CS 4758/6758: Robot Learning: Homework 5 preview

Due: April 20, 2010

1 Kalman Filters

1.1 Multiple sensors (30 pts)

Suppose we have a continuous system with state $x \in \Re^N$. The state at time k is determined by the equation:

$$x_{k+1} = x_k + w_{k+1} \quad \text{where } w_{k+1} \sim N(0, Q)$$

Now we have two sensors z_k and y_k. One sensor gives data at $k = 1, 3, 5, \ldots$, and the other sensor gives data at $k = 2, 4, 6, \ldots$.

$$z_k = x_k + v_k \quad \text{where } v_k \sim N(0, R_1)$$
$$y_k = x_k + e_k \quad \text{where } e_k \sim N(0, R_2)$$

Derive the time and measurement update equations for this situation.

1.2 Simple Kalman filter Matlab implementation (20 pts)

Implement a Kalman filter for a system with state and measurement update equations:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{w}_t \quad \text{where } \mathbf{w}_t \sim N(0, \mathbf{Q})$$

$$\mathbf{y}_t = \mathbf{B}\mathbf{x}_t + \mathbf{v}_t \quad \text{where } \mathbf{v}_t \sim N(0, \mathbf{R})$$

and constants:

$$\mathbf{A} = \begin{bmatrix} .99 & -.16 \\ .16 & 1.02 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} .5 & .5 \\ .5 & -.5 \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} 1 & .2 \\ .2 & 1 \end{bmatrix}$$

.

The file `hw5p2.txt` contains a sequence of measurements $\mathbf{y}_1 ... \mathbf{y}_t$. Supposing $\mathbf{x}_0 = [0, 0]^T$, compute and plot the corresponding $\hat{\mathbf{x}}_1 ... \hat{\mathbf{x}}_t$. Attach your source code.

2 Particle Filters: Localization using ROS

(50 points)

Background:

In ROS, code is divided into stacks based on its function. The navigation stack provides software to help a robot navigate a 2D environment by using a planar laser. One of the functions of the navigation stack is localization which will be the focus of this problem. [1]

ROS uses a package called `amcl` to calculate a robots position on a known map based on laser data. To install the necessary packages download and extract the resource files for this problem. Run `nav_install.sh` by opening a terminal window, changing to the directory you extracted and

[1] The included files assume you have ROS installed. If you had trouble running ROS last time, it is likely that this problem will also give you trouble. A computer will be made available in Upson 317 that will have ROS set up to allow you to do this assignment. You can also come to Jonathan's office hours Monday or Thursday.

running `./nav_install.sh` twice. To start amcl along with a simulated robot enter the following into the command line [2] :

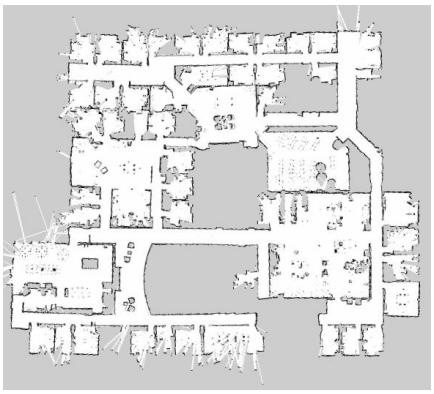
```
roscd localize_demo
roslaunch launch/simple_nav.launch
```

You can move the robot around with the mouse in stage to see how different movements affect the localizer. If the localizer gets too far off you may need to restart the simulation by killing it with ctrl+c and restarting it. Like the last assignment some computers may have trouble with the graphics of stage. You may want to minimize any windows that overlap the stage window and when you run controller.py you may need to minimize stage to get it to update.

To see a graph of the localizer's estimated position versus the real position open a new terminal window and run:

```
roscd localize_demo
./scripts/plot.sh
```

You can pause the graph and press the pan button to let you zoom in and out to capture the entirety of a session.



Question:

The robot, with a laser scanner, was placed in the lab (Fig. 2) but there was some error when initializing its location.

```
roscd localize_demo
roslaunch launch/off_nav.launch
```

In order to make the robot figure out where it is, we give you a piece of code (`controller.py` or `controller.cpp` in `localize.zip`) that randomly moves the robot around hoping that the localizer will be able to correct itself. This file can be run by entering:

```
roscd localize_demo/nodes
python nodes/controller.py
```

or

```
roscd localize_demo/nodes
make
./bin/controller
```

Please edit this file so the localizer corrects itself faster and crashes less into the walls.

[2] You may notice that the angle wraps around from -1 to 1 so the plot may jump between the two extremes.

1. Run the random controller for long enough that the true location and the localized position match and include the plot saved from `plot.sh` stretched to fit the time it takes to localize. If the robot seems stuck or the localization is way-off you may want to reset it.

2. Give a pseudocode algorithm that uses the laser data to improve the localization. (Briefly describe your strategy using this pseudocode.)

   Change `controller.py` to implement your algorithm, test it on `begentle.launch`, `bringiton.launch` and `nightmare.launch`.[3] Include the graph for each one of them for your controllers results from `plot.sh`.

[3] You do not need to complete nightmare.launch for receiving full credit; nightmare.launch is for bonus credit.