SC649 Project

"EKF localization to move around in an uncertain environment"

- By
- Shailesh K. Mahindrakar (24D1178)
- Krishna N Agaram (210051003)
- Govind Kumar (210050058)
- Under the supervision of
- Prof.Leena Vachchani

Q1. Calculation of robot pose using EKF

Extended Kalman Filter Algorithm

- Initialize with x_0
- At each time step n

① Predict

$$\hat{x}_{n,\text{prior}} = f\left(\hat{x}_{n-1,\text{update}}\right)$$

$$P_{n,\text{prior}} = F P_{n-1,\text{update}} F + Q$$

② Update

$$K_n = P_{n,\text{prior}} H^T \left(H P_{n,\text{prior}} H^T + R_{\text{measure}}\right)^{-1}$$

$$\hat{x}_{n,\text{update}} = \hat{x}_{n,\text{prior}} + K_n \left[z_{n,\text{measure}} - h(\hat{x}_{n,\text{prior}})\right]$$

$$P_{n,\text{update}} = (I - K_n H) P_{n,\text{prior}}$$

WirelessPi.com

Implementation Details (1/4): Prediction

- predicted_pose = predict_state(estimated_pose);

- P = F@P@F.T + Q

- We use X_t = X_{t-1} + [v cos(θ), v sin(θ), w] * K_samp, where [v, w] is the control input (standard unicycle situation) and K_samp the sampling frequency

- Here, estimate_pose represents μ_{t-1} and predicted_pose represents bμ_t (b for bar, i.e. predicted μ_t)

- F is the linearized state dynamics function, roughly identity. P after assignment is the predicted_covariance of x at time t.

Implementation Details (2/4): measurement and residual

predicted_measurement = predict_measurement(our_predicted_pose, landmarkA, landmarkB, landmarkC)

residual = Y_measured - predicted_measurement

Here, predicted_measurement corresponds to $h(\mu_{t-1})$ and Y_measured to z_t.

So the residual is $z_t - h(\mu_{t-1})$.

Implementation details (3/4): Kalman Gain

- H_t = get_current_H(our_predicted_pose, landmarkA, landmarkB, landmarkC)
- PH_T = np.matmul(P, H_t.T)
- S = np.matmul(H_t, PH_T) + R
- S_inv = np.linalg.inv(S)
- filter_gain = np.matmul(PH_T, S_inv)
- H_t represents the derivative of the measurement function h(.). filter_gain corresponds to the Kalman Gain.

Implementation Details (4/4): Update step

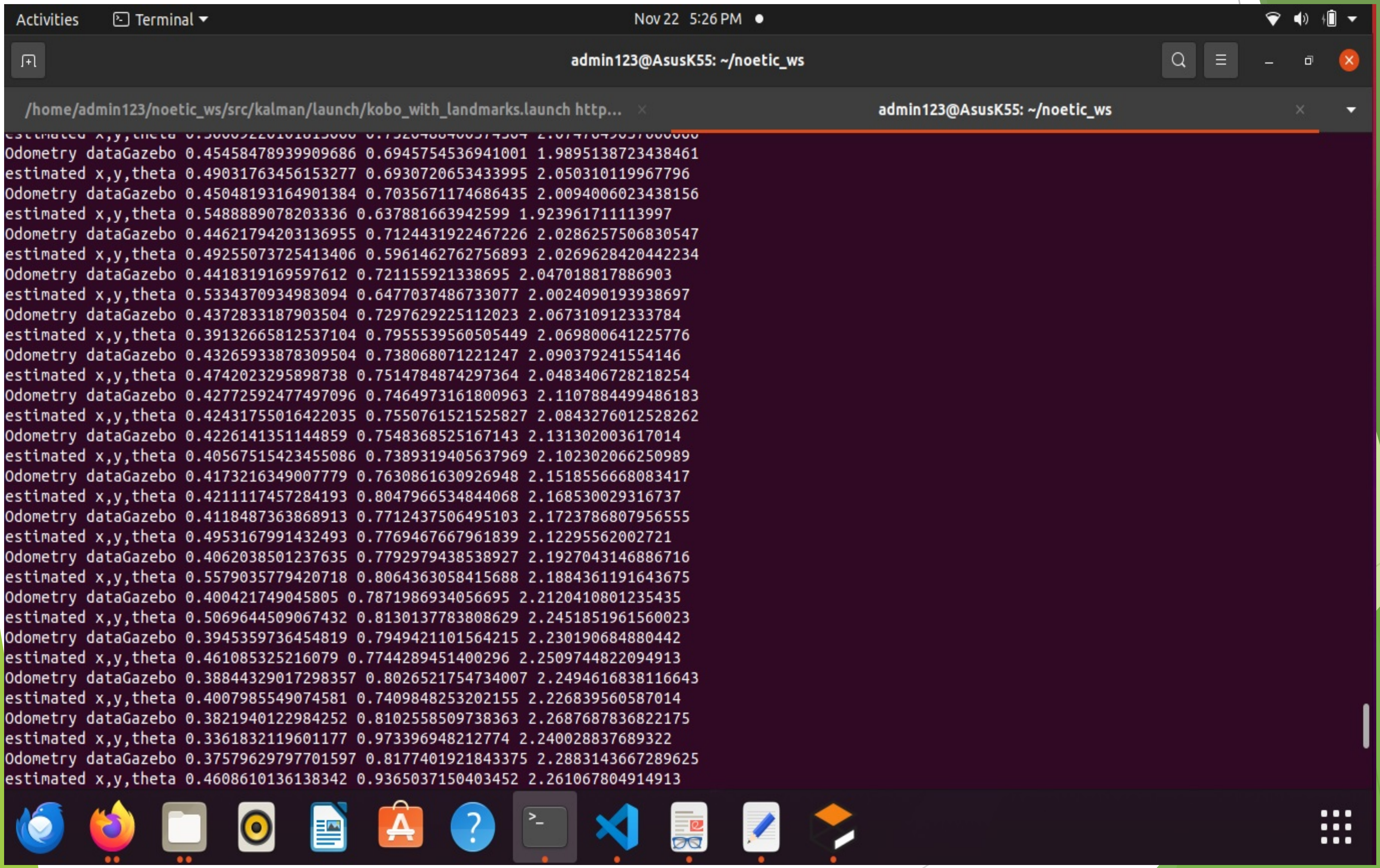
Finally, we update the predictions into estimates for position at time t.

estimated_pose = our_predicted_pose + np.matmul(filter_gain, residual)

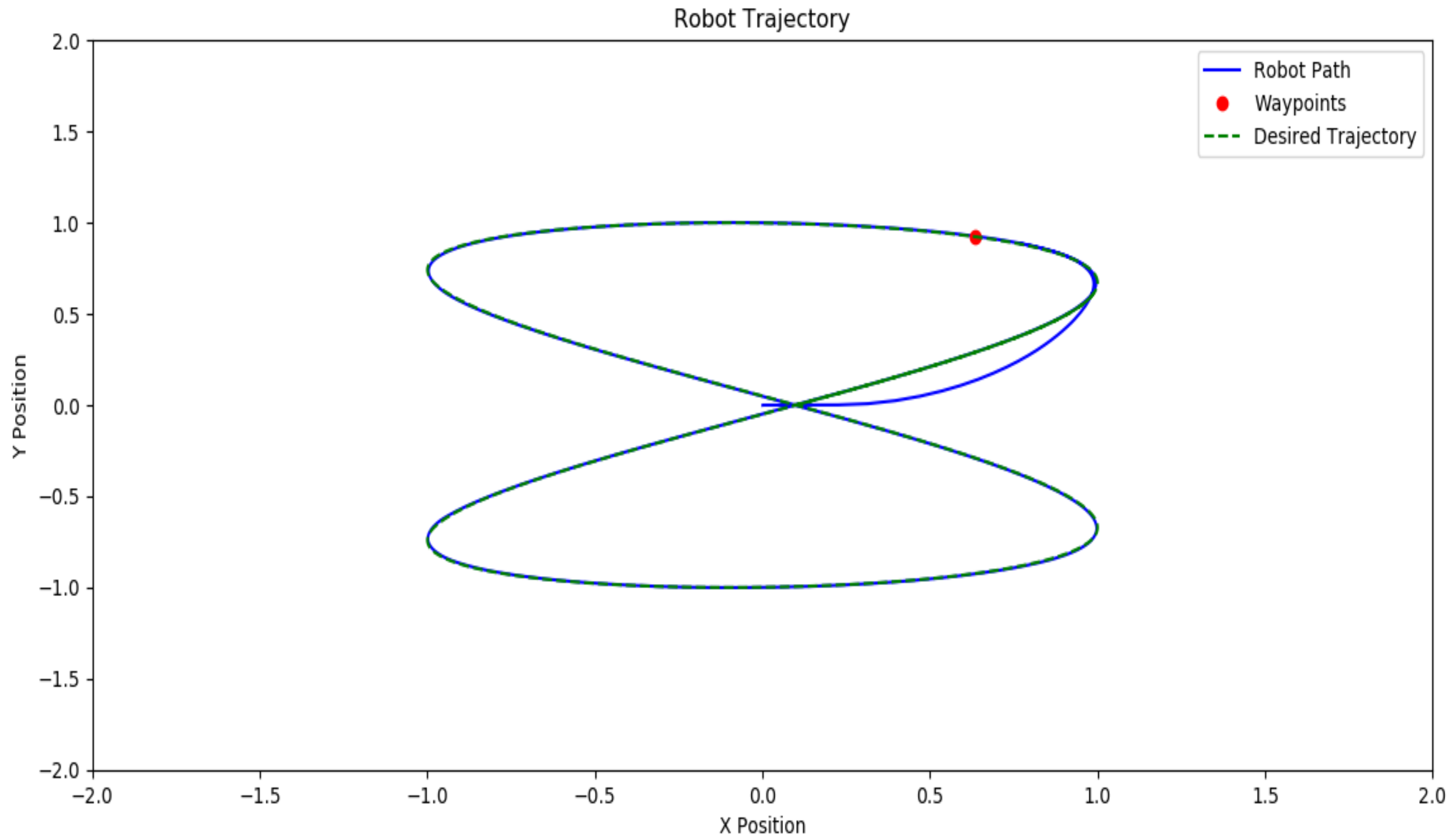
P = (I - np.matmul(filter_gain, H_t))@P

Estimated_pose = μ_t, estimated covariance is P.

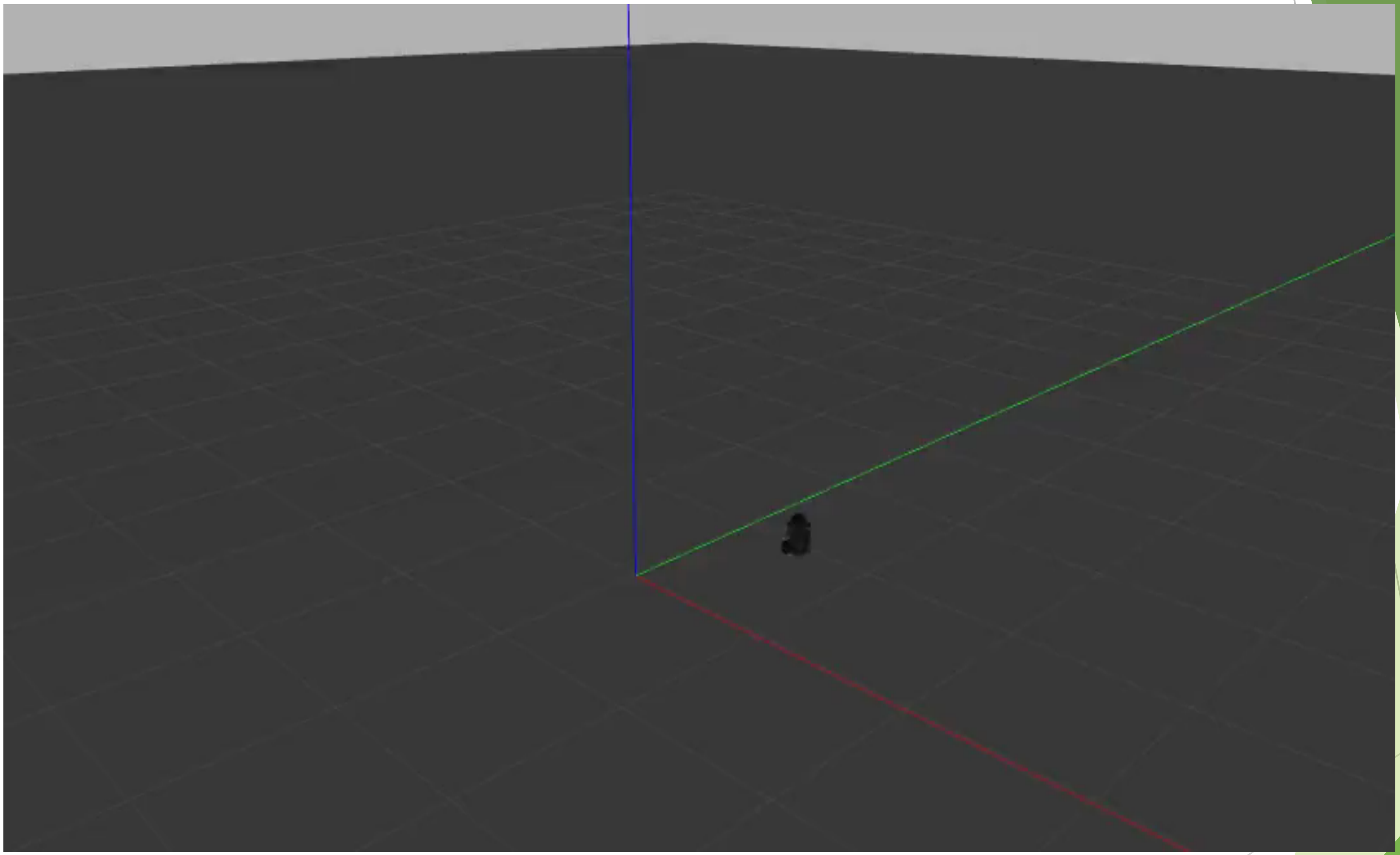
EKF works – we are able to capture almost the same odometry



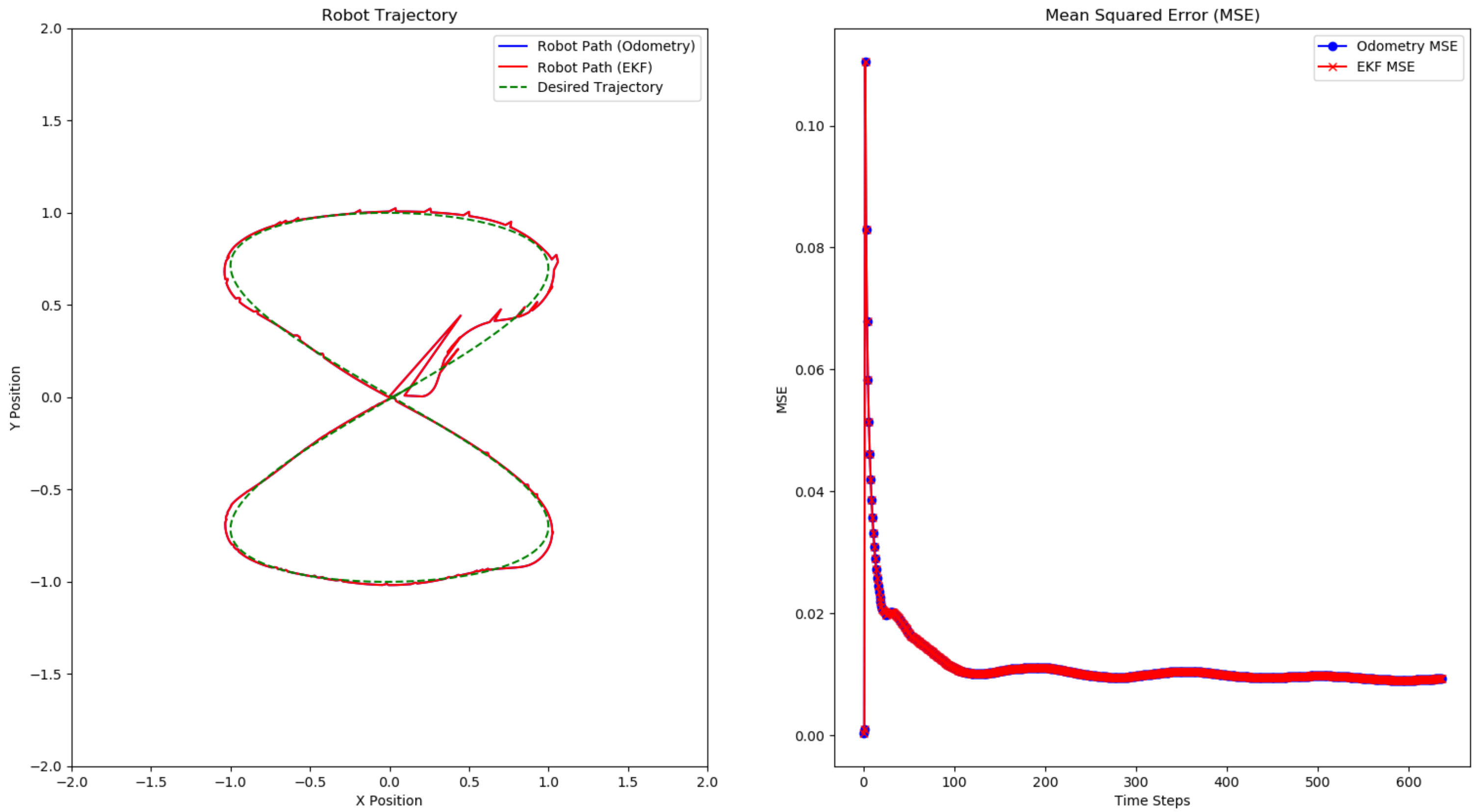
Q2. Trajectory tracking supported with videos and way-points (plotted). Example:



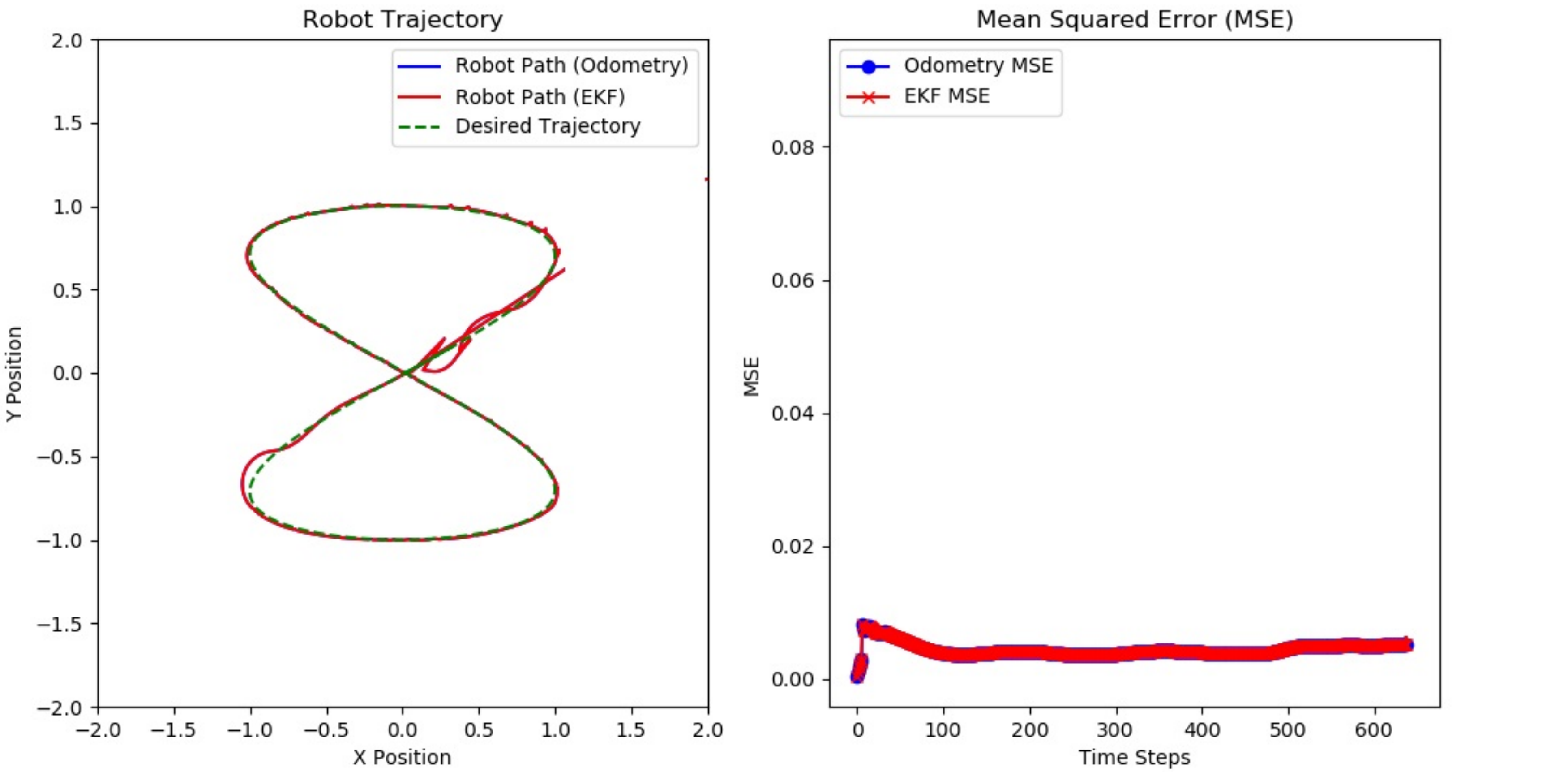
Corresponding Gazebo video



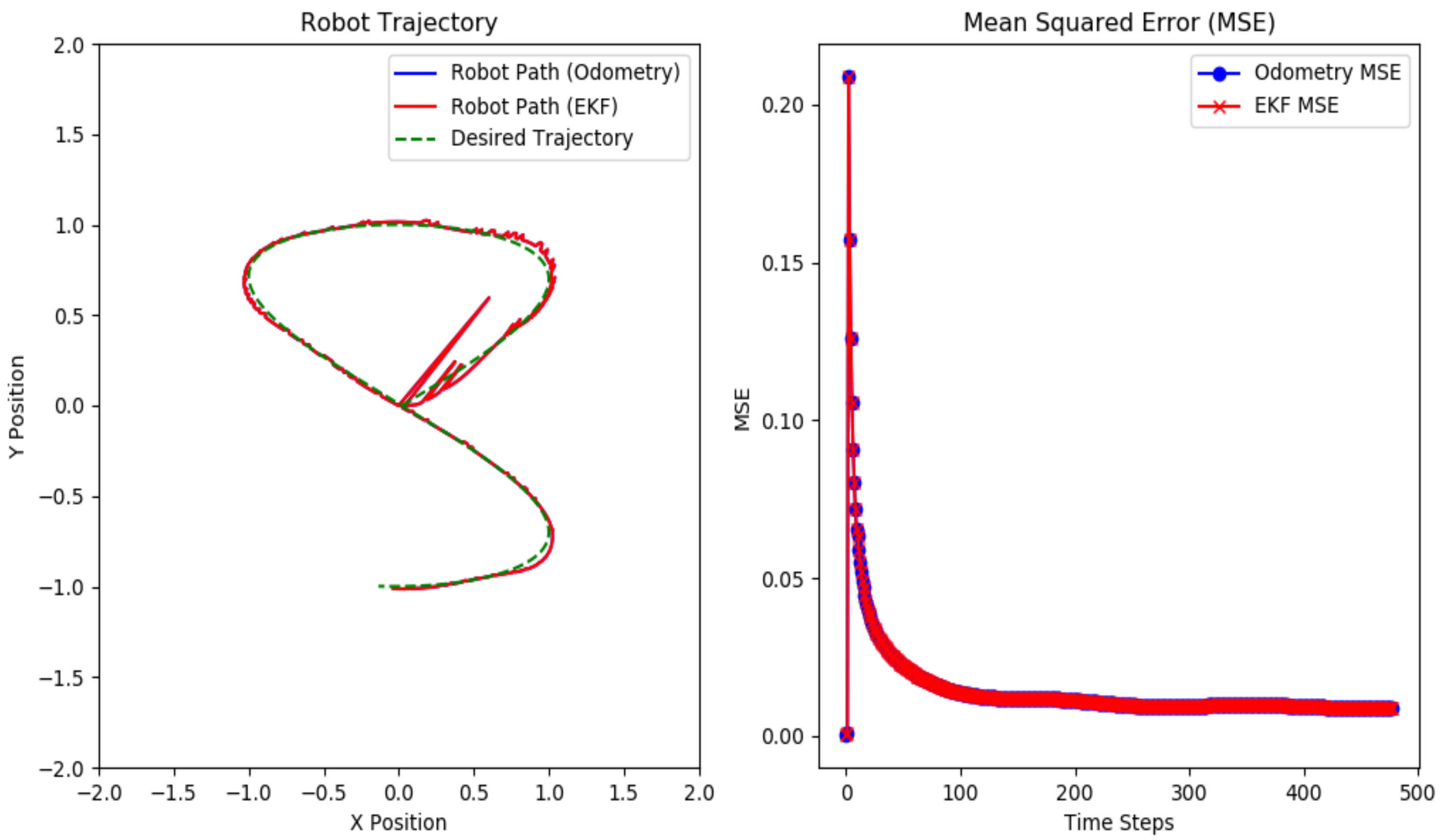
- For δ=0.02 ➔ MSE @ 0.01



- For δ=0.2 ➔ MSE @ 0.01



For $\delta=2$ → MSE @ 0.03

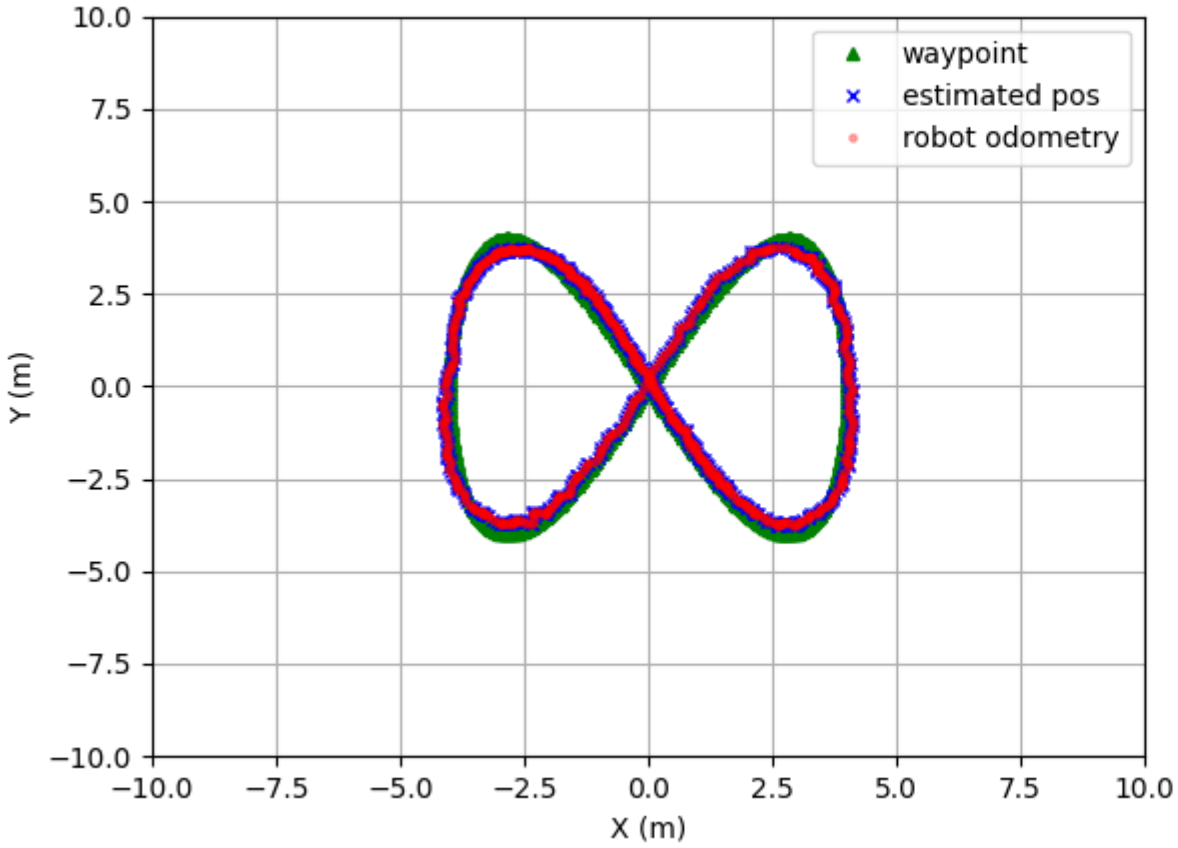


More experiments – we wanted to get the Meta logo

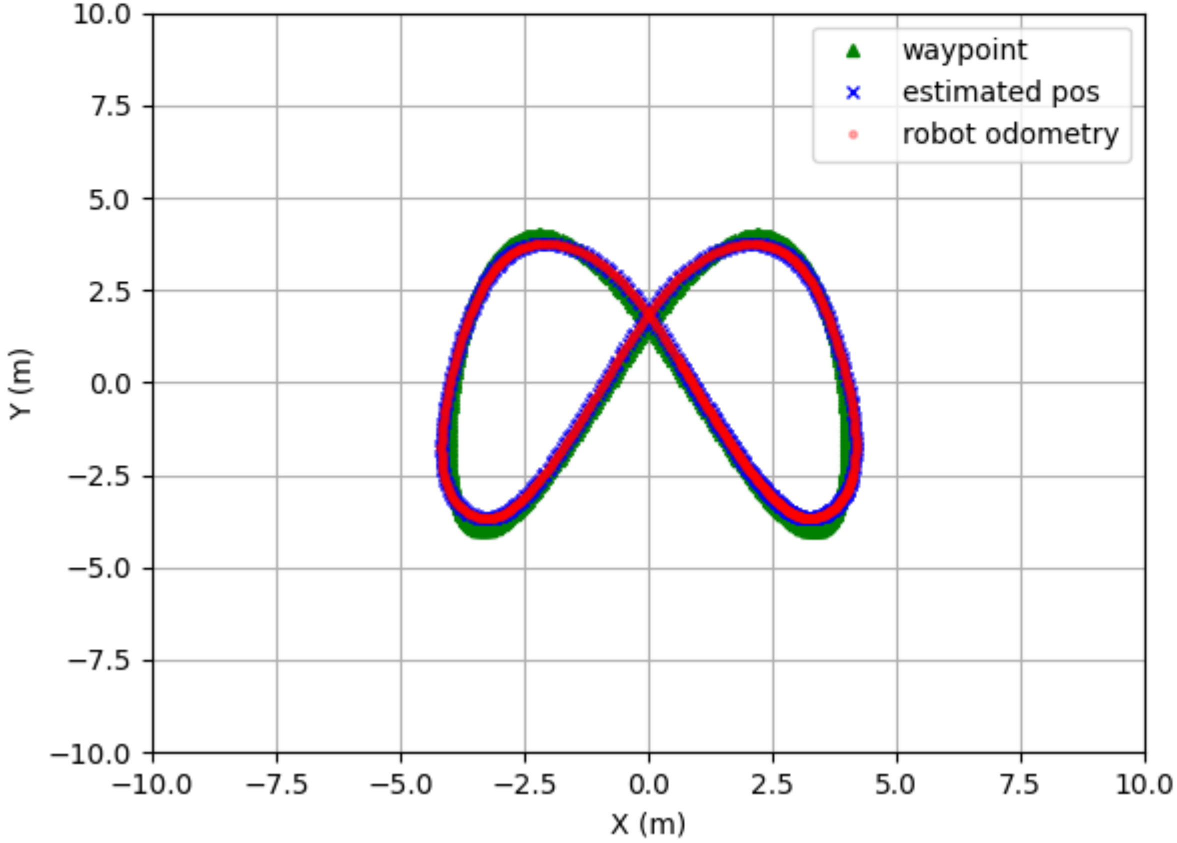
$$X, Y = 4\sin(2t+\delta), 4\cos(3t)$$

We set δ = 0.0, 0.2, 0.4 and 0.8 to see the difference in performance (MSE)

$\delta = 0$; MSE = .01

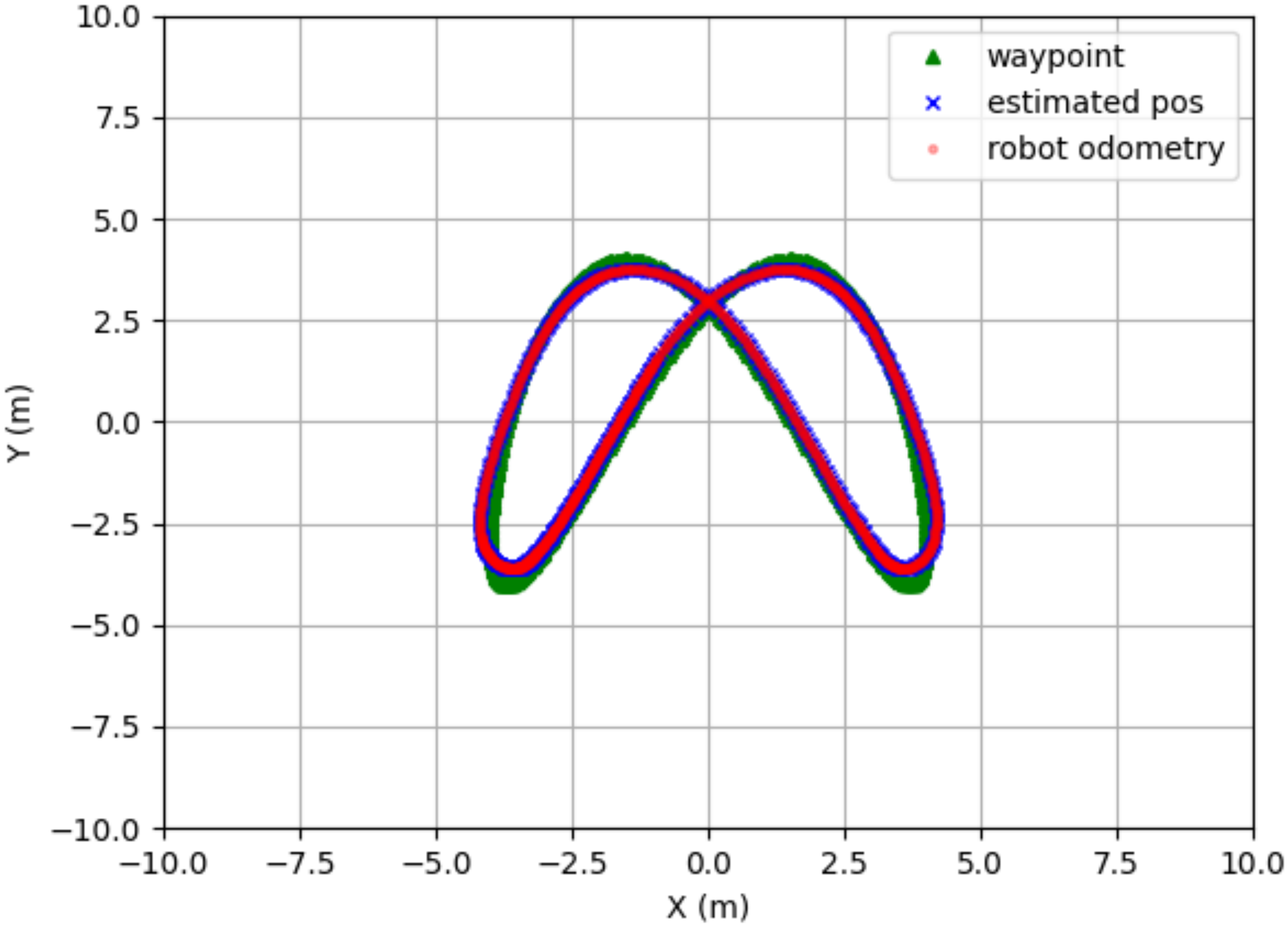


δ = .2; MSE = .01



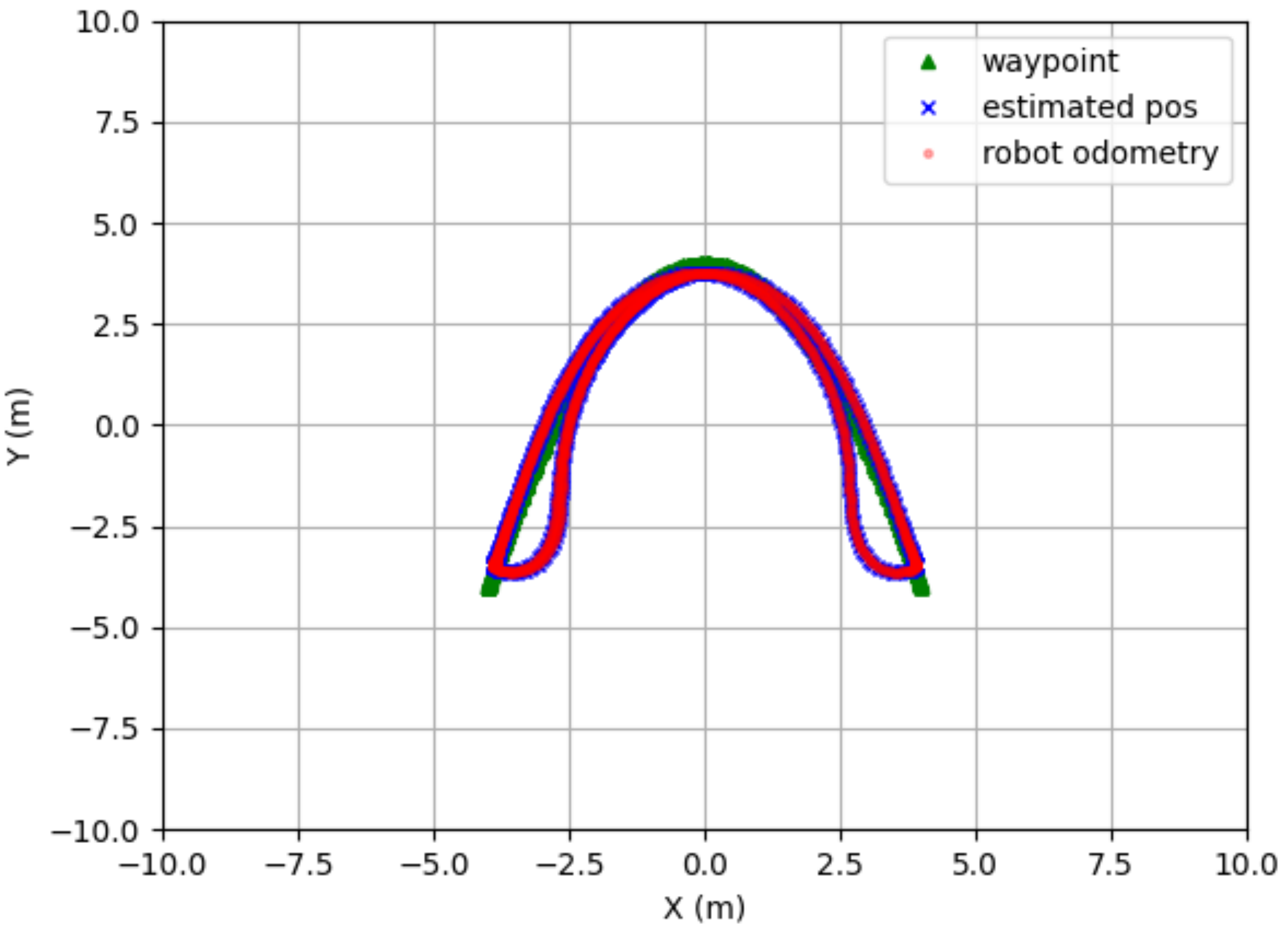
Meta

δ = .4; MSE = .01



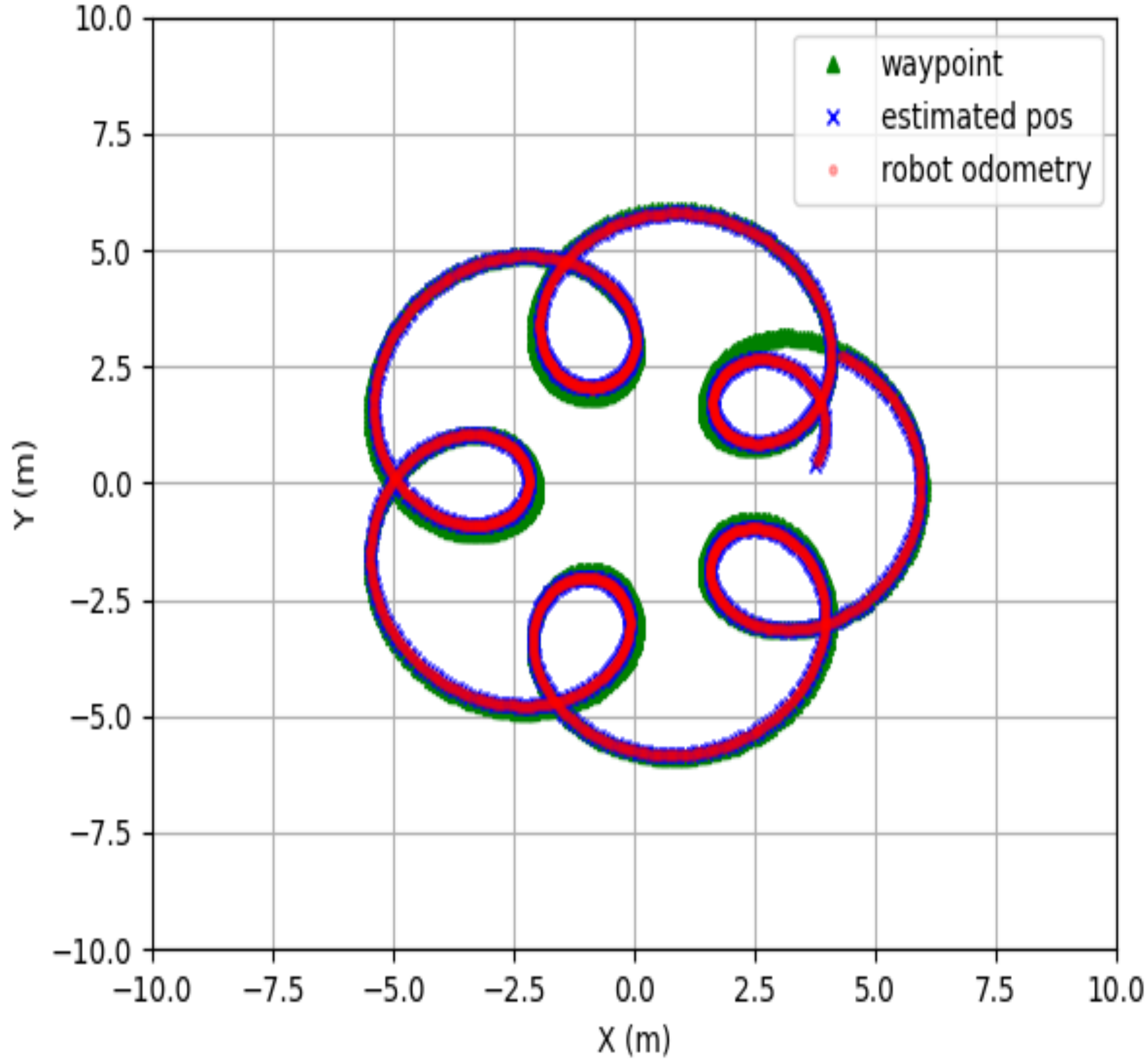
waypoint
estimated pos
robot odometry
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
Y (m)
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
X (m)

δ = .8; MSE = .01



waypoint
estimated pos
robot odometry
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
Y (m)
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
X (m)

- All experiments above have their videos on [gdrive]
- Another fancy experiment:



- Notice that the error stabilizes at 0.01 in each case.
- This corresponds to the fact that the *only error* left is the *variance* of the system distortion itself, i.e. the variance in noisy_pose as
compared to the true pose.
- This shows our filter essentially manages to filter the noise and capture the true pose with irreconcilable 0.01 due to variance in noisy_pose.

Q4. Choice of initial covariance matrices

Process Noise Covariance Matrix (Q):

- This matrix represents the uncertainty in *change of* the system model.
- Uncertainty because of inaccurate application of control/unmodeled dynamics of environment
- Initial value is the prior on this uncertainty. We chose a diagonal matrix with diagonal values 0.1 each.

Measurement Noise Covariance Matrix (R):

- Similar to Q, but in the measurement received. We use the values set in trilateration.py

Initial State Covariance Matrix (P):

- **P** represents the uncertainty in the initial state of the system. Since we ourselves set the start position (to origin), we have less uncertainty and use a diagonal matrix with 0.1 as diagonal entries. We could use the zero-matrix as well.

Why all diagonal? Initially, we can assume that errors are roughly uncorrelated by supposing that they were independently generated.

Q4: Effect of sampling time on initialization

- In the Kalman filter, the **sampling time** between corrections can influence the performance of the filter! It also influences the choice of initial covariances, as we shall see below.
- **Short Sampling Time (Higher Frequency):**
  - It is reasonable to assume that noise with shorter sampling times is smaller, because of the smaller time allowed for errors to occur. Thus we initialize the covariance to smaller values.
  - In fact, if the sensors/dynamics are noisy, frequent updates would need frequent measurements and dynamics evaluation, meaning it is hard to correct error because of this frequently accumulating sensor/dynamics error.
- **Long Sampling Time (Lower Frequency):**
  - In this case, it would be more likely for variance in errors to be larger, since there is more time for errors to add up. Thus, we would do well to choose a larger initial covariance.

Analysis of Results Based on Sampling Time Variation

When varying the sampling time, the following key behaviors were observed:

Fast Sampling (Shorter delta_t):

- The filter corrected the state estimates more frequently, leading to tighter tracking of the actual robot trajectory.
- However, in the presence of noisy sensor data, the frequent corrections sometimes led to more erratic state estimates, as noise influenced the correction step too frequently.
- The MSE plot showed faster convergence initially but also higher oscillations in the error due to the influence of measurement noise.

Slow Sampling (Longer delta_t):

- The filter corrections were less frequent, resulting in smoother state estimates as the process model dominated for longer periods.
- However, if the model did not perfectly capture the system dynamics, larger prediction errors accumulated between correction steps, causing larger deviations from the true state.
- The MSE plot showed slower convergence and, in some cases, a larger steady-state error, particularly when the process model was less accurate.

Thank You !

(for SC 649)