Topic in AI:
Robot Programming &
ROS

Prof. Ronen Brafman & Mr. Shai Givati
Goal

• Learn to use ROS: the Robot Operating System

• Learn to use Gazebo: the ROS 3D simulator

• Apply this knowledge to do some simple programming on a real robot
Video
What to Expect

• This is a mini-project. This means you will be mostly on your own

  • You will learn all the material from online tutorials, other resources, and from experiencing things on your own

• However, we provide help in the form of office hours support by Mr. Shai Givati, the Robotics lab engineer, and one of the course instructors.

• Cooperation among groups in learning the material is encouraged

• ROS requires working in Linux using C++ and Python

  • More advanced work on the robot can be done with JAVA — but not here

  • Programming a robot is hard, but rewarding. Unlike software in the virtual world, it is influenced by the real world, and does not always have the expected results
Work Plan

- **[Week 1]**
  - Read the ROS Introduction: [http://wiki.ros.org/ROS/Introduction](http://wiki.ros.org/ROS/Introduction)
  - (optional) Install ROS Kinetic on your laptop. Requires Ubuntu 16.04 (recommended) or another Linux distribution. See the ROS installation instructions. You will find installation instructions at [http://wiki.ros.org/ROS/Installation](http://wiki.ros.org/ROS/Installation)
  - Alternatively, you can use machines with ROS installed in the robotics lab (the office facing the student secretary offices)
- **[Week 2-3]** Run all (about 20) the beginner tutorials (you can choose either C++ or Python, where relevant)
- **[Week 4-6]** Work on basic robot motions in Gazebo.
- **[Week 7]** Assignment 2 due 25/11 + Class tutorial on robot use 25/11
- **[Week 7-9]** Implement simple motion between walls and object identification
- **[Week 10]** Assignment 3 due 16/12
- **[Week 10+]** Implement project on robot: maze challenge
- **[Semester break]** Demo project, upload code description to robot wiki
Assignments

• Assignment 1: (Requires ROS And OpenCV) simple image manipulation using ROS and OpenCV

• Assignment 2: (Requires ROS, Gazebo, Robot Model) Implement the following abilities in simulation: move forward, turn around, identify object based on color and return distance

• Assignment 3: Assignment 2, but on the real robot

• Assignment 4: Final project
Course Info

• Instructors:
  • Prof. Ronen Brafman
    • Office: 37/209
    • Email: brafman@cs.bgu.ac.il
    • Office hours: Sun/Wed 1-2
  • Mr. Shai Givati - Robotics Lab Engineer
    • Office 37/-103
    • Email: shaigiv@post.bgu.ac.il
    • Office hours: Sunday 10-12. Best to e-mail before coming.

• Meetings:
  • Intro (this) + 2 tutorials on robot use
  • Frontal checks and demo by appointment with instructor
  • Before you can use the robot, you will need to sign a form in which you verify that you understand the risks and that you will follow safety procedures

• Grade: Assignment 1 - 10%, Assignment 2 - 15%, Assignment 3 - 25%, Project - 50%
  • Assignment 1 (individual) — 10%
  • Assignment 2 (Individual) — 20%
  • Assignment 3 (3-4) — 30%
  • Assignment 4 (3-4) — 40%
Sources

- Much material is available online.
    - [http://file.allitebooks.com/20151124/Programming%20Robots%20with%20ROS.pdf](http://file.allitebooks.com/20151124/Programming%20Robots%20with%20ROS.pdf)
  - Book: ROS Robot Programming — specific for TurtleBot3
  - ROS Tutorial Videos [https://www.youtube.com/playlist?list=PLRG6WP3c31_U7TFGduEIJWVtkOw6AjjFf](https://www.youtube.com/playlist?list=PLRG6WP3c31_U7TFGduEIJWVtkOw6AjjFf)
ROS

• An open source, operating system for robots

• Provides following services:
  • hardware abstraction
  • low-level device control
  • implementation of commonly used functionality
  • message passing between processes
  • package management
  • Tools and libraries for obtaining, building, writing, and running code across multiple computers

• ROS provides many packages for diverse robotic tasks, starting with manipulation and navigation, to mapping environments and doing automated planning
ROS Distributed Architecture

Courtesy of Roi Yehoshua
Run-time

• A peer-to-peer network of processes (potentially distributed over multiple machines) that are loosely coupled and use the ROS communication infrastructure

  • Synchronous communication over services

  • Asynchronous communication over topics
ROS Core Concepts

• Nodes
• Messages and Topics
• Services
• ROS Master
• Parameters
• Stacks and packages

Courtesy of Roi Yehoshua
ROS Nodes

- Single-purposed executable programs
  - e.g. sensor driver(s), actuator driver(s), mapper, planner, UI, etc.
- Modular design
  - Individually compiled, executed, and managed
- Nodes are written using a ROS client library
  - roscpp - C++ client library
  - rospy - python client library
- Nodes can publish or subscribe to a Topic
- Nodes can also provide or use a Service

Courtesy of Roi Yehoshua
ROS Topics

• Nodes communicate with each other by publishing messages to topics
• Publish/Subscribe model: 1-to-N broadcasting

Courtesy of Roi Yehoshua
Topics

- Topics: Messages are routed via publish/subscribe semantics
- A node sends a message by publishing to a topic
- The topic is a name that is used to identify the content of a message
- A node interested in certain messages will subscribe to this topic
- Multiple nodes can publish/subscribe to the same topic
- Publishers/subscribers are unaware of each other
- A form of asynchronous communication
- Example: a sensor node publishes its reading to a topic. Other nodes can process it. They can publish the processed data to a different topic. Controller nodes can use that to decide how to control the motors
ROS Messages

• Strictly-typed data structures for inter-node communication
• For example, geometry_msgs/Twist is used to express velocity broken into linear and angular parts:

  Vector3 linear
  Vector3 angular

• Vector3 is another message type composed of:

  float64 x
  float64 y
  float64 z

Courtesy of Roi Yehoshua
Demo
ROS Services

• Synchronous inter-node transactions / RPC
• Service/Client model: 1-to-1 request-response
• Service roles:
  – carry out remote computation
  – trigger functionality / behavior
• Example:
  – map_server/static_map - retrieves the current grid map used by the robot for navigation

Courtesy of Roi Yehoshua
File System Support

- Packages: the main unit for organizing software in ROS. Contains runtime processes (nodes), datasets, configuration, etc.

- This is the most granular thing you can build and release

- Message types: message descriptions that define the data structures for messages sent in ROS

- Service types: service description that define the request and response data structure for services
ROS Packages

• Software in ROS is organized in packages.
• A package contains one or more nodes and provides a ROS interface
• Most of ROS packages are hosted in GitHub

Courtesy of Roi Yehoshua
ROS Package System

Taken from Sachin Chitta and Radu Rusu (Willow Garage)
ROS Master

- Enable ROS nodes to locate one another
- Think of it as a ROS directory service, sort of DNS
  - Provides naming & registration services for nodes, topics, services, etc

Courtesy of Roi Yehoshua
Parameter Server

- A shared, multi-variate dictionary that is accessible via network APIs.
- Best used for static, non-binary data such as configuration parameters.
- Runs inside the ROS master
Various Supplied Capabilities

• Coordinate transforms — useful for geometric reasoning

• ActionLib — an interface for interacting with preemptable actions (such as move to a location, perform scan)

  • This is like a service, but one that may take a long time, and requires periodic feedback about progress and the ability to stop the service

  • One can specify goals, feedback, and result

• Different classes of messages (actions, diagnostics, etc.)

• Plugin support — enables loading/unloading plugins dynamically without the application being aware of these earlier.

• Filters — various filters for data processing

• Robot models
Easy(?) Integration with Popular Open Source Projects

- Gazebo — a 3D robot simulator. A model of our Komodo robot already exists

- OpenCV — a large machine vision library

- PointCloudLibrary — library for manipulation and processing 3d data and depth image. For example, the Kinect we have returns this type of data

- MoveIt — a motion planning library
Robot Programming

- The general idea behind robot control is simple:
  - Sense-Process-Decide-Act loop
    - Sense the world to get information
    - Process the information (for example, process an image to identify the location of an object)
    - Decide what to do (sometimes trivial)
    - Send message to the actuators to carry out the decision
  - Repeat — you are not sure what was the effect of your actions, and the world can change, so you need to sense, process,…
Some Basic Concepts in Robotics

• Sensor types: RGB camera, 3D camera, sonar, laser scanner

• Actuators: motors that drive wheels, change arm angles, gripper

• Odometry, Navigation, localization, mapping, SLAM

• Coordinate systems, coordinate transforms

• Motion planning
• If you look around a bit, you’ll see that robots are probably the next big thing — and it will be very big.

• If you enjoy this course, start thinking about a graduate degree in this area — in a few years, this is likely to be a highly sought-after skill