Orientation Tracking

1. Introduction

As today's robotics technology improves, their tasks require more and more awareness of the surroundings and its own status. Without abundant amount of sensor readings, robots can't precisely analyze the environment and execute its missions. Robotic systems, especially fully autonomous machines, rely heavily on acquiring and processing tracking information such as GPS, speed, orientation, etc., and sensors readings sometimes are noisy and require filters. This project is Orientation Tracking on a platform set up with 3-axis gyroscope, accelerometer and a camera, where we use Unscented Kalman Filter to estimate orientation and try to construct a panorama based on orientation and camera images.

2. Problem Formation

Camera data includes RGB images with timestamps. Orientation data includes raw readings from the Inertia Measurement Unit and accelerometer with corresponding timestamps. Each entry in the dataset contains raw reading 3-axis acceleration [A_x A_y A_z] and 3-axis angular rotation $[\omega_x, \omega_y, \omega_z]$

The task is to track the orientation (yaw, pitch, roll) of the platform given an initial orientation, then stitch together a panorama using an array of orientations and corresponding camera image.

3. Technical Approach

First step is to un-bias raw voltage readings from the sensors to get correct values and units. Bias is obtained from averaging some of the first data points known as stationary.

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

$$\text{scale\_factor} = \text{V}_{reference} \,/\, 1023 * \text{sensitivity}$$

Where $\text{V}_{reference}$ – reference voltage for A/D converter

Sensitivity is sensor specification

In this project, quaternion is used to keep record of orientation. Quaternion has the form of $\text{q} = [\text{q}_s, \overrightarrow{\text{q}_v}] = \text{q}_s + \text{q}_1 i + \text{q}_2 j + \text{q}_3 k$ and they represent rotations in 4D space with no singularities.

A straightforward and crude way of updating orientation is simple integration, without applying any filter or processing. Simply use the equations for each angular rotation data $[\omega_x, \omega_y, \omega_z]$ and we'll have the updated orientation.

$$q_{rotation} = \left[0, \frac{1}{2}\omega_t \Delta t\right]$$

$$q_{t+1} = q_t \circ \exp(q_{rotation})$$

Where $\omega_t = [\omega_x, \omega_y, \omega_z]$

Δt – time elapsed since previous data point

$\circ$ – quaternion multiplication

exp(q) – quaternion exponential map

The result from simple integration is not that accurate as it's just additively cascading rotations to current orientation, and it doesn't take into account possible noise and variation. A much better method is to use Unscented Kalman Filter (UKF) to account for the noise and use acceleration data to correct estimation.

To construct UKF filter, we first initialize orientation $q_{0|0} = [1,0,0,0]$ and covariance $\Sigma_{0|0} = 0.0001\, I$ and assume noises $Q = \Sigma_{motion} = 0.0001\, I$ and $R = \Sigma_{observation} = 0.0001\, I$

Then for each data entry [A_x A_y A_z] and with $\overrightarrow{\omega_t} = [\omega_x, \omega_y, \omega_z]$,

1) Calculate $q_\Delta = [\cos\left(\frac{\alpha\Delta}{2}\right),\ \vec{e}_\Delta \sin\left(\frac{\alpha\Delta}{2}\right)]$, the incremental change in rotation

   Where $\alpha_\Delta = |\overrightarrow{\omega_t}| \cdot \Delta t$

   $\vec{e}_\Delta = \frac{\overrightarrow{\omega_t}}{|\overrightarrow{\omega_t}|}$

   Δt – time elapsed since previous data point

2) Find Sigma points {W}, which is array of noise vectors

$$W_{i,i+n} = columns(\pm\sqrt{n \cdot (\Sigma_{t|t} + Q)}$$

   Where n = 3, dimension of $\overrightarrow{\omega_t}$

   Q – covariance of motion noise

3) Find Sigma points {X}, which is array of current orientation affected by noise {W}

$$X_0 = q_{t|t}$$

$$X_i = q_{t|t} \circ \exp(q_{noise\_i}), i \in (1,2,3,4,5,6)$$

   where $q_{noise\_i} = \left[0, \frac{1}{2} W_i\right]$

4) Find Sigma points {Y}, which is array of predicted orientation affected by noise

$$Y_0 = q_{t|t}$$

$$Y_i = X_i \circ q_\Delta, i \in (1,2,3,4,5,6)$$

5) Find the mean $q_{t+1|t}$ and covariance $\Sigma_{t+1|t}$ of set {Y}

   For a set $\{Y\} = [q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$ and weights $\alpha = \left[2, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right]$

   Initialize guess $\widetilde{q_0} = Y_0 = q_{t|t}$ (current mean)

```
For t = 0,1 …, T:
    $q_i^e = [q_{s,i}^e,\ q_{v,i}^e] = \tilde{q}_t^{-1} \circ q_i$
      $[0, e_{v,i}] = 2\log(q_i^e)$          ⟵ Error rotation vector from quaternion
      $e_{v,i} = (-\pi + \mathrm{mod}(\|e_{v,i}\| + \pi,\ 2\pi)) \frac{e_{v,i}}{\|e_{v,i}\|}$        ⟵Restrict angles to [-π, π)
      $e_v = \sum_{i=1}^{i=7} \alpha_i e_{v,i}$
      $\tilde{q}_{t+1|t} = \tilde{q}_t \circ \exp([0, \frac{e_v}{2}])$      ⟵ Error rotation vector back to quaternion
      if $\|e_v\| < \epsilon$:
          return $\tilde{q}_{t+1|t}$

$\Sigma_{t+1|t} = \sum_{i=0}^{i=7} \alpha_i * \mathbf{e}_{v,i} * \mathbf{e}_{v,i}^T$      ⟵ error rotation vector from last iteration t
```

6) Construct Sigma points {Z}

$$g = [0,0,0,1] \text{ gravity quaternion}$$
$$Z_i = Y_i \circ \exp(\mathrm{g}) \circ Y_i^{-1}, i \in (0,1,2,3,4,5,6)$$

7) Find mean z^- and covariance P_{zz} of set {Z} with acceleration measurements

$$z_k = [A_x\ A_y\ A_z]^T$$
$$z^- = \sum_{i=0} weight_{mean\_i} * Z_i$$
$$P_{zz} = \sum_{i=0} weight_{cov\_i} * (Z_i - z^-)(Z_i - z^-)^T$$

With $weight_{mean\_0} = 0$ and $weight_{mean\_1,2,3,4,5,6} = \frac{1}{2*n}$, (n=3, dimension of z_{k})

$weight_{cov\_0} = 2$ and $weight_{cov\_1,2,3,4,5,6} = \frac{1}{2*n}$, (n=3, dimension of z_{k})

8) Calculate Kalman Gain $K_{t+1|t}$

The noise affected covariance

$$P_{vv} = P_{zz} + R = P_{zz} + \Sigma_{observation}$$

The cross-correlation matrix P_{xz} relates noise in prediction to noise in measurement

$$P_{xz} = \sum_{i=1}^{2n+1} \alpha_i * [\log(q_{t+1|t}^{-1} \circ Y_i).V] * [Z_i - z^-]^T$$

Where weights $\alpha = \left[2, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right]$

.V operator extracts q_v from $q = [q_s, \boldsymbol{q}_v]$

z^- - mean of {Z}

Innovation vector v is the difference between actual acceleration measurement and the predicted value z⁻

$$v = z_k - z^-$$

Finally, Kalman Gain

$$K_{t+1|t} = P_{xz} * P_{vv}^{-1}$$

9) Update predicted mean $q_{t+1|t+1}$ and covariance $\Sigma_{t+1|t+1}$

$$q_{t+1|t+1} = \mathrm{q}_{t+1|t} \circ \exp(\left[0, \frac{1}{2} * K_{t+1|t} * v\right])$$

$$\Sigma_{t+1|t+1} = \Sigma_{t+1|t} - K_{t+1|t} * P_{vv} * K_{t+1|t}^T$$

After all the orientations are estimated, we then proceed to construct panorama with timestamped images. First, we take set up a template of spherical coordinates with r=1 for each pixel in the M x N mask, which in this case is 240 x 320. We assume $\pm 22.5°$ vertical view and $\pm 30°$ horizontal view, which is this matrix of coordinates:

$\theta = -22.5°, \phi = 30°$	…	$\theta = 0°, \phi = 30°$	…	$\theta = 22.5°, \phi = 30°$
…	…	…	…	…
$\theta = -22.5°, \phi = 0°$	…	$\theta = 0°, \phi = 0°$	…	$\theta = 22.5°, \phi = 0°$
…	…	…	…	…
$\theta = -22.5°, \phi = -30°$	…	$\theta = 0°, \phi = -30°$	…	$\theta = 22.5°, \phi = -30°$

Then convert each spherical coordinates to cartesian and apply orientation transformation.

$$(\theta, \phi, r) \rightarrow (x, y, z)$$

$$x = rsin(\phi)\cos(\theta)$$

$$y = rsin(\phi)\sin(\theta)$$

$$z = rcos(\phi)$$

$$[0, (x', y', z')] = q_{t+1|t+1} \circ [0, (x, y, z)] \circ q_{t+1|t+1}^{-1}$$

Convert (x', y', z') back to cylindrical coordinate system to place in the composite panorama image.

$$r = 1$$

$$\theta' = \arctan(\frac{\sqrt{x'^2 + y'^2}}{z'})$$

$$z' = \mathrm{z}'$$

Make a blank panorama image with sizes few times larger than the original image, here 3*M and 3*N are used, making a 720 x 960 image.

The transformed cylindrical coordinates $(1, \theta', z')$ are stretch or compressed to fit in the corresponding location by mapping $-1 < z' < 1 \rightarrow 0 < z < 3M$

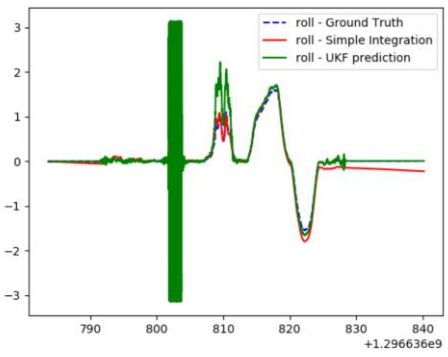
$$-\pi < \theta' < \pi \rightarrow 0 < \theta < 3N$$

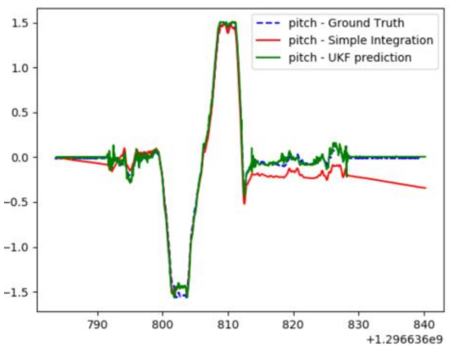
Lastly, place each pixel into the panorama image with mapped matrix indices.

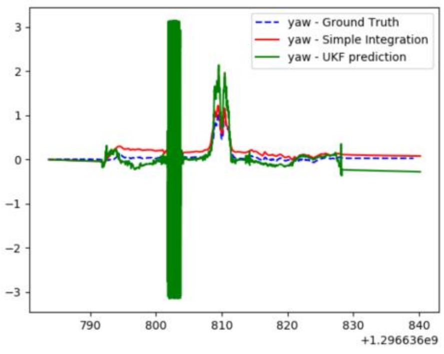
$$panorama[z, \theta] = current\_pixel$$

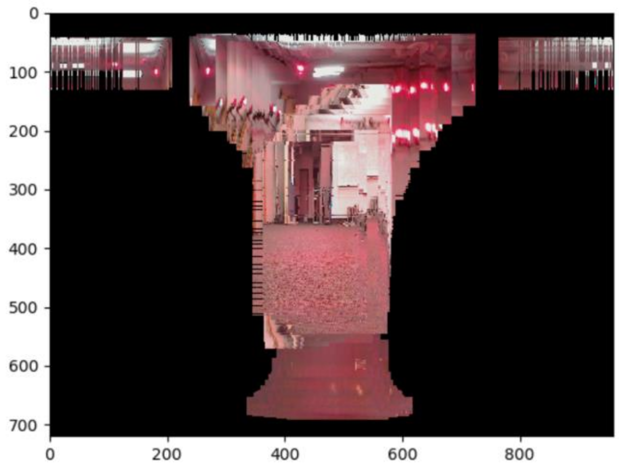
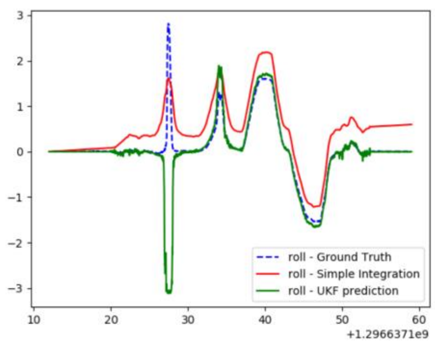
4. **Results and Discussion**

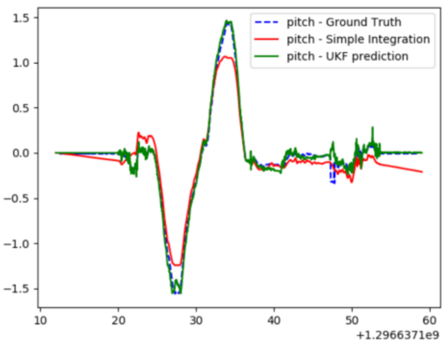
Here are some results from the training sets.

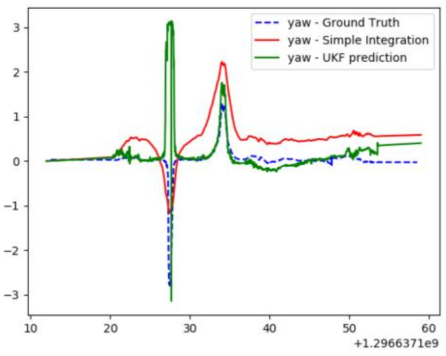


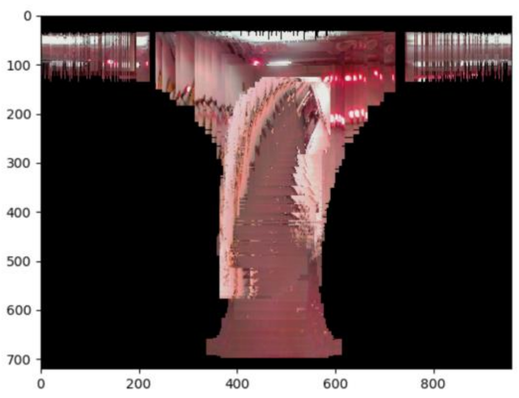


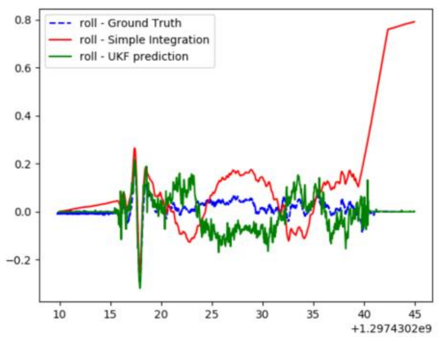


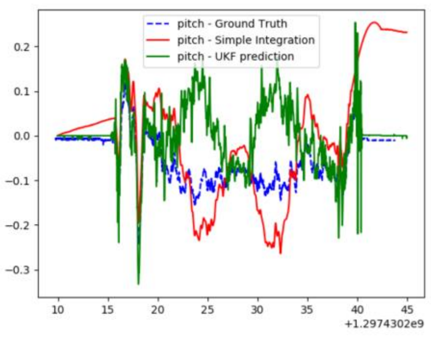



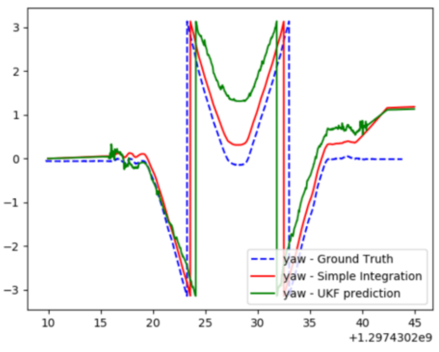


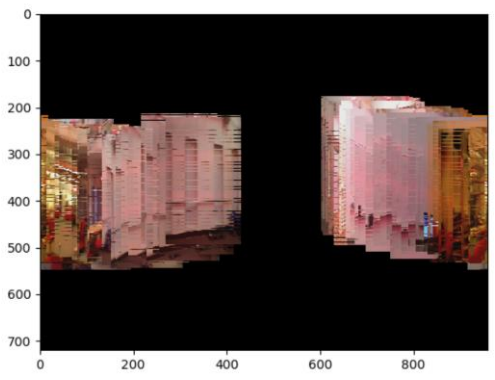


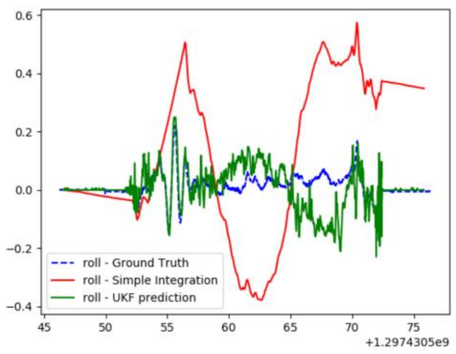


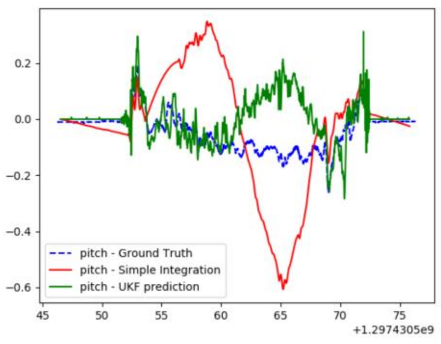

roll - Ground Truth
roll - Simple Integration
roll - UKF prediction
+1.2974302e9

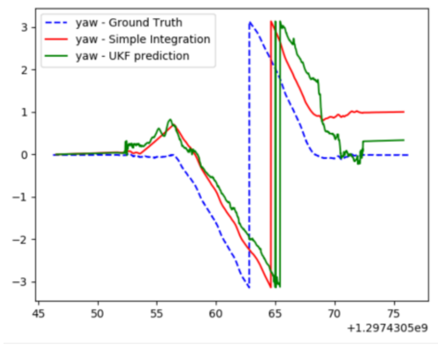

pitch - Ground Truth
pitch - Simple Integration
pitch - UKF prediction
+1.2974302e9

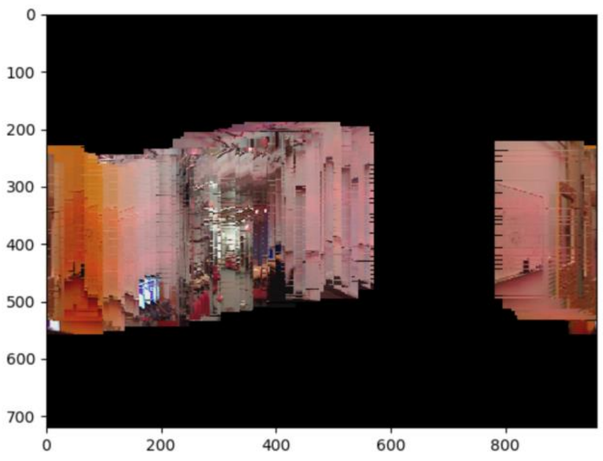

yaw - Ground Truth
yaw - Simple Integration
yaw - UKF prediction
+1.2974302e9




roll - Ground Truth
roll - Simple Integration
roll - UKF prediction
+1.2974305e9

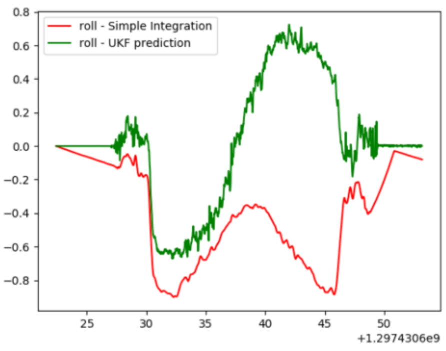

pitch - Ground Truth
pitch - Simple Integration
pitch - UKF prediction
+1.2974305e9

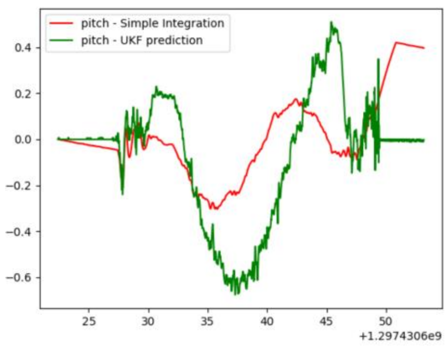

yaw - Ground Truth
yaw - Simple Integration
yaw - UKF prediction
+1.2974305e9

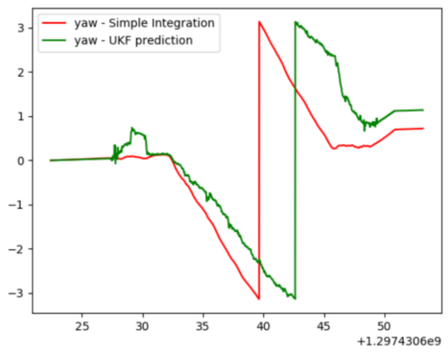


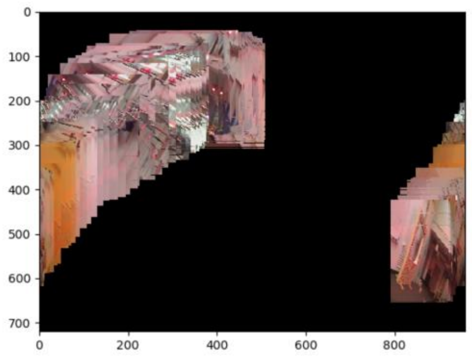
From the above training results, UKF prediction is slightly accurate than simple integration by taking measurement noise into account. However, for some unknown reason, the signs of estimation for yaw and roll are flipped for the dips early in the dataset 2, but it's not present in other datasets with similar dips. The panoramas are a bit choppy and discontinuous due to the jumpy estimations, but the rough outline of the stitched panorama is visible.

Testset 10

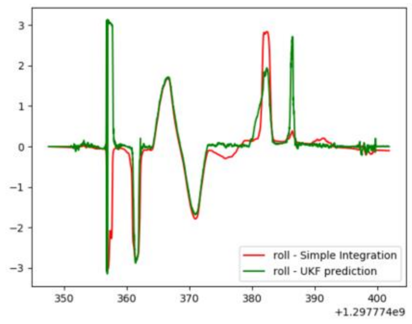


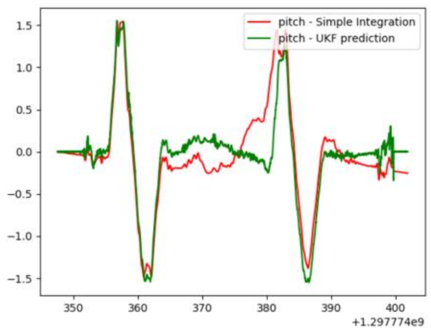


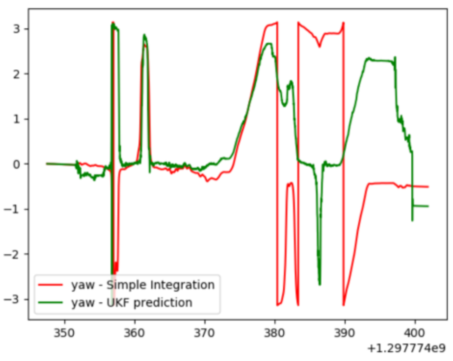


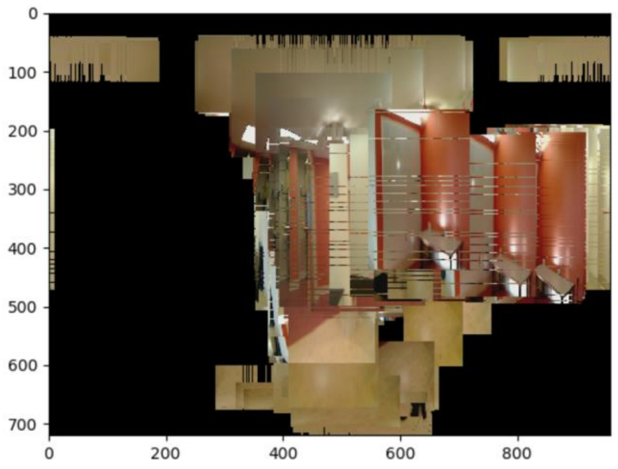


Testset 11

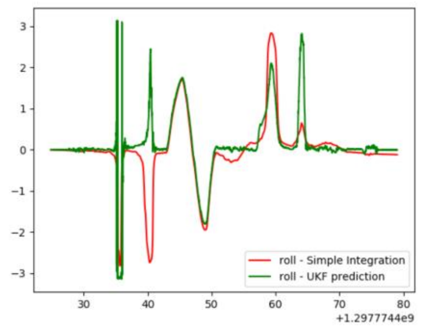


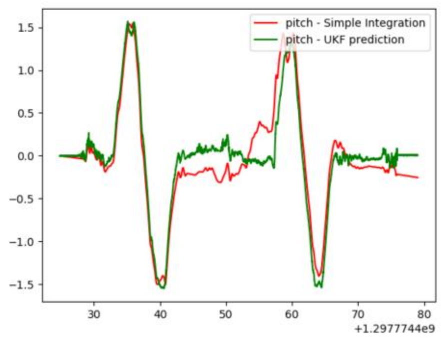


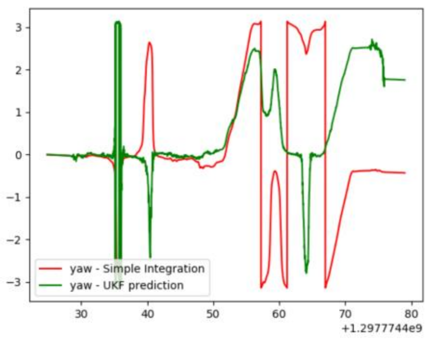


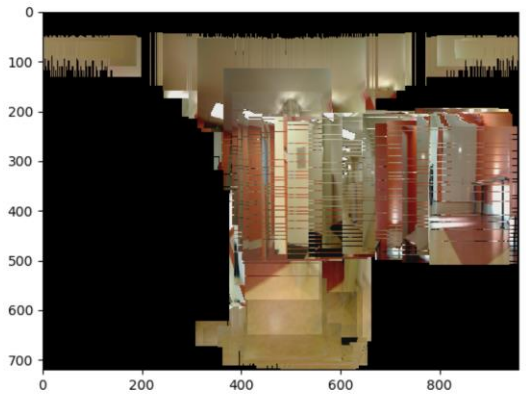


Testset 12

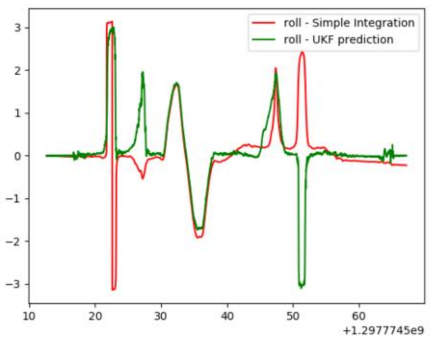


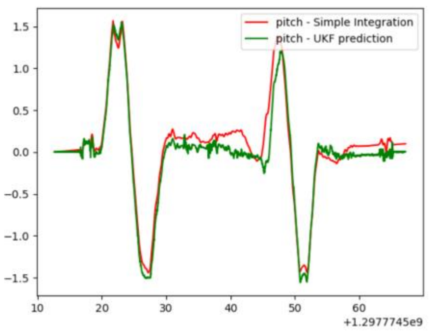


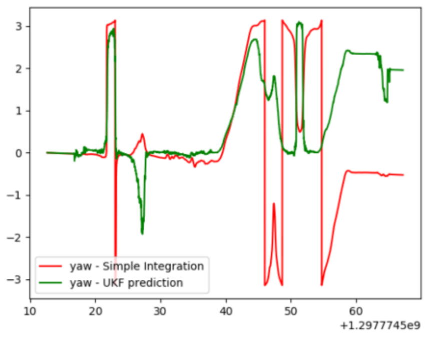


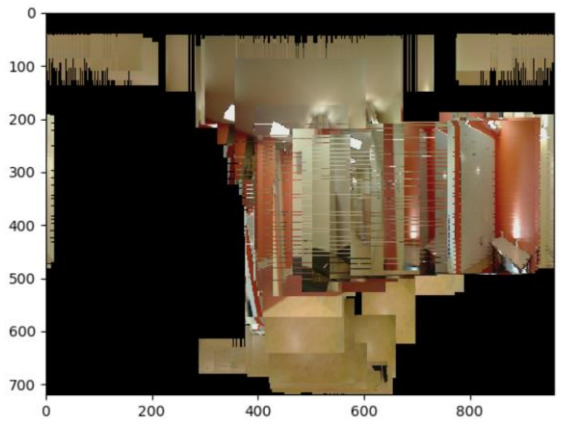


Testset 13









For testset 10, the panorama is not so recognizable possibly due the noisy estimation waveforms. It's also possible that the way I implemented image stitching is not ideal since putting conseqутive images with little orientation change together would overlap them multiple times and increase choppiness. The performance of estimation could also improve by tweaking the covariance parameters to ideal values for each dataset.