ROVI2 - Exercise for ROS Lecture

Thomas Fridolin Iversen (thfi@mmmi.sdu.dk) and
Mikkel Tang Thomsen (mtt@mmmi.sdu.dk)

February 17, 2015

1 Introduction to exercise

The purpose of this exercise is getting you started using the ROS framework and to use it in combination with RobWork, OpenCV and PCL. The exercise is structured as follows. The first section, section 2, an overview of a bunch of ROS tutorial that you should go through to get a basic understanding of the framework. In section 3 you should try out some example code, we provide, and inspect the code to get an understanding of the communication between nodes. In section 4 you should create your own ROS package and make nodes that communicate with the robot platform in RoboLab. Finally in section 5, some notes on setting up your virtual machine for communication with the RoboLab platforms. It is suggested to do as follows:

1. Go through the ROS tutorials
2. Inspect and run the provided code
3. Create a new package to communicate with the robot platform in RoboLab and create nodes to receive images and pointclouds as well as a node to communicate with the robot.
4. Try out your node at the robot platform and verify that you can control the robot and receive sensor data.

The steps are explained in more details in the sections below.

Note on virtual machine The exercise is made for the virtual machine and is suggested to use this as it will make life easier for you. Secondly it should be noted as already pointed out earlier, that we do not provide support for other platform than the virtual machine. A folder denoted catkin_ws is already present at the virtual machine home directory. It is suggested you delete this such that you start out with clean catkin workspace that you make by yourself.

2 Suggested ROS tutorials

It is suggested that you at least do the tutorials listed below. To some of the tutorials a note is added about what to focus on.

The ROS tutorials can be found at `www.ros.org` ⇒ wiki ⇒ Tutorials or go to: `http://wiki.ros.org/ROS/Tutorials`

The ROS tutorials are quite comprehensive and tend to go into details with some of the less important features and commands as well as the important ones. A list of suggested tutorials and what to focus on in the tutorials is therefore given below.

Beginner Level

1. Installing and Configuring Your ROS Environment

3. Creating a ROS package This tutorial may be skipped in the beginning, but it is worth a look here, when you have to create your own package.

4. Building a ROS Package The tutorial in itself is less important, but the important thing to notice in this tutorial is that you donâĂŹt build ROS using the traditional make commands. But instead using the catkin_make command. This command is executed within the root of the catkin workspace.

5. Understanding ROS Nodes Focus on roscore and rosrun commands
roscore : starts the ROS kernel
rosrun [package_name] [node_name] : starts ROS node [node_name] from ROS package [package_name]

6. Understanding ROS Topics This tutorial covers a lot of ways to extract data about the communication between ROS nodes. Its enough to read through this tutorial but take special care and tryout the commands explained in section: 1.2.2 , 1.2.3 , 1.2.4 , 1.3 as these are used fairly often when debugging and using ROS.

7. Understanding ROS Services and Parameters Sort of the same as above. Just read through this tutorial and notice that **rosservice list** gives you a list of the services running.

10. Creating a ROS msg and srv

11. Writing a Simple Publisher and Subscriber (C++)

14. Writing a Simple Service and Client (C++)

17. Recording and playing back data This is not important now, but its a nice feature, which can come in handy at a later point in your projects. Just notice what this feature is all about so that you know it exist.

Intermediate level

4. Running ROS across multiple machines This is not important now, but you will need this at a later point when using ROS across multiple computers (at the robot platforms in RoboLab). Notice that this tutorial explains how to do this.

3 Inspecting and running the provided code

A code example is provided where ROS, RobWork and OpenCV are all used in the same project. The code can be obtained from Blackboard and need to be transferred to the virtual machine and compiled. The example consist of two things:

1. A set of ROS packages

2. Workcell descriptions for RobWork
3. Files with .srv message files to be used with the different robots

3.1 ROS packages

The five provided packages are listed below.

camera_plugin Contains a plug-in for RobWorkStudio. The plug-in is able to show the camera images received inside RobWorkStudio

monocamera_dummy A package with a node from where images can be extracted through ROS. The node is supposed to simulate a real camera and has the same interface as will be used on the real workcell.

pa10_dummy Like the monocamera_dummy this package has a node, which simulates the PA10 robot. Joint configurations can be sent and received from this node.

pa10_plugin This plug-in is used be RobWorkStudio and can be used to transmit and receive joint configurations to pa10_dummy node.

serial A package required by the pa10_dummy. The package contains a serial communication protocol.

3.2 The workcell description:

Models of the physical workcells in RoboLab. For this exercise you should use the PA10 workcell as the plugins are made for this. When you continue creating your own package for the communication with the real robot you should utilise the workcell file according to the workcell you are working on.

- PA10
- Staubli RX60
- UR5

3.3 Using and compiling the source code

The source code and workcells are available through Blackboard. Start by creating and initialising the catkin workspace, on the virtual machine, using **catkin_init_workspace** in the **catkin_ws/src** folder. If you have already done that as a part of the tutorials you can skip this step and directly copy the packages from the source code files into your catkin workspace src-folder. Next, we need to ensure that ROS is sourced properly. Do this by editing your .bashrc file in your home directory. In a new terminal (ctrl+alt+t), type:

```
gedit .bashrc
```

Add the following line or lines to the end of the file if they are not already there:

```
# ROS
source /opt/ros/indigo/setup.bash
source ~/catkin_ws/devel/setup.bash
```

By doing so these are sourced when you start at new terminal. To verify that the ROS_PACKAGE_PATH is setup correctly, open a new terminal and type:

```
echo  $ROS_PACKAGE_PATH
```

This should result in something similar to this:

```
/home/student/catkin_workspace/src:/opt/ros/indigo/share:/opt/ros/indigo/stacks
```

When the paths for ROS is setup properly. Then go to the root of the workspace and build the workspace using **catkin_make**. The serial package might have to be compiled before the pa10_dummy package. If the compilation is terminated due to inability to find RobWork and RobWorkStudio you should add the following lines to your .bashrc file:

```
export RW_ROOT=~/RobWork/RobWork
export RWS_ROOT=~/RobWork/RobWorkStudio
```

Remember to restart the terminal such that the .bashrc file is sourced and retry compilation.

After the workspace is successfully compiled go to the binary folder of RobWorkStudio (RobWork/RobWorkStudio/bin/relwithdebinfo). Modify the RobWorkStudio.ini configuration file such that both the pa10_dummy and monocamera_dummy plugin are loaded on start. The configuration files should point the plugins, which are located in catkin_ws/devel/lib. Altnernatively you can load the plugins manually from RobWorkStudio by choosing menupoint: plugins $\Rightarrow$ load plugin.

3.4 Launching the code

In order to launch the system do as follows:

```
# Run the roscore
roscore
# Open a new terminal (ctrl+alt+t) and run the simulated PA10 node pa10_dummy
rosrun pa10_dummy pa10_dummy
# Open another terminal and run monocamera_dummy
rosrun monocamera_dummy monocamera_dummy
# Go to RobWorkStudio binary directory
cd ~/RobWork/RobWorkStudio/bin/relwithdebinfo
# Run RobWorkStudio with parameters if you want to load a workcell and
# a specific RobWorkStudio.ini file.
./RobWorkStudio --ini-file RobWorkStudio.ini "<path-to-workcell-file>"
# Otherwise run without parameters then it will use the RobWorkStudio.ini
# file located  from where you call RobWorkStudio
./RobWorkStudio
```

Play around with the program and plugins. Start a new terminal where you use some of the ROS commands to observe the communication, such as **rosservice lists** and **rostopic lists**. You should also try to look at some of the ROS graphical tools such as **rqt** and **rqt_graph**, **rviz** etc.

Finally you should have a look at the source code and examine how messages are defined and how the ROS communication is setup.

4 Creating a new package and connecting to the robot setups in RoboLab

The last exercise is where you are to make use of the knowledge you have acquired in the previous ROS-tutorials and the example plugin for the virtual machine and RobWork. The aim of this exercise is to give you a good starting point for the future project work on the real platforms. You should follow the steps as they come:

Getting started

1. Create a ROS package

Communicate with the cameras

2. Write a ROS node that capture images from the BumbleBee2 stereo camera and saves the images to your computer. The topic of the type **sensor_msgs::Image**

3. Write a ROS node (Could be the same as the one capturing stereo images) that capture PointClouds from the RGB-D device and saves the pointclouds to your computer. For additional information of dealing with PointClouds in ROS, see `http://wiki.ros.org/pcl/Overview`. The message that is output from the robot platform is of the type **sensor_msgs::PointCloud2**. Be aware that if you want to use pcl in ROS your node should depend on the package **pcl_ros**.

Communicate with the robots

4. Write a ROS node that reads out the configuration of the robot. Depending on the robot platform you are working on, you should find the proper service file(s) on Blackboard.

5. Choose three points that the Robot should move between be aware that the robot can and will collide with the environment if you choices are bad. Make the robot move between the three points

6. While moving the robot between the three points, record the configuration during the path and plot it to see the actual path compared to the three points you put into it.

Communicating with PG70 gripper (Optional)

7. Write a ROS node that reads the configuration of the gripper joint and is able to open and close the jaws. The service is available on Blackboard.

Connecting to the robotic platforms in RoboLab

8. Start the platform as explained by Jens Cortsen, including the ROS nodes that should be running on the platform computer.

9. Connect your pc by the cable next to the platform computer and setup the network on your virtual machine, see section 5.

10. Ensure you have connected to the local network by pinging the RoboLab computer's ip.

11. Verify that everything is setup correctly by using **rostopic list**, **rqt** or similar (on your pc) to see that the published topics are available.

12. Test your nodes and their ability to perform the intended task.

5 Setting up the virtual machine network for connecting to the RoboLab Platform networks

A few things needs to be configured before ROS can be used across multiple machines. First of all hosts file need to be configured. The hosts file associates ip addresses with names for conveniens.
From a terminal type:

```
sudo gedit /etc/hosts
```

In the file insert the following lines in the top of the file, exchange <robot-name-of-your-choice> with the robot in the cell, PA10, UR5 or RX60 (Check the ip adresses of the actual robotplatform you are using, at the pc open a terminal and type ifconfig to see the ip adress):

```
127.0.0.1 localhost
172.16.1.20 <robot-name-of-your-choice>
172.16.1.19 ubuntuRW
```

Save the file and exit.
Next up a network connection with a static ip-address have to be established. From the newtwork icon at the top menu select edit connections. Under the wired tap select add new connection' Under the tap IPv4 Settings change the method to 'Manual' and add:

```
ip-address:  172.16.1.19
netmask:     255.255.255.0
Gateway:     0.0.0.0
```

Select this configuration when connecting your computer to the switch.
Ensure that the network is properly configured by trying to ping the computer.
From a terminal type:

```
ping <robot-name-of-your-choice>
```

Finally ROS need to know where to look for the ROS master, when it is on an external computer.
Open a terminal and do as follows:

```
# Go to your home directory
cd ~
# Edit the .bashrc file
gedit .bashrc
# Add the following line to the end of the .bashrc files as a reference to the ROS master
export ROS_MASTER_URI=http://<robot-name-of-your-choice>:11311
```

6 Acknowledgement

Based on tutorial by Johan Sund Larsen (josl@mmmi.sdu.dk)