Computational Robotics, Fall 2019

Lab 3 - Kalman Filtering

Andrew Choi, Viacheslav Inderiakin, Yuki Shirai, Yusuke Tanaka

November 2019

1 Introduction

The goal of this lab was to build a localization system for a two-wheeled differential drive non-holonomic robot. Using software constructed environments, the robot was able to accurately estimate its pose in the face of uncertainty. An Extended Kalman Filter was implemented for estimation. Numerical simulations were conducted under different conditions.

2 Mathematical Formulation

Before delving into the details of the experiment, the Kalman Filter (KF) and Extended Kalman Filter (EKF) are first discussed.

2.1 Kalman Filter

Probably the best studied technique for implementing Bayes filters is the KF. The KF is an optimal estimator, which infers parameters of interest from indirect, inaccurate, and uncertain observations. It is a recursive algorithm meaning that new measurements can be processed as they arrive. Limitations of the KF include being unable to handle non-Gaussian noise and nonlinear dynamics. In other words, the KF is the best linear estimator with Gaussian noise.

The KF consists of two steps: prediction step and correction step. At the prediction step, the KF infers future belief using the motion model of the system. At the correction step, the KF modifies the estimated belief making use of the observation model. Here, these two steps of the KF are explained using equations.

2.1.1 Prediction step

Let's consider a linear dynamic system as follows:

$$x_{t+1} = Fx_t + Bu_t + w_t \tag{1}$$

where x_t is the state at time t, u_t is the control input at time t, and w_t is the independent zero-mean Gaussian noise with $\mathrm{E}\left[w_t w_t^{\mathrm{T}}\right] = Q > 0$

From Equation 1, the mean and covariance are predicted as follows:

$$\begin{aligned} \bar{x}_{t+1^-} &= \mathrm{E}\left[F\hat{x}_t + Bu_t + w_t\right] \\ &= F\mathrm{E}\left[\hat{x}_t\right] + B\mathrm{E}\left[\hat{u}_t\right] + \mathrm{E}\left[w_t\right] \\ &= F\bar{x}_t + Bu_t \end{aligned} \tag{2}$$

$$\begin{aligned} \bar{\Sigma}_{t+1^-} &= \mathrm{E}\left[\left(x_{t+1} - \bar{x}_{t+1}\right)\left(x_{t+1} - \bar{x}_{t+1}\right)^{\mathrm{T}}\right] \\ &= \mathrm{E}\left[\left(F\tilde{x}_t + Bu_t + w_t\right)\left(F\tilde{x}_t + Bu_t + w_t\right)^{\mathrm{T}}\right] \\ &= \mathrm{E}\left[F\tilde{x}_t\tilde{x}_t^{\mathrm{T}}F^{\mathrm{T}}\right] + \mathrm{E}\left[w_t w_t^{\mathrm{T}}\right] \\ &= F\Sigma_t F^{\mathrm{T}} + Q \end{aligned} \tag{3}$$

2.1.2 Correction step

Let's consider linear observation model as follows:

$$y_t = Hx_t + v_t \tag{4}$$

where y_t is the observed output at time t, v_t is the independent zero-mean Gaussian noise with $\mathrm{E}\left[v_t v_t^\top\right] = R > 0$.

Here, y_t under the condition on $\hat{x}_{t^-}$ is expressed as follows:

$$y_t \left|\hat{x}_{t^-} = Hx_t\right| \hat{x}_{t^-} + v_t \left|\hat{x}_{t^-} = Hx_t\right| \hat{x}_{t^-} + v_t \tag{5}$$

Then, by defining $\hat{x}_{t^+} = x_t \left|y_t = (x_t|\hat{x}_{t^-})\right| (y_t|\hat{x}_{t^-})$, the updated estimated states are as follows:

$$\begin{aligned} \bar{x}_{t^+} &= \bar{x}_{t^-} + \Sigma_{t^-} H^\top \left(H\Sigma_{t^-} H^\top + R\right)^{-1} (y_t - H\bar{x}_{t-}) \\ \bar{\Sigma}_{t^+} &= \bar{\Sigma}_{t^-} - \bar{\Sigma}_{t^-} H^\top \left(H\Sigma_{t^-} H^\top + R\right)^{-1} H\bar{\Sigma}_{t^-} \end{aligned} \tag{6}$$

or, Equation 6 is also expressed as follows:

$$\begin{aligned} \bar{x}_{t^+} &= \bar{x}_{t^-} + K_t \left(y_t - H\bar{x}_{t^-}\right) \\ \bar{\Sigma}_{t^+} &= \left(I - K_t H\right) \bar{\Sigma}_{t^-} \end{aligned} \tag{7}$$

2.2 Algorithm of the Kalman Filter

By combining the prediction and correction steps, the psuedocode for the KF is constructed and shown.

```
Algorithm 1: Kalman Filter
input   : $\bar{x}_t$, $y_t$, $\bar{\Sigma}_t$
output  : $\bar{x}_{t+1^+}$, $\bar{\Sigma}_{t+1^+}$

  Prediction:
  $\bar{x}_{t+1^-} = F\bar{x}_t + Bu_t$
  $\bar{\Sigma}_{t+1^-} = F\bar{\Sigma}_t F^\mathrm{T} + Q$
  Correction:
  $\bar{x}_{t+1^+} = \bar{x}_{t+1^-} + \bar{\Sigma}_{t+1^-} H^\top \left(H\bar{\Sigma}_{t+1^-} H^\top + R\right)^{-1} (y_t - H\bar{x}_{t+1-})$
  $\bar{\Sigma}_{t+1^+} = \bar{\Sigma}_{t+1^-} - \bar{\Sigma}_{t+1^-} H^\top \left(H\Sigma_{t+1^-} H^\top + R\right)^{-1} H\bar{\Sigma}_{t+1^-}$
```

2.3 Extended Kalman Filter

The Extended Kalman Filter (EKF) is the nonlinear version of the Kalman Filter. The EKF is able to deal with nonlinear dynamics of a model by linearizing the model.

Let's consider the nonlinear stochastic model as follows:

$$\begin{aligned} x_{t+1} &= f\left(x_t, u_t, w_t\right) \\ y_t &= h\left(x_t, v_t\right) \end{aligned} \tag{8}$$

Here, by linearizing the Equation 8, the equations are shown as follows:

$$\begin{gathered} \tilde{x}_{t+1} \approx F_t \tilde{x}_t + W_t w_t \\ \left[F_t\right]_{ij} = \tfrac{\partial f_i}{\partial x_j}\left(x_t, u_t, 0\right), \quad \left[W_t\right]_{ij} = \tfrac{\partial f_i}{\partial w_j}\left(x_t, u_t, 0\right) \end{gathered} \tag{9}$$

$$\begin{aligned} &\tilde{y}_t \approx H_t \tilde{x}_t + V_t v_t \\ &\left[H_t\right]_{ij} = \frac{\partial h_i}{\partial x_j}\left(x_t, 0\right), \left[V_t\right]_{ij} = \frac{\partial h_i}{\partial v_j}\left(x_t, 0\right) \end{aligned} \tag{10}$$

Then, the prediction step in the EKF is as follows:

$$\begin{aligned} \bar{x}_{t+1^-} &= f\left(\bar{x}_t, u_t, 0\right) \\ \bar{\Sigma}_{t+1^-} &= F_t \Sigma_t F_t^\top + W_t Q W_t^\top \end{aligned} \tag{11}$$

The correction step is as follows:

$$\begin{aligned}\bar{x}_{t+1^+} &= \bar{x}_{t+1^-} + \bar{\Sigma}_{t+1^-} H^\top \left(H\bar{\Sigma}_{t+1^-} - H^\top + R\right)^{-1} (y_t - h(\bar{x}_{t+1-}, 0)) \\ \bar{\Sigma}_{t+1^+} &= \bar{\Sigma}_{t+1^-} - \bar{\Sigma}_{t+1^-} H^\top \left(H\bar{\Sigma}_{t+1} - H^\top + R\right)^{-1} H\bar{\Sigma}_{t+1^-}\end{aligned} \tag{12}$$

Following this, the psuedocode for the EKF is shown below.

Algorithm 2: Extended Kalman Filter

```
input   : $\bar{x}_t$, $y_t$, $\bar{\Sigma}_t$
output  : $\bar{x}_{t+1^+}$, $\bar{\Sigma}_{t+1^+}$

 Prediction:
$\bar{x}_{t+1^-} = f(\bar{x}_t, u_t, 0)$
$\bar{\Sigma}_{t+1^-} = F_t \Sigma_t F_t^\top + W_t Q W_t^\top$
 Correction:
$\bar{x}_{t+1^+} = \bar{x}_{t+1^-} + \bar{\Sigma}_{t+1^-} H^\top \left(H\bar{\Sigma}_{t+1^-} - H^\top + R\right)^{-1} (y_t - h(\bar{x}_{t+1-}, 0))$
$\bar{\Sigma}_{t+1^+} = \bar{\Sigma}_{t+1^-} - \bar{\Sigma}_{t+1^-} H^\top \left(H\bar{\Sigma}_{t+1} - H^\top + R\right)^{-1} H\bar{\Sigma}_{t+1^-}$
```

2.4 Motion Model

A two-wheeled robot that employs differential drive simply describes a robot whose wheels are never steered. Instead, to invoke rotation, the angular wheel velocities are varied. Following this, by using the center point between the wheels as the reference point, the following transition equations can be constructed

$$\dot{x} = \frac{r}{2}(u_l + u_r)cos\theta \tag{13}$$

$$\dot{y} = \frac{r}{2}(u_l + u_r)sin\theta \tag{14}$$

$$\dot{\theta} = \frac{r}{L}(u_r - u_l) \tag{15}$$

where r is the wheel radius, u_r is the right wheel angular velocity (rad/s), u_l is the left wheel angular velocity (rad/s), L is the distance between the two wheels, and θ is the angular position of the robot. From this, it can be inferred that when the angular wheel velocities are equal, pure translation occurs while when the angular wheel velocities are at opposite velocities, pure rotation occurs as shown in Fig. 1 [1].

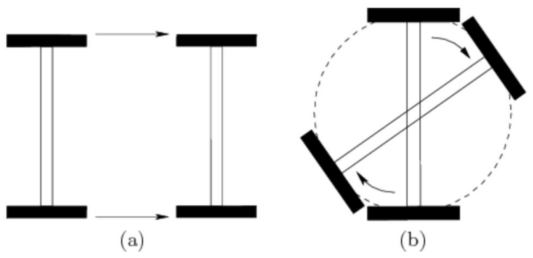

Figure 1: Pure translation and rotation in differential drive kinematics [1]

The above equations does not take into account uncertainty. In addition, the purpose of this work is to estimate the pose of the robot, so the true equation in consideration of uncertainty is now described as follows:

$$x_{t+1} = x_t + \frac{r}{2}(u_{l,t} + u_{r,t})sin\theta_t \Delta t + w_{x,t} \tag{16}$$

$$y_{t+1} = y_t + \frac{r}{2}(u_{l,t} + u_{r,t})cos\theta_t \Delta t + w_{y,t} \tag{17}$$

$$\theta_{t+1} = \theta_t + \frac{r}{L}(u_r - u_l)\Delta t + w_{\theta,t} \tag{18}$$

2.5 Observation Model

In this lab, the robot is equipped with two laser range sensors and one IMU, so that the observation equation are as follows:

$$\begin{bmatrix} \phi_t \\ \psi_t \\ r_t \\ \omega_t \end{bmatrix} = \begin{bmatrix} \sqrt{(x_t - \lambda_{f,x})^2 + (y_t - \lambda_{f,y})^2} \\ \sqrt{(x_t - \lambda_{r,x})^2 + (y_t - \lambda_{r,y})^2} \\ \tan^{-1}\left(\frac{y_t - \lambda_{f,x}}{x_t - \lambda_{f,y}}\right) - \theta_t \\ \frac{r}{L}(u_{r,t} - u_{l,t}) \end{bmatrix} + \begin{bmatrix} v_{\phi,t} \\ v_{\psi,t} \\ v_{r,t} \\ v_{\omega,t} \end{bmatrix} \tag{19}$$

where $\lambda_{f,x}$ and $\lambda_{f,y}$ represents the x and y element of a position of a landmark with a front laser sensor, and $\lambda_{r,x}$ and $\lambda_{r,y}$ represents the x and y element of a position of a landmark with a right laser sensor, respectively. The first two columns show a distance from a current position of a robot to observed landmarks. The third and fourth column are the angular pose and velocity of the robot respectively.

3 Experimental Setup

3.1 Robot Model and Assumptions

In the Extended Kalman Filter experiment, a two-wheeled differential drive robot is powered by bi-directional continuous rotation servos, part number FS90R as shown in Fig. 2. The wheel radius, r, and the separation distance between two wheel, L, are 50 mm and 90 mm, respectively. Each motor is signaled by PWM which controls the angular velocity from -60 to 60 RPM with a standard deviation of 5% of the max motor speed when considering slipping and motor internal friction.

The robot was given two different trajectories: straight and steered. In the straight path, the PWM on both servos are held constant with the same value over the simulation time. In the steered situation, one wheel is kept slower than the other to cause a turn. After half of the simulation time, the speed ratio u_r/u_l was flipped with the absolute velocity increased. Due to uncertainty in the motors, the robot was subject to undesirable drifting. As the control inputs were intensified, the robot pose uncertainty increased as well.

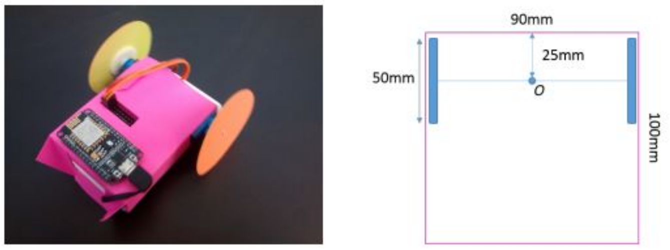

Figure 2: Robot Dimensions

3.2 Sensor Model and Assumptions

The robot is equipped with two laser range sensors, part number VL53L0X, and one inertial measurement unit (IMU), part number MPU-9250. One laser is projected toward the front of the robot, whereas the other is perpendicular and pointed to the right in a 750 mm by 500 mm rectangular open environment with 4 boundary walls. The IMU returns the robot's absolute orientation with respect to the north pole and angular rate. All measurements are functions of the robot state with noise.

3.3 Kalman Filter Implementation

In the simulation, the Extended Kalman Filter (EKF) is implemented over an Unscented Kalman Filter (UKF). EKF is an Kalman Filter which can handle non-linear model through linearization by computing the Jacobians of the system model as discussed in the mathematical formulation section. UKF is another method that deals with non-linearity by applying an unscented transformation (UT) which translates the original functions into a probability density function via sampling.

UKF has certain benefits; for instance, it can avoid calculating the Jacobian which could be computationally expensive and requires the system model to be differentiable. The UKF estimate could be closer to the true mean values typically if the estimate model and/or the observation model are highly non-linear. Nonetheless, due to the relatively simple system model, calculating the Jacobian of the robot model was seen to be more efficient than performing UT which requires sampling. The difference between EKF and KF is minimal and EKF only requires additional Jacobian calculation, while UT implementation is necessary for UKF, which raises complexity of the program with little or no improvements in this scenario.

3.4 Evaluation Metrics and Simulation Conditions

The implemented EKF was analyzed using three metrics: time, prediction error, and update error. Computation time for EKF completion was measured to evaluate its performance under different trajectories and different observation frequencies. Prediction and update errors were calculated using the mean squared error between prediction states and update states with the respective true state at each time step t. Covariance after a prediction step should tend to be higher than after a correction (observation) step due to higher uncertainty in motion. Consequently, prediction error will be greater than the update error for a certain time step unless the sensor input is highly unreliable. Lastly, the covariance of the state after a correction step should be lower than the prior prediction step due to the integration of sensor information.

4 Experimental Results and Analysis

To test the implemented Extended Kalman Filter, two different trajectories were tested with varying update rates. Furthermore, both scenarios of known and unknown initial state are conducted to the trajectories which are listed as follow:

1. Straight Trajectory: start at $(x = 100, y = 100, \theta = 60\,\text{deg})$, $u = [2, 2]$ (rad/s) for 10 seconds
2. Curve Trajectory: start at $(x = 250, y = 250, \theta = 60\,\text{deg})$, $u = [1.5, 0.2] \rightarrow [0.8, 4.2]$ for 5 seconds each

4.1 Straight Trajectory Analysis

For the straight trajectory scenario, the robot was fed ten identical control inputs of $w_r = 2, w_l = 2$ rad/s over 10 seconds. Although this would technically translate to a perfectly straight trajectory, the robot actually drifts due to the uncertainty of the motors. Since the standard deviation of the motors is 5% of the max motor speed (6.3 rad/s) and the robot is only outputting a rotational velocity of 2 rad/s, this drift is highly visible as shown below in Fig. 4.

In addition to the plotting of the robot pose, Fig. 3 also displays the robot uncertainty produced control and sensor noise before an observation (*gray ellipse*) and after an observation (*green ellipse*) Due to the constant availability of landmarks (*wall borders*), positional uncertainties quickly converge to their minimum as dictated by the motor and sensor noise. Furthermore, in both plots, the true state of the robot is plotted as blue dots connected by a black line while the concurrent predicted/estimated state is plotted as red dots connected by a red line. The estimated state after the observation can be thought of as the center of the green ellipse. Both ellipses show a range of six standard deviations of position uncertainty. The angular pose can be derived from the slope of the connecting lines. Two trials, one with insignificant stochasticity and the other with significant stochasticty are analyzed.

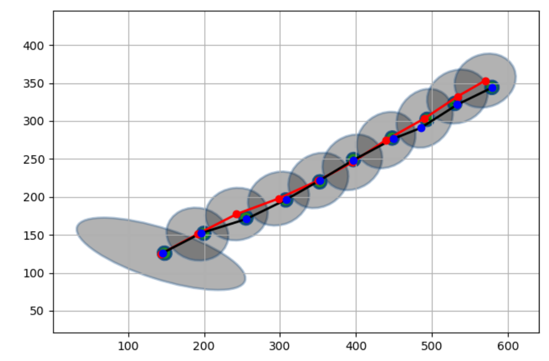
Figure 3: Clean Straight Trajectory Estimation

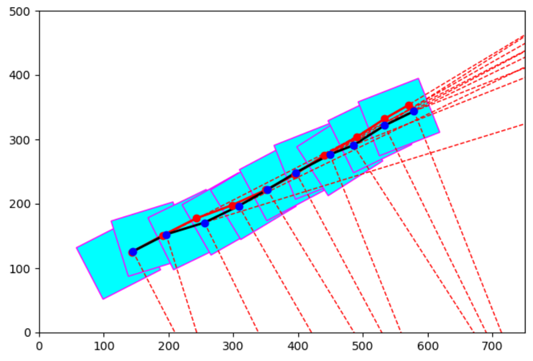
Figure 4: Clean Straight Trajectory Robot Pose

First, a trial with insignificant motor noise is shown. As seen from Fig. 4, even with the robot drifting off course, the trajectory can be seen to be roughly linear. Due to this favorable stochasticity, the EKF was able to accurately and efficiently estimate the state of the robot as shown in Fig. 3 where the estimated and true trajectories can be seen to be almost identical. This run produced the following prediction and update errors and computation time.

$$\text{Computation Time} = 107ms,\ \text{Prediction Error} = 157.069,\ \text{Update Error} = 93.720$$

Although the robot moved fairly straight in the above trial, in most cases, the robot trajectory is bound to suffer from noticeable drift due to the relatively high motor variance. To showcase such a trial, a trajectory is shown that was produced by the same control inputs as the one prior.

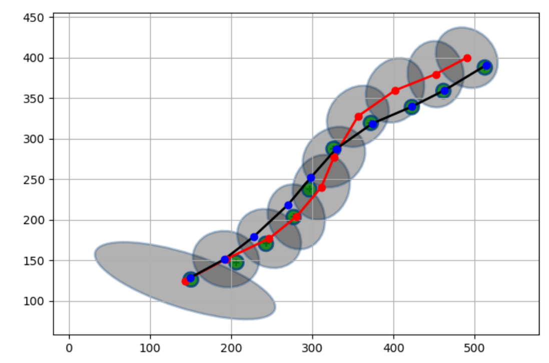
Figure 5: Noisy Straight Trajectory Estimation

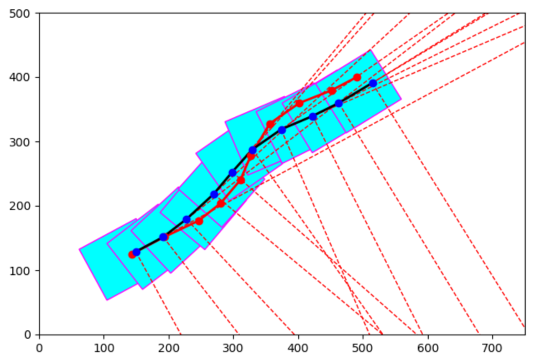
Figure 6: Noisy Straight Trajectory Robot Pose

Now, very notable drift can be seen in Fig. 6 as the robot starts to turn left and then back right. Consequently, the estimated and true trajectories start to differ as well as shown in Fig. 5. Notice that the largest deviations result during time steps when the motor noise is highest. In the third recorded state from the starting state, high motor noise causes the robot to take a sudden left turn. On the other hand, the EKF predicts that the robot will go straight as evident by the red node that is displaced by the previous trajectory slope. A deviation can be seen up until about the halfway point where the estimated state converges back to the true state by integrating information obtained by sensors into the filter. As motor noise causes the robot to now start to turn right, another mirrored deviation can be seen.

$$\text{Computation Time} = 104ms,\ \text{Prediction Error} = 673.901,\ \text{Update Error} = 423.634$$

Due to the error in predictions, the prediction error and update error have both increased roughly by a magnitude of 2.5 compared to the clean trajectory shown before. Even with this increase in error, the EKF still does a fairly good job of adjusting its state estimation towards the true state in the face of uncertainty. In addition to this, computation time seems to be unaffected which makes sense as the number of scans performed has remained constant. In the next section, a more complex trajectory is analyzed. The trade-off between computational efficiency and estimation accuracy is also discussed.

4.2 Curve Trajectory Analysis

For the curve trajectory, a control input of $w_l = 1.5$ and $w_r = 0.2$ rad/s is used for the first five seconds followed by a control input of $w_l = 0.5$ and $w_r = 4.2$ rad/s for the remaining five seconds. This trajectory produces a more complex trajectory for the EKF to estimate due to the change in wheel velocities halfway through. Like before, a trial with a relatively clean trajectory and another with a very noisy trajectory are shown and analyzed.

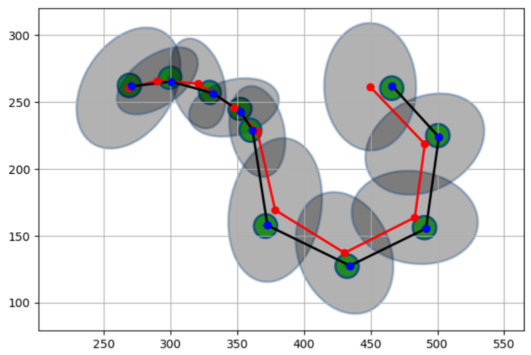

Figure 7: Clean Curve Trajectory Estimation

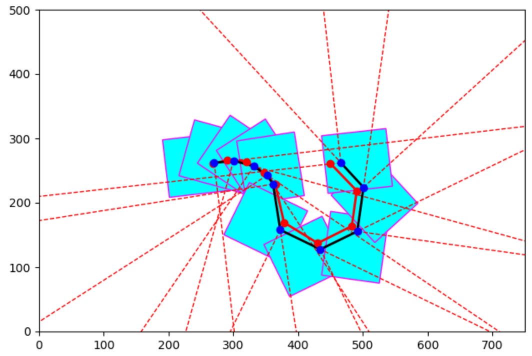

Figure 8: Clean Curve Trajectory Robot Pose

Much like before, when motor noise is low, excellent prediction can be seen from the EKF. Furthermore, due to the change in wheel velocities, an increase in the observation uncertainty (*gray ellipse*) can be seen from when the change occurs. Regardless, the EKF successfully adapts to the changed control inputs.

$$\text{Computation Time} = 107ms,\ \text{Prediction Error} = 176.143,\ \text{Update Error} = 64.889$$

The prediction and update errors are also on par with the clean straight trajectory regardless of the complex trajectory. Computation remains constant which indicates that it is entirely dependent on the number of scans, not the complexity of the trajectory. Although the EKF shows excellent results for when motor noise is minimal, this does not hold true for when the motor noise is considerably high as shown in the next figures.

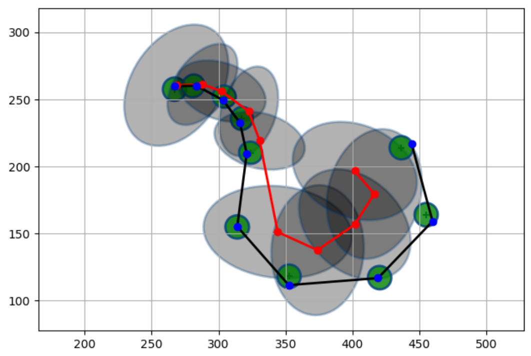

Figure 9: Noisy Curve Trajectory Estimation

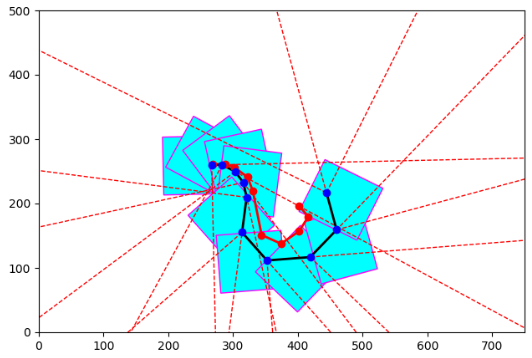

Figure 10: Noisy Curve Trajectory Robot Pose

A noisy run that causes the robot to continuously turn with a varying circumference results in the following plots shown above. Due to the consistent motor noise, the EKF estimates a relatively incorrect trajectory.

$$\text{Computation Time} = 109ms,\ \text{Prediction Error} = 1953.562,\ \text{Update Error} = 1126.625$$

The prediction and update errors are many magnitudes higher for the noisy condition versus the clean condition. For all the experiments performed thus far, a single control input and scan were performed every second. Sparsity of state estimation updates can lead to such inaccurate predictions as shown in Fig. 9. To combat such occurrences, it is possible to increase the scan rate of the EKF.

4.3 Increasing Scan Time: Computational Efficiency vs. Estimation Accuracy

To combat scenarios in which high motor noise and sparse scanning result in inaccurate estimations, the EKF can be performed more frequently. In the below example, the EKF is scanned every tenth of a second leading a 10x higher scan rate. Control inputs were also reapplied every tenth of a second as well.

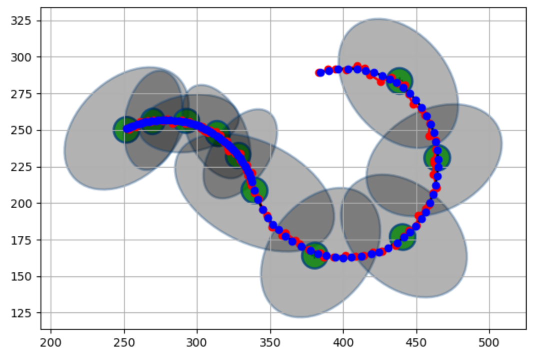

Figure 11: High Scan Curve Trajectory Estimation

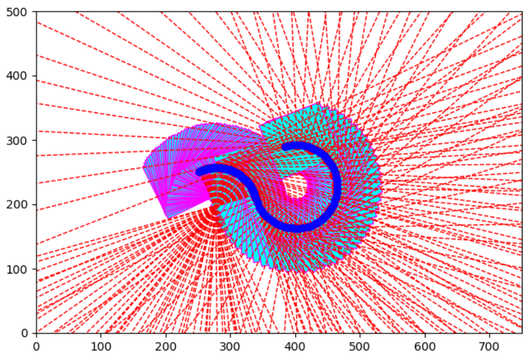

Figure 12: High Scan Curve Trajectory Robot Pose

$$\text{Computation Time} = 542ms,\ \text{Prediction Error} = 21.410\ \text{Update Error} = 21.105$$

It is here that we start to see the trade-off between computational efficiency and estimation accuracy that comes with a higher scan rate. Although the prediction and update errors have both drastically decreased, the computation time has increased by about a factor of 5.5. Due to this trade-off, careful tuning must be performed depending on what the specific system requirements.

4.4 Unknown vs. Known Location

All of the above examples have assumed a known initial pose. Now, both the linear and curve trajectories are observed for when the initial position is unknown.

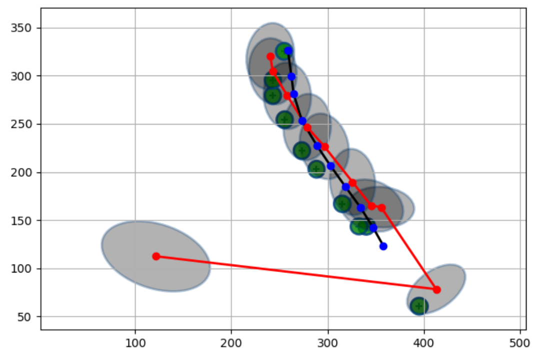

Figure 13: Unknown Straight Trajectory Estimation

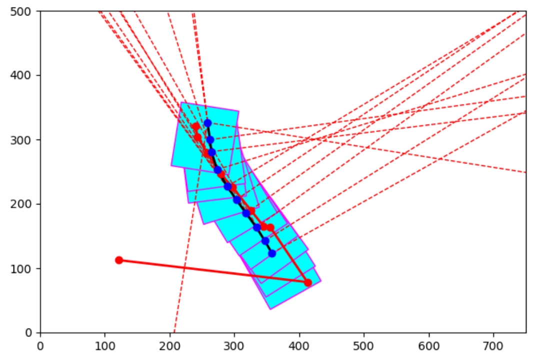

Figure 14: Unknown Straight Trajectory Robot Pose

To start off, the initial robot state is set to $(x = 100, y = 100, \theta = 60°)$. The true robot pose is in actuality randomized. With this randomization, it can be seen that after the first sensor measurement, the EKF significantly updates the estimated state towards the right of the map closer to the true state. After the second sensor measurement, the estimated state further approaches the true state after which the estimated states are roughly the true state.

$$\text{Computation Time} = 106ms,\ \text{Prediction Error} = 107120.124,\ \text{Update Error} = 105172.92$$

Due to the unknown starting location, a very high prediction and update error are present. This is due particularly because of the first two predicted states. It can be assumed that if further EKF scans were performed, these errors would start decrease due to averaging.

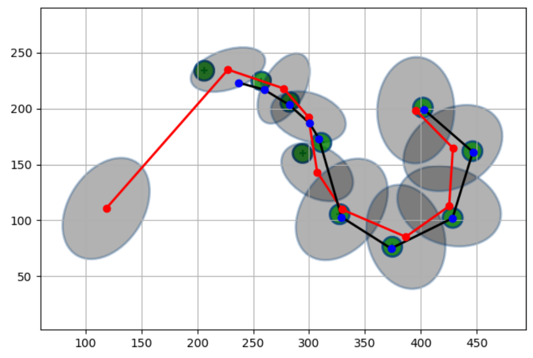

Figure 15: Unknown Curve Trajectory Estimation

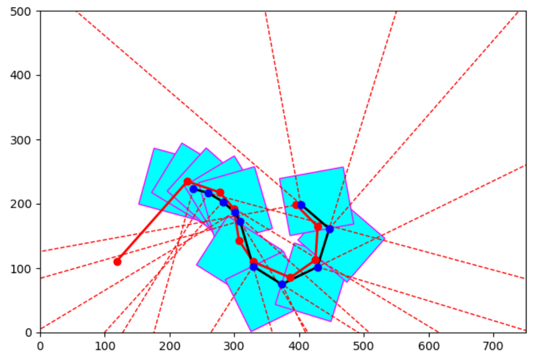

Figure 16: Unknown Curve Trajectory Robot Pose

The unknown initial pose for the curve trajectory looks much like the one for the linear trajectory except that the EKF is able to relocalize towards the true state after one measurement update rather than two.

$$\text{Computation Time} = 104ms,\ \text{Prediction Error} = 3147.353,\ \text{Update Error} = 311.758$$

The prediction error is much larger than the update error which is to be expected due to the initial unknown location. Since the robot was able to localize itself quickly from the sensor measurements, the update error remained comparatively small. Both these cases showcase the robustness of the EKF in the face of uncertainty.

4.5 Further Discussion

EKF implementation presented in this work was shown to be flexible and able to incorporate different noise models. As such, the motion model presented in section 2.4 can be extended to account for the relationship between actuator control inputs and the velocity error. In this case, standard error of an action will be calculated as 5% of current rotation speed instead of maximal speed. Doing so will decrease the uncertainty in position estimates, especially in case of curved trajectories. This change will not affect computational time of the algorithm.

It is also possible to make the simulation more realistic by including varying non-zero mean error into the rotation estimation done by gyro sensor. To do so, the state vector of the robot in section 2.3 must be extended to include time and recompute the Jacobian of the observation vector in equations (11) and (19) (because EKF algorithm implies zero-mean observation Gaussian noise). This will lead to an increase in computational cost and angle estimation uncertainty. The decrease in state estimation precision, however, will not be severe, because in given system rotation is estimated by inferring information from two sensors.

5 Conclusion

In this lab, an Extended Kalman Filter algorithm was implemented to estimate the pose of a two-wheeled non-holonomic differential drive robot. Each wheel was powered separately by continuous rotation servos. A motion model was derived using differential drive kinematics and had additive zero-mean Gaussian noise representing motor variance. Pose estimation were based on observations of two laser sensors and an IMU. The implementation was tested in different conditions and the computation time and pose estimation errors for both prediction/correction steps were analyzed.

The EKF showed acceptable performance for all of the test cases. Even in situations when completely incorrect initial state information was given to the robot, the EKF was able to relocalize close to the true location after 2-3 steps on average. It was also found that pose estimation uncertainty was higher for tests with higher maximal error in wheel angular speed and for more curved trajectories. In all of the above mentioned cases, uncertainty can be significantly decreased by decreasing time between observations and sacrificing efficiency. In terms of comparison with other state estimation algorithms, as discussed in section 2.3, it was decided that the higher computational complexity of UKF was not worth the expected minimal improvement to state estimation. Overall, EKF proved to be a robust state estimator as proven by the experiments.

6 References

[1] http://planning.cs.uiuc.edu/node659.html
[2] https://www.cs.cmu.edu/motionplanning/lecture/Chap3-Config-Space_howie.pdf