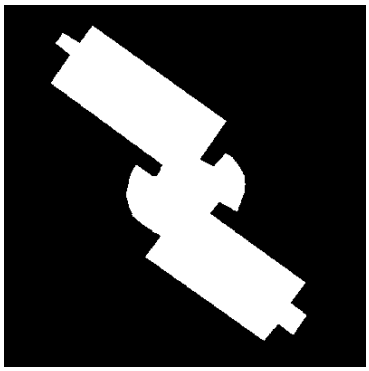
# COMP4680/8650: Advanced Topics in SML

## Assignment #2: Robot Localisation

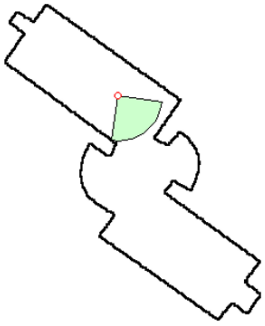
August 28, 2014

A robot is lost somewhere in the computer science building. Your job is to help find it!

(a) Satellite View of Building



(b) Map of Building



(c) Lost Robot

**Figure 1:** Computer Science Building at ANU.

## Logistics

In this project you will implement a particle filter for tracking a robot. You are provided with Matlab and Python starter code but may choose to write your particle filter in the programming language of your choice—but you cannot use any libraries beyond the standard one provided with the language.

You can work individually or in teams of two. Only one member of the team needs to submit the work.

The due date for this project is **23:55 on 26 September, 2014**. You are allowed to resubmit as often as you like and your grade will be based on the last version submitted. Late submissions will be penalized (0.75 for up to one day late, 0.5 for up to two days late). This project is worth 20% of your final grade.

You must submit your source code and a report with answers to the questions outlined below. Your report should include the names and university IDs of team members. Along with your source code you should submit a short `readme.txt` file that explains how to run it. Your code should also be well commented.

# 1 Probabilistic Filtering on a Hidden Markov Model

We are interested in computing the probability of the state $x_t$ at time $t$ given all previous measurements $y_1, \ldots, y_t$. That is, for each time step we would like to estimate

$$P(x_t \mid y_1, \ldots, y_t). \tag{1}$$

In this part of the project we ignore control inputs $u_1, \ldots, u_t$.

(a) **[5 Marks]** Show that the joint distribution over $x_{0:t}$ and $y_{1:t}$ can be written as

$$P(x_0, \ldots, x_t, y_1, \ldots, y_t) = P(x_0) \prod_{i=1}^{t} P(x_i \mid x_{i-1}) P(y_i \mid x_i) \tag{2}$$

where $P(x_0)$ is the prior. Clearly explain all independence assumptions that you used.

(b) **[5 Marks]** Show that

$$P(x_t \mid y_1, \ldots, y_{t-1}) = \int P(x_{t-1} \mid y_1, \ldots, y_{t-1}) P(x_t \mid x_{t-1}) \mathrm{d}x_{t-1} \tag{3}$$

(Note the conditioning stops at $y_{t-1}$).

(c) **[5 Marks]** Now show that

$$P(x_t \mid y_1, \ldots, y_t) = \frac{P(x_t \mid y_1, \ldots, y_{t-1}) P(y_t \mid x_t)}{\int P(y_t \mid x_t) P(x_t \mid y_1, \ldots, y_{t-1}) \mathrm{d}x_t} \tag{4}$$

(d) **[5 Marks]** Explain how you can use the above to recursively estimate $P(x_t \mid y_1, \ldots, y_t)$.

# 2 Robot Localisation: Particle Filter

We model the state at time $t$ as $x_t = (a_t, b_t, \theta_t)$, where $(a_t, b_t)$ is the location of the robot (pixel coordinates) in the map and $\theta_t$ is its heading relative to the $x$-axis (see Figure 2). Measurements at time $t$ are $y_t \in \mathbb{R}^{11}$ representing eleven independent noisy distance readings from a laser range finder at regular spaced intervals from $\frac{-\pi}{4}$ to $\frac{\pi}{4}$ radians around the robot's current heading (so the fifth measurement represents the measured distance in the direction that the robot is facing). Distance readings are provided in normalised (pixel) units. The laser range finder has a maximum range of 50 units.

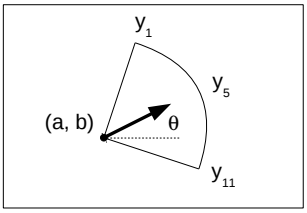



**Figure 2:** Representation of robot state.

Laser measurements can be corrupted by two types of noise: random Gaussian noise with variance 1 and sensor failure with probability 0.01 (which returns a maximum reading).

At each iteration the robot is instructed to move forward a given distance $u_t \in \mathbb{R}_+$. The robot may turn between $\frac{-\pi}{2}$ and $\frac{\pi}{2}$ radians, and the distance it travels is also corrupted by noise. Since the control instruction does not include angular information you will need to model this as noise.

(a) **[5 Marks]** Derive reasonable motion $P(x_t \mid x_{t-1}, u_t)$ and measurement $P(y_t \mid x_t)$ models.

(b) **[50 Marks]** You are provided with two scenarios (`easy` and `hard`) and some starter Matlab and Python code.[1] Comments in the code indicate where you should make your modifications. You may choose to ignore the helper code and use whatever programming language you like.

For each scenario you are provided with a map, a sequence of laser measurements $(y_t)$ and a sequence of odometry readings $(u_t)$. For the `easy` scenario you are also provided with a video and text file showing you the true robot trajectory to help debugging.

Implement a particle filter to recursively estimate $P(x_t \mid y_1, \ldots, y_t, u_1, \ldots, u_t)$. Determine the most likely trajectory of the robot for each scenario and plot it on the map. Be sure to describe how you determined the most likely trajectory. (You must submit your source code and include the map images in your report).

(c) **[5 Marks]** When does your algorithm become most confused about the location of the robot? How does your estimate change with the number of particles?

# 3 The Kidnapped Robot Problem

An evil Bayesian statistician kidnaps your robot and moves it to a different part of the building.

(a) **[10 Marks]** Modify your code to handle cases where at each time step there is som small probability that your robot is suddenly transported to another location on the map. Run your code on the scenario `kidnapped`.

(b) **[5 Marks]** Plot the most likely trajectory of your robot. Be sure to state how you determined the most likely trajectory.

(c) **[5 Marks]** Assuming that the robot was kidnapped exactly once, at what time point did this kidnapping occur?

[1] For visualization in Python you will need to install the Python Image Library (PIL or Pillow).