Technitute_

Advanced Kalman Filtering and Sensor Fusion

Linear Vehicle Tracker: Prediction Step

LKF Exercise 1

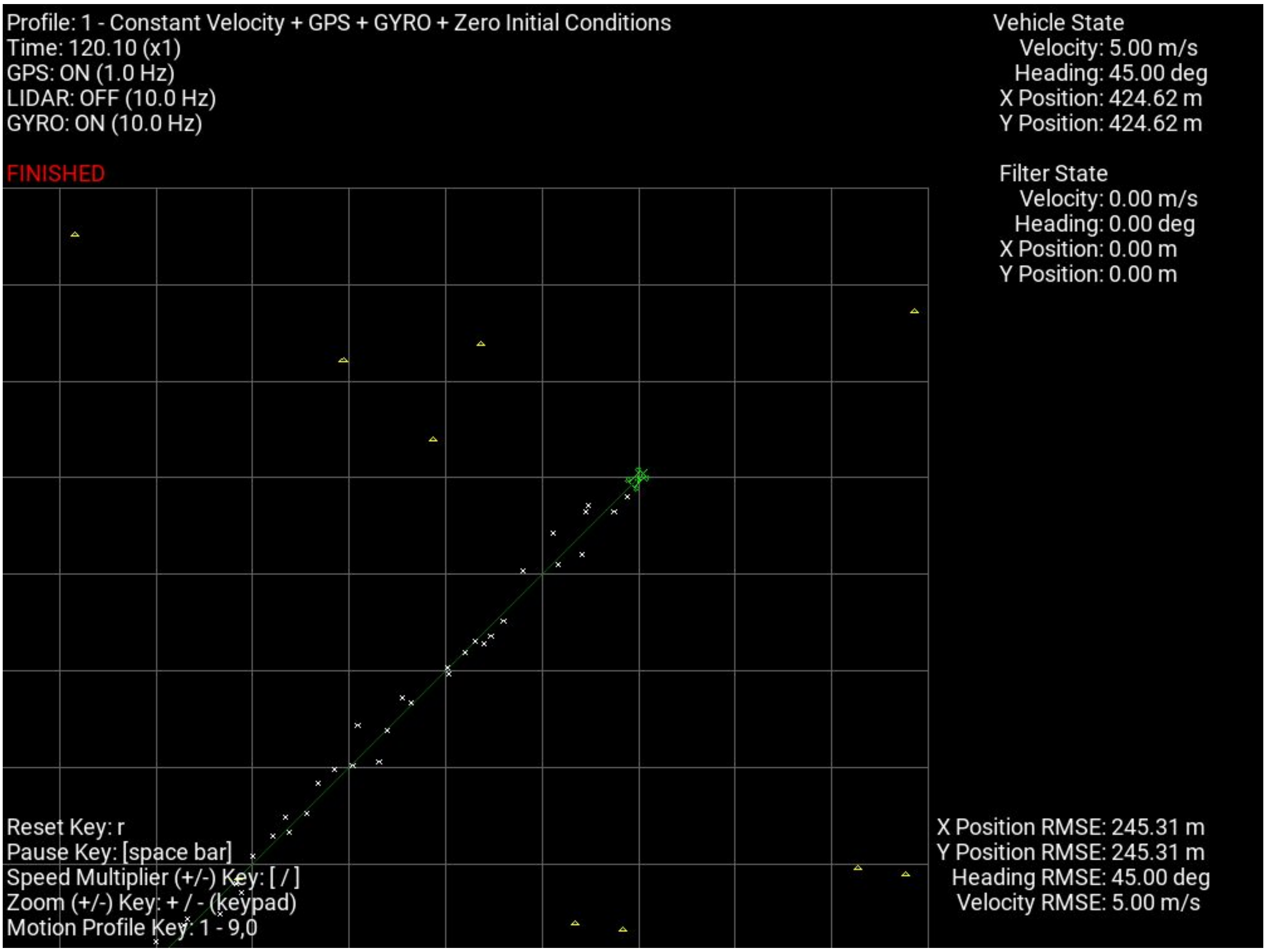
Linear Vehicle Tracker: Prediction Step Exercise

Overview

Implement the ***Linear Kalman Filter Prediction Equations*** and the ***2D Vehicle Process Model***.

Step 1 (Setup)

- Open the c++ file "**kalmanfilter.cpp**" which will be the file used in this exercise (it should be a new copy of the file "**kalmanfilter_lkf_student.cpp**" file).
- Compile the run the simulation as is, using profile 1. See that the car starts at the origin (0,0) and moves at a 45 degree angle at 5 m/s (vx,vy) = (5*cos(45deg),5*sin(45deg))



Linear Vehicle Tracker: Prediction Step Exercise

Step 2 (Setup the initial state and covariance)

- Modify the ***predictionStep()*** function
- Assume initial position is (0,0) and initial velocity is 5 m/s with a heading of 45 degrees
- Assume no initial uncertainty (Zero matrix)

```
void KalmanFilter::predictionStep(double dt)
{
    if (!isInitialised() && INIT_ON_FIRST_PREDICTION)
    {
        // Implement the State Vector and Covariance Matrix Initialisation in the
        // section below if you want to initialise the filter WITHOUT waiting for
        // the first measurement to occur. Make sure you call the setState() /
        // setCovariance() functions once you have generated the initial conditions.
        // Hint: Assume the state vector has the form [X,Y,VX,VY].
        // Hint: You can use the constants: INIT_POS_STD, INIT_VEL_STD
        // ----------------------------------------------------------------------- //
        // ENTER YOUR CODE HERE
            VectorXd state = Vector4d::Zero();
            MatrixXd cov = Matrix4d::Zero();

            // Assume the initial position is (X,Y) = (0,0) m
            // Assume the initial velocity is 5 m/s at 45 degrees (VX,VY) = (5*cos(45deg),5*sin(45deg))

            setState(state);
            setCovariance(cov);
        // ----------------------------------------------------------------------- //
    }
```

$$x = [0, 0, 3.536, 3.536]^T$$

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Linear Vehicle Tracker: Prediction Step Exercise

Step 3 (Implement the Prediction Step (Process Model))

- Modify the ***predictionStep()*** function
- Use the time step and define the F process model matrix
- Use the linear model equation to update the state

```
if (isInitialised())
{
    VectorXd state = getState();
    MatrixXd cov = getCovariance();

    // Implement The Kalman Filter Prediction Step for the system in the
    // section below.
    // Hint: You can use the constants: ACCEL_STD
    // ----------------------------------------------------------------------- //
    // ENTER YOUR CODE HERE


    // ----------------------------------------------------------------------- //

    setState(state);
    setCovariance(cov);
}
```

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x_k = \mathbf{F} x_{k-1}$$

Linear Vehicle Tracker: Prediction Step Exercise

Step 4 (Implement the Prediction Step (Covariance))

- Modify the ***predictionStep()*** function
- Define the Q matrix as a function of a variable accel_std
- Define the L matrix as a function of the time step
- Implement the covariance prediction step
- Assume the process model noise acceleration stdev is zero initially

```
if (isInitialised())
{
    VectorXd state = getState();
    MatrixXd cov = getCovariance();

    // Implement The Kalman Filter Prediction Step for the system in the
    // section below.
    // Hint: You can use the constants: ACCEL_STD
    // ----------------------------------------------------------------------- //
    // ENTER YOUR CODE HERE


    // ----------------------------------------------------------------------- //

    setState(state);
    setCovariance(cov);
}
```

$$\mathbf{Q} = \begin{bmatrix} \sigma_{a_x}^2 & 0 \\ 0 & \sigma_{a_y}^2 \end{bmatrix}$$

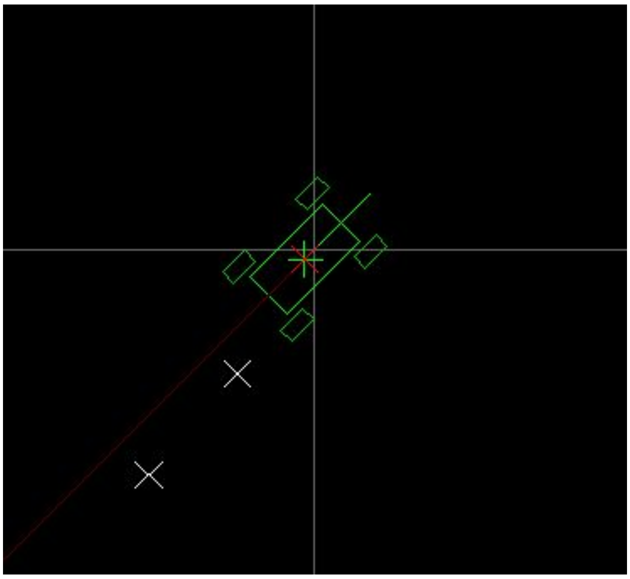
$$\mathbf{L} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}$$

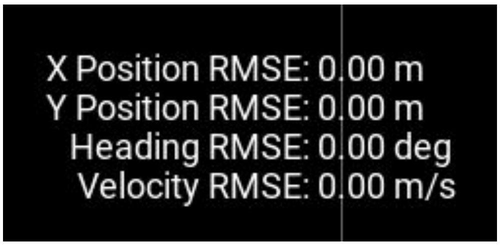
$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}^+\mathbf{F}^T + \mathbf{L}\mathbf{Q}\mathbf{L}^T$$

Linear Vehicle Tracker: Prediction Step Exercise

Step 5 (Run the Simulation)

- Check that the Prediction follows the truth fairly closely (This is because we initialized the filter with the TRUTH, see what happens with Profiles 2-4)

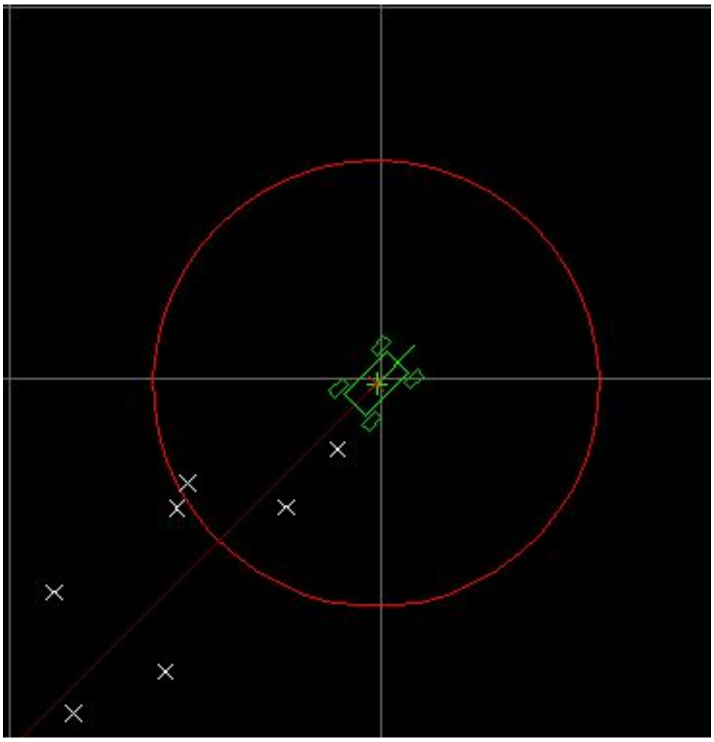




Linear Vehicle Tracker: Prediction Step Exercise

Step 6 (Check the Position Covariance Prediction is Working Correctly)

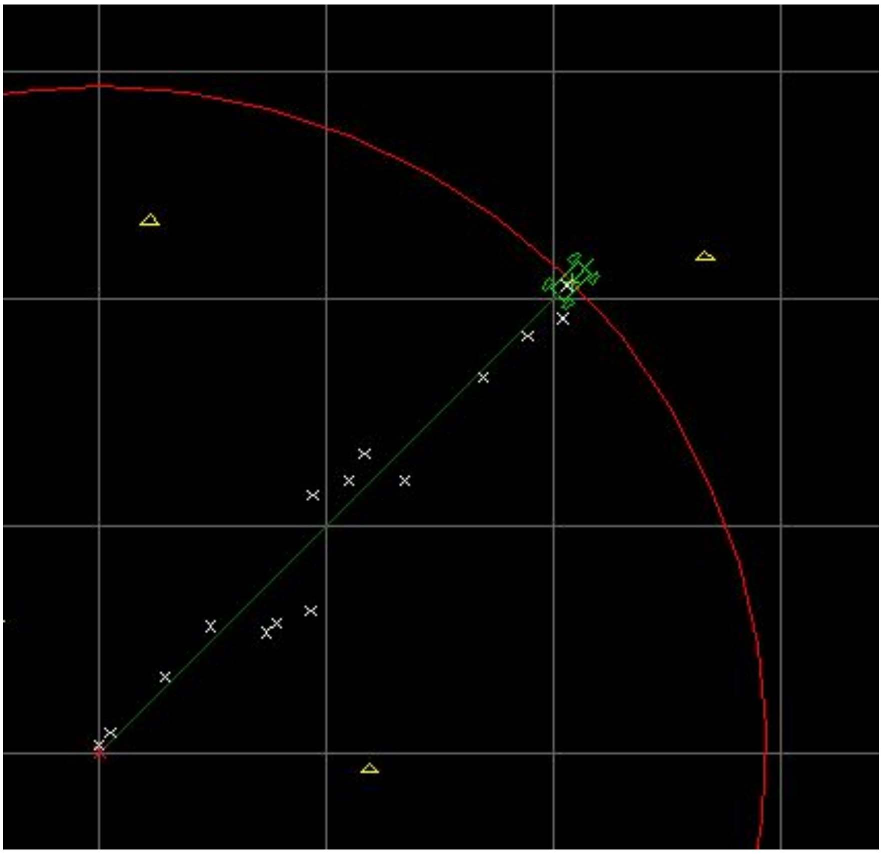
- Set the initial position x and y covariance to be (5)^2
- Run the simulation and see that the (3 Sigma) position uncertainty stays at approximately +/-15m (grid size = 25m)



Linear Vehicle Tracker: Prediction Step Exercise

Step 7 (Check the Velocity Covariance Prediction is Working Correctly)

- Set the initial state to be all zero (so estimate stays at origin)
- Set the initial position covariance to zero and the initial velocity x and y covariance to be (5/3)^2
- Run the simulation and see that the (3 Sigma) error position uncertainty grows at the same rate as the position changes



Linear Vehicle Tracker: Prediction Step Exercise

Step 8 (Check the Acceleration Covariance Prediction is Working Correctly)

- Set the initial state back to the original value
- Set the initial covariance to be all zero
- Set the process model accel_std to be 0.1
- Run the simulation and see that the (3 Sigma) velocity uncertainty grows quadratically with time