

Homework #2

Bayesian Statistics and Tracking

Due: Start of Wednesday lecture in week 7 (1 September)

1 Overview

The goal of this assignment is to test your understanding of Bayesian Statistics and Perception.

You should feel free to ask the Lecturer questions and discuss the assignment with others, but you should not copy someone else's work. (e.g. You can get someone to show you how to answer a question, but you should write your answer up yourself from what you remember at least 30 minutes after they've left.)

Total marks for this assignment: 15. It will be scaled to 5% of your grade.

This assignment is a programming assignment, but not on the robots. You will find in the CSE directory `~cs3431/public_html/wiki/assignments/2010/Homework2src/4` python files, `main.py`, `filter.py`, `model.py`, and `rand.py`.

You are to email `willlu@cse.unsw.edu.au` a single python source file, `model.py-userid`, containing the observation and motion models for the following robot. (This robot is a simplified form of an AIBO on the robocup field.)

2 Mad Lecturer Rules the Robot!!!!

There is a robot in your lab that moves about a 2D flat world. At night a mad computer science lecturer is creeping into your lab to move the robot around and get it lost. You have an some Kalman filter code and decide to re-purpose it to thwart the lecturers evil plans.

The robot has defined 'front'. It can walk forward and sideways, and turn on the spot. It turns about a central point, and has a camera mounted on a panning base immediately above this central point. All distances are measured in meters, all angles are measured in radians. All coordinate systems are right-handed with angles increasing counter-clockwise. The robot does not know its absolute position or which way it is facing in the world, and initially it is completely uncertain about its location.

When the lecturer creeps into the lab, he feeds the robots locomotion module walking commands. Unbeknownst to the lecturer, the locomotion module will be feeding those commands to the Kalman Filter as a column vector of mean movements,

$$\begin{bmatrix} \Delta Fwd \\ \Delta Left \\ \Delta \phi \end{bmatrix}$$
, for the motion (See Figure 1).

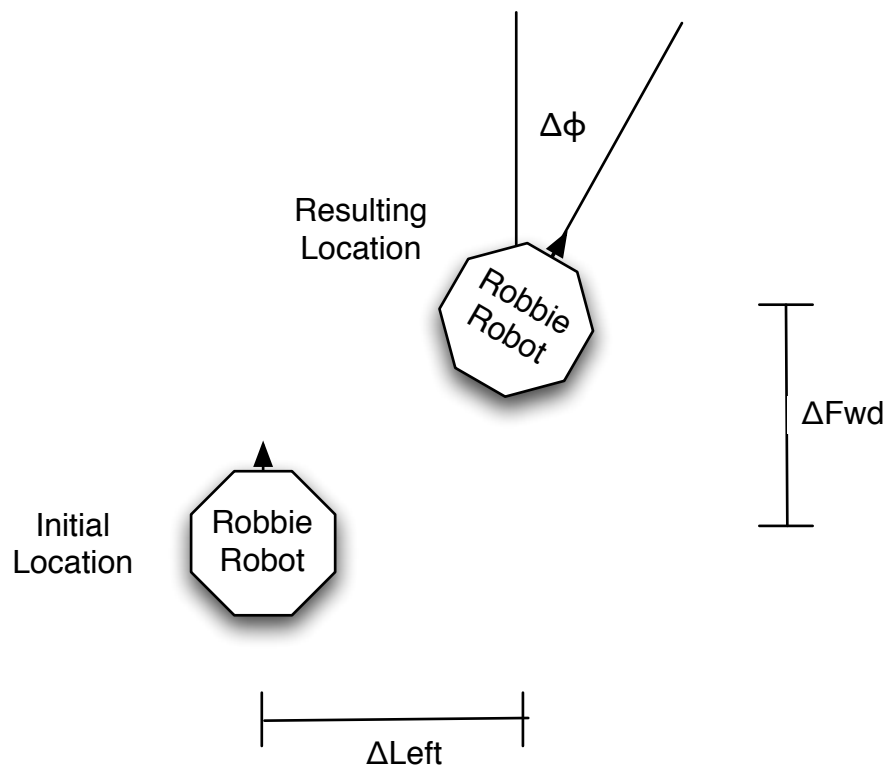


Figure 1: The numbers representing a change in position of our robot. Note that the ΔLeft and $\Delta\phi$ shown in the diagram are both negative for the displayed motion.

The robot's camera is also being left switched on. As the robot is being moved about it receives observations from that camera and vision processing software. In particular there are 4 coloured beacons placed about the robot's world that it can recognise. Each time the camera detects a beacon it passes an observation to the Kalman filter. This observation is a column vector with three numbers: $\begin{bmatrix} \text{ID} \\ \text{distance} \\ \Delta\text{angle} \end{bmatrix}$, the beacon ID, and the *relative* distance and angle from the robot to the beacon. The locations of the beacons on the field are shown in figure 2.

The robot's motion and observations are not perfect. In particular, for any movement there is gaussian noise with standard deviation $0.01m + 10\%$ forward/backward, $0.01m + 15\%$ left/right, and $5/180 * \pi + 10\%$ turning. (*i.e.* if the robot tries to walk forward 1m, then we expect it to have a standard deviation on that motion of $0.01 + 0.1 \times 1 = 0.11m$ forward/backward, $0.01 + 0.1 \times 0 = 0.01m$ left/right and $5/180 * \pi + 0.1 \times 0 = 5^\circ$ in angle.) For any observation there is gaussian noise with standard deviation $0.025m + 1\%$ in distance and $5/180 * \pi + 10\%$ radians in angle. Luckily, beacon identification is perfect (although you might want to think about how you would handle the case if it wasn't). Note that all percentages in this paragraph are absolute values – you don't get negative variance by moving right.

2.1 Question 2.1 (3 marks):

Change the implementation of the functions `expectedMotionResult()`, `motionSample()`, `expectedObservation()` and `observationProbability()` to model this robot. These functions should model the robot exactly as described.

2.2 Question 2.2 (3 marks):

You should note while answering the previous question that the motion model is non-linear. Describe in a comment in your python file how you might approximate the non-linear motion model with a linear motion model.

2.3 Question 2.3 (9 marks):

Change the implementation of the other functions in `model.py` to model this robot (use the best approximation you can where you need to return a linear approximation for a non-linear function).

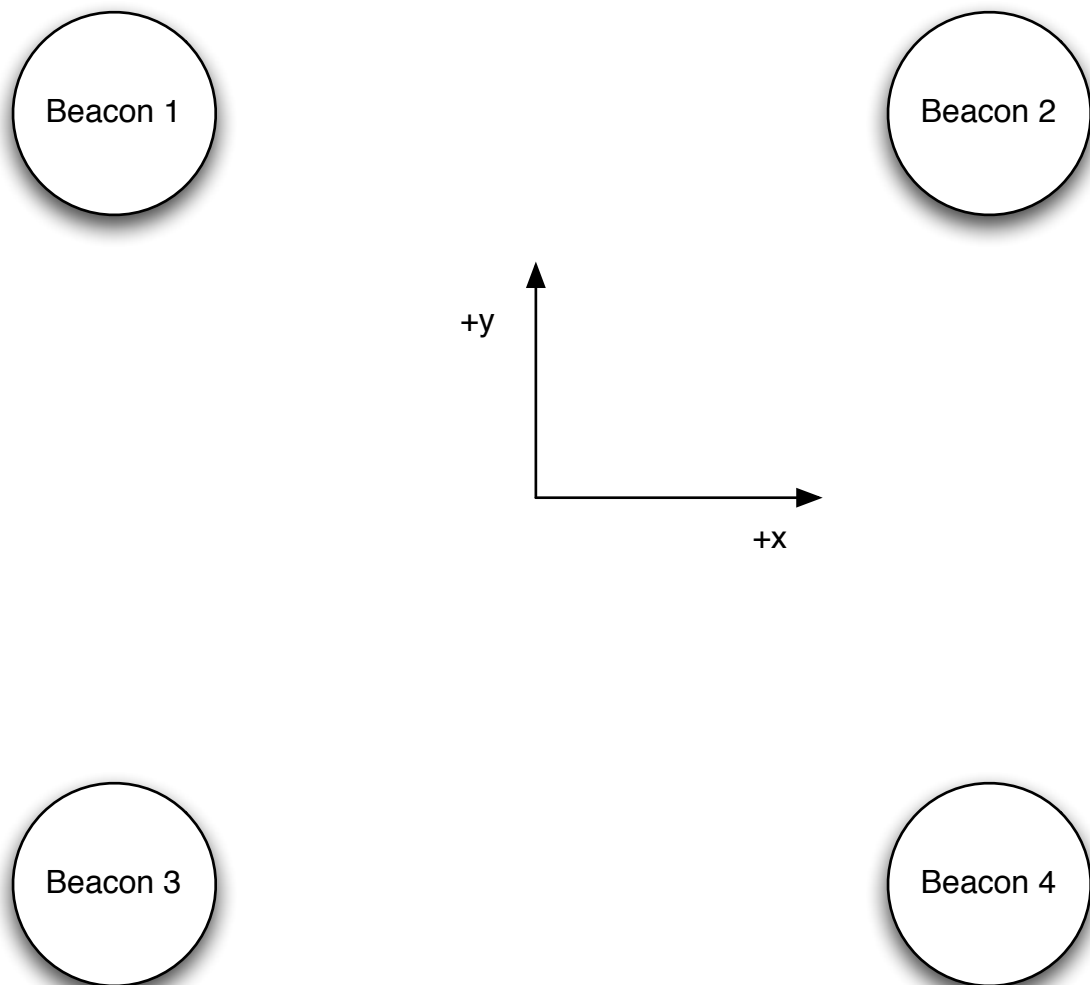


Figure 2: The coordinate system of the field. Beacons are placed at $\pm 2\text{m}$ around the origin with the shown IDs. In order the beacon locations are $(-2, 2)$, $(2, 2)$, $(-2, -2)$ and $(2, -2)$. The x -axis has a global angle of 0 radians.

Notes:

- For the observation matrix, H , make the row elements for the beacon ID all 0.
- The $\text{atan2}(dy, dx)$ function is useful for getting angles from relative coordinates.
- Approximate the partial derivatives of the $\text{atan2}(dy, dx)$ function as follows:
 $\frac{d}{dx}\text{atan2}(dy, dx) = \frac{dy}{dx^2+dy^2}$, and $\frac{d}{dy}\text{atan2}(dy, dx) = \frac{-dx}{dx^2+dy^2}$.
- You'll want to make sure the functions `normaliseStateAngles()` and `normaliseObsAngles()` know which elements of the state and observation vectors are angles. You'll also want to normalise the state angles at the end of your `motionSample()` function.