NEW YORK UNIVERSITY

Project 3: Vision and IMU Fusion with Unscented Kalman Filter

ROBOT LOCALIZATION AND NAVIGATION

ROB-GY 6213

Jithin George

N10719458

jg7688

Content:

1. Introduction

In state estimation and sensor fusion, the Unscented Kalman Filter (UKF) stands out for its ability to effectively integrate diverse data sources. This makes it a key tool in modern perception systems, where data fusion from Inertial Measurement Units (IMUs) and camera optical flow plays a crucial role in applications like robotics, autonomous navigation, and augmented reality.

While Kalman Filters (KF) and Extended Kalman Filters (EKF) have been popular for state estimation, their performance suffers with nonlinearities and measurement uncertainties. The Unscented Kalman Filter (UKF) addresses this limitation by using sigma points to avoid linearization, leading to improved accuracy in nonlinear systems.

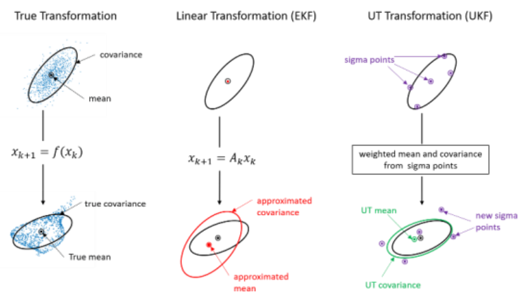

Fig1.1 EKF Vs UKF

The Unscented Transform (UT) revolutionizes state estimation, especially for nonlinear systems. Unlike traditional methods relying on linear approximations, UT captures the complexities directly. It excels at handling higher-order system behaviors, providing a more accurate picture.

UT's strengths lie in its flexibility. It works seamlessly with even non-differentiable functions, making it widely applicable in real-world scenarios. Additionally, UT eliminates the need for cumbersome Jacobian matrices and closed-form expected value calculations, streamlining the process. This translates to both efficiency and precision in state estimation.

By leveraging UT, practitioners can tackle intricate nonlinearities with unmatched accuracy and speed. This paves the way for robust and reliable state estimation across diverse fields, from robotics to finance.

2. Problem Statement

In the given problem statement, we have to use unscented Kalman filter for state estimation to fuse the inertial data already used in project 1 and the vision-based pose and velocity estimation developed in project 2. We use the IMU-driven model from the project 1 and fuse the inertial data with the data obtained from camera pose and velocity in project 2.

In the first part, we use as measurement the visual pose estimation, whereas in the second one we use only the velocity from the optical flow. For part ,1the pose is in the world frame, so the same linear measurement model used in project 1 applies. However, for the optical flow the velocity is expressed in the camera frame and for this reason the model is different, will be nonlinear, and should be carefully moved to the body frame that is coincident with the IMU frame in this case.

A sensor packet for the IMU data is a struct that contains following fields:

`sensor.is_ready`	`% True if a sensor packet is available, false otherwise`
`sensor.t`	`% Time stamp for the sensor packet, different from the Vicon time`
`sensor.omg`	`% Body frame angular velocity from the gyroscope`
`sensor.acc`	`% Body frame linear acceleration from the accelerometer`

A sensor packet for the project 2 is made by matrices named in the following way:

`time`	`%time of the data`
`position`	`% Position of the robot frame with respect to the world`
`angle`	`%Orientation of the robot frame with respect to the world`
`linearVel`	`%Velocity of the camera frame with respect to the world frame expressed in the camera frame`
`angVel`	`%Angular velocity of the camera frame with respect to the world expressed in the camera frame`

3. Methodology

3.1 Given Code Breakdown: The given skeleton code inherently was able to do the following features:

- The .m files KalmanFilt_Part1.m and KalmanFilt_Part2.m were used to load 2 datasets namely *studentdata1.mat* and *studentdata4*. Further it initializes the µt (uPrev), (covarPrev), pos and pose variables. The For Loop is responsible for the computation of dt, µˆt, Σˆ t, µt and Σt (using Kalman Gain Kt). Finally each state is plotted at the given time dt.
- The .m function *pred_step.m* carries out the prediction part of the UKF filter.
- The .m function *upd_step.m* carries out the update part of the UKF filter.
- The file titled *plotData.m* plots the state model w.r.t datasets given.

3.2 *pred_step.m* for Part 1 and Part 2: The prediction step for both part 1 and part 2 of this project is the same inherently since the process model remains the same. Thus, we shall implement the same logic for both parts. Now the process model shown below has a non-additive kind of noise. Therefore we implement UKF of the non-additive kind.

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

3.2.1 Computation of Sigma Points: For dealing with non additive kind we need to find `n',` which can be found as follows:

Code Excerpt:

```
% Initialising values for n, nq,
n_Prime, alpha, k, beta
    n=15;
    nq=12;
    n_Prime = n+nq;  % 27
    alpha = 0.001;
    k = 1;
    beta = 2;
```

where n is the size of the state vector and n_q is the size of the noise vector.

Now to compute the number of sigma points we use the formula;

`2*(n'+1)` = 55 Sigma Points.

We also know that;

$$x \sim N(\mu, \Sigma) \quad q_t \sim N(0, Q_t)$$

And the augmented state matrix is;

$$x_{aug} = \begin{bmatrix} x_{t-1} \\ q_t \end{bmatrix}$$

$$\mu_{aug,t-1} = \begin{bmatrix} \mu_{t-1} \\ 0 \end{bmatrix} \quad P_{aug} = \begin{bmatrix} \Sigma_{t-1} & 0 \\ 0 & Q_t \end{bmatrix}$$

We find our augmented matrix as follows;

$$\mathcal{X}^{(0)}_{aug,t-1} = \mu_{t-1}$$
$$\mathcal{X}^{(i)}_{aug,t-1} = \mu_{aug,t-1} \pm \sqrt{n' + \lambda'} \left[\sqrt{P_{aug}}\right]_i$$

To Find $\sqrt{P_{aug}}$ we use Cholesky factorization function of MATLAB. Next we find;

$$\lambda' = \alpha^2(n' + k) - n' = -27$$

α and k determine the sigma points spread. For a Gaussian prior distribution, good parameters are α = 0.001 and k = 1.

3.2.2 Propagate the Sigma Points through the non linear function: To get the process model we run the following model in an iterative FOR loop 55 times. To account for uncertainty in the system, we add noise to the output of the process model function. This noise represents the small changes that might happen to the state of the system during a short time interval (dt). We then add this noise to each "sigma point" in a set of 55 points. These adjusted sigma points represent the possible states of the system after the time interval dt.

$$\mathcal{X}^{(i)}_t = f(\mathcal{X}^{(i),x}_{aug,t-1}, u_t, \mathcal{X}^{(i),n}_{aug,t-1})$$
$$i = 0 \dots 2n' \quad \mathcal{X}^{(i)}_{aug,t-1} \begin{bmatrix} \mathcal{X}^{(i),x}_{aug,t-1} \\ \mathcal{X}^{(i),n}_{aug,t-1} \end{bmatrix}$$

3.2.3 Compute the predicted mean and covariance matrix: Here we compute the weights for the mean and covariance separately as follows:

$$W^{(c)'}_0 = \frac{\lambda'}{n' + \lambda'} + (1 - \alpha^2 + \beta)$$
$$W^{(c)'}_i = \frac{\lambda'}{2(n' + \lambda')} = 17857$$
$$W^{(m)'}_0 = \frac{\lambda'}{n' + \lambda'} = 964285$$
$$W^{(m)'}_i = \frac{\lambda'}{2(n' + \lambda')} = 17857$$

Using this we find the estimated mean and covariance with the following equation.

$$\bar{\mu}_t = \sum_{i=0}^{2n'} W^{(m)}_i \mathcal{X}^{(i)}_t$$
$$\bar{\Sigma}_t = \sum_{i=0}^{2n'} W^{(c)}_i \left(\mathcal{X}^{(i)}_t - \bar{\mu}_t\right)\left(\mathcal{X}^{(i)}_t - \bar{\mu}_t\right)^T$$

3.3 *upd_step.m* for Part 1 and Part 2: While the update step for Part 1 itself remains unchanged from Project 1, its inputs differ. Here the update step uses position and orientation data derived from the onboard camera's visual data.

In Part 2, the measurement input comes from the velocity calculated based on the optical flow data. This mathematical model for velocity is non-linear and doesn't inherently account for any errors or uncertainties. To address this, we'll be implementing the Unscented Kalman Filter with Additive Noise. This filter allows us to incorporate noise into the model, even though it's not originally included.

3.3.1 Computation of Sigma Points: For dealing with additive kind we need not find `n'`, instead we only need the state vector n=15 and that gives the sigma points to be 31.

$$\begin{aligned}
&\mathcal{X}^{(0)} = \mu \\
&\mathcal{X}^{(i)} = \mu \pm \sqrt{n+\lambda}[\sqrt{\Sigma}]_i \quad i = 1 \ldots n \\
&\lambda = \alpha^2(n+k) - n
\end{aligned}$$

The sigma points spread are determined by α and k. For a Gaussian prior distribution, good parameters are α = 0.001 and k = 1.

3.3.2 Computation of Sigma Points: Next we'll put the 31 sigma points through a specific non-linear function. Remember, in Part 2, we only considered the velocity data from the optical flow. This data captures both the linear velocity of the camera (how fast it's moving in a straight line) and its angular velocity (how fast it's rotating) relative to the world, all expressed within the camera's frame of reference. By processing each sigma point through this non-linear function, we can estimate how the system's state might change based on this velocity information. Using these two, we form the measurement model function g(x2, x3, $^{B}\omega^{W}{}_{B}$) as shown;

$$g(x_t) = R_B^C\ l(x_2, x_3) - R_B^C\ S(r_{BC}^B)\ R_C^B\ {}^C\omega_C^W$$

3.3.2 Compute mean and covariance matrix: Here we compute the weights for the mean and covariance separately as follows:

$$\begin{aligned}
W_0^{(c)} &= \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\
W_i^{(c)} &= \frac{\lambda}{2(n+\lambda)} = 31250 \\
W_0^{(m)} &= \frac{\lambda}{n+\lambda} = -937499 \\
W_i^{(m)} &= \frac{\lambda}{2(n+\lambda)} = 31250
\end{aligned}$$

Using this we find the current mean and covariance with the following equation.

$$\begin{aligned}
z_{\mu,t} &= \sum_{i=0}^{2n} W_i^{(m)} Z_t^{(i)} \\
C_t &= \sum_{i=0}^{2n} W_i^{(c)} (\mathcal{X}_t^{(i)} - \bar{\mu}_t)(Z_t^{(i)} - z_{\mu,t})^T \\
S_t &= \sum_{i=0}^{2n} W_i^{(c)} (Z_t^{(i)} - z_{\mu,t})(Z_t^{(i)} - z_{\mu,t})^T + R_t
\end{aligned}$$

$$\begin{aligned}
\mu_t &= \bar{\mu}_t + K_t(z_t - z_{\mu,t}) \\
\Sigma_t &= \bar{\Sigma}_t - K_t S_t K_t^T \\
K_t &= C_t S_t^- 1
\end{aligned}$$

These equations, when coded in MATLAB, represent the update step of the Unscented Kalman Filter with additive noise. Combining this update step with the prediction step discussed earlier gives us a complete UKF implementation. This powerful filter can then be used to estimate the position, orientation, and linear velocity of a quadrotor based on data from its onboard sensors (IMU and camera). Essentially, the UKF takes the initial state estimate (prediction step), incorporates new sensor measurements (update step with noise), and provides a more accurate picture of the quadrotor's state.

4. Results

Our implementation of the Unscented Kalman Filter (UKF) proved successful in estimating the state of the Micro Aerial Vehicle (MAV) in both setups we tested.

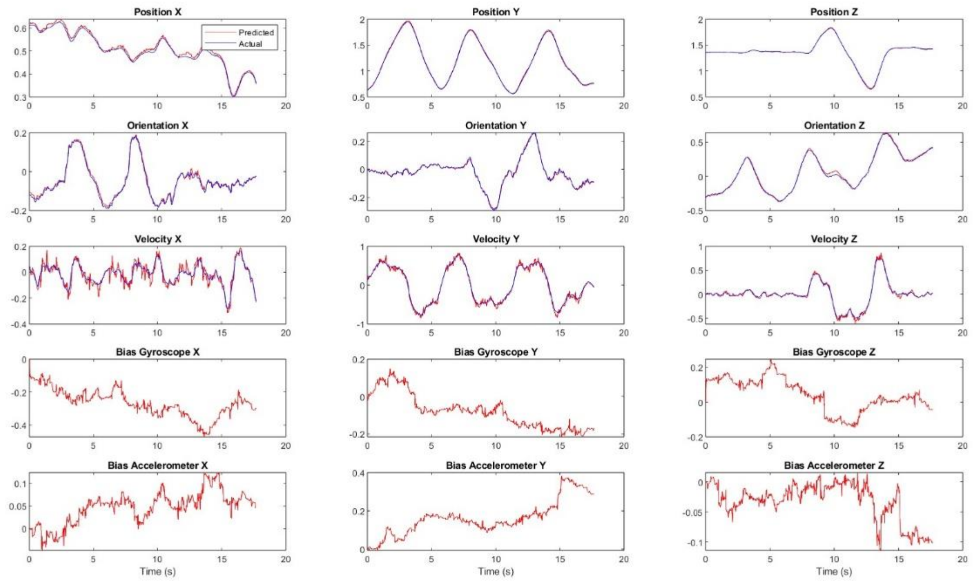

Fig4.1 Part 1 Dataset1

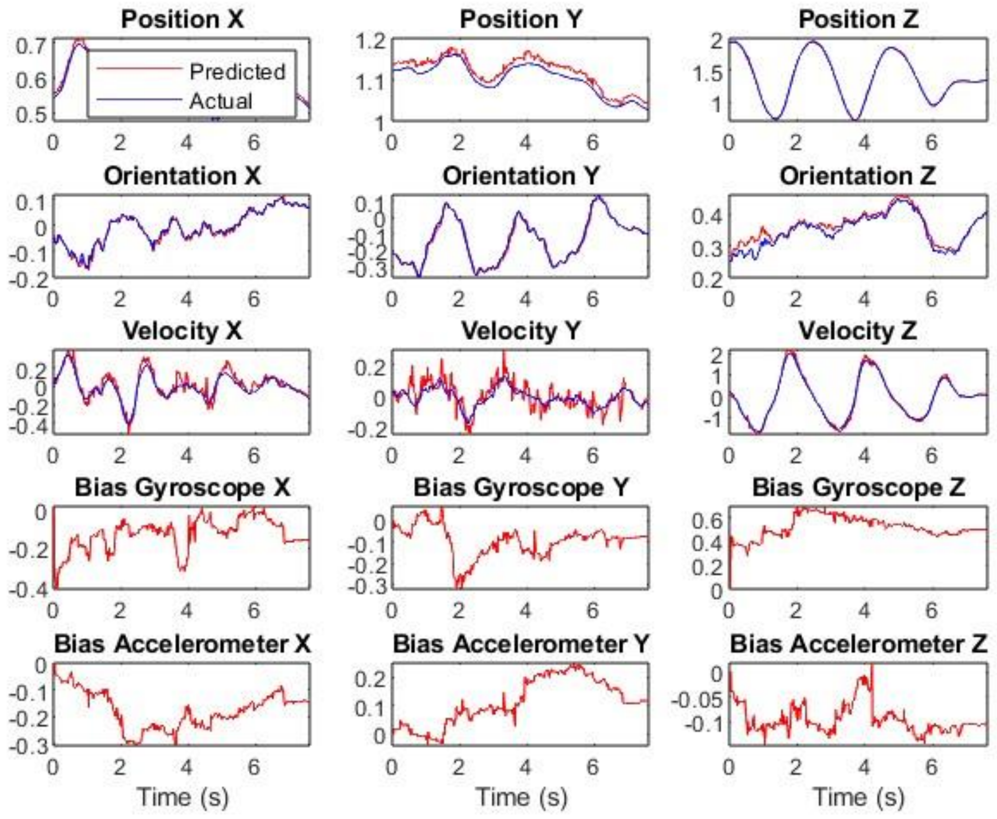

Fig4.2 Part 1 Dataset4

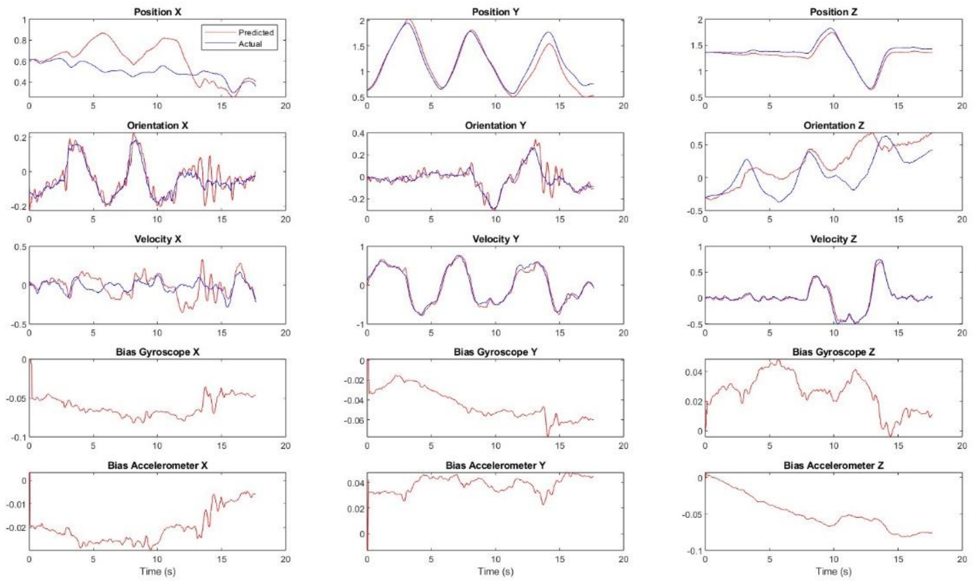

Fig4.3 Part 2 Dataset1

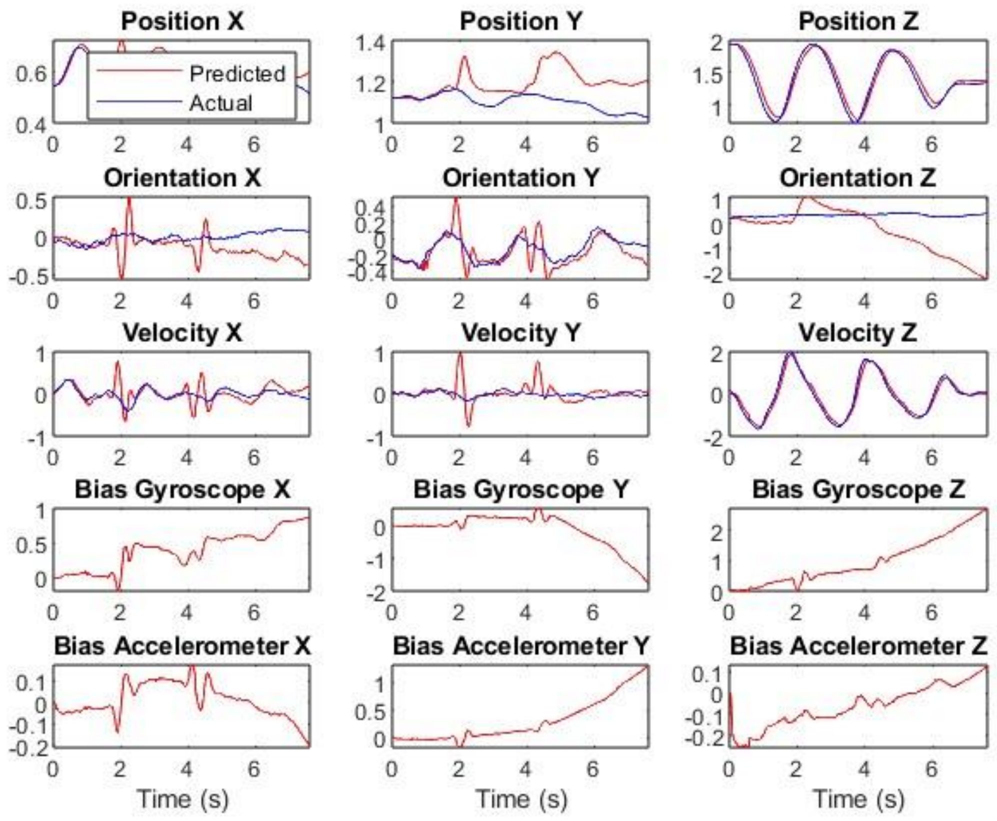

Fig4.4 Part 2 Dataset4