Aengus Kennedy
2025 04 18
Tufts ME0134
Lab 4

Implementation Notes

- I wrote all the code in this submission from scratch without the starting templates. I reused the forward kinematics code I wrote in Lab 3.
- The Kalman filter is implemented in the sixth cell of the file lab_4/l4data_processing.ipynb accessible in my ZIP submission or online at https://github.com/aengusk/ME0134-Aengus-Kennedy/.

Experiment

The robot was commanded to follow the following sequence of straight trajectories and point turns.

- forward 2 meters (speed 150 cm/s)
- turn -1.5 rotations (speed 150 cm/s) (where negative rotations are in the clockwise direction)
- forward 5 meters (speed 150 cm/s)
- turn -2.5 rotations (speed 150 cm/s)
- forward 1.5 meters (speed 150 cm/s)
- turn -8 rotations (speed 100 cm/s)
- forward 4 meters (speed 100 cm/s)

Results

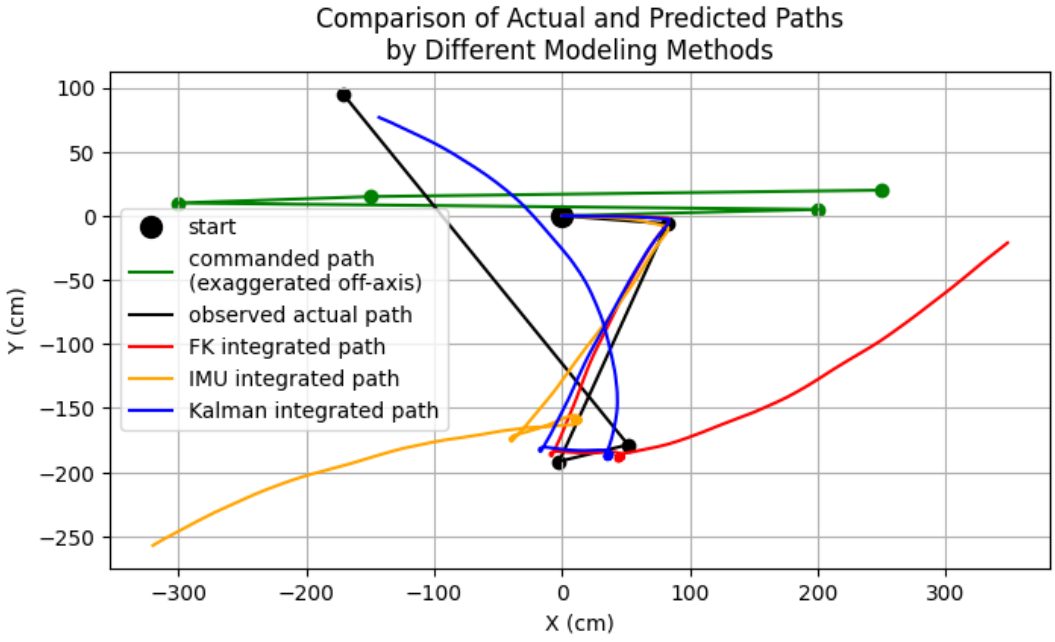

Fig. 1: Comparison of Actual and Predicted Paths
by Different Modeling Methods

Analysis

I tuned the Kalman filter's parameters by paying attention to how the **final yaw value** (corresponding to the angle of the final straight stretch of the robot's path) predicted by the model changed as I changed its parameters. The robot's true final yaw was about −5.67 rotations, right in between the value predicted by pure FK integration (−5.89 rotations, plotted in red) and pure IMU integration (−5.41, plotted in orange). The Kalman filter values of $P_\theta = 0.03$, $Q_\theta = 0.0001$, $R_\theta = 1.0$ best aligned with the true path (plotted in black) and predicted a final yaw of −5.56 (plotted in blue).

I chose a very small Q_θ value because I expect the IMU to accumulate error over time just as the forward kinematics model does. In most Kalman filters, the values in the matrix Q represent the increase in noise over time in the forward kinematics model, which should cause the Kalman filter to trust the forward kinematics model less and the sensor value more. In this case, though, our sensor data is itself a quantity integrated over time that will tend to accumulate error. Because of this, I thought that it would not be appropriate to decrease trust in the forward kinematics model and increase trust in the IMU data significantly over time, and because they will both accumulate error, the best course of action would be to trust them approximately equally over time as they both become less and less accurate. The small positive Q_θ value of 0.0001 comes from the fact that I am speculating that integrating encoder data will accumulate error slightly faster than integrating IMU data will, so trust should move very slightly toward the IMU over time.

In researching how to tune a Kalman filter when the sensor itself will accumulate error, I discovered that the Kalman filter model that we're drawing from actually assumes that sensor noise is additive (always appears as noise added to the true value), Gaussian (follows a Gaussian distribution when sampled over time) and zero-mean. The inaccuracies in our IMU data do not obey this paradigm because they are not zero-mean: the accumulation of integration error over time means that the difference between the IMU reading and the true yaw of the robot will tend to err more and more in the same direction as time goes on and the errors will not have a mean of zero over time. An arrangement whose noise I think would be zero-mean, additive, and Gaussian is if the robot were to determine its heading using a camera pointed straight up at a stationary AprilTag on the ceiling from whose apparent orientation from the camera's point of view the robot could infer its own orientation, or other arrangements like this where the robot's sensors are perceiving something concrete, not integrated over time.

Still, the Kalman filter had clear positive outcomes in this situation. The final position from the Kalman filter is 34 cm away from the ground truth final position, which is a much better prediction than using only encoders, which predicted a final location 2.12 meters away from the actual location. If the Kalman update step were disabled, integration drift would gradually accumulate in the model, and all future measurements would be decreasingly certain until the robot's location were rebased on a known waypoint. This experiment is an especially good illustration of the value of fusing sensor and model information because the final tail of the Kalman predicted path is positioned right between the tail of the purely FK integrated path on one side and the purely IMU integrated path on the other side. It was only possible to fuse these two predictions into a more accurate picture with an integration model like the Kalman filter.

The final rapid rotations in the robot's path were a moment in the trajectory when the Kalman filter could still keep track of the robot's general heading but the forward kinematics model could not. After the robot had finished rotating, it had in reality made about net +100° turn, but the kinematics model predicted about a net 0° turn. By factoring in the kinematics model, which predicted closer to a net 180° turn, the Kalman filter corrected the kinematics model's prediction to very close to the true value. Relative to

ground, the forward kinematics model predicted a θ of about 5°, the IMU predicted a θ of about 185°, and the Kalman filter predicted a θ of about 80°. The true value of θ was about 110°.

When the robot spins quickly or moves at high speeds, two idealized assumptions of the robot's motion might break down. First, the faster the robot is moving, the more encoder ticks are taking place between each sampling. Encoder information should usually be based on 10-100 ticks between each sampling, so it would not be ideal if the robot ended up driving so fast that more than 100 ticks were taking place each sample. Second, the faster the robot is going, the more likely its wheels are to slip on the ground, which would result in a change of position that the kinematics model would not know about. A slip would only be able to be detected by the IMU.

The biggest doubt I would place in my Kalman filter comes from the fact that I tuned its values while looking at their predictions trying to get the prediction to match the robot's ground truth path. It could be the case that now that it is tuned, this Kalman filter would work well on any path that this robot drives, but because more experimentaiton was not done, I cannot say for sure that the accuracy of this Kalman filter was not due to me manipulating it to be accurate for this particular trial.