Orientation Tracking from IMU Data and Optimization on Estimated Orientation

Yuxiang Kang
Department of Mechanical and Aerospace Engineering
University of California, San Diego
La Jolla, U.S.A
yuk007@ucsd.edu

Nikolay Atanasov
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, U.S.A
natanasov@eng.ucsd.edu

Abstract — **This paper considers a rotating body's orientation tracking problem with estimation optimization based on projected gradient descent algorithm. An IMU is attached to a rotating body, collecting linear acceleration and angular velocity data. With angular velocity data, an estimated orientation is created using quaternion kinematics motion model. The error between the estimated orientation and the motion and observation model prediction creates a cost function, which is tasked to be minimized by projected gradient descent algorithm to increase estimated orientation's accuracy. With estimated orientation, a panoramic image is constructed by stitching camera images obtained by the rotating body.**

Index Terms—**Orientation tracking, Projected gradient descent, Inertial Measurement Unit**

I. Introduction

IMU (Inertial Measurement Unit), initially used in aircraft navigation and large devices, is currently widely used in mobile electronic devices to track spatial motions [1], as micro-electromechanical system (MEMS) IMU is introduced with a very attractive feature of low cost, compact, and low processing power [2].

A typical IMU includes a triaxial accelerometer and a triaxial gyroscope, which measures the unit's linear acceleration and angular velocity, respectively. For accelerometer part, a mass block placed next to a metal plate is attached to a spring. When acceleration is applied to accelerometer, it measures the displacement of the metal plate with mass block with respect to another plate using capacitance. For gyroscope part, it uses Coriolis force to detect rotational velocity from the changing mechanical resonance of a tuning fork [2].

With the unit's linear acceleration and angular velocity, we can estimate the unit's orientation using IMU Observation Model [3]. However, IMU unit does not collect the exact spatial orientation, but the change of orientation with time. Hence, we are actually doing a integral calculation when estimation the current orientation with respect with the original one. As there's time gaps between each data segment, discrete acceleration and angular velocity are tasked to cover continuous motion. Apparently, there's error between estimated orientation and the motion model prediction [4].

To minimize the error, different optimization methods are often applied, including batch-optimization [5] and optimization based on neural network [6]. In this case, we use projected gradient descent algorithm to minimize the error.

Projected gradient descent algorithm is applied to large-scale minimization problems subject to simple bounds on the decision variables [7]. Projected gradient descent is based on gradient descent algorithm, whose fundamental idea is that minus gradient of a function points in the direction of steepest descent. In operation, we can iterate the function's input x_k:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

where α_k is step size. Compared with gradient descent, projected gradient descent algorithm applied a constrain on x_{k+1}, by project $x_k - \alpha_k \nabla f(x_k)$ onto a closed convex C. Thus, the iteration can be expressed as:

$$x_{k+1} = \prod_C (x_k - \alpha_k \nabla f(x_k))$$

By repeating the iteration, we can get x_k move along the direction of steepest descent, thus decreasing $f(x_k)$ under the constrain which corresponds to convex C.

Specifically, we are tasked to create an original estimated orientation based on IMU's angular velocity data and an orientation predicted by motion and observation model, which covers both angular velocity and linear acceleration. Then, we minimize the cost function which corresponds to the error between every new estimated orientation and the prediction. In this process, we are using a quaternion kinematics motion model, an accelerometer observation model and projected gradient descent algorithm. With optimized estimated orientation, a panoramic image is generated by stitching camera images obtained by the rotating body.

II. Problem Formation

A. Orientation Tracking

Consider an IMU attached to a rotating body. The IMU collects data I_t along a time sequence $1{:}T$:

$$I_t = [a_x\ a_y\ a_z\ \omega_x\ \omega_y\ \omega_z] = [a_t\ \omega_t]$$

Using a unit quaternion q_t to represent the body-frame orientation at time t ,we can randomly create an original estimated $\boldsymbol{q}_{1:T}$, which is a T*4 matrix. Meanwhile, based on IMU angular velocity data, we can apply a quaternion kinematics motion model $f(q_t, \tau_t\omega_t)$ to calculate $\boldsymbol{q}_{1:T}{}'$:

$$\boldsymbol{q}'_{t+1} := f(q_t, \tau_t\omega_t) \coloneqq \boldsymbol{q}_t{}' \circ \exp([0, \tau_t\omega_t/2])$$

where τ_t stands for the differences between consecutive time stamps.

To compare with IMU acceleration data, we can apply an observation model $h(\boldsymbol{q}_t)$ on estimated $\boldsymbol{q}$ to calculate estimated acceleration data $\boldsymbol{a}_t{}'$

$$\boldsymbol{a}'_t := h(\boldsymbol{q}_t) \coloneqq \boldsymbol{q}_t{}^{-1} \circ [0\ 0\ 0 - g] \circ \boldsymbol{q}_t$$

Apparently, our estimated $\boldsymbol{q}$ has error compared with actual orientation. As a result, calculation based on estimated $\boldsymbol{q}$ has error compared with IMU-based data, as shown in **chart 1**.

Estimated	IMU-based
$\boldsymbol{q}$	$\boldsymbol{q}_{t+1}{}'$
$\boldsymbol{a}_t{}'$	$\boldsymbol{a}_t$

Chart 1: parameters based on different origins

Apparently, when we optimize the estimated orientation $\boldsymbol{q}$, we hope both $(\boldsymbol{q}_{t+1} - \boldsymbol{q}_{t+1}{}')$ and $(\boldsymbol{a}_t{}' - \boldsymbol{a}_t)$ to be minimized. The optimization problem above can be presented with a cost function:

$$c(\boldsymbol{q}) = \frac{1}{2}\sum_{t=0}^{T-1} \left\| 2\log\left(\boldsymbol{q}_{t+1}{}^{-1} \circ f(q_t, \tau_t\omega_t)\right)\right\|_2^2$$

$$+\frac{1}{2}\sum_{t=1}^{T} \|\boldsymbol{a}_t - h(\boldsymbol{q}_t)\|_2{}^2$$

When minimizing $c(\boldsymbol{q})$, we need to keep q_t as a unit quaternion. Hence, we have a constrained optimization problem:

$$\min_{\boldsymbol{q}} c(\boldsymbol{q})$$

$$s.t. \|\boldsymbol{q}_t\|_2 = 1$$

After enough iterations, if the cost function converges, we'll have an accurate orientation sequence $\boldsymbol{q}_{opt}$.

B. Panorama reconstruction

Consider a camera also attached to the rotating body in ***A***. While rotating, the camera keeps taking photo of surroundings at its current orientation. By comparing the photos' time stamps with the optimized quaternion's, we can match some photos with orientations in $\boldsymbol{q}_{opt}$.

Now what we have is the photos' RGB data and the corresponding camera orientation. By applying geometric models, we can get the corresponding coordinates in world frame for each pixel in each photo. Then for each original photo, we can draw it in world frame. By applying this operation to all the matched photos, we'll construct a new panorama picture in world frame.

III. Technical Approach

A. Orientation Tracking

To accomplish orientation tracking and get a more accurate estimated orientation, we process the data from IMU with the method shown in **Fig 1**.

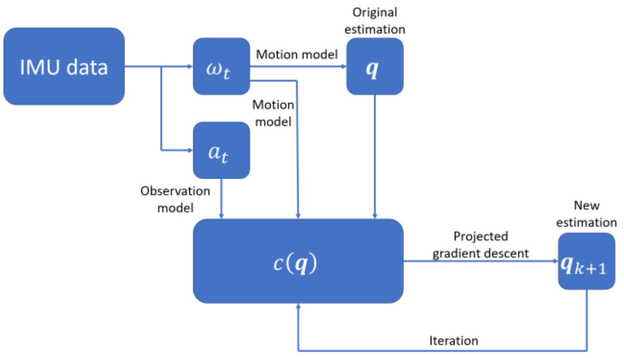

Figure 1: Data processing in Orientation Tracking

In our program, to reduce the original estimated orientation's cost function, we set IMU based orientation sequence $\boldsymbol{q}_{t+1}{}'$ as the original estimated $\boldsymbol{q}$.

To better operate quaternion calculations, we firstly define a program which includes quaternion multiplication, inverse, exponential, logarithm, and conjugation. Then we can just call the functions when we need quaternion calculations.

For the IMU, there is a stational bias. After we read the raw data collected by IMU, we use the following equations to delete the bias:

$$value = (raw - bias) * scale\_factor$$
$$scale_{factor} = Vref/1023/sensitivity$$

According to "IMU_reference.pdf", accelerometer's sensitivity equals 300, gyroscope's sensitivity equals 3.33. To determine bias, we assume the first 100 element along time stamps are in stationary state. By averaging the IMU date of these elements, and comparing them with the ideal state:

$$\boldsymbol{a}_0 = [0\ 0\ g], \boldsymbol{\omega}_0 = [0\ 0\ 0]$$

we can find proper bias to correct the raw data.

Then based on observation and quaternion kinematics motion model, together with IMU data, we've got $\boldsymbol{q}$, $\boldsymbol{q}'$, $\boldsymbol{a}_t{}'$, $\boldsymbol{a}_t$ for cost function calculation.

For the cost function's optimization problem, we can use projected gradient descent algorithm to iterate $\boldsymbol{q}$:

$$\boldsymbol{q}_{k+1} = \frac{\boldsymbol{q}_k - \alpha_k \nabla c(\boldsymbol{q}_k)}{\|\boldsymbol{q}_k - \alpha_k \nabla c(\boldsymbol{q}_k)\|}$$

where α_k stands for step size.

The denominator part normalizes $\boldsymbol{q}_k - \alpha_k \nabla c(\boldsymbol{q}_k)$, representing the constrain. Repeat the iteration and we can get more accurate estimated orientation. Meanwhile, as our original estimated orientation is exactly same as quaternion kinematics motion model's result, to avoid the first part of the cost function to be zero, we added a disturbance into the original estimated orientation sequence. Then, after using "autograd" package to calculate cost function's gradient, we can start our optimization and iterations.

B. Panorama reconstruction

In this part of the project, we've already get the optimized orientation. With the orientation, we can transform a pixel's coordinate form the sphere coordinate in body frame$\{b\}$ to the cylinder sphere in world frame$\{s\}$. This Geometric transformation model is shown in **Fig 2**.

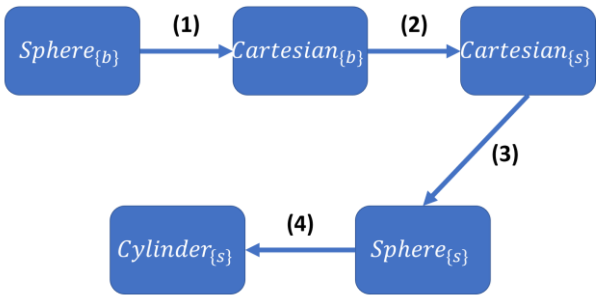

Figure 2: Geometric transformation model for pixels

In this model, **(1) (3)** represents the convert between sphere coordinate $(\lambda, \varphi, 1)$ and cartesian coordinates (x,y,z):

$$\begin{cases} x = cos\varphi cos\lambda \\ y = cos\varphi sin\lambda \\ z = sin\varphi \end{cases}, \begin{cases} \lambda = \arctan\frac{y}{x} \\ \varphi = arcsin\,z \\ r = 1 \end{cases}$$

(2) represents rotating a coordinate in body frame with rotation matrix $\boldsymbol{R}_{sb}$, which comes from orientation quaternion $\boldsymbol{q}$:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\{s\}} = \boldsymbol{R}_{sb} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\{b\}}$$

(4) represents unwrapping a sphere coordinate $(\lambda, \varphi, 1)$ to a cylinder surface coordinate, then to a plane coordinate (x, y):

$$\begin{cases} x = \frac{\lambda}{2\pi} * L \\ y = \frac{\varphi}{\pi} * h \end{cases}$$

Here L represents the perimeter of the cylinder's cross section, and h represents the height of the cylinder. In this particular case, we choose $L = 1920, h = 960$, which means our reconstructed panorama photo's size is 1920*960.

Before applying this geometric transformation, we first calculate every pixel's sphere coordinate based on the camera's parameters. After the transformation, we can paint in a new canvas with the RGB value of every pixel in every photo, guided by the pixel's plain coordinate

IV. Results

A. Orientation Tracking

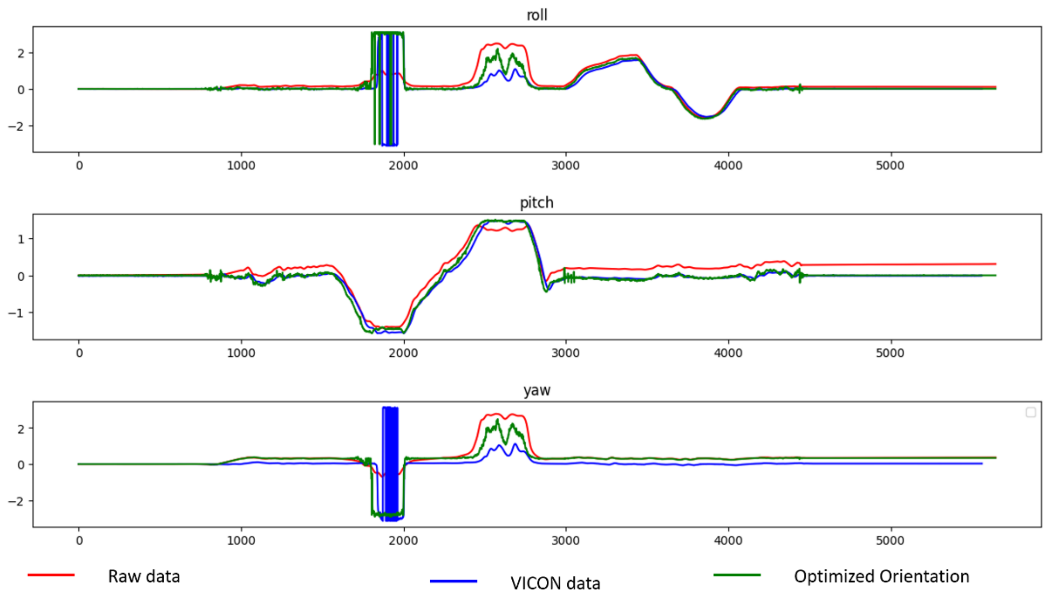

Figure 3: Orientation of Dataset 1(train set)

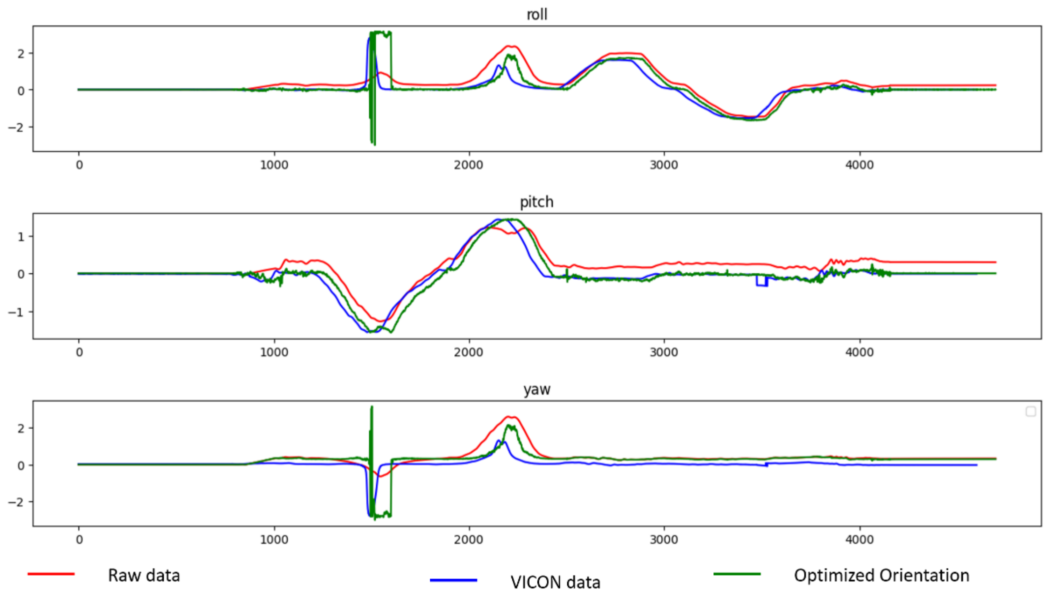

Figure 4: Orientation of Dataset 2(train set)

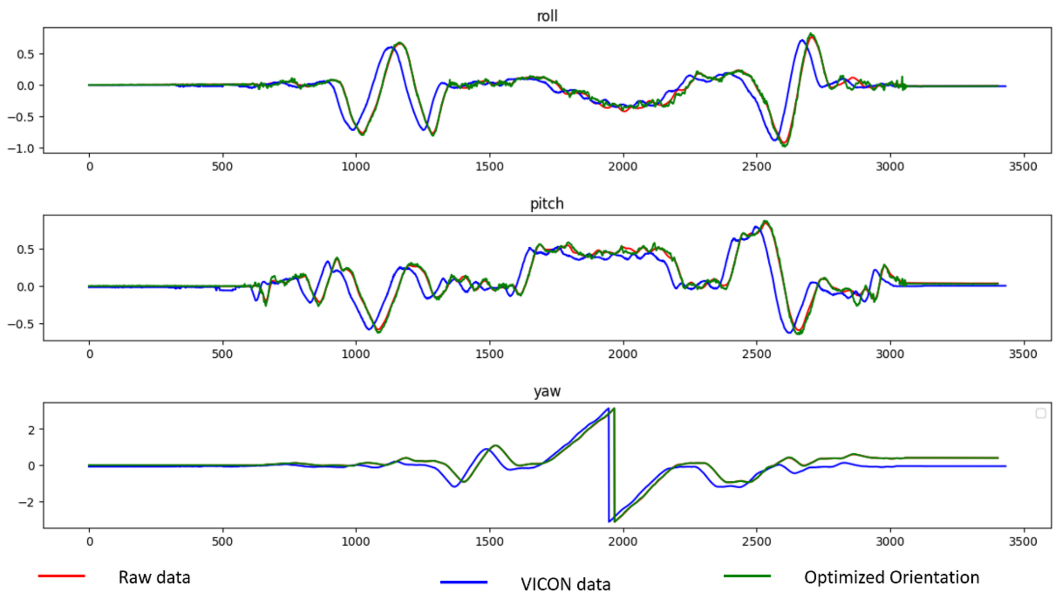

Figure 5: Orientation of Dataset 3(train set)

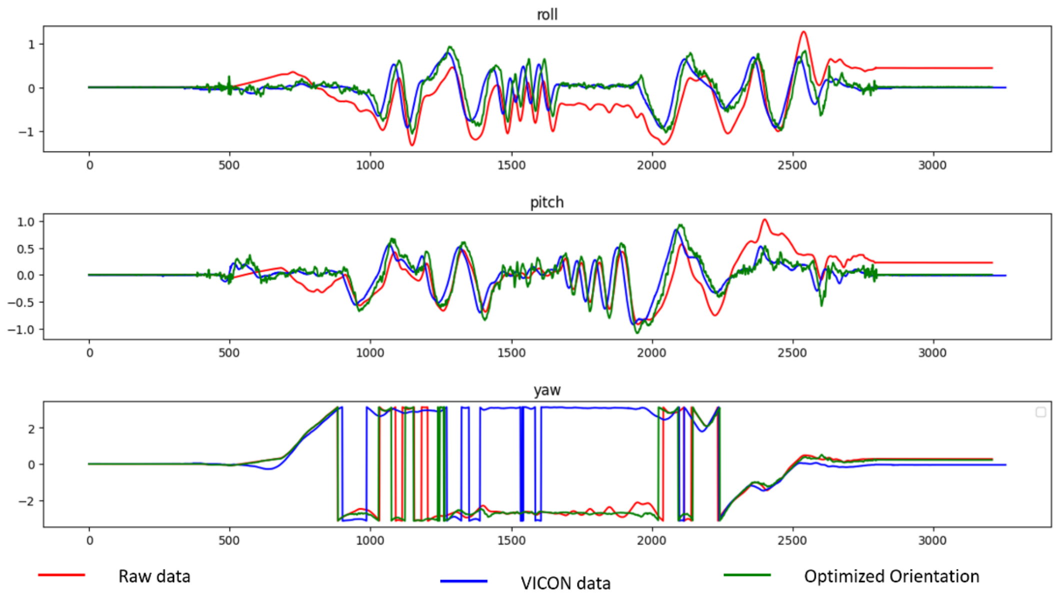

Figure 7: Orientation of Dataset 5(train set)

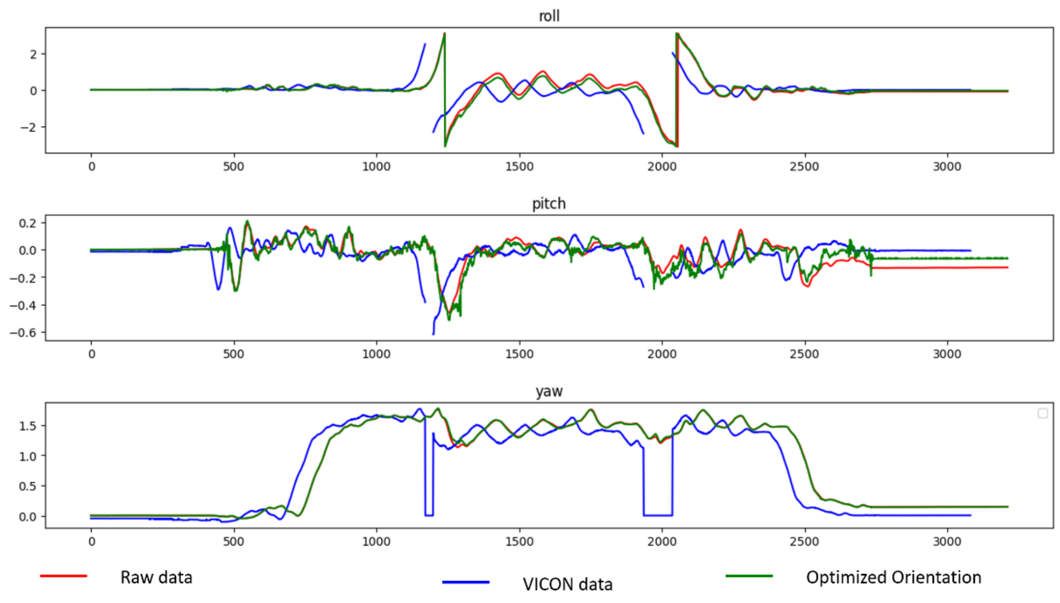

Figure 8: Orientation of Dataset 6(train set)

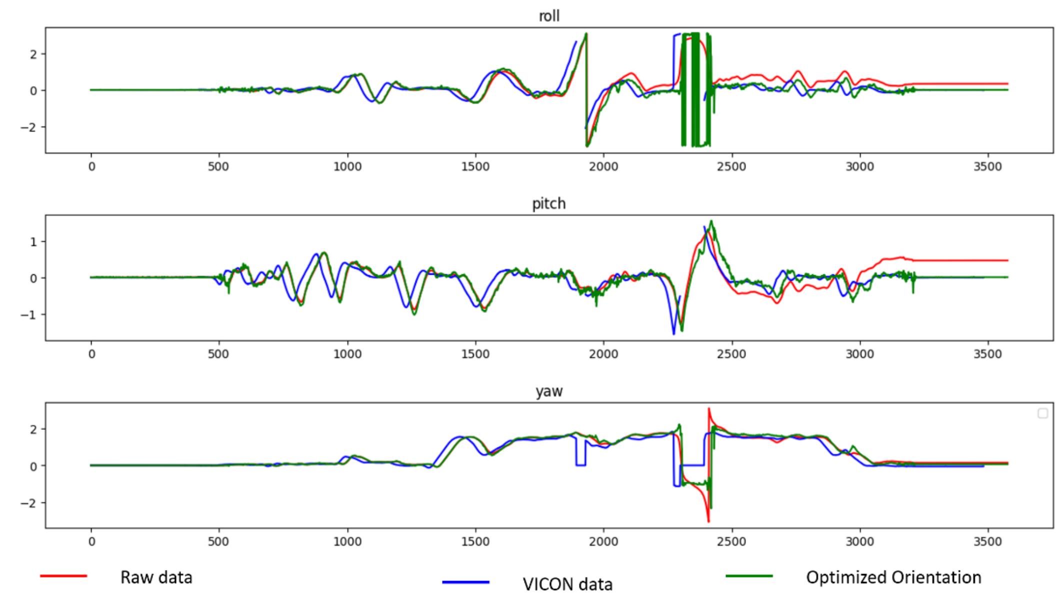

Figure 9: Orientation of Dataset 7(train set)

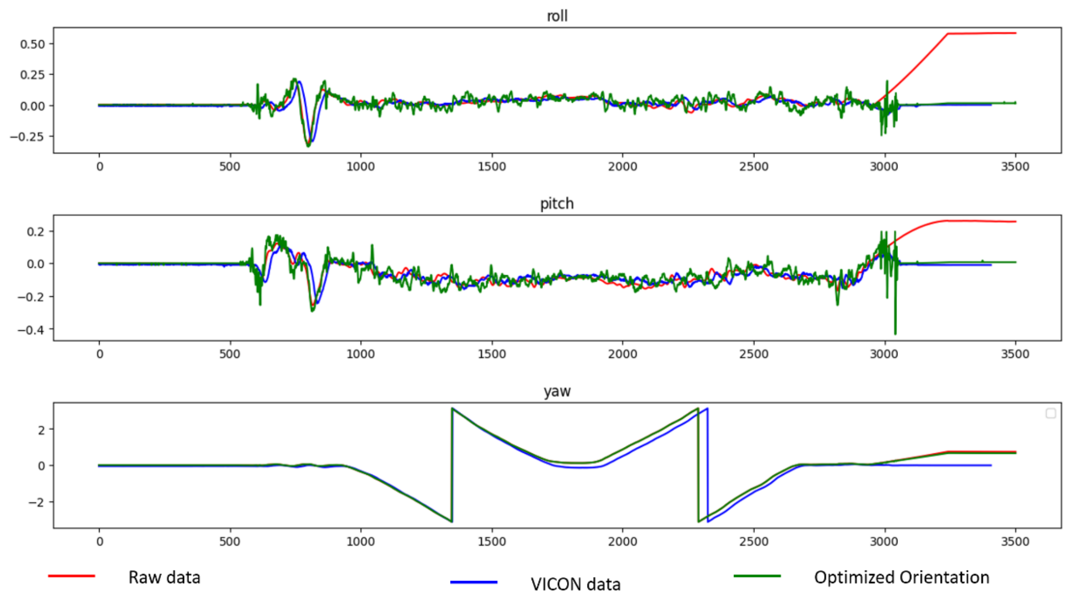

Figure 10: Orientation of Dataset 8(train set)

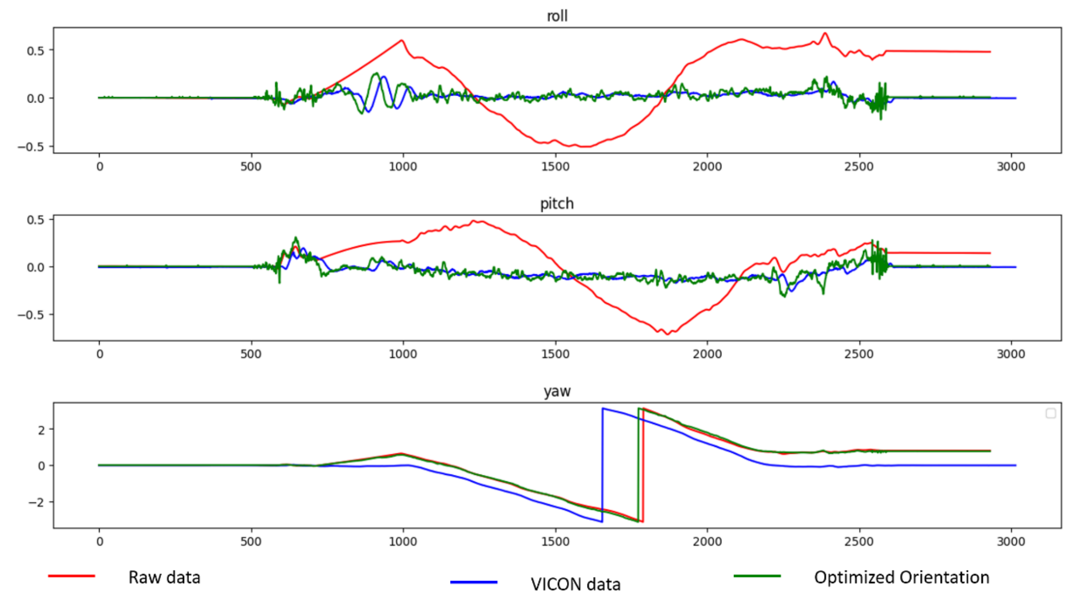

Figure 11: Orientation of Dataset 9(train set)

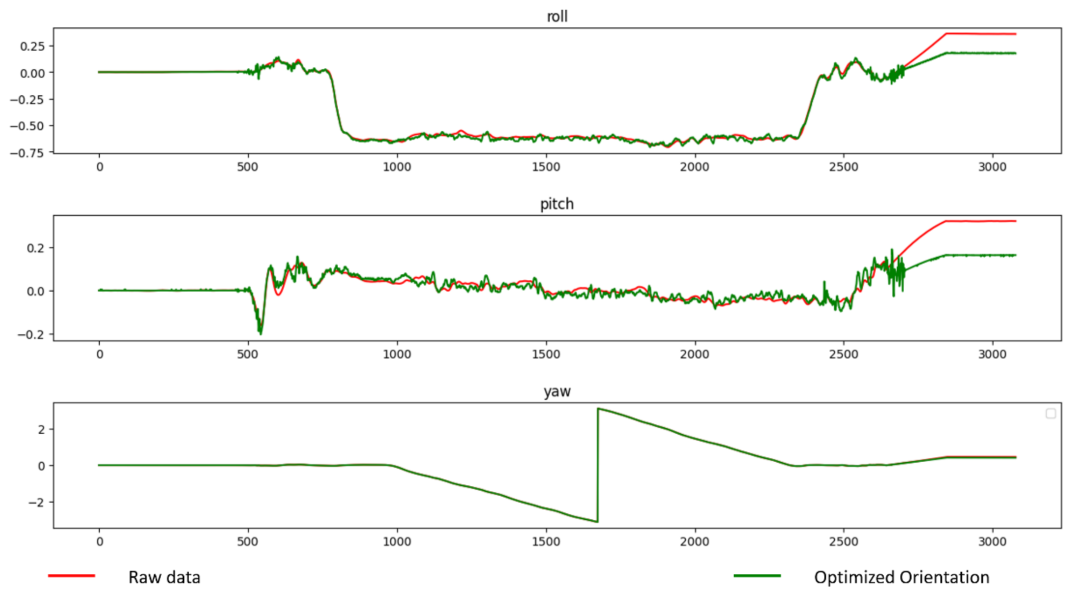

Figure 12: Orientation of Dataset 10(test set)

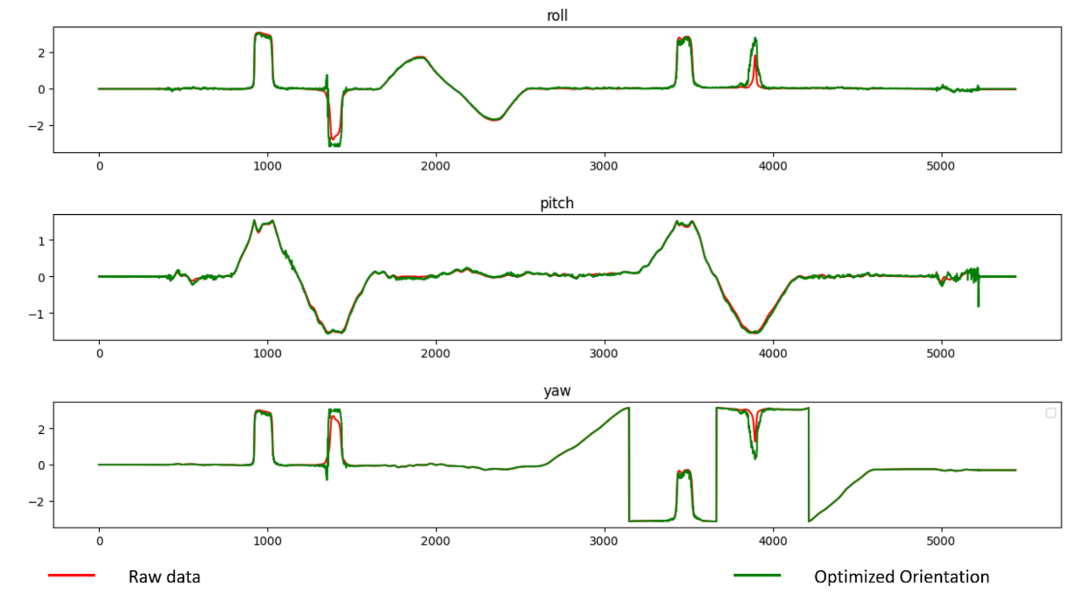

Figure 13: Orientation of Dataset 11(test set)

B. Panorama reconstruction

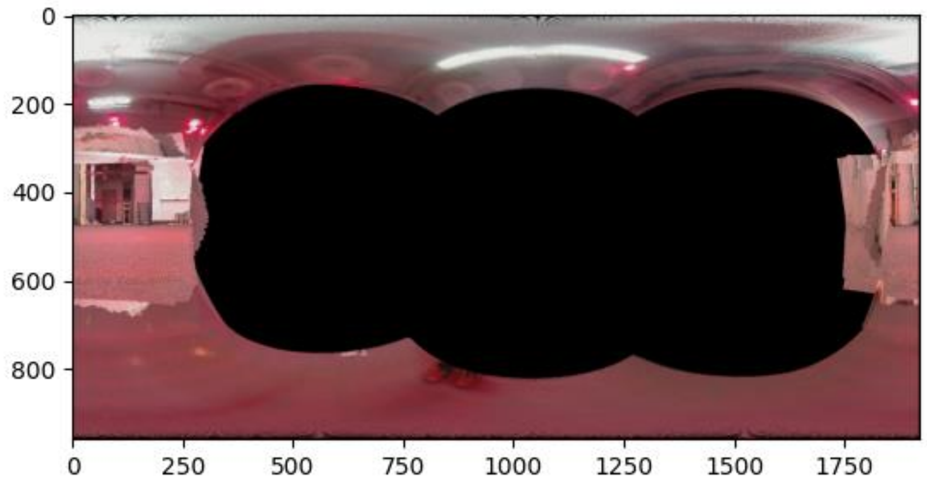

Figure 14: Panorama photo reconstructed from Dataset 1(train set)

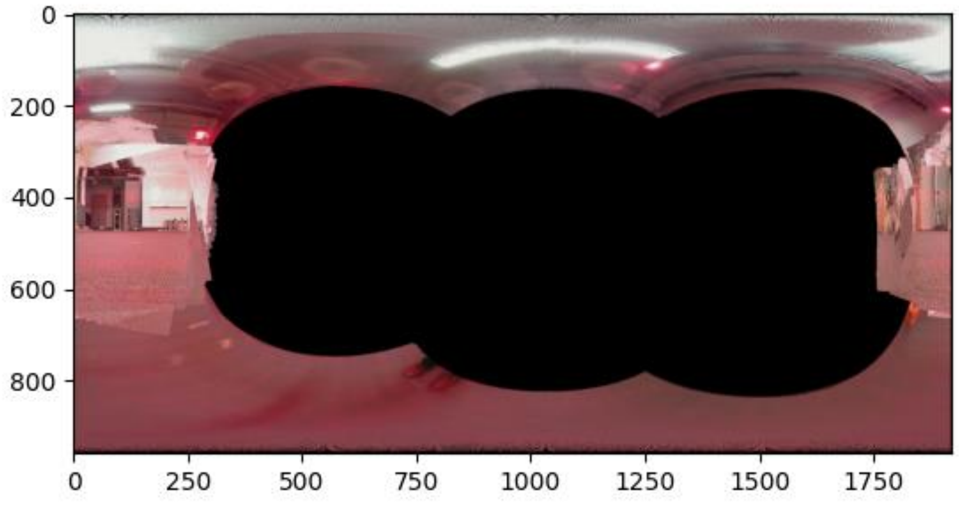

Figure 15: Panorama photo reconstructed from Dataset 2(train set)

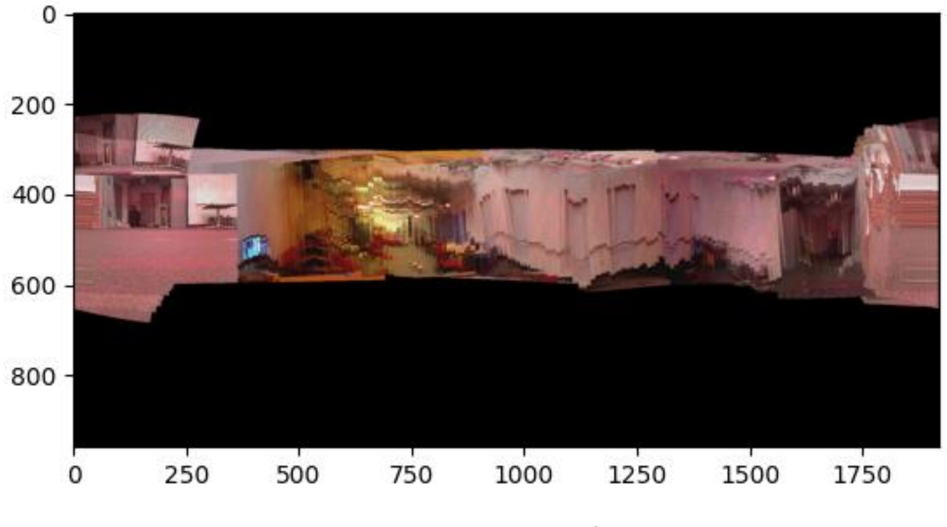

Figure 16: Panorama photo reconstructed from Dataset 8(train set)

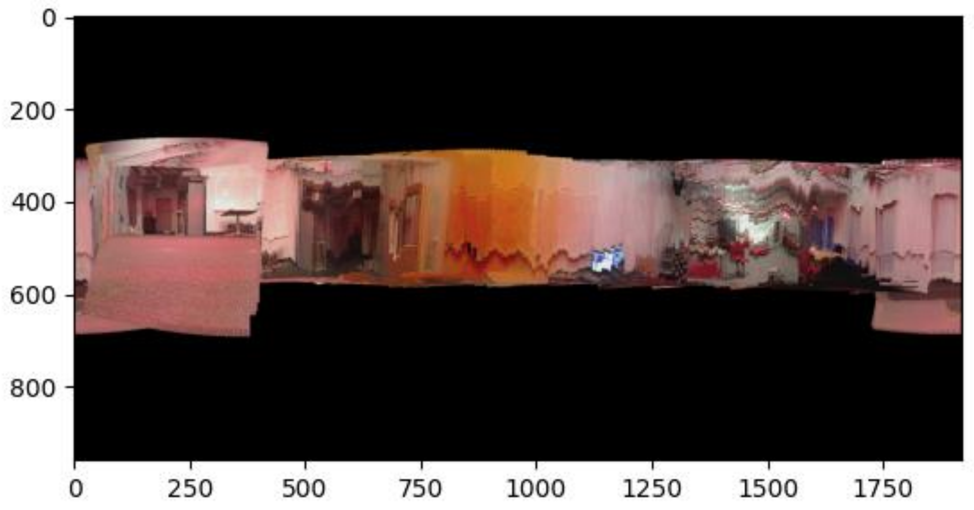

Figure 17: Panorama photo reconstructed from Dataset 9(train set)

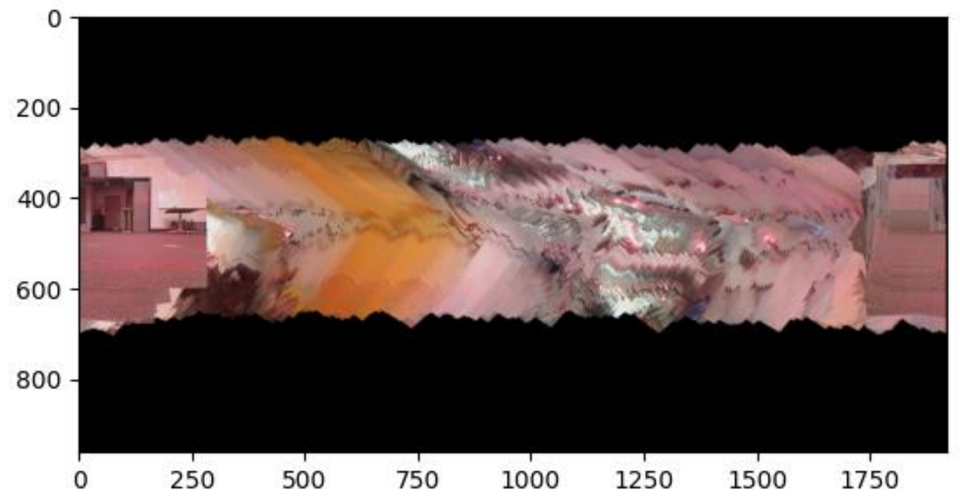

Figure 18: Panorama photo reconstructed from Dataset 10(test set)

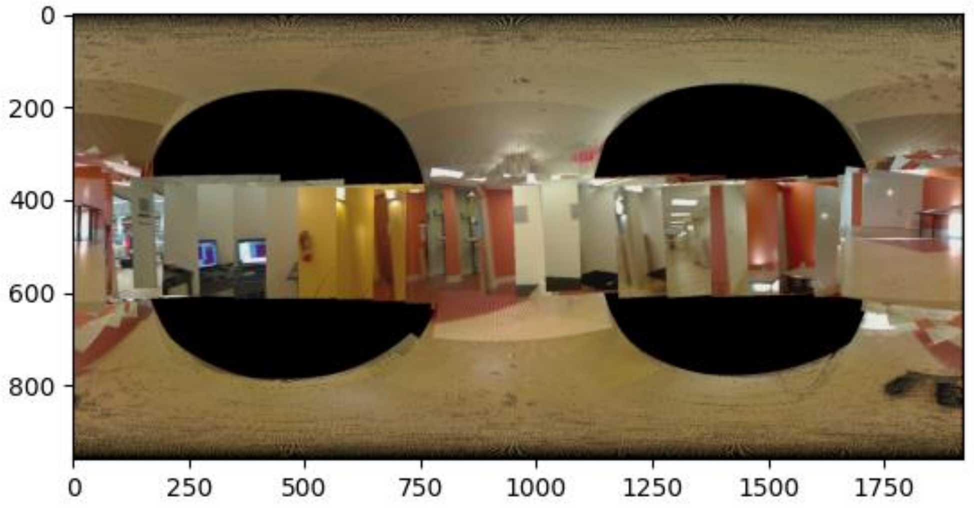

Figure 19: Panorama photo reconstructed from Dataset 11(test set)

V. Evaluation

A. Orientation Tracking

Because VICON and IMU have different time stamps, yet we are forcing them to be plotted in the same figure with same start point, we mainly care about the shape of the orientation curve, instead of phase or location. In other words, horizontal differences between estimated curve and ground truth curve are acceptable.

In **Fig 3- Fig 13**, we can see that after 100~300 iterations, compared with original estimated orientation (raw data), the optimized orientation angles are apparently closer to the ground truth angles (VICON). During the optimization, we found that step size plays an interesting role in gradient descent. Too big step size leads to larger error, or may even cause the cost function to diverge. On the contrary, too small step size leads to longer time to converge. After a few experiments, we finally chose 5e-2 as the step size, which balance the accuracy and efficiency. In a typical projected gradient descent process, the cost function's change is shown in **Fig 20**.

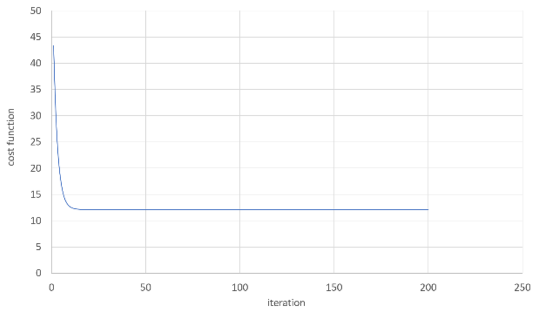

Figure 20: cost function's change with iterations

Observing the optimized result from train sets, we found that this optimization algorithm is good at eliminating large and rapid errors (**Fig 9**) and is capable of eliminating cumulative errors. However, when the ground truth data varies at a high frequency with small range, this gradient descent model needs much more iterations to adjust the estimated orientations.

B. Panorama reconstruction

For panorama reconstruction, a very important part is choosing proper coordinates to draw the final panorama photo. To avoid distortion and deformation, we chose the cylinder coordinate based on the camera's viewing angle.

In **Fig 14- Fig 17**, we can see that the photos from camera data are stitched properly, construction a continuous panorama photo.

In **Fig 18**, however, the panorama photo is vague. This may be the result of rotating the camera at high velocity.

In **Fig 18**, the problem is that the panorama photo has many tears. Decrease the rotation velocity or increase the overall sampling time may be helpful.

References

[1] Ahmad, N., Ghazilla, R. A. R., Khairi, N. M., & Kasi, V. (2013). Reviews on various inertial measurement unit (IMU) sensor applications. International Journal of Signal Processing Systems, 1(2), 256-262.

[2] Tanenhaus, M., Carhoun, D., Geis, T., Wan, E., & Holland, A. (2012, April). Miniature IMU/INS with optimally fused low drift MEMS gyro and accelerometers for applications in GPS-denied environments. In Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium (pp. 259-264). IEEE.

[3] Brossard, M., & Bonnabel, S. (2019, May). Learning wheel odometry and IMU errors for localization. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 291-297). IEEE.

[4] Madgwick, S. O., Harrison, A. J., & Vaidyanathan, R. (2011, June). Estimation of IMU and MARG orientation using a gradient descent algorithm. In 2011 IEEE international conference on rehabilitation robotics (pp. 1-7). IEEE.

[5] Fleps, M., Mair, E., Ruepp, O., Suppa, M., & Burschka, D. (2011, September). Optimization based IMU camera calibration. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3297-3304). IEEE.

[6] Kim, S. U., Lee, J., Yoon, J., Ko, S. K., & Kim, J. (2021). Robust methods for estimating the orientation and position of IMU and MARG sensors. Electronics Letters, 57(21), 816-818.

[7] Schwartz, A., & Polak, E. (1997). Family of projected descent methods for optimization problems with simple bounds. Journal of Optimization Theory and Applications, 92, 1-31