ENAE788M Project 1b
Team Bouncing Rainbow Zebras

Erik Holum
Graduate Student
University of Maryland
Email: eholum@gmail.com

Edward Carney
Graduate Student
University of Maryland
Email: carneyedwardj@gmail.com

Derek Thompson
Graduate Student
University of Maryland
Email: derekbt@yahoo.com

***Abstract*—This project present results from determining orientation of a platform from gyroscope and accelerometer measurements. Results are presented using an implementation of the Madgwick filter, the Kalman filter, and the Unscented Kalman Filter (UKF). Results are compared to data captured by a Vicon imaging system, and links to videos animating the orientation through time are provided. While all filters provide adequate tracking of the attitude the UKF filter can be seen to provide the most accurate output as it most closely follows the Vicon data.**

I. Introduction

The purpose of this project is to estimate the orientation of a platform based on data collected from a mounted 6-Degree of Freedom Inertial Measurement Unit (6 DoF IMU). This was done using four different types of filters, an only linear acceleration filter, an only gyroscopic filter, a complementary filter, and a Madgwick filter. In all of cases, accelerometer bias and scale data were provided; gyroscope bias was determined for each data set based on the first three hundred gyroscope measurements. These parameters allowed conversion of raw data values to SI units. The filters were executed on 5 training sets and 4 previously unseen sets. For the training data sets, corresponding 'truth' orientation data was collected from a Vicon motion capture system and was available for comparison against estimated values.

In this report, we will discuss the implementation details of each filter. Then present results comparing the computed attitude estimates using each filter type. Relevant lessons and observations are discussed.

II. Scaling and Removing Bias

This section describes the methods for estimating and removing bias and scaling the raw data values collected from the IMU to SI units for the accelerometer and gyroscope.

A. Accelerometer

To interpret the raw IMU accelerometer readings, the data was multiplied by the given scale factor, s_a and added to the given accelerometer bias b_a for each axis of the accelerometer. This resulted in the measured acceleration as a function of gravitational acceleration.

$$\hat{\tilde{a}}_i = ((a_i \times s_{a,i}) + b_{a,i}) \tag{1}$$

This was done for each axis $i \in \{x, y, z\}$. The values for both b_a and s_a were provided and were constant for each set of data considered here.

B. Gyroscope

The data from the gyroscopes were also processed with a known scale factor s_g, but with a computed bias factor b_g. The scale factor was given from a IMU data sheet as,

$$s_g = \frac{3300}{1023} \times \frac{\pi}{180} \times 0.3 \tag{2}$$

The gyroscope bias was calculated from the average of the first 200 gyroscope measurements. The gyroscope is assumed to be steady for the first 200 measurements of each data set to determine initial angular rates. For each axis, the bias was determined as

$$b_{g,i} = \frac{1}{k} \sum_{j=1}^{k} \omega_i \tag{3}$$

where $k = 200$. These terms were used to calculate the desired rad/s angular rate, $\tilde{\omega}$, from the raw gyroscope data using the equation below.

$$\tilde{\omega_i} = s_{g,i} \times (\omega_i \times b_{g,i}) \tag{4}$$

Similar to the accelerometer readings, this was done for each axis $i \in \{x, y, z\}$. Unlike the accelerometer data, the bias value b_g was computed for each data set.

III. Interpreting the data

The data was initially read into the the script through the MATLAB files and converted into arrays using the supporting NumPy library in Python. The raw Vicon, IMU, and IMU parameter data was stored for each individual test. In order to compare the computed attitude during testing, the Vicon data was converted from the provided rotation matrix format to Roll, Pitch, and Yaw Euler angles. The rotation matrix provided by the Vicon data is represented as follows,

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix},$$

Euler angles were computed via the following conversions:

$$Roll, \phi = tan^{-1}(r_{3,2}/r_{3,3}) \tag{5}$$

$$Pitch, \theta = tan^{-1}(-r_{3,1}/\sqrt{r_{3,2}^2 + r_{3,3}^2}) \tag{6}$$

$$Yaw, \psi = tan^{-1}(r_{2,1}/r_{1,1}) \tag{7}$$

These values formed the truth data against which the estimated values could be compared.

IV. Kalman Filter

For our own understanding, and because it was simpler to implement, we tried our hand at writing a basic Kalman filter as described in both the slides and references provided. Our methods are summarized in this section. However, we found tuning the gains to be particularly cumbersome. Also, regardless of what we tried there were some scenarios where we were unable to get our implementation to out perform the simple complementary filter from problem set 1a. Rather than spend more time debugging this, we felt that even the basic implementation was enough to get a better handle on understanding gains and concepts; therefore, we include the somewhat under-performing filter regardless.

The measurement model estimates the system state $\bar{\mu}$, through a linear motion model. This uses the current state estimate, linear model A, system input matrix B, and input u_t. Additionally an estimate for the process covariance $\bar{\Sigma}$, is computed using the previous covariance matrix and process noise matrix Q.

$$\bar{\mu}_t = Ax_{t-1} + B\omega_t \tag{8}$$

$$\bar{\Sigma}_t = A\Sigma_{t-1}A^T + Q \tag{9}$$

For our model, our states are defined as $x = [\phi, \theta, \psi]^T$ and the input is set as the gyro measurements. With this the linear process A is set to the identity matrix and the system input matrix B is set to $I_3 * dt$. The process noise covariance matrix is initialized to $0.5 * I_3$

After the process model, the measurement model is computed to get to get an updated covariance and model estimate. The system measurement z_t is computed from the accelerometer data using the equations below.

$$\begin{aligned} \phi &= tan^{-1}(a_y/\sqrt{a_x^2 + a_z^2}) \\ \theta &= -tan^{-1}(a_x/\sqrt{a_y^2 + a_z^2}) \\ \psi &= tan^{-1}(\sqrt{a_x^2 + a_y^2}/a_z) \end{aligned} \tag{10}$$

Due to the output from the computed measurement the system measurement matrix C is computed as the identity matrix I_3. The measurement noise matrix, R, was initialized to $.5 * I_3$. With the computed measurement, the Kalman gain can be computed from the estimated covariance matrix, system measurement matrix, and the measurement noise matrix using the following equation.

$$K_t = \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1} \tag{11}$$

This is used to estimate the posterior μ_t and update the covariance matrix Σ_t.

$$\mu_t = \bar{\mu}_t + K_t(z_t - C\bar{\mu}_t) \tag{12}$$

$$\Sigma_t = \bar{\Sigma}_t - K_t C\bar{\Sigma}_t \tag{13}$$

During each step the gyro measurements are used as the inputs to the process model to get a predicted state. Then, using the measured state calculated from the acceleration measurements, the Kalman gain is computed. This determines the weight the filter puts on the measured values against the predicted values. Along with this the covariance matrix represents the confidence in the process model versus the measurement model.

For the data set the process and measurement noise matrix were initialized as follows.

$$Q = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{14}$$

$$R = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{15}$$

V. Unscented Kalman Filter

The Kalman filter generally provides good estimates of the system attitude; however, it makes assumptions that are not always viable for certain systems. The normal Kalman filter assumes process models can be estimated with a linear model. The advantage of a Unscented Kalman filter is that during the process the estimated state and the estimated covariance matrix are processed through the actual system dynamics. The system is able to handle nonlinear processes as it estimates the state and covariance through numerous sigma points run through the system's nonlinear processes. The filter starts by initializing in the same way as the normal Kalman filter with a process noise matrix Q, measurement noise matrix R and a covariance matrix P. From here the the disturbances $\mathcal{W}$, to generate the sigma points are calculated from the covariance and process noise matrices with the equations below where S is the matrix square root of the equation using the *Cholesky Decomposition*. For attitude estimation we have a state vector of 7 states.

$$x = [q_w q_x q_y q_z \omega_x \omega_y \omega_z]^T \tag{16}$$

This is composed of the attitude quaternion and the Euler rotation rates. The states has 6 degrees of freedom so we use a dimensionality n of 6. The filter initializes P to and nxn matrix of zeros as the covariance will quickly settle to the correct value. Additionally the sigma points were calculated using $\sqrt{n}$ instead of $\sqrt{2n}$ as described in 2 to improve tracking.

$$\begin{aligned} S &= \sqrt{P_{t-1} + Q} \\ \mathcal{W}_{i,i+2n} &= columns(\pm\sqrt{n}S) \end{aligned} \tag{17}$$

This results in 12 sigma points where each sigma point is a column of the resulting matrix in both the negative and positive case. The sigma points $\mathcal{X}$ can be computed using the disturbances above. The disturbances are of dimensionality 6 so the first 3 are applied to the state orientation (measured via quaterions) and the second 3 to the state angular rate.

$$(q_{\mathcal{W}})_i = [\cos(0.5*|\vec{\mathcal{W}_{1:3,i}}|*dt), \frac{\vec{\mathcal{W}_{1:3,i}}}{|\vec{\mathcal{W}_{1:3,i}}|}\sin(0.5*|\vec{\mathcal{W}_{1:3,i}}|*dt)]^T \tag{18}$$

$$(\omega_{\mathcal{W}})_i = [\mathcal{W}_{4,i}, \mathcal{W}_{5,i}, \mathcal{W}_{6,i}]^T \tag{19}$$

With this the actual sigma points are calculated by adding the disturbances to the current state.

$$\mathcal{X}_i = \begin{pmatrix} q_{t-1}q_{\mathcal{W}i} \\ \omega_{t-1} + \omega_{\mathcal{W}i} \end{pmatrix} \tag{20}$$

Once the sigma points have been computed the process model points can be computed. Each sigma point is run through the system process model to get individual state estimates $\mathcal{Y}_i$. For this system the process model propagates the attitude quaternion using the current rotation rates. The quaternion change q_Δ from the process model is computed in the same manner that the sigma point quaternion disturbance was computed however using the current angular velocity ω_{t-1} instead of the quaternion disturbance $\mathcal{W}_{1:3,i}$.

$$\mathcal{Y}_i = \begin{pmatrix} q_{t-1}q_{\mathcal{W}}q_\Delta \\ \omega_{t-1} + \omega_{\mathcal{W}} \end{pmatrix} \tag{21}$$

Instead of getting the estimated covariance matrix and estimated state through the linear process model and current covariance matrix in the normal Kalman filter, the estimated state and covariance is computed from the result of the sigma points. This allows the filter to model the non-linear process of the system.

The mean of the sigma points is calculated by iterating a mean attitude error until a mean attitude is converged upon. The iteration computes an initial quaternion q_t as the first sigma point. During each iteration, the error $\vec{e_i}$, from each sigma point to the quaternion q_t is calculated. The average of the these error vectors are taken to get an average error $\vec{e}$, which is then transformed back into a quaternion and applied to q_t to move it in the direction of the true mean attitude. This loop is iterated until the mean error has decreased below an acceptable threshold. This iteration process is described below.

$$\bar{\mu} = \begin{bmatrix} \bar{q} \\ \bar{\omega} \end{bmatrix} \tag{22}$$

With the mean state $\bar{\mu}$ from the sigma points the covariance estimate $\bar{P}$, and the mean centered sigma disturbances $\mathcal{W}'_i$ can be computed.

$$\mathcal{W}'_i = \begin{bmatrix} \mathcal{X}_i - \bar{x} \end{bmatrix} = \begin{bmatrix} q_{\mathcal{X}}(\bar{q})^{-1} \\ \omega_{\mathcal{X}} - \bar{\omega} \end{bmatrix} \tag{23}$$

$\mathcal{W}'_i$ is computed by multiplying the inverse of the mean attitude quaternion to the quaternion portions of the sigma points and subtracting the mean angular velocity from the angular velocity portion of the sigma points.

Data: $\mathcal{Y}_i$
Result: $\bar{\mathcal{Y}}$
$q_t = q_{\mathcal{Y},0}$
$\bar{\omega} = \frac{1}{2n}\sum \omega_i$
while *not looped for max iterations* **do**
  $q_{inv} = q_t^{-1}$
  for q_i *in* $\mathcal{Y}_i$ **do**
    $\vec{e_i} = q_i q_{inv}$
  end
  $\vec{e} = \frac{1}{2n}\sum \vec{e_i}$
  $q_t = \vec{e}q_t$
  if $\vec{e} < threshold$ **then**
    $\bar{q} = q_t$
  end
end

Algorithm 1: Mean Sigma Point Calculation

The covariance matrix can be computed with the mean centered disturbances $\mathcal{W}'_i$ now that they are centered around the mean sigma point.

$$\bar{P} = \frac{1}{2n}\sum \mathcal{W}'_i \mathcal{W}'^T_i \tag{24}$$

With the estimate and covariance estimates computed the process model propagation is complete. The measurement model starts by calculating the estimated system measurements $\bar{z}_i$ from the sigma points. g is the gravity frame represented through quaternions.

$$\bar{z}_i = \begin{bmatrix} \mathcal{Y}_q^{-1} g \mathcal{Y}_q \\ \mathcal{Y}_\omega \end{bmatrix}_i \tag{25}$$

The mean of these computed quaternions and angular velocities are computed into a mean measurement estimate $\bar{\mathcal{Z}}$.

$$\bar{\mathcal{Z}} = \frac{1}{2n}\sum \bar{z}_i \tag{26}$$

The covariance of the measurement estimates $P_z z$ is also computed from the measurement estimates.

$$P_{zz} = \frac{1}{2n}\sum [\bar{\mathcal{Z}} - \bar{z}_i][\bar{\mathcal{Z}} - \bar{z}_i]^T \tag{27}$$

From the measurement covariance the innovation covariance can be calculated as the sum of the measurement covariance and the measurement noise matrix.

$$P_{vv} = P_{zz} + R \tag{28}$$

In order to get the Kalman gain to update the state the cross correlation matrix P_{xz} must be computed.

$$P_x z = \frac{1}{2n}\sum [\mathcal{W}'_i][\bar{\mathcal{Z}} - \bar{z}_i]^T \tag{29}$$

With the updated covariance matrix and innovation covariance matrix the Kalman gain K can be computed.

$$K = P_{xz} P_{vv}^{-1} \tag{30}$$

With the Kalman gain computed the state estimate and the state covariance can be updated. The new covariance is the estimated covariance minus the product of the Kalman gain, innovation covariance matrix, and Kalman gain transpose. The state is calculated as the previous state plus the product of the Kalman gain and the difference between the measurement readings $\hat{z}$ and the estimated measurement readings.

$$P_t = \hat{P}_{t-1} - KP_{vv}K^T \tag{31}$$

$$\hat{x}_t = \hat{x}_{t-1} + K(\hat{z} - \bar{\mathcal{Z}}) \tag{32}$$

For the data set the process and measurement noise matrix were initialized as follows.

$$Q = \begin{bmatrix} 105 & 0 & 0 & 0 & 0 & 0 \\ 0 & 105 & 0 & 0 & 0 & 0 \\ 0 & 0 & 105 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 \end{bmatrix} \tag{33}$$

$$R = \begin{bmatrix} 11.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 11.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 \end{bmatrix} \tag{34}$$

VI. Results

A. Training Data Sets

Euler angles for the training data sets are shown in Figures 1, 2, 3, 4, 5, and 6.

B. Test Data Sets

Euler angles for the real test data sets are shown in Figures 7, 8, 9, and 10.

VII. Videos

We have uploaded our video results to YouTube, and provide links to each test set here. Note that the 'real' sets will only have 4 plots.

A. Training Data Videos

1) https://youtu.be/VM1YS0bOKDg
2) https://youtu.be/P8SSOgIim8U
3) https://youtu.be/1cOpra$_F sxI$
4) https://youtu.be/V6YbjHI7Ep4
5) https://youtu.be/EjkcpqC0IRo
6) https://youtu.be/FQUTHbK2JIQ

B. Real Data Videos

1) https://youtu.be/3Qpu9Kg0Y2s
2) https://youtu.be/jKs2lmPqCck
3) https://youtu.be/$_V G5ZU0MloU$
4) https://youtu.be/PkIjVZsywTw

VIII. Important Lessons Learned

1) Gain sensitivity: Through implementing the Unscented Kalman filter we found the output to be extremely dependent on the choice of noise matrices. The filter would only give a remotely reasonable estimate for very narrow band of gains. In particular the process noise matrix Q would result wildly different outputs for small changes. Initially during debugging it was difficult to determine if errors in the output were attributed to incorrect implementation of the filter or incorrect gains. In the end, tuning of gains took comparable time to the actual creation of the filter.

2) Training on multiple data sets: As a result of the high degree of tuning gains described in the previous section, the final tune of the filter gave better results for some sets over others. This resulted to an 'overtuning' for certain scenarios, which had to be tuned back to allow for superior overall performance.

3) Use of numpy Libraries: Tracking and resolving differences in notation between the Kraft paper 2 and other sources proved to be a challenge, especially when debugging. However, taking the time to understand each equation lead to some pretty handy discoveries in numpy. Most relevant was that the $np.cov$ function can calculate the biased covariance of state vectors and sigma points without us having to deal with transforming between quaternions and Euler angles. It also allowed us to remove quite a bit of custom, very buggy code.

IX. Conclusion

It is clear from the test and actual data presented here that this implementation of the UKF filter outperformed both the standard Kalman filter and the Madgwick filter. This is primarily apparent in the test data sets where the UKF can be seen to follow the Vicon data most closely. Interestingly, there are small number of cases where none of the filters implemented here adequately align with the Vicon data. One such case can be seen in the Yaw values for test case 4, where none of the filters appear to track the true orientation of the platform particularly well.

Significantly, it can be seen that even in cases where the UKF orientation does not align with the Vicon data for a period of time, that the filter will quickly 'recover' from these erroneous values and resume tracking the Vicon data.

Acknowledgment

The authors would like to thank the professors for this course, Nitin J. Sanket and Chahat Deep Singh, as well as Dr. Inderjit Chopra.

References

[1] S. O. H. Madgwick, *An efficient orientation filter for inertial and inertial/magnetic sensor arrays*. University of Bristol Rep., 2010

[2] Kraft, Edgar. "A quaternion-based unscented Kalman filter for orientation tracking." Proceedings of the Sixth International Conference of Information Fusion. Vol. 1. 2003.

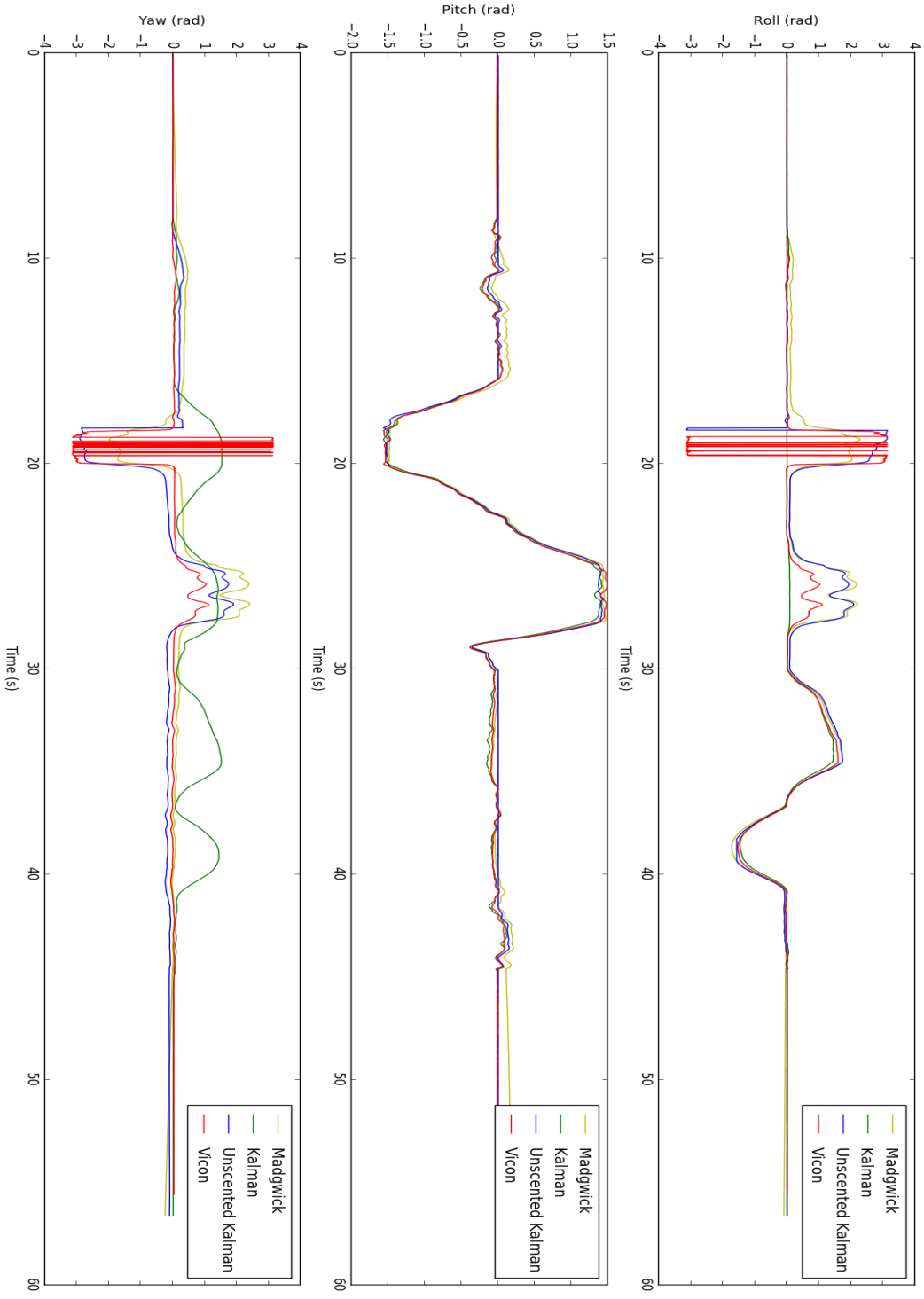

Fig. 1. Euler angles for training data set 1.

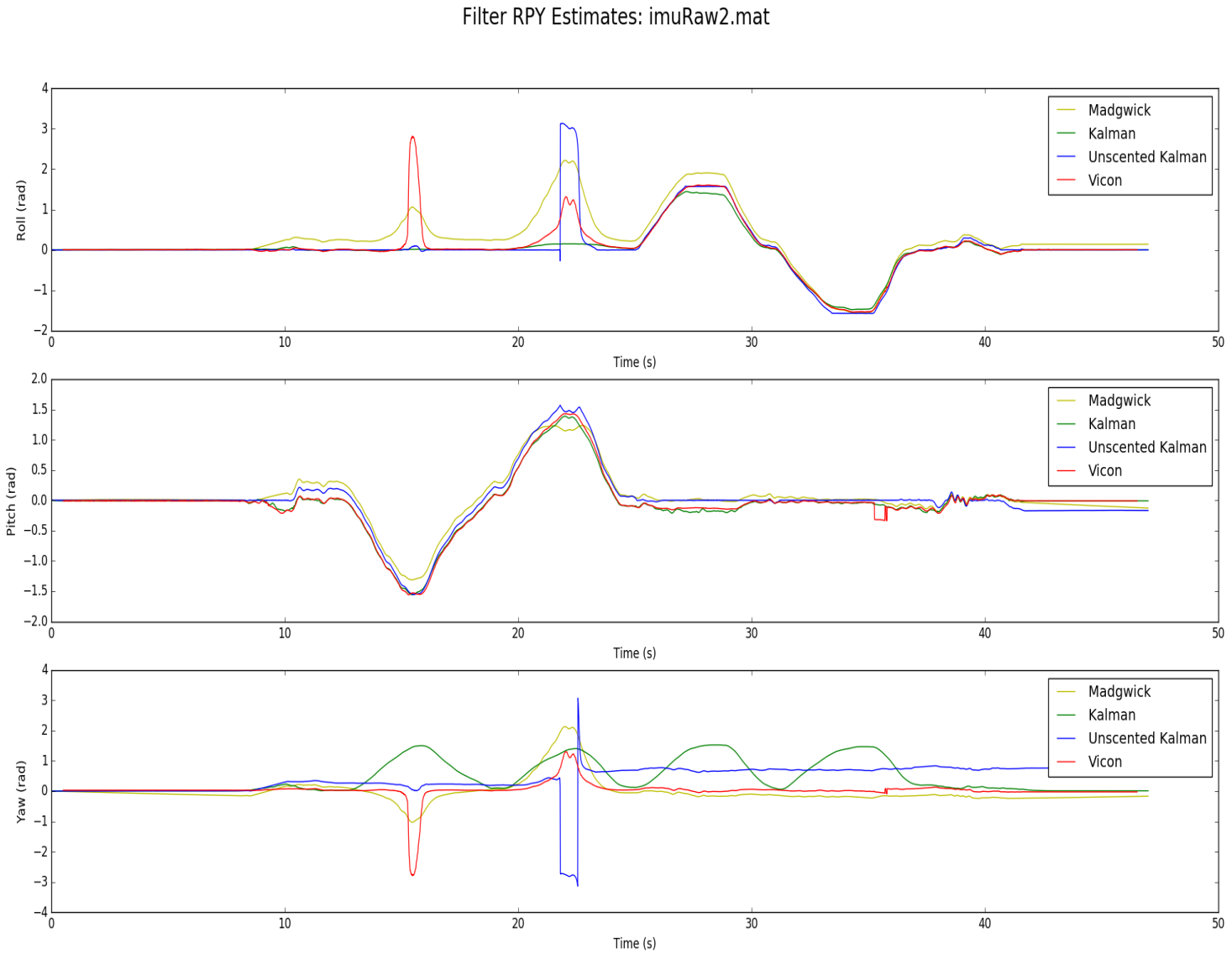

Fig. 2. Euler angles for training data set 2.

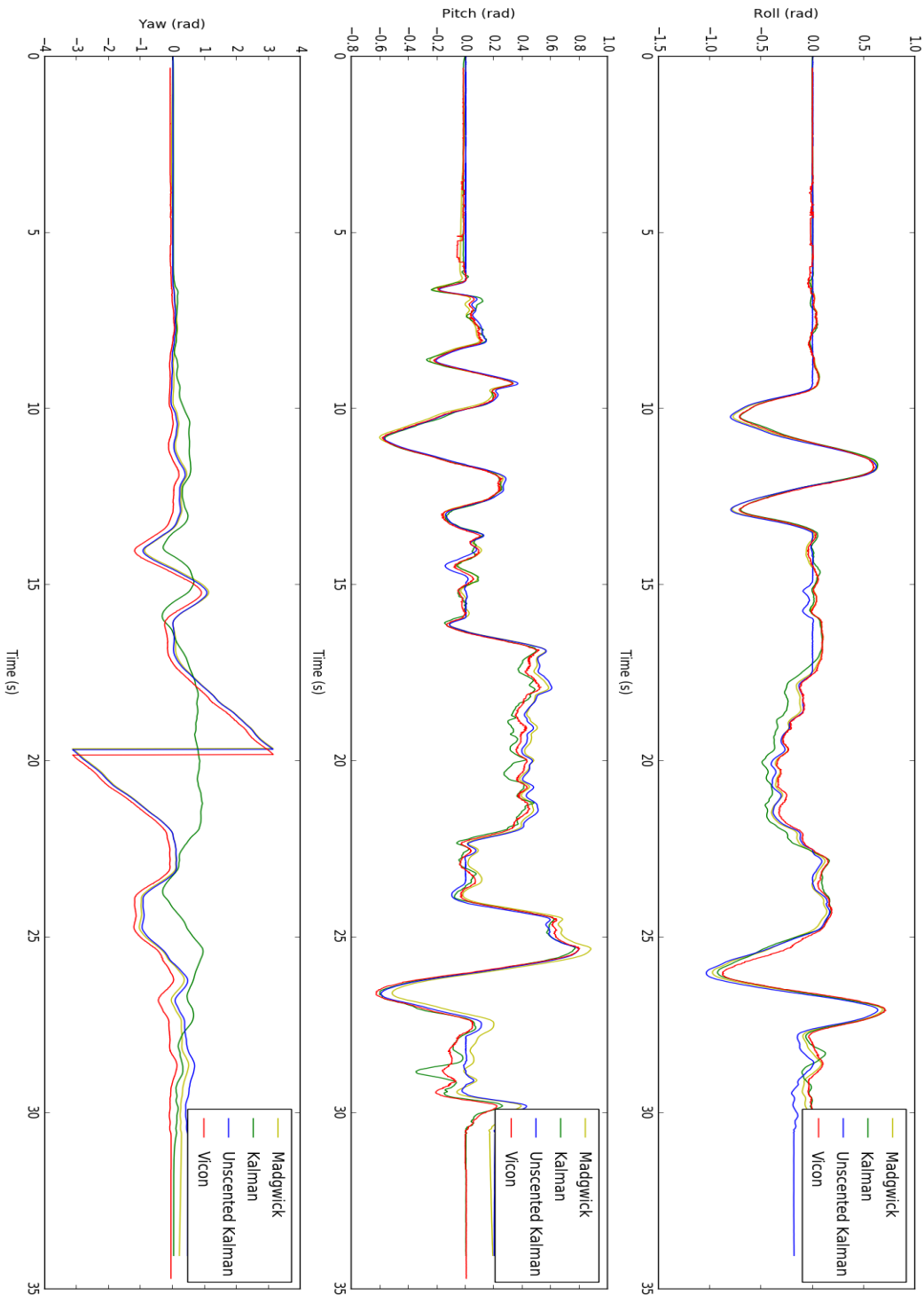

Fig. 3. Euler angles for training data set 3.

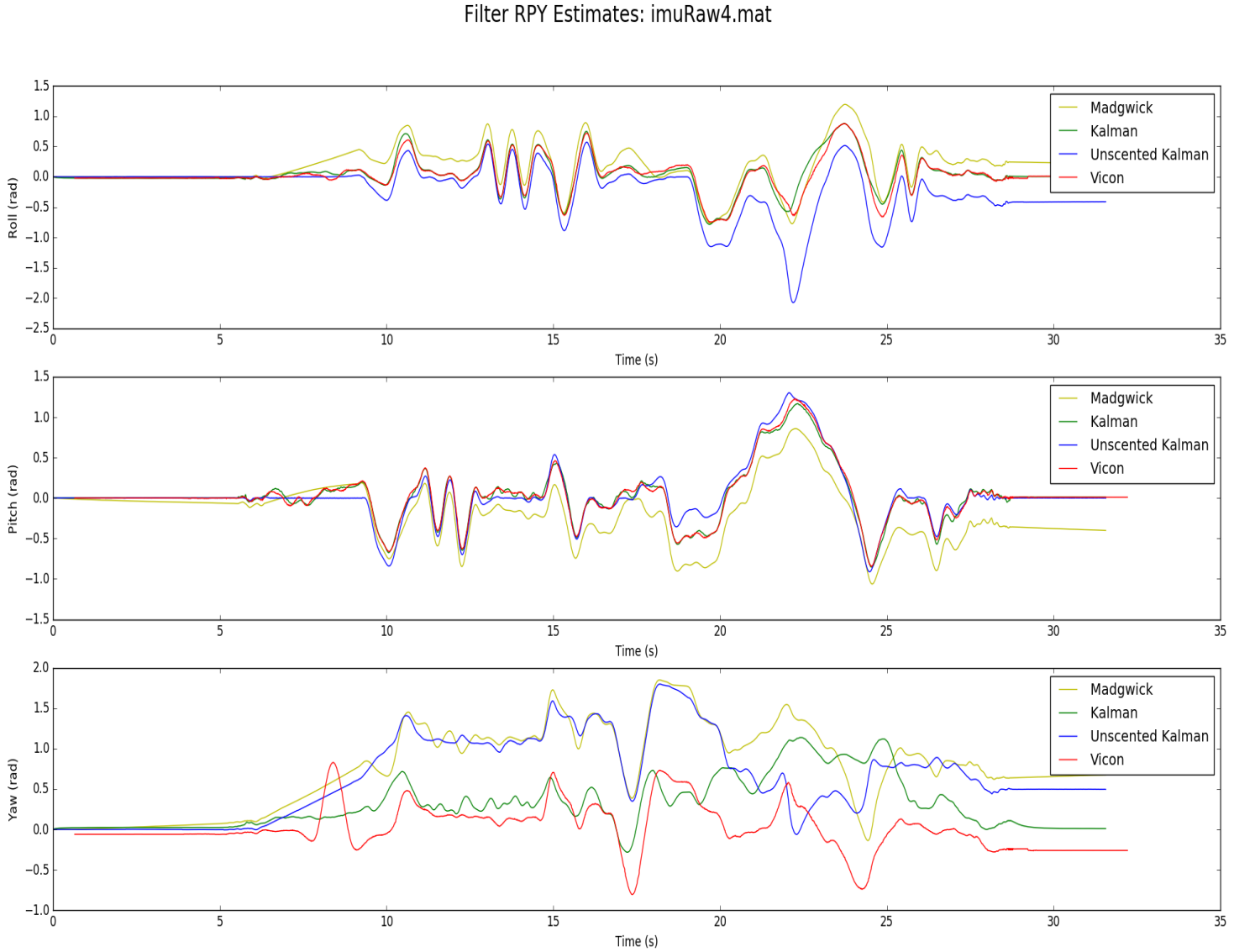

Fig. 4. Euler angles for training data set 4.

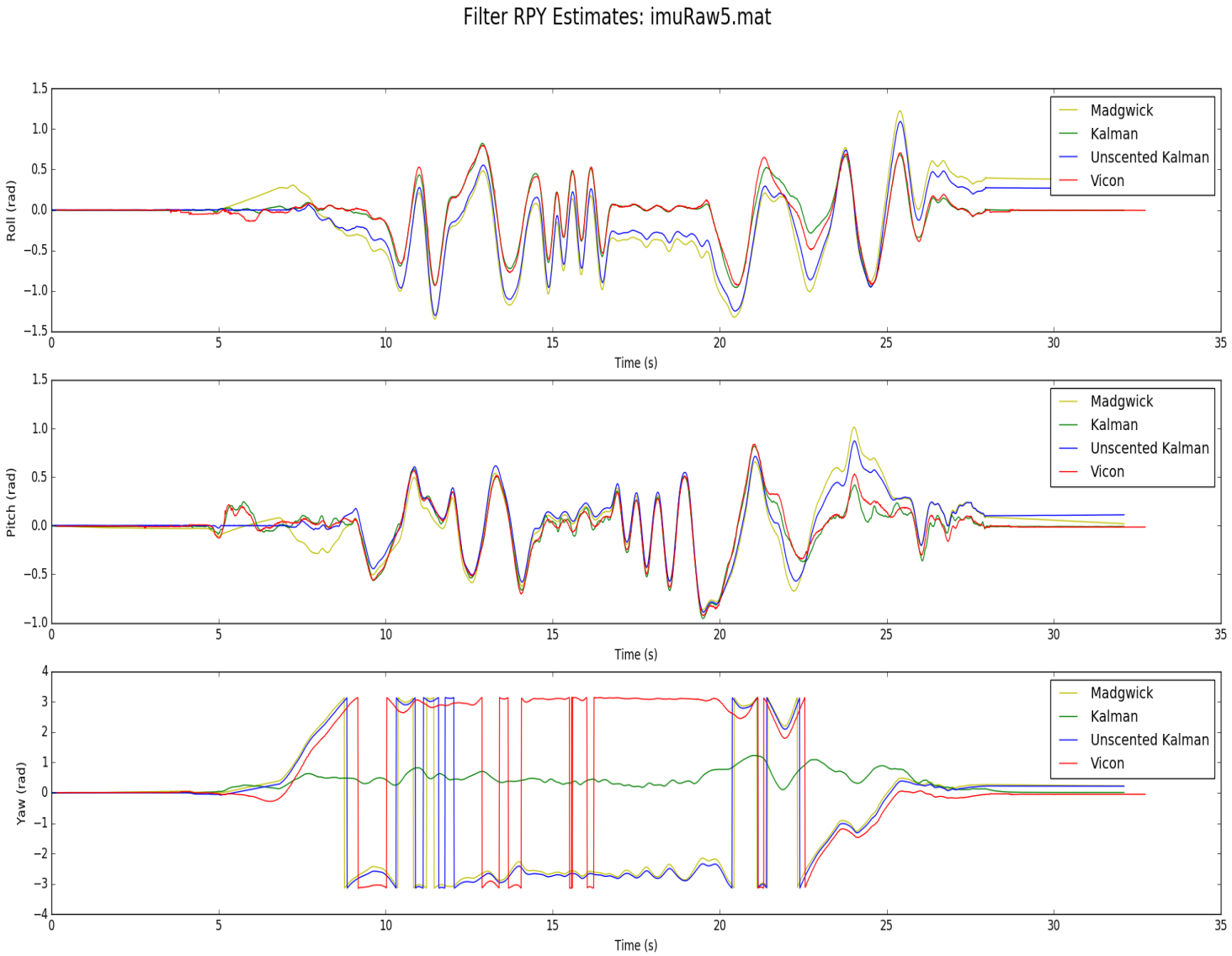

Fig. 5. Euler angles for training data set 5.

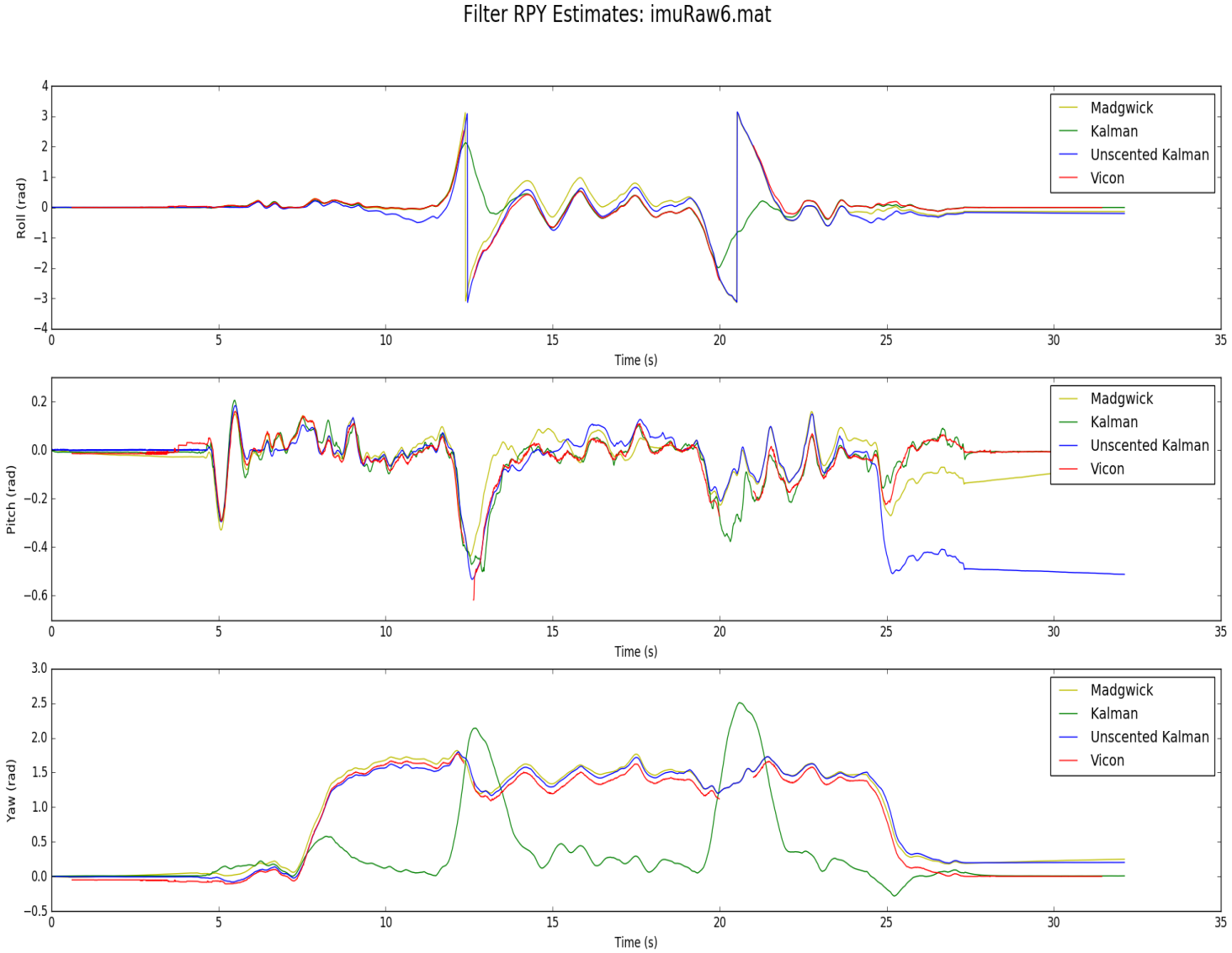

Fig. 6. Euler angles for training data set 6.

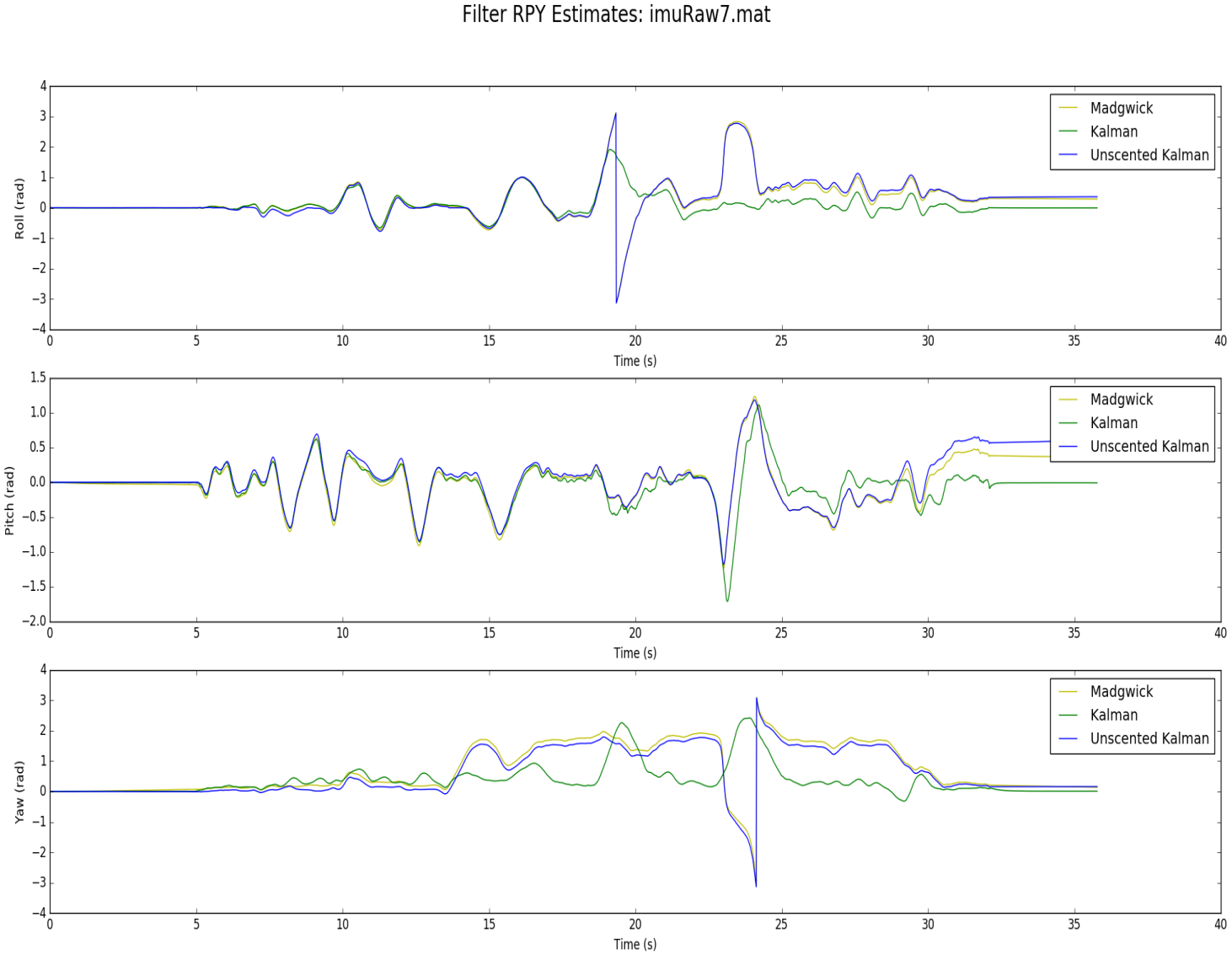

Fig. 7. Euler angles for real data set 1.

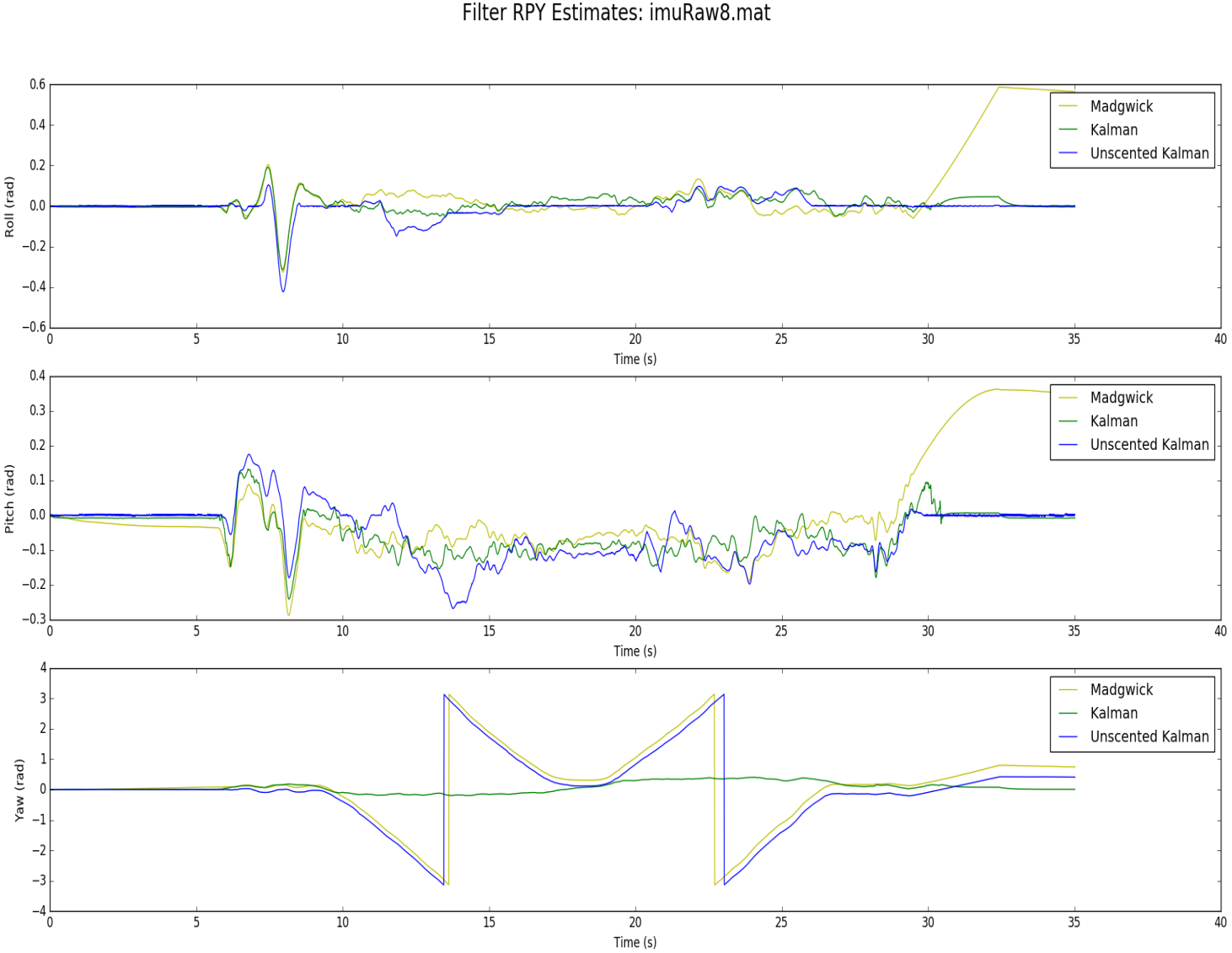

Fig. 8. Euler angles for real data set 2.

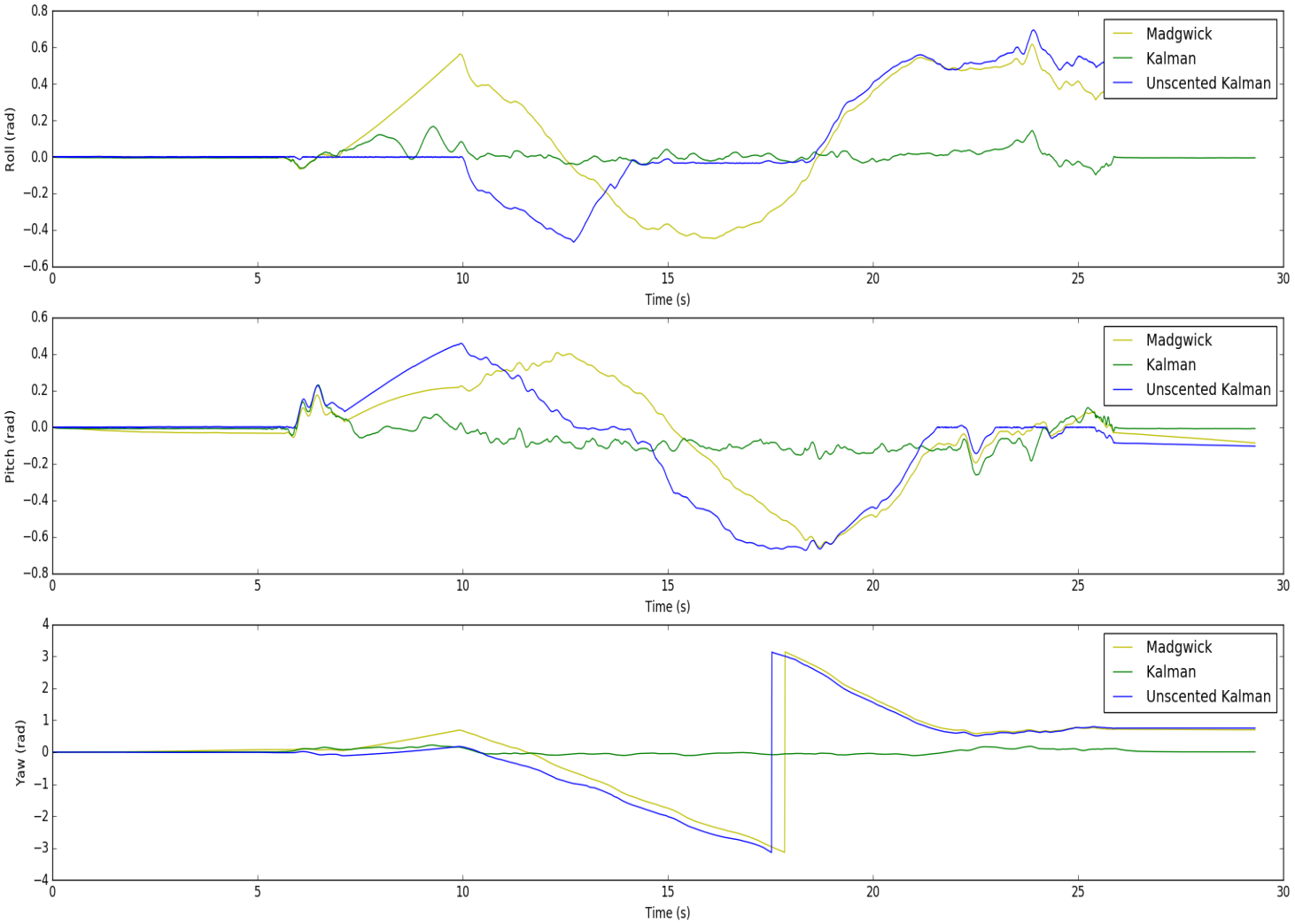

Fig. 9. Euler angles for real data set 3.

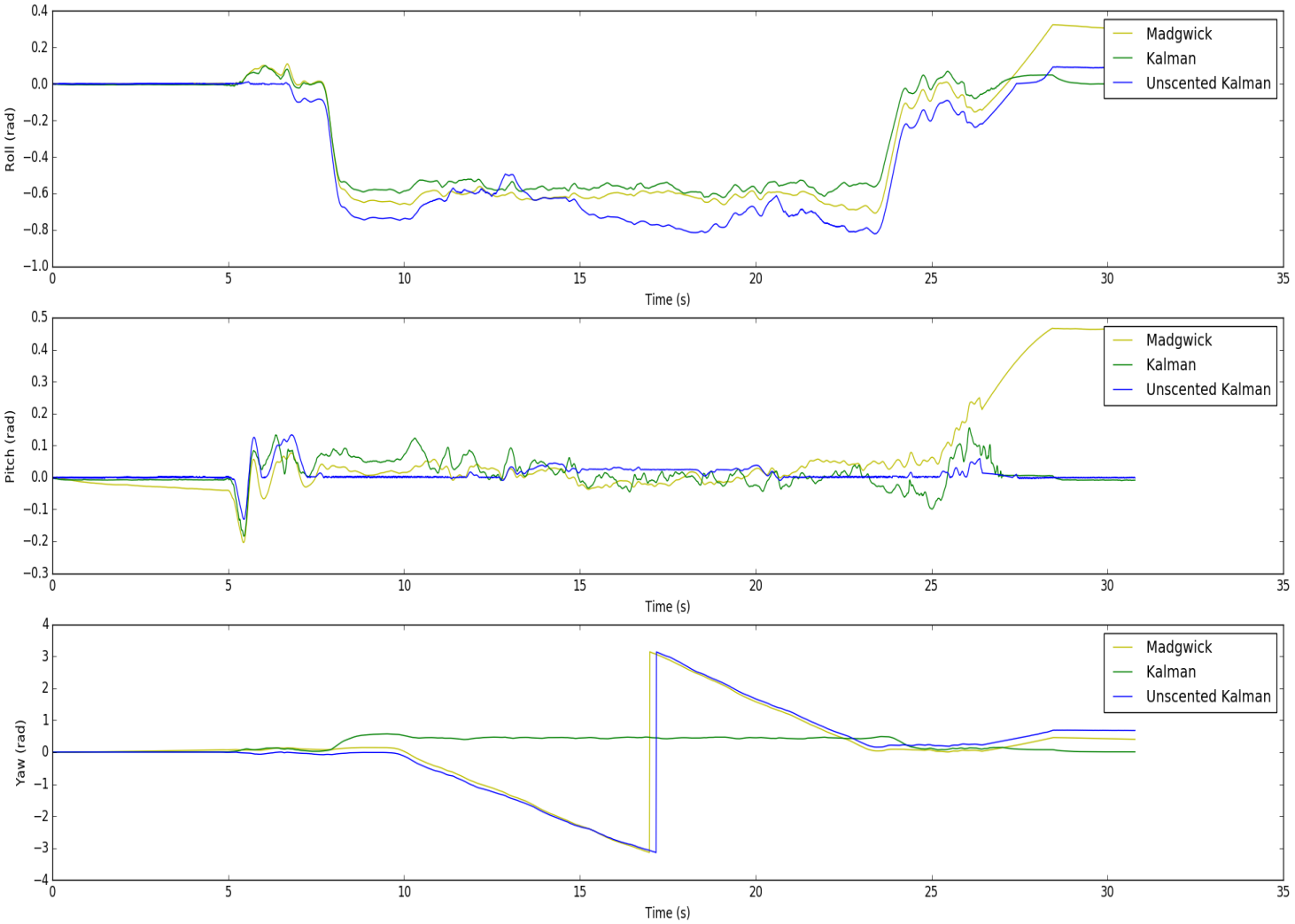

Fig. 10. Euler angles for real data set 4.