Orientation Estimation with Panoramic Image Stitching using an Unscented Kalman Filter

Alexander Khoury
A11563298
15 November 2017

I. INTRODUCTION

Many of the technological advancements we take for granted today, utilize some form of orientation estimation. Whether it is as simple as tilting our phone sideways to get a better look at a picture or document, or even something more complicated, like a drone stabilizing itself in the air, robust orientation estimation is needed. Sensors that measure how a device is moving in space can be very noisy or even drift, and simple estimation techniques may not be enough for certain applications. In this paper, we discuss an algorithm for non-linear orientation estimation for an 6-DOF IMU mounted on a camera. We apply the orientation estimates to images taken at those orientations to generate a panorama of the environment around it.

II. PROBLEM FORMULATION

Using data from an Inertial Measurement Unit (IMU), implement an Unscented Kalman Filter to track the 3-D orientation of a rotating camera. Then use the estimated orientations construct a panorama from a dataset of images collected.

- **Unscented Kalman Filter using data from IMU**: Remove the bias and scale factors from the IMU data. Then implement a quaternion-based UKF based on Edgar Kraft's paper.
- **Image Stitching into Panorama:** Use the estimated orientations to project the associated images onto a unit sphere. Then unwrap the image sphere to create a panoramic image.

III. TECHNICAL APPROACH

A. Sensor Calibration

An IMU is a sensor that measures accelerations and rotational velocities along/around all three Cartesian axes.

$$\begin{aligned} &Accelerations\ (g):\ [a_x, a_y, a_z] \\ &Rotational\ Velocities\ (rads/sec):\ [w_x, w_y, w_z] \end{aligned} \tag{1}$$

The sensor reports its values using an ADC, so the values must be converted to physical units using the given conversion formulas.

$$\begin{aligned} value\ &=\ (raw\ data\ -\ bias)\cdot scale\ factor + offset \\ scale\ factor &= \frac{Vref}{1023\cdot sensivity} \end{aligned} \tag{2}$$

Sensor bias can be calculated by taking the average of the first 2 seconds of data, where we assume the IMU to be steady.

Offset is 0 for all measurements except a_z where it is 1. This is because if the IMU is not moving and level, we should be seeing $1g$ in a_z.

Sensitivity values and voltage reference values can be found in the datasheet for the sensors.

$$\begin{aligned} &gyroscope\ sensitivity = 3.33 * 180/\pi \\ &accelerometer\ sensitivity = 330 \\ &Vref = 3300\ mV \end{aligned} \tag{3}$$

B. Unscented Kalman Filter: Background

To implement the Kalman filter, or any non-linear variant, we require knowledge on how to predict the next state given the current state. This relationship is oftentimes called the motion model. The motion model is often times derived from the kinematics of the system, where it is a function of the current state and control input. A generalized motion model equation can be written as:

$$x_{t+1|t} = a(x_{t|t}, u_t, \mathrm{w})$$

where $x_{t|t}$ represents the state at time t thus $t+1$ is the next state. The motion model is then some function $a(\cdot)$ that predicts the next state from the previous state as well as the control input u_t. The notation $t+1|t$ denotes the state $t+1$ using sensor information up to time t. We also assume that this prediction does not perfectly predict the next state, thus we describe this uncertainty in the form of the process noise w.

The Kalman filter works on a two step calculation, predict and update. As mentioned before, the motion model is used to predict the state $x_{t|t}$ to the next, $x_{t+1|t}$.

Subsequently the update step takes that prediction and corrects it with a measurements taken from sensors. Thus, a model corresponding to our sensors is needed, aptly named the observation model:

$$x_{t+1|t+1} = h(x_{t+1|t}, \mathrm{v})$$

The function $h(\cdot)$ describes the transformation from our predicted state $x_{t+1|t}$ to our updated state $x_{t+1|t+1}$. Again

we assume our measurement from our sensor is noisy, thus adding a term modeling this noise v.

Important assumptions are made to derive a non-linear filter such as the UKF:

- The prior pdf of $x_{0|0}$ is Gaussian.
- The motion model noise, process noise, w is Gaussian.
- The observation model noise, measurement noise, v is Gaussian.
- The process noise w and measurement noise v are independent of each other and of the state x_t.
- The posterior of the prediction step of the state $x_{t+1|t+1}$ is forced to be Gaussian via approximation (moment matching).

The prior, process noise, and measurement noise can then be modeled as Gaussian:

$$\begin{aligned} x_{t|t} &\sim \mathcal{N}(\mu_{t|t}, \Sigma_{t|t}) \\ \mathrm{w} &\sim \mathcal{N}(0, W) \\ \mathrm{v} &\sim \mathcal{N}(0, V) \end{aligned} \tag{4}$$

In a non-linear filter, the predicted distribution are no longer guaranteed Gaussian, therefore, to prevent the distribution from becoming more and more complicated through iterations, we approximate the distribution as Gaussian, via moment matching. To achieve this, we calculate the first two moments of the resulting distribution, and replace it with a Gaussian distribution with those same two moments. In order to do this, five integrals must be computed corresponding to the moments of the joint distribution $(x_{t+1|t}, z_{t+1})$:

$$\begin{pmatrix} x_{t+1|t} \\ z_{t+1} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_{t+1|t} \\ m_{t+1|t} \end{pmatrix}, \begin{bmatrix} \Sigma_{t+1|t} & C_{t+1|t} \\ C_{t+1|t}^T & S_{t+1|t} \end{bmatrix} \right)$$

$$\begin{aligned} \mu_{t+1|t} &= \int\int a(x, u_t, \mathrm{w})\Phi(x; \mu_{t|t}, \Sigma_{t|t})\Phi(w; 0, W)dxdw \\ m_{t+1|t} &= \int\int h(x, \mathrm{v})\Phi(x; \mu_{t|t}, \Sigma_{t|t})\Phi(v; 0, V)dxdv \\ \Sigma_{t+1|t} &= \int\int a(x, u_t, \mathrm{w})a(x, u_t, \mathrm{w})^T \\ &\qquad \cdot \Phi(x; \mu_{t|t}, \Sigma_{t|t})\Phi(w; 0, W)dxdw - \mu_{t|t} \cdot \mu_{t|t}^T \\ C_{t+1|t} &= \int\int (a(x, u_t, \mathrm{w}) - \mu_{t|t})(h(x, \mathrm{v}) - m_{t|t})^T \\ &\qquad \cdot \Phi(x; \mu_{t|t}, \Sigma_{t|t})\Phi(v; 0, V)dxdv \\ S_{t+1|t} &= \int\int h(x, \mathrm{v})h(x, \mathrm{v})^T\Phi(x; \mu_{t|t}, \Sigma_{t|t})\Phi(v; 0, V)dxdw \\ &\qquad - m_{t|t} \cdot m_{t|t}^T \end{aligned} \tag{5}$$

Where $\Phi(x; \mu_{t|t}, \Sigma_{t|t})$ is the Gaussian distribution of x with mean $\mu_{t|t}$ and covariance $\Sigma_{t|t}$. Then the conditional Gaussian distribution of $(x_{t+1|t+1}|z_{t+1})$, or updated distribution, can be computed from these moments:

$$\begin{aligned} K_{t+1|t} &= C_{t+1|t}S_{t+1|t}^{-1} \\ \mu_{t+1|t+1} &= \mu_{t+1|t} + K_{t+1|t}(z_{t+1} - m_{t+1|t}) \\ \Sigma_{t+1|t+1} &= \Sigma_{t+1|t} - K_{t+1|t}S_{t+1|t}K_{t+1|t} \end{aligned} \tag{6}$$

With non-linear models, it can easily be seen that these integrals can become quite complex, or even unsolvable. Thus, we once again must turn to approximations.

There are two mainstream methods of approximating these moment integrals, the
Extended Kalman Filter (EKF), and the **Unscented Kalman Filter** (UKF). The Extended Kalman Filter uses **first-order Taylor series approximations** around a nominal state to linearize the motion and observation models, making the integrals straightforward to compute. However this is highly dependent on how closely the linearization approximates the models. If this approximation is not accurate, the EKF will not produce good results.

In these cases, the UKF outperforms the EKF, because it is able to use the non-linear models without approximation to propagate states. To do this, the UKF utilizes the **unscented transform**. In the unscented transform, $2d+1$ sigma points are calculated, where d is the dimension of the state. The sigma points are chosen to capture the shape of the original distribution, and the advance they hold over linearization is that they can be propagated through the nonlinear models, and greatly simplify the integrals discussed for moment matching. Instead of having to calculate the integrals for the all the points in the distribution, only the weighted sums of the integrands over the sigma points need to be calculated (this will be more clearly defined later).

The first sigma point is chosen to be the mean, and the rest of sigma points are calculated using the mean-shifted columns of the square root covariance of the state:

$$\mathcal{X}^{(0)} = \mu_{t|t}, \;\; \mathcal{X}^{(i)} = \mu_{t|t} \pm \beta{\cdot}d{\cdot}columns(\sqrt{\Sigma_{t|t}}) \;\; for \;\; i = 1 \cdots 2d \tag{7}$$

Where $\sqrt{\Sigma_{t|t}}$ can be calculated using the Cholesky Decomposition, $\Sigma_{t|t} = \sqrt{\Sigma_{t|t}}\sqrt{\Sigma_{t|t}}^T$, and β represents a tunable scaling factor that is applied to said columns, determining spread. A good starting point is $\beta = 1$.

Further assuming noise is additive, the five moment integrals can then be rewritten using the sigma points $\mathcal{X}^{(i)}$:

$$\text{Weights: } \; W_0^{(m)} = 0, \quad W_0^{(c)} = 2, \quad W_i^{(m)} = W_i^{(c)} = \frac{1}{2d}$$

$$\begin{aligned} \mu_{t+1|t} &= \sum_{i=0}^{2d} W_i^{(m)} a(\mathcal{X}_{t|t}^{(i)}, u_t) \\ m_{t+1|t} &= \sum_{i=0}^{2d} W_i^{(m)} h(\mathcal{X}_{t+1|t}^{(i)}) \\ \Sigma_{t+1|t} &= \sum_{i=0}^{2d} W_i^{(c)} (a(\mathcal{X}_{t|t}^{(i)}, u_t) - \mu_{t+1|t})(a(\mathcal{X}_{t|t}^{(i)}, u_t) - \mu_{t+1|t})^T \\ C_{t+1|t} &= \sum_{i=0}^{2d} W_i^{(c)} (a(\mathcal{X}_{t+1|t}^{(i)}, u_t) - \mu_{t|t})(h(\mathcal{X}_{t+1|t}^{(i)}) - m_{t|t})^T \\ S_{t+1|t} &= \sum_{i=0}^{2d} W_i^{(c)} (h(\mathcal{X}_{t+1|t}^{(i)}) - m_{t+1|t})(h(\mathcal{X}_{t+1|t}^{(i)}) - m_{t+1|t})^T + V \end{aligned} \tag{8}$$

Thus if we specify the motion and observations models $a(\mathcal{X}_{t|t}^{(i)}, u_t)$ and $h(\mathcal{X}_{t+1|t}^{(i)})$, we can implement our UKF using equations (6) and (7).

C. Unscented Kalman Filter: Implementation

1) Quaternion Math:

Orientation will be defined as a quaternion q, where $q = q_0 + q_1 i + q_2 j + q_3 k$, or is often seen as a scalar and vector component $[q_s, q_v] = [q_0, [q_1, q_2, q_3]]$. Quaternions do not suffer from gimbal lock, which is ideal for tracking orientation of any body with complex movements. In order to use quaternions in our algorithm, simple mathematics need to be defined for quaternions. Let q and p be quaternions, with scalar components q_s, p_s and vector components q_v, p_v:

$$\begin{aligned}
&\text{Addition:} \quad q + p = [q_s + p_s, q_v + p_v] \\
&\text{Multiplication:} \quad q \circ p = [q_s p_s - q_v^T p_v, q_s p_v + p_s q_v + q_v \times p_v] \\
&\text{Conjugate:} \quad \bar{q} = [q_s, -q_v] \\
&\text{Norm:} \quad |q| = \sqrt{q_s^2 + q_v^T q_v} \\
&\text{Inverse:} \quad q^{-1} = \frac{\bar{q}}{|q|^2} \\
&\text{Rotation:} \quad [0, x'] = q \circ [0, x] \circ q^{-1} = [0, R(q)x] \\
&\text{Exp Map:} \quad \exp(q) = e_s^q \cdot [\cos||q_v||, \frac{q_v}{||q_v||} \sin||q_v||] \\
&\text{Log Map:} \quad \log(q) = [\log|q|, \frac{q_v}{||q_v||} \arccos\frac{q_s}{|q|}]
\end{aligned} \tag{9}$$

Note that exp map can construct a quaternion from a rotation vector w: $q = \exp([0, \frac{w}{2}])$. And the rotation vector can be recovered using Log map, $[0, w] = 2 \cdot \log(q)$. We will use these two to convert to and from quaternions to simplify calculations.

2) State Definition:

The 7-dimensional state vector of our Kalman filter will consist of a quaternion, $q_t \epsilon \mathbf{R}^4$ that represents the orientation of the state, and the angular velocity $w_t \epsilon \mathbf{R}^3$.

$$x_t = \begin{pmatrix} q_t \\ w_t \end{pmatrix} \epsilon \mathbf{R}^7$$

where q_t is a unit quaternion with components $[q_{t,s}, [q_{t,v}]]$ and the three components of the angular velocity w_t are $[w_{t,x}, w_{t,y}, w_{t,z}]$.

Recall that with the UKF motion and observation models, we assume that there is process noise w and measurement noise v is affecting the system. And as a result to the moment matching, the probability density function of our state is also be Gaussian. Thus, the covariance matrices $\Sigma_{0|0}$, W, V for the Gaussian models can be initialized to $\gamma \cdot I$, where γ can be tuned to increase accuracy of the estimate. A good starting vale Also as per the problem, we can assume our system is relatively level at time t = 0, so we can initialize our state to $q_0 = [1, 0, 0, 0]$ and $w_0 = [0, 0, 0]$.

3) Motion Model:

The motion model for the seven dimensional state vector $\mathcal{X}_{t+1|t}^{(i)}$:

$$\mathcal{X}_{t+1|t}^{(i)} = a(\mathcal{X}_{t|t}^{(i)}, u_t) = \begin{pmatrix} q_{t|t}^{(i)} \circ q_{\mathrm{w}} \circ q_{u_t} \\ w_{t|t} + \mathrm{w}_w \end{pmatrix} \tag{10}$$

where $\circ$ represents quaternion multiplication. w_w represents the noise in w, and q_{w} represents the noise affecting $q_{t|t}^{(i)}$. $q_{t|t}^{(i)}$ is the quaternion portion of the state $\mathcal{X}_{t|t}^{(i)}$, and $w_{t|t}$ is the rotational velocity portion of the state. q_{u_t} is the control input affecting the state, which is defined as the increment in orientation calculated using the differential rotational velocity from the last time step.

$$q_{u_t} = \exp\left([0, \frac{w_{t|t}}{2}]\right)$$

where $\exp(\cdot)$ represents the quaternion exponential map that constructs a quaternion from a rotational vector.

Because the noise in the motion model is defined additively, we can instead model the noise during the calculation of the sigma points,

$$\mathcal{X}^{(0)} = \mu_{t|t}, \quad \mathcal{X}^{(i)} = \mu_{t|t} \pm \beta \cdot d \cdot columns(\sqrt{\Sigma_{t|t} + W}) \tag{11}$$

where W is the covariance of the process noise w. Further explained in section 5), our dimension d will be 6 instead of 7 because our covariance matrices are 6×6. This simplifies our model to:

$$\mathcal{X}_{t+1|t}^{(i)} = a(\mathcal{X}_{t|t}^{(i)}, u_t) = \begin{pmatrix} q_{t|t}^{(i)} \circ q_{u_t} \\ w_{t|t} \end{pmatrix} \tag{12}$$

4) Observation Model:

$$\mathcal{Z}_{t+1}^{(i)} = h(\mathcal{X}_{t+1|t}^{(i)}) = \begin{pmatrix} \bar{q}_{t+1|t}^{(i)} \circ [0, g] \circ q_{t+1|t}^{(i)} \\ w_{t+1|t} \end{pmatrix} \tag{13}$$

where g is a vector corresponding to gravity in our system. Thus $[0, g] = [0, [g_x, g_y, g_z]] = [0, [0, 0, 1]]$. $\bar{q}_{t+1|t}^{(i)}$ is the conjugate of the quaternion $q_{t+1|t}^{(i)}$. The quaternion component of the observation model corresponds to rotating a gravity vector by all of our sigma points, which then become our predicted measured value from the accelerometer. Only the vector component of the resulting quaternion is taken, giving us a predicted measurement with dimensions 6×1.

5) Computing Covariance with a Quaternion State:

Computing the covariance given a quaternion state can be computed by constructing a rotation vector from the quaternion with the log map, and using that rotation vector instead of the quaternion in the state. Therefore, the covariance equations in (7) can be computed with a 6-dimensional state, instead of 7. Therefore, all covariance matrices will be $\mathcal{R}^{6\times6}$.

6) Quaternion Averaging:

Calculating the mean $\mu_{t+1|t}$ in equation (7) is non-trivial, as it corresponds to taking the weighted average of $2d+1$ quaternion vectors, which cannot be computed with traditional methods.

Consider a group of n quaternions $[q_i]_{i=1}^n$ with corresponding weights $[\alpha_i]_{i=1}^n$. A weighted quaternion average can be computed with the pseudocode in Algortihm 1, where $\tilde{q}$ is the previous average estimate:

Algorithm 1 Weighted Quaternion Average

$\tilde{q}_{k=0} \leftarrow$ previous quaternion estimate from UKF
for $k = 0, 1, \cdots K$:
  # compute quaternion error between $\tilde{q}_k$ and $[q_i]_{i=1}^n$
  $q_i^e = [q_{s,i}^e,\ q_{v,i}^e] = \tilde{q}_k^{-1} \circ q_i$
  # compute the error rotation vector with log map
  $[0,\ e_{v,i}] = 2 \cdot \log(q_i^e)$
  # restrict angles to $[-\pi, \pi)$
  $e_{v,i} = (-\pi + \text{mod}(||e_{v,i}|| + \pi, 2\pi)) * \frac{e_{v,i}}{||e_{v,i}||}$
  # take the weighted average of the error vectors
  $e_v = \sum_{i=1}^n \alpha_i e_{v,i}$
  # shift average estimate towards true average w/ exp map
  $\tilde{q}_{k+1} = \tilde{q}_k \circ \exp([0,\ \frac{e_v}{2}])$
  # define average convergence based on norm of e_v
  if $||e_v|| < \epsilon$:
    return $\tilde{q}_{k+1}$

7) Update Step with Quaternions:

In order to update the quaternion portion of the state, the update equation of (6), is different for the quaternion part:

$$\hat{q}_{t+1|t+1} = \hat{q}_{t+1|t} \circ \exp([0, \frac{1}{2} K_{t+1|t}(z_{t+1} - m_{t+1|t})])$$

D. Panoramic Image Stitching

1) Time Matching

The camera and the IMU report their data at different rates, such that there are less images than IMU data. In order to compensate for this, the orientation with time-stamp closest to each image was used.

2) Convert Index Array to Spherical Coordinates

The main idea to this algorithm is to project the images onto the unit sphere based on its orientation. To do this, we can calculate what a unrotated index array will correspond to in the unit sphere, using information about the FOV of the camera. These coordinates take the form $(r = 1, \theta, \phi)$.

3) Convert Index Array to Cartesian

Next, we can convert these locations to 3-D Cartesian coordinates, such that we can apply the rotation transformations. These equations take us from Spherical to Cartesian coordinates $(r, \theta, \phi) \rightarrow (x, y, z)$

$$\begin{aligned} x &= r \cdot \sin(\phi)\cos(\theta) \\ y &= r \cdot \sin(\phi)\sin(\theta) \\ z &= r \cdot \cos(\phi) \end{aligned} \tag{14}$$

where r is forced to be 1.

4) Rotate the Index array by each Quaternion Orientation

Using properties of quaternions, we are able to rotate each index array to their corresponding location in Cartesian coordinates using quaternion multiplication.

$$[0, v'] = q^{-1} \circ [0, v] \circ q = [0, R(q)x]$$

Note that this corresponds to the inverse rotation as defined previously.

5) Convert back to Spherical

In order to make a panoramic image, we need to project these back onto a unit sphere, where we can unwrap the transformed index collection into one stitched 2-D image. We can achieve this using the following equations on our point set:

$$\begin{aligned} r &= x^2 + y^2 + z^2 \\ \theta &= \arccos(\frac{z}{r}) \\ \phi &= \arctan(\frac{y}{x}) \end{aligned} \tag{15}$$

6) Unwrap the Image Sphere
After scaling, shifting, and (int) casting the coordinates such that they represent reasonable indices (positive throughout and unique within one image), we paint each pixel in the 2-D image in the location of the coordinates, with color defined by the corresponding image.

$$\begin{aligned} i &= \frac{width \cdot 5}{2\pi} \cdot \theta \\ j &= \frac{width \cdot 4}{\pi} \cdot (\phi + \pi) \end{aligned} \tag{16}$$

IV. Results and Discussion

A. Training

The training data given included 9 training datasets to work with, each consisting of raw IMU data and a truth dataset, where the truth data was captured using a Vicon motion capture system. Four of these datasets contained images taken whilst the camera was rotating, allowing construction of a panoramic image for that set. As a baseline for performance, we can preform a simple integration of the IMU data, which is calculated by taking the current estimate and propagate that through the quaternion portion of the noiseless motion model, denoted by equation (10):

$$q_{t+1|t} = q_{t|t}^{(i)} \circ \left([0, \frac{w_{t|t}}{2}]\right) \tag{17}$$

These quaternions plotted against the Vicon data as Euler angles:

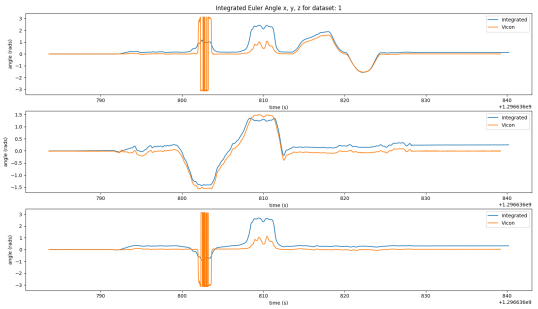

Fig. 1. Integration Estimation for Dataset 1 (Training)

As you can see, the integrated angles, shown in blue, match the shape of the true angles, shown in orange; however the amplitudes are incorrect, and sudden changes in angle are not reflected. This is not sufficient for our case, thus this validates our need for more advanced estimation techniques. Below is the result of the UKF on the same dataset.

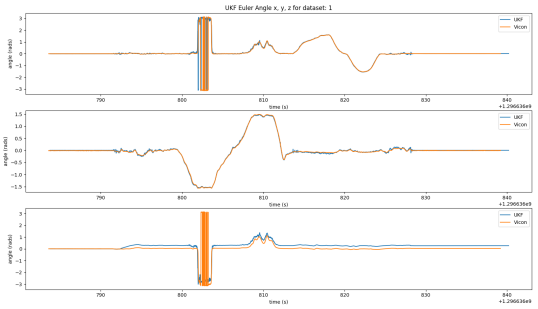

Fig. 2. UKF Estimation for Dataset 1 (Training)

Clearly it can be seen that the estimated orientation follows the truth dataset very closely, and completely matches in many areas. With this improved estimate, we can hope to reconstruct a panorama of the environment around it. Below is the constructed panorama from the above UKF estimates.

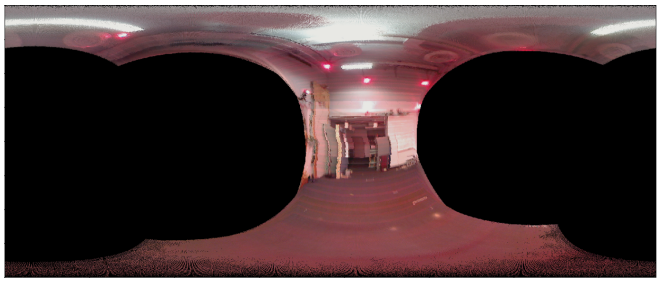

Fig. 3. Panorama for Dataset 1 (Training)

Sometimes, small oscillations may appear in the UKF estimate, resulting in a noisy estimate. This can be combated by adjusting β in equation (11), as well as the covariances with γ defined in the section "State Definition".

The estimates for dataset 8 (Training) contained small oscillations, and by adjusting $\beta : 1 \rightarrow .01$, I was able to achieve much better results:

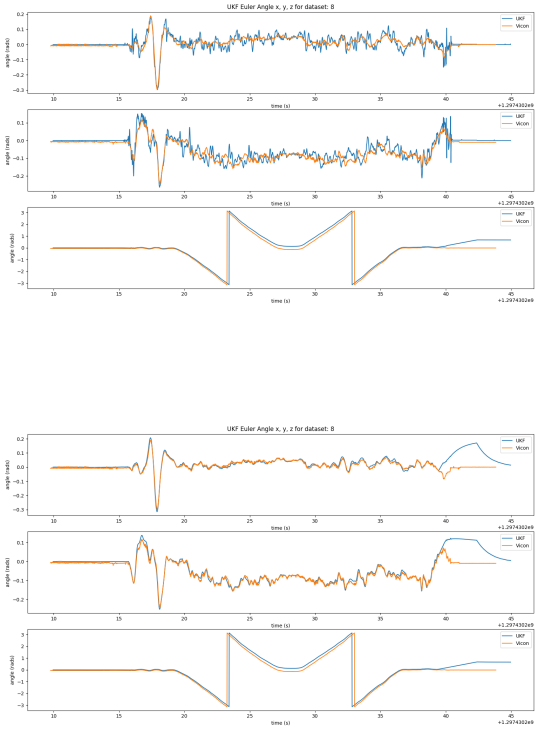

Fig. 4. UKF Estimation before noise correction (top), and after noise correction (bottom), Dataset 8 (training)

Overall, I would say the performance on the training set is successful, and other than having to tweak parameters to reduce noise, tracked the truth data pretty well. The performance on the datasets containing images are shown in Figures 5-8 below.

B. Test Results

Separate test data was obtained, and unlike the training data, there was no Vicon truth data. Therefore, plots from the test data only contain the estimate. Based on the quality of the panoramas achieved, orientation tracking for all of the test datasets was successful, however improvements could still be made. Ideally the image stitching would look more seamless, rather than the stacking appearance achieved. Figures 9-12 below depict these results.

REFERENCES

[1] E. Kraft, A Quaternion-Based Unscented Kalman Filter for Orientation Tracking, in *Proc. Sixth International Conference of Information Fusion*, vol. 1, 2003, pp. 4754.

Training Results:

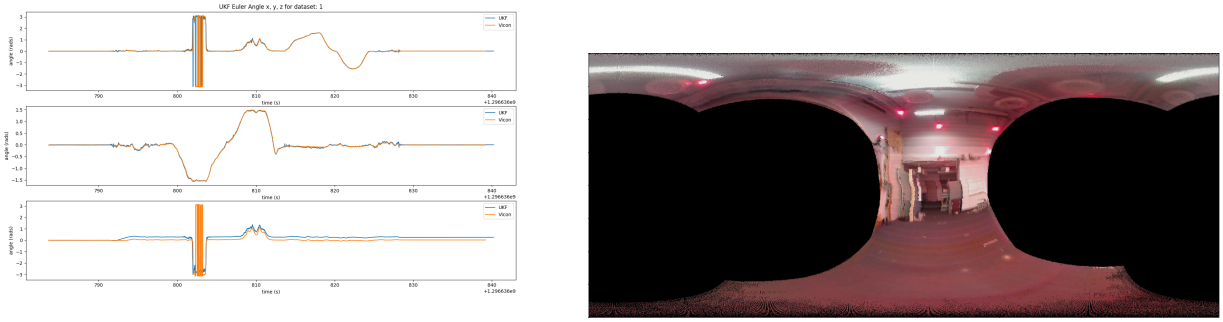

Fig. 5. UKF Estimation (Left) and Panoramic Image (Right) for Dataset 1 (Training)

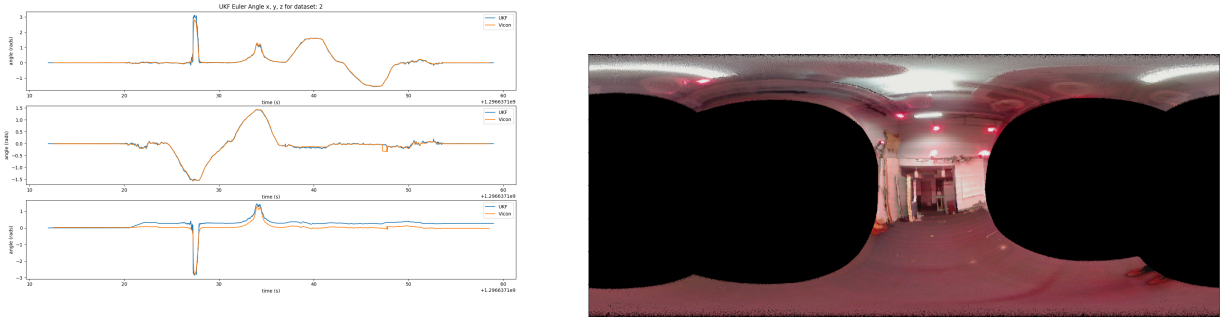

Fig. 6. UKF Estimation (Left) and Panoramic Image (Right) for Dataset 2 (Training)

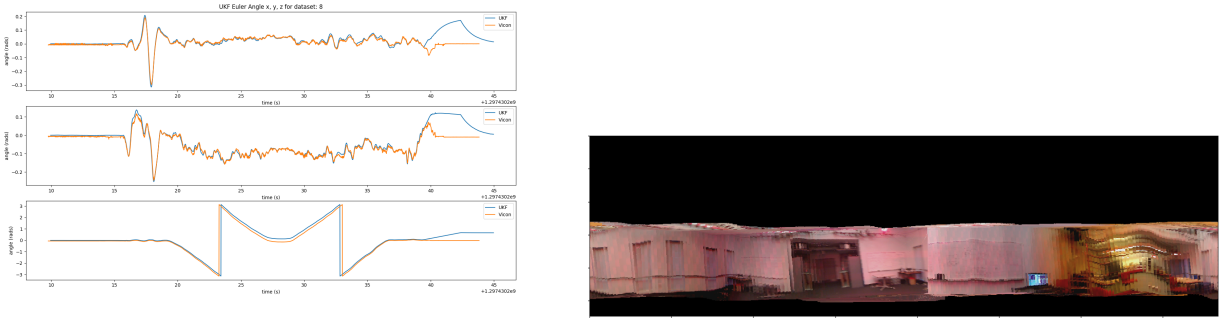

Fig. 7. UKF Estimation (Left) and Panoramic Image (Right) for Dataset 8 (Training)

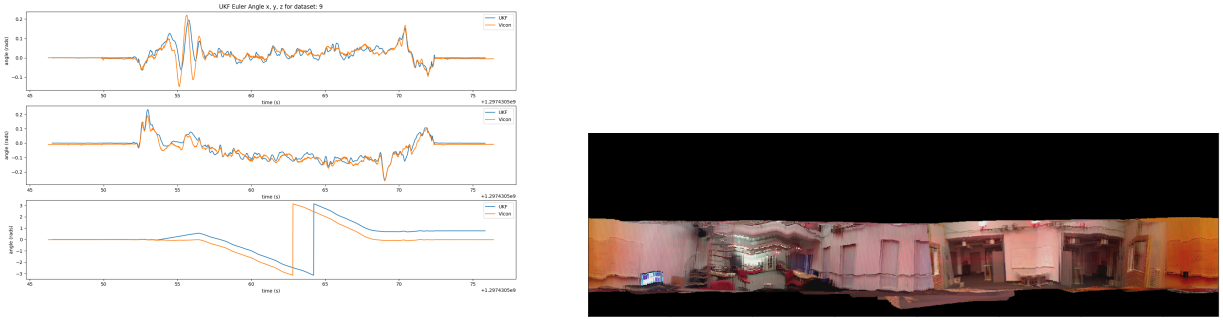

Fig. 8. UKF Estimation (Left) and Panoramic Image (Right) for Dataset 9 (Training)

Test Results:

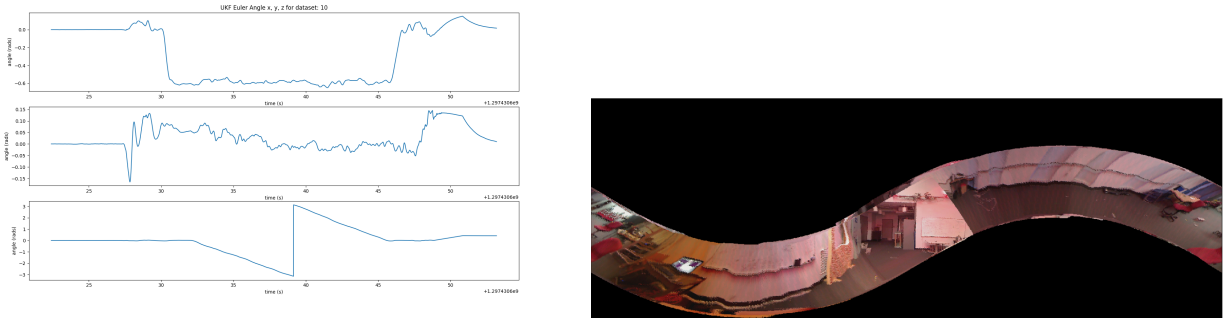

Fig. 9. UKF Estimation (Left) and Panoramic Image (Right) for Dataset 10 (Test)

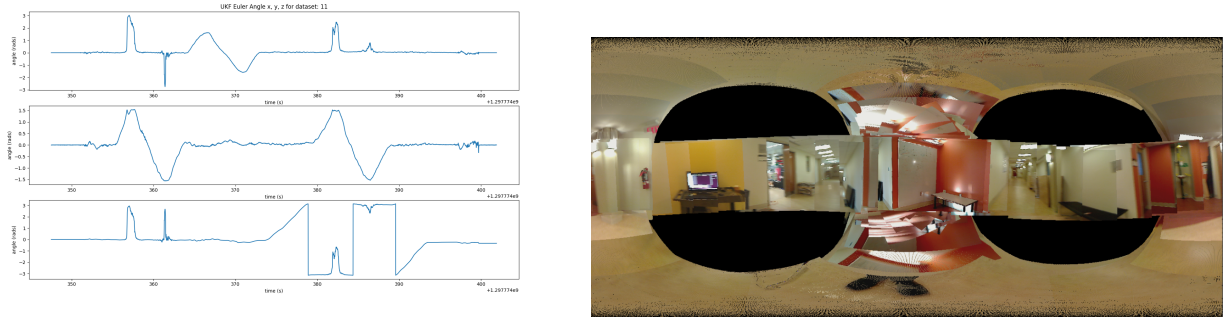

Fig. 10. UKF Estimation (Left) and Panoramic Image (Right) for Dataset 11 (Test)

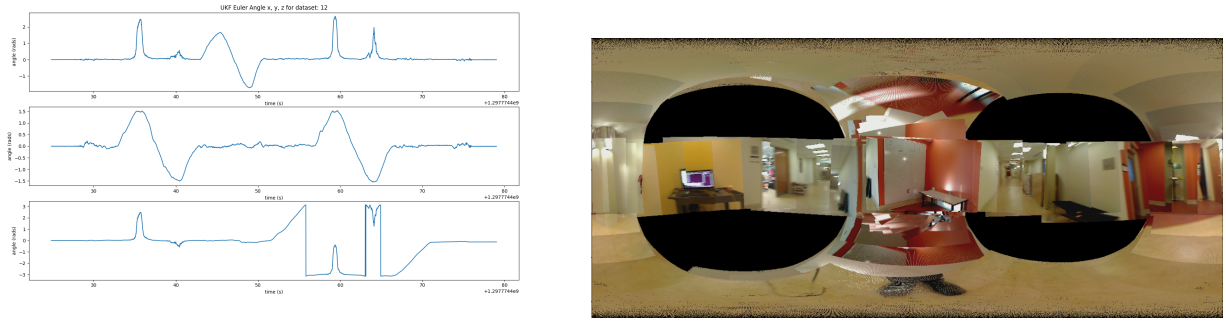

Fig. 11. UKF Estimation (Left) and Panoramic Image (Right) for Dataset 12 (Test)

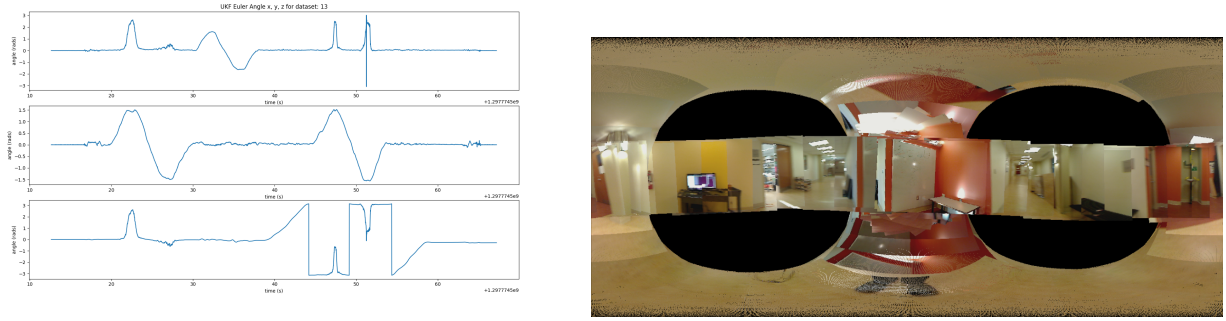

Fig. 12. UKF Estimation (Left) and Panoramic Image (Right) for Dataset 13 (Test)

Results from non-image training data:

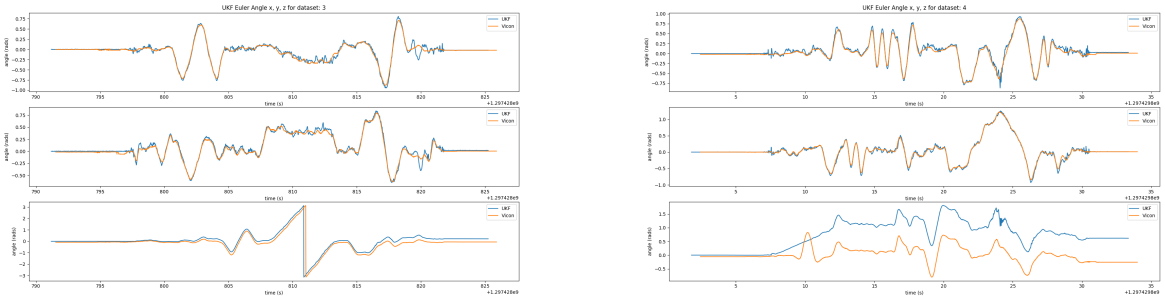

Fig. 13. UKF Estimation for Dataset 3 (Left) and Dataset 4 (Right))

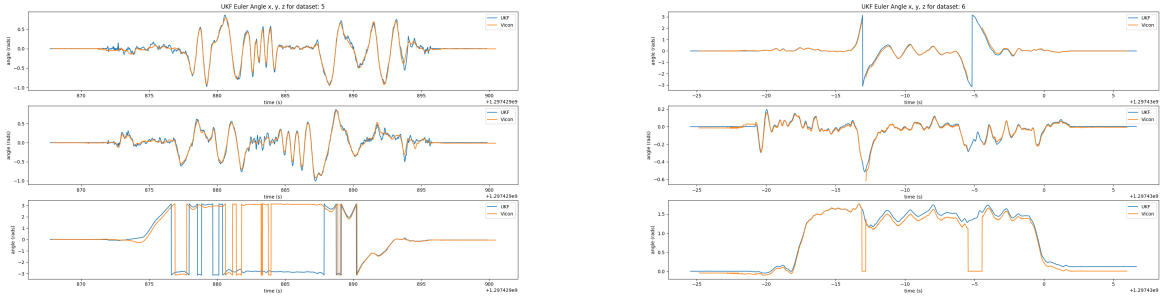

Fig. 14. UKF Estimation for Dataset 5 (Left) and Dataset 6 (Right)

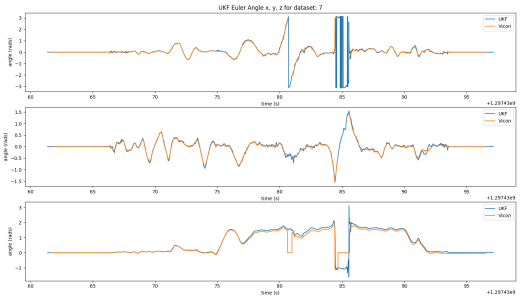

Fig. 15. UKF Estimation for Dataset 7