Final Project

ELEC4010K: Machine Learning and Information Processing for Robotic Perception

Prof. LIU Ming

November 17, 2017

1 Overview

Deadline : 5:30pm 12 Dec 2017
Location : CYT-2011 (RI)
Grouping : **2** in a group; with exception at most 3 persons per group

Please **make an appointment** before the deadline by email and then bring your computer to office to give a live demonstration. Please try the best to collaborate with your partners. In the demonstration, please notify the TA who has done which part of work.
Further contact: qinghai.liao@connect.ust.hk

2 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:

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.

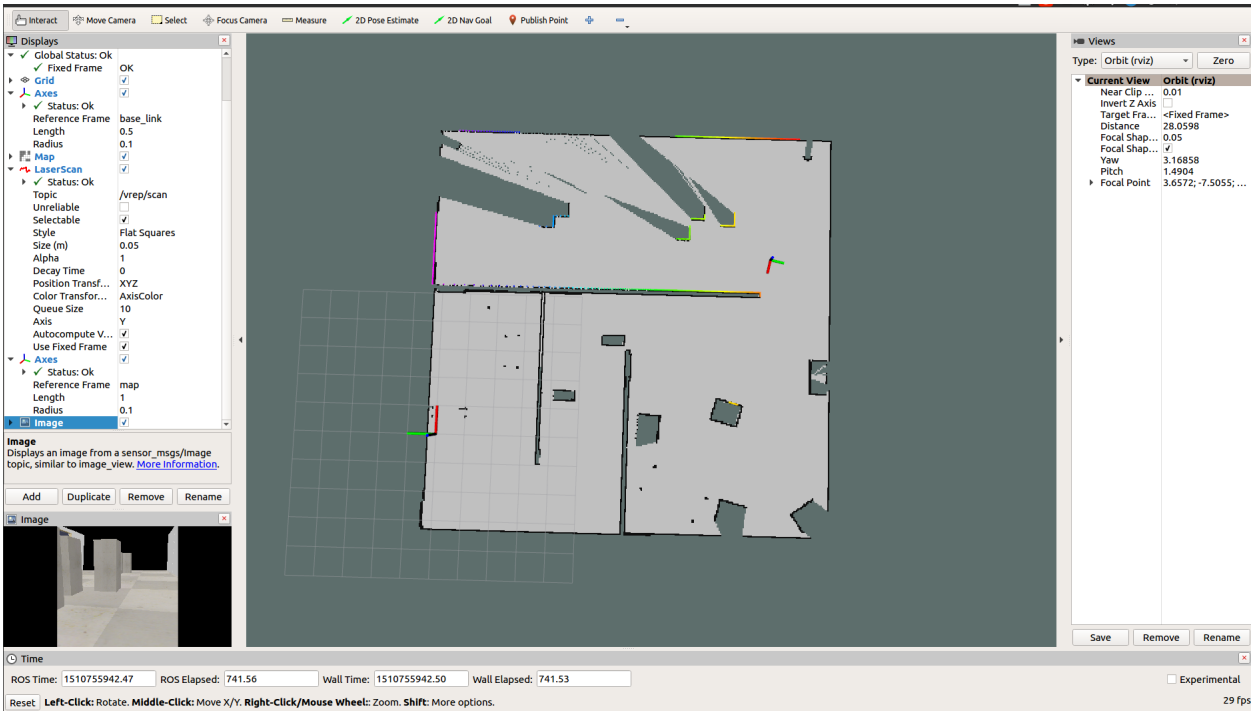


Figure 1: Grid map in Rviz

3 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[3]
5. Be familiar with C++ or Python
6. Master the content of first lab session

4 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 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.

As introduced in the first lab session, in the simulation script we publish/subscribe several topics:

1. • Name: /vrep/image
   • Type: sensor_msgs/Image
   • Pub/Sub: Publish

[1] http://wiki.ros.org/ROS/Tutorials
[2] http://www.coppeliarobotics.com/
[3] Refer the first lab session material

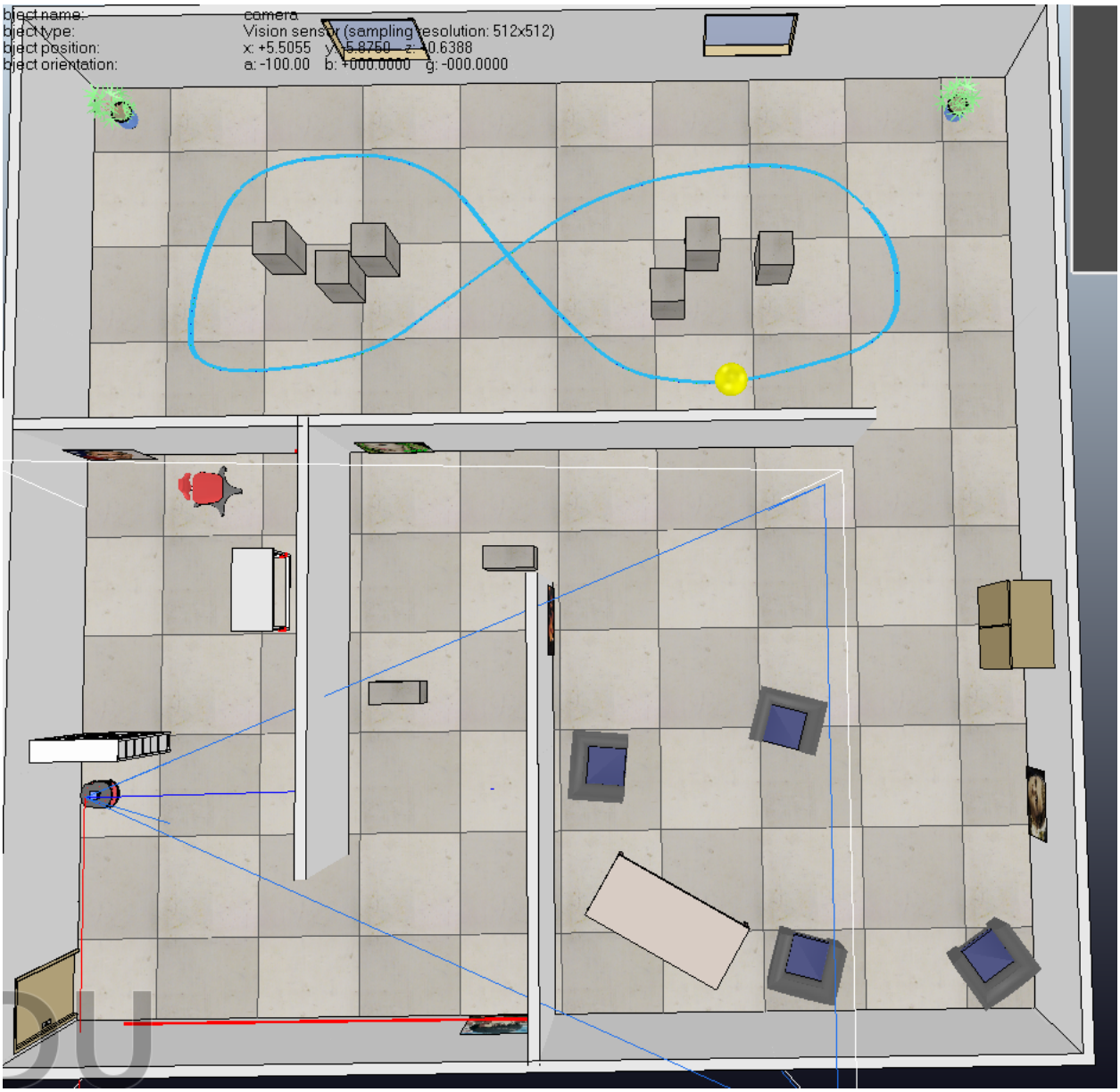


Figure 2: Overview of Simulation Environment

- Comment: The image captured by vision sensor on mobile robot. Refer section 4.1

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.

4.1 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[4]. Hence, in your node implementation, please call the cv::flip to reverse the image and then do later processing.

4.2 Prior Knowledge & Hints

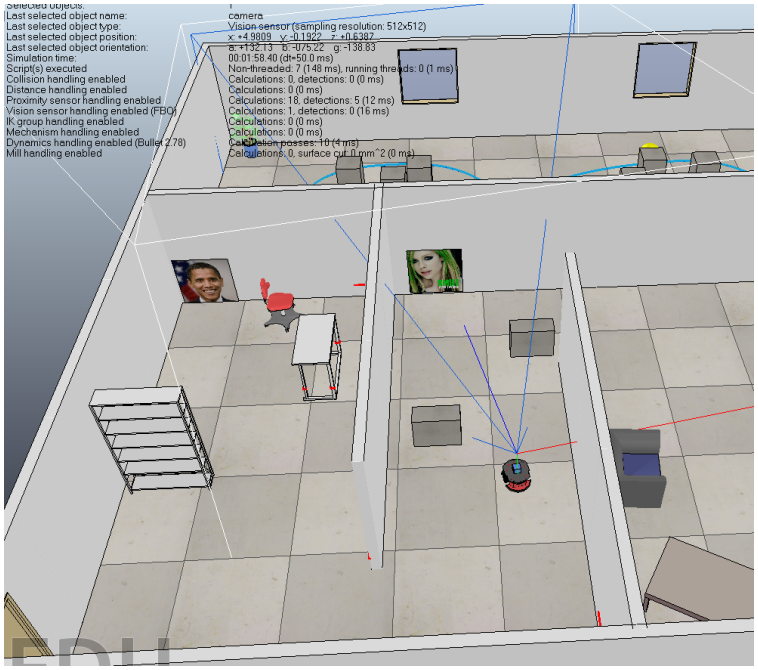
1. The size of all five target images in simulation environment is 1x1 (meter).
2. We choose the perspective-type vision sensor for simulation and the model is shown in Fig. 4 and more detail at link[5]. The parameters are listed in Table. 1.

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

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



(a) V-REP side

(b) ROS side

Figure 3: Left-right reversed problem

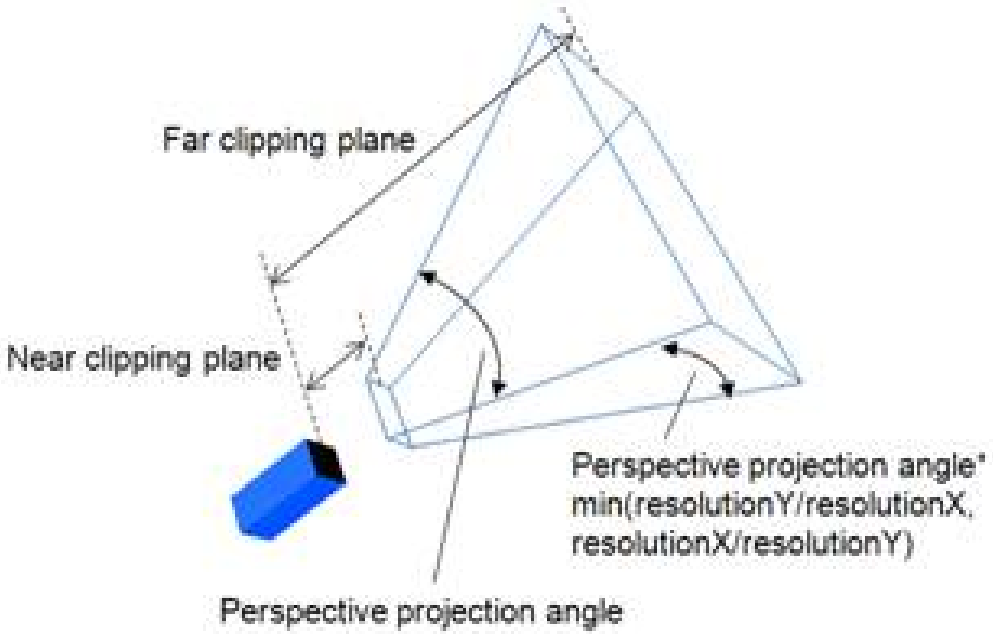


Figure 4: Vison sensor model of V-REP

5 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 iamges 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

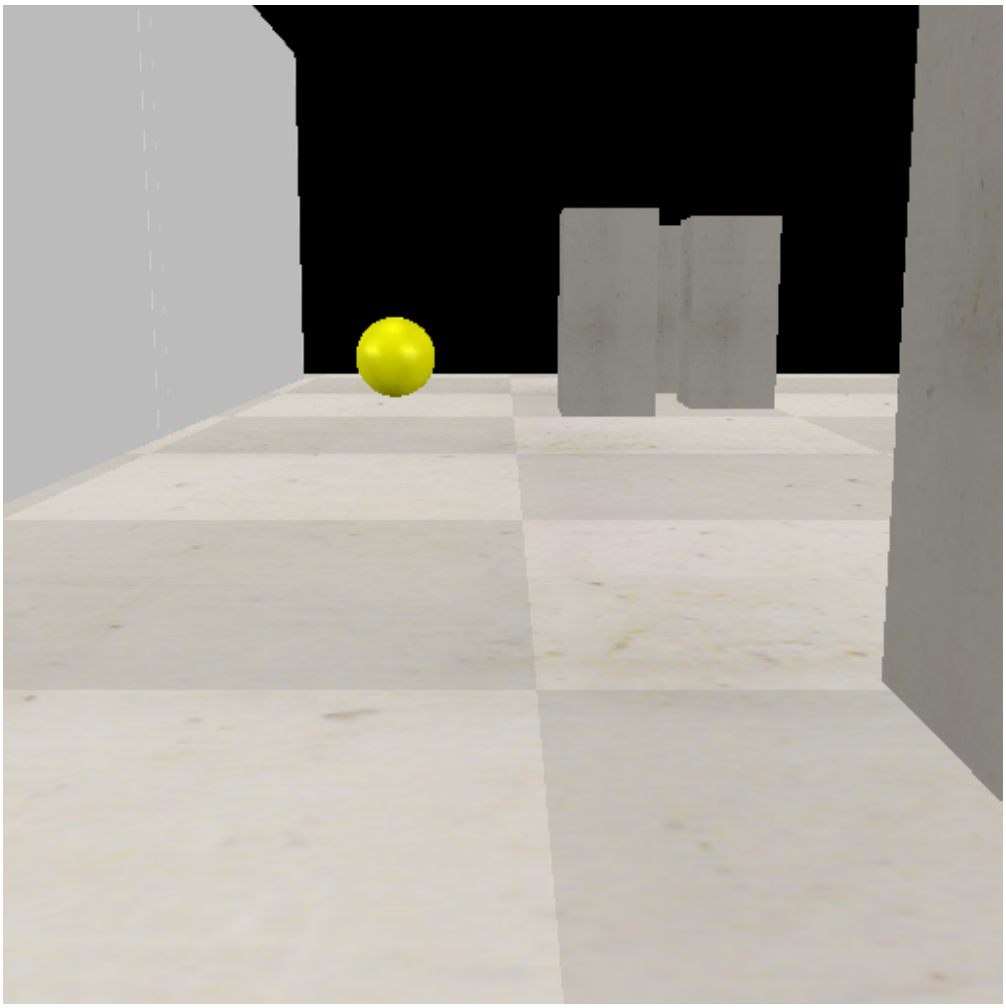


Figure 5: Yellow ball to follow

6 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.

7 Notices

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