Robot Localisation and Navigation Project 3 Report:

Name: Santosh Srinivas Ravichandran
Net ID: sr6411

Abstract: This project focuses on developing an Unscented Kalman Filter (UKF) to fuse inertial data from an IMU with vision-based pose and velocity estimates for improved state estimation of a robot. The project is divided into two parts: 1) Fusing IMU data with visual pose estimates expressed in the world frame, and 2) Fusing IMU data with velocity estimates from optical flow expressed in the camera frame. The UKF is expected to better capture system nonlinearities compared to the Extended Kalman Filter used in previous projects, at the potential cost of increased runtime. The algorithms will be tested on two datasets, with performance evaluated against ground truth data from a Vicon motion capture system. Thus, the project serves to integrate concepts learned throughout the course, advancing skills in multi-sensor fusion and state estimation for robotics applications.

Dataset Description:

Sampled Data from the IMU contains the angular velocity and acceleration of the body from the gyroscope and accelerometer which will be used as control input.

Sampled Vicon data that is synchronized with the IMU data contains the pose and velocity of the robot in the world frame.

The project 2 results that include the visual pose of the robot in the world frame and velocity estimates of the robot in the camera frame are used for the measurement update. The corresponding data for dataset 1 and dataset 4 is used.

Project Methodology:

The project is divided into 2 parts and the methodology for each is discussed below.

Part 1 Methodology:

Part 1 involves fusing IMU data with visual pose estimates by implementing the UKF. For each time step, the state is predicted using the IMU-driven motion model and the UKF prediction step and then updated using the visual pose measurements from project 2 data.

pred_step.m

Function definition: The given function performs the prediction step of the Unscented Kalman Filter (UKF) algorithm. The function takes the following inputs:

- uPrev: Mean of the previous state estimate (vector)
- covarPrev: Covariance of the previous state estimate (matrix)
- angVel: Angular velocity control input at the current time step from the IMU data (vector)
- acc: Acceleration control input at the current time step from the IMU (vector)
- dt: Time difference between the current and previous time step (scalar)

Description:

The process model is from the IMU and is same as project 1:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} position \\ orientation \\ \text{linear velocity} \\ gyroscope\,bias \\ accelerometer\,bias \end{bmatrix}$$

$$\dot{x} = f = \begin{bmatrix} x_3 \\ G(x_2)^{-1} \cdot (\omega_m - x_4 - n_g) \\ g + R(x_2)(a_m - x_5 - n_a) \\ n_{bg} \\ n_{ba} \end{bmatrix}$$

$$x_t = f(x_t, u, n)$$

$$x_t = x_{t-1} + n_{bgd}\,;\ (dont\,discretize\,noise)$$

$$x_t = x_{t-1} + fdt\,(for\,other\,states)$$

Step1: Compute sigma points with state augmentation.

Because of the non-linearity, UKF uses sigma points to compute the predicted mean of the states.

Sigma points are calculated by taking a distribution of points around the previous state value.

Augmented states are obtained by concatenating the state with the noise.

$n' = n + n_q \, where \, n \, is \, \dim ension \, of \, state \, and \, nq \, is \, of \, noise$
$x^{(0)}_{aug,t-1} = \mu_{aug,t-1}$

$x^{(i)}_{aug,t-1} = \mu_{aug,t-1} \pm \sqrt{(n' + \lambda')}. \left[\sqrt{(\Sigma_i)}\right] \; i = 1, 2.. \, n'$

$\mu_{au\,\mathbf{g},t-1} = \begin{bmatrix} \mu_{t-1} \\ 0 \end{bmatrix}; \; P_{au\,\mathbf{g},t-1} = \begin{bmatrix} \Sigma_{t-1} & 0 \\ 0 & Q_t \end{bmatrix}$

$\lambda' = \alpha^2 \cdot \left(n' + k\right) - n'; \; \alpha = 0.001; \; k = 1; \; \beta = 2$

Step 2: Propagate sigma points through the nonlinear function f

$x^{(i)}_t = f\left(x^{(i),x}_{au\,\mathbf{g}}, \; x^{(i),q}_{au\,\mathbf{g}}\right)$

Take the state and noise part separately of the augmented state and pass it through the function f as defined in the process model. Thus for each sigma point the output is computed as below.

$x_t = \; f(x_t, u, n)$

$x_t = x_{t-1} + n_{bgd} \; ; \; (dont \, discretize \, noise)$

$x_t = x_{t-1} + fdt \; (for \, other \, states)$

Step 3: Compute the mean and covariance
$\mu_t^- \; = \; \Sigma W_i^{m'} \cdot x_t^{(i)} \; \; i = 0, 1, 2.. \, .2n$

$W_0^{(m')} = \lambda' / \left(n' + \lambda'\right) \; ; \; W_i^{(m')} = 1 / \left(2. \left(n' + \lambda'\right)\right)$

$\Sigma_t^- \; = \; \Sigma \, W_i^{(c')} \left(x_t^i - \mu_t^-\right) \left(x_t^i - \mu_t^-\right)'$

$W_0^{(c')} = \lambda' / \left(n' + \lambda'\right) + \left(1 - \alpha^2 + \beta\right) \; ; \; W_i^{(c')} = 1 / \left(2. \left(n' + \lambda'\right)\right)$

Update_step.m

Function Definition:

The function upd_step performs the update step of the Kalman filter algorithm. It takes the camera data from project 2 (z_t), the estimated covariance matrix (covarEst), and the estimated mean of the state (uEst) as inputs, and returns the updated state estimate (uCurr) and the updated covariance matrix (covar_curr).

Description:

Once the predicted mean and co-variance is computed, the update using the pose measurement from the camera is used for the state update. Since the measurement pose from the camera is already in the world frame, the measurement model is linear. Thus the update equation is the same as used in project 1.

$$Z_t = Cx + \eta$$

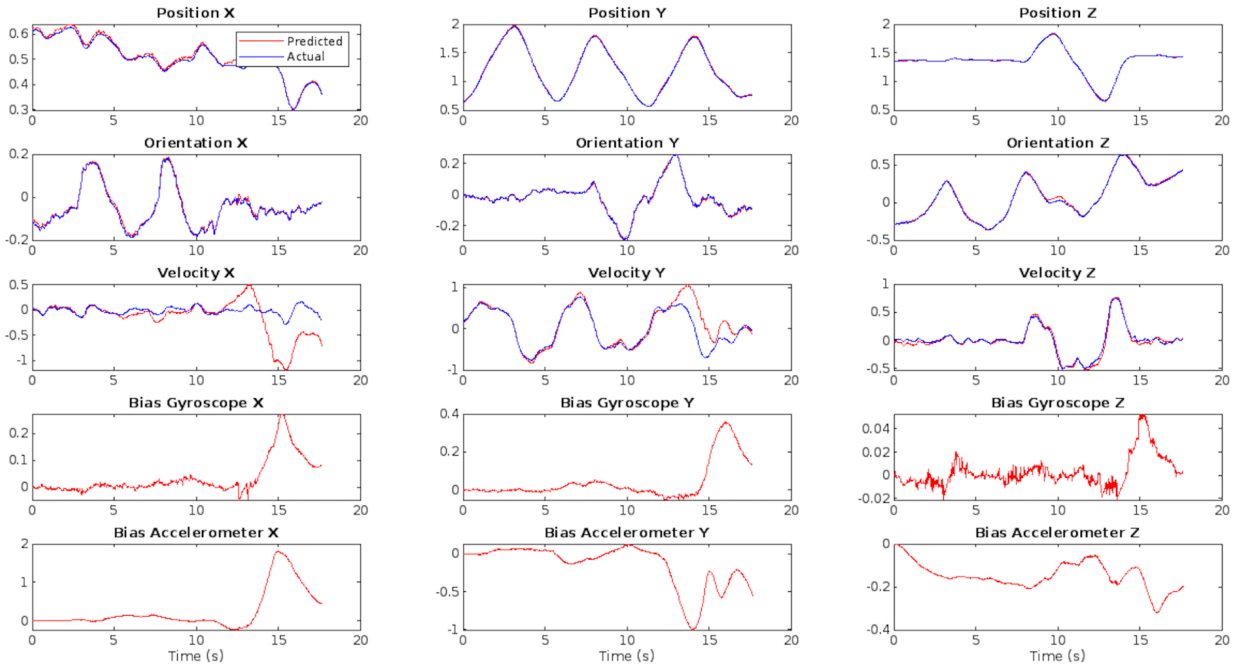
$$\mu_t = \mu_t^- + K_t\left(z_t - C\mu_t^-\right)$$

$$\Sigma_t = \Sigma_t^- - K_t C\,\Sigma_t^-$$

$$K_t = \Sigma_t^- C^T\left(C\Sigma_t^- C^T + R\right)^{-1}$$

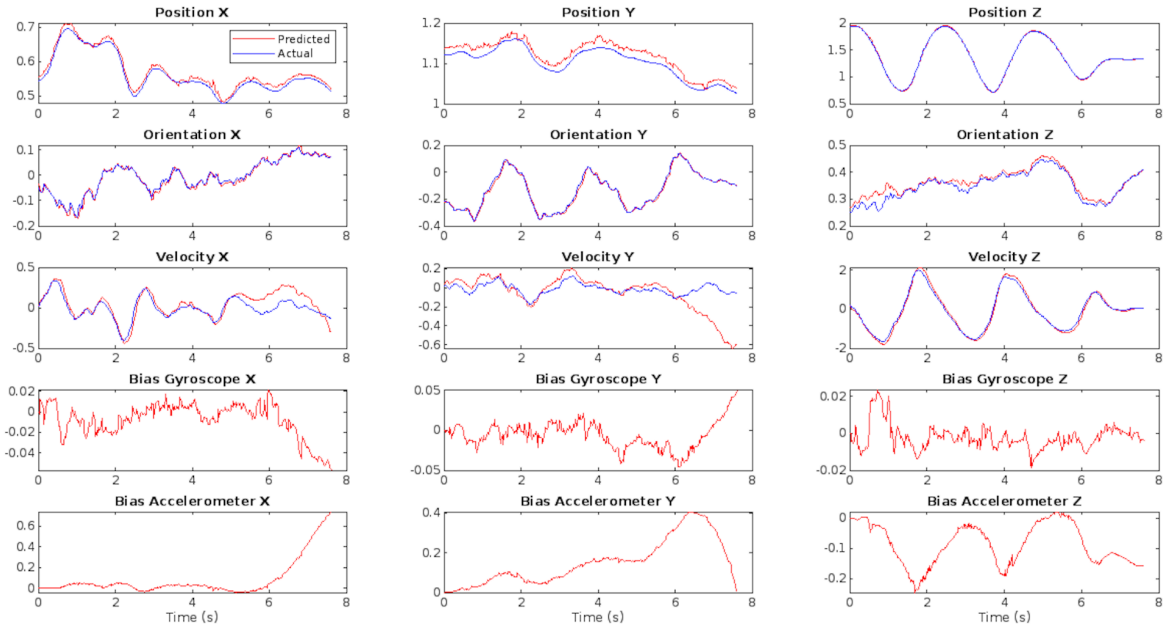
Results:

Dataset 1:



For position and orientation X, Y, and Z, the UKF seems to have performed well, with the predicted values closely following the actual values. The velocity predictions are reasonable but show some deviation, especially noticeable in the X-axis. There's a significant spike in bias gyroscope X and Y axes and a noticeable drift in the Z-axis, which suggests that there may be unmodeled dynamics or sensor errors affecting the gyroscope readings.

Dataset 4:



The position and orientations predictions are fairly accurate with the predicted values closely following the actual values across all axes. Velocity in the X and Z axes follows the actual values quite well, while there are some discrepancies in the Y-axis.The bias of the gyroscope shows more noise compared to Dataset 1 but less dramatic spikes. There is a clear upward trend in the bias on the Y-axis, which could point to a systematic error in the gyroscope or a change in the sensor's operating conditions.

Part 2 Methodology:

Pred_step.m:

This function is exactly the same as the one we used for part 1.

update_step.m:

Step1: Compute sigma points with state augmentation.

$$z_t = g(x_t) + v_t;\ v_t = N(0, V_t)$$

$$x_t^{(i)} = \mu_t^- \pm \sqrt{(n+\lambda'')}.\sqrt{(\Sigma^-)_i}\, i = 1 \to n$$

$$x_t^0 = \mu_t^-$$

Since noise is additive, we don't include an augmented state. The predicted mean and covariance are used to generate the sigma points directly without adding noise in the augmentation. The value of lambda is the same as defined earlier.

Step 2: Propagate sigma points through the nonlinear function g:

All the computed sigma points are passed through g.

$$g_\mu = {}^{c}R_b.\ l(x_2, x_3) - {}^{c}R_b.\ S\left({}^{B}r_{BC}\right).\ {}^{b}R_c.\ {}^{w}\omega_c$$

$$l(x_2, x_3) = R(x_2).x_3$$

$${}^{w}\omega_c\ is \text{ obtained from the measurement passed. } {}^{b}R_c\ is\ given$$

Thus using the above the output Zt after passing through g is obtained for all sigma points.

Step 3: Compute the mean and covariance

The mean , cross covariance and covariance of all the output sigma points are computed using the weighted formula below.

$$Z_{\mu_t} = \Sigma W_i^{m'} \cdot Z_t^{(i)}\ i = 0, 1, 2..2n$$

$$W_0^{(m')} = \lambda/(n+\lambda)\,;\ W_i^{(m')} = 1/\left(2.\left(n' + \lambda'\right)\right)$$

$$S_t = \Sigma\, W_i^{(c')}\left(Z_t^{(i)} - Z_{\mu_t}\right)\left(Z_t^{(i)} - Z_{\mu_t}\right)'\ i = 0, 1, 2..2n$$

$$R_t = \Sigma\, W_i^{(c')}\left(X_t^{(i)} - \mu_t^{-}\right)\left(Z_t^{(i)} - Z_{\mu_t}\right)'\ i = 0, 1, 2..2n$$

$$W_0^{(c')} = \lambda/(n+\lambda) + \left(1 - \alpha^2 + \beta\right)\,;\ W_i^{(c')} = 1/(2.(n+\lambda))$$

Step 4: Update using Kalman Gain:

Finally we use the conditional distribution to get the final state incorporating the measurement.

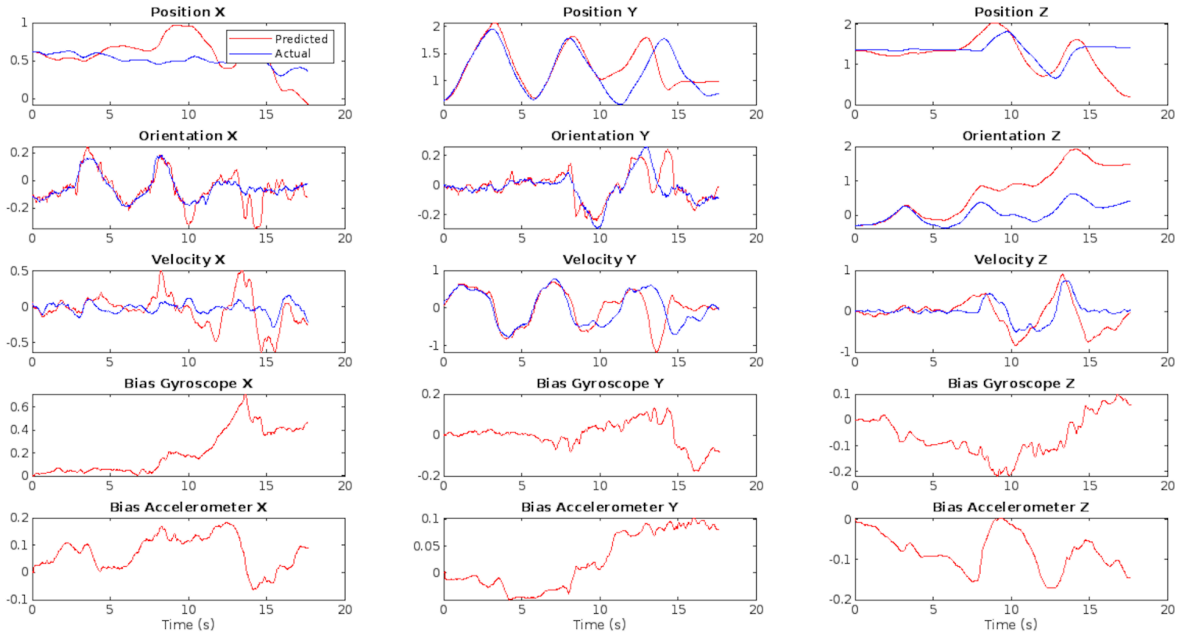
$$\mu_t = \mu_t^- + K_t(Z_t - Z_\mu)$$

$$\Sigma_t = \Sigma_t^- - K_t S_t K_t'$$

$$K_t = C_t S_t^{-1}$$

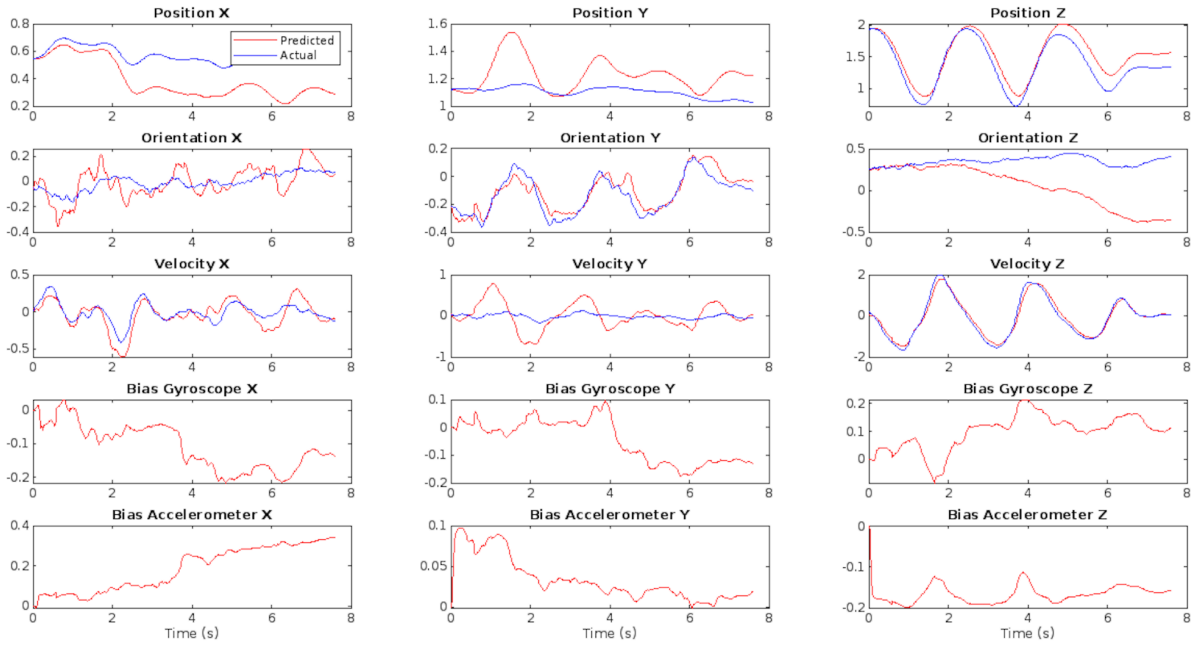
Results:

Dataset 1:



Position (X, Y, Z): The predicted trajectory generally follows the actual one, but with noticeable deviations. Orientation (X, Y, Z): There are discrepancies between predicted and actual values, particularly in the Z axis. Velocity (X, Y, Z): The filter is able to track the velocity quite well, which is expected since velocity is used in the measurement update. Bias Gyroscope (X, Y, Z): The bias is increasing in all axes, which could indicate drift in the gyroscope sensors. Bias Accelerometer (X, Y, Z): Similar to the gyroscope. This trend could point to accelerometer drift.

Dataset 4:



Position (X, Y, Z): The UKF appears to track the actual position more closely in this dataset (notice the scale), suggesting better sensor data or a model that better fits this particular scenario.Orientation (X, Y, Z): There is less deviation in the orientation predictions compared to Dataset 1, but some error is still present, especially in the Z-axis. Velocity (X, Y, Z): The predicted velocities closely follow the actual velocities, which aligns with the use of velocity in the measurement updates. The UKF seems well-calibrated to handle velocity measurements in this dataset.

Overall, the UKF performs reasonably in both models and datasets. Some deviations could potentially be addressed by refining and adjusting the UKF noise parameters. It's also important to note that the measurement data used for the update itself might not be as perfect since it is the data coming from the optical flow and pose measurements from project 2 which might not be as accurate as the vicon data.

Conclusion:

In this project, we successfully implemented an Unscented Kalman Filter (UKF) to fuse inertial data from an IMU with vision-based pose and velocity estimates for improved state estimation of a robot. The UKF algorithm demonstrated its ability to effectively capture system nonlinearities and provide accurate state estimates by leveraging the IMU-driven motion model and incorporating visual measurements. The fusion of IMU data with visual pose estimates and optical flow velocity estimates resulted in a more precise and robust estimation of the robot's trajectory and velocity compared to using individual sensor modalities. The performance of the UKF was evaluated against ground truth data from the Vicon motion capture system, showing significant improvement in estimation accuracy while some deviations also highlighted the

limitations and areas for future improvement. This project lays the foundation for further research and development in autonomous aerial robotics.