Final Project

ELEC6910R & COMP6211C: Robotic Perception and Learning

Prof. LIU Ming

Deadline : 23:59 May 13th 2019
Demo Time : May 14th to 17th 2019
Location : CYT-2013b (RI)
Grouping : **2** in a group; with exception at most 3 persons per group

1 Environment

For this project, you are going to build a SLAM system based on V-Rep and ROS on a Ubuntu system.

1. You are highly suggested to install the Ubuntu 16.04 as dual boot with your Windows or Mac OS. Virtual machine may meet some compatible problems with ROS and V-Rep.
2. For ROS, the version Kinetic is recommended. You can download it from this link http://wiki.ros.org/kinetic/Installation/Ubuntu
3. You can find the V-Rep in the attachment with version 3.5
4. The Lua code on V-Rep has been provided. You only need to implement your ROS package. The V-Rep ROS interface will be compiled by yourself with the provided package.

2 Submission

2.1 Code Section 80%

All necessary code for running your program as well as a brief user guide for the TA to run the programs easily to verify your results, all compressed into a single ZIP or RAR file. Submit this file to canvas before deadline. Late submission will not be accepted.

2.2 Report Section 10%

Write a project report in PDF format to show how you implement this project. You should also indicate how the work is assigned between group members.

2.3 Demo Section 10%

You are going to bring your laptop and do a live demonstration for TA in Robotics Institute. A detailed demo schedule will be posted later for each group.

3 Vrep ROS Interface Instruction

1. add the root path of your vrep to the environment variable with name **VREP_ROOT** in the .bashrc file
2. copy the **vrep_ros_interface** folder we provide to your /catkin_ws/src and compile your catkin work space
3. After successfully build, you will find the **libv_repExtRosInterface.so** in /catkin_ws/devel/lib/libv_repExtRosInterface.so. Copy this file to your vrep root folder.
4. Open a new terminal to run *roscore*, and open another terminal to start vrep. If you meet the problem of launching vrep, try to launch without *sudo* command. Make sure you find the log of launching vrep with the content like this:

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

5. Start the env.ttt in vrep, press the start button, you can use the *rostopic list* command to check the interface between Vrep and ROS

4 Tasks

You have to finish five tasks with given simulation environment and give a live demonstration based on which we grade your final project.

Here are the tasks and their weights in Code section:

1. Build 2D grid map with laserscan data and show it via rviz such as Fig. 1 — 20%
2. Control the mobile robot in the simulation environment with keyboard (drive it to move) — 5%
3. Image Recognition and localization. There are five images of different people in the environment and you have to
   (a) judge whether the target images occurred in current vision data(we provided)
   (b) if yes, estimate the location of target images
   (c) add *markers* to the map in rivz which stands for the target images position

   40%
4. Visual Servoing. There is a slowly moving yellow (rgb:255,255,0) ball (Fig. 5) in the environment and you have to write program (rosnode, in c/c++ or python) to control the mobile robot to follow the ball. — 30%
5. Write a launch file to *roslaunch* all of above programs at once. — 5%

NOTICES: you can use existing packages for task 1 & 2. You are encouraged to write your own code for other tasks.

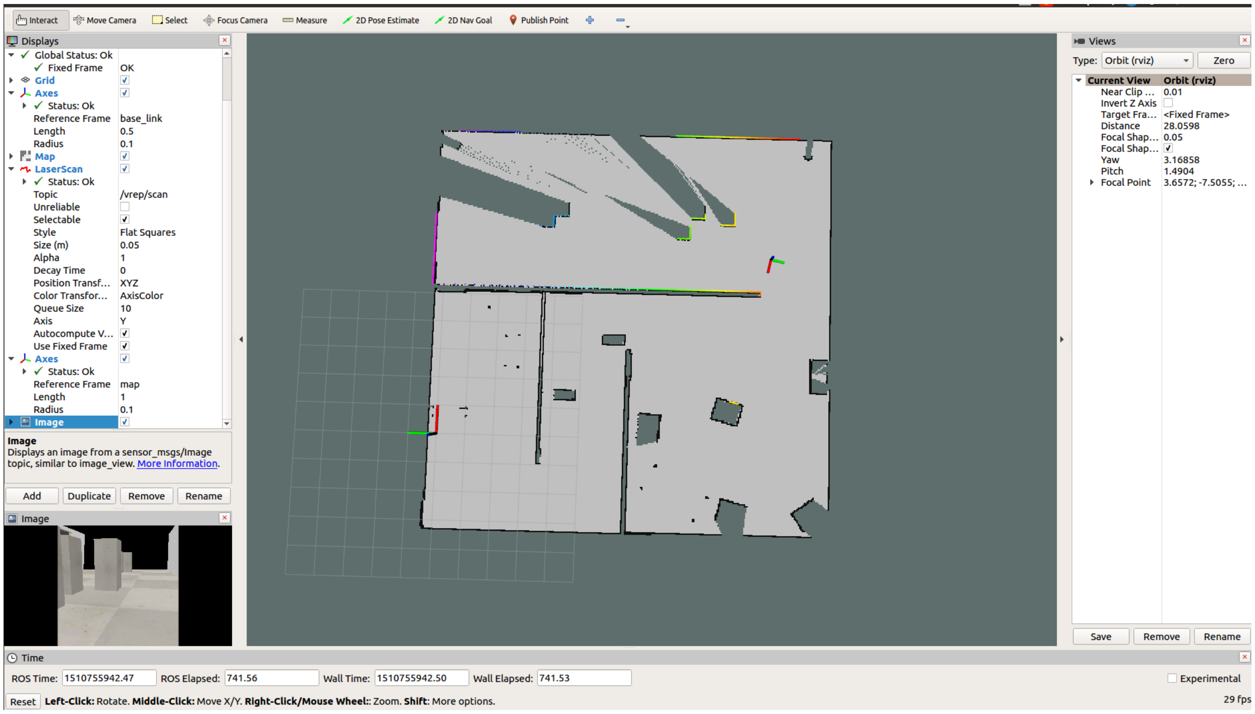


Figure 1: Grid map in Rviz

5 Prerequisite

To finish the final project, you are supposed to

1. Have basic knowledge of Linux (e.g. Ubuntu)
2. Finish the beginner level tutorials[1] of ROS and know how to use existing packages
3. Use V-REP[2] to perform simulation
4. Be able to integrate ROS and V-REP
5. Be familiar with C++ or Python

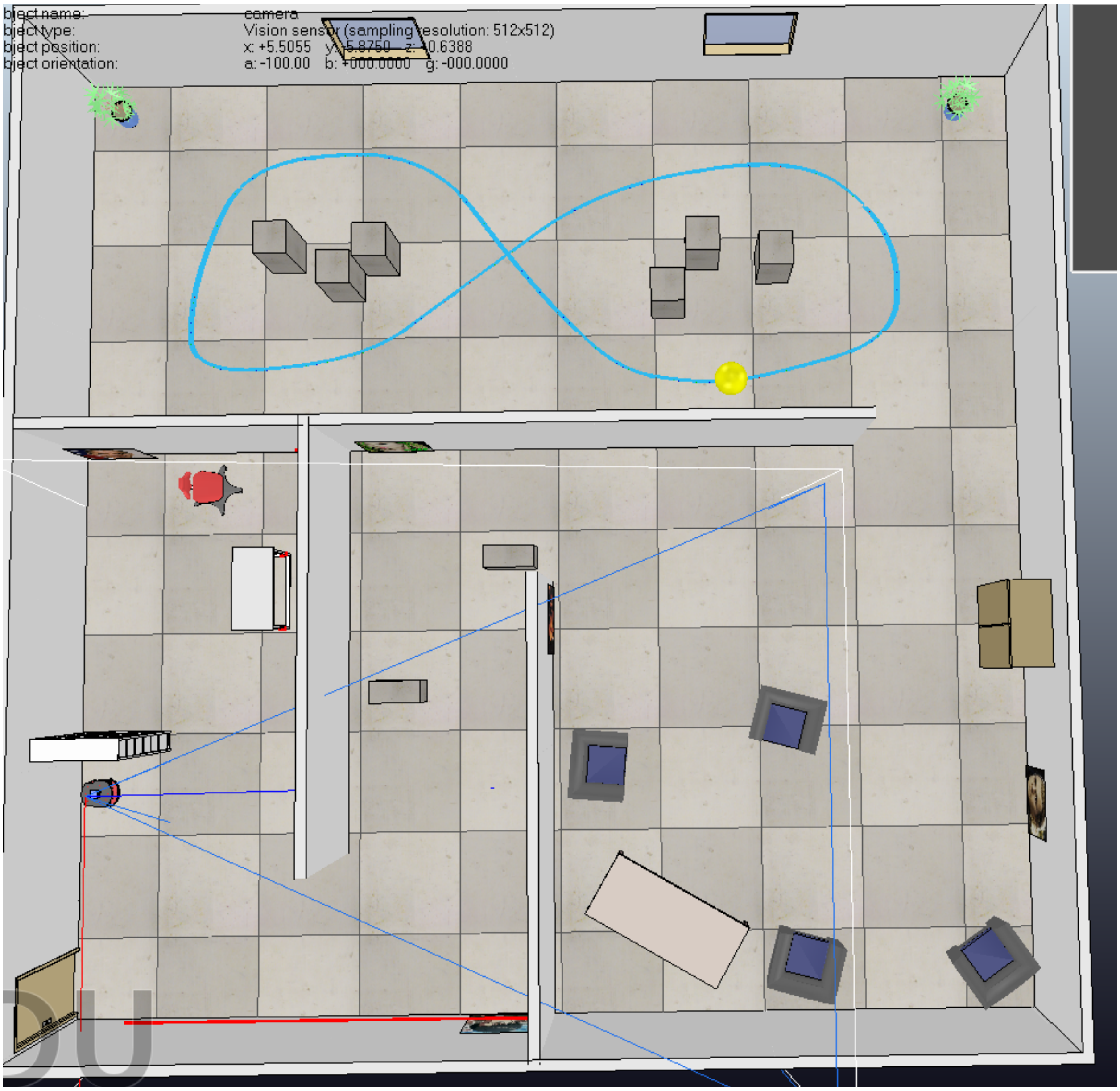


Figure 2: Overview of Simulation Environment

6 Simulation Environment

A V-REP scene file, named *env.ttt*, is given with this document. The Fig. 2 shows the simulatin environment in which the blue path of yellow ball is **invisible** for the vision camera on mobile robot. The

[1]http://wiki.ros.org/ROS/Tutorials
[2]http://www.coppeliarobotics.com/

simulation environment already includes tested scripts which implements all functions. Hence, it is **not at all** necessary to alter the simulation scene. You only need to click the start button and then work with ROS nodes and topics.

In the simulation script we publish/subscribe several topics:

1. - Name: /vrep/image
   - Type: sensor_msgs/Image
   - Pub/Sub: Publish
   - Comment: The image captured by vision sensor on mobile robot. Refer section 6.1
2. - Name: /vrep/scan
   - Type: sensor_msgs/LaserScan
   - Pub/Sub: Publish
   - Comment: The laser scan data
3. - Name: /vrep/cmd_vel
   - Type: geometry_msgs/Twist
   - Pub/Sub: Subscribe
   - Comment: The velocity command to mobile robot
4. - Name: /vrep/laser_switch
   - Type: std_msgs/Bool
   - Pub/Sub: Subscribe
   - Comment: Enable/Disable the /vrep/scan, disable it can speed up the simulation, you only need publish this topic once. Default: Enable
5. - Name: /vrep/camera_switch

- Type: std_msgs/Bool
- Pub/Sub: Subscribe
- Comment: Enable/Disable the /vrep/image, disable it can speed up the simulation, you only need publish this topic once. Default: Enable

And we also publish two transform by ros tf. If you don't know tf or tf2 of ros please refer to the (tutorial). We define the frame_id of mobile robot, laser, vision sensor are *base_link*, *laser_link*, *camera_link* respectively. By listening to the tf you can get the transformation of these objects, which is critical for task 2&3.

6.1 Image Problem

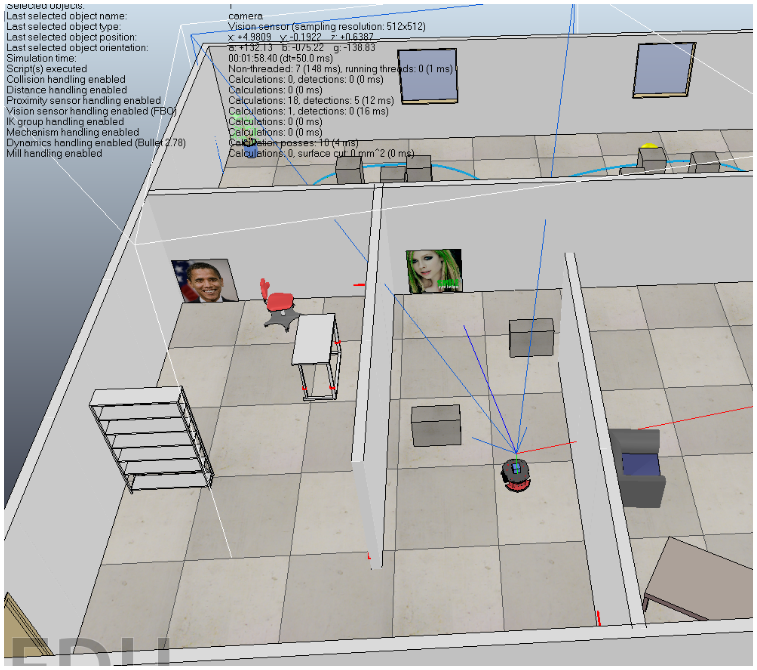
There is one problem of V-REP vision sensor which you have to solve. The image you get at ros node side is left-right reversed, as shown in Fig. 3. We mount the camera with conventional coordinate, x-axis(right), y-axis(down), z-axis(forward). In Fig. 3a we see the picture of Avril Lavigne is at left side but it's at right side of Fig. 3b.

It's easy to solve this problem by OpenCV *CV::flip* function[3]. Hence, in your node implementation, please call the cv::flip to reverse the image and then do later processing.

6.2 Prior Knowledge & Hints

1. The size of all five target images in simulation environment is 1x1 (meter).

[3] Link CV::flip



(a) V-REP side

(b) ROS side

Figure 3: Left-right reversed problem

2. We choose the perspective-type vision sensor for simulation and

the model is shown in Fig. 4 and more detail at link[4]. The parameters are listed in Table. 1.

Table 1: Vision Sensor Parameters

Near clipping plane (m)	0.01
Far clipping plane (m)	10
Perspective angle(degree)	45
Resolution X (pixel)	512
Resolution Y (pixel)	512

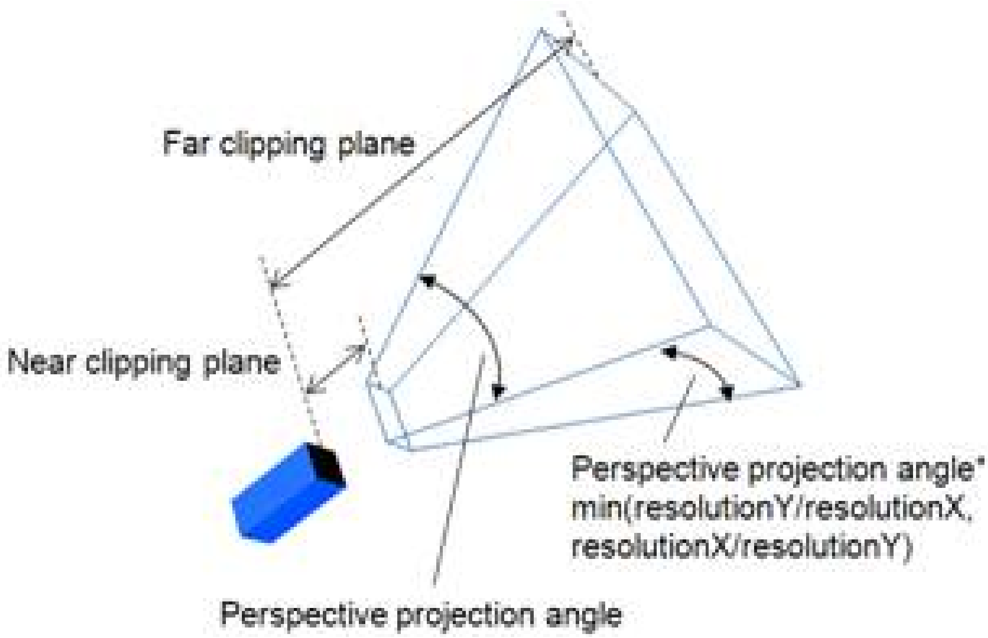


Figure 4: Vison sensor model of V-REP

3. The color of ball is RGB:#FFFF00 which could be used to simplify the tracking task. Don't forget to reverse the image.
4. Refer rviz/DisplayTypes/Marker[5] and learn how to show marker in rviz.
5. Hint: publish data to */vrep/laser_switch* to disable the laser to speed up the simulation when switch to auto-tracking mode.

[4] http://www.coppeliarobotics.com/helpFiles/en/visionSensorPropertiesDialog.htm
[5] http://wiki.ros.org/rviz/DisplayTypes/Marker

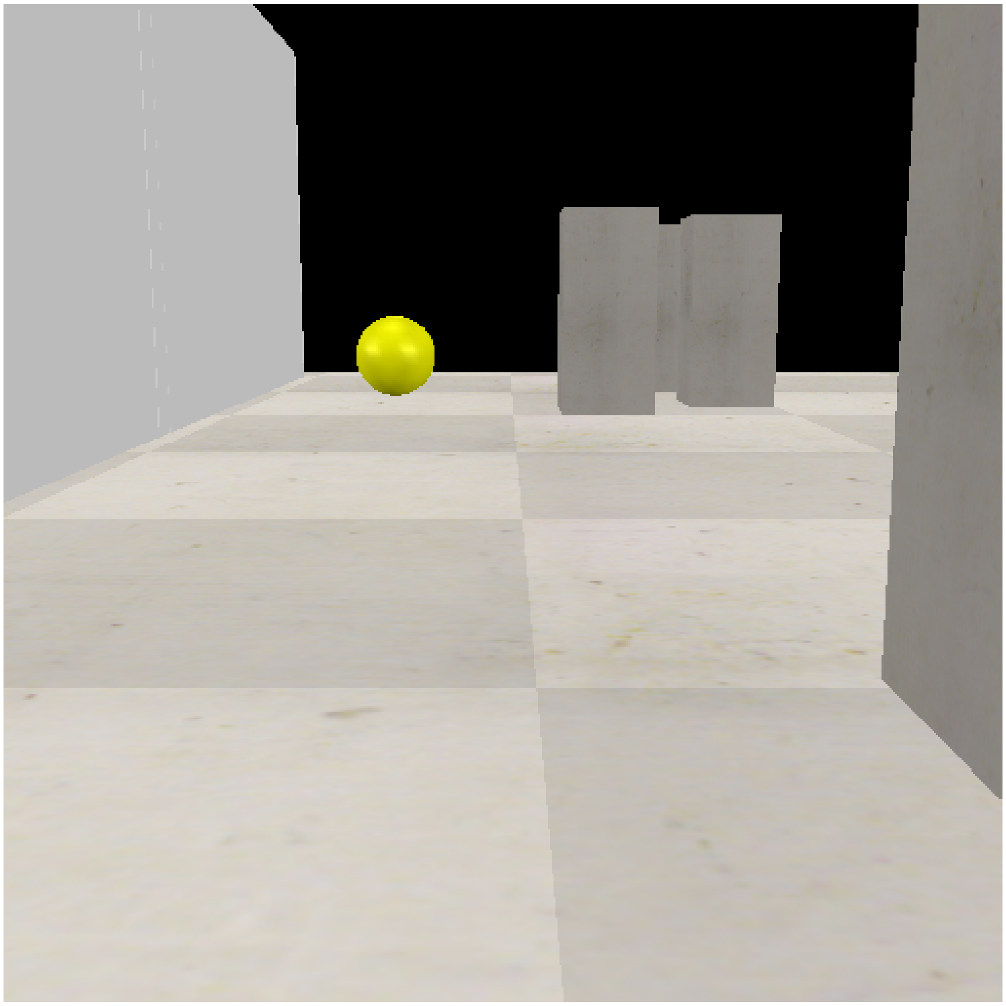


Figure 5: Yellow ball to follow

7 Evaluation

The evaluation is based on the all tasks.

The demonstration process :

1. Use your **launch file** launch ros and ros nodes (including rviz), start V-REP simulation
2. control the mobile robot move in the environment by **keyboard**
3. at same time, the grid map should be shown in rviz. Fig.1

4. at same time, if the target images occurs in the filed of view of vision sensor, its location should be estimated and **marker** should be shown in rviz
5. move robot to upper room and switch to auto-tracking mode. The robot should **automatically track and follow** the yellow ball without keyboard control

8 Possible Add-up points

- Besides detection, you can also recognize the faces on the wall.
- Automatic exploration of the environment.
- Controller performance evaluation for the tracking problem.
- Any further contributions not included in the standard tasks and specifications.

9 Notices

1. NO PLAGIARISM. It's much easier to analyse code than words.
2. Late demonstration is NOT accepted.