History and ROS overview

Ioannis REKLEITIS
Computer Science and Engineering
University of South Carolina
yiannisr@cse.sc.edu
Outline

• Problems to solve in robotics
• Robot software architectures with some history
• ROS
Three Main Problems in Robotics

1. Where am I? (Localization)
2. What the world looks like? (Mapping)
   – Together 1 and 2 form the problem of Simultaneous Localization and Mapping (SLAM)
3. How do I go from A to B? (Path Planning)
   – More general: Which action should I pick next? (Planning)
The first industrial robot, UNIMATE, in 1954

- Designed by George Devol, who coins the term Universal Automation
- Name shortened to Unimation, which becomes the name of the first robot company (1962)
Reactive architecture

Sense → Act

Sense
Act
Mobile Robots: 1950

• Walter’s Tortoise

Source: sciencemuseum.org.uk

https://www.youtube.com/watch?v=wQE82derooc
Deliberative architecture

Sense → Plan → Act → Sense
Shakey (1966 -1972)

Source: wikipedia.org

https://www.youtube.com/watch?v=7bsEN8mwUB8
Stanford Cart

- 1973-1979
  - Stanford Cart developed by Hans Moravec
  - Use of stereo vision.
  - Took pictures from several different angles
  - The computer gauged the distance between the cart and obstacles in its path to do basic collision avoidance
  - About **15 min** to think about each image, then drives 1 foot or so.

Source: stanford.edu
Hybrid architecture

Plan

Sense

Act
MIT Talos (2007)

- Participated in the DARPA Grand Challenge

https://www.youtube.com/watch?v=F_tk6C9KGL4

Source: mit.edu
Spectrum of control

Source: [Arkin, 1998, MIT Press]
Middleware

Software → Middleware → Hardware/Software
WillowGarage PR2 (2007)

Source: WillowGarage
ROS

• The Robot Operating System (ROS) is a flexible framework for writing robot software
  – It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms
• Developed and Maintained by the Open Source Robotics Foundation (OSRF)
• “The primary goal of ROS is to support code reuse in robotics research and development.”
• ROS is based on publish/subscribe message passing approach
• The core elements are:
  – ROS master: process that provides naming and registration to the rest of the nodes
  – Nodes: processes implementing robotic components
  – Topics: named buses over which nodes exchange messages
ROS overview

• Process for making two nodes interact with each other
ROS overview

- Process for making two nodes interact with each other

```java
advertise("images")
```

- `ros "master"
- `camera`
- `viewer`
ROS overview

• Process for making two nodes interact with each other
ROS overview

• Process for making two nodes interact with each other

![Diagram of ROS interaction between a camera and a viewer]

```python
ros
"master"

subscribe("images")

camera

viewer
```
ROS overview

• Process for making two nodes interact with each other
• Process for making two nodes interact with each other
ROS overview

• Process for making two nodes interact with each other

![Diagram showing ROS process for two nodes interacting](image-url)
ROS overview

- Process for making two nodes interact with each other
ROS overview

• Process for making two nodes interact with each other
ROS overview

- Process for making two nodes interact with each other
ROS overview

- Typically, a node represents one task (driver, localization, mapping, path planning, ...)
- Nodes run in parallel
- To debug problems, use `rqt_graph`
ROS overview

• The main mean of communication in ROS are topics and messages

• However, there are other ways for nodes to communicate with each other
  – Services: similar to Remote Procedure Calls
  – Actionlib: preemptable tasks
ROS overview

• How to decide what to use:
  – Topics: especially for stream of data
  – Services: execution of fast tasks
  – Actions: execution of tasks that need to be tracked and should be preempted in some cases
• Parameters can be easily set
  – Statically: rosparam
  – Dynamically: rqt_reconfigure
• Be careful in which namespace the parameters are defined: global or private
ROS overview

- Logging data streams can be achieved by using rosbag
- Remember the ROS parameter sim_time especially to run algorithms on bag files

Source: ros.org
ROS overview

• Logging messages are published in rosout topic
  – Different log levels should be used according to the severity of the message

• rqt_console can be used to visualize them
ROS tools

- **Navigate:** roscd, rosls
- **Setup:** catkin_init_workspace, catkin_create_pkg
- **Configure:** package.xml, CmakeLists.txt
- **Build:** catkin_make
- **Execute:** roscore, rosrun, roslaunch, rosparam, rqt_reconfigure
- **Inspect:** rosnodes, rostopic, rosservice
- **Debug:** rqt_graph, rostest, rqt_plot
- **Log & Analyze:** rosbag, rqt_bag, rqt_console
ROS packages

source code
header declarations
scripts
message definitions
service definitions
configuration files
launch files
metadata
...

package

stack

package_n

package_two

package_one
• Note that in Ubuntu, there are many packages ready just to be installed with `sudo apt-get install`
From source code to executable

1. Write program within ROS package
2. Build (compile, link, install executables)
3. Export the package
4. Run within ROS
ROS tools

- rviz can be used to visualize data

Source: iheartrobotics.com
ROS tools

- Stage, 2D simulator
ROS tools

• Gazebo, 3D simulator
• In a multirobot settings, a possibility is to share the ROS master over all of the computers
However, to have the system more robust, *multi-master-fkie* can be used to allow robots to see other ROS masters.
Summary

• Problems to solve in robotics
  – Localization, Mapping, Planning

• Robot software architectures with some history
  – Deliberative
  – Reactive
  – Hybrid

• Middleware
  – ROS
    • Core elements
    • Packages
    • Tools
Questions?