MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.01—Introduction to EECS I
Fall Semester, 2007

**Assignment 12, Issued: Tuesday, Nov. 20**

## To do this week

### ...in Tuesday software lab

1. Start writing code and test cases for the numbered questions in the software lab. Paste all your code, including your test cases, into the box provided in the "Software Lab" problem on the on-line Tutor. What you submit to the tutor will not be graded; what you hand in next Tuesday will.

### ...before the start of lecture next Tuesday

1. Do the lab writeup, providing written answers (including code and test cases) for **every** numbered question in this handout.

On Athena machines make sure you do:
`athrun 6.01 update`
so that you can get the `Desktop/6.01/lab12` directory which has the files mentioned in this handout. You need the whole lab12 distribution for the software lab; use one of our laptops for the software lab.

# State estimation

In the next two software and two robot labs, we'll use basic probabilistic modeling to build a system that estimates the robot's pose, based on noisy sonar and odometry readings. We'll start by building up your intuition for these ideas in a simple simulated world, then we'll move on to using the real robots.

This week we'll start by working with a non-deterministic grid-world simulator. You should work with your partner on this.

## Grid World Simulator

Don't use the Idle Python Shell for this lab. You can use the Idle editor, but you cannot run the code from inside Idle.

In the `lab12` folder you will find
`StateEstSM.py`.
Open this file in Emacs, Idle or whatever editor you like. **If you use Idle as an editor, do not try to evaluate the python commands in the Idle Python Shell.**

Then, open a Terminal window (if you're on Athena, remember to do `add -f 6.01`), then connect to the lab12 director

```
cd ~/Desktop/6.01/lab12
```

and type `python`.

```
> python
Python 2.5 (r25:51918, Sep 19 2006, 08:49:13)
[GCC 4.0.1 (Apple Computer, Inc. build 5341)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

You can type commands to Python here. In this Python, type

```
>>> from StateEstSM import *
```

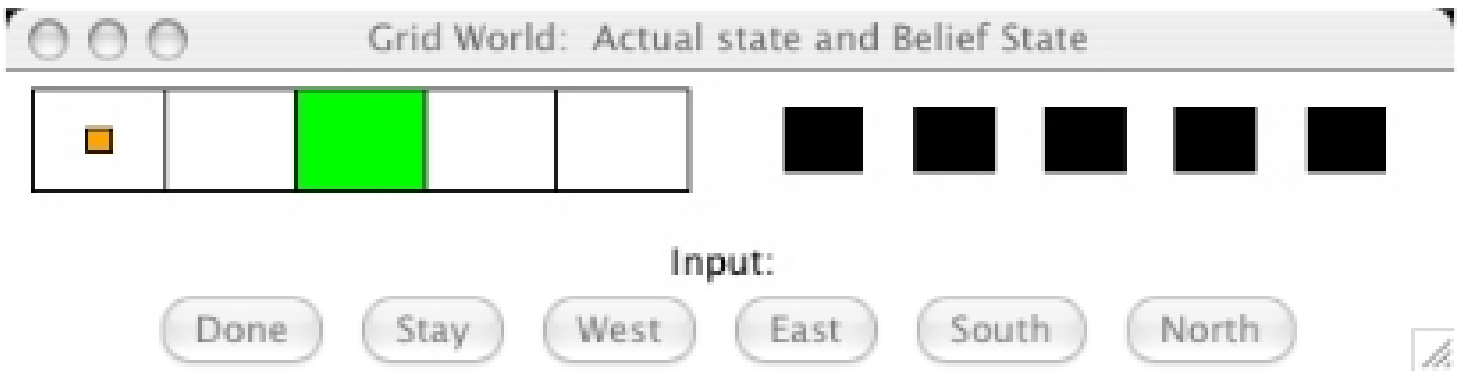
As the lab goes along, if you edit `StateEstSM.py`, then you'll need to go back to this window and type

```
>>> reload(StateEstSM)
>>> from StateEstSM import *
```

You'll see a command at the end of the `StateEstSM.py` file, that says:

```
tw = makeGridSim(5, 1, [[],[],[[2,0]],[]], [0,0], perfectSensorModel, \
                 perfectMotionModel)
```

Execute this command in Python (on your Terminal window). When you evaluate it, you should see a window that looks like this:




This is a world with 5 possible "states", each of which is represented as a colored square (on the left). The possible colors of the states are white, black, red, green, and blue. In this example, four squares are white and one is green. There is a small orange rectangle representing the square that our simulated robot is actually occupying. On the right are five squares that start out being black; the color in those squares represents how likely the robot thinks it is that it's in that square. This is called the *belief state*. The colors illustrating the belief state are: black, when the value is what the uniform distribution would assign (0.2 for 5 states), shades of green when the probability is higher than the uniform, and shades of red when it is below the uniform value.

The arguments to the `makeGridSim` function are:

- The dimension of the world in x
- The dimension of the world in y
- Four lists of coordinates, each specifying the location of colored squares. The first list gives the locations of black squares, the second red, the third green, the last blue. Squares unspecified in any of those lists are white.
- A pair of indices specifying the robot's initial location
- A model of how the sensors work
- A model of how the actions work

So, this grid world is 5-by-1, with one green square, the robot initially at location (0,0), and perfect sensor and motion models.

You can issue commands to the robot by typing:

```
tw.run()
```

and clicking on the buttons in the simulator window. The buttons correspond to the five available actions: Stay, West, East, South and North. Clicking on the Done button returns control to Python. The window will remain after you click Done. **Do not kill the window until you are completely done with it.** You can type:

```
tw.reset()
tw.run()
```

to start interacting with the window again after you've typed Done.

**Question** 1. Just to get the idea of how this works, move the robot east and west a few times, and write down what is written on the screen. It shows the actual numeric values associated with the robot's belief that it is in each of the squares. You should be able to relate this, at least roughly, to the HMM example in the notes.

This model is a slight variation on a hidden Markov models (HMM) that we saw during lecture. The only difference is that the robot can select different actions (like trying to move north or trying to move south), and those different actions will cause different state-transition distributions.