Implementation of a 'Non-Stinky' Unscented Kalman Filter (UKF) for Estimating 3D Attitude

Vishnu Sashank Dorbala
UID 116907569
University of Maryland, College Park

Surabhi Verma
UID 116949460
University of Maryland, College Park

Late Days Used: 2

I. Problem Statement

Given the linear acceleration and angular velocity readings from an Inertial Measurement Unit (IMU), we aim to estimate the attitude using an 'Non-Stinky' Unscented Kalman Filter (UKF). In order to achieve computational efficiency, we use quaternions to carry out all our operations as opposed to euler angles. Our results with UKF are compared with ground truth 3D orientation values obtained from a VICON motion capture system and also with the outcome of a Madgwick Filter.

II. Introduction

Unlike the simple Kalman Filter which works only when there is a linear transformation between actual and measured readings, the UKF works well with non-linear functions. While the Extended Kalman Filter (EKF) also works on non-linear functions, it is more computationally expensive as it requires the computation of a Jacobian. The UKF in comparison is 'Non-Stinky/Unscented', as it avoids the computation of the 'Stinky' Jacobian entirely.

Another key difference between EKF and UKF is that the latter uses only a single point (mean) to approximate the outcome, while UKF relies on the appropriate selection of a bunch of 'weighted sigma points' for approximation.

For these reasons, we look to implement a UKF for accurate and more efficient state estimation.

III. Data Pre-processing

We are given 6 Degree of Freedom (DoF) data from an Inertial Measurement Unit (IMU) in the order $\begin{bmatrix} a_x & a_y & a_z & \omega_z & \omega_x & \omega_y \end{bmatrix}^T$. This corresponds to the linear acceleration along the x, y, z axis and the angular velocity about the z, x, y axis respectively. These readings, however, are not in their corresponding physical units. Given below is the conversion of a raw acceleration value along a given axis 'n' (a_n) into ms^{-2}.

$$\widetilde{a_n} = (a_n * s_{a,n}) + b_{a,n}; n = x, y, z \tag{1}$$

Here, $b_{a,n}$ and $s_{a,n}$ represent the bias and scaling factors for acceleration along the n^{th} axis. These values are provided to us. For the raw angular velocity value (w_n) about the n^{th} axis the conversion into physical units can be written in $rads^{-1}$ as

$$\widetilde{\omega_n} = \frac{3300}{1023} \times \frac{\pi}{180} \times 0.3 \times (\omega_n - b_{g,n}); n = x, y, z \tag{2}$$

where, $b_{g,n}$ is the gyroscope bias for the angular velocities about the n^{th} axis. The bias is calculated by taking the average of the first 200 values individually of each angular velocity. We also convert the VICON data, given in terms of rotation matrices in the Z-Y-X convention, into Euler angles. This is described as follows.

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} atan2(r_{32}, r_{33}) \\ atan2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ atan2(r_{21} + r_{11}) \end{bmatrix} \tag{3}$$

IV. Method

There are two major stages involved in our implementation of the UKF using quaternions. First is the prediction stage, where we select weighted sigma points from the input distribution, perform a non-linear 'unscented transformation' operation on them, and then predict the mean and variance of the outcome.

The second is the update stage, where we predict the measurement from a Gaussian approximation of the mean and variance obtained, and update the state.

A. Defining the State Vector and Process Model

We take the state vector for our system as a combination of the 3D orientation in quaternion format (q), and the angular velocity ($\vec{\omega}$). That gives us,

$$x = \begin{pmatrix} q \\ \vec{\omega} \end{pmatrix} \tag{3}$$

This has 7 components, 4 from q as q_w, q_x, q_y, q_z, and 3 from $\vec{\omega}$ as $(\omega_x, \omega_y, \omega_z)$. 'q' here is a unit quaternion. This constraint deprives the quaternion from one degree of freedom, which makes for 6 degrees of freedom in total, 3 for q, and 3 for $\vec{\omega}$.

We define a process model A for predicting the evolution of the state vector x as follows:

$$x_{k+1} = A(x_k, w_k) \tag{4}$$

Here, w_k is the process noise described by a six dimensional vector,

$$w_k = \begin{pmatrix} \vec{w_q} \\ \vec{w_\omega} \end{pmatrix} \tag{5}$$

As equation 4 has a wide range of possible combinations, for simplicity, we choose the angular velocity ω to be constant over a time Δt i.e., $\omega_{k+1} = \omega$

The process model can be defined by:

$$q\Delta = [cos(\frac{\alpha_\Delta}{2}), \vec{e_\Delta}\frac{\alpha_\Delta}{2}] \tag{6}$$

where, the angle $\alpha_\Delta = |\vec{\omega_k}|.\Delta t$ and axis $\vec{e_\Delta} = \frac{\vec{\omega_k}}{|\vec{\omega_k}|}$.

B. Selection of Sigma Points

The number of sigma points chosen ($2n$) depends on the degrees of freedom (n) of our system. In our case, as we compute an orientation in 3 dimensions, we choose 6 sigma points for the UKF. The sigma points are distributed evenly about the mean (μ) of the input distribution. We use the following equation to define them.

$$\chi_i = \hat{x}_{k-1} + \mathcal{W}_i$$

Here, $\mathcal{W}_i$ describes a set of $2n$ elements distributed around a mean value of zero. x_{k-1} is the state vector of the previous estimate. We compute a square root matrix S, and subsequently get $\mathcal{W}$ as:

$$S = \sqrt{P_{k-1} + Q}, \textit{ and}$$
$$\mathcal{W}_{i,i+n} = columns(\pm\sqrt{2n}S)$$

Here, P_{k-1} is an 6x6 matrix describing the covariance of the state vector x_{k-1}. It is initialized P0 from a unit quaternion with zero angular velocities. Q is the process model covariance.

The final sigma points then become:

$$\chi_i = \begin{bmatrix} q_{k-1}q_{\mathcal{W}} \\ \vec{\omega_{k-1}} + \vec{\omega_{\mathcal{W}}} \end{bmatrix} \tag{7}$$

C. Sigma Point Transformation

After defining the sigma point set, χ_i, we transform each of them using the process model. This is described as:

$$\mathcal{Y}_i = A(\chi_i, 0) \tag{8}$$

Note that no noise is added to this estimate, because we already define the sigma points taking the process noise into account.

Here, the set of points $\mathcal{Y}_i$ can be considered as an *a priori* for the Gaussian. We need to derive an *a posteriori* from this. $\bar{x}_k$ and $\bar{P}_k$ are the mean and covariance of set $\mathcal{Y}_i$.

D. Computing Mean of Sigma Points

As we compute everything in quaternions, the general Barycentric mean formula i.e,

$$\bar{\mathcal{Y}} = \frac{1}{2n}\sum_{i=1}^{2n}\mathcal{Y}_i$$

does not work. This is because quaternions mathematics works differently from regular vector math.

In order to compute the mean of $\bar{\mathcal{Y}}$, we use a gradient descent algorithm that tries to optimize for the mean $\bar{q}_t$ by iteratively minimizing its average distance from elements of $\mathcal{Y}_i$. This distance, defined as an error e_i, describes the orientation (angle) between the two elements in quaternion space. e_i is defined as

$$e_i = q_i.\bar{q}_t^{-1} \tag{9}$$

where q_i the set element in a sigma point set ($\mathcal{Y}$ in our case), and $\bar{q}_t$ is the mean obtained in the previous iteration. $\bar{q}_t$ is initialized with a unit quaternion. We compute e_i for all the elements of the sigma set.

Now, vector representation, $\vec{e}_i$, of e_i (assuming $e_i = a + bi + cj + dk$) is given by

$$\vec{e} = \frac{\theta}{sin(\theta/2)}[b, c, d] \tag{10}$$

where θ is the rotation angle, defined by $\theta = 2\cos^{-1}(a)$. $\vec{e}$ is called the adjustment vector.

Now, the mean can be computed as,

$$\vec{e} = \frac{1}{2n}\sum_{i=1}^{2n}\vec{e}_i$$

.

After converting this back into a quaternion, we can use it to compute the estimate for the next iteration as,

$$\bar{q}_{t+1} = e\bar{q}_t \tag{11}$$

We run this loop iteratively 1000 times and take the final q_{t+1} value as the quaternion mean $\bar{x}_k$.

E. Computing Covariance for Sigma Points

The covariance matrix for the 6 sigma points we obtain can be computed as

$$\bar{P}_k = \frac{1}{2n}\sum_{i=1}^{2n}\mathcal{W}_i'\mathcal{W}_i'^T \tag{12}$$

W_i' is a six dimensional representation of the transformed sigma point set $\mathcal{Y}_i$. We get this by subtracting each point in $\mathcal{Y}_i$ with the mean vector, and converting the quaternion part to a rotation vector.

$$\mathcal{W}_i' = \begin{bmatrix} vec(q_i\bar{q}^{-1}) \\ \vec{\omega}_i - \bar{\vec{\omega}} \end{bmatrix}, \tag{13}$$

where, vec(.) implies conversion of the quaternion result into vector.

F. Measurement Model and Covariances

A measurement model H is defined that draws the relationship between the measurement value z to the value of the state vector x, but while introducing measurement noise v. In our case, the measurement model H needs to be defined for IMU readings of gyroscope and accelerometer data.

After obtaining the transformed sigma point space $\mathcal{Y}$, we project it into a 3D measurement space. This is done as

$$\mathcal{Z}_i = \begin{bmatrix} q_i * g * \bar{q}_i^{-1} \\ \vec{\omega}_k \end{bmatrix} \tag{14}$$

Here, g is the gravity vector defined as a quaternion with only a unit value along the z axis.

Now, the covariance of the measurement model ($\mathcal{Z}$) is given by

$$P_{zz} = \frac{1}{2n}\sum_{i=1}^{2n} \mathcal{W}\mathcal{Z}_i' \mathcal{W}\mathcal{Z}_i'^T \tag{15}$$

where WZ_i is the transformed six dimensional set for points in $\mathcal{Z}$.

The *innovation* term ν_k is the difference between the actual measurement z_k and its predicted value $\bar{z_k}$, i.e,

$$\nu_k = z_k - \bar{z_k} \tag{16}$$

Here, the innovation covariance is given by,

$$P_{\nu\nu} = P_{zz} + R \tag{17}$$

where R is measurement noise covariance.

The cross-covariance is then given by

$$P_{xz} = \frac{1}{2n}\sum_{i=1}^{2n} \mathcal{W}_i' \mathcal{W}\mathcal{Z}_i'^T \tag{18}$$

G. Updating Kalman Gain and State

The Kalman Gain for UKF is given by,

$$K_k = P_{xz}\dot{P}_{\nu\nu}^{-1} \tag{19}$$

The state is then updated as,

$$\hat{x_k} = \hat{x_k}^- + K_k\nu_k \tag{20}$$

and the state covariance becomes

$$P_k = P_k^- - K_k P_{\nu\nu} K_k^T \tag{21}$$

V. Plots and Analysis

We have 10 instances of raw data from the accelerometer and gyroscope present in the dataset. VICON data is available only for the first 6 instances (the training set). The remaining 4 sets do not have VICON data, and make up the test set. We hence plot the Euler angles from the VICON along with Madgwick and UFK only for the training set. This is done for the purposes of comparison. The following observations are made.

- The Performance of UKF is largely dependent on the initial estimates of P_k, Q, R. There could be multiple scenarios: convergence to a wrong estimate, large time for convergence, satisfactory convergence only for some states.
- The number of tuning parameters (P_k, Q, R) for UKF is larger in comparison to the Madgwick Filter (β). This makes it harder to get the fine-tuning right with all the 6 datasets and with all the estimated states from a given dataset.
- We set the initial estimates of P_k, Q, and R as identity matrices scaled by different factors. The scale, especially for Q and P, were found to be very sensitive to changes in the order of 10^{-6} and 10^{-2} respectively.
- The UKF plot seems to closely follow the Madgwick plot on the test set, especially with the Phi (Yaw) values.

VI. Rotplot Videos

In order to view the 3D orientation of the estimated Euler angles, we check the rotation on a 3D object. These 'rotplot' outcomes have been uploaded here.[1].

VII. Lessons Learnt

The outcome of the UKF is highly dependant on the tuning parameters (initial covariance matrix assumptions). This was especially true with the initialization of the Process Model Covariance matrix Q. Even very small changes in the order of 10^{-6} caused drastically different outcomes.

Another observation we made was that there were less hassles in tuning the Madgwick filter compared to the UKF. This is also because there were more parameters to tune in UKF compared to Madgwick. In situations where a quick implementation of a state estimator is necessary, one might consider using the Madgwick Filter instead of the UKF for a faster implementation.

VIII. Conclusion

In conclusion, we observe that UKF performs better than the Madgwick in some instances. However, as Madgwick still gives comparable results and is simpler to tune, it might be a more suitable filter to implement, depending on the constraints of the problem.

IX. Acknowledgments

We would like to thank the Teaching Assistants of the course, Abhinav Modi and Prateek Arora, as well as the instructors, Nitin. J. Sanket and Chahat Deep Singh for their advice and constant guidance throughout the course of this assignment.

X. References

- Madgwick Filter Tutorial - Link
- Madgwick Filter Paper - Link
- Pyquaternion Library - Link
- Working with Quaternions - Link
- Visualizing Quaternions - Link

[1]Link to Rotplots

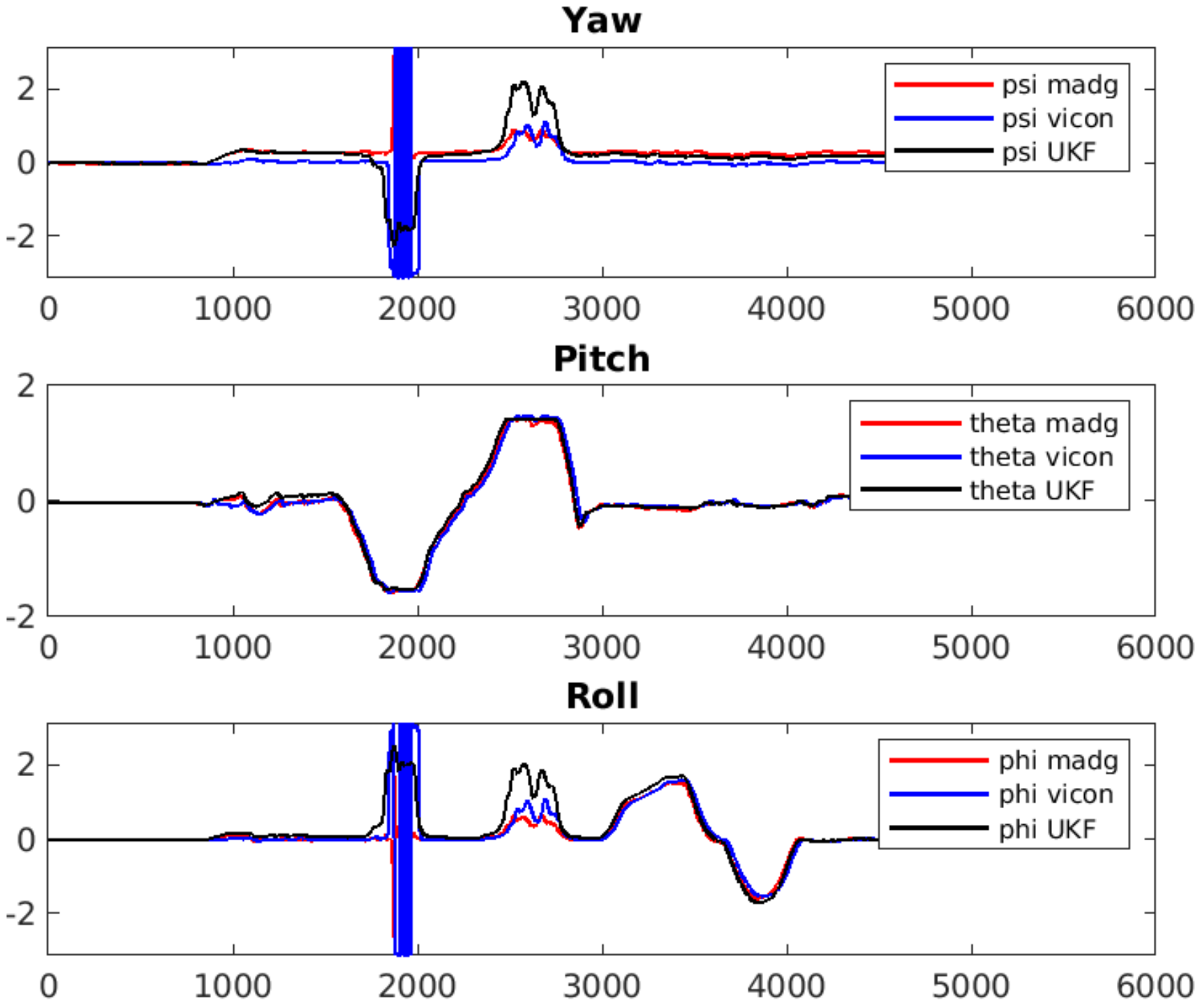
Yaw
psi madg
psi vicon
psi UKF
Pitch
theta madg
theta vicon
theta UKF
Roll
phi madg
phi vicon
phi UKF
2
0
-2
0
1000
2000
3000
4000
5000
6000

Fig. 1: Raw 1

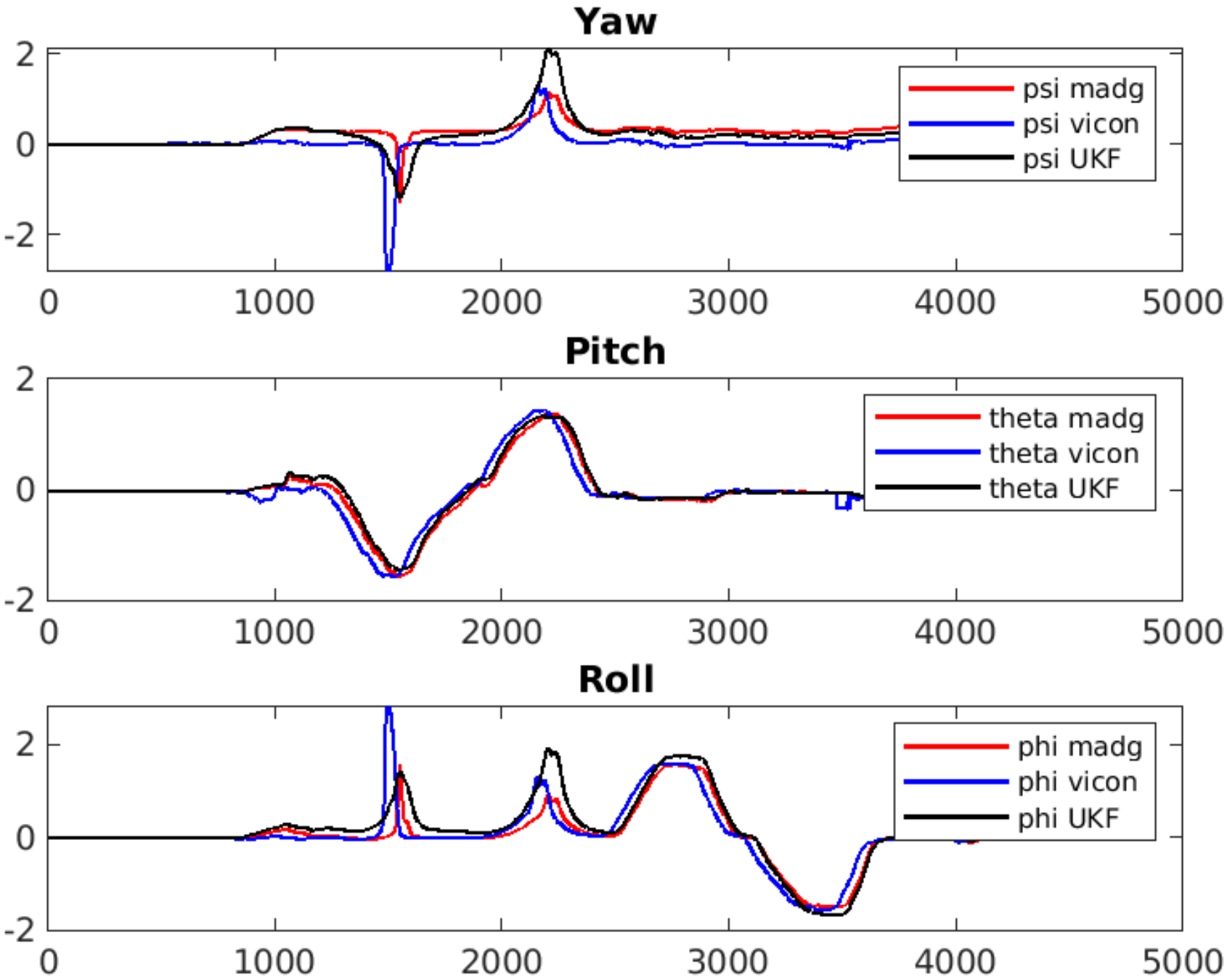
Yaw
2
0
-2
0
1000
2000
3000
4000
5000
psi madg
psi vicon
psi UKF
Pitch
2
0
-2
0
1000
2000
3000
4000
5000
theta madg
theta vicon
theta UKF
Roll
2
0
-2
0
1000
2000
3000
4000
5000
phi madg
phi vicon
phi UKF

Fig. 2: Raw 2

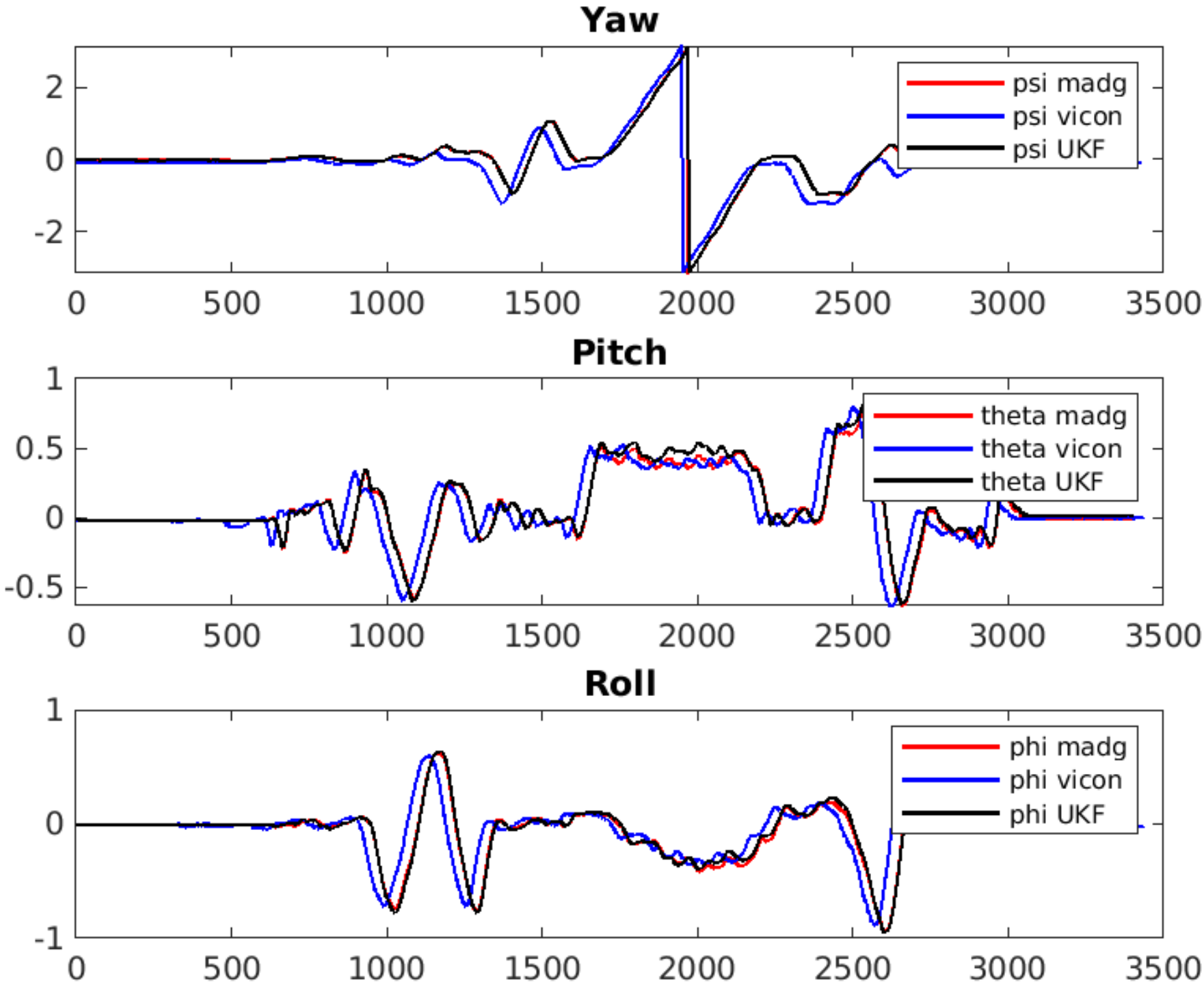
Yaw
psi madg
psi vicon
psi UKF
2
0
-2
0
500
1000
1500
2000
2500
3000
3500
Pitch
theta madg
theta vicon
theta UKF
1
0.5
0
-0.5
0
500
1000
1500
2000
2500
3000
3500
Roll
phi madg
phi vicon
phi UKF
1
0
-1
0
500
1000
1500
2000
2500
3000
3500

Fig. 3: Raw 3

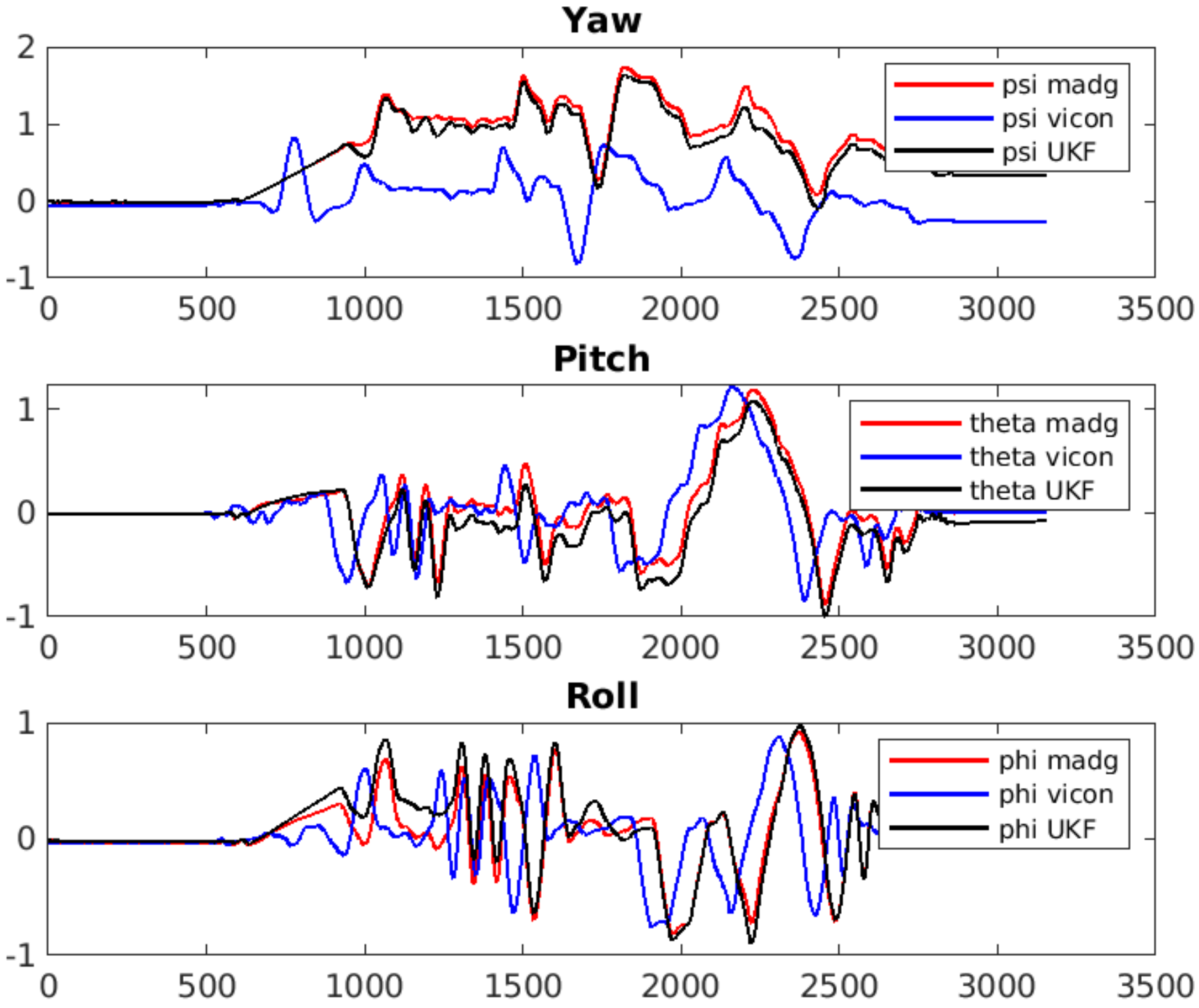
Yaw
psi madg
psi vicon
psi UKF
Pitch
theta madg
theta vicon
theta UKF
Roll
phi madg
phi vicon
phi UKF
2
1
0
-1
0
500
1000
1500
2000
2500
3000
3500

Fig. 4: Raw 4

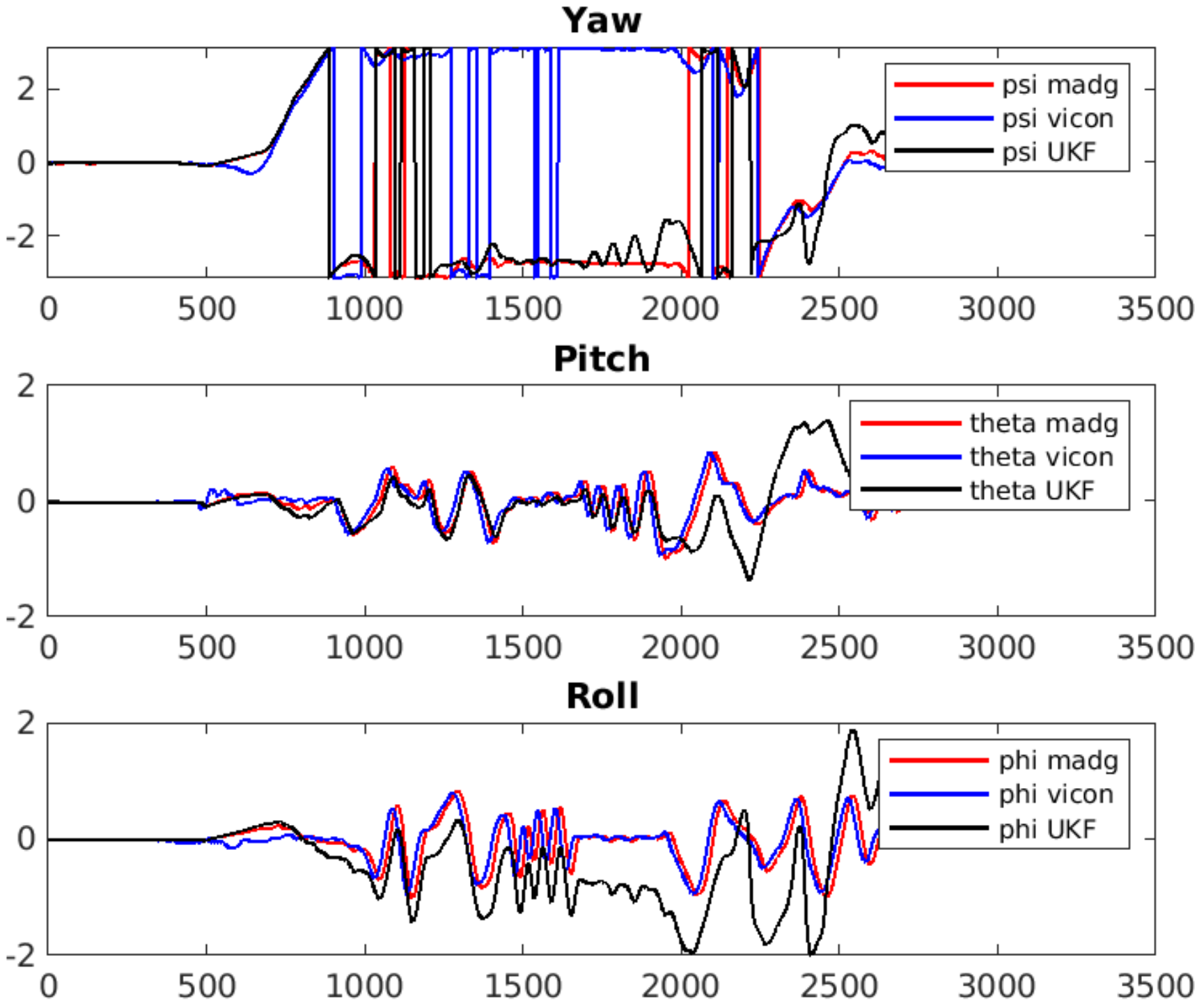
Yaw
psi madg
psi vicon
psi UKF
Pitch
theta madg
theta vicon
theta UKF
Roll
phi madg
phi vicon
phi UKF
2
0
-2
0
500
1000
1500
2000
2500
3000
3500

Fig. 5: Raw 5

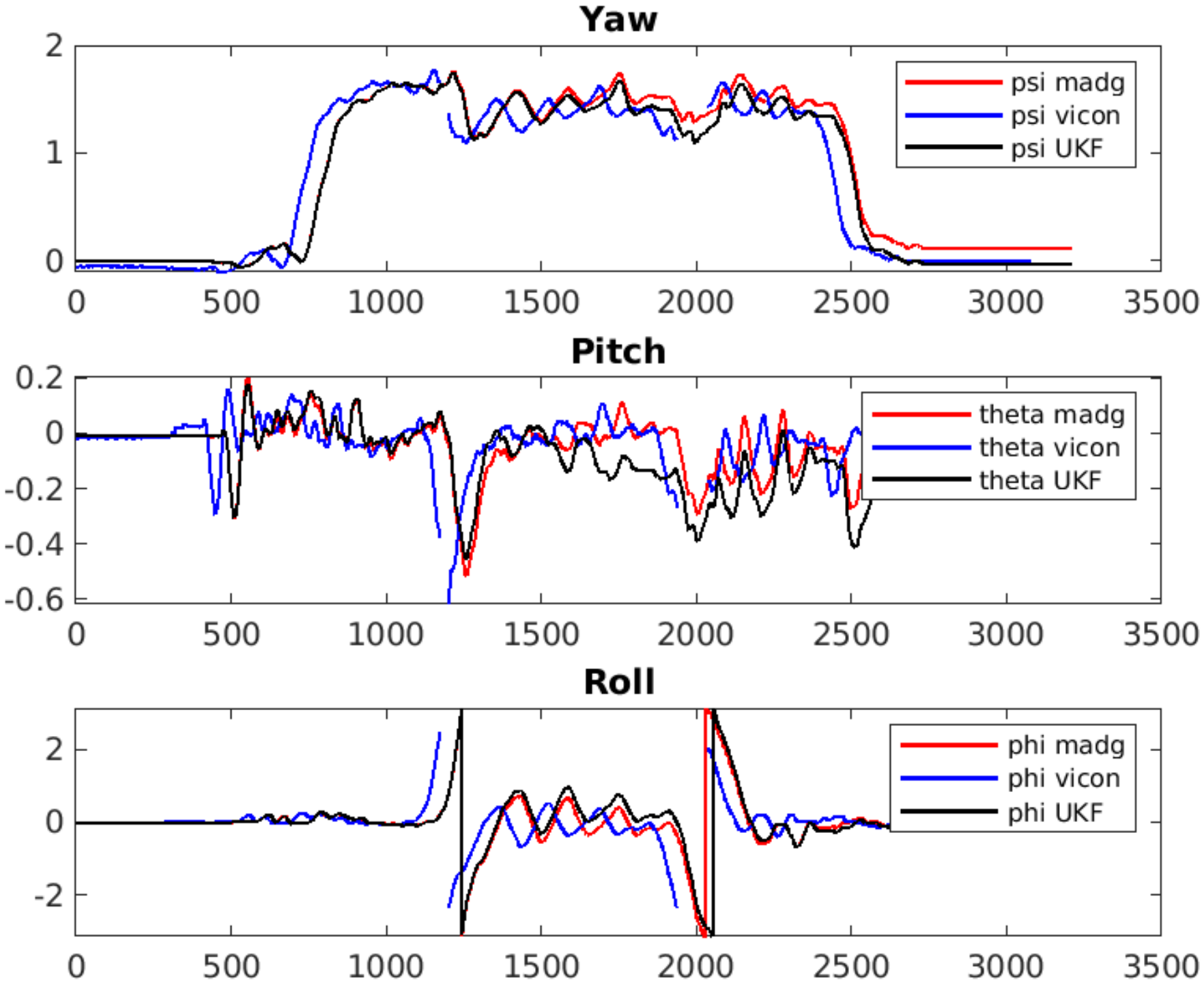
Yaw
psi madg
psi vicon
psi UKF
Pitch
theta madg
theta vicon
theta UKF
Roll
phi madg
phi vicon
phi UKF

Fig. 6: Raw 6

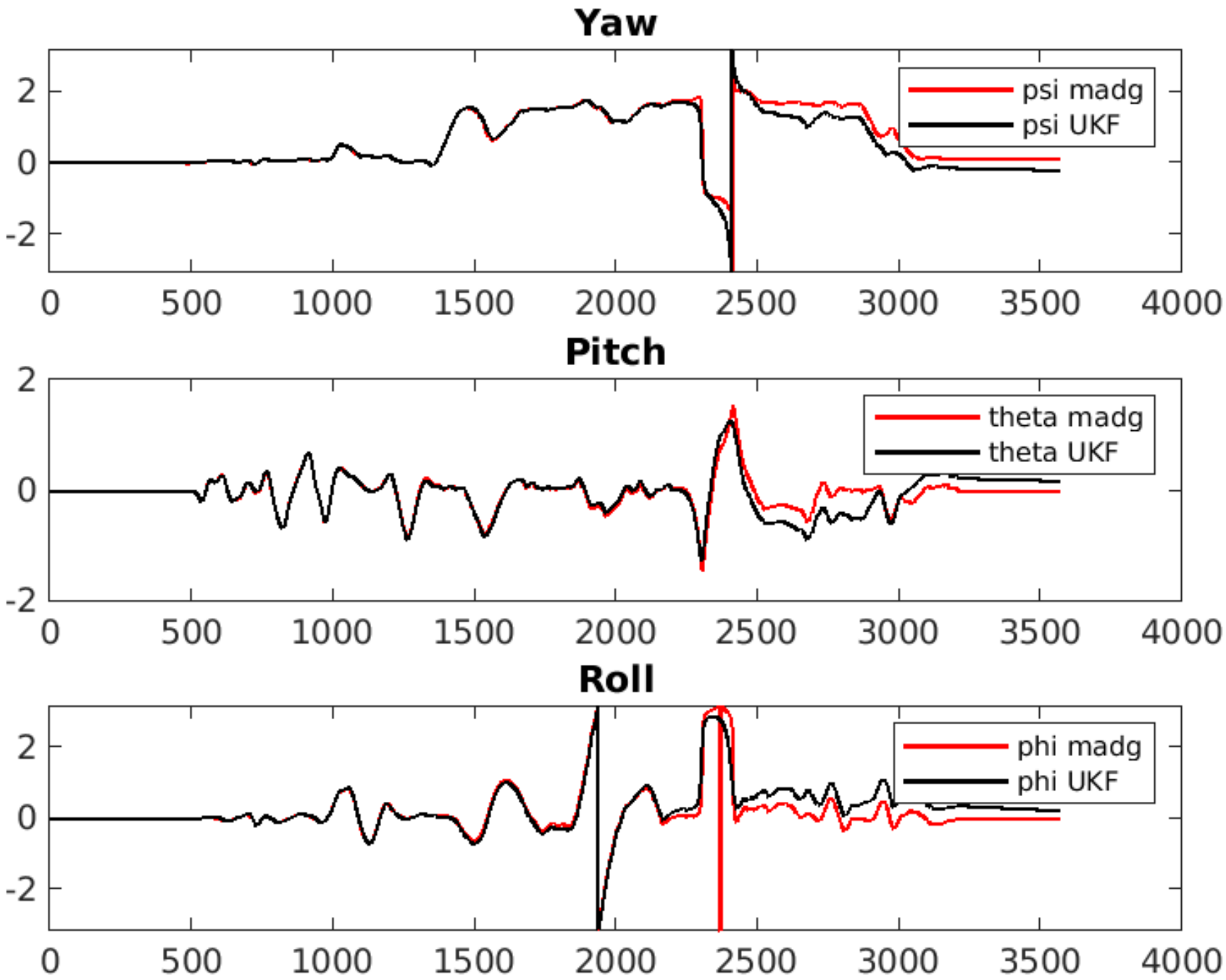
Yaw
psi madg
psi UKF
Pitch
theta madg
theta UKF
Roll
phi madg
phi UKF
2
0
-2
0
500
1000
1500
2000
2500
3000
3500
4000

Fig. 7: Raw 7

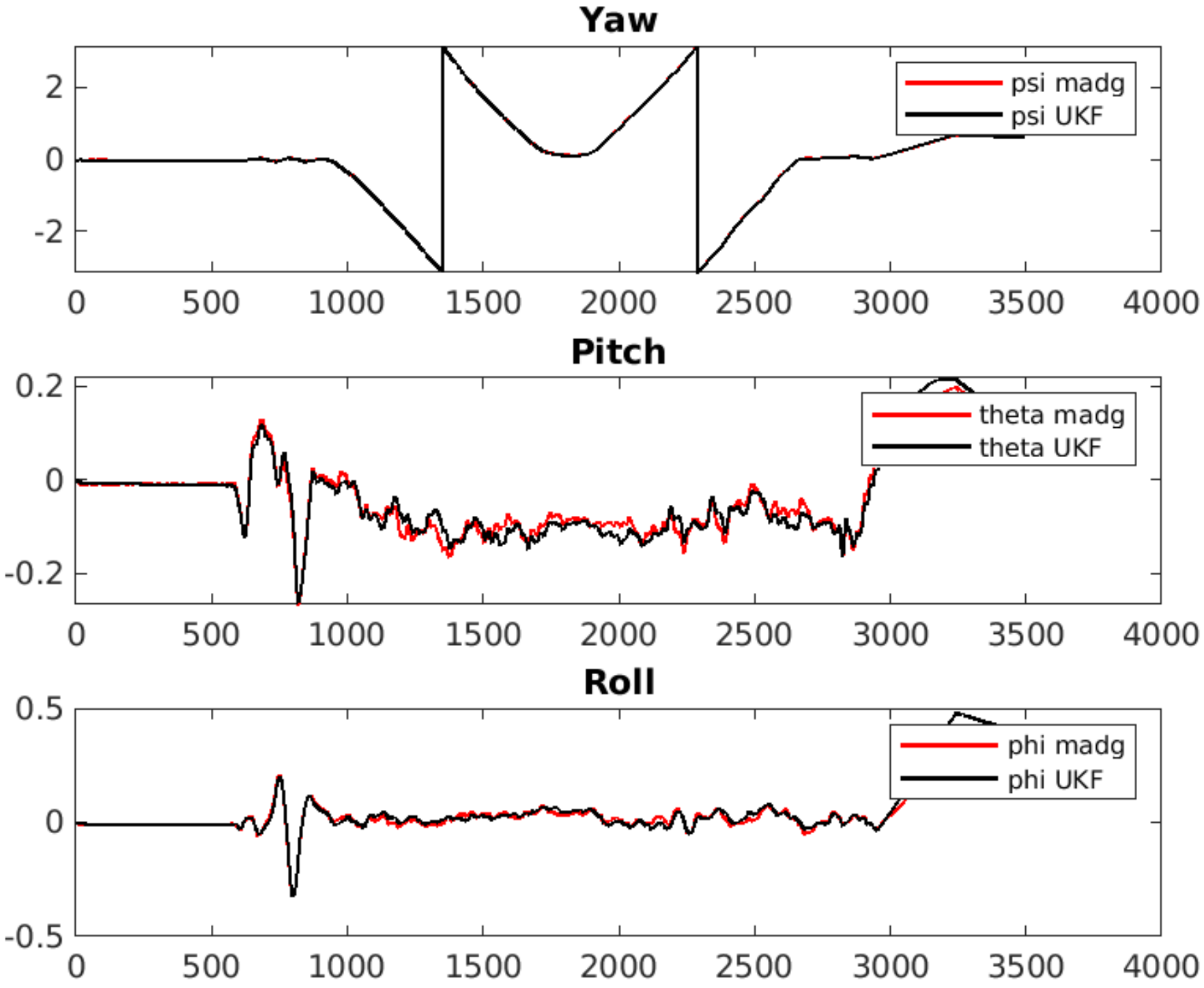
Yaw
psi madg
psi UKF
2
0
-2
0
500
1000
1500
2000
2500
3000
3500
4000
Pitch
theta madg
theta UKF
0.2
0
-0.2
0
500
1000
1500
2000
2500
3000
3500
4000
Roll
phi madg
phi UKF
0.5
0
-0.5
0
500
1000
1500
2000
2500
3000
3500
4000

Fig. 8: Raw 8

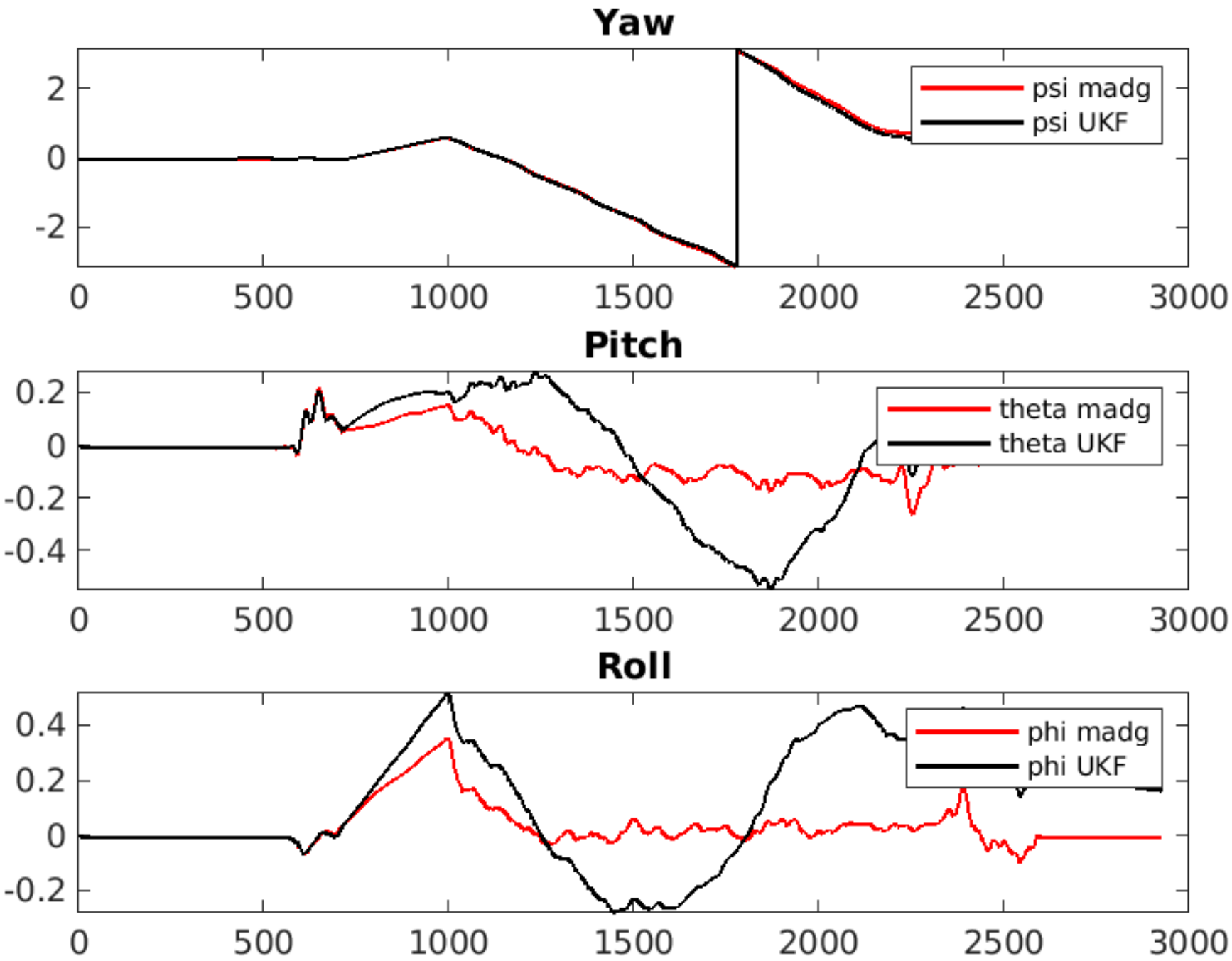
Yaw
psi madg
psi UKF
2
0
-2
0
500
1000
1500
2000
2500
3000
Pitch
theta madg
theta UKF
0.2
0
-0.2
-0.4
0
500
1000
1500
2000
2500
3000
Roll
phi madg
phi UKF
0.4
0.2
0
-0.2
0
500
1000
1500
2000
2500
3000

Fig. 9: Raw 9

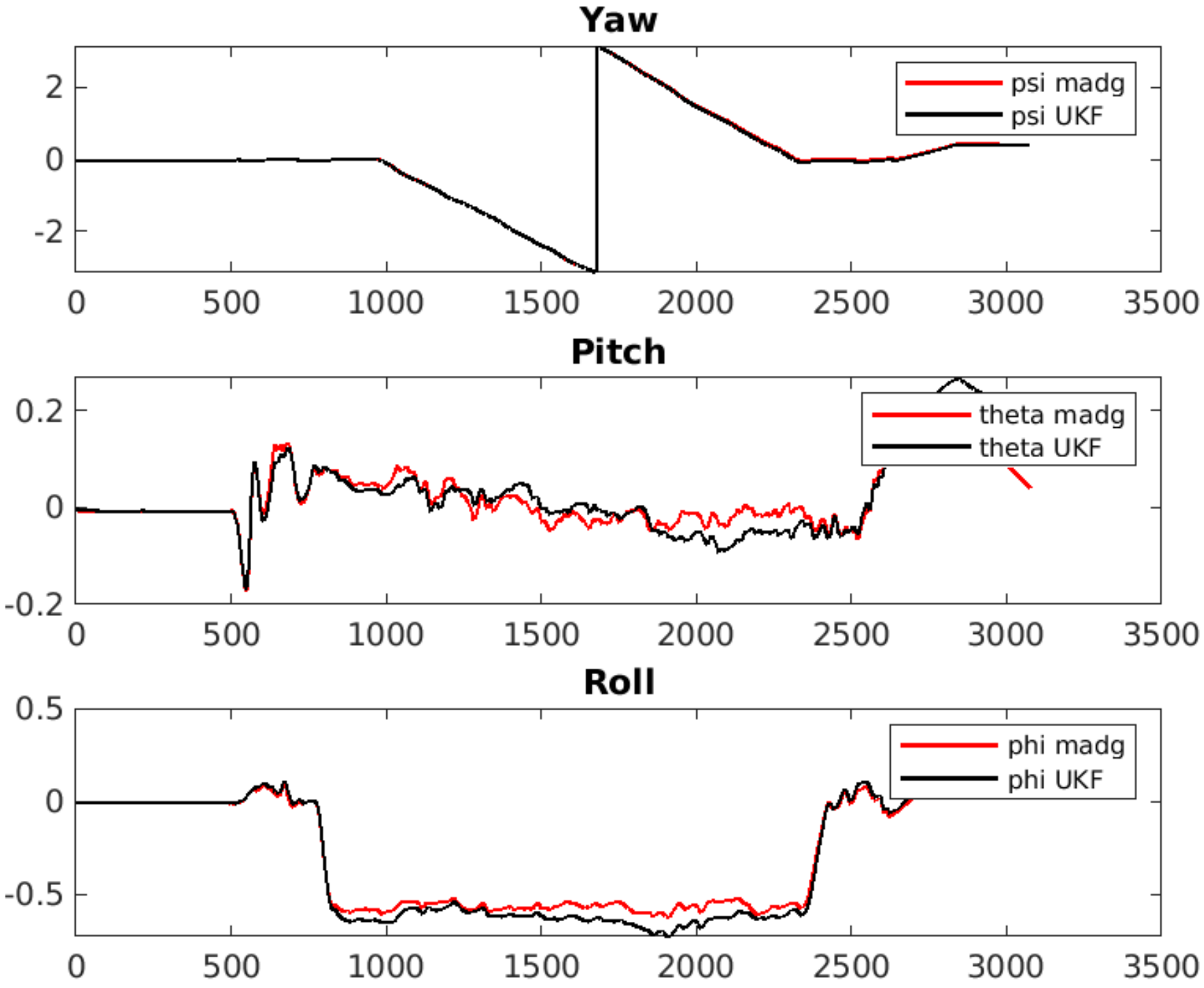
Yaw
psi madg
psi UKF
2
0
-2
0
500
1000
1500
2000
2500
3000
3500
Pitch
theta madg
theta UKF
0.2
0
-0.2
0
500
1000
1500
2000
2500
3000
3500
Roll
phi madg
phi UKF
0.5
0
-0.5
0
500
1000
1500
2000
2500
3000
3500

Fig. 10: Raw 10