69156 Simultaneous Localization and Mapping

Lab 1: SLAM for Karel in 1D

Problem: consider the robot Karel, that lives in 1D space. Karel can move forward and backwards in its 1D world. There are some beepers placed in Karel s world, that respond when Karel pings from within a certain range of distances. Karel is equipped with: (1) an odometer that provides a measurement of its own motion; (2) a sensor that detects beepers that respond to Karel pinging and can estimate the distance to said beepers.

Tasks:

1. Download the Matlab skeleton code **karel_lab1.m** from Moodle2. Run the program to see Karel carrying out simple odometry.
2. Complete the missing parts of the algorithm (functions **update_map** and **add_new_features**). In my solution, this amounts to 10 plus 10 lines of code (although admittedly, not obvious code).
   - *Tip 1:* You need to understand and use the maths of the Kalman filter equations.
   - *Tip 2:* Implement and test **add_new_features** first. Play with the resulting program to see how large a map can be made (in terms of the number of beepers) without you getting bored waiting for it to finish.
   - *Tip 3:* For matrix operations you do not need (and should avoid) the **for** instruction. It is especially slow in Matlab. Instead, use vector and matrix algebra operations.
3. Implement the Sequential Map Joining idea from Lesson 1 to build n sequential maps and join them together to obtain the full map. This amounts to another 20 lines of code in my solution.
   - *Tip 1:* You need to understand and use the maths of linear combinations of Gaussian vectors.
   - *Tip 2:* First try to do just two maps and then join them.
   - *Tip 3:* In this case you might find useful, and it is admissible, to use a single **for** instruction to iterate and generate the n sequential maps.
4. Analyze the respective computational costs of the full map .vs. sequential map joining.
   - *Tip 1:* Matlab can compute elapsed time, this is already in some parts of the skeleton code, learn how it works.
   - *Tip 2:* Learn to use *sparse* matrices in Matlab, they play a fundamental role in computational cost.
5. Have some more time in your hands and would like a better grade? Solve some of the proposed exercises and include them in your report. You can also improve your grade if you solve and implement exercise 11.

Results: do as much as you can and have time for. I expect that you should be able to complete up to task 4 (for a maximum grade of 8.0). Additional stuff can increase your grade. Write and submit a full, self-contained, well-detailed report (but with no unnecessary detail) by **Feb. 10**.