Nader Ahmed
CS 469 HW1
Oct 9, 2023

**1. Design a motion model to estimate the positional (2-D location and heading) changes of a wheeled mobile robot in response to commanded speed (translational and rotational) controls. Clearly define the inputs, outputs and parameters of this model, and report the math. Explain your reasoning, as needed. Is this model linear in its inputs? Why or why not?**

The motion model I developed comes down to two situations:

If there is only a translational speed, $v$, or if the rotational speed, $w$, is sufficiently small, the robot will move straight in the direction of its current heading. This position can be determined by:

$$x_{new} = x + v_{hat} * \Delta t * cos(\theta)$$
$$y_{new} = y + v_{hat} * \Delta t * sin(\theta)$$
$$\theta_{new} = \theta$$

If there is both a translational and rotational speed the robot will instead move in a curved path. The radius of curvature can be determined by $R = v/w$, and new positions can be given by:

$$x_{new} = x - \frac{v_{hat}}{\omega_{hat}} * sin(\theta) + \frac{v_{hat}}{\omega_{hat}} * sin(\theta + \omega_{hat} * \Delta t)$$
$$y_{new} = y + \frac{v_{hat}}{\omega_{hat}} * cos(\theta) - \frac{v_{hat}}{\omega_{hat}} * cos(\theta + \omega_{hat} * \Delta t)$$
$$\theta_{new} = \theta + \omega_{hat} * \Delta t$$

The model is nonlinear; this can clearly be seen by how trigonometric functions are used to relate the inputs $v, w$ to the outputs $x_{new}, y_{new}, \theta_{new}$. This nonlinearity is incompatible with filtering methods that assume linear system dynamics, like the classic Kalman Filter. As a result, the use of an Unscented Kalman Filter (UKF) is necessary.

**2. Test your motion model on generating movement for the following sequence of commands. Report the values of any parameters. Report the resulting plot**

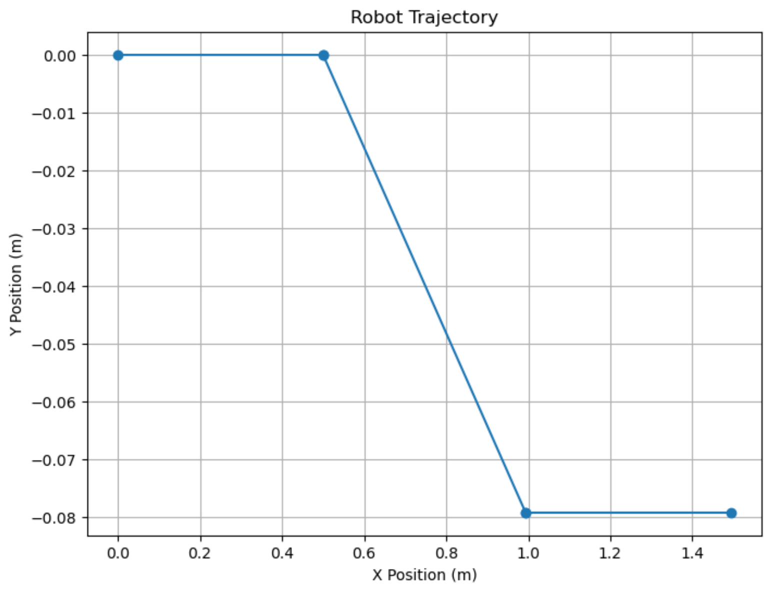


Figure 1: Motion model calculated trajectory given sample commands

**3. Test your simulated controller on the robot dataset. Issue the controls commands (in _Odometry.dat) to your controller, and compare this dead-reckoned path (i.e. controls propagation only) to the ground truth path. Report and discuss the resulting plot.**

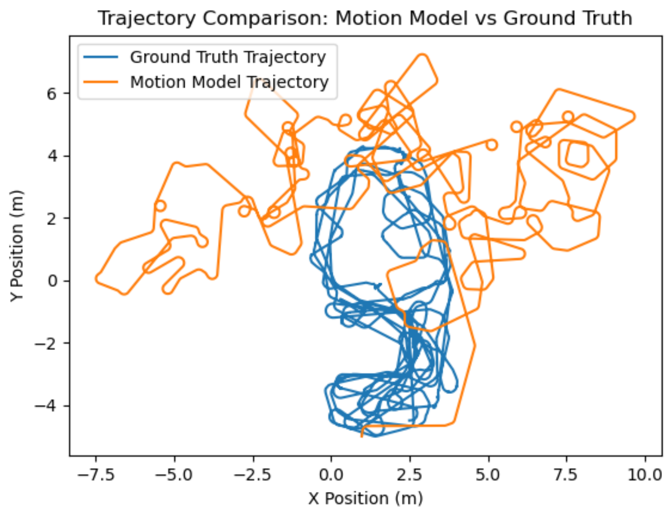


Figure 2: Comparison of the dead-reckoned trajectory with the ground truth

The ground truth trajectory (in blue) represents the actual path taken by the robot, while the dead reckoned-path (in orange) shows the predicted path based solely on the provided velocity commands. It appears that near the beginning, both trajectories align relatively close. As time progresses, however, there is an accumulation of error and the trajectories diverge wildly. The motion model implemented does not capture real-world disturbances such as wheel slippage, uneven terrain, and discrepancies between commanded and actual velocities. It is evident that external data (such as measurements) is necessary in order to accurately recover the ground truth trajectory.

**4. Describe, concisely, the operation of your assigned filtering algorithm. Include any key insights, assumptions, requirements. Use equations in your explanation.**

The Unscented Kalman Filter (UKF) is a variant of the Kalman Filter (KF) designed for systems with nonlinear dynamics. While the Extended Kalman Filter (EKF) would linearize the motion model using a Taylor Expansion, the UKF applies the transformations on a set of deterministically chosen sigma points and then recovers information about the Gaussian. Thus, it requires that the initial state be Gaussian and that the state, process noise, and measurement noise covariances be known. The steps can be summarized as follows:

1. Generate $2n + 1$ sigma points, where n is the dimensionality of the state, λ is a scaling parameter, and μ is the mean:

$$\begin{aligned} X^{[0]} &= \mu && \text{for i=0} \\ X^{[i]} &= \mu + (\sqrt{(n + \lambda) * \Sigma})_i && \text{for i=1...n} \\ X^{[i]} &= \mu - (\sqrt{(n + \lambda) * \Sigma})_{i-n} && \text{for i=n+1...2n} \end{aligned}$$

2. Prediction step: Transform each sigma point through the nonlinear motion model
   a. Compute the weighted sum of the transformed sigma points. Recover a mean and covariance. The weights should be given by:

$$\begin{aligned} w^{[0]}_m &= \frac{\lambda}{n+\lambda} \\ w^{[0]}_c &= w^{[0]}_m + (1 - \alpha^2 + \beta) \\ w^{[i]}_m &= w^{[i]}_c = \frac{1}{2(n+\lambda)} && \text{for i=1,...,2n} \end{aligned}$$

   where $w_m$ denotes weights for the mean and $w_c$ for the covariance.

   b. Store the result for the next iteration

3. Update Step: Re-generate sigma points around the mean and covariance
   a. Transform sigma points through the measurement model
   b. Compute the weighted sum of transformed measurements
   c. Compute the measurement covariance
   d. Compute the cross-covariance between state and measurements
   e. Compute the Kalman gain and use it to update the state estimate and covariance

Moreover, the UKF is more computationally expensive than the EKF given its propagation of sigma points (especially so for higher-dimensional systems). However, for systems with extremely nonlinear dynamics, it is more accurate in that it does not merely use a linearization around the mean of the current state Gaussian as the EKF does.

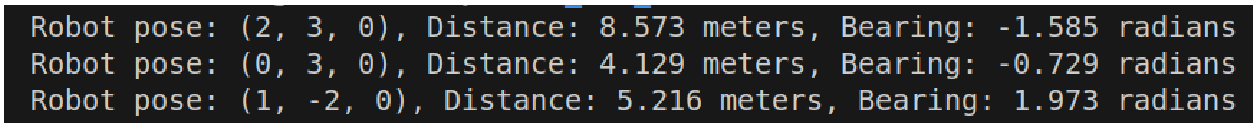
**5. Design a measurement model for your filter. Report all maths and reasoning; include images for illustration when appropriate.**

My measurement model takes the current pose of the robot $(x, y, \theta)$ , a landmark barcode number, and the ground truth position data of the landmarks as input. It associates the landmark barcode to its true position and calculates the distance between it and the robot. It then calculates the relative bearing, or the angle between the robot's orientation and the direction to that landmark.

A right triangle can be drawn with the hypotenuse being the distance from the robot to the landmark. Thus, it naturally follows that the distance and relative bearing be calculated as:

$$d = \sqrt{\Delta x^2 + \Delta y^2}$$
$$arctan(\frac{\Delta y}{\Delta x}) - \theta.$$

**6. Test your measurement model on predicting the range and heading of the sample landmarks when the robot is located at the associate positions. Report your results. (Remember, you have access to the ground truth landmark positions.**

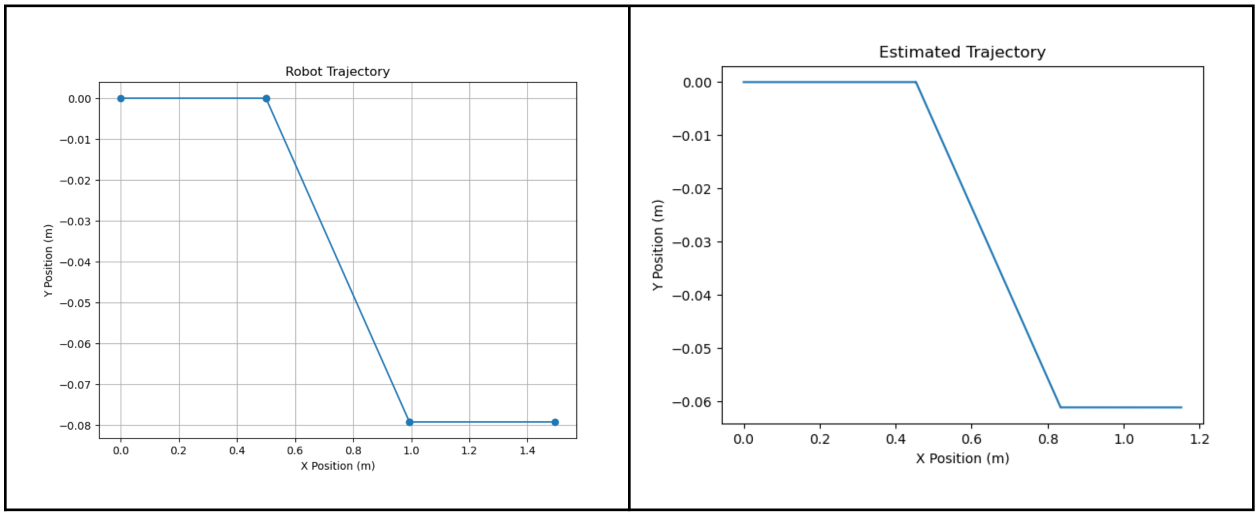


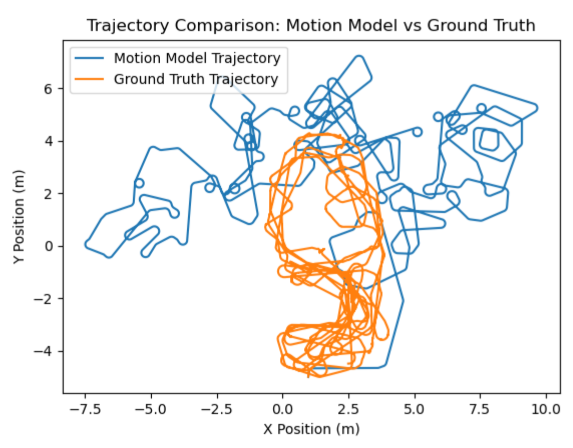
```
Robot pose: (2, 3, 0), Distance: 8.573 meters, Bearing: -1.585 radians
Robot pose: (0, 3, 0), Distance: 4.129 meters, Bearing: -0.729 radians
Robot pose: (1, -2, 0), Distance: 5.216 meters, Bearing: 1.973 radians
```

Figure 3: Calculated distance and bearing given robot pose and sample landmark location (6,13,17)

**7. Implement the full filter.**

**8. [*] Compare the performance of your motion model (i.e. dead reckoning) and your full filter on the sequence of commands from step 2, and also on the robot dataset (as in step 3). Report the results, explain any differences (what performs well? what performs poorly? under what conditions? and why?). Ground your explanations in the algorithm maths.**





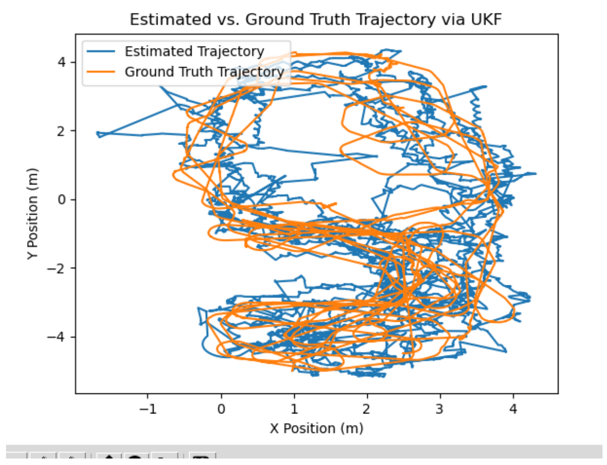


Figure 4: (top) left: sample dead-reckoned trajectory, right: sample predicted trajectory
(bottom) left: robot dead-reckoned trajectory , right: robot predicted trajectory

As seen in the top row, the dead-reckoned trajectory is identical before and after applying the UKF. This is due to the lack of any noise or discrepancies in the system. There are also no measurements to be applied (no update steps).

In the bottom row, we can see the drastic impact of the UKF in helping to closely track the ground truth trajectory. The accumulation of error that caused massive deviations in the purely dead-reckoned path is mostly gone. With that being said, however, there is still a very jagged path after applying the UKF. This is likely being caused by a multitude of things. For one, the real-world process and measurement noises (described by Q and R) are likely very different from what I have set. Additionally, with more time the parameters controlling the spread of the sigma points and their weights could have been tuned (α, β, $k$). I was also unsure of the initial covariance.

Looking at the recordings for the robot footage, it would seem that the measurements were unstable and would frequently drop out or overflow. This uncertainty could have been better modeled in my system with more knowledge about the equipment and data collection procedures.

**9. [*] Examine the effect (trends? limits?) of uncertainty in the algorithm (i.e. changing the noise parameters). Examine its performance at different parts of the execution (e.g. when missing an expected measurement reading). Report plots and/or tables, and provide explanations**

Changing various parameters has significant effects on the outcome of this program. Through qualitative testing, the following has been observed:

| | |
|---|---|
| Alpha: Determines the spread of sigma points around the mean | • Low α: UKF trajectory is less responsive to sudden changes<br>• High α: High level of divergence from the ground truth |

| | |
|---|---|
| Beta: Prior knowledge about state distribution | • $\beta = 2$ is optimal for Gaussian distributions |
| Kappa: Scaling parameter | • $\kappa = 0$ |
| Q: Process noise | • Underestimated Q: UKF trajectory quickly deviates since filter is overconfident in motion model<br>• Overestimated Q: UKF trajectory appears smoother but does not accurately follow the ground truth at all<br>• Finding an appropriate Q has been difficult |
| R: Measurement noise | • I have found that the bearing data has more noise than the range, so having a relatively larger value for the second element has helped track the ground truth<br>• Overestimating the measurement noise too much causes the trajectory to look increasingly like the dead-reckoned path |

With more time, I would have liked to implement more advanced testing algorithms to measure the error between the ground truth and predicted trajectory for given parameter values. I also sought to investigate different schemes for synchronizing control commands and observations. Initially, I performed multiple update steps in succession for timesteps where more than one landmark was visible. I then changed the design of my filter to perform prediction steps with zero vector control commands in between each measurement. I was unable to conclude how much this improved the performance of my filter.