

CSE 397/498-013
Introduction to Mobile Robotics

Homework: Localization and the Extended Kalman Filter

Report Due Date: Tuesday, 13 Dec 05 submitted via Blackboard NLT 1045
(NO ANALOG SUBMISSIONS)

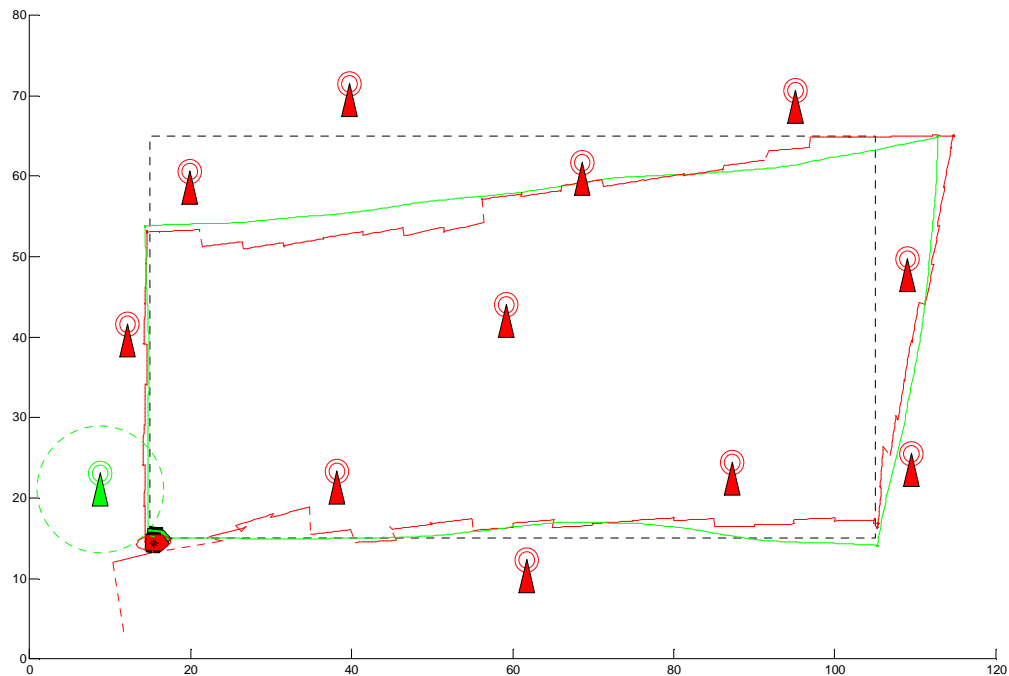


Figure 1: Sample EKF Simulation

A. **Objectives:** The goal of this assignment is to implement an EKF that will be used to localize a mobile robot traveling within an array of RF beacons.

B. **Robot Model:**

1. **Motion Model:** Again, we are using a differential drive kinematic model. However, we again will assume that there is uncertainty in the robot motion model.
 - a) Assume that both the left and right wheels are subjected to independent random noise. Values of 10-20% of the actual DISTANCE TRAVELED for the standard deviation are reasonable.

NOTE: Since the wheel will have separate noise inputs this will also result in errors accumulating in orientation.

- b) Assume a constant linear robot velocity of 5 m/s
- c) Assume the update rate for the odometry is 10Hz.

2. Sensor Model (See Skeleton .m file for more details):

- a) We are assuming that the robot receives distance measurements to beacons that are within range (max range is 20 meters).
- b) Further assume that the ranges are corrupted with Gaussian noise that is proportional to the actual range (default is $0.15 \cdot \text{actualRange}$).
- c) Beacons are created at a position taken from mouse input via the

```
b(i) = makeBeacon(x,y,20,'r');
```

- d) Beacons are tested to determine if they are in range via the

```
[range,b(i)] = queryBeacon(b(i),robot);
```

- e) Beacons that are not within the actual range of the robot are plotted as red.
- f) Beacons that are within range (active) are plotted as green with a circle reflecting the measured range (see far left beacon in Figure 1 above).
- g) Beacons are queried at a rate of 1 Hz.

C. Requirements:

1. You **MUST** use Matlab to complete this assignment.
2. This is an individual assignment. Each student is required to submit his/her own work in order to receive credit.
3. As stated on the web page, if you turn this assignment in late without coordinating with me first you will receive a 0.
4. **LOOK OVER THE LECTURE NOTES. THERE IS A LOT OF INFORMATION CONTAINED IN LECTURE 13 ESPECIALLY THAT IS DIRECTLY RELEVANT TO THIS HOMEWORK!**

D. The Assignment:

1. There is a skeleton .m file that everyone **MUST** use as the basis for their homeworks. Exceptions will not be accepted. Download this from the course web page.
2. Initial values for the sensor/motion model parameters are included in this .m file. You may modify these as you deem necessary.

3. The robot is initially placed at position 15, 15 with orientation = 0.
4. The robot's belief in its initial position is corrupted as reflected in the .m file.
5. The true robot is plotted as green (what the simulator knows).
6. The robot's belief (from the EKF) is plotted in red.
7. At all times in addition to the robot's belief, you must also plot an ellipse around the robot expected pose that reflects 3 standard deviations in positional uncertainty. You can obtain this from the covariance matrix P . **HINT: You can use the SVD function in Matlab to obtain a decomposition of the covariance matrix that will yield the ellipse's size as well as its orientation. DO NOT PLOT AN ORTHOGONAL ELLIPSE UNLESS IT IS ORTHOGONAL**
8. You will most likely have to write the PlotEllipse function yourself (let me know if you find a good one online).
9. During the measurement update rate, the robot will query each beacon to see if it is within range. Those responding will be used sequentially in the MeasurementUpdate phase. So, if three beacons are within range, there will be three calls to MeasurementUpdate.
10. Assume that a beacon out of range responds with a -1.
11. During each call to MeasurementUpdate, we not only calculate the new mean and covariance for the robot, we also replot the corresponding robot pose and uncertainty ellipse.
12. The time update reflect the motion portion of the EKF. During each step, you recalculate the robot's belief AND actual robot position. **NOTE: The actual robot's motion is corrupted by noise, but the robot's belief (robotHat) is NOT corrupted by noise (as the noise values are not known to the robot - only to the simulator).**
13. Take a look at the call to TimeUpdate. You will notice that there are 2 additional arguments - legV and legOmega. These correspond to the current robot linear and angular velocities, respectively.

```
[robot, robotHat, P] = TimeUpdate(robot, robotHat, P, legV, legOmega);
```

14. The goal of your robot is NOT to travel around aimlessly. Instead, it attempts to follow the rectangular path outlined in Figure 1. There are 8 "legs" to this path. They are:

- a) Travel from $(15, 15, 0)$ to $(105, 15, 0)$
- b) Travel from $(105, 15, 0)$ to $(105, 15, \pi/2)$
- c) Travel from $(105, 15, \pi/2)$ to $(105, 65, \pi/2)$
- d) Travel from $(105, 65, \pi/2)$ to $(105, 65, \pi)$
- e) Travel from $(105, 65, \pi)$ to $(15, 65, \pi)$
- f) Travel from $(15, 65, \pi)$ to $(15, 65, 3\pi/2)$
- g) Travel from $(15, 65, 3\pi/2)$ to $(15, 15, 3\pi/2)$
- h) Travel from $(15, 15, 3\pi/2)$ to $(15, 15, 0)$

15. At the conclusion of traveling these legs the robot would have returned to its initial position in an ideal world. From Figure 1, we see the world is not ideal.

- a) The robot travels each leg of the path open loop.
- b) Linear and angular velocities for each of the legs are $(5, 0)$ and $(0, \pi/4)$ depending on whether the robot is translating or rotating, respectively.
- c) Only when it is within some tolerance of the objective pose for each leg will it transition to the new leg.
- d) **NOTE: Transitions are based on the robot's BELIEF of its post - NOT the actual pose as this is unknown to the robot.**

16. You are to plot the paths that the actual robot takes, as well as the robot's belief. For the latter, transitions that occur during the measurement update phase are to be plotted as dashed lines, while time update motion is plotted as solid lines. This will clearly delineate the impact of the measurement update phase.

E. A write up, to include:

1. Images from simulation trials.
2. Derivations of ALL EKF equations that you used.
3. Your Matlab source code.
4. Answers to the following additional questions:
 - a) Your fellow student makes the observation that although the robot is running the path open loop, its performance seems better than that with open loop odometry. Explain why this is/is not the case.
 - b) The robot has no sensors to measure orientation, yet somehow it seems to be able to infer its orientation solely from range measurements which can only yield position information. Is this just luck, or is there a more subtle reason? Explain.

- c) How would you integrate a gyroscope into the above framework. Justify your answer.
1. Derive the new EKF equations for this EKF (including Jacobians).
- d) A student gets the lab's compass working and decides to integrate that into the original EKF. Rewrite the EKF equations to reflect this change in configuration.
- e) Plot the robot position error and covariance (product of major and minor ellipsoid axes) vs. time. Does this converge? Is this expected? Why/why not?
- f) Plot the orientation covariance over time. Do we ever see this decrease during the time update? If so, explain how this could occur?
- g) At times we see dramatic jumps in the expected robot's pose. When do these occur? Explain why these are expected?
- h) During the measurement update phase, there may be several beacon measurements available that are processed serially by the filter. Does the order in which the measurements are processed affect filter performance? Why or why not?