RobotLab 4: SLAM Part I

ENGN 4627/6627 Robotics 2016 S2

September 22, 2018

1 Purpose

This lab allows you to prepare your turtlebot to perform a full filtering based SLAM algorithm.

The aim of this lab is to get you familiar with:

- Sensor Calibration
- Data Acquisition
- State estimation using filtering approach

This lab consists of three tasks; sensor calibration, generating and plotting acquired data and getting familiar with different filters and how they could be used in a SLAM framework.

The full mark for Lab 4 is 40 points, divided as follows: 20 points for Lab Report, and 20 points for project.

2 Preparation

1. You should complete Lab 3, and be familiar with the cylinder detection package that was provided to you `https://github.com/crodriguezo/cylinderDetector`.
2. You are expected to have some basic knowledge about ROS bags, how to generate and replay them. This was covered in tutorial 1. The two most basic commands are:
   $rosbag record -a # record everything,
   # press Ctrl+C to stop
   $rosbag play <your bagfile> # replay
   For more detailed documentation, you can check `http://wiki.ros.org/rosbag/Commandline`

3. The ipython notebooks provided can be run using ipython notebook, that you can download $sudo apt-get install python-notebook, or you can run online on `http://jupyter.org/` just adding the line "%matplotlib notebook" to enable animation on figures.

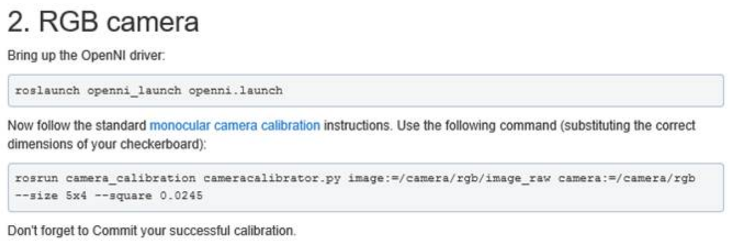
4. Help scripts are provided to you on Wattle: *Lab4_for_students.zip* file. Rosbags are available for you to download at `http://users.cecs.anu.edu.au/~u5419700/sample1.bag`, `http://users.cecs.anu.edu.au/~u5419700/sample2.bag` and `http://users.cecs.anu.edu.au/~u5419700/sample3.bag` These will be of great importance to help you complete the tasks of this lab.

3 Task 0: Sensor Calibration

You are required to perform sensor calibration. This means that you need to calibrate for both geometry and color segmentation. Geometry Calibration involves calibrating for both the RGB and IR Camera.

Color segmentation could be run using the 1 line command
$rosrun cylinder colorSegmentation

Geometry calibration is done as shown in Fig. 1 for RGB Camera calibration and Fig. 2 for IR Camera calibration.



2. RGB camera

Bring up the OpenNI driver:

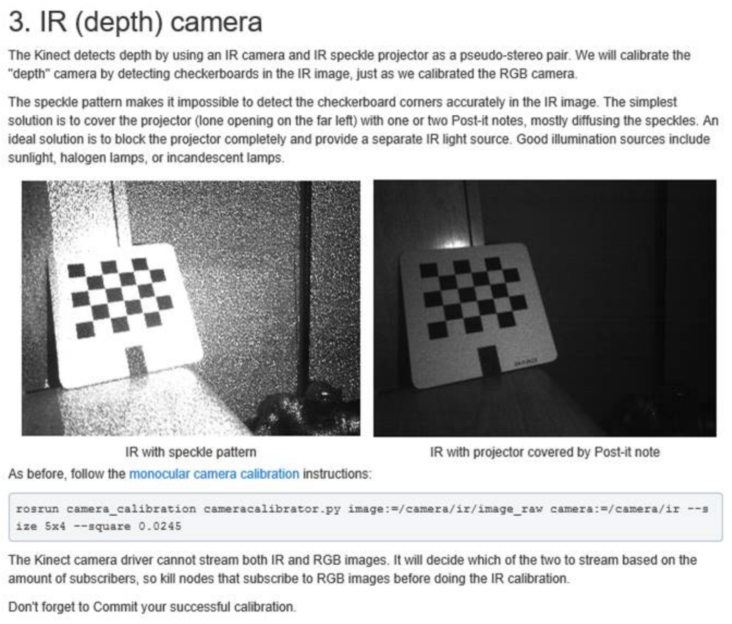
```
roslaunch openni_launch openni.launch
```

Now follow the standard monocular camera calibration instructions. Use the following command (substituting the correct dimensions of your checkerboard):

```
rosrun camera_calibration cameracalibrator.py image:=/camera/rgb/image_raw camera:=/camera/rgb
--size 5x4 --square 0.0245
```

Don't forget to Commit your successful calibration.

Figure 1: RGB Camera Calibration



3. IR (depth) camera

The Kinect detects depth by using an IR camera and IR speckle projector as a pseudo-stereo pair. We will calibrate the "depth" camera by detecting checkerboards in the IR image, just as we calibrated the RGB camera.

The speckle pattern makes it impossible to detect the checkerboard corners accurately in the IR image. The simplest solution is to cover the projector (lone opening on the far left) with one or two Post-it notes, mostly diffusing the speckles. An ideal solution is to block the projector completely and provide a separate IR light source. Good illumination sources include sunlight, halogen lamps, or incandescent lamps.

As before, follow the monocular camera calibration instructions:

```
rosrun camera_calibration cameracalibrator.py image:=/camera/ir/image_raw camera:=/camera/ir --s
ize 5x4 --square 0.0245
```

The Kinect camera driver cannot stream both IR and RGB images. It will decide which of the two to stream based on the amount of subscribers, so kill nodes that subscribe to RGB images before doing the IR calibration.

Don't forget to Commit your successful calibration.

Figure 2: IR Camera Calibration

Both geometry calibration and color segmentation were covered in tutorial 4 by Cristian, you can review to it on Echo: `https://wattlecourses.anu.edu.au/course/view.php?id=17901`

For more details on geometry calibration, you are advised to follow the wiki ROS tutorial on `http://wiki.ros.org/openni_launch/Tutorials/IntrinsicCalibration`

P.S: NO GRADES ARE ASSIGNED TO THIS TASK, HOWEVER THIS IS CONSIDERED A MAIN TASK FOR YOUR SLAM FINAL ALGORITHM AND PROJECT TO WORK PROPERLY.

4 Task 1: Data Acquisition and Plotting

Using the ROS bag provided, you are required to generate two graph files; one "Trajectory and Points file" containing Robot 2D Poses and Landmark 2D Positions, and the second "Measurements file" containing Odometry Measurements and Landmark Measurements each on a different line starting with the correct label ($"POSE2D", "POINT2D", "ODOMETRY\_MEAS2D", "LANDMARK\_MEAS2D"$) as explained in the tutorial and use the measurement file along with the help functions provided to plot the robot path and landmark positions.

PLEASE NOTE:

- POSE2D and POINT2D are outputs of the Relative2AbsolutePose and Relative2AbsoluteXY functions respectively.
- **HINT:** You will need to think of a way to associate which landmark measurements were acquired from which robot pose.

5 Task 2: State Estimation using Filtering

5.1 Gaussian Distribution

You are required to run the ipython notebook for the 1D and 2D Gaussian Distribution changing the mean and variance (squared standard deviation) and report results in the form of plots and comments about each plot. Try at least *three different values* for the mean and variance in each case (1D and 2D).
Work individually!

5.2 Kalman Filter

You are required to run the ipython notebook for the 1D and 2D Kalman Filter changing each of the following parameters:

- initial robot position
- initial uncertainty
- measurement error
- motion error

and reporting results in the form of plots and comments about each plot. Try at least *three different values* for each of the parameters while keeping the others constant to be able to get an understanding of the effect of each on the system.
Work individually!

5.3 Particle Filter

You are required to run the ipython notebook for the Particle Filter (Robot Simulator) changing each of the following parameters:

- forward noise
- turn noise
- measurement noise
- number of particles

and reporting results in form of plots and comments about each plot Try at least *three different values* for each of the parameters while keeping the others constant to be able to get an understanding of the effect of each on the system.
Work individually!

Marking:

MARKING WILL BE ANNOUNCED IN THE NEXT LAB AS BOTH LABS CONTRIBUTE THE TWO PARTS OF SLAM

6 What to Hand In (due in Week 13 Sunday Midnight 12:00 AM)

Pack your source code, your graph file and your Lab report (A4, no page limit) in a single Zip file. Name your zip file in the following format:
" **⟨your_last_name⟩_⟨your_uni_id⟩_Lab_⟨ X ⟩.zip** ".

Upload your single zip file to Wattle, at the corresponding Lab upload link. Since this lab is done by your group, the source code you hand in will be very similar if not identical.

However, your Lab Report must be your own **individual work**, which should not be similar to your group mates. Your overall Lab marks will be based on the quality of source code, graph files, and the quality of your Lab Report.

The first half of the Lab Report will be based around answering some technical questions. These do not have to be formal, but are compulsory to test your understanding of the work done in this lab.

DUE DATE:

Lab 4 (Project) demonstration will be due in Week 12's Lab sessions, giving the **full 3 hours** for demo assessment.

The lab report (see next page for part I) will due on Sunday evening of the following week (Week 13).

7 Turtlebot Lab 4 (Part I) Report Problems

Note: Lab Report must be your individual work.

MORE DETAILS OF THE FINAL LAB/PROJECT REPORT AND GRADING WILL BE ANNOUNCED IN THE NEXT LAB INSTRUCTIONS AS BOTH CONTRIBUTE THE TWO PARTS OF SLAM

Question 1: **Theory**
Briefly explain the following terms, each in no more than five lines of text. You may include mathematic expressions in your answers if you find that is easier.

1. Bayesian formula
2. Total probability theorem
3. Markov assumption
4. SLAM
5. Multivariate distributions

Question 2: **Gaussian Motion Model**

1. What is the distribution of the prior in both of the cases below:

   a) The initial estimate is a 1D Gaussian with $\mu_p = 2$ and $\sigma_p = 4$ and we apply a motion with $\mu_m = 1$ and $\sigma_m = 2$.

   b) The initial estimate is a 2D Gaussian with $\mu_p = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ and $\Sigma_p = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$ and we apply a motion with $\mu_m = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\Sigma_m = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$.

2. Given two 1-dimensional Gaussian distributed random variables X $\sim N(u_1, s_1)$, Y $\sim N(u_2, s_2)$.

   Please derive the probability density function of their weighted sum Z, where $Z = w_1 * X + w_2 * Y$, and the two weights w_1 and w_2 are two fixed and given scalars. Please give detailed steps of your derivation.