Perception in Robotics
Term 3, 2020. PS3

Gonzalo Ferrer
Skoltech

February 28, 2020

This problem set is a single task comprising 10% of your course grade, and it is to be done individually. You are encouraged to talk at the conceptual level with other students, discuss the equations and even the results, but you may not show/share/copy any non-trivial code.

Submission Instructions

Your assignment must be received by 11:59p on Monday, March 9th. You are to upload your assignment directly to the Canvas website as two attachments:

1. A PDF with the written portion of your document, solving the tasks proposed below. Scanned versions of hand-written documents, converted to PDFs, are perfectly acceptable. No other formats (e.g., .doc) are acceptable. Your PDF file should adhere to the following naming convention: `alincoln_ps3.pdf`.
2. A `.tgz` or `.zip` file *containing a directory* named after your uniqname with the structure shown below.

```
alincoln_ps3.tgz:
alincoln_ps3/run.py
alincoln_ps3/field_map.py
...
alincoln_ps3/slam/* //new files created by you
alincoln_ps3/sam.{avi,mp4}
```

Homework received after 11:59p is considered late and will be penalized as per the course policy. The ultimate timestamp authority is the one assigned to your upload by Canvas.

SLAM problem set

Code

To begin, you will need to download `ps3-code.zip` from the Canvas course website. This `.zip` file contains some Python files. The project is a modified version of the localization simulator originally used in PS2. The number of landmarks is now configurable and multiple range/bearing/markerId tuples are produced at each time step. The odometry model remains the same as in PS2.

Below you will find descriptions of the files included in the zip file. You may end up not using every single function. Some are utilities for other files, and you don't really need to bother with them. Some have useful utilities, so you won't have to reinvent the wheel. Some have fuller descriptions in the files themselves.

Things to implement

- Include all those necessary calls on the `run.py` file to update and correct the SLAM problem. Plotting is also necessary.
- Implement the square root SAM with known correspondences and unknown.

Utilities (essentially similar from PS2)

- `README.md` – Some commands examples for installing the environment, testing and evaluating the task.
- `run.py` – Main routine, with multiple options, allowing you to solve the task with no need to modify the file.
- `field_map.py` – for plotting the map.
- `tools/task.py` – General utilities available to the filter and internal functions.
- **`tools/jacobian.py`** – Jacobians derived for 2d planar robot and landmark observations.
- `tools/data.py` – Routines for generating, loading and saving data.
- `tools/objects.py` – Data structures for the project.
- `tools/plot.py` – All utilities for plotting data.
- `slam/slamBase.py` – An abstract base class to implement the various SLAM algorithms.

Task 1: Prerequisites to build SAM with known DA (40 points)

In this task, you will implement the prerequisites for landmark-SAM. Your robot is driving around an environment obtaining observations to a number of landmarks. The position of these landmarks is not initially known, nor is the number of landmarks. For now, we'll simplify the problem: when the robot observes a landmark, you know which landmark it has observed (i.e., you have perfect data association).

For this problem, your landmark observations are $[range, bearing, markerId]$ tuples. Your robot state is $[x, y, \theta]$ (cm, cm, radians) and the motion control is $[\delta_{\text{rot1}}, \delta_{\text{trans}}, \delta_{\text{rot2}}]$ (radians, cm, radians).

You are encouraged to implement the equations in the simplest and most literal way possible: don't worry about computational or memory efficiency.

A. (10 pts) Write the Mahalanobis distance of the first order approximation for each of the three factors appearing in this problem: transition, observation and initial condition (anchor factor). For this task, you can write the equations as a function of the following terms (see L10.6-7):
   - G Transition function Jacobian (code found in `task/jacobian.py` for later use).
   - H Observation function Jacobian.
   - M Covariance of motion model in **action** space.
   - Q Observation Covariance.
   - Σ_0 Initial covariance corresponding to the prior distribution of x_0

B. (10 pts) Write the numerical value of the pre-multiplying matrices $\Sigma^{-T/2}$ at $t = 1$ for the observations and transition.

C. (20) Create a function that generates the adjacency matrix A for any given number of observations. Print the numerical value at $t = 1$.

Task 2: SAM evaluation (60 points)

Implement the functions for `samPrediction` and `samUpdate`. Here, you have freedom to create new files and modify existing ones (mainly `run.py`) if it is required. Your code should be easy to read and well-documented. In the document indicate which files you have created or modified.

A. (40 pts) At each time iteration, solve SAM (batch problem). You are allowed to use any method to solve the LSQ problem, such as direct pseudo-inverse, numpy LSQ routines, etc. Results will be evaluated with the video, see B.

B. (10) Include in the plot of the final map (only last instant of time) showing 3-σ error ellipses for the feature positions and also the true beacon locations. Generate a movie of your SLAM filter operating, with each frame showing the position and (x, y) 3-sigma uncertainty of the robot and all the landmarks.

   Note: Follow the example code provided to you in PS2 `run.py` for generation of the movie `python run.py -s -i slam-evaluation-input.npy -m sam`

C. (10 pts) Calculate and plot the Chi squared error at each instant of time. Comment on the results.

D. *Optional* (15 *extra* pts) Calculate a solution implementing either QR or Cholesky and back-substitution to solve the system of equations. Comment on the results.