Mobile Robot Control / State Machines

Contents

1 Overview

The aim of this Practical Exercise is to introduce you to some simple robot control strategies using State Machines, while at the same time introducing you to ROS.

2 Simulation Environment Setup

In order to get the robot simulation environment working, three components will need to be installed. All installations need to occur on a single Linux machine. Advanced users would later be able to split the controller and simulation between two different computers, or even redirect the controller to a real robot.

- Firstly the Robot Simulation Environment. For this exercise, we will be using Coppelia Sim. The free Educational Licence for downloading can be found at: https://coppeliarobotics.com/ubuntuVersions

  In order to enable the ROS Plugin, the compiled plug-in: libsimExtROSInterface.so needs to be copied into the correct directory. You will find the file in:

  `CoppeliaSim/compiledROSPlugins`

  Copy this file into:

  `CoppeliaSim/`

- Secondly, MATLAB is will be used for programming the controller. The link to MATLAB download can be found through the mathworks website. The University of Applied Sciences Emden Leer has a Campus Licence allowing students free download. The portal to access the service can be found here: https://de.mathworks.com/academia/tah-portal/hochschule-emden-leer-40712566.html. During the installation of MATLAB ensure the following toolboxes are installed:

  - Simulink
  - Stateflow
  - Robotics Toolbox
  - ROS Toolbox

- Finally, ROS needs to be installed on the Linux machine. For this process, following the instructions found at: http://wiki.ros.org/melodic/Installation/Ubuntu You need to complete the installation down to the end of 1.6 Environment setup.

3 Environment Startup and Testing

In a new Terminal Window enter the command:

```
user@computer:~$ roscore
```

The ROS server should startup with a response similar to:

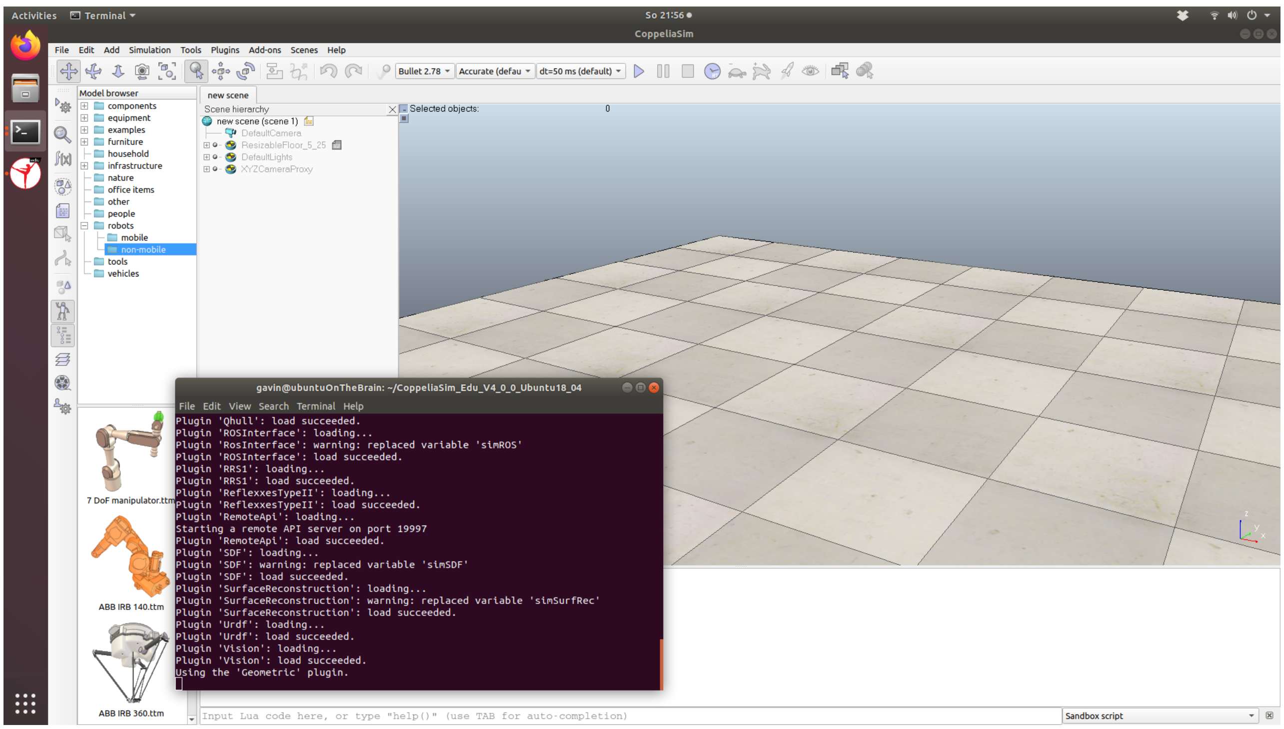
```
... logging to /home/user/.ros/log/43963834-66fd-11ea-93db-305a3a0046ba/roslaunch-2306.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.xxx.xxx:52867/
ros_comm version 1.14.3


SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.3

NODES

auto-starting new master
process[master]: started with pid [2316]
ROS_MASTER_URI=http://192.168.xxx.xxx:11311/

setting /run_id to 43963834-66fd-11ea-93db-305a3a0046ba
process[rosout-1]: started with pid [2327]
started core service [/rosout]
```

Step two is getting the Simulation Environment to access the started ROS server. By starting up CoppeliaSim using the command:

```
./coppeliaSim.sh
```

in a terminal running in the installation directory, you should see in the terminal following the startup:

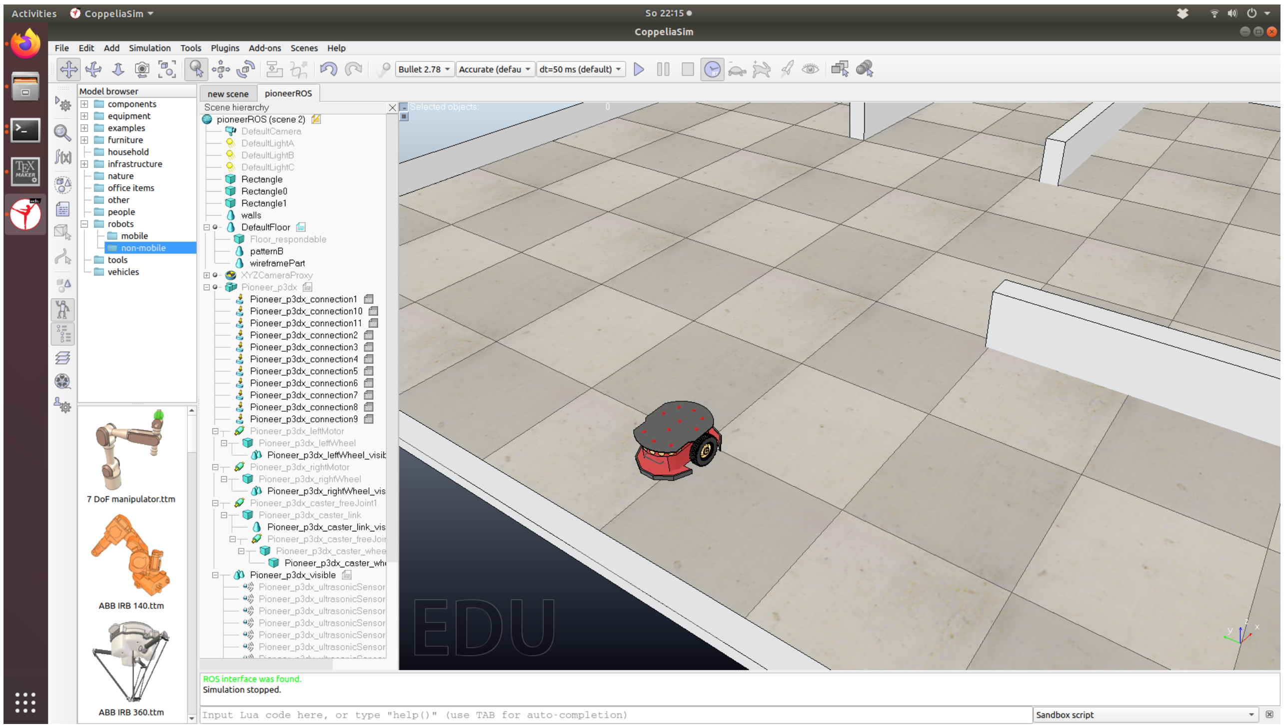
```
...
Plugin 'ROSInterface': loading...
Plugin 'ROSInterface': load succeeded.
...
```



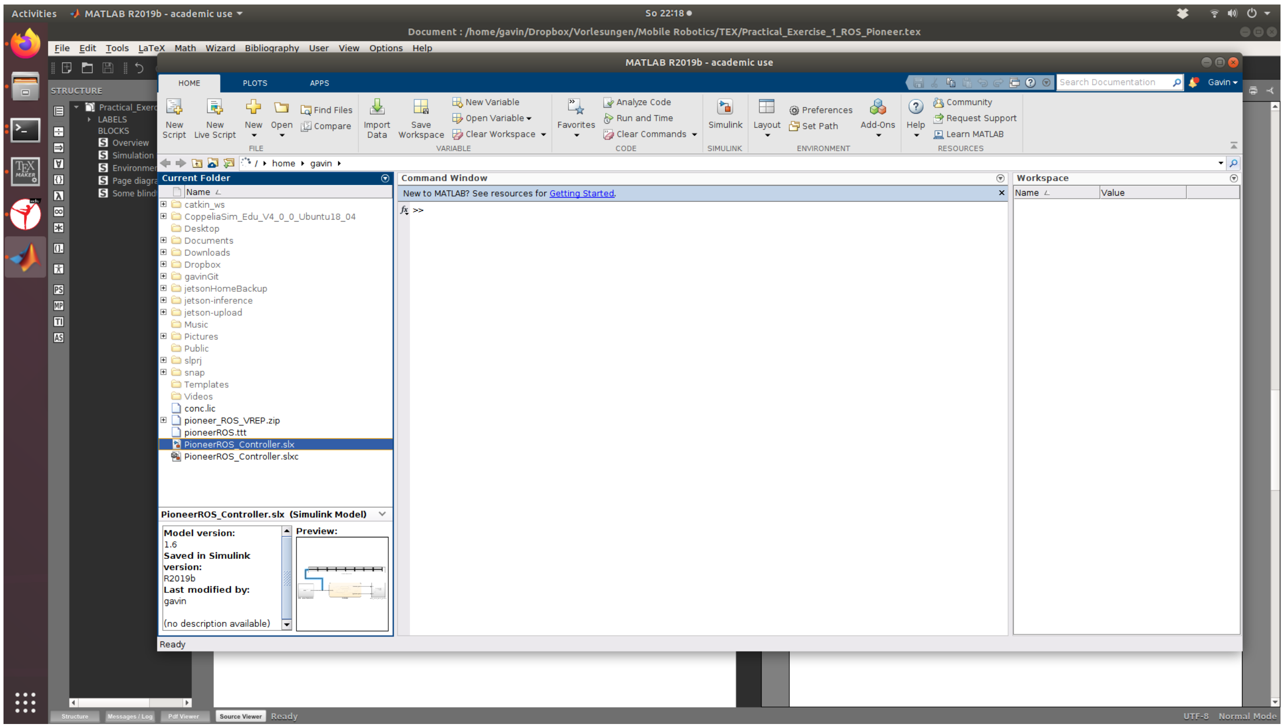
In Moodle a Scenario can be found in the Pracital Section under the name: "Practial 1 - Pioneer ROS VREP Model". Download this ZIP File, extract the two files, and in Coppelia Sim load the pioneerROS.ttt scenario.

When Play in pressed, the following should be visible in the command window below the scenario screen:

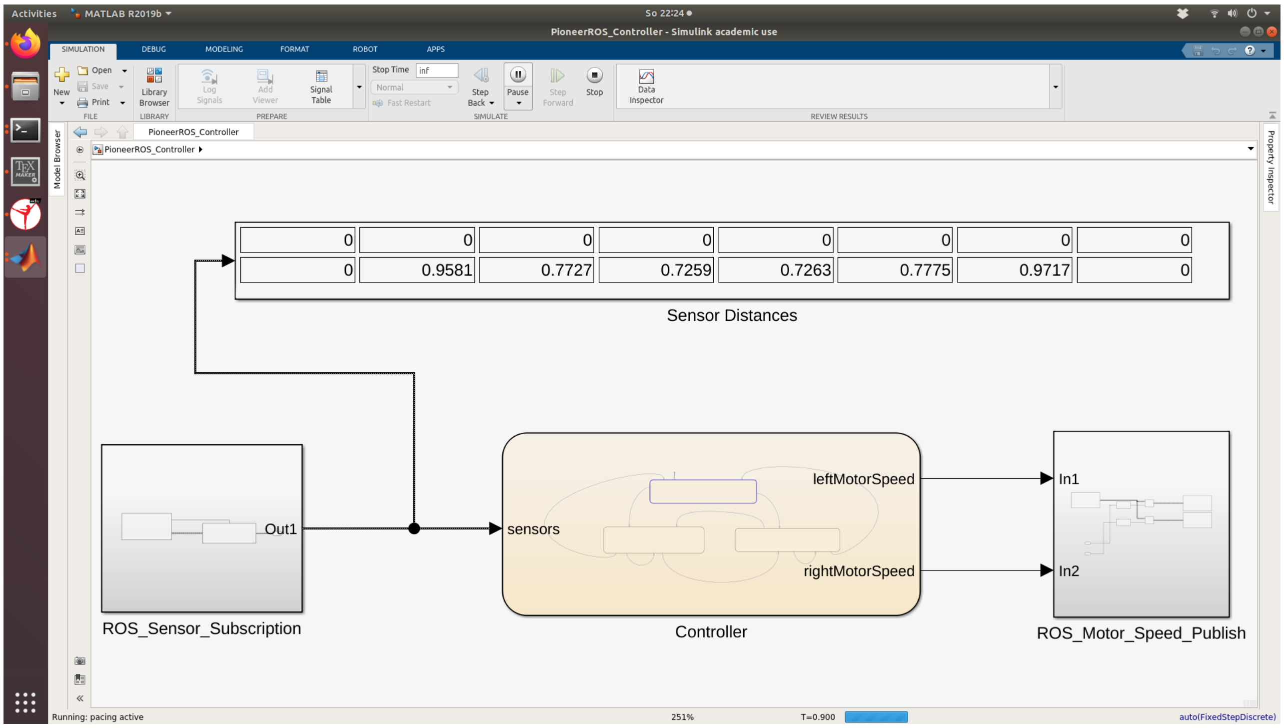
```
Simulation started.
ROS interface was found.
```



Finally startup MATLAB, navigate to the directory of the downloaded scenario, and double click on the file: PioneerROS_Controller.slx. This should start up Simulink with a Template controller model.



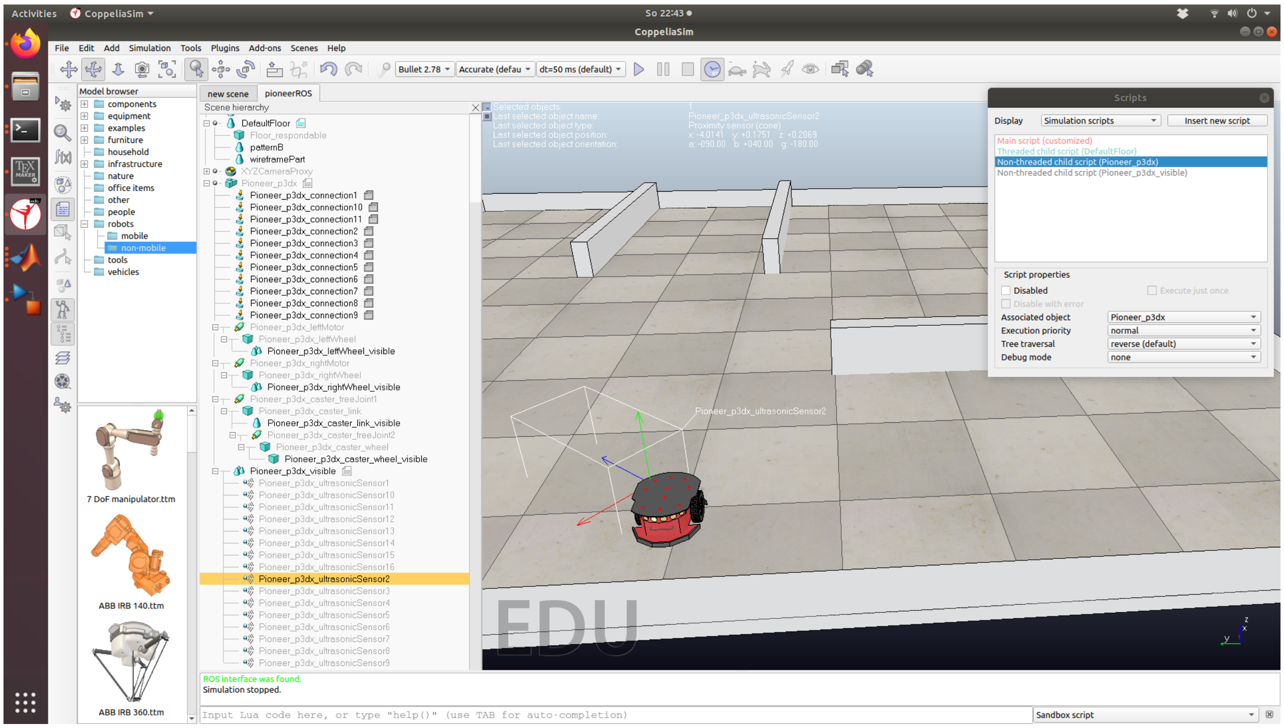
Activities
MATLAB R2019b - academic use
So 22:18
Document : /home/gavin/Dropbox/Vorlesungen/Mobile Robotics/TEX/Practical_Exercise_1_ROS_Pioneer.tex
File Edit Tools LaTeX Math Wizard Bibliography User View Options Help
MATLAB R2019b - academic use
HOME PLOTS APPS
New Script
New Live Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Favorites
Analyze Code
Run and Time
Clear Commands
Simulink
Layout
Preferences
Set Path
Add-Ons
Help
Community
Request Support
Learn MATLAB
FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES
Search Documentation
Gavin
STRUCTURE
home gavin
Current Folder
Name
catkin_ws
CoppeliaSim_Edu_V4_0_0_Ubuntu18_04
Desktop
Documents
Downloads
Dropbox
gavinGit
jetsonHomeBackup
jetson-inference
jetson-upload
Music
Pictures
Public
slprj
snap
Templates
Videos
conc.lic
pioneer_ROS_VREP.zip
pioneerROS.ttt
PioneerROS_Controller.slx
PioneerROS_Controller.slxc
Command Window
New to MATLAB? See resources for Getting Started.
Workspace
Name
Value
PioneerROS_Controller.slx (Simulink Model)
Model version:
1.6
Saved in Simulink version:
R2019b
Last modified by:
gavin
(no description available)
Preview:
Ready
UTF-8 Normal Mode



Starting the Simulink Model, should then show the received sensor data from the Pioneer Robot. Inside the Simulation, the robot should then start to move as the template controller in MATLAB starts sending it motor speed commands.

4 Sensor Data and Motor Commands

The Sensor Data being published from the robot, are the distances from the ultrasound sensors around the robot. They report up to a maximum distance of 1m. If nothing is detected, the sensor will report a value of 1.0. The robot publishes the distances from 16 ultrasound sensors. 8 from the front and 8 from the back. To view the actual direction of the ultrasound sensors, click on one of the descriptors in the scenario tree structure.



In the Simulink Model, four of the front sensors are grouped into a detector for the front left, and four for a detector for the front right. Look in the MATLAB Function block, left of the stateflow block. The output of this block, provides two signals, objectFwdLeft and objectFwdRight that the controller uses to react when the robot encounters an object.

The Stateflow diagram includes two outputs, leftMotorSpeed and rightMotorSpeed. These are the wheel speeds that are published back to the robot to allow the robot to move.

5 Your Programming Task

In this programming task, you are to program the robot to follow the wall of the virtual room around in a clockwise direction. You are to maintain a distance of between 15cm and 50cm from the wall at all times.

You can use a combination of stateflow and direct control theory to achieve this, or just purely stateflow. Think here in terms of the Paradigm lecture. Top level Modi could include as examples:

- Search for Wall
- Following Wall
- Follow Corner
- ...

When in these Modi, you could steer the robot directly in the stateflow, as per the provided template. Or alternately, use the values of select sensors together with a proportional controller to have the robot follow the wall more accurately.

Your final controller should be uploaded to Moodle, prior to the practical next Monday.