ROAS 6000A Final Project

Prof. Ming LIU

1 Overview

Deadline of Presentation: 16 May 2023
Deadline of Report: 23 May 2023
Grouping: Groups of 2-3 are allowed.

Please make an appointment with the me before the deadline by email and then bring your computer to office to give a live demonstration or you can complete the demonstration via Zoom.

2 Tasks

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

Here are the tasks and their weights:

1 Control the mobile robot in the simulation environment with keyboard (drive it to move) 5%

2 Build 2D grid map with laserscan data and show it via rviz such as Fig. 1 20%

3 The room is devided into serveral areas, as Fig 2 shows, let the robot judge which area it locates. 20%

4 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 rviz which stands for the target images position

   30%

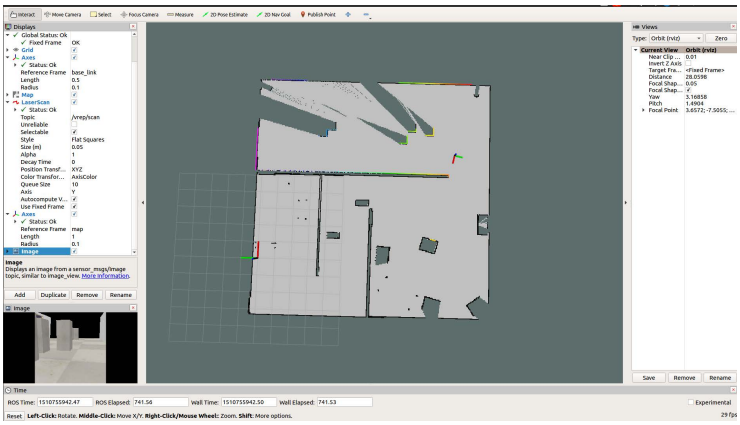


Figure 1: Grid map in Rviz

5 Visual Servoing. There is a slowly moving yellow (rgb:255,255,0) ball (Fig. 6) in the environment and you have to write program (rosnode, in c/c++ or python) to control the mobile robot to follow the ball. 20%

6 Write a launch file to roslaunch all of above programs at once. 5%

NOTICES: **you can use existing packages**, but you are encouraged to write your own codes.

3 Prerequisite

To get started in 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. Be familiar with C++ or Python.
4. Be familiar with codes with ROS.

Some tools/libraries that we may need to know to learn to complete the tasks:

- Rviz
- ROS Navigation Stack
- OpenCV

[1] http://wiki.ros.org/ROS/Tutorials

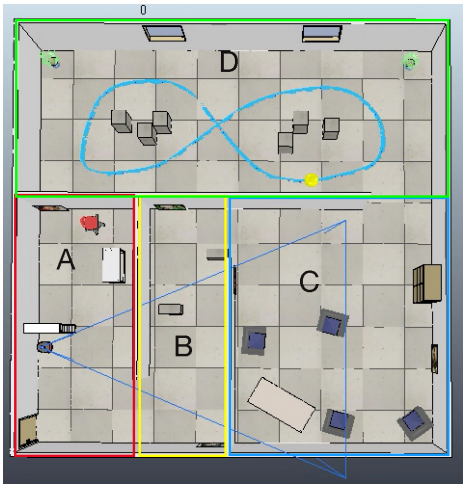


Figure 2: divided areas

4 Simulation Environment

A V-REP scene file, named env.ttt, is given with this document. The Fig. 3 shows the simulation 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
- Comment: The image captured by vision sensor on mobile robot. Refer section

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

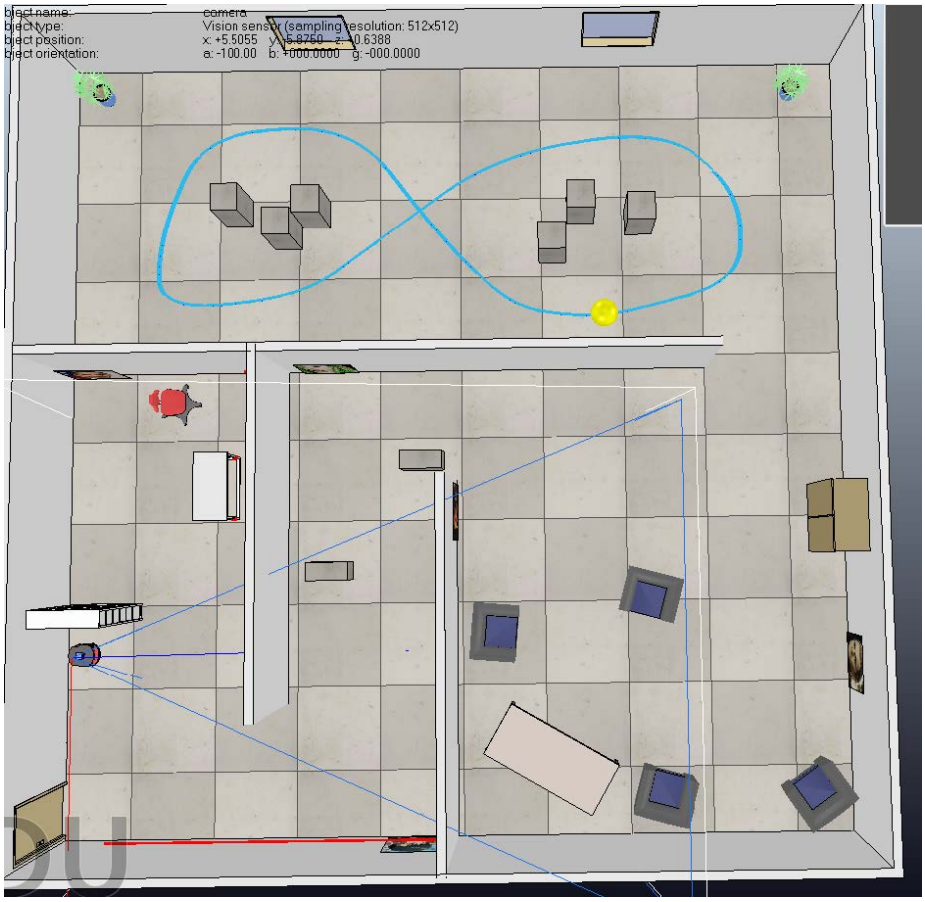


Figure 3: Overview of Simulation Environment

- 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 transforms 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.

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. 4. We mount the camera with conventional coordi- nate, x-axis(right), y-axis(down), z-axis(forward). In Fig. 4(a) we see 5 the picture of Avril Lavigne is at left side but it's at right side of Fig.4(b).

It's easy to solve this problem by OpenCV CV::flip function [2]. Hence, in your node implementation, please call the cv::flip to reverse the image and then do later processing.

5 Prior Knowledges & Hints

1 Vrep:

(a) You don't need to modify the environment in V-Rep to complete the task, but you are encouraged to explore it.

(b) You can pick up the robot and place it to anywhere you like, so you don't need to drive manually across the room.

2 Vrep to ROS:

(a) The tf tree published by the simulation environment does not conform well with the ROS libraries. But it is completely fine to finish the project without dealing with the tf tree (it takes you some time to learn it well).

[2] Link CV:filp

(b) Publish data to /vrep/laser_switch or /vrep/camera_switch to disable the laser/camera to speed up the simulation when sometimes you don't need the laser data or camera data.

(c) The simulation environment will set the ROS master to use simulation time, while also **synchronize** the timer of ROS with the process of the simulation. In codes, time measured from ROS APIs and that from System APIs will be **different**.

3 Simulation environments:

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

(b) We choose the perspective-type vision sensor for simulation and the model is shown in Fig. and more detail at link [3]. 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

(c) The color of ball is RGB:#FFFF00 which could be used to simplify the tracking task.

4 ROS and Codes:

(a) It is totally fine to use existing ROS packages.

(b) Refer to CvBridge to transfer between ROS image messages to OpenCV objects.

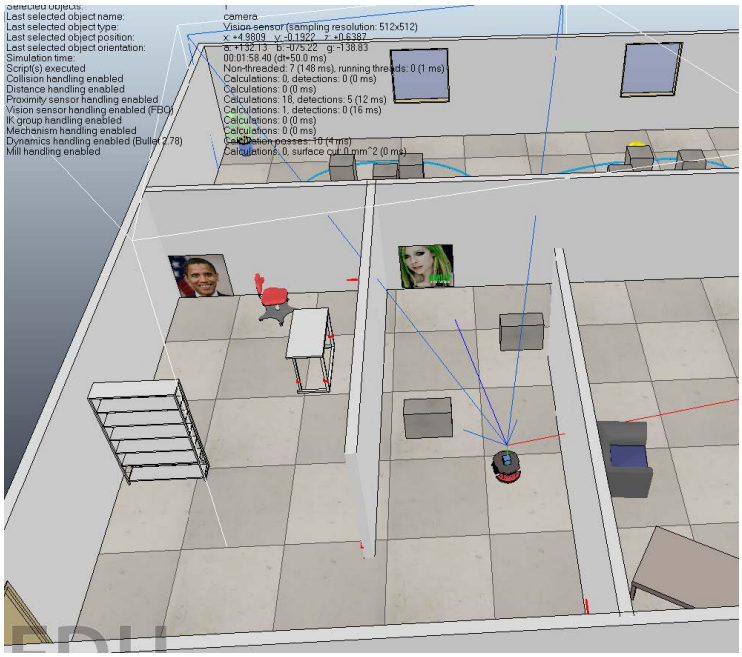
(c) Refer to rviz/DisplayTypes/Marker [4] and learn how to show marker in rviz.

(d) Record your drive with ROS Bag. Then replaying it and fine-tune your perception algorithms. So you don't need to run the simulation **and** heavy computation at the same time.

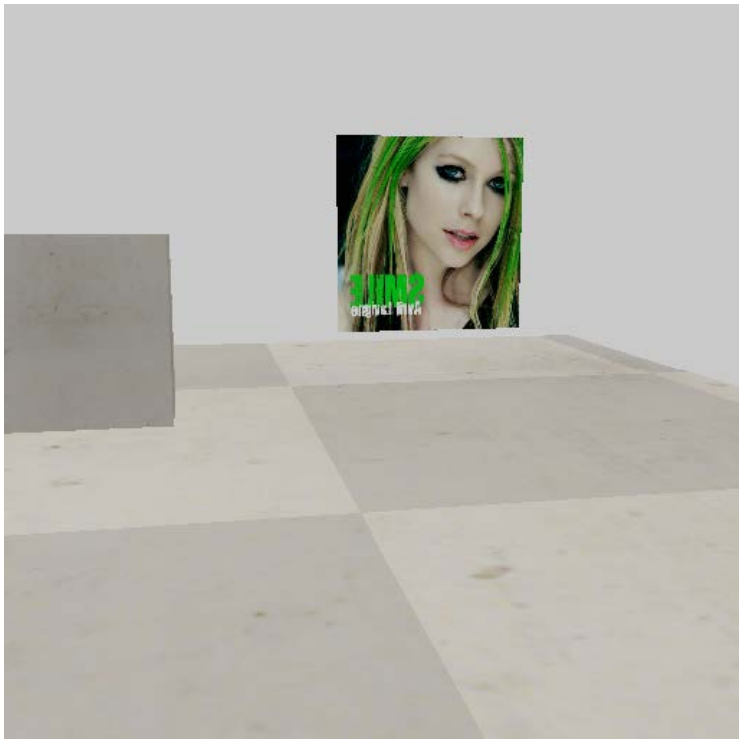
6 Evaluation

The evaluation is based on the all tasks.
The demonstration process :

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



(a) V-REP side



(b) ROS side

Figure 4: Two Subfigures

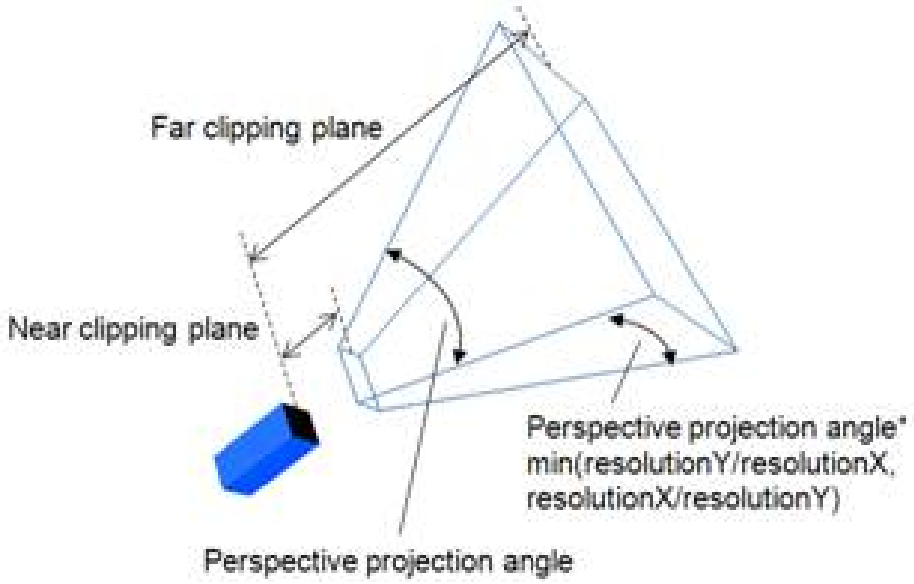


Figure 5: Vison sensor model of V-REP

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 field 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

7 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.

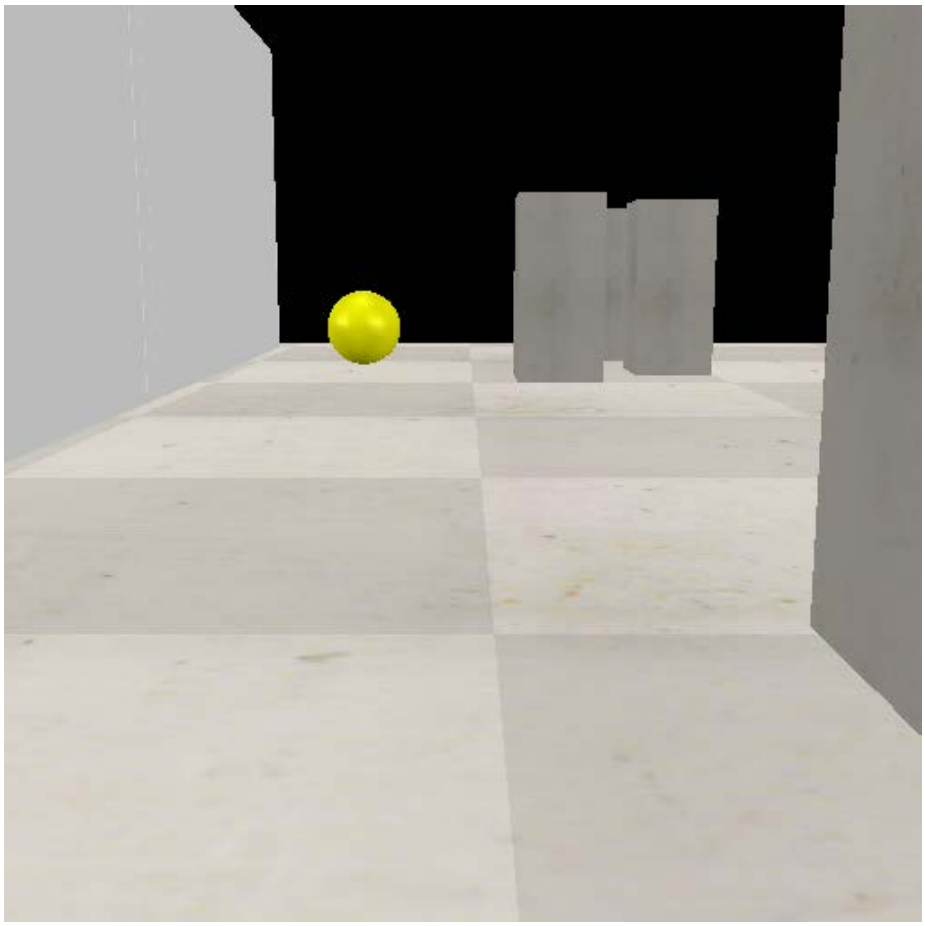


Figure 6: Yellow ball to follow

You can also setup your own research environment, such as exploration with human existence, multi robots collaboration, path planning, etc. Innovation is encouraged. The points will be counted according to the complexity and novelty of your work.

8 Notices

1 NO PLAGIARISM. It's much easier to analyse code than words.

2 Late demonstration is NOT accepted. The penalty policy is 10% points off per day.