Particle Filter

Project #4

Introduction

In this assignment you will use the same data as the Nonlinear Kalman Filter assignment. In this assignment, instead of implementing a Kalman Filter you will implement a Particle Filter for the estimation. Refer to the introduction in the Nonlinear Kalman Filter on the input data and system models. We will be using the same 15 state model.

Task 1: Particle Filter Implementation

Your first task is to implement a particle filter to track the state of the quadrotor. The algorithm for a particle filter is readily implemented by using an object-style range-based `for` loop. However, such loops tend to be computation inefficient, particularly in Python. Therefore, I recommend using a vectorized approach and storing the particle list in an array and making use of linear algebra libraries (NumPy, Eigen, et cetera) to perform the actual computation. Again, despite this linear algebra-based implementation, I still strongly recommend not using MATLAB. If you choose to use a vectorized approach, be careful and understand how the library orders operations. Python is row-wise (ex: N x 15).

In the prediction step, for each particle you need to take the original particle, sample from the noise distribution, and use the measured inputs plus noise to determine the future state of that particle using the process model. You should use the noise covariance values you found in the Nonlinear Kalman Filter assignment as a starting point for the process noise.

In the update step, you need to use the observation model and the measurements to calculate its importance weight for each particle and then use the low variance resampling algorithm to find your updated particle set. Again, you should use the covariance values you found in the previous assignment as a starting point for the observation model.

Task 2: Navigation solution and particle count investigation

One notable drawback of the particle filter is the lack of a ready navigation solution. While a Kalman filter provides a mean and covariance, the particle filter has only the particle distribution. Therefore we must introduce some method of sampling the particle field to determine a singular position estimate. There are a couple ways of doing this, please examine the error from truth by taking the position estimate as: the highest weighted particle, the average over all particles, and the weighted average over all particles. Finally, compute the root-mean-square error over all particles with respect to truth to have a general gauge of how your implementation is performing. RMSE is the typical error characterization of a particle filter, however it is only useful when you have a ground truth estimate.

With the error and navigation solution metrics as defined above established, investigate the performance of the particle filter for a 250, 500, 750, 1000, 2000, 3000, 4000, and 5000 particles. Are there any trends to these results?

Task 3: Comparison to the nonlinear Kalman filter

Finally, please discuss in 2-3 paragraphs the results from this assignment as well as the previous Nonlinear Kalman Filter assignment. Some specific points to discuss:

- Ease of implementation
  - Which method was easier to write the code for? Why?
  - Which method was easier to tune parameters for?
- Speed of code
  - Which method runs faster?
  - Why might this be important?
- Accuracy of results: which method yielded more accurate tracking results?

Grading Rubric

Task 1	**Excellent – 5 pts** The student has correctly implemented a particle filter and through iteration arrived at accurate tuning parameters.	**Acceptable – 4 pts** The student has correctly implemented the particle filter including the resampling method but lacks sufficient tuning.	**Marginal – 2 pts** The student has attempted to implement a particle filter, but there are minor errors.	**Unacceptable – 0 pts** The student fails to implement a particle filter.
Task 2	**Excellent – 5 pts** The student has correctly implemented the error metrics and navigation solution methods and provided a detailed response to the question asked with plots.	**Acceptable – 4 pts** The student has correctly implemented the error metrics and navigation solution methods and provided a basic response.	**Marginal – 2 pts** The student has attempted to implement the error metrics and navigation solutions but there are some minor errors.	**Unacceptable – 0 pts** No answer was given, the response was lacking sufficient detail, or the answer was completely incorrect.
Task 3	**Excellent – 5 pts** The student has provided a response that sufficiently answers the questions and reflects on how that relates to their own implementation.	**Acceptable – 4 pts** The student has provided a response that sufficiently answers the questions.	**Marginal – 2 pts** The student has provided a minimal analysis and does not draw from their results.	**Unacceptable – 0 pts** No answer was given, the response was lacking sufficient detail, or the answer was completely incorrect.
Code Quality	**Excellent – 5 pts** The student has intuitive, concise, well written code that follows a logical and organized structure. Comments are provided and document the functionality of the code.	**Acceptable – 4 pts** The student's code makes sense given the context of the problem. Some comments or documentation is provided.	**Marginal – 2 pts** The student's code is poorly structured and not intuitive. Little to no comments or documentation is provided.	**Unacceptable – 0 pts** The student's code fails to run, runs incorrectly, or otherwise fails to address the problem.