Andrew Gates

Professor Nikolay Atanasov

ECE 276A

22 November 2017

Project 2: Orientation Tracking

Introduction:

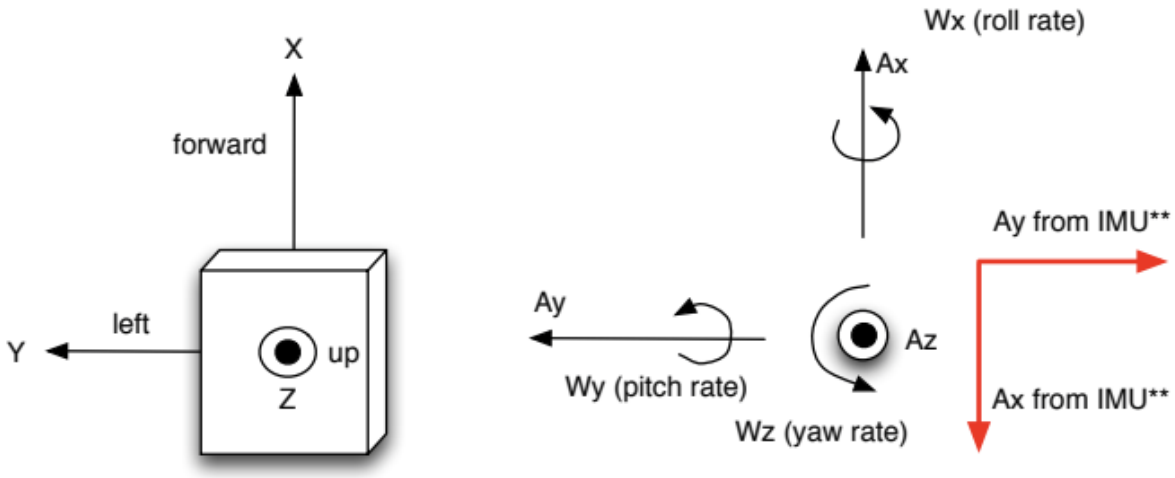
This problem is important on multiple levels. Anything in life that moves, does so in a 3-D space, meaning that if we can retrieve data from its movement in this 3-D space, we can track it and estimate its movement. 3-D orientation tracking is implemented in numerous systems such as robotics, airplanes, space shuttles, virtual reality and etc. Any system that moves in a 3-D field uses some sort of 3-D orientation tracking. In this case we used data from an IMU in order to track its movement in space. But this can be applied to any system that has movement in an X, Y and Z plane that can be tracked. In our system we used an Unscented Kalman Filter (UKF), but it can be used with a plain Kalman Filter, Extended Kalman Filter (EKF), Hybrid Kalman Filter or various other filters. With many options on how to do 3-D orientation tracking and many applications it can be applied to, it's no wonder this is such an important topic in robotics and various other fields.

Problem Formulation:

In our project, we are using an ArdulMU+V2 Inertial Measurement Unit. From this IMU we are given an IMU packet which contains acceleration along the various axes (Ax, Ay, Az), rotation rates along the various axes (Wx, Wy, Wz), and the corresponding time stamp of each reading (Ts). This corresponds to the raw 10 bit ADC values. We are also given a Vicon packet which contains a rotation matrix for each reading. This will be used to retrieve over 5,000 readings, each with corresponding values of Ax, Ay, Az, Wx, Wy, Wz and Ts. Using these values we need to track how this IMU is moving over time. This can be done by applying an Unscented Kalman Filter to each time step, calculating the necessary values, and then using those to determine the updated mean and covariance to be used in the next iteration of the UKF. Once we have these values over the 5,000+ readings we can then graph them and compare them against the Vicon packet.

Technical Approach:

Our IMU data orientation is shown here –



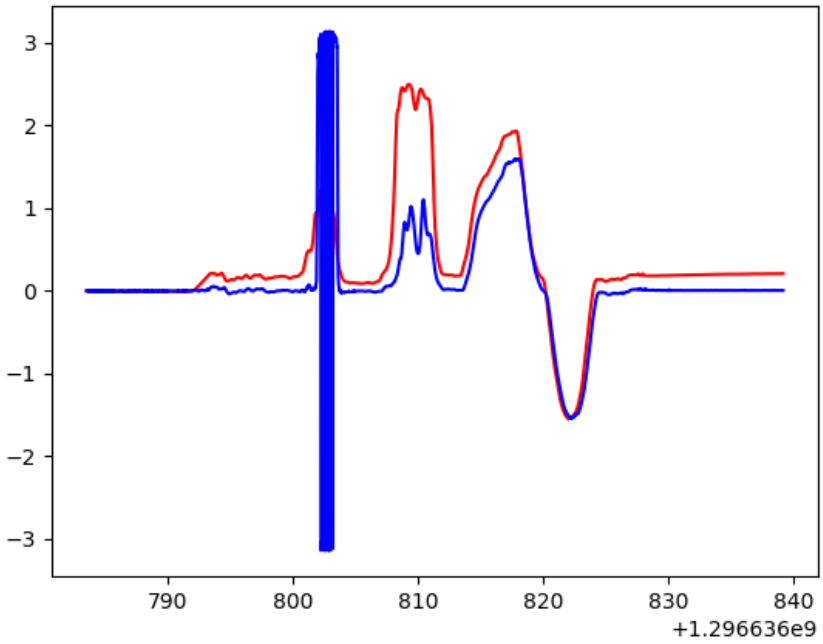
But first the IMU data needs to be calibrated using the functions –

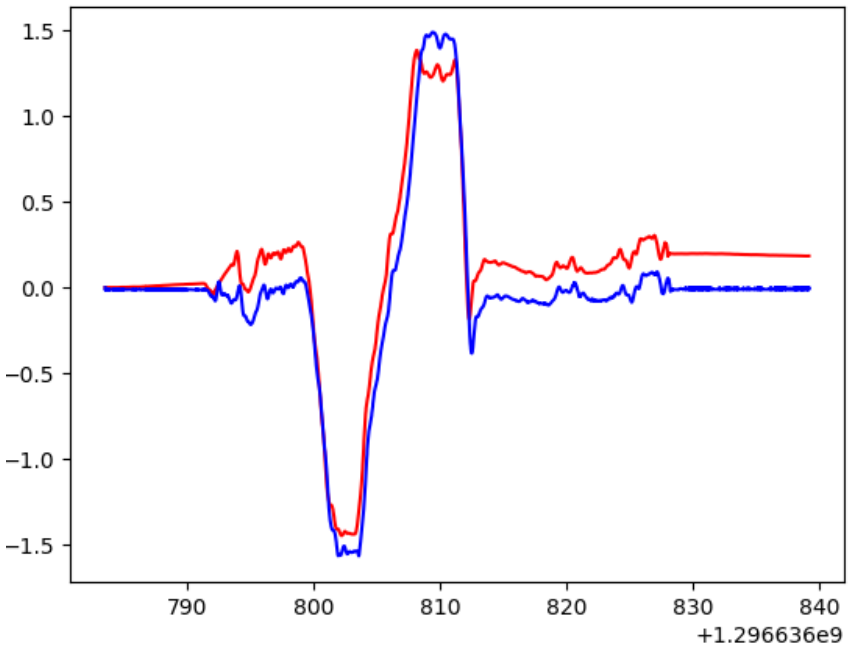
$$value = (raw - bias) * scale\_factor$$

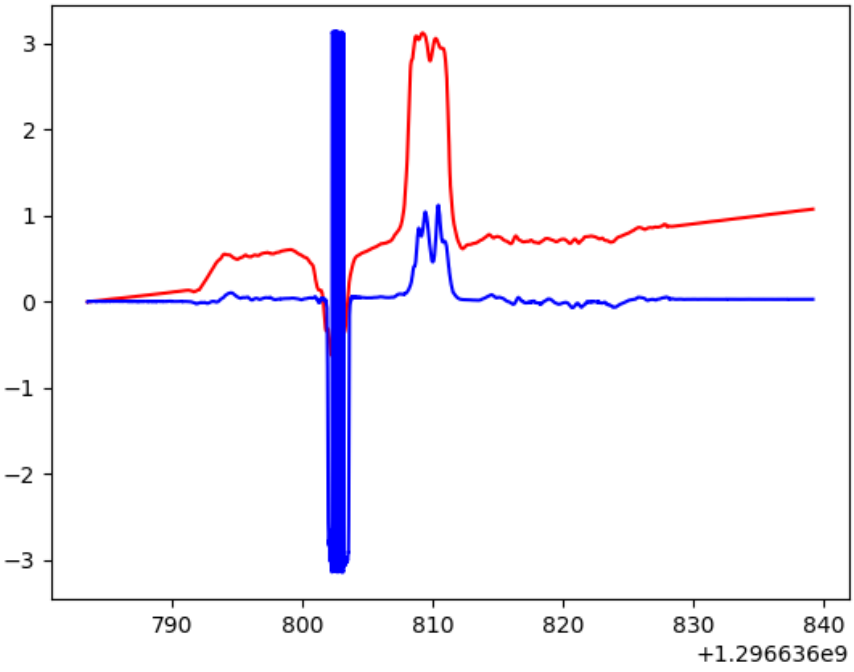
$$scale\_factor = Vref / 1023 / sensitivity$$

Where the scale factor for acceleration is 300 and the scale factor for rotation is 3.3. We use the first 100 time stamps as our bias for Ax, Ay, Az, Wx, Wy and Wz. Once we do this all of our data will be calibrated to the ground truth from the Vicon motion capture system.

We can convert this data into Euler angles and then plot this against the Vicon rotation matrices to verify that our IMU data was calibrated properly. In the graphs below red is the IMU data and blue is the Vicon data. The first graph is roll, second is pitch and third is yaw.







Now that the data was calibrated properly, we also need to implement quaternion functions to be used for the UKF -

Quaternion Properties

Addition	$q + p = [q_s + p_s,\ \mathbf{q}_v + \mathbf{p}_v]$
Multiplication	$q \circ p = \left[q_s p_s - \mathbf{q}_v^T \mathbf{p}_v,\ q_s \mathbf{p}_v + p_s \mathbf{q}_v + \mathbf{q}_v \times \mathbf{p}_v\right]$
Conjugate	$\bar{q} = [q_s,\ -\mathbf{q}_v]$
Norm	$\lvert q\rvert := \sqrt{q_s^2 + \mathbf{q}_v^T \mathbf{q}_v} \quad \lvert q \circ p\rvert = \lvert q\rvert\lvert p\rvert$
Inverse	$q^{-1} = \frac{\bar{q}}{\lvert q\rvert^2}$
Rotation	$[0,\ \mathbf{x}'] = q \circ [0,\ \mathbf{x}] \circ q^{-1} = [0,\ R(q)\mathbf{x}]$
Rot. Velocity	$\dot{q} = \frac{1}{2}[0,\ \omega] \circ q = \frac{1}{2} E(q)^T \omega = \frac{1}{2} q \circ [0,\ \omega_B] = \frac{1}{2} G(q)^T \omega_B$
Exp	$\exp(q) := e^{q_s} \left[\cos \lVert\mathbf{q}_v\rVert,\ \frac{\mathbf{q}_v}{\lVert\mathbf{q}_v\rVert} \sin \lVert\mathbf{q}_v\rVert\right]$
Log	$\log(q) := \left[\log \lvert q\rvert,\ \frac{\mathbf{q}_v}{\lVert\mathbf{q}_v\rVert} \arccos \frac{q_s}{\lvert q\rvert}\right]$

As well as a quaternion averaging function –

Quaternion Averaging

- Given a collection of quaternions $\{q_i\}_{i=1}^n$ with associated weights $\{\alpha_i\}_{i=1}^n$, we can obtain a weighted average quaternion $\bar{q}$, which is not unique and depends on the initialization point as follows.

Algorithm 1 Quaternion average

Input: $\{q_i\}_{i=1}^n$, $\{\alpha_i\}_{i=1}^n$, initial guess $\bar{q}_0$
for $t = 0, \ldots, T$ **do**
  $q_i^e = \left[q_{s,i}^e,\ \mathbf{q}_{v,i}^e\right] = \bar{q}_t^{-1} \circ q_i$
  $[0,\ \mathbf{e}_{v,i}] = 2 \log(q_i^e)$ ▷ Error rot. vector from quaternion
  $\mathbf{e}_{v,i} = (-\pi + \mathbf{mod}(\lVert\mathbf{e}_{v,i}\rVert + \pi, 2\pi)) \frac{\mathbf{e}_{v,i}}{\lVert\mathbf{e}_{v,i}\rVert}$ ▷ Restrict angles to $[-\pi, \pi)$
  $\mathbf{e}_v = \sum_{i=1}^n \alpha_i \mathbf{e}_{v,i}$
  $\bar{q}_{t+1} = \bar{q}_t \circ \exp\left(\left[0,\ \frac{\mathbf{e}_v}{2}\right]\right)$ ▷ Error rot. vector to quaternion
  if $\lVert\mathbf{e}_v\rVert < \epsilon$ **then return** $\bar{q}_{t+1}$

Once we have the data calibrated and the quaternion functions setup we can go about creating our UKF. We follow the outline given by Edgar Kraft in the paper “A Quaternion-Based Unscented Kalman Filter for Orientation Tracking” to create our UKF. The main difference between this and a normal UKF is this example uses quaternions to do all of the calculations. In my project I am using 6 sigma points to do the calculations, but this can be extended to more sigma points for better accuracy.

Prediction Step –

$$E_t^{(i)} = columns(\pm\sqrt{n(P_{t|t} + Q)})$$

$$q_{t+1|t}^{(i)} = q_{t|t} o exp([0, \frac{1}{2}E^{(i)}]) o exp([0, \frac{1}{2}\omega_t \triangle t])$$

$$q_{t+1|t}^{mean} = avgquat(q_{t+1|t}^{(i)}, \alpha^{(i)})$$

$$e^{(i)} = q_{t+1|t}^{(i)} o (q_{t+1|t}^{mean})^{-1}$$

$$P_{t+1|t} = \frac{1}{6}\sum_{1}^{6} e_{t+1|t}^{(i)} (e_{t+1|t}^{(i)})^T$$

In the prediction step we use the take the calculate all of the sigma points (q_{t+1|t} ^{(i)}) which we then use in the avgquat function to take the average of all 6 sigma points. We also have a 0 sigma point but in this case the weight is 0 so it is not used in any of these calculations. Once we have the average of the 6 sigma points then we can calculate e^{(i)} which represents translating the quaternion to a rotation vector. Which can then be used to calculate P_{t+1|t}, which represents the covariance.

Update Step –

$$(0, Z_{t+1}^{(i)}) = \bar{q}_{t+1|t}^{(i)} o [0, 0, 0, 1] o q_{t+1|t}^{(i)}$$

$$Z_{t+1}^{mean} = 2Z_{t+1}^{0} + \frac{1}{6}\sum_{1}^{6} Z_{t+1}^{(i)}$$

$$P_{ZZ} = 2(Z_{t+1}^{0} - Z_{t+1}^{mean})(Z_{t+1}^{0} - Z_{t+1}^{mean})^T + \frac{1}{6}\sum_{1}^{6}(Z_{t+1}^{(i)} - Z_{t+1}^{mean})(Z_{t+1}^{(i)} - Z_{t+1}^{mean})^T$$

$$P_{VV} = P_{ZZ} + P_R$$

$$P_{XZ} = 2e_t^0(Z_{t+1}^{0} - Z_{t+1}^{mean}) + \frac{1}{6}\sum_{1}^{6} e_t^{(i)}(Z_{t+1}^{(i)} - Z_{t+1}^{mean})$$

$$K_{t+1} = P_{XZ} + P_{VV}^{-1}$$

Taking the sigma points that we calculated before, we calculate Z_{t+1}^{(i)} which we use to calculate the mean of the Z matrix (Z_{t+1}^{mean}). Once we have this we can then calculate P_{ZZ} which is the uncertainty of the predicted measurements. Using this and adding on P_{R} which is the noise (in our case we used .001I matrix) we get P_{VV}. Now to calculate P_{XZ} which is the cross correlation matrix we use a similar method as P_{ZZ}, but we now multiply in the rotation vectors as well. Finally we can get K_{t+1} which represents the Kalman Gain.

Lastly –

$$q_{t+1|t+1} = exp([0, \frac{1}{2}K_{t+1}(Z - Z_{t+1}^{mean})])oq_{t+1|t}^{mean}$$

$$P_{t+1|t+1} = P_{t+1|t} - K_{t+1}P_{VV}(K_{t+1})^T$$

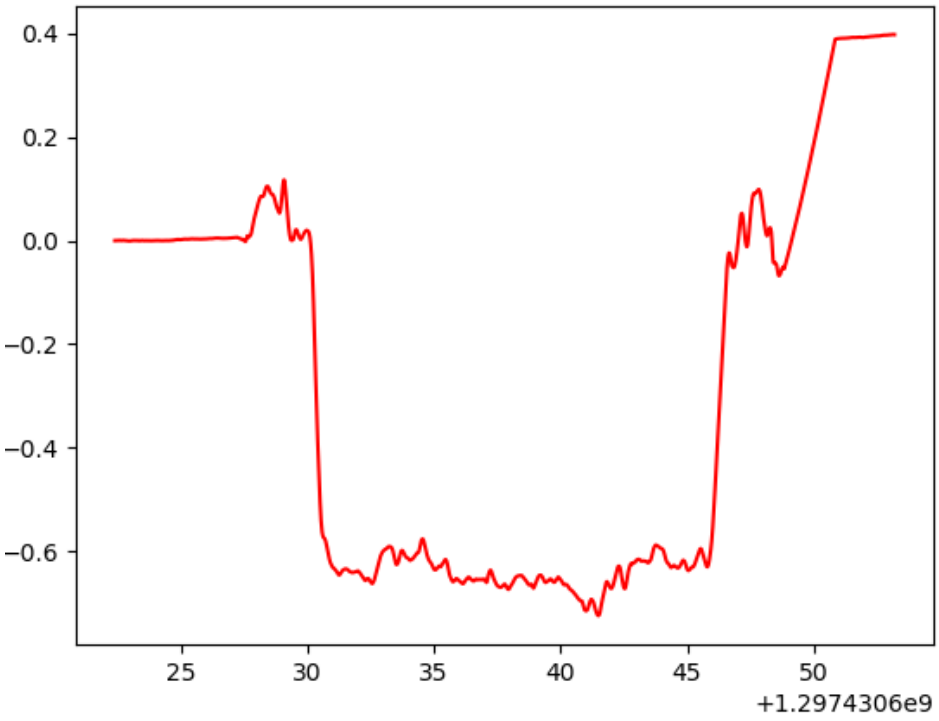
Now using what we have previously calculated we can get q_{t+1|t+1} which represents the updated quaternion.. Also we get P_{t+1|t+1} which represents the updated covariance. Both of these replace q_{t|t} and P_{t|t} for the next iteration of the loop.

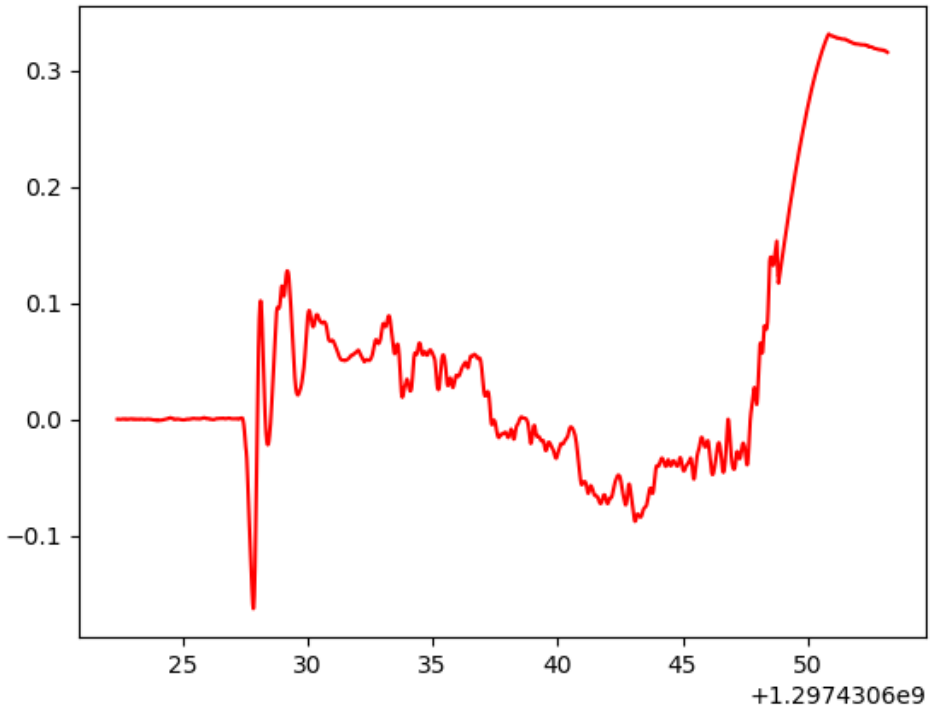
Once we run this once for each set of data we will then have a list of all of the quaternion values for each set of data. Using these quaternion values we can then graph these to determine the change in orientation, which should match what was given by the Vicon packet when running it on the trainset.

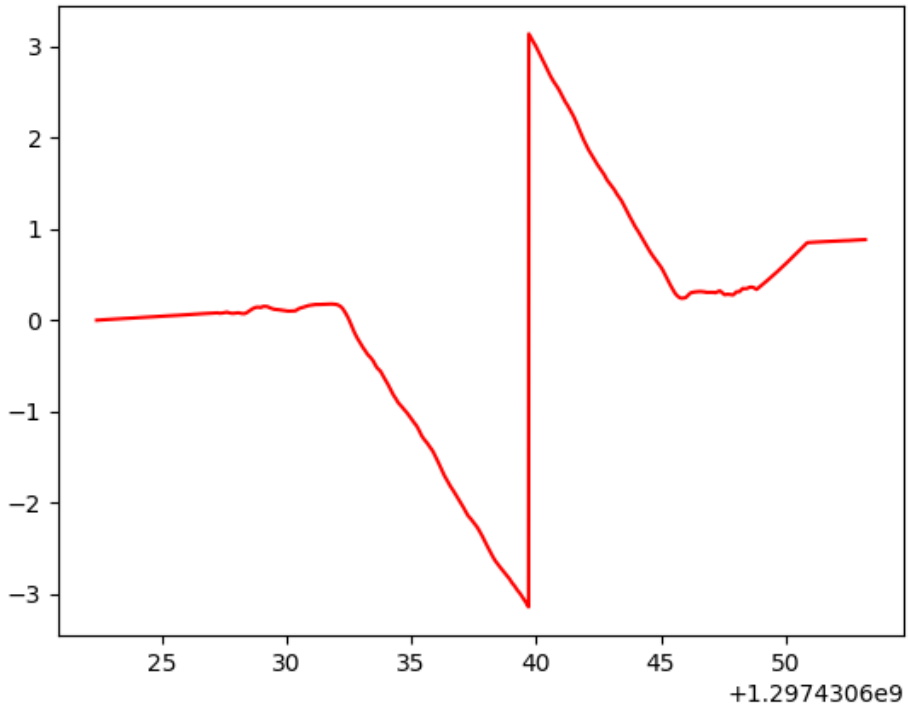
Lastly once we have this list of all of the quaternions corresponding to the movement of the IMU at each time step, we can generate a panorama image by taking each quaternion at each time step and rotating the corresponding camera image by that quaternion. Once we do this for each camera image we can stitch them together to get a panorama view.

Results:

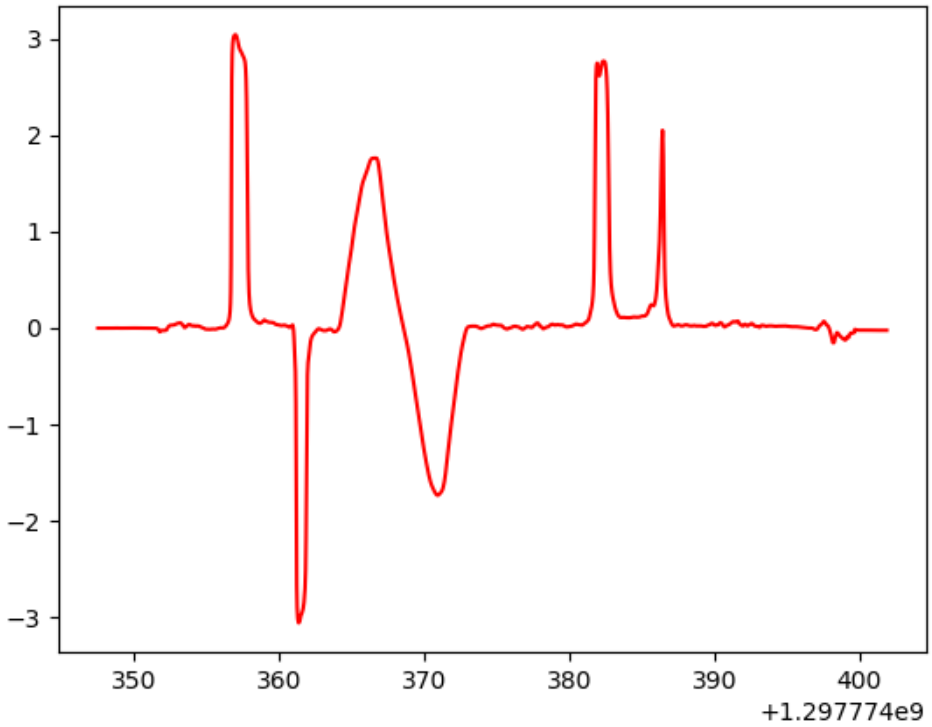
Trainset 10 (Roll, Pitch, Yaw) -

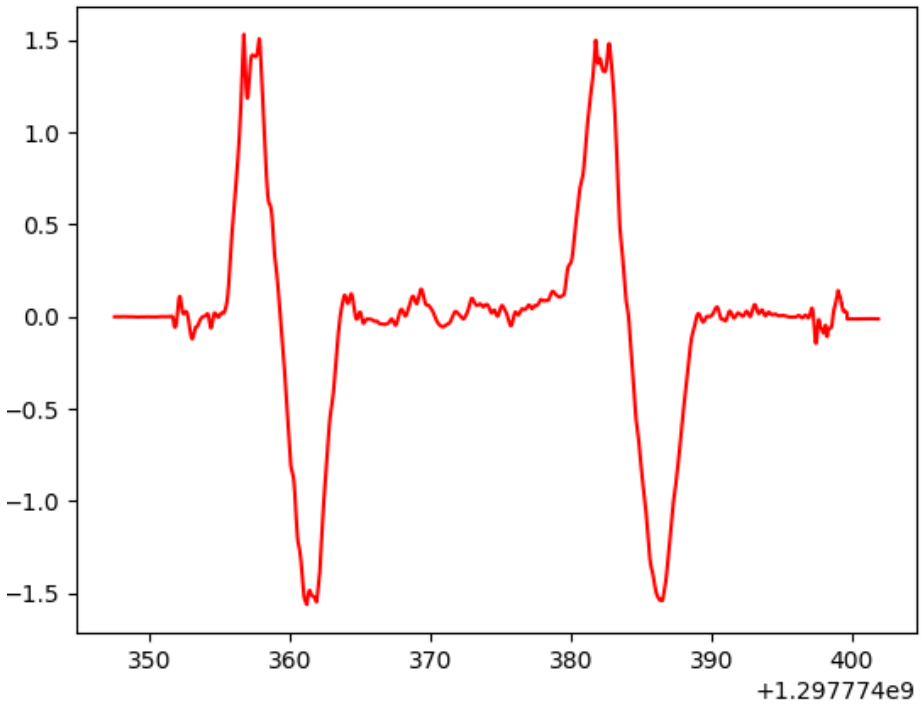


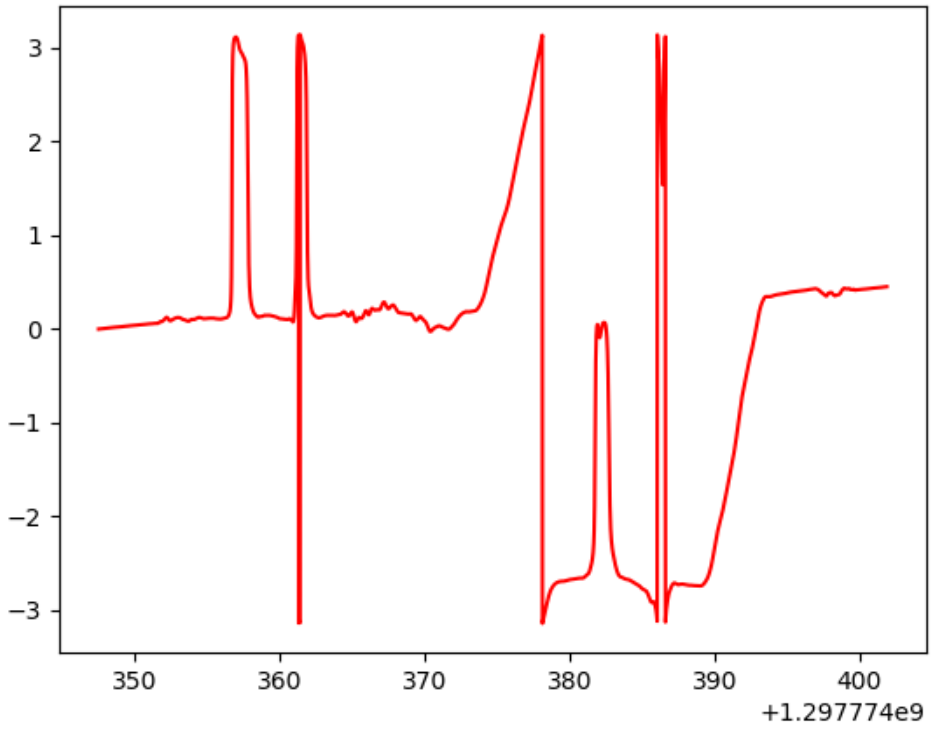




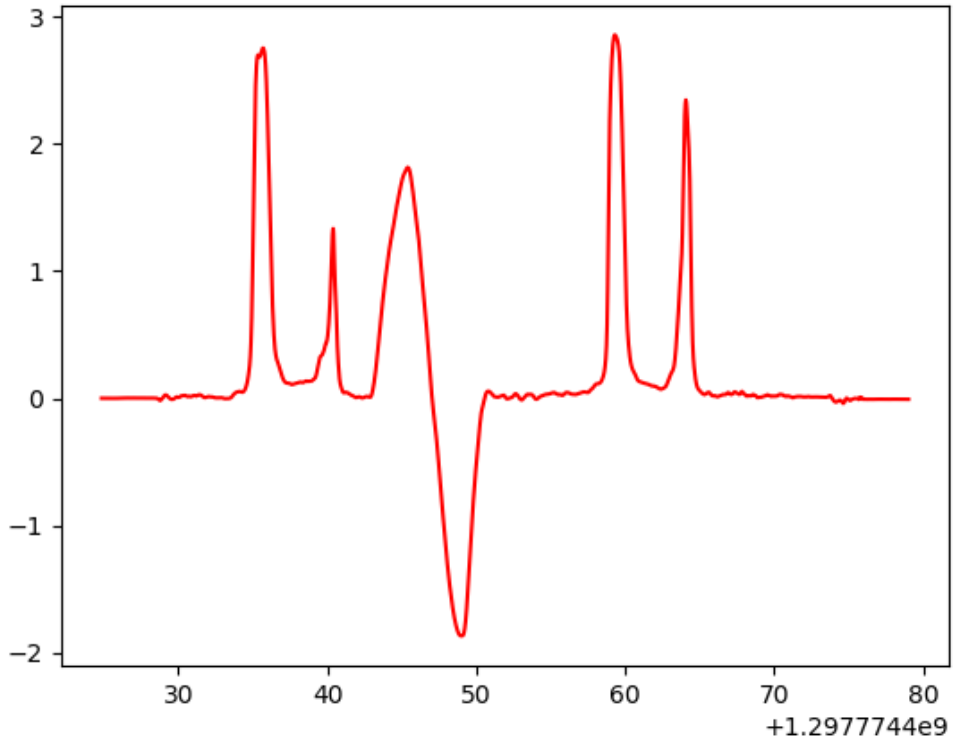
Trainset 11(Roll, Pitch, Yaw) –

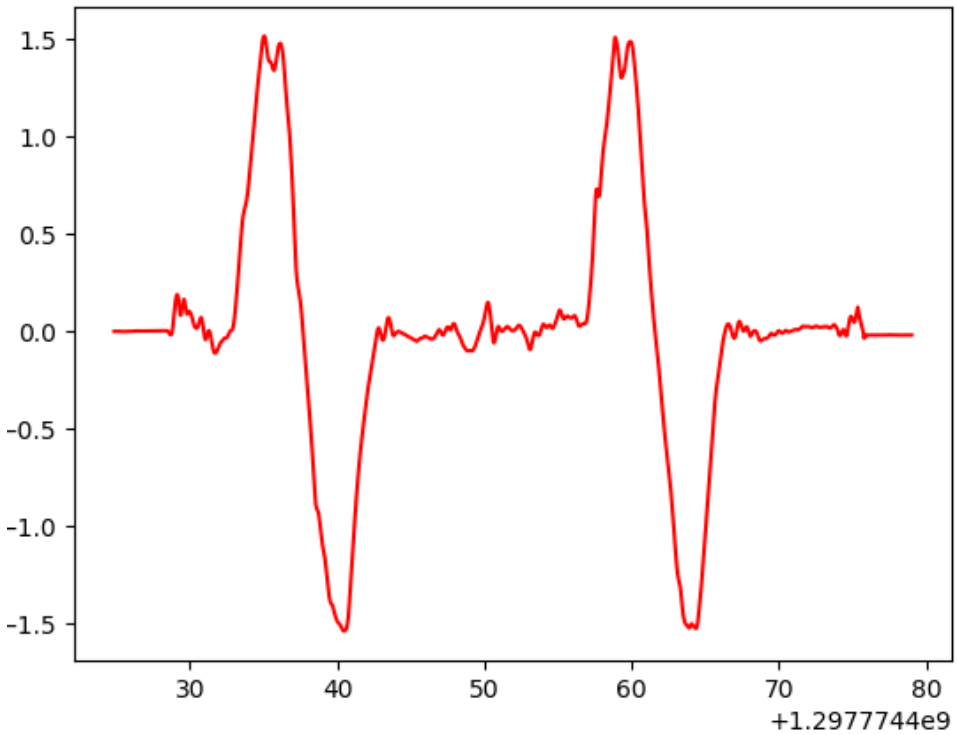


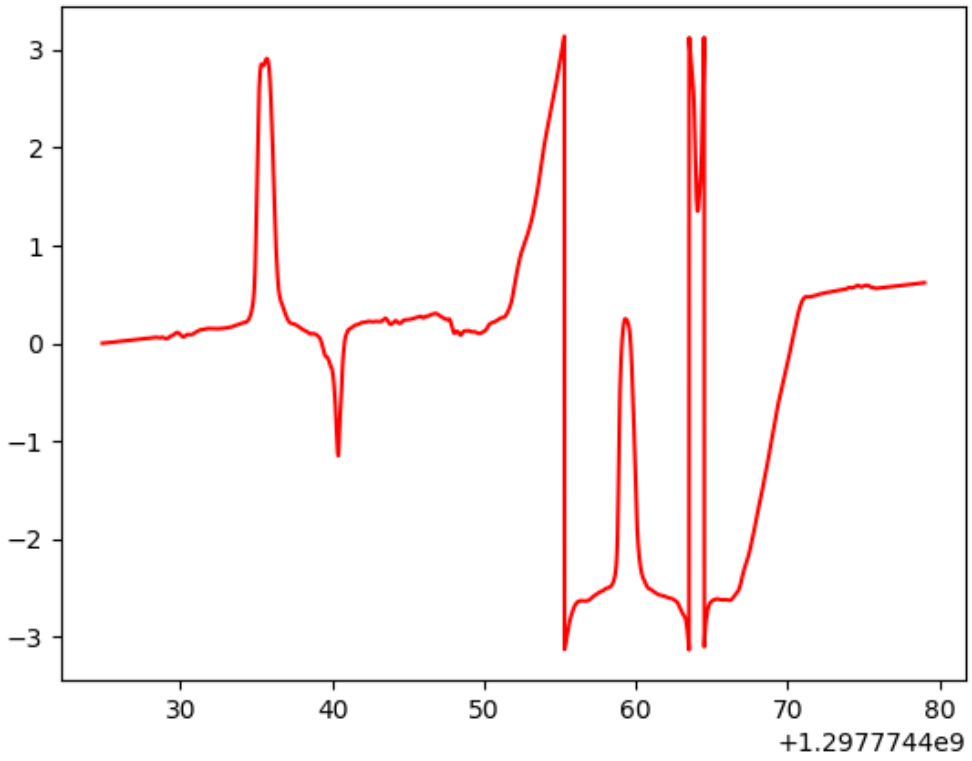

1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
350
360
370
380
390
400
+1.297774e9


3
2
1
0
−1
−2
−3
350
360
370
380
390
400
+1.297774e9

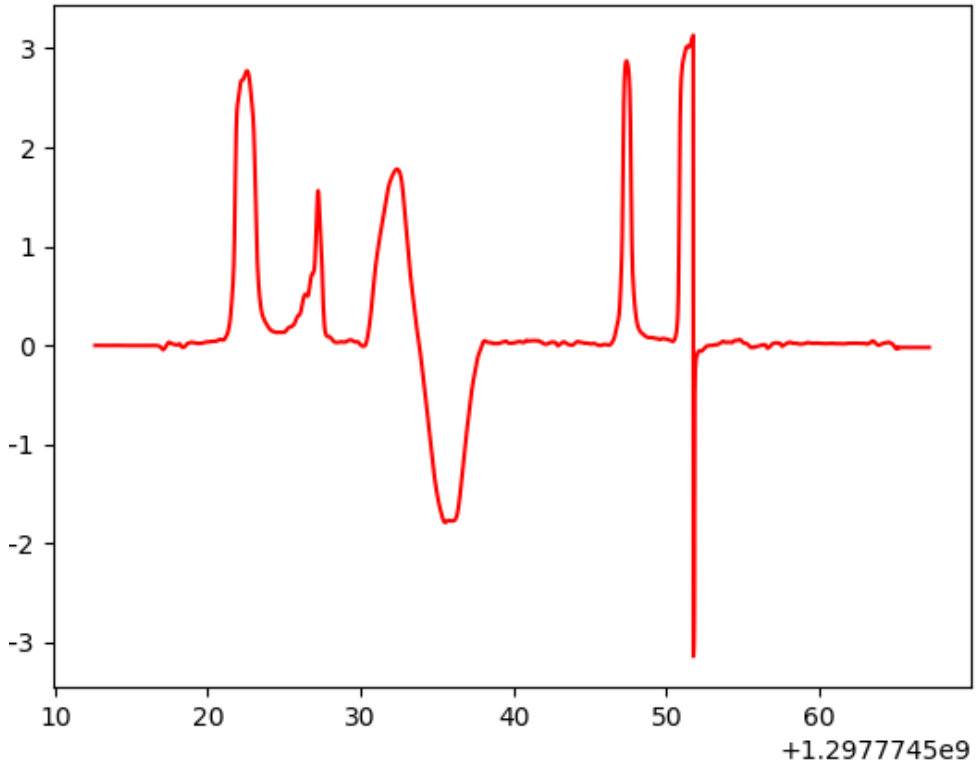
Trainset 12 (Roll, Pitch, Yaw) –

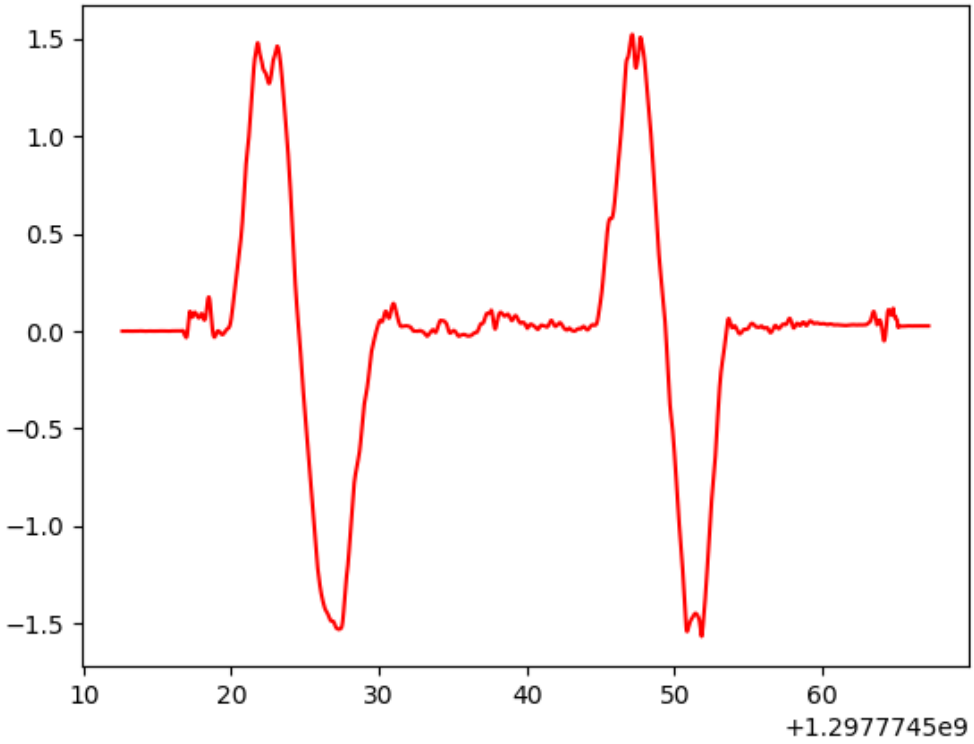


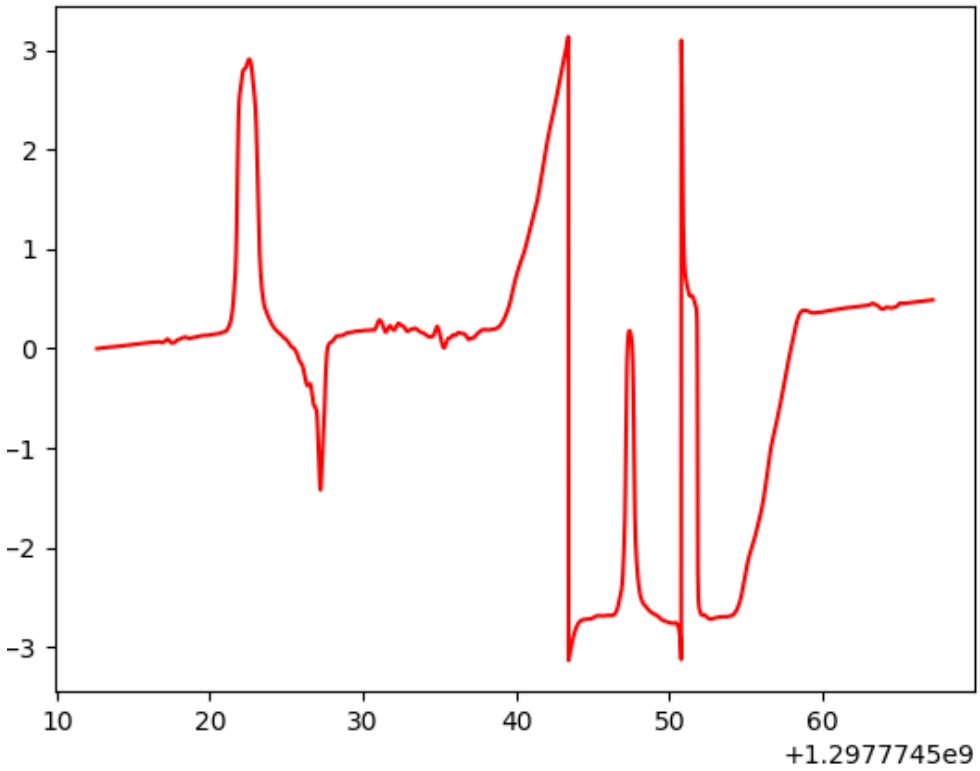




Trainset 13 (Roll, Pitch, Yaw) –




1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
10
20
30
40
50
60
+1.2977745e9


3
2
1
0
−1
−2
−3
10
20
30
40
50
60
+1.2977745e9

Unfortunately my UKF had some error that I could not find, so they did not always turn out accurately. They did for the most part on certain sets, but others it performed poorly on.

I did not get the panorama portion working, but to translate what I had done to the panorama I would have taken these graphs of data for roll, pitch and yaw for each trainset and then rotated the camera picture by the corresponding angle rotations.