COMP-417 Assignment Ø

The objective of this assignment is for you to become familiar with the Robot Operating System (ROS) middleware, which you will be using in future assignments to program various fundamental algorithms in robotics.

IMPORTANT: If at any time you run into problems, please do not hesitate to contact the TA for assistance (malika@cim.mcgill.ca and msandeep@cim.mcgill.ca). The class on Wed. September 18th, 2013 will cover various ROS-related topics, so you will be expected to have fully completed this assignment prior to then.

1. Installing ROS

ROS is officially supported on Ubuntu Linux (www.ubuntu.com), so if you would like to install ROS on your personal computer, we highly recommend that you install it alongside the latest version of Ubuntu, 12.04 LTS / Precise Pangolin. If you would like to install Ubuntu Linux on your personal computer (or dual-booting or triple-booting), but are experiencing difficulties, please contact the TA for support. The following assume that you are installing ROS on Ubuntu.

The Ubuntu Linux computers in Trottier 3rd floor should all have ROS Fuerte installed already, so you may choose to use those machines to solve your assignments. Please notify us if you need a particular ROS package that is not installed on these machines. If you plan to use these computers for your assignments, you may skip to part 2.

Please follow the following instructions to install ROS:

http://www.ros.org/wiki/fuerte/Installation/Ubuntu

If you have sufficient hard drive space, ideally you should install ros-fuerte-desktop-full version.

2. Learning ROS

Now go through the following tutorials to learn the basics of ROS:

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

Whenever an exercise is offered in both Python and C++, please go through the C++ version. Also, pay particular attention in completing Section 3 of Tutorial 1, even if you are using the Trottier 3rd floor computers.

Note: When running the tutorials always select **fuerte** and **rosbuild**. **DO NOT USE catkin.**

As we discussed in class, we will be providing sample code in C++ for future assignments, but you are allowed to solve the assignment in any language that you like, as long as your solution demonstrates the correct results.

3. Installing and learning StageROS

We will be using the Stage robot simulator software for most of our assignments in this class. The StageROS ROS package allows ROS programs to interact with Stage, and is packaged with the ros-fuerte-desktop-full package distribution. Please go through the following StageROS tutorial:

http://www.ros.org/wiki/stage/Tutorials/SimulatingOneRobot

Note that in step 2 of the tutorial, you are asked to install the teleop_base ROS package. In the future, you may need to install new ROS packages. We recommend that you first try installing a package through your OS' packaging tool (e.g. Apt / Synaptic). If that is not an option, then you can also try installing it semi-automatically using rosws. We will now install additional ROS packages that are needed by the StageROS tutorial using both methods.

3.1 Installing ROS package using OS' package manager

Assuming that you are using Ubuntu Linux, you can install the ROS joystick package, called Joy, by running the following commands in a terminal:

```
> sudo apt-get install ros-fuerte-joystick-drivers
```

As a side-note, you can search for available packages in Linux by executing:

```
> apt-cache search [query terms, e.g. "ros joystick drivers"]
```

You may alternatively choose to use a number of alternative GUI/command-line package managers (e.g. Ubuntu Software Centre, Synaptic) to search for and install packages.

3.2 Installing ROS packages using rosws

Next we will install the control_toolbox and teleop_base ROS packages:

1. Go to www.ros.org
2. Search for control_toolbox in the top-right search box
3. Make note of the following line on the control_toolbox ROS webpage:

   ```
   Source: svn https://code.ros.org/svn/wg-ros-pkg/stacks/pr2_controllers/branches/pr2_controllers-1.4/control_toolbox
   ```

This tells us that the `control_toolbox` package is using SVN as its revision control software, and also provides us with the corresponding repository link.

4. In a terminal, execute the following command:

```
> rosws set control_toolbox --svn https://code.ros.org/svn/wg-ros-pkg/stacks/pr2_controllers/branches/pr2_controllers-1.4/control_toolbox
```

```
> rosws update control_toolbox
```

5. Repeat steps 1-3 for the `teleop_base` ROS package, and then execute the following commands in a terminal:

```
> rosws set teleop_base --svn https://code.ros.org/svn/wg-ros-pkg/branches/trunk_cturtle/sandbox/teleop_base
```

```
> rosws update teleop_base
```

6. Next you will need to re-initialize your ROS environment:

```
> source ~/fuerte_workspace/setup.bash
```

7. You will likely experience in the future that some ROS packages may need to be updated or tweaked to make them compatible with the latest versions of other dependent libraries. Assuming that we are using Ubuntu Linux 12.04 LTS and ROS Fuerte, we will need to make the following modifications due to changes in the APIs for tinyxml and Joy ROS packages:

   - Remove line 9 in `~/fuerte_workspace/control_toolbox/manifest.xml`:

```
<depend package=" tinyxml" />
```

   - Change line 38 of `~/fuerte_workspace/control_toolbox/src/pid.cpp` from:

```
#include "tinyxml/tinyxml.h"
```

to:

```
#include <tinyxml.h>
```

   - Change line 41 of `~/fuerte_workspace/control_toolbox/include/control_toolbox/sinusoid.h` from:

```
#include <tinyxml/tinyxml.h>
```

to:

```
#include <tinyxml.h>
```

   - Change line 8 in `~/fuerte_workspace/teleop_base/manifest.xml` from:

```
<depend package=" joy" />
```

to:

```
<depend package=" sensor_msgs" />
```

- Change line 37 in ~/fuerte_workspace/teleop_base/teleop_base.cpp from:

```
#include “joy/Joy.h”
```

to:

```
#include “sensor_msgs/Joy.h”
```

- Change line 121 in ~/fuerte_workspace/teleop_base/teleop_base.cpp from:

```
void joy_cb(const joy::Joy::ConstPtr& joy_msg)
```

to:

```
void joy_cb(const sensor_msgs::Joy::ConstPtr& joy_msg)
```

- In ~/fuerte_workspace/teleop_base/teleop_base.cpp, find and change all instances of the following:

```
from joy_msg->get_buttons_size() to joy_msg->buttons.size() [lines 126 and 132]

from joy_msg->get_axes_size() to joy_msg->axes.size() [lines 137, 141, and 145]
```

- Change line 104 in ~/fuerte_workspace/teleop_base/src/teleop_base_keyboard.cpp from:

```
boost::thread t = boost::thread::thread(boost::bind(&TBK_Node::keyboardLoop, &tbk));
```

to:

```
boost::thread t = boost::thread(boost::bind(&TBK_Node::keyboardLoop, &tbk));
```

8. You can now build both ROS packages by running:

```
> rosmake control_toolbox

> rosmake teleop_base
```

4. Learning StageROS

Now go through the following tutorial (from step 3 onwards):

http://www.ros.org/wiki/stage/Tutorials/SimulatingOneRobot

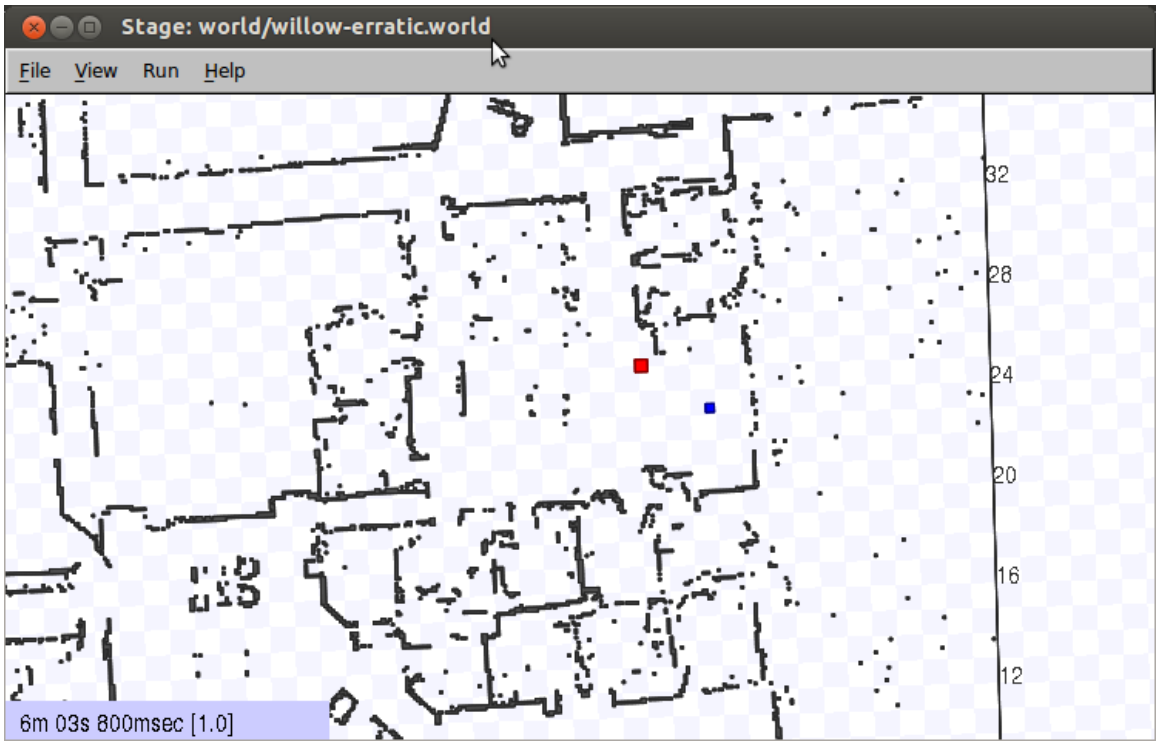
Note that you will need to execute `roscore`, `stageros`, `teleop_base`, and `rviz` simultaneously in separate terminals. Also, in step 6 you will need to use an updated `rviz` configuration file, which you can download from:

http://www.cim.mcgill.ca/~yiannis/COMP417_2012_A0P4.vcg

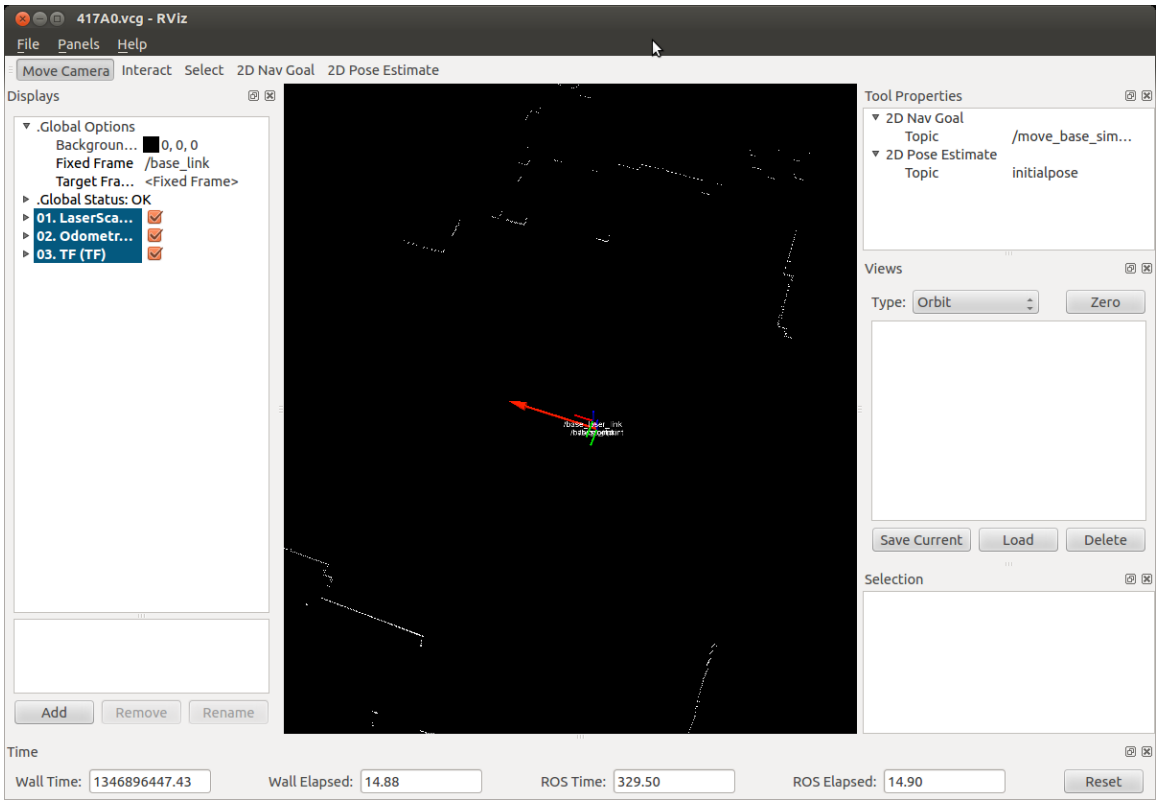
Assuming that you have saved `417A0.vcg` to your Desktop, you can start rviz using the following terminal command:

```
> rosrun rviz rviz -d ~/Desktop/COMP417_2012_A0P4.vcg
```

You may need to move/zoom the world in the Stage window (using the left/middle mouse buttons) to see a large red box and a smaller blue square robot:



Your teleop_base commands will affect the blue robot only. Also, as you move the robot around, you should observe updated laser scan points in the rviz window, which should look like the following:


417A0.vcg - RViz
File Panels Help
Move Camera Interact Select 2D Nav Goal 2D Pose Estimate
Displays
.Global Options
Backgroun... 0, 0, 0
Fixed Frame /base_link
Target Fra... <Fixed Frame>
.Global Status: OK
01. LaserSca...
02. Odometr...
03. TF (TF)
Add Remove Rename
Tool Properties
2D Nav Goal
Topic /move_base_sim...
2D Pose Estimate
Topic initialpose
Views
Type: Orbit Zero
Save Current Load Delete
Selection
Time
Wall Time: 1346896447.43 Wall Elapsed: 14.88 ROS Time: 329.50 ROS Elapsed: 14.90 Reset