# Orientation Estimation Report

The report contains five parts, introduction, problem formulation, technical approach, tests results and brief summary.

The key point in the report is the achievement of UKF (Unscented Kalman Filter), including noise optimization, cylinders projection and panorama construction. In addition, the report explains difference between the predicted and update result, as well as problems in stitching image step and improvements.

## I. Introduction

This project is aimed to implement a Kalman filter to track three dimensional orientation of certain object. Given IMU sensor readings from gyroscopes and accelerometers, then need to estimate the underlying 3D orientation by learning the appropriate model parameters from ground truth data given by a Vicon motion capture system.

After the estimation of camera orientation, generate a real-time panoramic image from camera images using the 3D orientation filter.

In general, construct one kind of Kalman filter, i.e, UKF or EKF with given raw data to estimate the orientation of camera, then based on obtained data to stitch image and generate panoramic image.

## II. Problem formulation

### A. Understanding of IMU

The very first step also might be one of the most important key points in this project is to understand how IMU works. In other words, data is crucial for the whole course. Only when we know how IMU works that can we get a good understanding of the measurement acceleration data as well as gyro data, and also will get to know the problems might happen during the whole process.

### B. Data Conversion

Since the given data is obtained directly from the IMU sensor, which means those are raw and analog ones and cannot be applied into filter without any handling.

Therefore, converting the raw A/D values to physical units is necessary and also important, because you cannot get correct estimation by using wrong converted data even with the perfect algorithm and Kalman filter.

The key point in data handling step is to find optimal bias and sensitivity value to make raw data reasonable.

### C. Quaternions and Space Transfer Problem

In the Kalman Filter, we use quaternion to represent state and iterate to update it then achieve filter function. One thing is, quaternion has different operations compared with other vectors, since orientations are periodic, which means quaternions are members of a homogenous Riemannian manifold but not of a vector space. For example, summing up two quaternions directly to represent the change of orientation is illegal which will generate wrong orientation. Also for averaging calculation of quaternions, applying barycentric method is wrong. In a word, dealing with quaternions properly is the key to the project since we need states to represent measurement orientations.

For another thing is, dimension match issue matters a lot to the performance of UKF. We use quaternion to represent each state which is 4D, while the measurement acceleration data obtained from IMU is 3D and belong to vector space. So when calculating covariance matrix or selecting sigma points from covariance matrix to represent state, we need to make the dimension match. Although there exist approaches to convert quaternion into rotation vector and rotation vector to quaternion space, when and which vector needed to convert is pretty important to decide.

### D. Optimal Process and Measurement Noise

Since the data is obtained in the physical environment, the consideration of noise is necessary. In the Kalman Filter, we need to add noise in both motion and observation models to stimulate the real – time process. Therefore, choosing suitable represent form of noise as well as optimal scale factor are pretty important. For example, moving IMU quickly or change the orientation in a high frequency will generate more noise compared with static one.

In order to optimal UKF's performance, noise debugging matters a lot.

### E. Images Stitching and Panorama Construction

Visualization is a recommended tool to check the accuracy of estimated orientation. One of approaches is stitching image data together via corresponding estimated rotation matrix to generate a panorama and check whether the final result is reasonable or not. Here are three problems.

First, the time stamp in cam data and IMU data might not be matched, therefore, it's highly possible that you will find delay problem when stitching images, for example, the contour of image is moving or rotating but the image content is static.

Second, translation problem. Since the data set doesn't include translation information, so when camera translated without rotating, the contour of image will keep static while content is changing which will disturb the final panorama.

Third, moving camera to some extreme degree will cause worse performance in the panorama. For example, tilting camera up or down to 90 degree will make the pixels' corresponding position go to infinity. This special case also needed to be handled with.

## III. Technical Approach

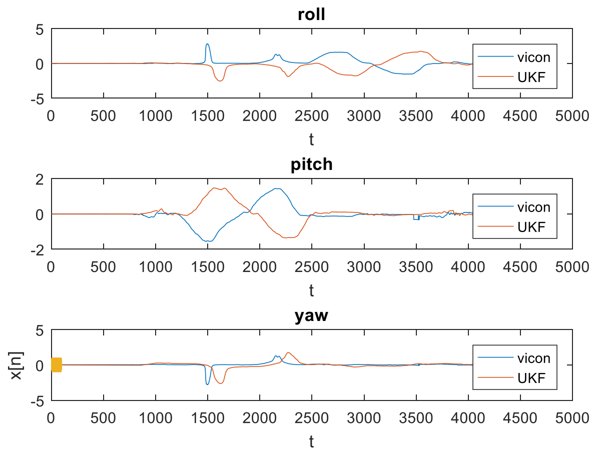
This part shows some several technical approaches in my code to deal with mentioned problems above. And we mainly focus on noise optimization and image stitching approach.

### A. Data Handle

The key point when dealing with data is finding corresponding bias and sensitivity value for IMU raw data.

Use the information in the datasheet directly is not a good idea. It's recommended to use the data when IMU in the static state to optimize bias value. In addition, keeping the unit of data consistent in the whole project is quite important.

For another thing is, it's better to look at data with understanding of the structure of IMU. In this case, IMU Ax and Ay direction is flipped (due to device design), so positive acceleration in body frame will result in negative acceleration reported by the IMU. At the beginning of the project, I missed this point and get my result like below,



The roll and pitch estimation are reversed due to the above reason.

Therefore, it's quite important to handle data properly and there is no other trick just follow the pipeline.

## B. *Quaternion Calculation*

Space transformation plays a main role in the algorithm of Kalman Filter. In the most case, our state is a quaternion vector while the IMU measurement is in the vector space. However, quaternion is a special type vector that cannot apply normal operations such as barycentric averaging method. Therefore, the issue that how to keep calculations between two vector spaces attaches a great importance to the accuracy of project. Here is what I am doing to try to avoid mistakes.

I designed a new class of quaternion in Python to hold all quaternion operations, such as multiplication, averaging, exponential and log approach. Then it's quite easy for me to make sure my quaternion operation is correct by comparing the result with the one calculated by built in function in Matlab.

More details in the averaging approach. Unlike barycentric method to compute mean of vector, we need to choose iteration and make mean quaternion converge. Just set threshold of error between current and previous iteration turn to be the terminal condition of iteration.

## C. *Dimension Match*

When comes to dimension match, the one and only one key is being more and more careful and patient to deal with each conversion between quaternion and rotation space. I choose 4D vector to represent state

$$q_k = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}$$

And use gyro data as control signal as input of the filter

$$q\Delta = \left[\cos\left(\frac{\alpha\Delta}{2}\right), \vec{e}_\Delta \sin\left(\frac{\alpha\Delta}{2}\right)\right]$$

Both two are quaternion vectors so when apply the control signal into the state, adding is illegal, we should use the multiplication of quaternion. That's pretty trick and special in the project that, no addition or subtraction for quaternions, we should use multiplication to implement above two basic operations.

The procedure is similar in UKF and I will not go to more details. Here is what I am doing to make dimension match and avoid mistakes.

First, when picking up sigma points, it's better to convert each column vector which is 3D in covariance matrix into quaternion space instead of converting state quaternion into vector space, the reason is, when converting quaternion into rotation vector space, sometimes might happen convergence problem which make the result bad.

Second, in the update step, we use Kalman Gain to update our state vector. However, state is 4D quaternion while Kalman Gain term is 3D, so remember use exponential method to convert 3D into 4D quaternion space and then, do not add but applying multiplications to update quaternion.

In a word, I try to keep my state always in 4D form without converting it into 3D vector, and keep in mind that quaternion cannot add or subtract, using multiplication is the correct approach..

## D. *Prediction First and then Update*

Due to the complexity of the UKF which including motion model and measurement model, it's quite difficult to debug when meeting some problems. Therefore, separating two models first and then combined together is a pretty fine approach, that is, implement prediction step first without updating and make the motion model reasonable, after that,, use measurement model to improve filter result.

The above strategy will reduce the likelihood to make mistakes and also do a lot of favor when debugging. For example, if your prediction step is fine but filter result is not good, the problem is highly possible in the update step.

## E. *Noise Issue*

It's pretty tricky to select suitable noise value to make UKF perform well. The filter is supposed to add total two noise, one for motion model and we call it Q, the other is for measurement model which is R. Therefore, here comes the approach to select the scale factor for two noises.

My suggestion and also what I am doing is, implementing prediction step first and try to optimize Q noise. Then adding update step and try different value of R to make the performance of filter much reasonable.

When looking at the Kalman Gain, noise R is something like a 'confidence factor', that is, the value of R represents how much the filter trust the actual data compared with current prediction state.

The fact is, measurement data in IMU must have some errors, such as some glitches or unusual orientation change. In that case, it's better to adjust R's value based on three data, current prediction state, current actual measurement state and previous actual state. Once you find there exist large difference between current and previous state, setting a large value of R is better since the Kalman Gain contains the inverse of R. Similarly, when finding the change is reasonable, choosing small value of R is much better.

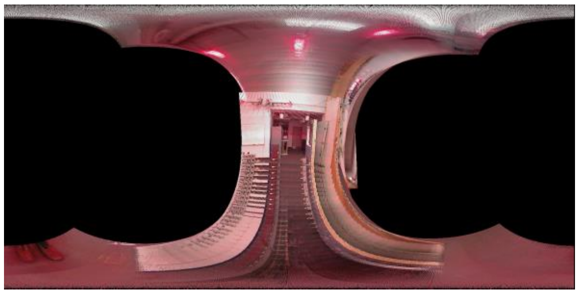
## *F. Image Stitching and Panorama Construction*

The approach in creating a panorama is,

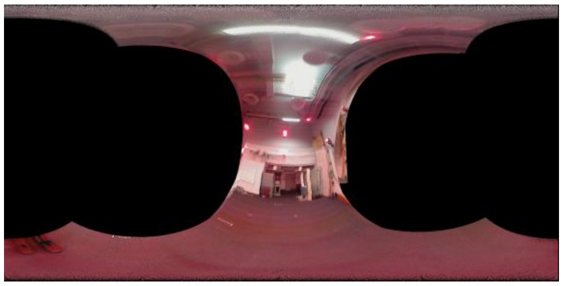
1. Mapping pixel position in a longitude, latitude coordinate which represented in spherical.
2. Convert spherical system into Cartesian coordinate and then apply corresponding estimated rotation matrix to make the position physically.
3. Then convert Cartesian into spherical again.
4. Map spherical system into a cylinder and finally cut off the surface of cylinder to become a rectangle space with $\pi$ height and $2\pi$ width. And the rectangle is the panorama.

Of course, high accuracy of estimated rotation matrix is quite important, and matching the time stamp of camera image with rotation matrix also matters a lot. That is, IMU data and camera might exist some bias in time domain. Therefore, in order to get good performance of panorama, figuring out this offset and try to compensating it is crucial.

Below are two examples to show how importance to match time step for cam and IMU.
Before time matching

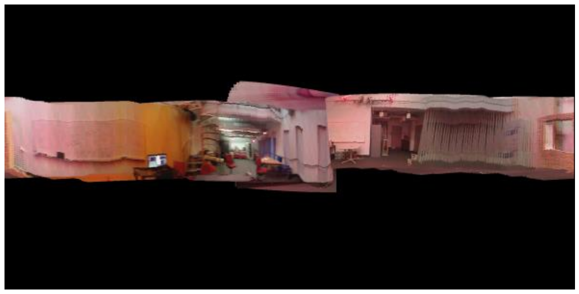


You can find the significant delay between rotation matrix and image content, especially when the camera tilt down and up, the rotation matrix cannot follow the step with image content changing which make the ground view terrible.
Here is the update one with time offset and the result performs better.

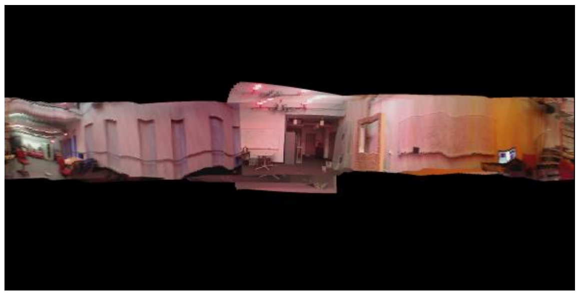


You can also find the video via his link
[https://drive.google.com/open?id=0B-YfsvV6PlJRd1ZkYmxIZzlPWTg](https://drive.google.com/open?id=0B-YfsvV6PlJRd1ZkYmxIZzlPWTg)
The video name is ‘tesr1_parnoma.avi’.

Another example that shows the improvement.
Before handling

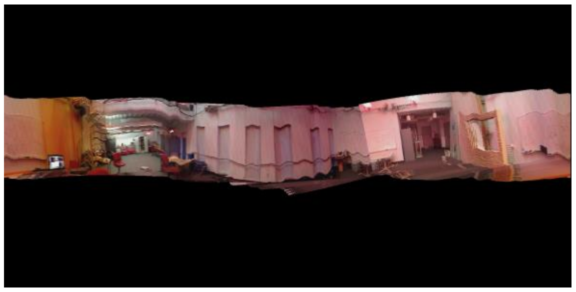


Obviously, the right half is terrible since static image with left yaw direction rotation matrix.
After matching, the result is better.



You can also find its video in the above link.

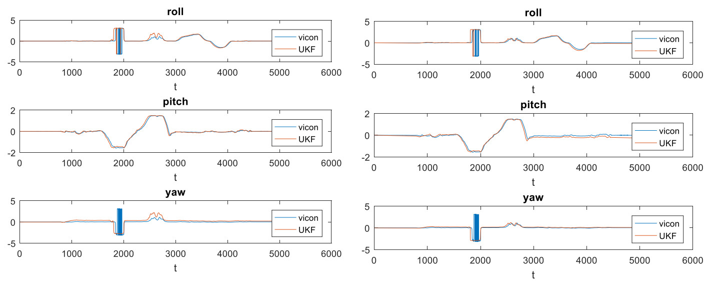
Another good panorama



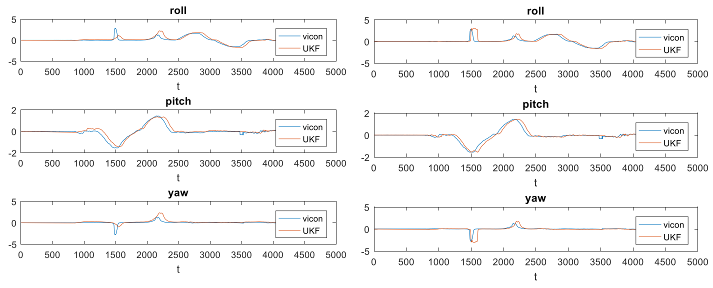
## IV. Test Results and Discussion

Total 9 imuRaw training set and 4 test set. Below are part of imuRaw results and all test ones.
(The left figure is the predicted result and right one is obtained after adding update step.)

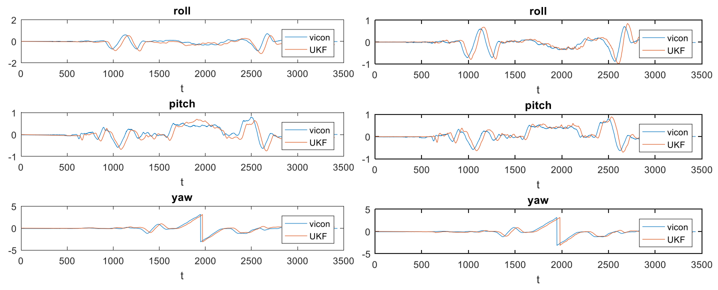
### *imuRaw1*



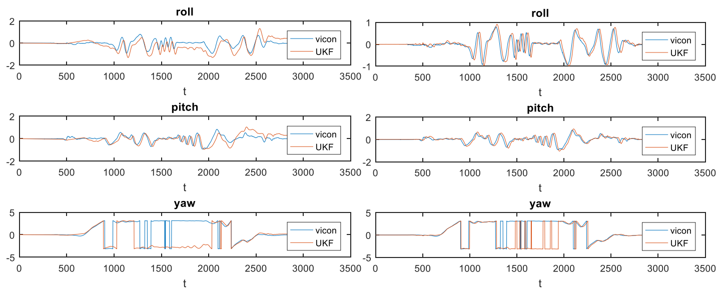
### *imuRaw2*



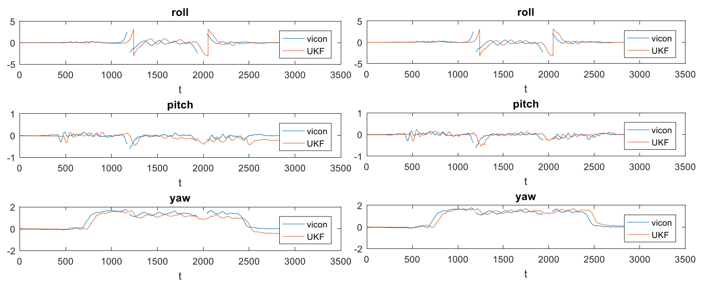
### *imuRaw3*



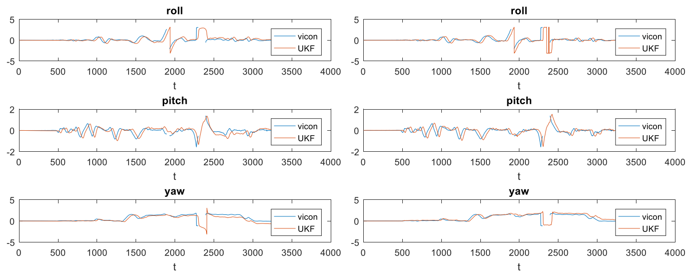
### *imuRaw5*



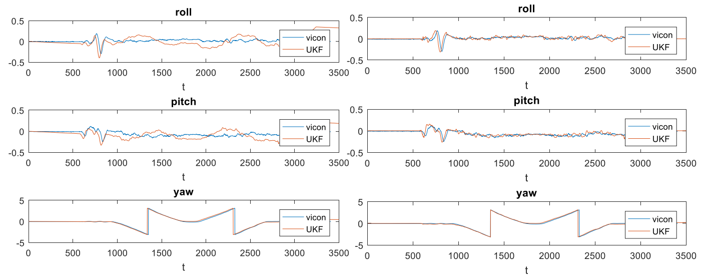
### *imuRaw6*



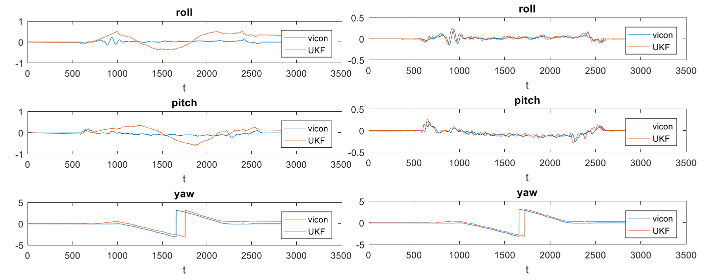
### *imuRaw7*



### *imuRaw8*



### *imuRaw9*

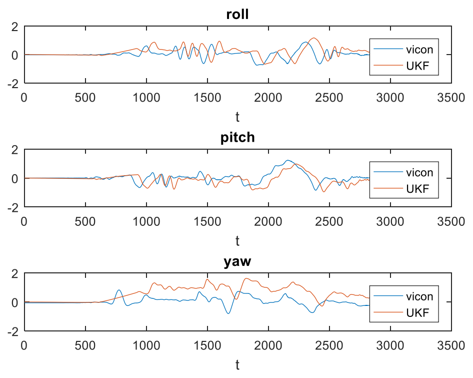


As you can see, my prediction step performs pretty reasonable and well, and my update step can even improve the filter and makes it better.

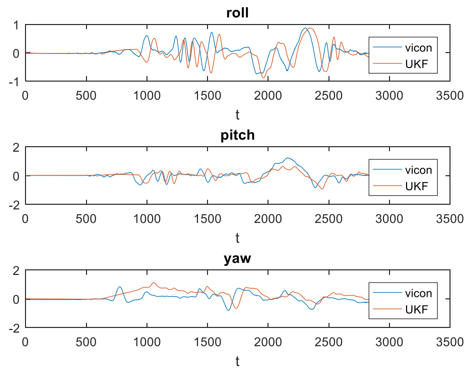
However, when the camera rotates in a quite fast speed, it may introduce more noise in the system which makes my result preforms worse since the actual measurement may mislead the state and update to wrong orientation.

For imuRaw4, although my prediction step performs not so bad, when adding update step, the result in fact didn't improve much.

The prediction result



Then after adding update step

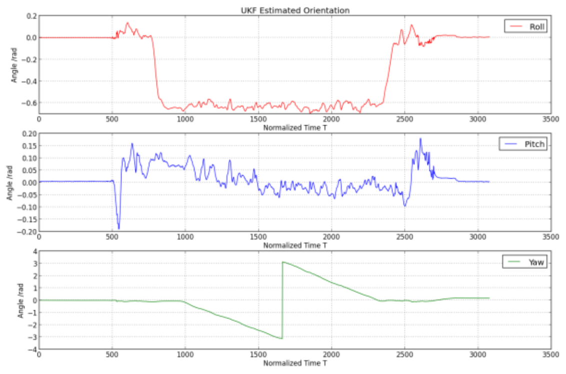


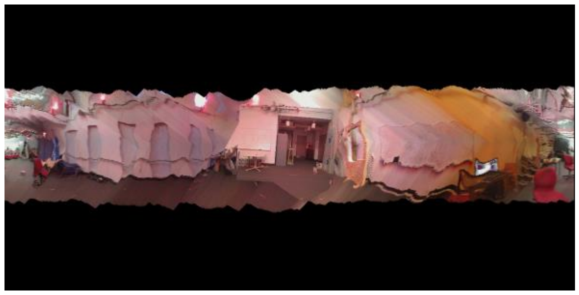
In fact, update step truly worked but not significant as previous training data. In my view, the reason is the value of my scale factor of measurement noise.

One possible improvement is, setting real – time noise to simulate physical rotation movements, meanwhile, also setting suitable 'confidence factor' for noise in the measurement model and try to make the system trust true data and reduce the impact brought by error measurement.

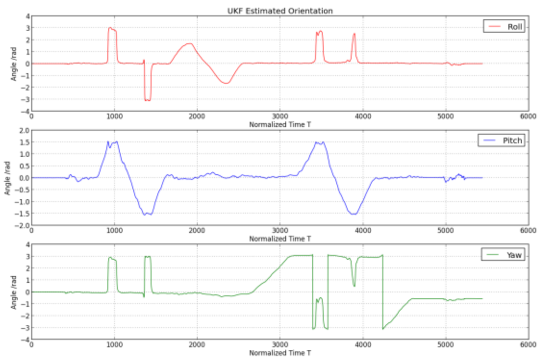
<u>Four test estimated results and corresponding panoramas</u>

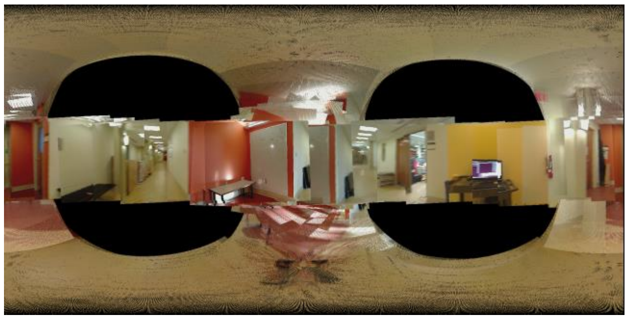
<u>Test 10</u>



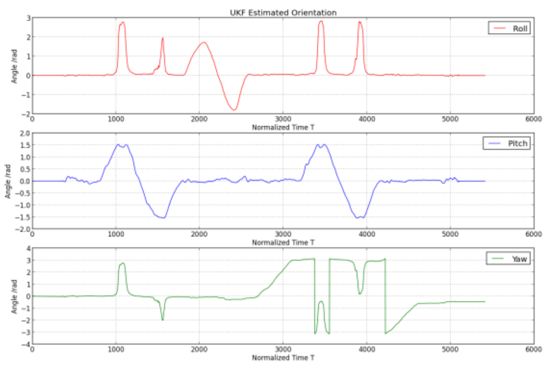


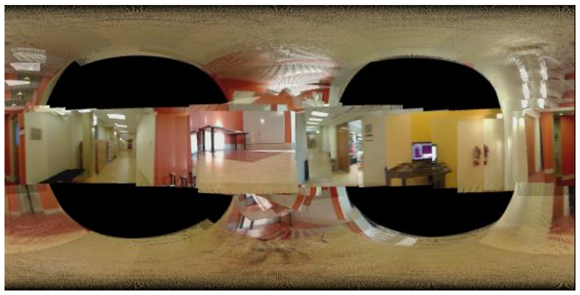
<u>Test 11</u>



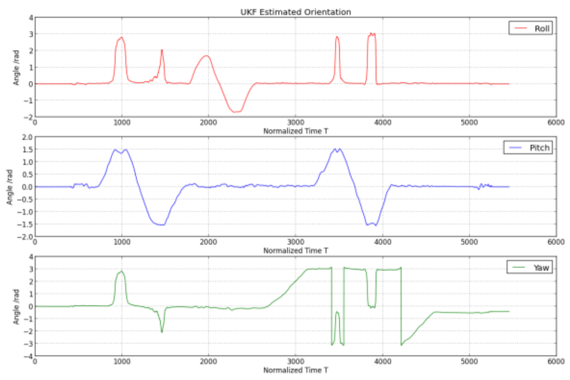


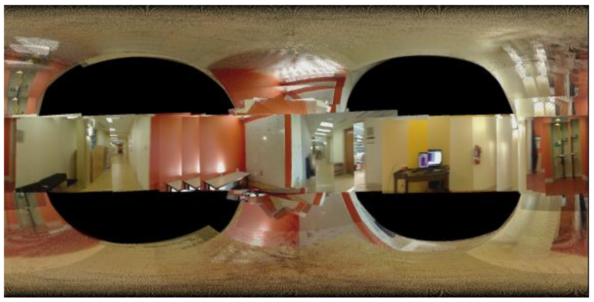
Test 12





Test 13





## V. BRIEF SUMMARY

The main idea of this project is, using raw data obtained from IMU to construct Kalman Filter and achieve the estimation of camera orientation. Finally generate the panorama based on estimated rotation matrix and corresponding camera images.

The key points in this project are dealing with quaternions properly, focusing on dimension match and vector space conversion issues, implementing prediction and update step in correct strategy.

However, words are always easier than behaviors. I truly met multiple problems during the process, such as adding or subtracting quaternions directly rather than using multiplications, update steps performing bad since some errors happen in my vector space transformation, and also feeling pretty painful when stitching images and constructed panorama.

Finally, after multiple times of testing and with the help of professor, kind TAs and classmates, the UKF performed not bad and also can generated reasonable panoramas and videos.