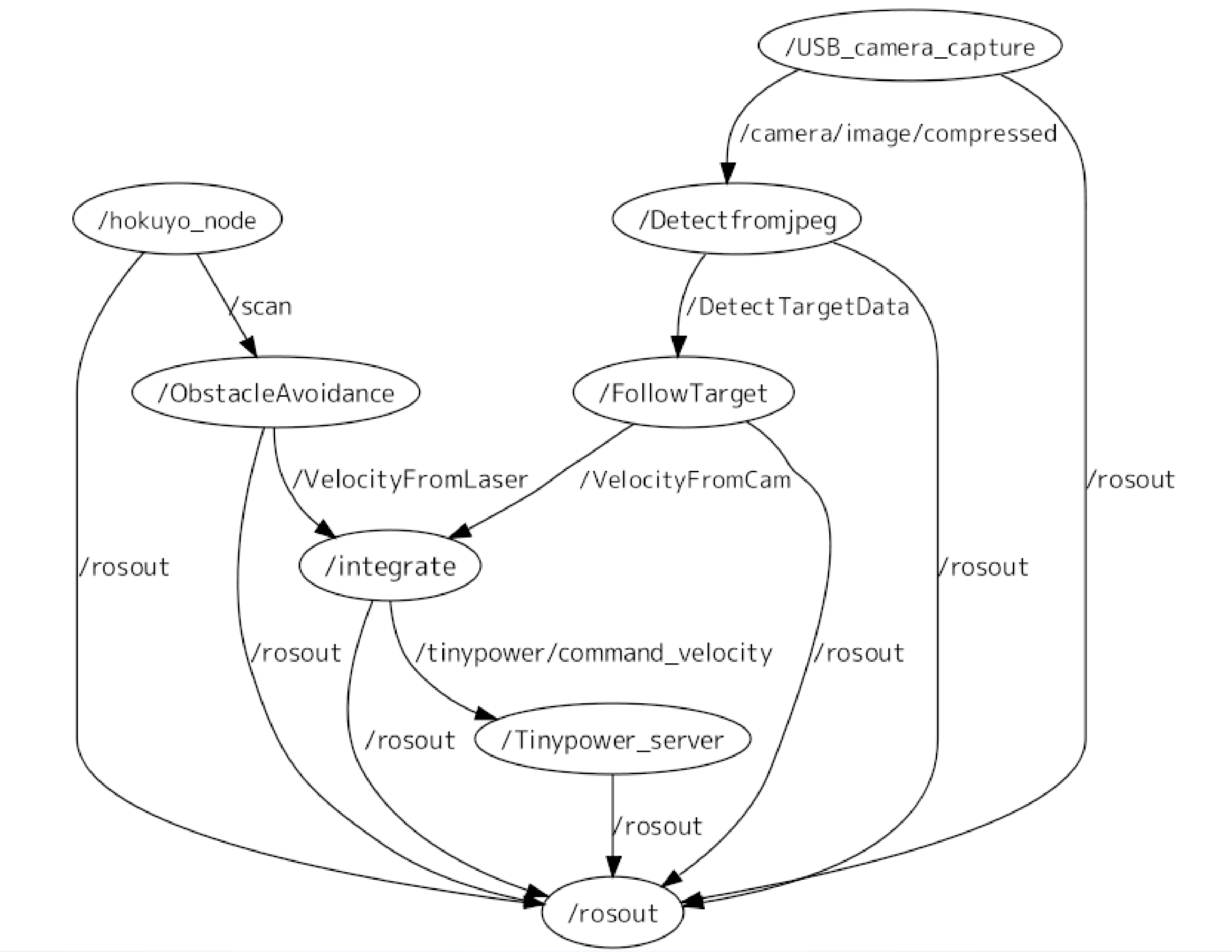
rosbag record [topics] -o <output_file>

Play back recording

rosbag play <input_file> --clock

Useful ROS Debugging Tools

`rqt_graph`

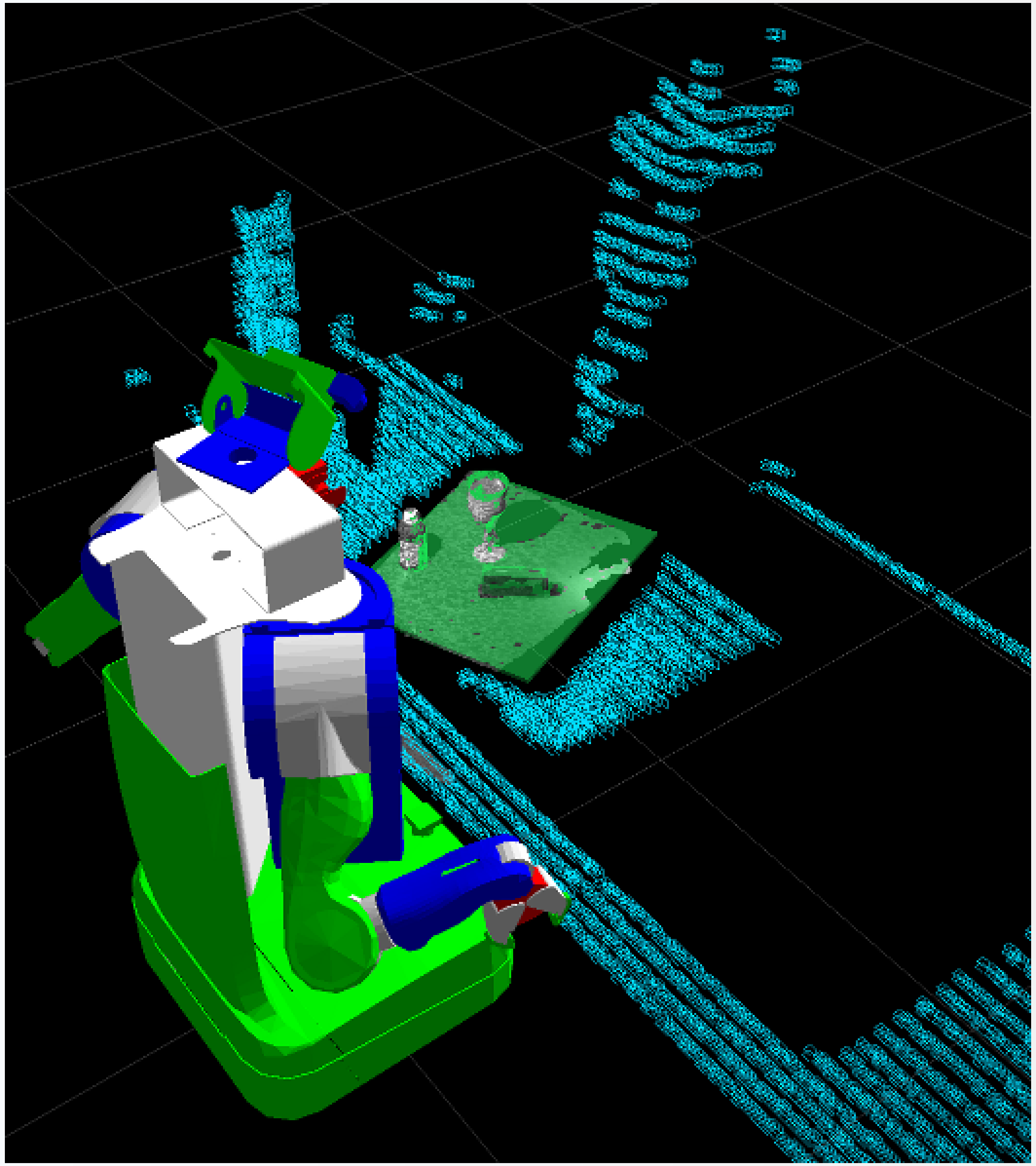


ROS Visualization

Visualize:

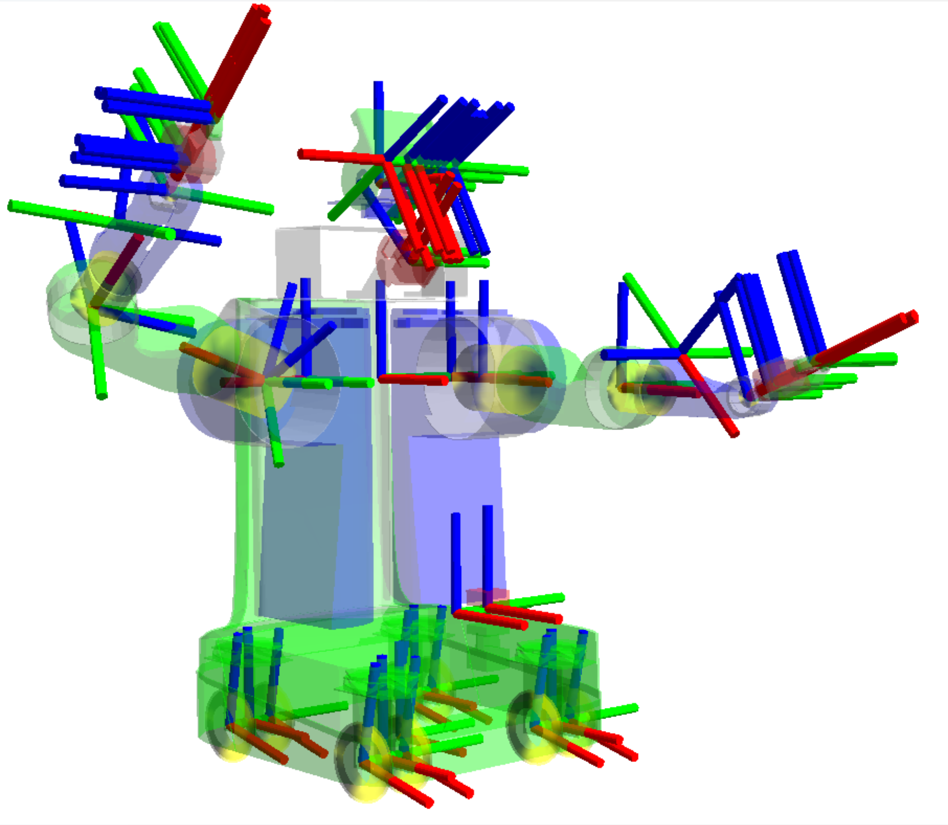
- Sensor data
- Robot joint states
- Coordinate frames
- Maps being built
- Debugging 3D markers

VIDEO



rviz

ROS Transformations



- “TF” = Name of Transform package

- TF Handles transforms between coordinate frames : space + time

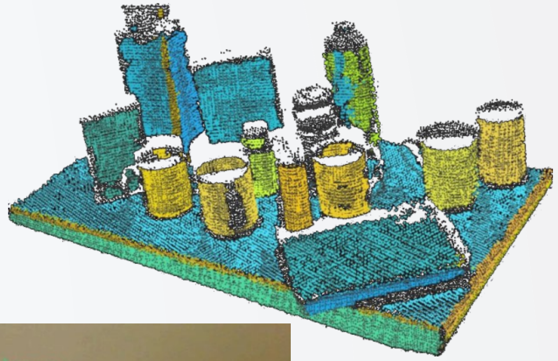
- tf_echo : print updated transforms in console

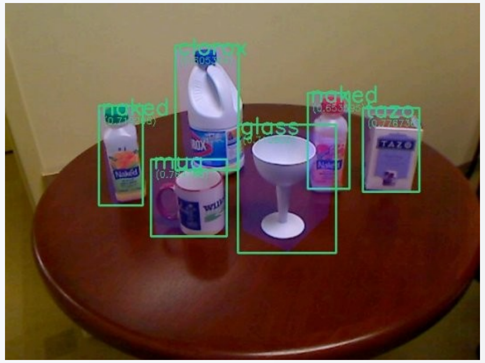
<u>*Example:*</u>

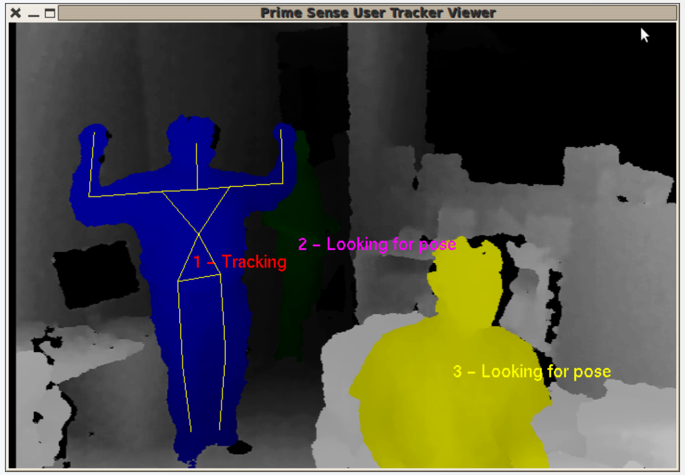
```
rosrun tf tf_echo [reference_frame] [target_frame]
```

Packages

- Perception
  - Point Cloud Library (PCL)
  - OpenCV
  - Kinect/OpenNI







ROS Simulator

Gazebo

- Can simulate different robots, sensors, and environments
- Develop algorithms and test in the simulator
- If model is good enough, same code will work on the real robot with similar performance.

ROS Resources

- http://www.ros.org
- http://wiki.ros.org
- ROS Tutorials: http://wiki.ros.org/ROS/Tutorials
- Gazebo: http://gazebosim.org/