CALIFORNIA INSTITUTE OF TECHNOLOGY

ME/CS 132a, Winter 2011

Homework #3 Due: February 22nd, 2011

In this homework, you will design a localization algorithm for an autonomous underwater vehicle (AUV). Within the hull of the AUV is a transceiver which emits a "call" pulse into the water column. Beacons deployed at known locations on the seabed detect this pulse and reply with "acknowledge" pulses which are detected by the transceiver. The difference in time between "call" and "acknowledge" is proportional to the distance between vehicle and each beacon. Using these time measurements, the 3D position of the AUV relative to the beacons can be inferred. We assume that the distance moved between transmission (to the beacons) and reception of acoustic signals (from them) is small, so that these data measure the instantaneous distance between the robot and the beacons.

The localization problem will be formulated as a nonlinear optimization problem, and you will implement a gradient descent method to solve it. A Matlab file called `DoAUVLocalization.m` has been provided as a template; but you can implement your algorithm in your favorite language if you take the time to translate the skeleton Matlab code.

The main loop in this code simulates the vehicle moving trough a series of positions in the water above the beacons. Run the code to see this. The green points represent the actual robot position (`XvTrue`). The red cross represents the position estimate (`Xv`), which has some noise and so is not precisely at the true initial position. Your code will be able to update the position of `Xv` so that it will track `XvTrue`.

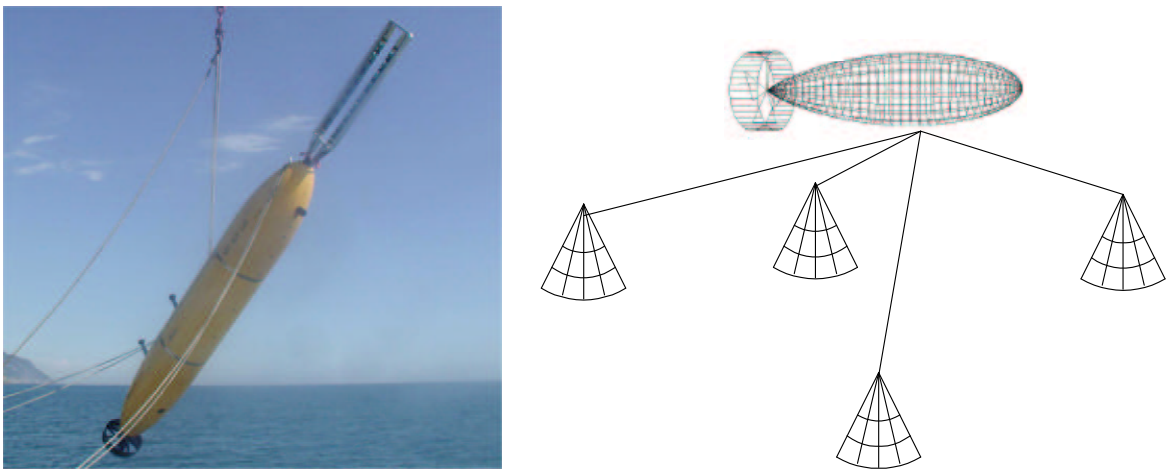

Figure 1: An autonomous underwater vehicle can localise relative to sonar beacons on the seabed.

1. [5 points] Complete the implementation for the function:

   ```
   z = ObsModelH(Xv, BeaconPosition, c)
   ```

   This should output the measurement that the sensor would produce, `z` (time-to-acknowledgement), given a particular 3D robot position, `Xv`, beacon position, `BeaconPosition`, and speed of sound, `c`. A value for the speed of sound is given in the example code.

   Add code to the main loop so that the vehicle can make observations for all beacons at each timestep. You should use the provided function `SimulateObservation`. This will use the ground truth observation model you just wrote to simulate observations that have been corrupted by some sensor noise.

   Plot the simulated observations as a function of time along the trajectory given in the code (put all 5 observations on the same plot).

2. [5 points] Derive an analytical expression for the Jacobian of the observation model with respect to the AUV position and implement this function in Matlab. To double-check that your code is correct, write a second function that computes the Jacobian using a first-order difference approximation. Compare the results of the two functions on the data and make sure that they are close enough if the difference interval is chosen appropriately.

3. [5 points] Formalize the localization problem as a nonlinear least-squares optimization problem. Write down the nonlinear objective function being minimized, and derive an expression for its gradient.

4. [15 points] Implement a solver for the optimization problem just defined. This is a function that takes as input: an initial estimate for the robot position, the beacons configuration, the current observation, a tolerance for the solution, and a maximum number of iterations; and whose output is the optimal AUV position. Implement the solver using gradient descent; or any other more complex method (e.g., Newton's method).

   Note: you have to implement this from scratch, and cannot use Matlab's optimization functions. You can use those functions to verify your results.

5. [5 points] Implement a simple localization method using the nonlinear optimization algorithm you just wrote. At each step, the estimate obtained in the previous step is used as the initial guess for optimization (the first step uses some *a priori* information on the position, as given in the code.)

   Reasonable parameters for your solver are: tolerance equal to 1cm, and maximum number of iterations equal to 40.

   Integrate this code in the main loop.

   Display three plots showing the estimated and true values of the three coordinates along the example trajectory.

6. [10 points] The uncertainty in the least squares estimate of the vehicle location, $\mathtt{x}_v$, is given by

   $$\mathtt{P} = \left\{ \Sigma_i \nabla \mathtt{H}_i^T \mathtt{R}^{-1} \nabla \mathtt{H}_i \right\}^{-1} \tag{1}$$

   where the i^{th} observation is given by $\mathtt{z}_i = \mathtt{h}_i(\mathtt{x}_v) + \mathtt{w}$ and the covariance of the noise, $\mathtt{w}$, is $\mathtt{R}$. $\mathtt{H}_i$ is the Jacobian of the function h w.r.t $\mathtt{x}_v$ for the i^{th} observation.

The uncertainty in AUV position can be plotted as a 3D ellipsoid centered on the estimated position. Alternatively, a 2D slice of this uncertainty can be plotted as an ellipse representing the covariance in that plane. The Matlab code provided includes functions for plotting 3D ellipsoids or 2D ellipses in the x-y plane.

Plot the x-y covariance as an ellipse centered on the x-y estimates of positions along the example trajectory.

7. [10 points] Study how the shape of the ellipse varies as a function of vehicle position and beacon geometry. Consider two cases:

   (a) Three co-linear beacons.
   Coordinates: `[-1 0 0], [0 0 0], [+1 0 0]`).

   (b) Three beacons arranged in an equilateral triangle.
   Coordinates: `[-1 0 0], [0` $\sqrt{3}$ `0], [+1 0 0]`).

   For each case, let the vehicle position range over a grid of positions. Produce a 3D plot with the z-axis proportional to the area enclosed in the 1σ bound for the estimate at each x-y location at a constant height above the seabed [1].

   Is the uncertainty always finite?

8. [5 points] Show how the covariance produced by Equation 1 can be obtained without the summation. (hint: Concatenate the observations and their associated matrices) The final form you should obtain is:

$$\mathtt{P} = \{\mathtt{FGK}\}^{-1} \tag{2}$$

Notes on extra credit problems

- "graduate level" and "undergraduate level" only indicate the difficulty level of the problems. Undergraduate students can attempt either one, or both.
- A partial solution/analysis which is *formal* and *clear* is scored much higher than a long tortuous answer without mathematical substance.

[1] Note that the volume of an ellipsoid is proportional to the product of its eigenvalues.

9. Localization with highly uncertain model [extra credit, undergraduate level]

The usual assumptions when dealing with localization problems is that the observation model is known more or less precisely, that the noise is small with respect to the measurements, that it has Gaussian distribution, and that it acts additively on the measurements. In that case, the uncertainty of the robot position's estimate can be approximately described with ellipsoids. However, if one of more of those assumptions does not hold, the shape of the resulting uncertainty looks very different.

Consider a robot moving on a plane which has available n "uncalibrated" beacons measurements $\boldsymbol{y} = \{y_i\}_{i=1}^n$ whose sensor model is described by

$$y_i = f(\|\boldsymbol{x} - \boldsymbol{b}_i\| + \epsilon_i), \qquad \text{for } i = 1, \ldots, n,$$

where:

- $\boldsymbol{x} \in \mathbb{R}^2$ is the robot position on the plane;
- $y_i \in \mathbb{R}$ is the measurement corresponding to the i-th beacon;
- $\boldsymbol{b}_i \in \mathbb{R}^2$ is the i-th beacon's position on the plane, which is a known quantity.
- $\epsilon_i \in [0, \epsilon_{\max}]$ is a noise term associated with the i-th measurement. The probability distribution of ϵ_i is uniform on the interval $[0, \epsilon_{\max}]$, and the noise for different measurements is independently distributed.
- $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a <u>completely unknown</u> function, except for the following two facts: $f(0) = 0$, and f is monotone ($f' > 0$). Examples of functions that respect these constraints are $f(d) = d$, $f(d) = 2d$, $f(d) = d^7$, $f(d) = \log(1 + d)$.

Your mission, should you choose to accept it, is to describe the shape of the resulting uncertainty given this sensor model. More formally, given a set of observations $\boldsymbol{y} = \{y_i\}_{i=1}^n$, define the subset $\mathcal{X}(\boldsymbol{y}) \subseteq \mathbb{R}^2$ as the part of the configuration space which is compatible with the observations $\boldsymbol{y}$:

$$\begin{aligned} \mathcal{X}(\boldsymbol{y}) &\triangleq \{\boldsymbol{x} \in \mathbb{R}^2 \mid p(\boldsymbol{x}|\boldsymbol{y}) \neq 0\} \\ &= \{\boldsymbol{x} \in \mathbb{R}^2 \mid \text{there exists } f \text{and } \{\epsilon_i\}_{i=1}^n \text{ such that } y_i = f(\|\boldsymbol{x} - \boldsymbol{b}_i\| + \epsilon_i) \quad \}. \end{aligned}$$

Describe the geometric shape of $\mathcal{X}(\boldsymbol{y})$ for $n \geq 0$ beacons in general position.

Notes:

- "describe" means "draw the shape" + "prove that it is that way".
- $n \geq 0$ means "for all possible n", and it is a good advice to start at $n = 0$ and work your way up until you can proceed by induction.
- The question asks you to consider a generic function f, and not just a particular one.
- "in general position" means that the beacons are random points in $\mathbb{R}^2$: no group of 3 is collinear, and there is no other "lucky" coincidence.
- This is a one-shot problem: the robot does not move, and receives only one set of n measurements. There is no other information on the position $\boldsymbol{x}$ other than the measurements $\boldsymbol{y}$.

10. Simultaneous Localization and Mapping with highly uncertain model

[extra credit, graduate level]

Suppose that we have the same sensor model as in the previous problem, but now we do not know the positions $\{\boldsymbol{b}_i\}_{i=1}^n$ of the beacons *a priori*, and we want to estimate them together with the robot pose. In the literature, this problem is called SLAM (Simultaneous Localization And Mapping). It is evident that, given only one set of measurements, there is really little one can say about the pose of robot and beacons. Therefore, we will consider the scenario in which the robot moves and collects multiple measurements.

Suppose the scenario is captured by the following model:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{v}_0, \\ y_i(t) &= f(\|\boldsymbol{x}(t) - \boldsymbol{b}_i\|), \qquad \text{for } i = 1, \dots, n, \end{aligned}$$

where:

- $\boldsymbol{x}(t) \in \mathbb{R}^2$ now varies in time; and $\boldsymbol{x}(0)$ is *unknown*.
- $\boldsymbol{v}_0 \in \mathbb{R}^2$ is an *unknown* but fixed velocity, such that $\|\boldsymbol{v}_0\| > 0$.
- $\boldsymbol{b}_i \in \mathbb{R}^2$ is now an *unknown* but fixed quantity (the beacons do not move).
- The measurements $y_i(t)$ now depend on time, and they are deterministic (we removed the noise ϵ_i).
- $f : \mathbb{R}^+ \to \mathbb{R}^+$ has the same definition and properties as before ($f(0) = 0$, $f' > 0$, but otherwise *unknown*).

Assumptions on the beacons geometry:

- The beacons are in general position.
- There are "enough" beacons — we are not interested in pathological cases such as $n = 1$.

Questions:

1. *(analysis)* The augmented state space for this problem is $\varphi = (\boldsymbol{x}(t), \boldsymbol{v}_0, \{\boldsymbol{b}_i\}_{i=1}^n, f)$; even if they do not change, $\boldsymbol{v}_0$, $\{\boldsymbol{b}_i\}_{i=1}^n$ and f are part of the state because they are not immediately observable. It is evident that φ is not entirely observable from the data; for example, the robot cannot estimate the direction of $\boldsymbol{v}_0$ in an external reference frame from only the relative observations.
   Determine exactly how much of φ can be observed from the available data.
2. (*synthesis*) Derive the equations describing a filter that can estimate the observable part of φ from the data.

General guidelines for problem sets involving software

- Send all the software required via email to <andrea@cds.caltech.edu>. It is most helpful if you package code, data, and answers in a single .zip or .tgz file.
- **NEW:** For the derivations part/discussion, you can either turn in a physical copy in class[2], or include the writeup in the zip file sent by email.
- **NEW:** Please separate code & commentary: do not write your discussion/derivation in the source files, but in a separate report file, clearly labeled as such.
- **NEW:** If you are using wordprocessing software like Microsoft Word, and especially if you are using formulas, include a PDF version as well, because there is no guarantee that the formatting will be conserved on the TA's computer. If your version of Word does not come with PDF export, you can use the computers in the library, which should have that capability.
- **NEW:** The only good method for writing professional math is using LaTeX. If you plan to stay in academia at least a couple of years more, it is definitely worth to learn it. Software like LyX [3] offer a "What You See Is What You Mean" interface to LaTeXand are a good compromise between power and usability.
- We expect your solutions to be in the spirit of "reproducible research"[4]. Include instructions/scripts that allow reproducing your experiments with relatively little effort. For example, include a script "main.m" that calls the other files.
- Your software is evaluated on clarity and elegance as well as correctness. Comment your code. Insert a comment for everything that is not immediately obvious; and do *not* comment what *is* obvious. Think whether you would still understand the code you write after locking it up in a drawer for a year.
- You will be given code examples in a few languages (Matlab, C++, Python), but you are free to use any language with which you are comfortable.
- You are encouraged to use professional libraries (such as OpenCV) for reading/writing files and analogous tasks. However, you cannot use functions which the homework implies you have to write yourself (e.g., you cannot use the Harris detector in OpenCV).
- You are not required to follow the implementation guidelines perfectly. If you know a better alternative for a certain step, it is OK to use it.
- You are responsible for the parameters you choose. If we give you a "reasonable" value for a parameter that does not appear to work, you should try other values.
- You cannot share code for homework or look at other people's code. You are free to discuss general ideas about the problem. Reading aloud your code does not count as discussion.
- The only way to get software right is to insert consistency checks that make sure that your assumptions during the computations are correct.

  [CS majors]: Numerical code is a perfect occasion to learn and start to use *unit tests*[5].

[2] Or in Andrea Censi's mailbox in Steele in case of late homework.
[3] `http://www.lyx.org/`
[4] `http://reproducibleresearch.net/`
[5] `http://en.wikipedia.org/wiki/Unit_testing`

- Read Dijkstra's words from his 1972 ACM Turing Lecture[6]:

  > *The competent programmer is fully aware of the strictly limited size of his own skull; therefore he approaches the programming task in full humility, and among other things he avoids clever tricks like the plague.*

  Be aware of the limited size of your skull, and split the code in smaller functions that are easier to get right.

  [CS majors]: Print out a picture of Dijkstra and hang it in your room; meditate daily on one of his EWDs[7]; when in doubt, ask yourself: *what would Dijkstra do?*

- Remember: in elegant code, bugs have no space to hide.

[6] `http://www.cs.utexas.edu/~EWD/ewd03xx/EWD340.PDF`
[7] `http://www.cs.utexas.edu/~EWD/`