CSCI–4190 Introduction to Robotic Algorithms, Spring 2003
Assignment 2: out February 20, due February 27 and March 6

Localization and the extended Kalman filter

In this assignment, you will write a program that simulates the motion of a mobile robot and keeps track of the uncertainty in its configuration. In the first part of the assignment, the robot will execute only movement commands; the uncertainty in the robot configuration will increase due to accumulated errors from executing each movement. In the second part of the assignment, the robot will occasionally use a simulated laser rangefinder to measure the distance to an obstacle in a given direction. We assume the robot has a map, so it can compare the sensor reading with the expected value (based upon the current estimated robot configuration). You will implement an extended Kalman filter to merge this uncertain measurement with the current estimate of the robot configuration.

Due dates

- The written questions (Section 5 are due on February 27.

- Your program and other stuff (see Section 6) are due on March 6.

Support code libraries

The support code will make use of the *CGAL*, *dolt*, and *filereader* libraries, as before. For this assignment, there is one more library: *SVL* (Simple Vector library). This library provides classes to represent vectors and matrices as well as mathematical operations, in particular the matrix inverse. You can find documentation at:

```
http://www.cs.cmu.edu/~ajw/doc/svl.html
```

There is a link to download it under the "Description" section.

1   Program operation and interaction

When your program starts, it should draw the world (boundary and obstacles) and the robot (in its starting configuration). In general, your program should always draw on the screen:

- the robot in its current estimated position and orientation (also include a point at the reference point of the robot)

- a 95.4% confidence limit on the robot's position (using an ellipse) and on the robot's orientation (using a cone)

- a path from the start, through all previous estimated final positions, to the current estimated position

- a point at the robot's true position and a line segment showing its true orientation

After this point, the user should be able to press 'q' to quit or 'n' to execute the next command. There are two types of commands: a "move" command and a "sense" command.

- If the command is a "move" command, then your program should:

    - Give the commanded motion to the robot simulator

- Compute an updated estimate of the robot configuration based on the commanded motion
- Update the display for the new configuration estimate and uncertainty

- If the command is a "sense" command, then your program should:

  - Display the beam from the robot (i.e. a line of the appropriate length in the proper direction from its current estimated configuration).
    Since the robot is not where it thinks it is, the end of this line will either not reach an obstacle or will intersect an obstacle.
  - Compute a revised estimate of the robot configuration and its uncertainty.
  - Wait for the user to press 'n' again, and then update the display.
    The beam should be removed from the display, and the last point of the robot path should be changed to the current estimated position.

Your program should also print a narrative of its calculations to `cout`. For example:

```
==>The robot moved: distance=1.75, angle=90
Current state estimate is (x, y, th) = (1.10114, -1.25, 86.0965)
New state estimate is (x, y, theta) = (1.22095, 0.505793, 181.969)
New state covariance matrix is:
[[0.0200199 -0.000998142 -0.00991148]
[-0.000998142 0.00285555 0.000676311]
[-0.00991148 0.000676311 0.0126447]]

True state is (x, y, th) = (1.15, 0.5, 180.000)
Uncertainty ellipse is:
  minor radius is 0.131259
  major radius is 0.35163
  orientation is 169.91084 degrees
  2*sigma_theta is 12.44284  degrees

==>Commanding the robot to sense at an angle of 90
The range is 2.00572

The predicted range was 2.00698
The derivatives are:
  dh/dx = 0, dh/dy = 1.00059, dh/dth = 0.0689879
Revised state estimate: (x, y, theta) = (1.22157, 0.504722, 181.936)
Revised covariance matrix is:
[[0.0203742 0.0134885 -0.00853237]
[0.0134885 0.0301436 -0.0188613]
[-0.00853237 -0.0188613 0.0181603]]
Uncertainty ellipse is:
  minor radius is 0.0481291
  major radius is 0.343865
  orientation is 75.51344 degrees
  2*sigma_theta is 10.01445 degrees
```

2   Simulated robot operation

The support code will provide a simple robot simulator. The motion commands to the robot will be a forward straight motion followed by a turn in place (about the reference point). The forward distance and turning angle are the parameters for a command.

The robot, however, does not execute the commanded motion perfectly. It adds some Gaussian noise to the commanded motion and updates the actual configuration of the robot accordingly. The width of the Gaussian distribution, however, depends on the magnitude of the motion so that smaller motions have less error. In fact, we will assume that the standard deviation of the Gaussian is proportional to to magnitude of the motion command (or to the magnitude of the laser rangefinder reading). The constants of proportionality are given in the problem specification file:

```
commandDstSigmaRatio
commandAngSigmaRatio
rangeSigmaRatio
```

For example, if `commandDstSigmaRatio` is 0.02, then for a commanded motion of $d = 10$ m, $\sigma_d = 0.2$ m. You should assume that the error in distance for the forward motion and the error in angle for the turn are independent.

This robot will be equipped with a simulated a laser rangefinder. To take a measurement, you must specify the angle (relative to the forwards direction of the robot) in which to point the laser rangefinder. We will assume that the sensor is pointed exactly at the specified angle. The laser rangefinder then emits a pulse of light and measures the time until it sees the reflection from the first obstacle it encounters. Of course, there is some error in a real measurement, so some Gaussian noise is added to the distance measurement.

3 The extended Kalman filter

The state of the robot will be the vector $x = [\ x \quad y \quad \theta\ ]^T$ and the motion command will be the input vector $u = [\ d \quad a\ ]^T$. where d is the distance and a the angle. We have assumed that there is noise associated with the input u, so we must represent this using the covariance matrix:

$$P_u = \left[\begin{array}{cc} \sigma_d^2 & \sigma_{da} \\ \sigma_{ad} & \sigma_d^2 \end{array} \right] \tag{1}$$

At each step, we will maintain an estimate of the state, notated $\hat{x}(k)$ for step k. To characterize its uncertainty, we maintain an associated state covariance matrix (here, also for step k):

$$P_x(k) = \left[\begin{array}{ccc} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_\theta^2 \end{array} \right] \tag{2}$$

At the beginning, when we know the robot's configuration exactly, the elements of the state covariance matrix are all zero.

3.1 Compounding

We use Φ to represent the function that calculates the next state as a function of the currest state estimate and the input. Our model of the system evolution is:

$$x(k+1) = \Phi[x(k), u(k)] \tag{3}$$

Note that this differs slightly from the usual formulation of the extended Kalman filter in that there is no additional noise term $w(k)$. This is because we are assuming that there is noise associated with the elements of $u(\cdot)$ instead.

At step k, we have an estimate of the state $\hat{x}(k)$ and its covariance $P_x(k)$. To update the estimate, we compute:

$$\hat{x}(k+1|k) = \Phi[\hat{x}(k), u(k)] \tag{4}$$

Recall that $\hat{x}(k+1|k)$ is a preliminary estimate of the new state after the motion command at step k.

In order to compute the covariance of this preliminary estimate, we must linearize the system. The first order approximation to Φ about the current state estimate is:

$$\Phi[x, u] \approx \Phi[\hat{x}(k), u(k)] + J_x(x - \hat{x}(k)) + J_u(u - u(k)) \tag{5}$$

where J_x and J_u are the Jacobians of Φ:

$$J_x = \begin{bmatrix} \frac{\partial \Phi_x}{\partial x} & \frac{\partial \Phi_x}{\partial y} & \frac{\partial \Phi_x}{\partial \theta} \\ \frac{\partial \Phi_y}{\partial x} & \frac{\partial \Phi_y}{\partial y} & \frac{\partial \Phi_y}{\partial \theta} \\ \frac{\partial \Phi_\theta}{\partial x} & \frac{\partial \Phi_\theta}{\partial y} & \frac{\partial \Phi_\theta}{\partial \theta} \end{bmatrix} \qquad J_u = \begin{bmatrix} \frac{\partial \Phi_x}{\partial d} & \frac{\partial \Phi_x}{\partial u} \\ \frac{\partial \Phi_y}{\partial d} & \frac{\partial \Phi_y}{\partial u} \\ \frac{\partial \Phi_\theta}{\partial d} & \frac{\partial \Phi_\theta}{\partial u} \end{bmatrix} \tag{6}$$

Note that these Jacobians should be evaluated at $\hat{x}(k)$.

The preliminary state covariance can then be computed as:

$$P_x(k + 1|k) = J_x P_x(k) J_x^T + J_u P_u(k) J_u^T \tag{7}$$

If there is no measurement after the motion, then this preliminary estimate becomes the final state estimate at step $k + 1$, i.e. $\hat{x}(k + 1) = \hat{x}(k + 1|k)$ and $P_x(k + 1) = P_x(k + 1|k)$.

3.2   Merging

When we do have a measurement at step k, this information must be incorporated into the state estimate and its covariance. We assume that this measurement process can be modeled by the equation:

$$z = H[x] + v(k) \tag{8}$$

The H function computes the sensor value from the state, and v is a random variable representing the noise in the measurement. In our problem, the measurement z will be a scalar, the measured range from the sensor.

We compute the error between the actual measurement $z(k)$ and the predicted measurement (without noise) based on our best estimate of the state:

$$r(k + 1) = z(k + 1) - H[\hat{x}(k + 1|k)] \tag{9}$$

Now we can compute the final state estimate at step k

$$\hat{x}(k + 1) = \hat{x}(k + 1|k) + K(k + 1) r(k + 1) \tag{10}$$

where $K(k + 1)$ is the Kalman gain:

$$K(k + 1) = P(k + 1|k) J_H^T (J_H P(k + 1|k) J_H^T + C_v)^{-1} \tag{11}$$

Here, J_H is the Jacobian of the H function and C_v is the covariance matrix for the measurement noise. The new state covariance matrix is:

$$P(k + 1) = (I - K(k + 1) J_H) P(k + 1|k) \tag{12}$$

4   Miscellaneous notes

- For orientation uncertainty, assume that this is a 1-dimensional Gaussian with variance σ_θ^2; the 95.4% confidence limit is the interval $\pm 2\sigma_\theta$ about the mean.

- For position uncertainty, take the upper left 2×2 submatrix of P_x. See the notes, Section 5.1.10, for information on how to calculate ellipse parameters from this covariance matrix.

- I suggest you adopt the following convention: all internal representation of angles should be in radians, all external representations (e.g. when printed to the screen or read from a file) should be in degrees.

5 Written questions

1. Write out the specific formulas you will use in this assignment, in particular:

   - $\Phi[x, u]$ from Equation 3
   - the Jacobians J_x and J_u based on Equation 6 (i.e. calculate the derivatives)
   - in the covariance matrix P_u, what will the σ_{da} and σ_{ad} terms be?

2. Copy the equation for the Kalman gain (Equation 11) and indicate the size of each matrix in this formula. Also indicate the size of the matrix resulting from the inverse operation.

3. Write out the form of the Jacobian J_H. (See Equation 6 for an example). Explain what each term is. (Relate it to the sensor or measurement process, not just "this is the derivative of z with respect to x.") Explain how you would compute these derivatives. (You do not have to calculate these derivatives, as it is a little tedious, and I have already done this for you in the support code.)

6 Stuff to turn in with your program

You should test your program to verify that it is working properly and to explore the effectiveness and limits of the extended Kalman filter. Turn in the following when you turn in your program:

1. Convince us that your program is working correctly by submitting input files that you used to test your code! The input files should be turned in electronically with your code, but you should include a hardcopy description of the tests and how they indicate that your program is correct.

2. Create a test input that illustrates "filter divergence". Turn in the input files electronically and describe (on hardcopy) your example and how it illustrates filter divergence.

3. Answer the following question (to be turned in on hardcopy):

   When calculating confidence limits, Section 4 directed you to use only the $\sigma_x^2, \sigma_y^2, \sigma_{xy}$, and σ_{theta}^2 terms from P_x. This means that the $\sigma_{x\theta}$ and $\sigma_{y\theta}$ terms were not used. In general, these elements of the covariance matrix will not be 0. What does this imply about the position and orientation and their uncertainty?