CS 309: Autonomous Intelligent Robotics
FRI I

Lecture 9: Introduction to ROS

Instructor: Justin Hart

http://justinhart.net/teaching/2018_spring_cs309/

A couple of quick notes

- Homework 1: Due tonight
  - Any questions?
- Don't forget what you learned in the PDDL lectures
  - That homework goes out tonight!
- Reading 1: Due Monday night
  - It should be short and fast. Try to enjoy it. I'll bring in videos!
  - It was featured in a short documentary: https://www.youtube.com/watch?v=-iaiRW3URto

ROS

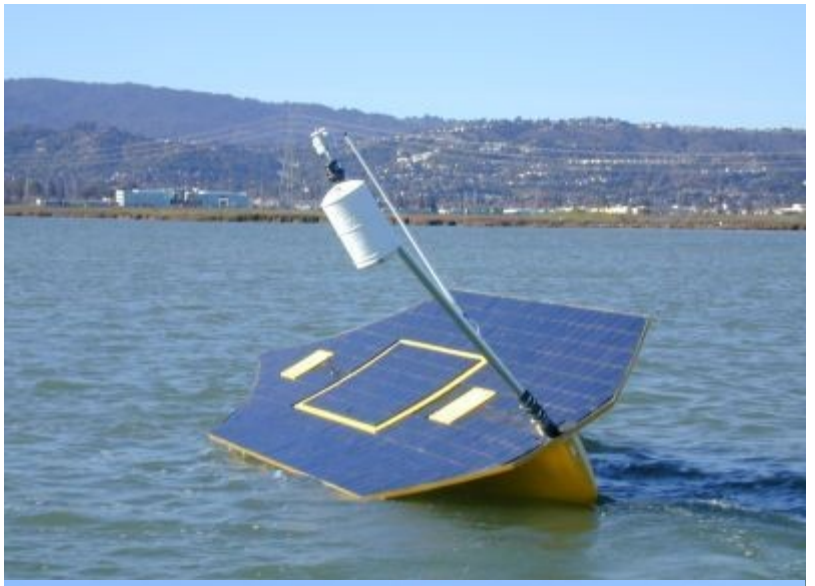
- Robot Operating System
- A middleware layer that provides communication between robotics software packages
- A collection of utilities relevant to robotics

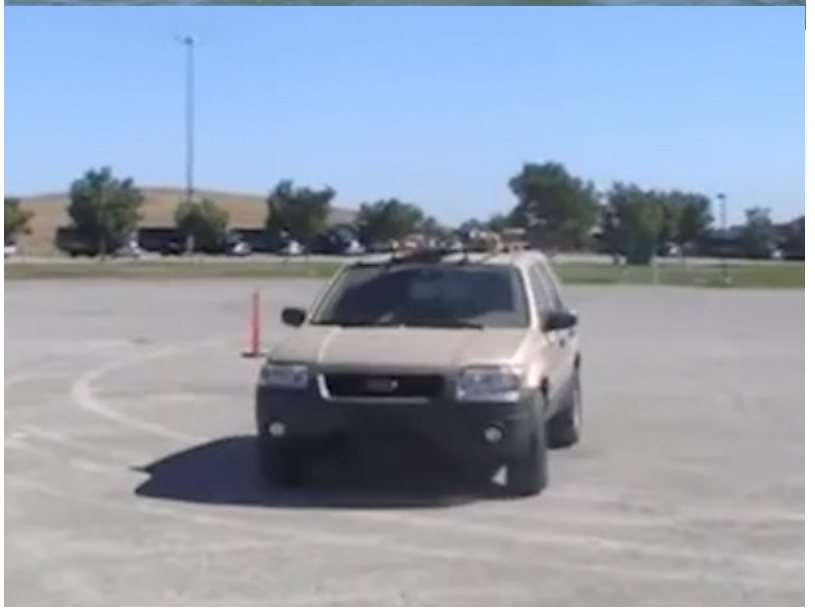
ROS – A brief history

- Prior to ROS, basically every robot ran highly customized software
  - Though many still do.
- A robot may require computer vision software, kinematic solvers (motion), navigation software, and so forth.
  - Before ROS, this meant digging through software libraries and piecing it into your robot's custom software

ROS – A brief history

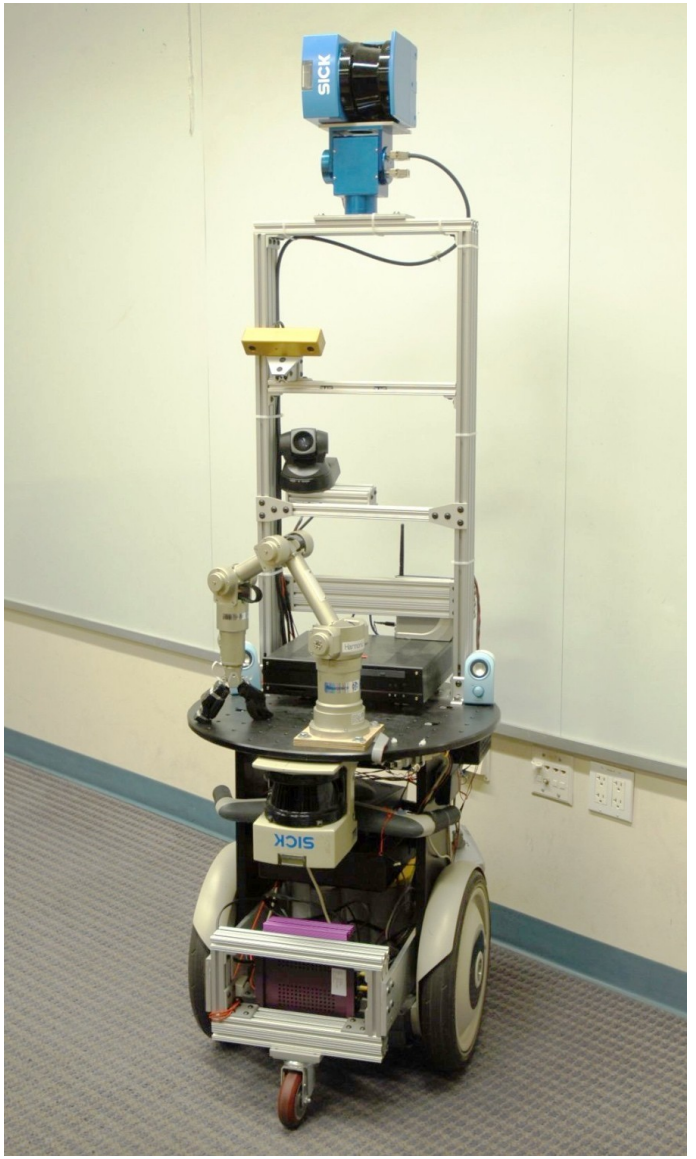
- In 2006 Willow Garage was founded
  - Willow Garage was a robotics company and incubator
  - First two projects
    - DARPA Urban Challenge autonomous vehicle race
    - Solar-powered boat





ROS – A brief history

- Around the same time, the STAIR program at Stanford had 4 robots
  - Wouldn't it be great if these all had the same basic starter software?
  - Again, at the time, there was no common robotics software platform

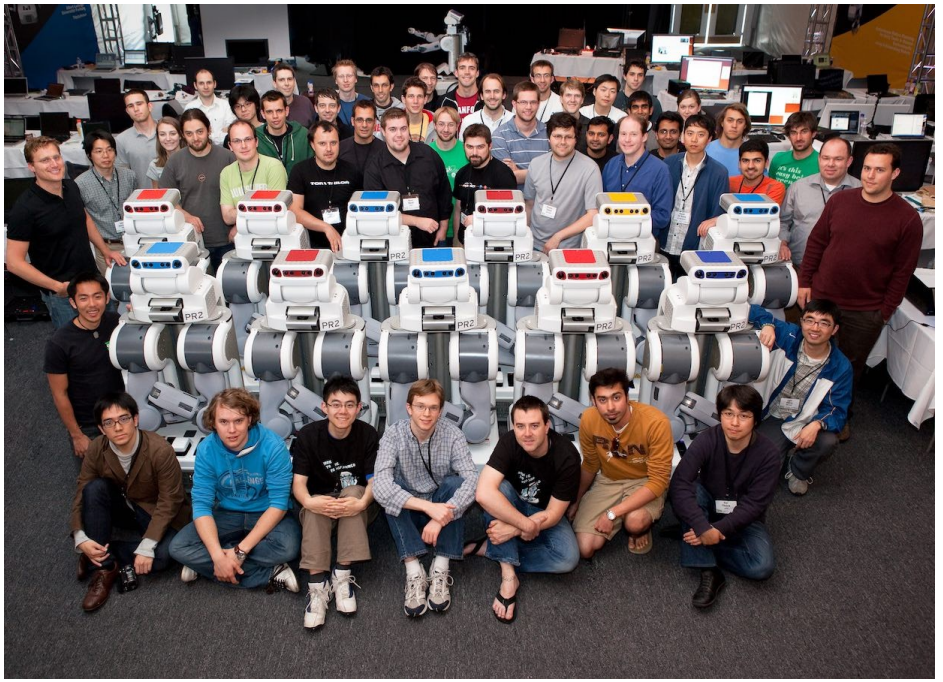


ROS – A brief history

- In 2007, the Stanford AI lab made the first ROS release
- In 2008, two concepts were pitched to Willow Garage (which was only a couple of miles from Stanford)
  - Build a common robotics hardware platform – the Personal Robot 1 (PR1)
  - Build a common robotics software platform – ROS
- Willow Garage hires a bunch of people, kicks off a number of internal projects

ROS – A brief history

- By 2010 ROS had grown
- Willow Garage offered the PR2 for sale
  - Price ~$400,000, each
- 11 schools were included in a beta program and got theirs for free



ROS – A brief history

- The robot and ROS had been built-up together
  - Creating a robotics ecosystem with the PR2 and ROS at the center of it
- The schools had to open source software developed on the PR2
  - Which resulted in a large collection of ROS software
- ROS became the closest thing to a “starter kit” for robotics that has ever existed
  - The result is that ROS became the dominant technology in robotics both in academia and commercially

ROS – A quick overview

- Communications
  - ROS Topics
    - Publish/Subscribe
    - A “node” (a ROS program) may “publish” a topic
      - For instance, a node connected to a sensor may publish 3D point cloud data
    - A node “subscribe” to a topic in order to use that data
    - Many nodes may concurrently subscribe to topics

ROS – A quick overview

- Communications
  - ROS Services
    - Remote Procedure Call
      - Allows one ROS node to offer a function and another ROS node to call that function
      - As such, functions can reside in entirely different computer programs and still be called
      - This is useful if one program should exclusively handle some type of request, or can be packaged to handle such a request
        - “Tell me how fast the robot is moving”
        - “Use PDDL to compute a plan for me”
        - “Change the robot's navigation goal”

ROS – A quick overview

- Communications
  - ROS actionlib
    - RPC + Feedback
      - Use in the same places you would a ROS service, but the service can provide feedback
        - “Use the arm to pick up that object.”
          - Feedback tells you progress towards that goal
        - “Navigate to this location.”
        - “Say the following..”

ROS – A quick overview

- Simulation – Gazebo
  - 3D robot simulation
  - Works with most ROS software
    - Publishes ROS topics
    - Services ROS actionlib and service calls
  - Users can download models of real robots or build them themselves
  - Users can download or build models of real places
  - We have the 3rd floor of GDC in Gazebo

ROS – A quick overview

- Simulation – Gazebo
  - Watch video

ROS – A quick overview

- Visualization – rviz
  - ROS visualizer
  - Visualizes many kinds of data
    - TF (transform) frames
      - Locations and directions (poses)
    - URDF – Universal robot definition file
      - 3D robot model data
    - Point Cloud
      - 3D vision data
    - Camera Images
    - Mapping Data
    - Markup
    - Many others

ROS – A quick overview

- Visualization – rviz
  - Watch video

ROS – A quick overview

- Then, there is a large software collection that does basic tasks, these can be joined in “stacks” of programs, and nodes can interface to these stacks
- Packages include
  - Perception
    - Finding known objects, planes, shapes
  - Navigation
    - Most robots can drive themselves out of the box
  - More complicated packages and stacks that build complex features
    - MoveIt

ROS – A quick overview

- MoveIt
  - Provides a pipeline for complex motion planning, such as robot arms grasping and manipulating objects
  - Pipeline parts include
    - Perceptual data input
    - Models of the robot so the system knows how to move
    - An assortment of motion planners
    - Methods to customize all of these pieces
    - Simulation and visualization

ROS – A quick overview

- MoveIt
  - Watch video

Installing ROS

- Ubuntu 16.04 – You should already have this
- ROS installation instructions can be found here:
  - Follow these only if you are installing on your personal machine
  - http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment
  - You need to pick “Kinetic Kame”
    - Ubuntu
      - Amd64
  - Set up sources.list
    - sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
  - Add keys identifying this as a trusted source
    - sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
  - sudo apt-get update
  - sudo apt-get install ros-kinetic-desktop-full

Installing ROS

- If you miss a package, you will know it because the machine will tell you
  - It will also tell you how to install it
  - If it doesn't, Google will almost always be able to tell you
  - Also, we'll help you get your machine configured
- If you are using a lab machine, Kinetic is already installed
  - Both the undergraduate lab, and the BWI Lab
- In general, if it involves sudo, you can't do it on a lab machine, and probably don't need to

Installing ROS

- Setting up rosdep
  - sudo rosdep init
  - rosdep update
- rosdep is used to set up dependencies
  - For instance, your package may require another package. This will help automatically set that up

Configuring your environment

- http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment
- You can manually configure your environment, but probably will not want to keep doing this
  - source /opt/ros/<distro>/setup.bash
    - Where distro is kinetic
- Configuring your environment sets up “environment variables”
  - $PATH
    - Where programs can be found
  - $ROS_PACKAGE_PATH
    - Where ROS packages, containing packaged stacks and programs can be found
  - Others

Creating a ROS Workspace

- mkdir -p ~/catkin_ws/src
- cd ~/catkin_ws/src
- catkin_init_workspace
  - This is different from the guide, both work
- catkin build

Navigating ROS

- http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem
- We'll just follow the tutorial here

Navigating ROS

- http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem
- We'll just follow the tutorial here

Cmake – A brief sidebar

- For our first homework assignment, we are using make, which simplifies building software
- Cmake, or cross-platform make is intended to simplify creating Makefiles
- Makefiles, in large software systems, cross-reference many dependencies. Cmake is intended to help manage this

Cmake and package.xml

- ROS packages require two files with similar contents
  - Cmake tells the system how to build your software
  - package.xml is a “manifest”
    - It tells ROS things like the name of the package, runtime dependencies, and build dependencies
- We will walk through both now

Building your package

- cd ~/catkin_workspace
- catkin_make
- Source devel/setup.bash
  - This puts your catkin worskspace into your ROS_PACKAGE_PATH

That was a lot of talking

- Are there any questions before we push ahead?

A quick primer on ROS Nodes

- http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes
- The basic idea here is to show that you run
  - roscore
    - Which manages communications
  - turtlesim_node
    - Which connects to roscore in order to communicate

Understanding ROS Topics

- http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics
- ROS Topics run on a publish/subscribe architecture
  - A “publisher” provides a stream of data
  - A “subscriber” listens to this stream
- The first demo shows
  - turtlesim_node, which is a subscriber
  - turtlesim_teleop_key, which is a publisher
- If alongside this we run rostopic echo, we can see the keys telling the turtle what to do