Orientation Estimation using Unscented Kalman Filter

Sai Krishnan Chandrasekar
Graduate Student, Robotics
University of Pennsylvania
Email: saikrish@seas.upenn.edu

Abstract: *Sensor fusion and filtering of readings from drifty sensors are two important aspects of working with multiple sensors. Kalman filtering is a widely used algorithm that takes data from different sensors, filters it and returns a more accurate reading by fusing data from all the sensors. In this project, an Unscented Kalman Filter (UKF) was designed and implemented to track the orientation of a camera. Once the orientation of the camera was identified, the goal was to rotate the camera images accordingly and stitch them to form a panorama.*

1 Introduction

The main goal of this project was to estimate the orientation of a camera. A Kalman Filter is generally used for such applications. It is used to improve the accuracy of the orientation estimation system on the whole by fusing data from all the sensors available at hand. However, Kalman Filters work only for linear systems. Instead of linearzing the system, an Unscented Kalman Filter was used. An UKF estimates the motion by computing the ”sigma points” for the motion model and uses these to track the sensor inputs. Once the orientation of the camera was identified, the goal was to rotate the camera images accordingly and stitch them to form a panorama.

The data provided was: VICON estimates, Intertial Measurement Unit values and Camera images. Using the Gyroscope and the Accelerometer values from the IMU, an Unscented Kalman Filter model was learned. This was used to predict the orientation of the camera at each given time step. The VICON estimates were used as ground truth to verify the predicted states and track the accuracy of the UKF. This was coded up on Python and OpenCV libary was used for image processing.

2 Problem Formulation

The problem at hand was divided into 5 parts:

1. Preprocessing of data
2. Quaternion Helper Functions
3. UKF
4. Orientation Estimation
5. Panorama Stitching

3 Pre-Processing of data

3.1 IMU

: The IMU data consisted of 2 parts. Accelerometer values and Gyroscope values. These values are provided as raw outputs from the sensors. These were calibrated using the formulae:

$$value = (raw - bias) * scalefactor$$

Where,

$$scale_factor = V_{ref}/1023 * sensitivity$$

The bias values were estimated in such a way that the initial accelerometer value read: [0,0,0] and the initial gyro value read: [0,0,1]. These values were chosen because the initial state is rest.

3.2 VICON

: The VICON data was 3d Rotation matrices at each time step. This was obtained using the VICON motion capture system that computes the orientation of the system using motion capture of reflective markers placed on the system that is being tracked. This data is as close to ground truth as possible. The given rotation matrices were converted into euler angles.

3.3 Camera

: The camera images didn’t require any pre-processing.

3.4 Time Synchronization

: The timestamps on the data from IMU, VICON and Camera were all out of sync. The timestamps had to be synchronized. This was crucial for VICON and Camera data or

IMU and Camera data as the images had to undergo the corresponding rotation in order to be stitched together for the final panorama. This was done by finding the nearest timestamp in the VICON data or the IMU data for the corresponding camera image. The corresponding rotation matrix was applied on the image to rotate it to world coordinates.

4 Quaternion Helper Functions:

All the measurements from IMU were converted into quaternions in order to eliminate singularities. This required coding up a lot of quaternion functions, the most important of them being a quaternion averaging function. This function computed a weighted average of multiple quaternions. This was used in the Kalman Prediction step to average the sigma points (Explained below).

5 UKF

An Extended Kalman Filter is, as the name suggests, an extension of the Kalman Filter for non linear systems. It approximates the state distribution by using a Gaussian random variable which is propagated through the linearized model of the system. An UKF also propagates a gaussian variable through the system but it uses a more deterministic sampling approach. It chooses a set of ”sigma points” at each step that capture the mean and covariance of the data as accurately as possible. These sigma points are few in number but since they’re carefully chosen, we do not miss out on the overall structure of the data. This makes the UKF very efficient and accurate as well.

In this project, the data that the UKF was used over was the IMU measurements: Acceleration from the Accelerometer and Angular velocity from the Gyroscope. A 4 state UKF with the gyro measurements as control inputs was used. This resulted in good performance, so a 7 state UKF was not implemented.

A high level overview of the UKF is described below:

5.1 Kalman Prediction:

The Kalman prediction function takes in the following inputs:

The current state: q_t
The control input: u_t
Covariance of the data: P_t
Uncertainty or noise: Q

The Kalman prediction function returns the following:

Predicted state for the next time step: q_{t+1}
Predicted Covariance for the next time step: P_{t+1}

These predictions are given in as inputs to the update step.

5.1.1 Sigma Points:

The key idea behind the UKF are the sigma points. These sigma points are carefully chosen such that they capture the full structure of the data. These points are obtained by Cholesky Decomposition of the total uncertainty. In other words, the sum of the covariance matrix and the noise.

The decomposition of P+Q is multiplied by $\sqrt{2n}$ and $-\sqrt{2n}$ where n is the number of dimensions, in this case: 3. This results in six 4×1 quaternions. These represent the sigma points.

5.1.2 Apply the Motion Model:

The motion model is applied to these six sigma points. This effectively transforms them from their current state to the next state, i.e., the state of the system after the motion has occured. This is achieved by multiplying the sigma points with the control input u_t.

5.1.3 Quaternion Averaging:

The resulting six quaternions are averaged to produced one average quaternion that represents the mean of the new structure of the data. This is the predicted state: q_{t+1}.

5.1.4 Compute New Covariance:

The new covariance is computed using the error obtained from the quaternion averaging step. The quaternion averaging results in some error as it only tries to minimize the error, but cannot produce a perfect average. This results in the new predicted Covariance: P_{t+1}.

5.2 Kalman Update:

The Kalman Update step takes in the following inputs:

Initial state estimate: g
sigma points
Predicted mean from Kalman Predict step: q_{t+1}
Predicted Covariance from Kalman Predict step: P_{t+1}
Sigma Points
Accelerometer Input

The Kalman Update Step outputs the following:

Updated mean state: q_u
Updated Covariance: P_u

These are passed in as inputs to the predict step at the next time step. Thus it’s a cycle of:
Kalman Predict $\rightarrow$ Kalman Update $\rightarrow$ Kalman Predict

5.2.1 Get new sigma points

New sigma points are obtained by rotating the quaternion g. This is denoted by z Their average z_m is calculated.

5.2.2 Multiple covariances:

z and z_m are used to calculate multiple covariances. P_{zz}: Measurement Covariance. P_{xz}: Cross covariance between the measurement and the covariance from the update step. P_{vv}: Total Uncertainty in the update step. This is $P_{xz} + R$.

5.2.3 Kalman Gain and Innovation term:

The crux of any Kalman filter is the Kalman Gain. The Kalman Gain determines "how much" of an update should take place in this step. This is calculated using the formula:

$$K = P_{xz}P_{vv}^{-1}$$

The innovation term, I, is the difference between the accelerometer input and the average of the new sigma points.

$$I = Acc_i - z_m$$

5.2.4 Updated Q and Updated P

:

The updated state q_u is calculated as a product of K, I and the predicted q, q_i:

$$q_u = K * I * q_i$$

Similarly the updated covariance is:

$$P_u = P_i - K * P_{vv} * K^T$$

These are sent in as inputs to the Kalman predict function at the next stage.

6 Orientation Estimation:

The estimated states are the orientations of the state in the form of quaternions. These were converted into rotation matrices and into euler angles. The rotation matrices were used in the next step (Panorama Stitching) while the euler angles were used to track the progress of the Kalman Filter. The graphs are shown below. The first 9 are the training results. The last 4 are the testing results.

7 Results:

The results shown below are for the 9 training datasets. **The blue indicates VICON data. The Red indicates UKF data.**

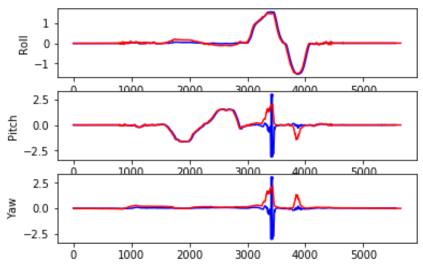

Fig. 1. UKF estimate for Dataset 1.png

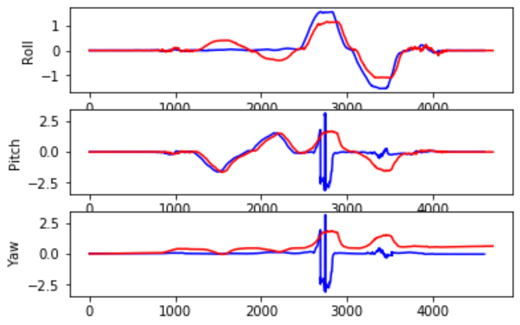

Fig. 2. UKF estimate for Dataset 2.png

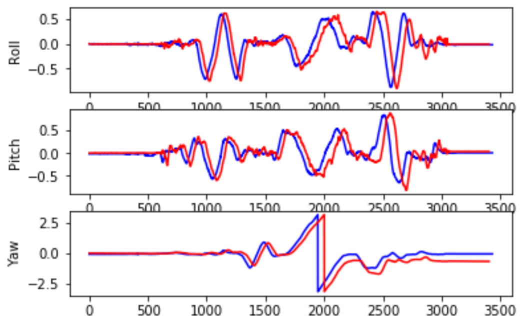

Fig. 3. UKF estimate for Dataset 3.png

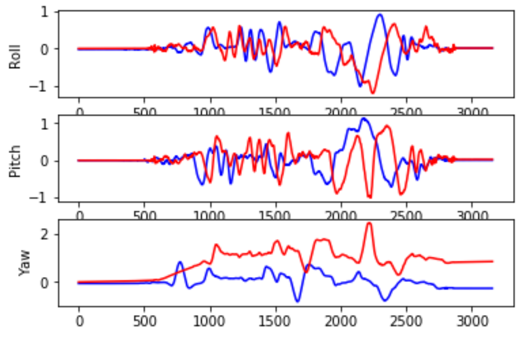

Fig. 4. UKF estimate for Dataset 4.png

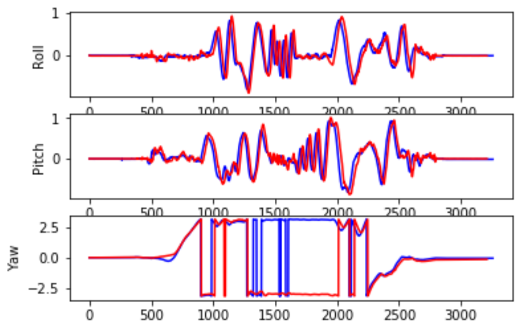

Fig. 5. UKF estimate for Dataset 5.png

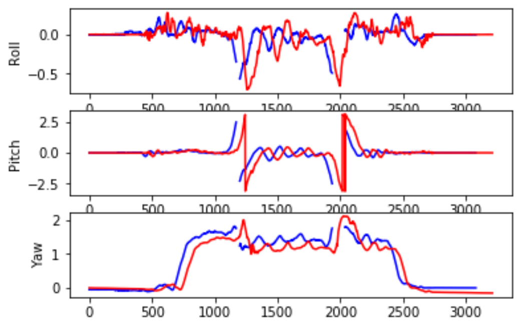

Fig. 6. UKF estimate for Dataset 6.png

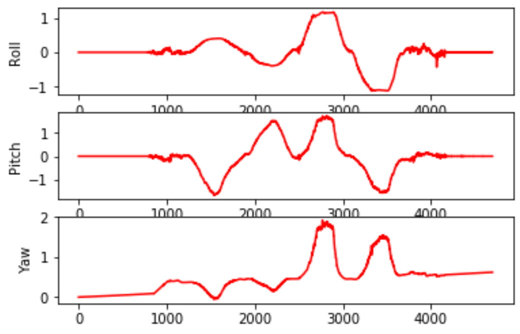

Fig. 10. UKF estimate for Dataset 10.png

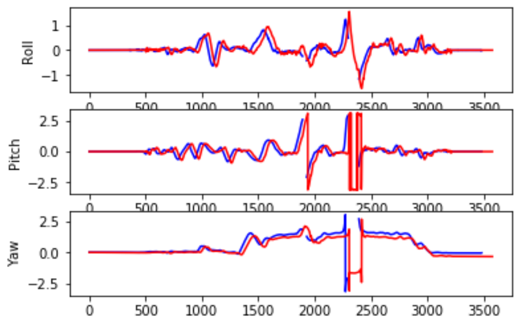

Fig. 7. UKF estimate for Dataset 7.png

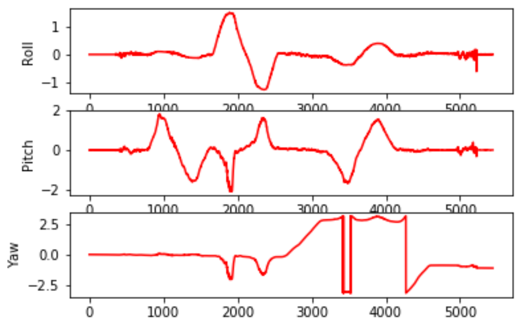

Fig. 11. UKF estimate for Dataset 11.png

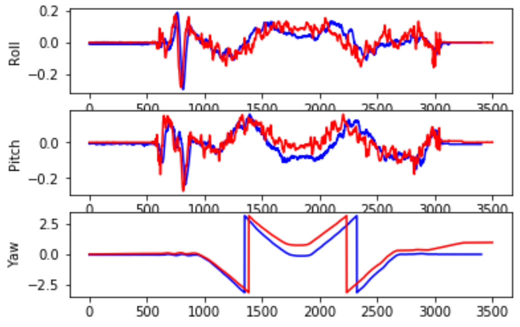

Fig. 8. UKF estimate for Dataset 8.png

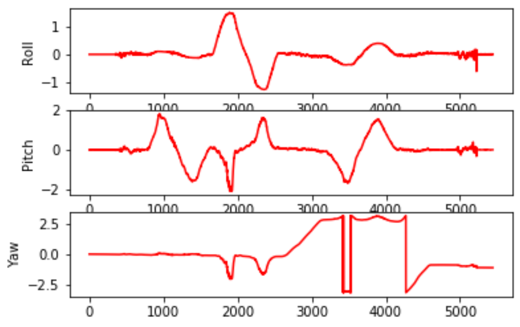

Fig. 12. UKF estimate for Dataset 12.png

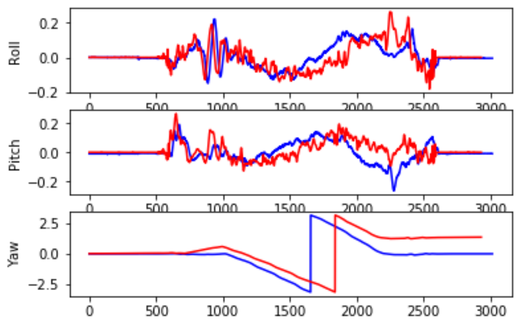

Fig. 9. UKF estimate for Dataset 9.png

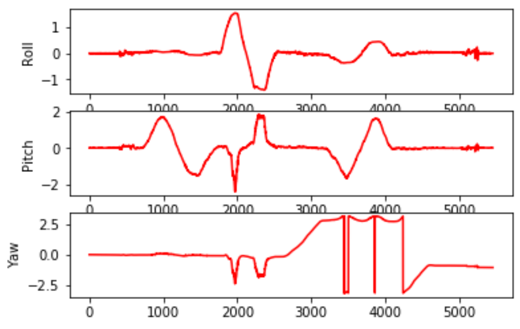

Fig. 13. UKF estimate for Dataset 13.png

8 Panorama Stitching:

This step was implemented partially, but the results weren't good enough. The stitching was performed using a few basic inbuilt functions but largely using codes written by me. The methodology is listed below:

8.1 Centering the image:

The new positions of each pixel with respect to the image with the reference point at the centre of the image was calculated.

8.2 Estimate Latitude, Longitude and Radius:

Using the horizontal and vertical field of view and the centred pixel values, the latitude and longitude of each pixel of the image was calculated. The radius was assumed to be one. This essentially converts the image into a sphere.

8.3 Convert to world cartestian coordinates

The resulting spherical coordinates are converted to cartesian coordinates. The corresponding rotation matrix to that image is obtained by time synchronization of the VICON or the UKF data (Rotation matrix computed from IMU). The cartesian coordinates are rotated using the rotation matrix to obtain the world cartesian coordinates.

8.4 Cylindrical Projection

These are transformed to spherical coordinates and then projected onto a cylinder for cylindrical warping. This is done by computing the height and azimuth angle of the pixel in the cylindrical space from the latitude and longitude obtained by converting the world cartesian coordinates into spherical coordinates. The depth of the cylinder is assumed to be 1m.

8.5 Unwrap the Cylinder

The cylinder is unwrapped to a rectangular image with height π radians and width 2π radians. The corresponding image's pixel values are transferred to the locations on the final image.

8.6 Stitching results:

NOTE: Unfortunately, I had a bug in my code and hence was unable to transfer the pixel values to the correct locations. Hence I'm not attaching the outputs of the stitched Panoramas. I'm however, attaching my panorama stitching code as a separate folder. I do not expect credit for this, but merely wanted to show that I have made an attempt to complete the stitching process.

8.7 Things to improve

1. Tuning the initial estimates: The initial estimates for P, Q, R and g determined the UKF's performance greatly. Similarly the initial estimate for the average quaternion in quaternion averaging determined the final output as well. These could have been tuned better.
2. Do not trust Gyro Values too much: The Gyro values become steadily more untrustworthy as time passes. This is because gyroscopes drift with time. I made the mistake of assigning a larger R value than P and Q and in other words, ended by trusting the gyro values more than the accelerometer values. This can be seen in the plots as the UKF estimate starts to worsen over time. When I reduced the value of R, the results improved.
3. Learning algorithm for parameters: A learning algorithm to learn the optimal initial states for the parameters might be effective. Or at least a learning algorithm that initialises better values for the future datasets based on the previous datasets might improve accuracy.
4. Read the instructions carefully: I did not read the part of the instructions which said the code should be able to run if VICON data wasn't passed to it. Hence, my code requires the VICON data to run (I plot the VICON data).

9 Conclusion

The aim of the project was to estimate the orientation of the system using the IMU and VICON readings. This was done using an UKF as explained above. The UKF performed well, with a speed of approximately **70 Hz** per timestep. With further optimization, this can be effectively used for real time tracking of estimation and for further improving the stability of the system by feeding the UKF model to the controller that controls the system.

The panorama stitching was performed using OpenCV and Numpy packages. The orientation estimates seem good and once the bug (mentioned above) is fixed, the results should look good.

This project was a great learning experience. Kalman Filters have been covered in theory in a lot of places but this project was a great platform to see it work in practice.