ECE276A HW2 Report
Samuel Sunarjo
A12340230

Introduction

- Our first problem - track the 3-D orientation of a rotating body using IMU - is important because knowing the orientation of a robot itself is often necessary for a robot to make decisions or actions. Tracking orientation is the essence of autonomous robot that is designed to have mobility and conduct any dynamic movement in space. Noise filtering is almost necessary for any kind of system.
    - My approach to tracking 3-D orientation is to assume that the state variables can be approximated by Gaussian distributions and implement an unscented Kalman filter to filter out the noise from the IMU measurements and construct a more accurate estimate of the orientation and angular velocity of the robot
- Our second problem - panorama by stitching camera images - is important because there are often many different sensors and state variables on the robot. Camera and IMU are very common component on a robot, as they are inexpensive and informative. Creating a panorama with the camera and IMU illustrates the concept of sensor fusion in a system.
    - My approach to the panorama is to use the IMU and Vicon measurements to transform the camera images to the world frame and project the images onto the panorama.

Problem Formulation

Tracking the 3-D orientation

- Given a set of measurements from the IMU with the time each data point is sampled at, the goal for this part of the project is to construct a more accurate estimates of the orientation of the robot in the body frame. The IMU provides a set of measurements $D = \{A_i, W_i\}^N$ where $A_i \in R^3$ is the acceleration, $W_i \in R^3$ is the angular velocity, and N is the number of samples. The Vicon data is also available, which contains the data $V = \{R_i\}^N, \; where \; R_i \in R^{3x3}$ is the rotation matrix defined as the transformation from body frame to the world frame, and N is the number of samples.

Panorama

- Given a set of camera images with the time each image is sampled at, the goal for this part of the project is to construct a panorama in the world frame based on the orientation estimate of the robot body frame. The images are taken in the robot body frame. The images are formatted as $M = \{P_v, P_h, C\}^N$, where $P_v \in R^1$ is the pixel position on the vertical axis and $P_h \in R^1$ is the horizontal axis, and $C \in R^3$ corresponds to the RGB channels of the pixel, and N is the number of samples. The IMU data is also given in the same format as described above.

Technical Approach

Tracking the 3-D orientation

- First, the imu data needs to be **preprocessed**. The data are in digital bits and needs to be converted to physical units through a bias from the first 100 samples and a scaling factor, which is composed of the analog voltage reference V_{ref}, degree to radian conversion, and sensitivity scaling. The IMU was at rest for the first 100 samples and the average value of these samples is the bias for offset calibration of the sensor. The order of the angular velocity data are in Z, X, Y instead of X, Y, Z, which needs to be re-ordered.

- The unscented Kalman filter is implemented rather than the regular Kalman filter because of the assumption of nonlinear process and measurement model.
- The Kalman filter **state vector** $x_k = \{q_k, \omega_k\} \in R^7$, where $q \in R^4$ is a quaternion that represent the current orientation of the robot and $\omega \in R^3$ represents the angular velocity of the robot for the k-th state. The **process model** A for the state vector is $x_{k+1} = A(x_{k,} w_k)$, where $w_k \in R^6$ is the six-dimensional noise vector that is applied before the process model. The first three components of w_k ,denoted as w_q or q_w, affects the orientation and the last three components, denoted as w_ω, affects the angular velocity. The current state quaternion is rotated by the current state angular velocity, and the angular velocity is assumed to be constant. The process model entails the following:
  - $q_{k+1} = q_k q_w q_\Delta$
  - $\alpha_\Delta = |\omega_k| * \Delta t$, $\zeta_\Delta = \frac{\omega_k}{|\omega_k|}$
  - $q_\Delta = [cos(\frac{\alpha_\Delta}{2}),\ \zeta_\Delta sin(\frac{\alpha_\Delta}{2})]$
  - $\omega_{k+1} = \omega_k + w_\omega$
- The **unscented Kalman filter** estimates the state vector space by a probability distribution, characterized by the mean $\hat{x}_k$ and covariance $P_k \in R^{6x6}$ for the k-th state. The process and measurement model will transform the distribution. Unlike the regular Kalman filter, a set of **sigma points** X_i are calculated based on the mean and covariance of the current state through Cholesky Decomposition of the sum of state vector covariance P_k and process noise covariance $Q \in R^{6x6}$ where $Q = 0.0001 * I_{3x3}$:
  - $S = \sqrt{P_k + Q}$
  - $W_i = columns(\pm\sqrt{2n * (P_k + Q)}) = [q_W, \omega_W]$
  - $X_i = \hat{x}_k + W_i = [\hat{q}_k \cdot q_W,\ \hat{\omega}_k + \omega_W]$
- These sigma points X_i are then propagated through the process model A, and the distribution for next state vector is computed as the mean and covariance of the transformed sigma points Y_i, which is often referred as the **a priori estimate**. The quaternion mean needs to be computed through quaternion averaging and the angular velocity mean can be computed through the normal barycentric mean. Note that the noise vector is not in this equation because it was applied already during the construction step of sigma points X_i. The **weights** for the mean should be $0, \frac{1}{2n}, \frac{1}{2n}, ...$, and for the covariance $2, \frac{1}{2n}, \frac{1}{2n}, ...$, where $n = 6$ is the dimension of the covariance matrix.
  - $Y_i = A(X_i)$
  - $\hat{x}^-_{k+1} = mean(\{Y_i\})$
  - $P^-_{k+1} = covariance(\{Y_i\})$
- The set of sigma points Y_i in the a priori estimate will pass through the **measurement model** H to form the set Z_i, which contains the gravity measurement Z_g and angular velocity Z_ω in the robot body frame. The gravity Z_g is calculated through rotating the gravity g in the world frame by the state quaternion q_k, and there is no transformation for the angular velocity portion. $\hat{z}^-_{k+1}$ represents the measurement values we expect to measure.
  - $Z_i = H(Y_i)$
  - $z^-_{k+1} = mean(\{Z_i\})$
  - $P_{zz} = covariance(\{Z_i\})$
- Next, the **innovation** $v_{k+1} = z_{k+1} - z^-_{k+1}$ is computed, where z_{k+1} is the IMU measurement at time $k+1$. This essentially accounts for the difference between our estimated measurements and the actual

measurements. Its covariance is: $P_{vv} = P_{zz} + R$, where R is the measurement noise covariance and assumed to be $R = 0.0001 * I_{3x3}$.

- Lastly, the Kalman gain is computed $K_{k+1} = P_{xz} \cdot P_{vv}^{-1}$ for the update of our state mean and covariance estimate. P_{xz} is the cross correlation matrix between the state vector space and the measurement space, which can be calculated as the covariance between the set Y_i and Z_i . The update rule for the **a posteriori estimate** $\hat{x}_{k+1}$ and P_{k+1} is as the following:
  - $\hat{x}_{k+1} = \hat{x}^{-}_{k+1} + K_{k+1} v_{k+1}$
  - $P_{k+1} = P^{-}_{k+1} - K_k P_{vv} K_K{}^T$
- Then iterate from the beginning with these estimates.

Panorama

- Once we know the orientation of the robot, we can piece the camera images on the panorama based on the indicated orientation for each image. The task can be think of as finding the corresponding panorama pixel coordinate for each image and its pixels.
- The signs of these pixel coordinates are then inverted because the pixel axis and cartesian axis are entirely flipped. Half of the height and width are subtracted from these coordinates because I assumed the center of the image to be the origin.
- The image pixel coordinates are then converted to latitude and longitude by multiplying the coordinates by the ratio of the vertical and horizontal field of view, $45^\circ \; and \; 60^\circ$, and the image height and width, $240 \; and \; 320$, with radius being 1.
- Then these coordinates are converted to Cartesian coordinates.
- So far, these coordinates are in the robot frame. They are transformed to the world frame by multiplying the robot frame Cartesian coordinates by the rotation matrix.
- The Cartesian coordinates in the world frame are then converted back to latitude and longitude with radius of 1.
- The latitude and longitude are mapped to panorama pixel coordinates by multiplying by the ratio of the vertical and horizontal angle range on the panorama, $180^\circ \; and \; 360^\circ$, and the image height and width, $960 \; and \; 1920$. The panorama height and width are picked by the ratio between the panorama angle range and the original image field of view, $240 * \frac{180^\circ}{45^\circ} = 960, \; 320 * \frac{360^\circ}{60^\circ} = 1920$.
- Again, these coordinates are then inverted because of the sign difference between the cartesian axis and pixel axis. Finally, half of the panorama height and width is added to these panorama pixel coordinates to center the origin back to the top left corner.
- Now, the coordinates mapping is complete. The rest is just moving the RGB values from the image coordinates to their corresponding panorama coordinates as computed.

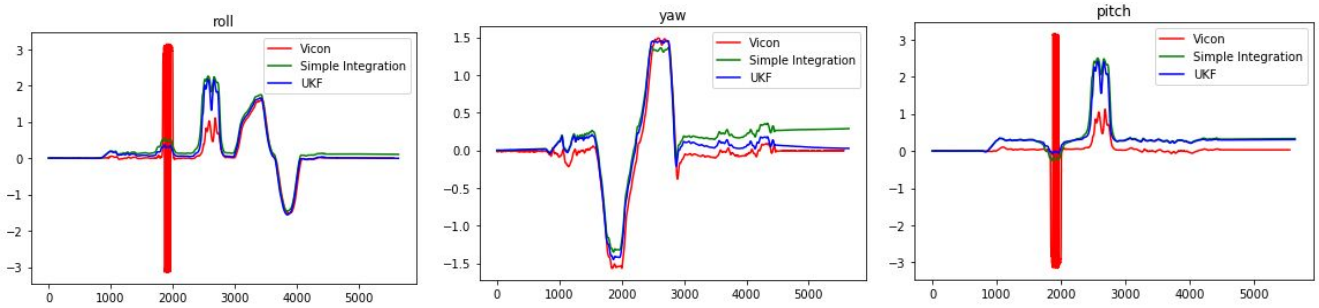
Results and Discussion

Tracking the 3-D orientation

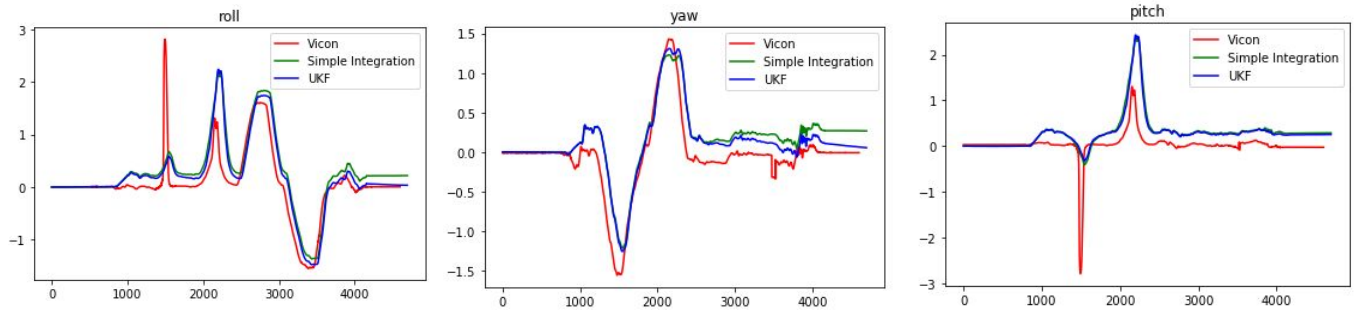
- In the following plots, I plotted the euler angles from three sources: the Vicon data, simple integration without filtering, and UKF output.
- In most cases,such as the data set 1 and 2, the UKF is performing slightly better, especially in terms of filtering out the drifting. However, it can be seen that the simple integration and UKF trajects pretty closely for the most part, meaning that our UKF is not outperforming the simple integration. The main reasons could be that our assumption of the variance being 0.0001 for the process and measurement

noise. Another factor could be the inaccuracy and incompleteness in our process and/or measurement model. Different weights for the mean and covariance might improve the UKF performance as well.

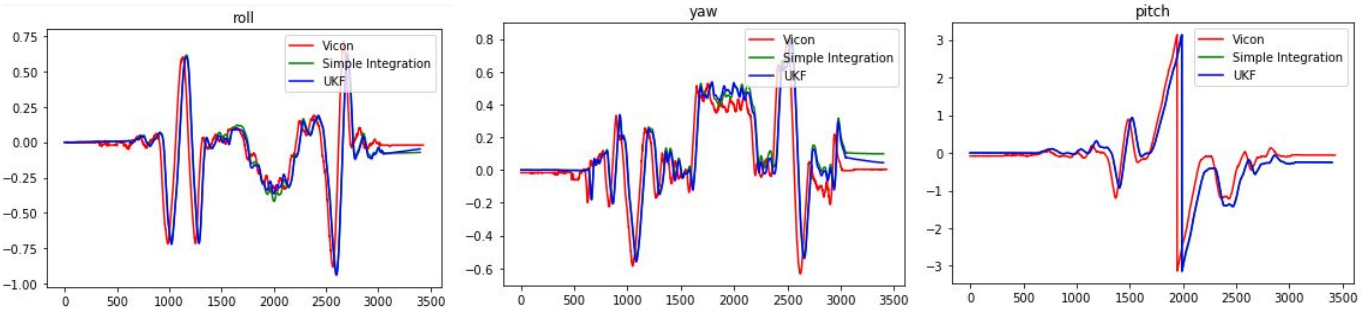
- **Training data set 1**



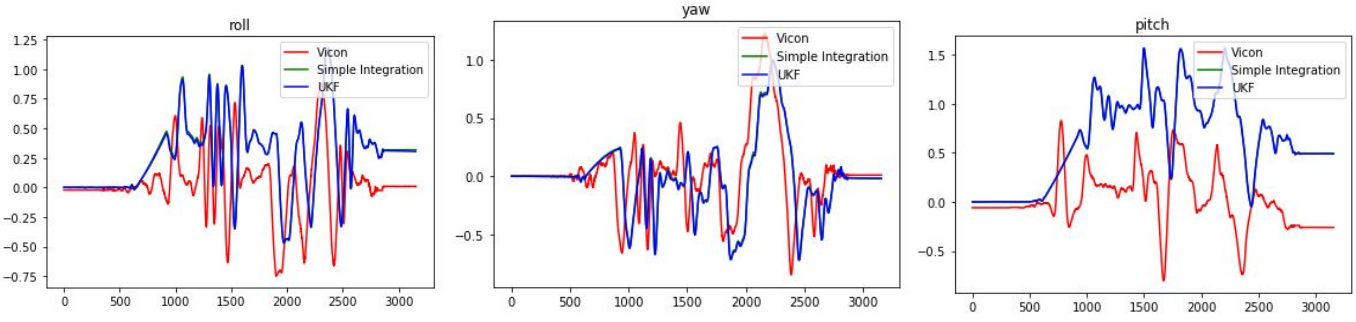
- **Training data set 2**



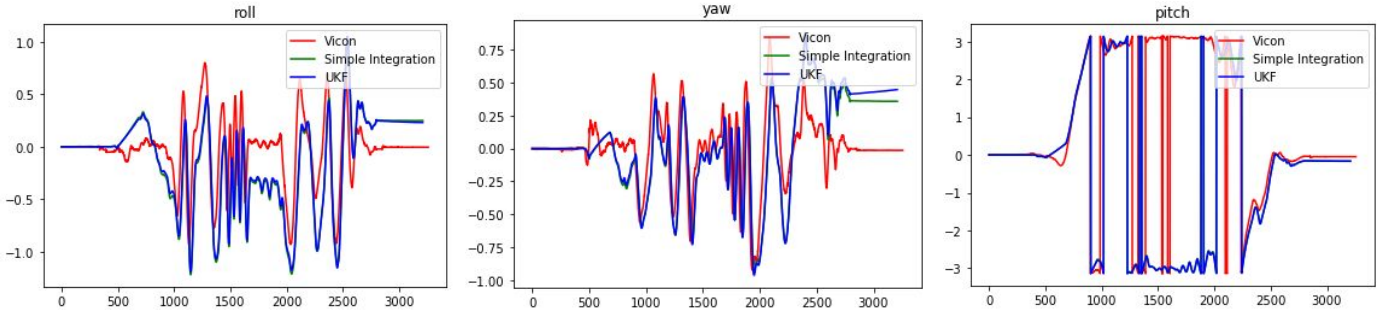
- **Training data set 3**



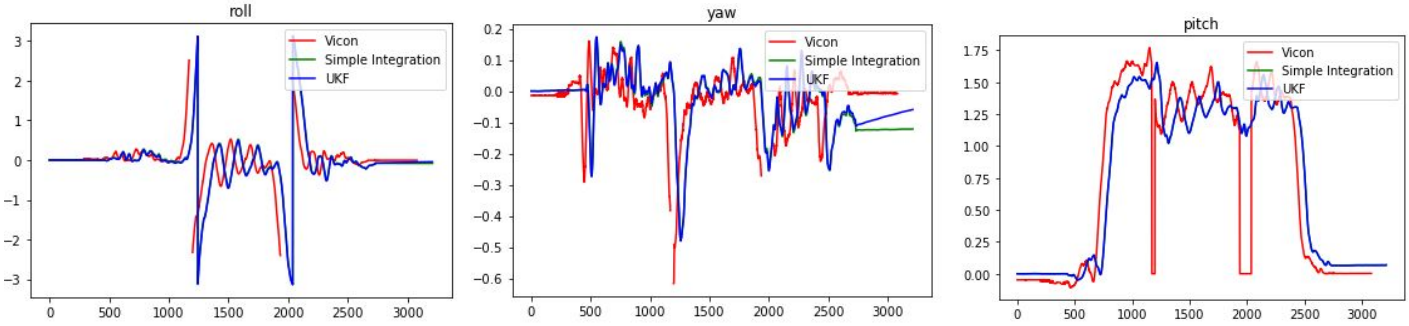
- **Training data set 4**



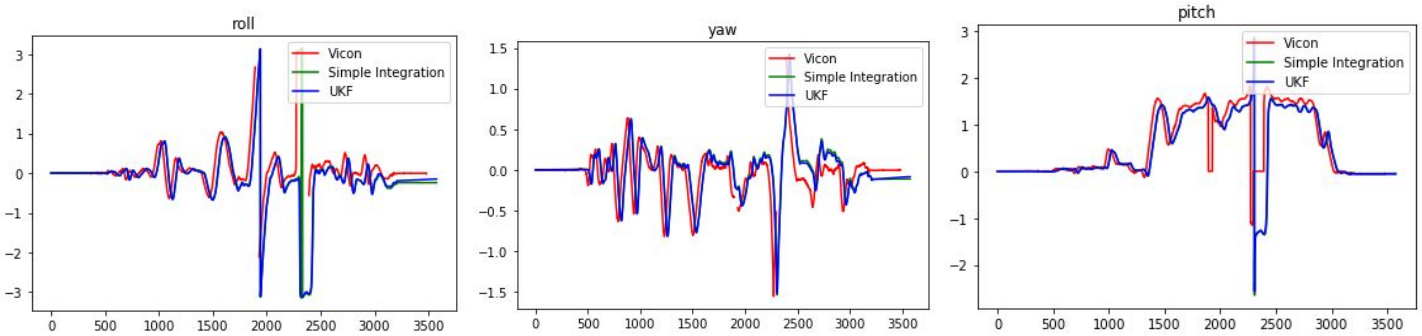
- **Training data set 5**



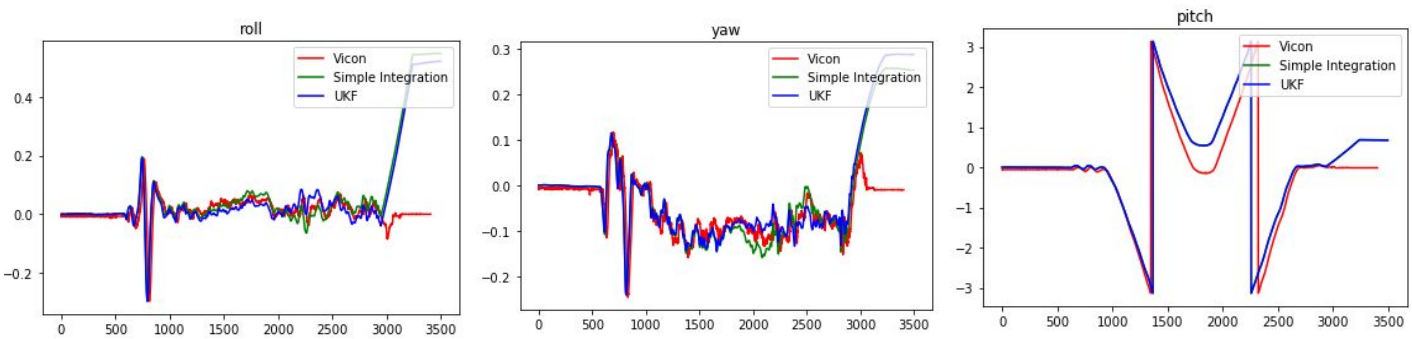
- **Training data set 6**



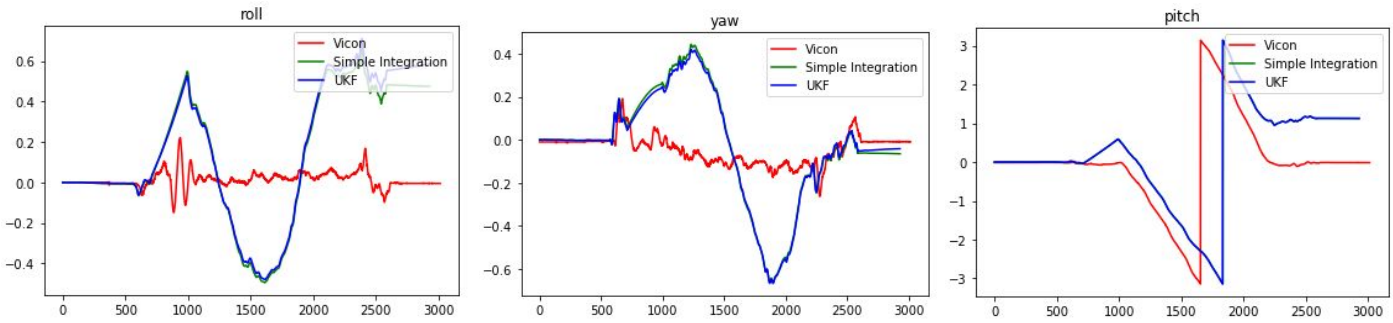
- **Training data set 7**



- **Training data set 8**

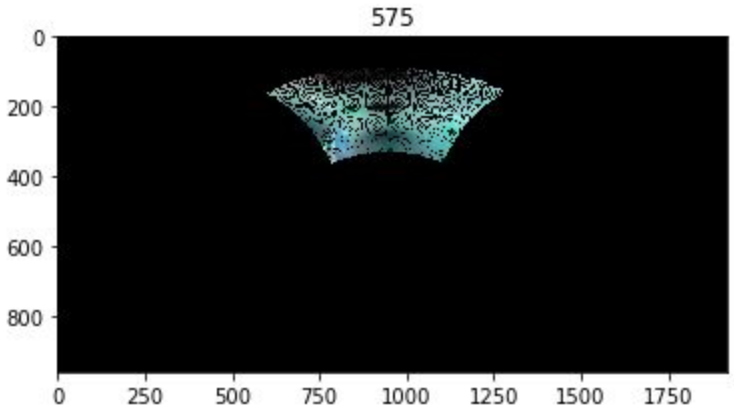


- **Training data set 9**



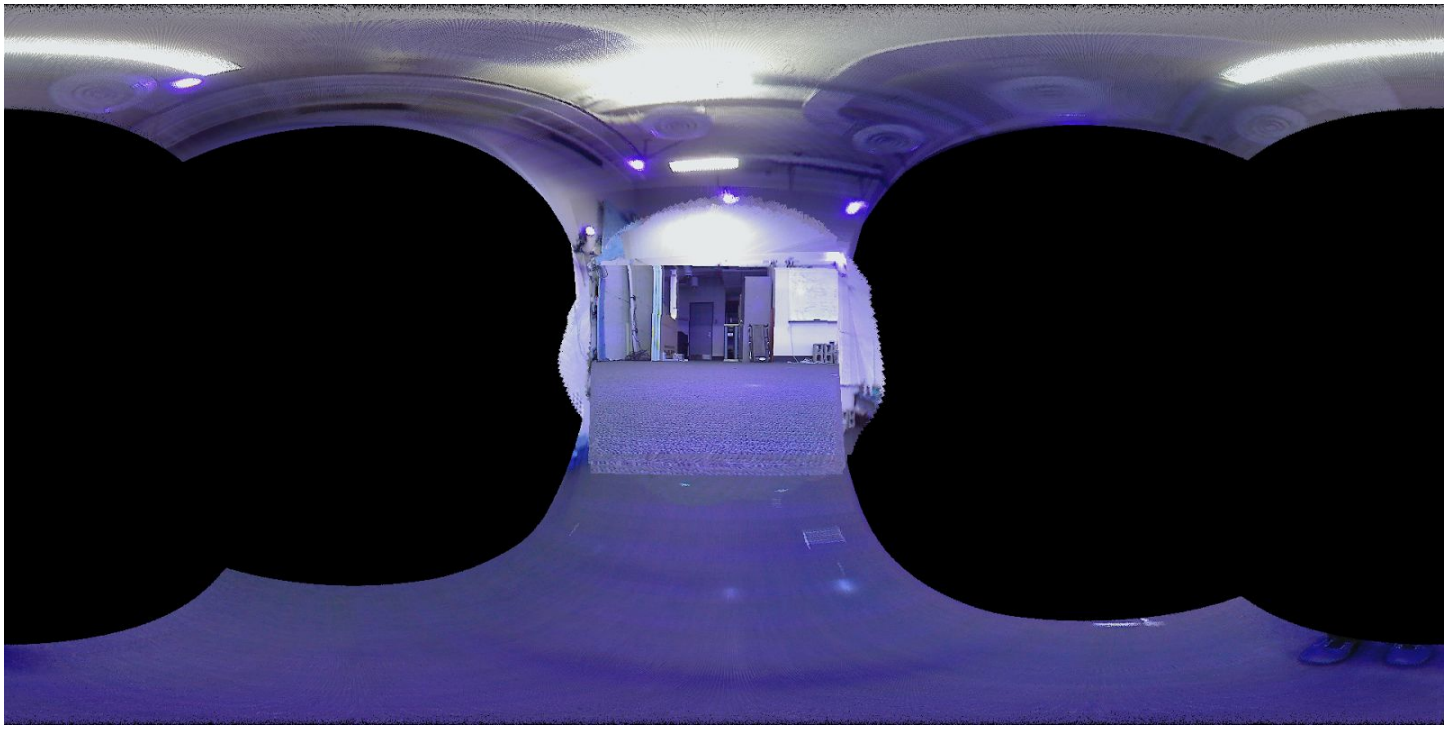
Panorama

- Overall, the panoramas look perceivable. The distortion at the top and bottom of the panorama is mainly caused by the imperfect projection from spherical to cylindrical and pixel coordinates. The distortion is particularly significant for images that orient upward or downward. We can see from the image below that the upward images are no longer in a square shape because of our assumption that the images are taken on a spherical surface with radius 1.

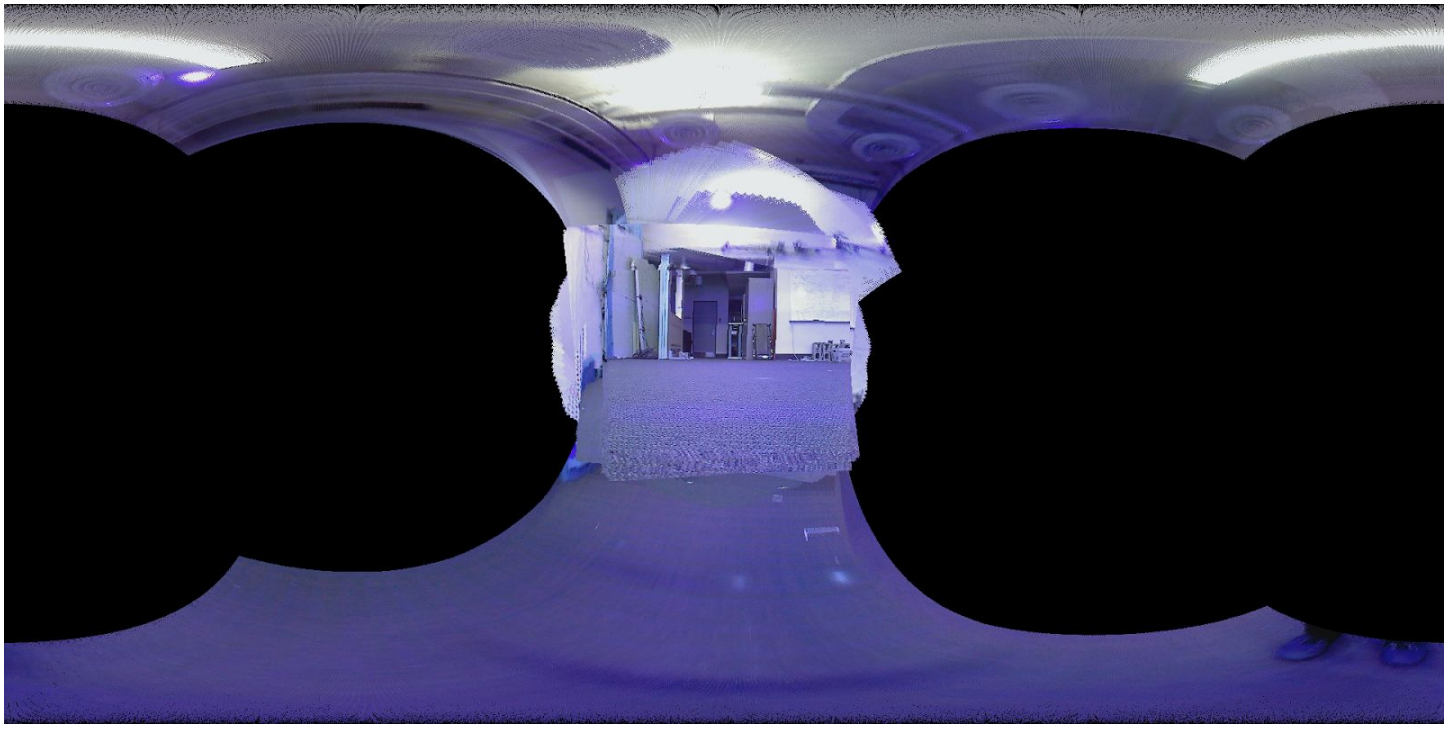


- The discontinuity in the center of the panorama is caused by the image dataset itself, where the camera is lifted up in the beginning and put down at the end, while the euler angles stay constant. Knowing the correct orientation of the camera is very significant in constructing the panorama. The training set panoramas looks relatively smoother and reasonable, as they are constructed based on the Vicon data. The testset orientations are estimated based on the UKF, and the resulting panoramas, such as testset 11-13, seem disconnected compared to the panorama constructed with accurate orientation.

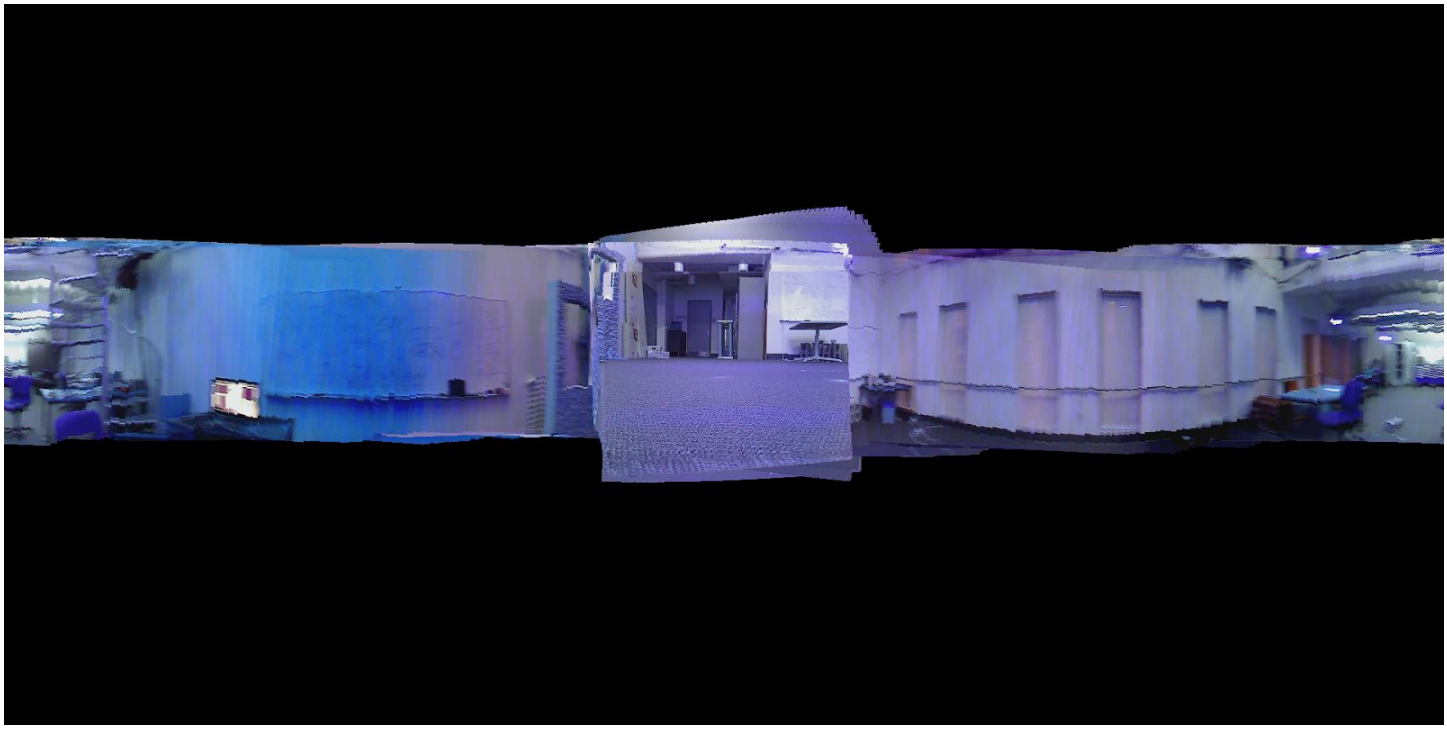
- **Training data set 1**



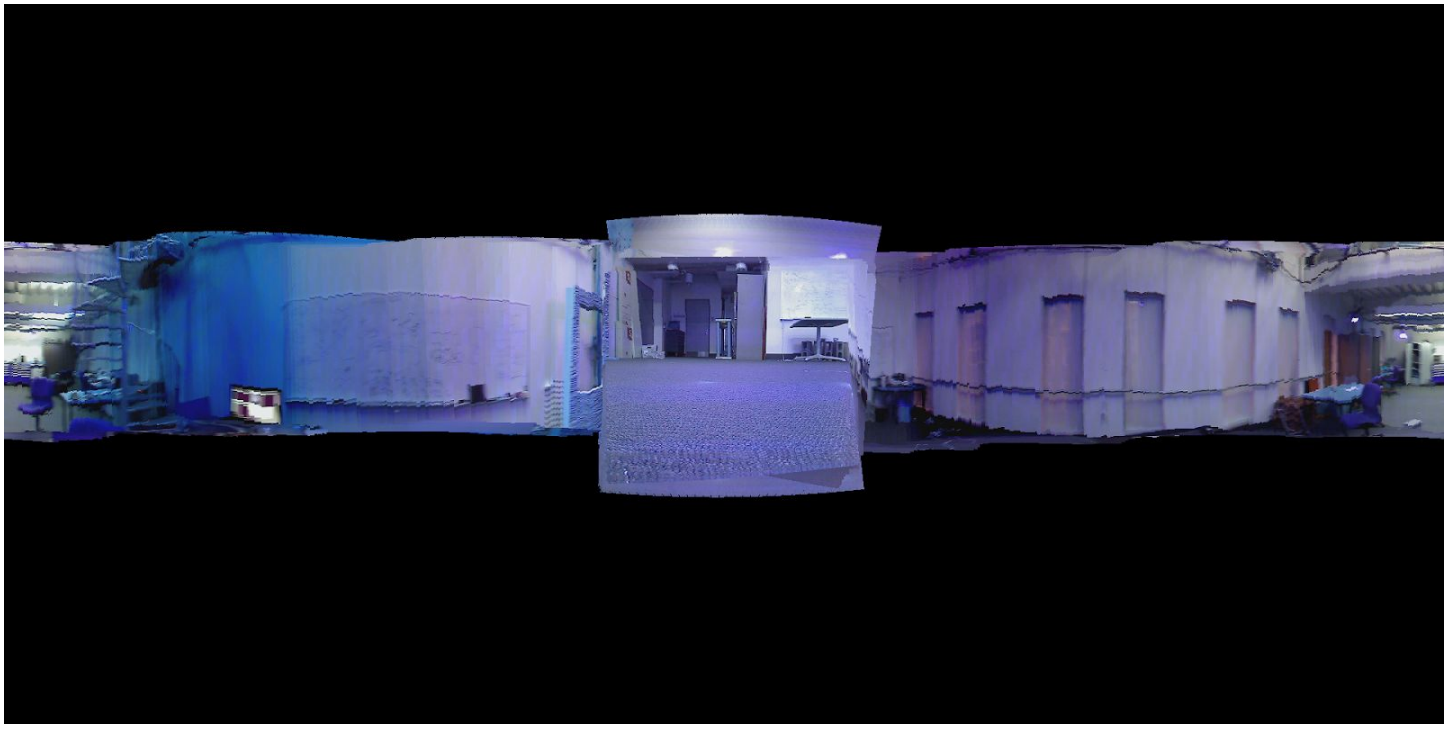
- **Training data set 2**



- **Training data set 8**

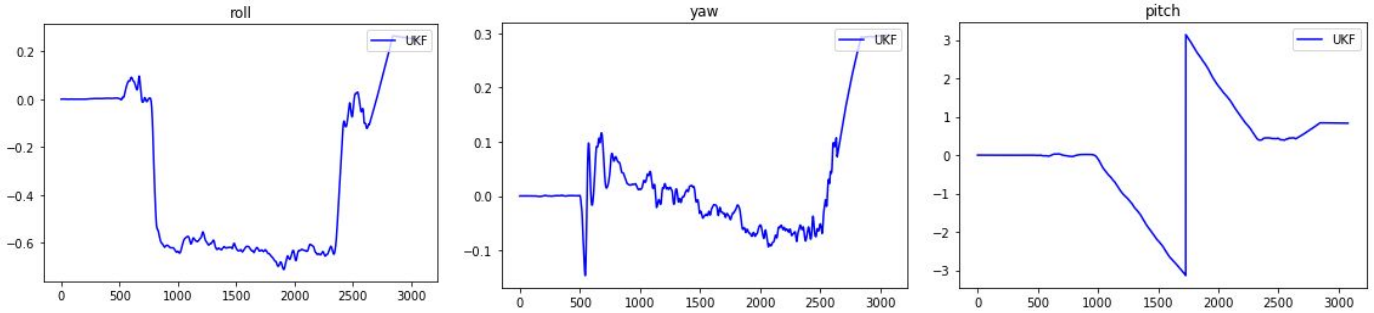


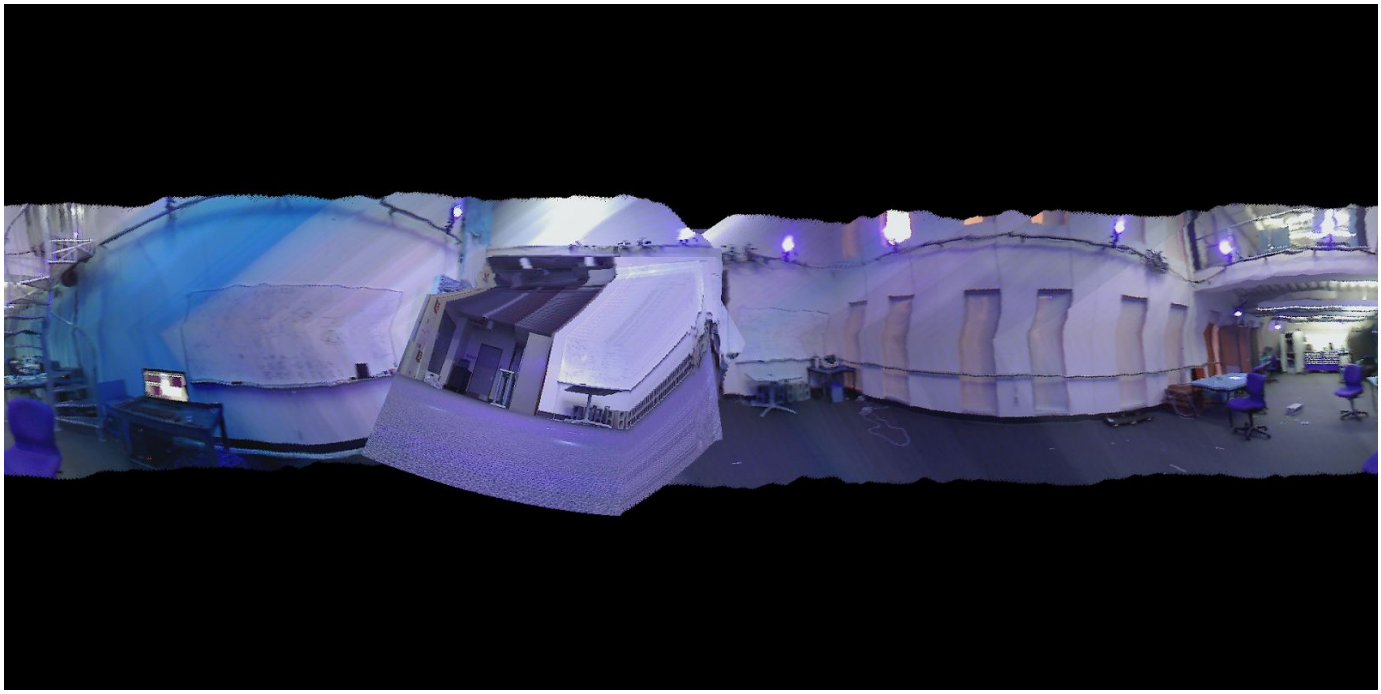
- **Training data set 9**



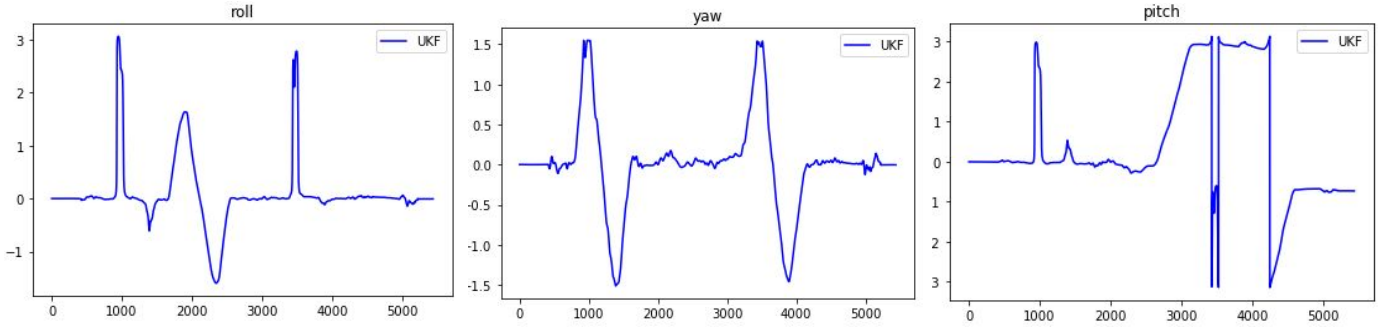
Testing results

- **Testing data set 10**

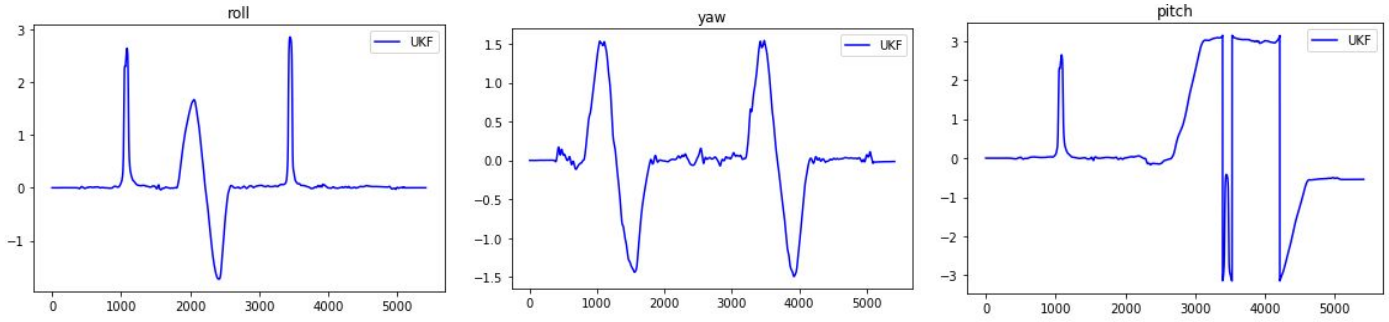




- **Testing data set 11**



- **Testing data set 12**



- **Testing data set 13**