ENAE788M Assignment 2 - Attitude Estimation using Unscented Kalman Filter (UKF) Filter

Estefany Carrillo, Mohamed Khalid M, and Sharan Nayak

I. INTRODUCTION

In this project, we have implemented the Unscented Kalman Filter (UKF) to estimate orientation values from IMU data. This filter uses a quaternion representation of orientation and incorporates process and nonlinear measurement models in order to generate an estimate. A key aspect to ensure of when implementing this filter is to express the state vector as a combination of a unit quaternion for orientation and a vector for angular velocity. Another key aspect is that the process noise should be applied before the process model.

In the following sections, we present a description of our implementation of the UKF filter and the parameter values used to tune the resulting estimation.

A. Generating Sigma Points

The provided gyroscope data has angular velocity data in the order ω_z, ω_x, ω_y respectively. We set the order to the usual convention of ω_x, ω_y and ω_z. The bias b_g is calculated by taking the mean of first 300 samples. The gyroscopic values are then converted to physical units of rad/s.

We extract the initial rotation matrix from the Vicon data corresponding to the timestamp closest to the first IMU timestamp. We represent this matrix as a quaternion and use it as the initial quaternion. We extract the first angular velocity vector from the IMU data and combine it with the initial quaternion to obtain the initial state vector.

We set the estimate error covariance P_{k-1} as $P_{k-1} = 0.2 \times I_{6\times6}$ and the process noise covariance Q as $Q = 0.1 \times I_{6\times6}$. We then use these two matrices to compute the square root of their sum via a *Cholesky Decomposition*. From the resulting square root, we form the matrix $\mathcal{W}_i$ with $2n$ columns. Combining the quaternion and angular velocity extracted from each column of $\mathcal{W}_i$ with our previous estimate x_{k-1}, we obtain the set of disturbed vectors $\mathcal{X}_i$ containing $2n$ *sigma* points.

B. Transformations of the sigma points

In this step, the differential rotation expressed as quaternion is multiplied by the quaternion in each *sigma* point of $\mathcal{X}_i$. This quaternion combined with the angular velocity vector in the corresponding *sigma* point results in the projected point that goes into the set of state vectors $\mathcal{Y}_i$.

C. Computation of the mean

From this set, we compute the mean of $\mathcal{Y}_i$ and set it to the priori estimate. We compute the *barycentric* mean for the angular velocity components in $\mathcal{Y}_i$ and perform the intrinsic grade descent algorithm to compute the mean of the orienatation component in in $\mathcal{Y}_i$. In the algorithm, we set the maximum number of iterations to 50 and an error threshold of 0.00000000000001. We observed that it only took a few iterations to generate a value within the threshold specified. Then, using the a priori state vector estimate, we compute the covariance and conclude with the time update step.

Similarly, we obtain the measurement mean and covariance. The difference between the measured value and the measurement estimate is used to compute the innovation covariance P_{vv}. We set the measurement nose R as $R = 0.1 \times I_{6\times6}$. Then, we obtain the cross correlation matrix P_{xz} from the set of projected measurement vectors $\mathcal{Z}_i$ and $\mathcal{W}_i$.

D. Kalman Gain and Update

From the obtained cross correlation matrix and covariance matrix, we compute the Kalman gain K_k and use it to calculate the a posteriori estimate. This concludes the measurement update step.

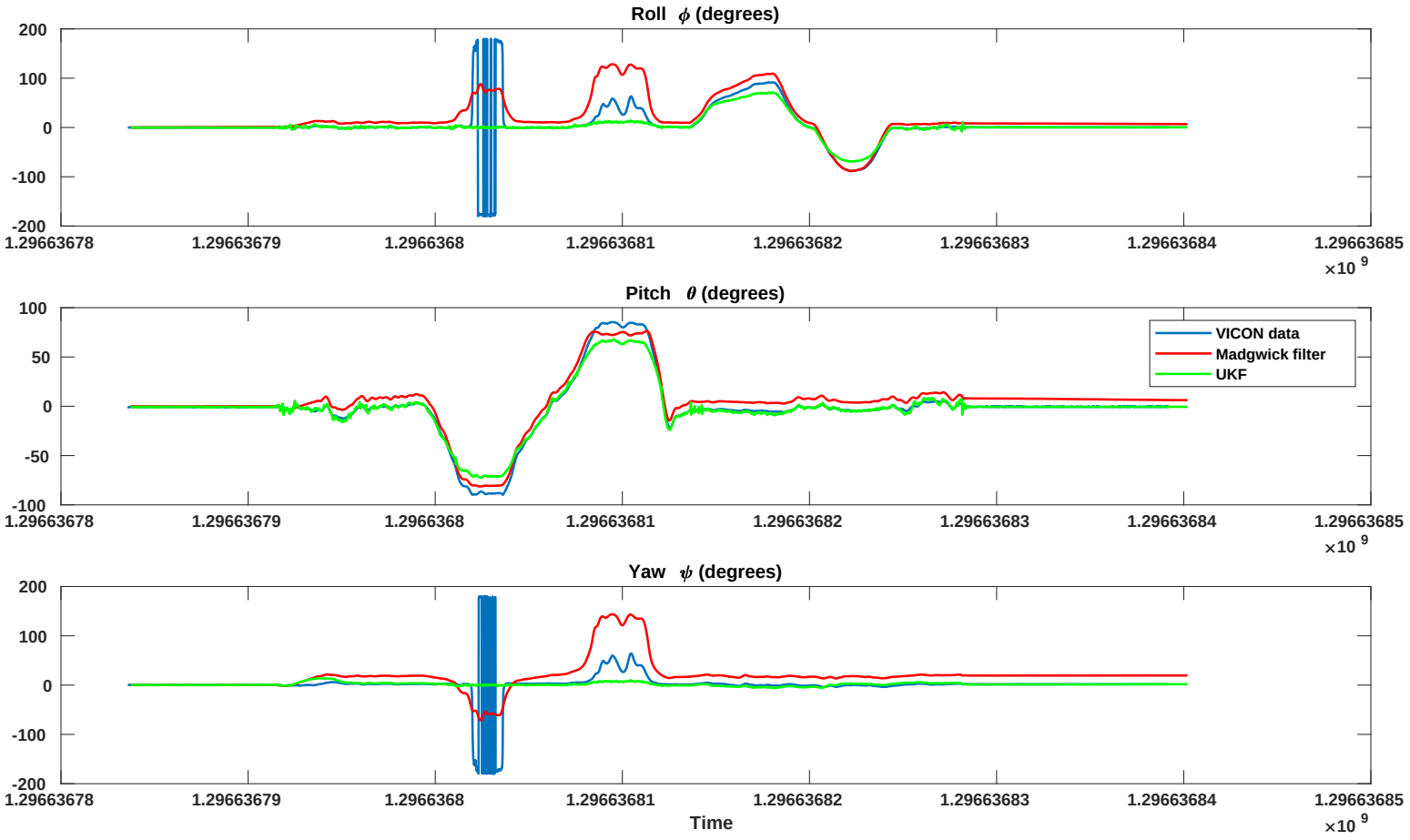
E. Implementation Details

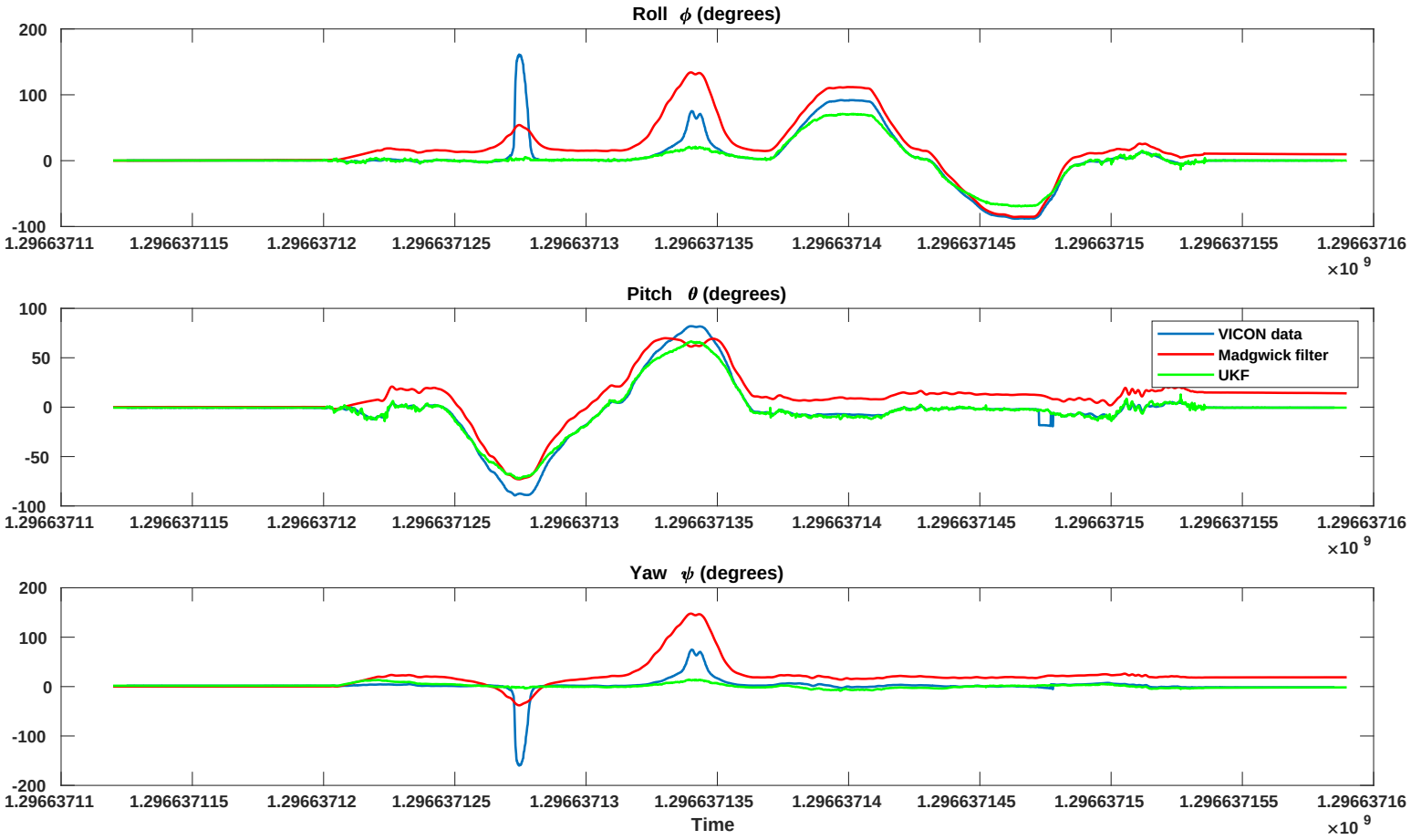
The UKF filter was implemented in Python 2.7. We utilized the following libraries: time, numpy, io, Rotation from scipy, math, pyplot, and in particular numpy.linalg. The later was used to perform the *Cholesky Decomposition*. A couple of useful functions were implemented including: $convertQuaternionToVector$ and $qconvertVectorToQuaternion$, which helped us in going from one representation to the other.

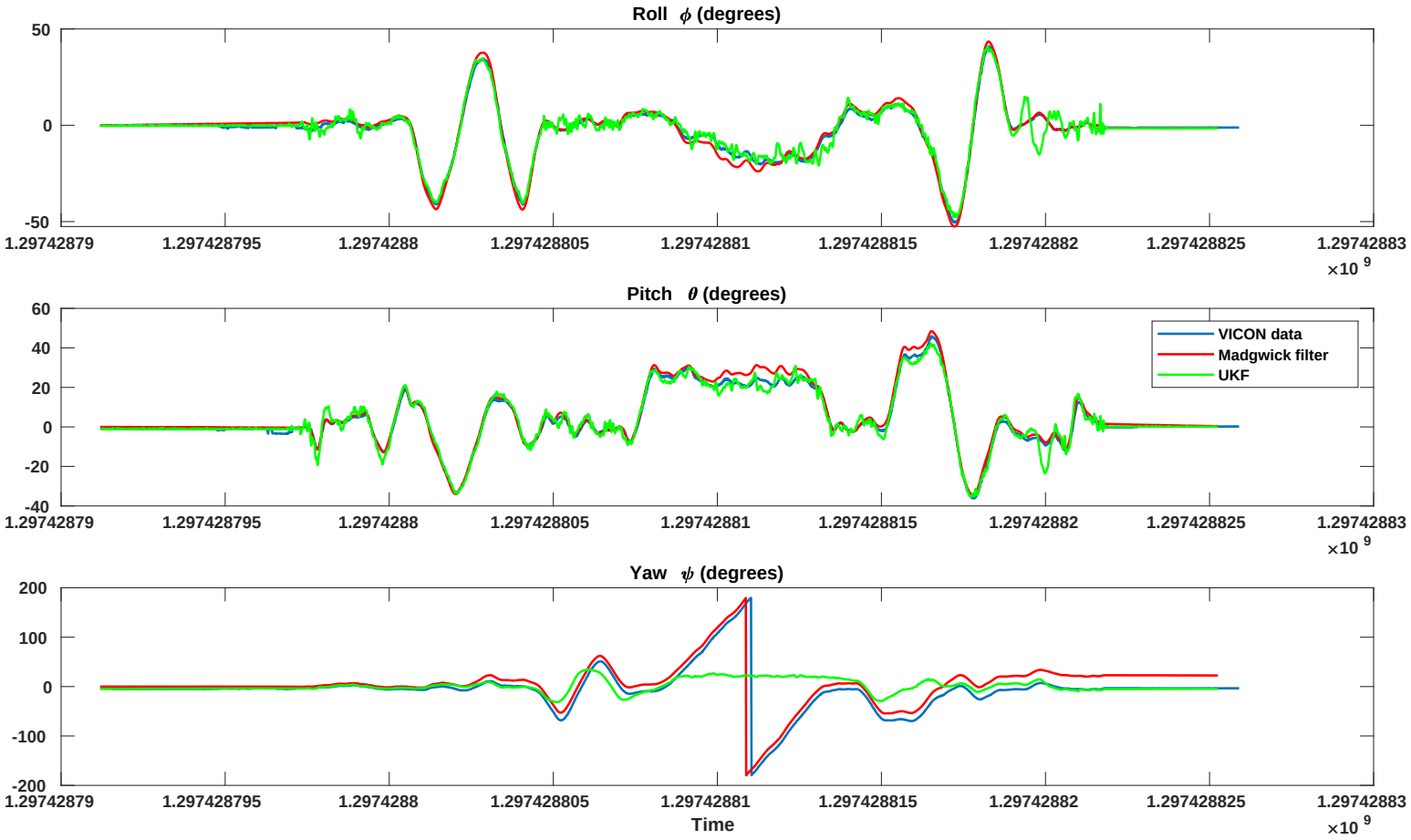
F. Analysis and Results

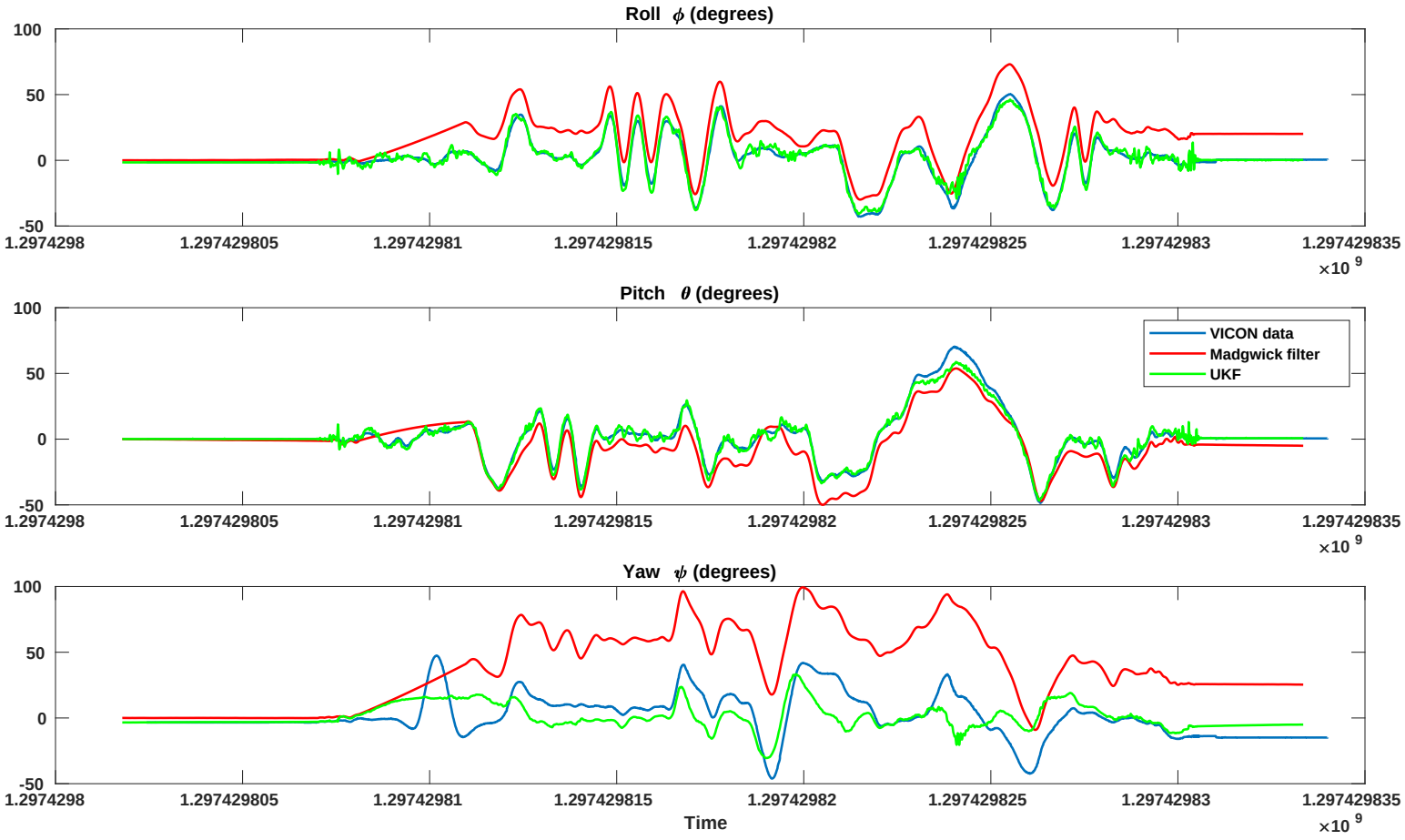
We have included plots of all training data sets and test data sets in the following pages starting with train data set 1 and ending with test data set 10. From these results, we were able to make a few observations. For example, it can be observed in the first plot, that while the UKF estimate tends to be lower than the Vicon estimate, the Madgwick filter estimates tends to be higher. To improve the UKF estimate, a possible solution would be to increase the Kalman gain.

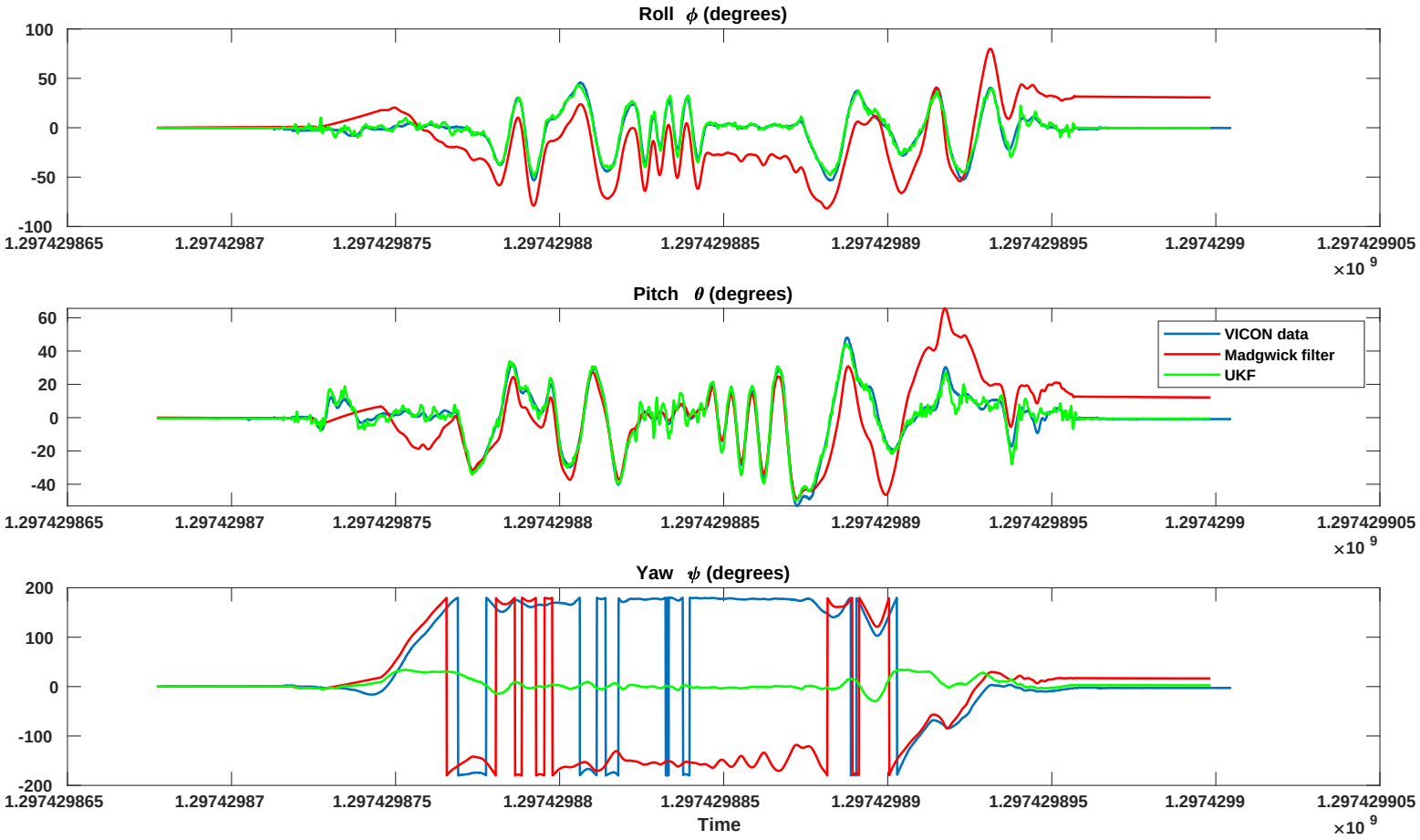
Another observation we make is that in the time instances during which the Vicon estimates appear to be oscillating rapidly, the Madgwick filter appears to be more successful in following these changes while the UKF stays constant. This can be observed in a couple of plots particularly for the yaw estimation. This could be a potential issue when relying on UKF estimates especially in the cases where there are sudden changes to attitude as it appears that our filter implementation does not follow these changes. Despite trying different tunning of the parameters, the estimate from the UKF did not capture these rapid or abrupt changes in attitude.

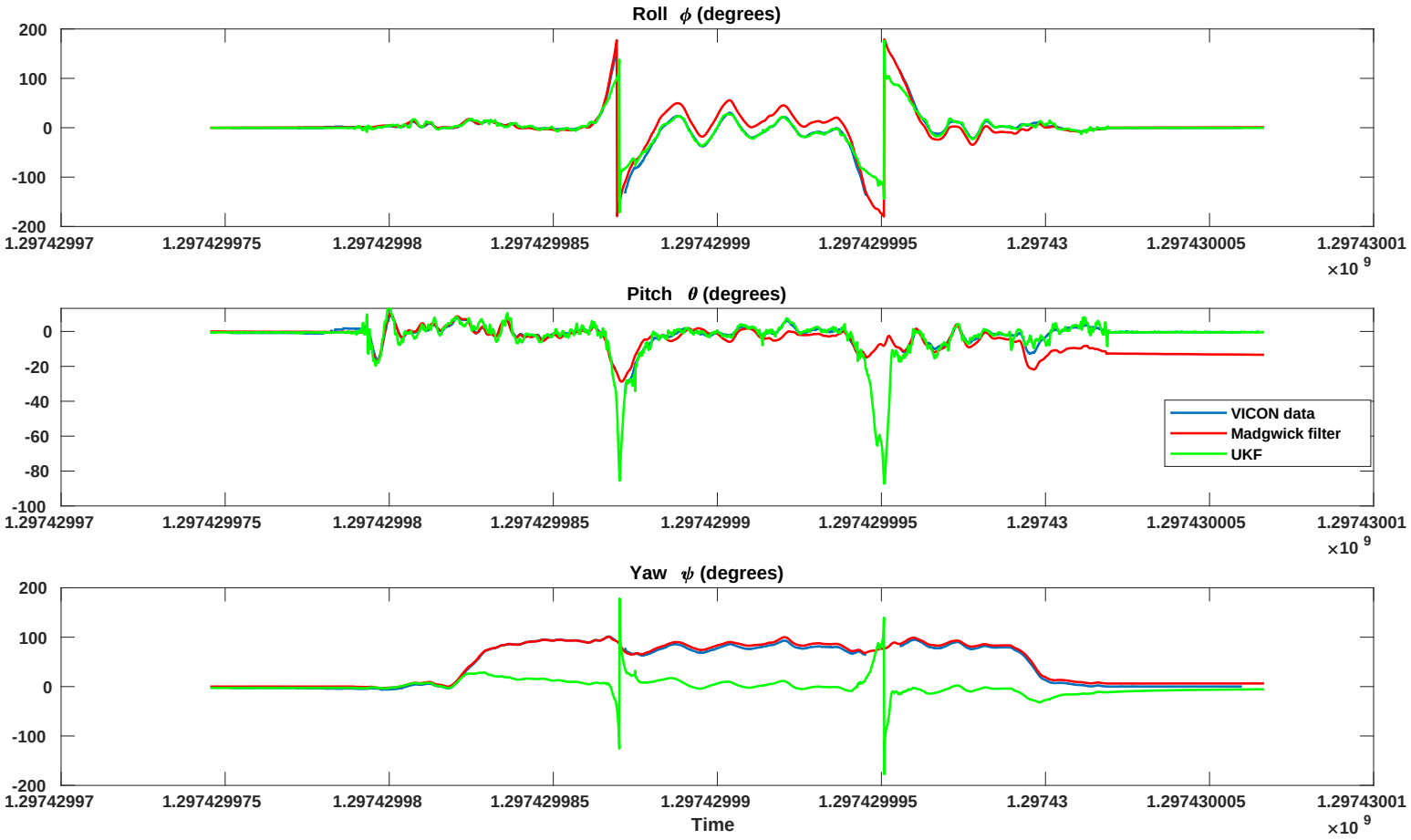

Roll ϕ (degrees)
200
100
0
-100
-200
1.29663678
1.29663679
1.2966368
1.29663681
1.29663682
1.29663683
1.29663684
1.29663685
×10 9
Pitch θ (degrees)
100
50
0
-50
-100
VICON data
Madgwick filter
UKF
Yaw ψ (degrees)
Time

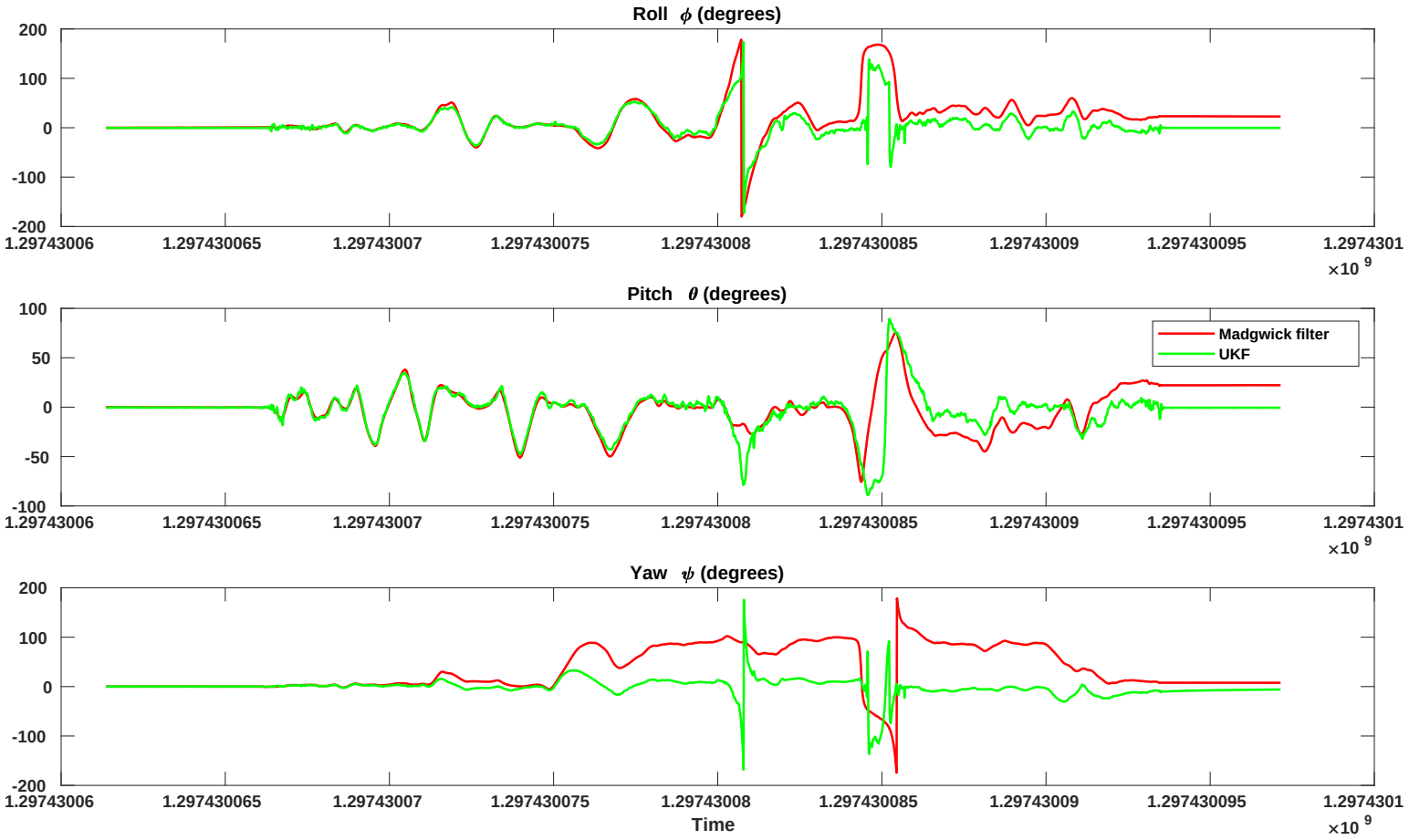

Roll ϕ (degrees)
200
100
0
-100
1.29663711
1.296637115
1.29663712
1.296637125
1.29663713
1.296637135
1.29663714
1.296637145
1.29663715
1.296637155
1.29663716
×10 9
Pitch θ (degrees)
100
50
0
-50
-100
VICON data
Madgwick filter
UKF
Yaw ψ (degrees)
200
100
0
-100
-200
Time

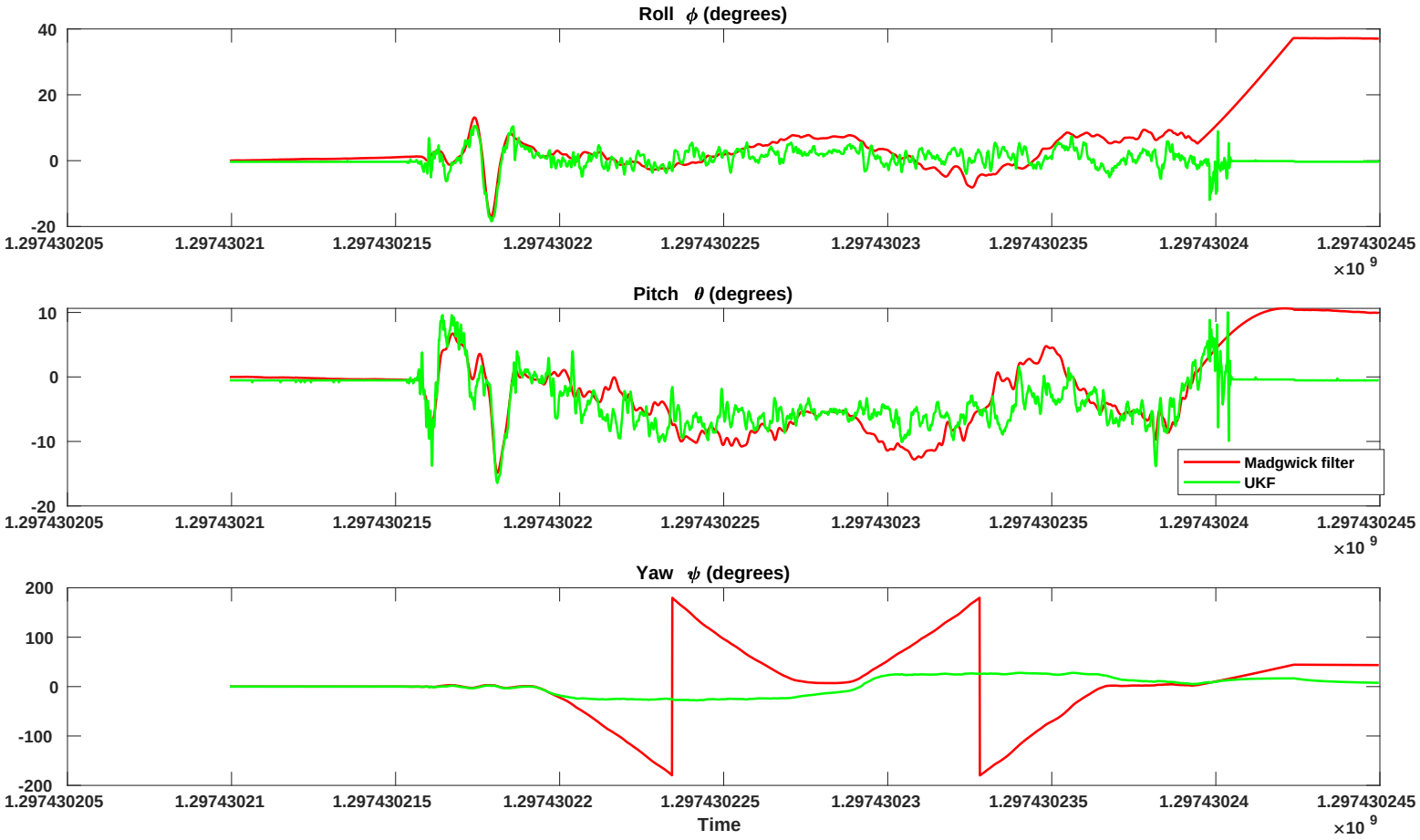

Roll ϕ (degrees)
50
0
-50
1.29742879
1.297428795
1.2974288
1.297428805
1.29742881
1.297428815
1.29742882
1.297428825
1.29742883
×10 9
Pitch θ (degrees)
60
40
20
0
-20
-40
VICON data
Madgwick filter
UKF
Yaw ψ (degrees)
200
100
0
-100
-200
Time

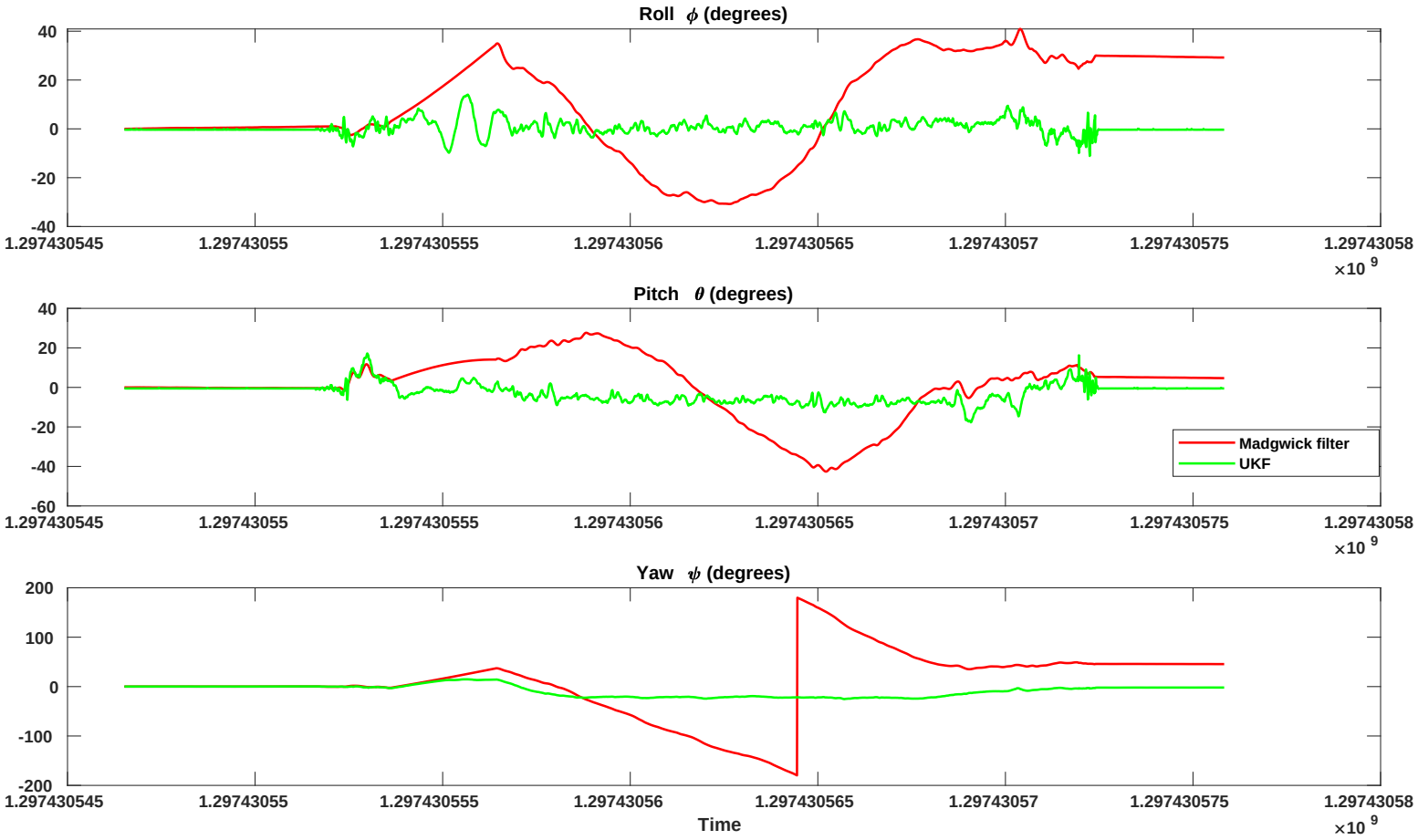

Roll ϕ (degrees)
100
50
0
-50
1.2974298
1.297429805
1.29742981
1.297429815
1.29742982
1.297429825
1.29742983
1.297429835
×10 9
Pitch θ (degrees)
VICON data
Madgwick filter
UKF
Yaw ψ (degrees)
Time

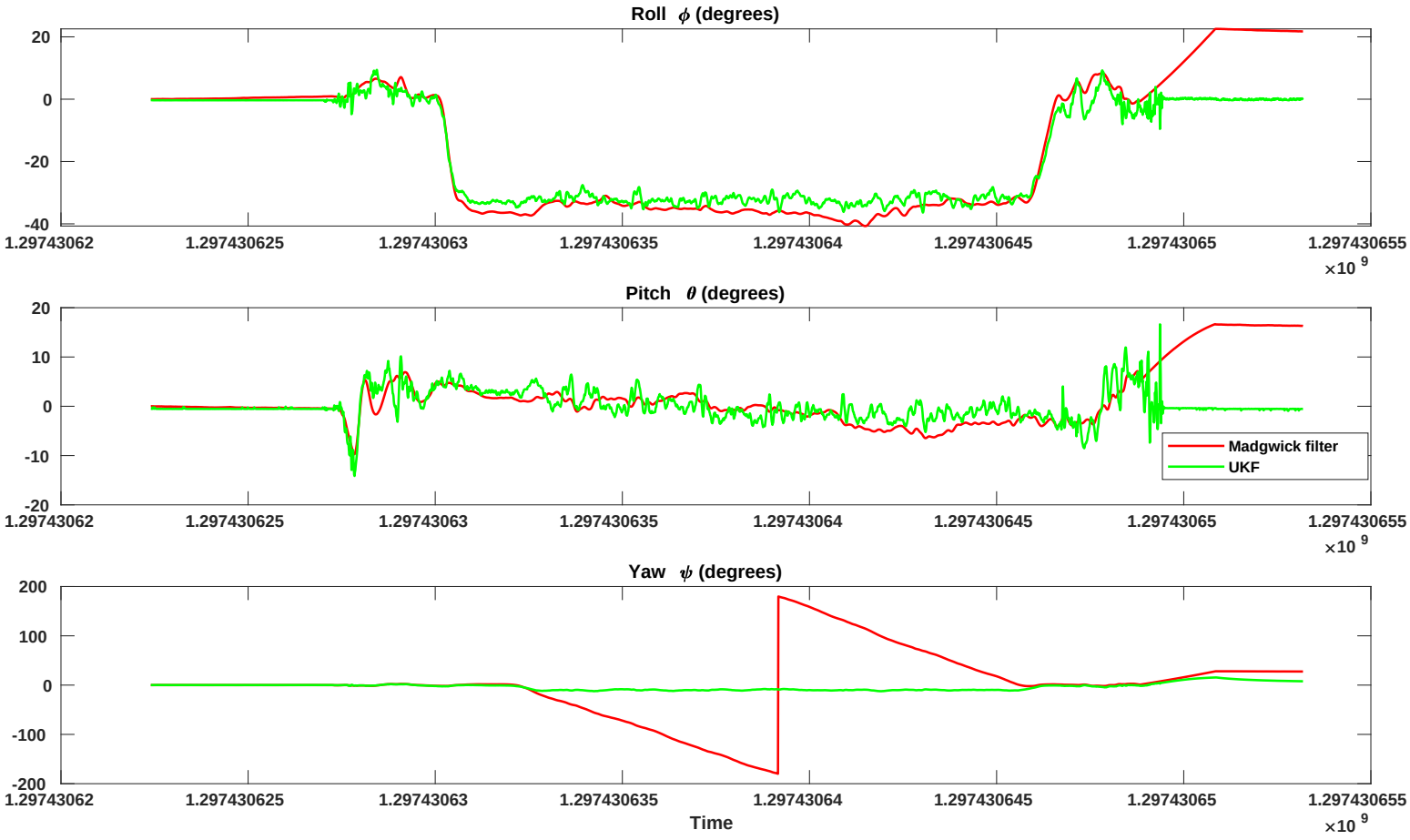

Roll ϕ (degrees)
100
50
0
-50
-100
1.297429865
1.29742987
1.297429875
1.29742988
1.297429885
1.29742989
1.297429895
1.2974299
1.297429905
×10 9
Pitch θ (degrees)
60
40
20
0
-20
-40
VICON data
Madgwick filter
UKF
Yaw ψ (degrees)
200
100
0
-100
-200
Time


Roll ϕ (degrees)
200
100
0
-100
-200
1.29742997
1.297429975
1.29742998
1.297429985
1.29742999
1.297429995
1.29743
1.297430005
1.29743001
×10 9
Pitch θ (degrees)
0
-20
-40
-60
-80
-100
VICON data
Madgwick filter
UKF
Yaw ψ (degrees)
Time


Roll ϕ (degrees)
200
100
0
-100
-200
1.29743006
1.297430065
1.29743007
1.297430075
1.29743008
1.297430085
1.29743009
1.297430095
1.2974301
×10 9
Pitch θ (degrees)
100
50
0
-50
-100
Madgwick filter
UKF
Yaw ψ (degrees)
Time


Roll ϕ (degrees)
40
20
0
-20
1.297430205
1.29743021
1.297430215
1.29743022
1.297430225
1.29743023
1.297430235
1.29743024
1.297430245
×10 9
Pitch θ (degrees)
10
0
-10
-20
Madgwick filter
UKF
Yaw ψ (degrees)
200
100
0
-100
-200
Time


Roll ϕ (degrees)
40
20
0
-20
-40
1.297430545
1.29743055
1.297430555
1.29743056
1.297430565
1.29743057
1.297430575
1.29743058
×10 9
Pitch θ (degrees)
40
20
0
-20
-40
-60
Madgwick filter
UKF
1.297430545
1.29743055
1.297430555
1.29743056
1.297430565
1.29743057
1.297430575
1.29743058
×10 9
Yaw ψ (degrees)
200
100
0
-100
-200
1.297430545
1.29743055
1.297430555
1.29743056
1.297430565
1.29743057
1.297430575
1.29743058
×10 9
Time


Roll ϕ (degrees)
20
0
-20
-40
1.29743062
1.297430625
1.29743063
1.297430635
1.29743064
1.297430645
1.29743065
1.297430655
×10 9
Pitch θ (degrees)
20
10
0
-10
-20
Madgwick filter
UKF
Yaw ψ (degrees)
200
100
0
-100
-200
Time