# RobotLab 4: SLAM Part II

ENGN 4627/6627 Robotics 2016 S2

September 22, 2018

## 1 Purpose

This lab allows you to implement a Extended Kalman Filter SLAM algorithm on the turtlebot. The sensors of use are: Kinect, odometry, IMU (optional).

The aim of this lab is to integrate:

- Online data acquisition
- State estimation using filtering approach

in a SLAM framework and be able to perform an estimation of the robot trajectory and map of the environment and to evaluate it using a ground-truth map of the environment.

This instruction consists of one task; integrating acquired data into a Kalman filter approach to estimate the robot poses and landmark positions.

Doing this lab demo automatically entitles you to qualify for a competition **with prizes** unless you deliberately choose not to go for the competition.

The full mark for Lab 4 is 30% of the total mark, and will be distributed as follows: **15 points for Lab Report**, and **20 points for project DEMO**.

## 2 Preparation

1. You should be familiar with the cylinder detection package that was provided to you `https://github.com/crodriguezo/cylinderDetector`.
2. You should complete lab 3 and know how to use measurements from onboard sensors.
3. You should have knowledge of Kalman filters and how they work.
4. You should have finished the first part of the Lab 4.

5. Help scripts are provided to you on Wattle: *Lab4_part2_for_students.zip* file.

   ROS bags are available for you to download at at `http://users.cecs.anu.edu.au/~u5419700/sample4.bag`, `http://users.cecs.anu.edu.au/~u5419700/sample5.bag` and `http://users.cecs.anu.edu.au/~u5419700/sample6.bag`

   **Description of help functions:**

   - def Relative2AbsolutePose (robot_abs, u):
     Calculates the robot new pose given previous pose and motion
     Input:
     - absolute coordinate of robot [x1,y1,theta1]
     - motion command with respect to robot frame [dx, dy, dtheta]

     Output:
     - absolute coordinate of robot next pose [x2,y2,theta2]
   - def Relative2AbsoluteXY (robot_abs,landmark_meas_xy):
     Calculates Landmark's absolute coordinate
     Input:
     - robot's absolute coordinate [x,y,theta]
     - landmark's measurement with respect to robot frame [x,y]

     Output:
     - landmark's absolute coordinate [x,y]
   - def Absolute2RelativeXY (robot_abs,landmark_abs):
     Expresses Landmark's Coordinate on Robot Frame
     Input:
     - robot's absolute coordinate [x,y,theta]
     - landmarks absolute coordinate [x,y]

     Output:
     - landmark's relative coordinate with respect to robot frame [x,y] i.e. landmark's measurement from robot
   - def ErrorFunction (solutionFile, GTFile):
     Calculates the error between GT data and SLAM obtained data
     Input:
     - solution file containing:
       * absolute coordinates of estimated landmark positions
     - GT file containing:

        * GT absolute coordinates of landmark positions

    Output:

    - error = Relative GT landmark positions - Relative estimated landmark positions

- def RelativeLandmarkPositions(landmark_abs, next_landmark_abs):
  Calculates the relative landmark positions
  Input:

    - absolute coordinate of landmark [x1,y1]
    - absolute coordinate of landmark next position [x2,y2]

    Output:

    - relative position [dx, dy]

A "README.txt" file is also provided explaining what each function does, its inputs and outputs and how to use it with some toy examples for some of the functions.

# 3 Task 0: Sensor Calibration

# 4 Task 1: Data Acquisition and Plotting

# 5 Task 2: State Estimation using Filtering

Tasks $0 \rightarrow 2$ were discussed in part I of the lab instructions.

# 6 Task 3: Kalman Filter Based SLAM

You are required to implement your own kalman filter equations utilizing measurements from robot on-board sensors to estimate for the robot poses and landmark positions in the environment.
An easy to follow skeleton of the algorithm is provided in the tutorial slides and a Python file "SLAM_students.py".
The solution of your estimation algorithm should be written to a solution file to be compared vs ground truth. This file should have the same format as the "Trajectory and Points" file from last lab, i.e. each line of should start with POINT2D and then label, x, y of the cylinder in world coordinates. Given that in the final demo, you are required to only provide a solution file with the landmarks positions in the environment, your solution file should only contain 2D POINTS.

## 6.1 Training

For you to practice, we advise you to use the provided rosbag files or to create your own rosbag file to do tests and compare with ground truth.
Ground truth landmark positions are provided for you to perform tests and evaluate the performance of your algorithm based on an error calculation function that is also provided in the help scripts "ErrorFunction.py" and that takes as input the arguments your solution file and the ground truth file. Error is calculated as the difference between relative ground-truth landmark positions and relative estimated landmark positions.

Given that in the final demo, you will be required to provide a solution file containing only landmark positions, the error calculation function calculates error of the estimated cylinders versus ground-truth cylinders, and thus your solution file should only contain 2D POINTS.
**This is your chance to get the best accuracy of your algorithm. In the final demo, you will not be provided a ground truth!**

## 6.2 Final Demo

You are required to run your robot in a 3m $\times$ 3m environment with scattered cylinders for exactly 2 minutes. Every team can chose the trajectory and the running mode (teleop or autonomously) that they consider would benefit the completion of the task and will provide the best accuracy of the landmark position estimation.
The goal is to map the environment as accurately as possible. Please consider designing the trajectory that would provide best coverage and accurate estimation of the position of the cylinders.
The ground truth for the positions of the cylinders in the final demo will **NOT** be provided a priory.
Each team will be given **120 seconds** to move the robot around the environment (Try to think of a "good robot trajectory" to cover most of the environment in the least possible time, so you can maybe have time to re-visit parts of the environment more than once) After running your robot, you will be required to submit a solution file, in this case, your solution will only contain 2D Points specifying positions of the cylinders in world coordinates. This will be compared to a Ground Truth file, that only tutors have, and you will be given a number that is a measure of the error of your estimation.
Each team will be granted a total of **10 minutes**; it is advisable to distribute them as follows; **4 minutes for preparation, 2 minutes to move your robot (mandatory), and 4 minutes to report your solution**. The order of presenting the final demo is according to your number in the list and there are no second attempts, so be prepared.

**SET OF RULES**

- Environment is a 3m x 3m with scattered cylinders.
- No robot trajectory to follow, Try to think of a "good robot trajectory".
- Each team has a total of **10 minutes**.
  - **4 minutes to set up the robot in the start position**
  - **2 minutes to move the robot and map the landmarks (strict time!)**
  - **4 minutes to report your solution**
- Order of presenting is according to your number in the list.
- No second attempts are allowed.

### 6.3 Competition [Optional]

Competition will follow the same set of rules as the final demo.

The accuracy of the returned map of the environment will determine the top three teams, however points will be added for the below extra miles.

Here is a list of extra miles you can go to win the competition:

- Upgrade sensor model.
- Post process measurements.
- Use odometry as an extra measurement.
- Utilize the IMU sensor readings.
- Autonomously navigate the robot in an optimized way to cover the whole environment in the least possible time, this will involve some sort of obstacle avoidance given that you have no prior knowledge of the cylinders position in the environment.

**HINT**: Some of these improvements are really easy to implement and can greatly improve your solution so you are highly advised to try them.

# 7 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 Lab Report should demonstrate your understanding of the subject asked. The full marks of the report is 20 marks; the demonstration amounts 20 marks, which includes 2 marks for the clarity of your source code.

The first half of the report will be about reporting the results and discussions of the lab tasks. Please follow a technical report writing style (containing, but not limited to: introduction, methods, experimental results, discussions, conclusions, references, etc.). This contributes to up to 10 marks of the report.

The second half of the Lab Report will be based around answering the technical questions in the two instruction documents. These are compulsory to test your understanding of the work done in this lab. This contributes to up to 10 marks of the report.

**DUE DATE:**

The Project Demonstration will be due to Week 12's Lab sessions, dedicating the **full 3 hours** to the demo assessment.

For your convenience, the lab report will be due one week later, on the Sunday of the following week (Week 13). However, you are strongly advised to complete your project report by Week-12, in order for a better preparation of your Final Exam.

# 8 Turtlebot Lab 4 (Part II) Report Problems

Note: Lab Report must be your individual work.
Note: Question 1 and 2 are in the Part I instruction.

**Question 1**: **Theory** (1 mark)

**Question 2**: **Gaussian Motion Model** (2 marks)

**Question 3**: (2 marks)
Suppose you are a witness to a night-time hit-and-run incident involving a taxi in Athens. It is known a priori that all taxi cars in Athens are either blue or green.

You swear, under oath, that the taxi was blue. Extensive testing shows that, under the dim lighting conditions, discrimination between blue and green is 75% reliable.

Is it possible to calculate the most probable color for the taxi? (Hint: distinguish carefully between the proposition that the taxi is blue and the proposition that the taxi appears to be blue.)

What is your final estimation, given a fact that 9 out of 10 Athenian taxis are green? Explain how you reach your estimation.

**Question 4**: **Markov Localisation** (3 marks)

A robot is living in a 1D world. The world is divided in 5 cells and it is cyclic. The current belief of the robot is:

| | | | | |
|---|---|---|---|---|
| 1/9 | 1/3 | 1/3 | 1/9 | 1/9 |

The robot moves one cell to the right, $u = 1$. The motion is noisy with:

$$p(x_i|x_i) = 0.1$$
$$p(x_{i+1}|x_i) = 0.8$$
$$p(x_{i+2}|x_i) = 0.1$$

This means that the probability that the robot did not move after the action is 0.1, the probability that the desired action executed is 0.8 and the probability that the motion overshut is 0.1.

What is the belief of the robot after the noisy action was executed.

| | | | | |
|---|---|---|---|---|
| ? | ? | ? | ? | ? |

Explain the steps involved in the computation of the robots prior. In the case you implemented a short program to do that, write down the pseudo code.

(Hint: Apply the total probability theorem)

**Question 5**: **Kalman Filter Localisation** (2 marks)

a) Which are the steps of a Kalman filter and what are the names of the belief after each step?

b) Which intermediate steps are needed to derive the prior:

$$\overline{bel}_{(x_t)} = \eta \int exp\left\{-\tfrac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T \Lambda_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\right\}$$
$$exp\left\{-\tfrac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})\right\} dx_{t-1}$$

from the current belief:

$$bel_{(x_t)} = \int p(x_t|u_t, x_{t-1}) \qquad bel(x_{t-1})dx_{t-1}$$
$$\Downarrow \qquad\qquad \Downarrow$$
$$\sim N(x_t; A_t x_{t-1} + B_t u_t, \Lambda_t) \qquad \sim N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})$$

c) Compare Kalman filter algorithm with Markov algorithm for mobile robot localisation. Please list their pros and cons. Which algorithm can be used to solve the kidnapped robot problem (i.e. location recovery)?