Cognitive Systems
Practical Exercises - Introduction

Prof. Dr.-Ing. habil. Alois Knoll
Alexander Perzylo, Nikhil Somani, Ahmed Ghazi, Hendrik Walzel
Robotics and Embedded Systems (Informatik VI)
TUM / fortiss
Outline

• Demonstrator overview
• Preparation for practical session
• Basic introduction to ROS
• ROS components for demonstrator
• Live Demo
• ROS tutorial
Demonstrator Overview

Gripper Bot

- Robot Motion
- Grasping

=> Pick and Place Task

Resources

- 3DOF Robot
- Gripper
- Table with work pieces
Demonstrator Overview

Cam Bot
- Object Detection
- Robot Motion

Resources
- 3DOF Robot
- Calibrated Monocular Camera
Preparation

Requirements

• Group work
• Min 1 C++/Java GURU per group :-)  
• Virtual box installation - [https://www.virtualbox.org](https://www.virtualbox.org) or (native Ubuntu 12.04)  
• ROS Hydro installation
Basic Introduction to ROS

- Meta operating system serving as a middleware
- Linux OS, Limited support for Windows
- Open Source under BSD License (core parts)
- Developed by Willowgarage
- Provides communication infrastructure for information exchange
  - Synchronous (Client/Server)
  - Asynchronous (Publish/Subscribe)
- Custom text based Message and Service description language
- Central resource registry called roscore
- C++/Java/Python/Prolog/Lisp/JavaScript
ROS Packages

- Container for software components
- cmake based build system
- Dependencies on other ROS packages through package.xml file
- Command line tool catkin_make builds ROS packages
- Command line tool rossrun to run ROS packages
- Command line tool roslaunch to startup multiple nodes which are defined in an XML based launch file
ROS Messages

• Messages are exchanged through ROS topics
• ROS topic is identified by a topic name
• Publish and Subscribe mechanisms for exchanging data
• ROS topic is connected to one specific message type
• Text based custom message definitions
• Availability of basic data types (int, float, bool etc) which can be aggregated to custom structures
ROS Service

- Hosted service is identified by a service name
- ROS service is connected to one specific service type
- Text based custom service definition consisting of request and response definitions
- Service implementation through callback functions
- ROS Client is blocked until response is generated
ROS Utilities

- `roscd <package_name>`: change to directory of ros package
- `catkin_make --pkg <package_name>`: make ros package
- `rosservice`
  - `rosservice list`: lists all registered services
  - `rosservice call`: manually issue a service call from a command line
- `rostopic`
  - `rostopic list`: lists all registered topics
  - `rostopic echo <topicname>`: listen to a rostopic
ROS Components for Demonstrator

ROS workspace location: ~/catkin_cogsys-ros/
Collection of ros packages serving as software building blocks for the demonstrators:

<table>
<thead>
<tr>
<th>Cambot</th>
<th>Gripperbot</th>
</tr>
</thead>
<tbody>
<tr>
<td>camera</td>
<td></td>
</tr>
<tr>
<td>object_detector</td>
<td>gripper_control</td>
</tr>
<tr>
<td>robot_control</td>
<td>robot_control</td>
</tr>
<tr>
<td>messages_services</td>
<td>messages_services</td>
</tr>
<tr>
<td>comm_lib</td>
<td>comm_lib</td>
</tr>
<tr>
<td>cogsys_visualization</td>
<td>cogsys_visualization</td>
</tr>
</tbody>
</table>
ROS Package

camera
• Function: Grabbing images from camera and publishing them
• Interface
  – ROS Topic: /camera/image
  – ROS Service: /camers_srvs/CameraInfo
• Message:
  – sensor_msgs/Image
  – device_msgs/CameraCalibData
ROS Package

object_detector
- Function: Detects 3D position of workpieces on the table wrt reference frame
- Interface: ROS Topic /object_detector/objects_data
- Msg: /actor_msgs/ActorVec
  - ActorVec: array of message actor_msg/Actor
ROS Package

robot_control
• Function: Hosts services to move robot in cartesian space (or joint space)
• Interface:
  – ROS Service /cambot_control/move_to_cs
  – ROS Service /gripperbot_control/move_to_cs
• Service
  – MoveToCS
  – request
    • string effector (must be „wrist“, „gripper“ or „camera“)
    • float32 x,y,z
  – response
    • bool success (if target position could be reached or not)
ROS Package

gripper_control
• Function: Hosts services to open/close gripper
• Interface:
  – ROS Service /gripper_control/open_gripper
  – ROS Service /gripper_control/close_gripper
• Service definitions
  – gripper_control_srvs/OpenGripper (empty list)
  – gripper_control_srvs/CloseGripper (empty list)
ROS Package

robot_control_viz
- Function: Same functionality as the robot_control package, but in simulation

cogsys_visualization
- Function: Launches the visualization and scene for robot_control_viz
ROS Package

In folder messages_services:
• Custom message type and service definitions

comm_lib
• Convenience wrappers and utility functions
LIVE DEMO?
Running the demo on your system (if you have a working camera connected)

```bash
somani@fortiss-n-063:~$ roscore

somani@fortiss-n-063:~$ roslaunch cogsSys_visualization cogsSys_demonstrator.launch

somani@fortiss-n-063:~$ roslaunch robot_control_viz camrobot.launch

somani@fortiss-n-063:~$ roslaunch robot_control_viz gripperbot.launch

somani@fortiss-n-063:~$ rosruncamera camera

somani@fortiss-n-063:~$ rosrunobject_detector objdetect

somani@fortiss-n-063:~$ rosrundemocogsSys_cpp democogsSys_cpp
```
Running the demo on your system in simulation (as shown in session 1)
Hands on Part

ROS TUTORIAL

http://wiki.ros.org/ROS/Tutorials

(Please use hydro release ONLY!)