ROS: the Robot Operating System

ROS

These are fantastic resources. Go see the source material!

Most slides drawn, with thanks, from:

Dolev Shapira: https://www.cs.bgu.ac.il/~ben-shahar/Teaching/Computational-Vision/StudentProjects/ICBV151/ICBV-2015-1-DolevShapira/object-recognition_slides.ppt

Quick Introduction to ROS: https://www.cs.columbia.edu/~allen/F19/NOTES/ROS%20Tutorial.pptx

Introduction to ROS – UW: https://courses.cs.washington.edu/courses/cse490r/18wi/quiz_section_slideseature

Robot Operating System, by Alec Poitzsch: http://courses.csail.mit.edu/6.141/spring2014/pub/lectures/Lec05-ROS-Lecture.pptm

ROS Basics: http://www.rsl.ethz.ch/education-students/lectures/ros.html

1

Bookkeeping

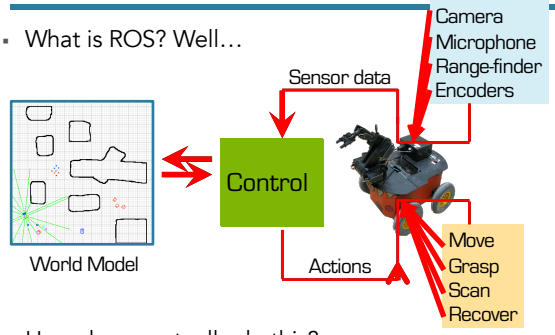
- Schedule is updated
- Videos, slides, etc. are up
- Next class: back to kinematics
- Next homework Friday
- Today: the Robot Operating System, + projects

2

2

ROS: Robot Operating System

- What is ROS? Well…



- How do we actually do this?

3

3

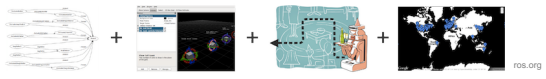
ROS

- …is not actually an operating system.
- But it is a *meta*-operating system that handles:
  - Communication
  - Sharing common tasks
  - Moving from simulation to reality
  - Representing a robot (shape, controllers, …)
- ROS lets you
  - Avoid reinventing a lot of wheels!
  - Handle the nuts and bolts of communication
  - … with some startup effort.

4

4

What does that include?



Plumbing	Tools	Capabilities	Ecosystem
Process management	Simulation	Control	Package organization
Inter-process communication	Visualization	Planning	Software distribution
Device drivers	Graphical user interface	Perception	Documentation
	Data logging	Mapping	Tutorials
		Manipulation	

5

5

Problem 1: sequential programming

- Conventional programming:

```
goForward(1);
turnLeft(Math.PI/2);
Image image = camera.getImage();
double distance = computeDistanceToObject(image);
goForward(distance - 1);
(x, y) = getMyPositionFromTheEncoderCounts();
…
```

What happens if an obstacle appears while you are going forward?

What happens to the encoder data while you are turning?

What if some other module wants the same data?

6

6

Solution 1: "Callbacks"

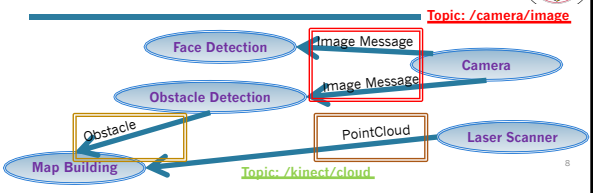
- Callback:
  - Function called whenever data is available for processing
  - An asynchronous callback can happen any time
- Examples:
  - Run the relevant callback function whenever:
    - An image is read from the camera
    - The odometry sensor reports new data

```
void imageCallback(ImageMessage image)
        // process the latest image

void odometryCallback(OdometryMessage data)
        // handle latest odometry data

void main()
        initialize();
        subscribe("image_msgs",
                imageCallback);
        subscribe("odometry_msgs",
odometryCallback);
```

7

7

Problem 2: complexity

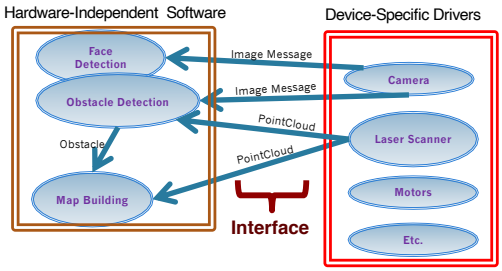


- Solutions:
  - Separate processes: Cameras, Odometry, Laser Scanner, Map Building can all be separated out: interact through an interface
  - Interfaces: Software processes ("nodes" in ROS) communicate about shared "topics" in ROS
  - Publish/Subscribe: Have each module receive only the data (messages) it requests

8

8

Problem 3: Hardware-specific code



- Solution 3: Abstract hardware!

9

9

History of ROS

- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory
- Since 2013 managed by OSRF
- Today used by many robots, universities and companies
- De facto standard for robot programming

10

10

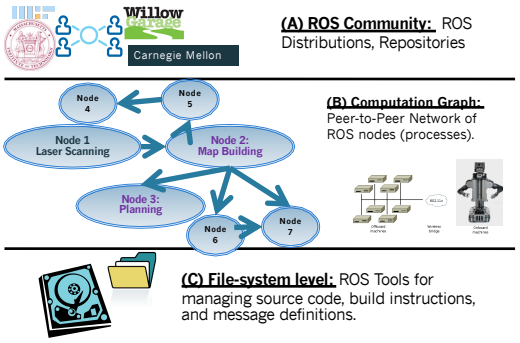
ROS philosophy

- **Peer to peer:** Individual programs communicate over defined API (ROS messages, services, etc.)
- **Distributed:** programs can be run on multiple computers and communicate over the network
- **Multi-lingual:** ROS modules can be written in any language for which a client library exists
  - In practice: C++ and Python
- Free and open-source

11

11

Conceptual levels of design



12

12

Tools-based software design

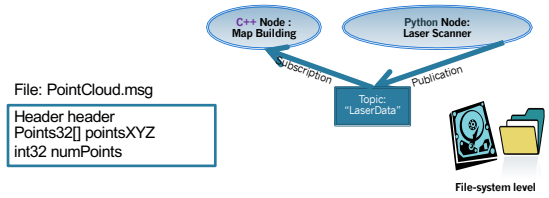
- Tools for:
  - Building ROS nodes
  - Running ROS nodes
  - Viewing network topology
  - Monitoring network traffic
- Many cooperating processes, instead of a single monolithic program.

13

13

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
```

14

14

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 Community

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

15

15

Communication

- ROS provides an interface that allows the user to create modular pieces that communicate
- Approximate equivalents:
  - Nodes ≅ "processes"
  - Topics ≅ "message boards"
  - Subscribers ≅ Nodes "watching" the topics and reading incoming messages
  - Publishers ≅ Nodes publishing ("posting") messages to topics

16

16

Peer to Peer Messaging

- No central server through which all messages are routed
- "Master" service runs on one machine for name registration + lookup
- Messaging Types:
  - Topics : Asynchronous data streaming
  - Parameter Server

Sidebar: computer science is getting better about terminology like "master/slave" systems, but it persists here. We'll get there.

17

17

Peer to Peer Messaging

- **Master:** Lookup information, think DNS
- **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

18

18

ROS master/clients

- Master:
  - Manages the communication between nodes (processes)
  - Every node registers at startup with the master

Start a master with

```
> roscore
```

19

19

ROS master/clients

- Clients:
  - Single-purpose, executable program
  - Individually compiled, executed, and managed
  - Organized in packages

Run a node with

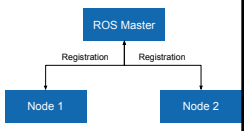
```
> rosrun package_name node_name
```

See active nodes with

```
> rosnode list
```

Retrieve information about a node with

```
> rosnode info node_name
```



20

20

ROS topics

- Nodes communicate over *topics*
- Nodes can publish or subscribe to a topic
- Typically, 1 publisher and n subscribers
- Topic is a name for a **stream** of messages

List active topics with

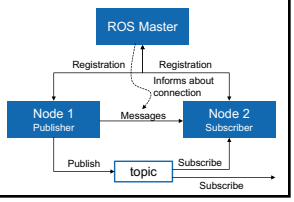
```
> rostopic list
```

Subscribe and print the contents of a topic with

```
> rostopic echo /topic
```

Show information about a topic with

```
> rostopic info /topic
```



21

ROS messages

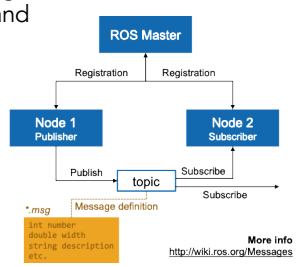
- Data structure defining the type of a topic
- Nested structure of integers, booleans, strings etc., and arrays of objects
- Defined in *.msg files

See the type of a topic

```
> rostopic type /topic
```

Publish a message to a topic

```
> rostopic pub /topic type args
```

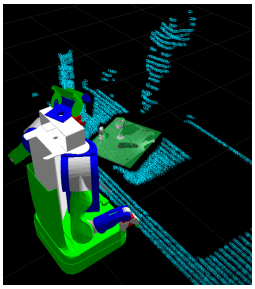


22

22

ROS Visualization

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



23

23

rviz

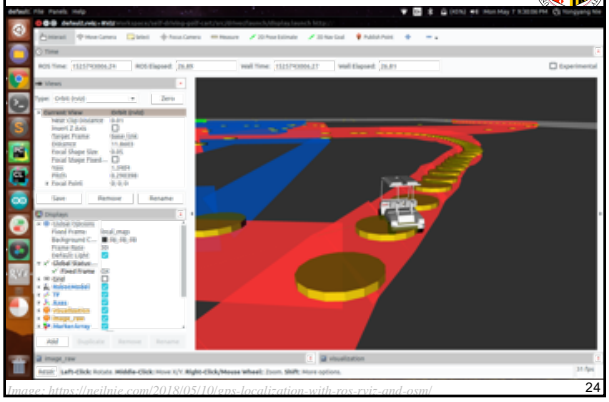


Image: https://nellmie.com/2018/05/10/ros-localization-with-ros-rviz-and-osm/

24

24

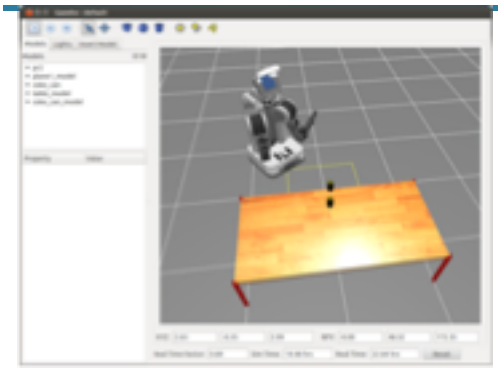
More solvable problems:

- I don't have a robot in front of me
- I want to try something that may break my robot
- Setting up the robot takes too much time, I want to test changes to my code quickly

25

25

Gazebo simulator



26

26

Gazebo simulator

- Same interface as real robot
- Add/remove 3D items in environment
- Physics engine to simulate effects of motor commands
- Collision detection
- Updated sensor feedback
- Debugging info

27

27

Sources of information

- Tutorials: http://wiki.ros.org/ROS/Tutorials
  - We'll start with these?
- http://answers.ros.org/questions/
- Googling – ROS is surprisingly widely used!

- To start with:
- http://wiki.ros.org/ROS/Installation

28

28