Project 1: Orientation Tracking

Justin Volheim
Electrical and Computer Engineering
Department UCSD
jvolheim@ucsd.edu

Abstract

This document focuses on Orientation tracking based on IMU measurements with optimization of quaternions and gradient decent. It will go on to show how accurate these tracking estimates are by creating a panoramic image using images taken from the IMU body frame while rotating. The method discussed in this project can be adapted for environment mapping and live orientation estimations of a moving ridged body.

Introduction

In this project we were tasked with solving the problem of accurately estimating the orientation of a rotating body throughout its movement using IMU measurements and to utilize these estimated orientations to create a panoramic image from the onboard camera. The task of orientation estimation is utilized throughout countless hardware applications from robotics to VR headsets and even has key elements used in software game design. The panoramic image construction part of this project has various real-world applications in mapping but also serves as a great visualization for how well our orientation estimation preforms. The approach used to solve this problem utilizes various quaternion-based equations followed by gradient decent for optimization of the overall orientation tracking. The panorama is constructed utilizing various coordinate transformations from the image pixel to a world frame location.

Problem Formulation

The problem of making a panoramic image based on IMU data can be broken up into two distinct parts as follows.

Orientation Estimation: In this task the main goal is to estimate a set of quaternion values that best represent the ridged body rotations based on the IMU data. As the IMU data comes in the form [[Ax,Ay,Az,Wz,Wx,Wy], [[…]]…., N samples] with corresponding time stamps we want to use equations and methods that incorporate all 6 coordinates and optimize to get the best results. These three known functions will be utilized to accomplish this task and explained in the next section.

$$\mathbf{q}_{t+1} = f(\mathbf{q}_t, \tau_t \boldsymbol{\omega}_t) := \mathbf{q}_t \circ \exp([0, \tau_t \boldsymbol{\omega}_t / 2]).$$

$$\mathbf{a}_t = h(\mathbf{q}_t) := \mathbf{q}_t^{-1} \circ [0, 0, 0, -g] \circ \mathbf{q}_t.$$

$$c(\mathbf{q}_{1:T}) := \frac{1}{2} \sum_{t=0}^{T-1} \|2 \log(\mathbf{q}_{t+1}^{-1} \circ f(\mathbf{q}_t, \tau_t \boldsymbol{\omega}_t))\|_2^2 + \frac{1}{2} \sum_{t=1}^{T} \|\mathbf{a}_t - h(\mathbf{q}_t)\|_2^2,$$

The first equation utilizes the angular accelerations to predict the next quaternion location given a current quaternion location. The second uses quaternions to estimate linear acceleration and the third equation combines the two functions into a cost function. Note this problem is designed for a backwards looking model that optimizes based on the previous datapoints to predict the orientation over a period of movement.

Panoramic construction: In this task the goal is to utilize a set of images which have been taken from the ridged body while it rotates and stitch them together accurately to give a panoramic view of the surrounding environment. This task requires accurate orientation estimations as it will rely on them for relating the image frame to the world frame. The camera data is of the form [[240,320,3],….,k images] with associated time stamps. Note the number of images is less than the number of IMU datapoints so the data values are not 1 to 1. A good way to visualize this task would be to take a bunch of pictures standing in one place while rotating around, then printing them out, laying them flat on a table and solving how they all connect. However, in our case we won't be using how similar the images are at there edges to map them together but rather the orientation at which the image was taken.

Technical Approach

Orientation Estimation: For the orientation estimation we will be utilizing several mathematical equations and methods to accurately estimate the rotation of the ridged body from the IMU data. The first such equation utilizes the angular acceleration parts of the data i.e. [Wz,Wx,Wy] denoted by ω at time stamp t to estimate the quaternion for the next time step given the current quaternion location.

$$\mathbf{q}_{t+1} = f(\mathbf{q}_t, \tau_t \boldsymbol{\omega}_t) := \mathbf{q}_t \circ \exp([0, \tau_t \boldsymbol{\omega}_t / 2]).$$

Note: exp(*) is quaternion exponential and τ is the difference in timesteps.

This equation is used to estimate the quaternion at each time step by starting with $q_0 = [1,0,0,0]$ and looping through all the angular acceleration data. The second equation shown here allows us to convert quaternion values into linear acceleration a_t.

$$\mathbf{a}_t = h(\mathbf{q}_t) := \mathbf{q}_t^{-1} \circ [0, 0, 0, -g] \circ \mathbf{q}_t.$$

This equation is used by feeding the values from eq 1 into it to produce estimates of the a_t values. The third and fourth equations shown here are cost and minimization equations that combine the estimates from both equation 1 and 2 in an equally waited manner. Note: since our data is in units of gravity the g in the above equation is 1.

$$c(\mathbf{q}_{1:T}) := \frac{1}{2} \sum_{t=0}^{T-1} \| 2 \log(\mathbf{q}_{t+1}^{-1} \circ f(\mathbf{q}_t, \tau_t \boldsymbol{\omega}_t)) \|_2^2 + \frac{1}{2} \sum_{t=1}^{T} \| \mathbf{a}_t - h(\mathbf{q}_t) \|_2^2,$$

$$\min_{\mathbf{q}_{1:T}} \; c(\mathbf{q}_{1:T}) \quad \text{s.t.} \quad \|\mathbf{q}_t\|_2 = 1, \qquad \forall t \in \{1, 2, \dots T\}$$

As this is a cost function by finding its minimum value, we will also find the quaternion values at which we achieve the best orientational estimate. Naturally as we are looking for the minimum of an equation, we will use gradient decent with respect to the quaternions. The gradient decent update formula is as follows.

$$q = \frac{q_0 - \propto * \, grad(c)}{||q_0||}$$

With q_0 being value of the previous update step and $\propto$ being the learning rate. I started with $\propto = 0.1$ and decreased it conditionally for the best results. Using this set of optimized quaternions, we will transform them into rotation matrices for ease of use when creating the panoramic image.

Challenges encountered: For this task the main challenge is computational optimization. When first writing the code it took over 3min to calculate the cost value with an unknown amount of time to compute the gradient. To solve this issue, I vectorized all the functions and utilized the jax.jit library to precompile commonly used functions. After this long process the result was a training function that runs to completion with over 50 epochs in around 15 seconds.

Panoramic construction: The process to construct a panoramic image from our data requires first that you have accurate orientation estimates i.e., the results from the previous section.

Step 1: With these estimated rotation matrixes, the first step is to associate each image with a specific rotation matrix based on the unix-timestamp of each set. Specifically, I looped through the images and rotation matrices storing the matrix with the largest timestamp that was less than or equal to the timestamp of the image.

Step 2: In this step we convert each image row/column value to a corresponding phi, theta spherical coordinate using these equations.

$$phi = -\frac{pi}{8} + \frac{pi}{4}\left(\frac{row}{imageheight - 1}\right)$$

$$theta = -\frac{pi}{6} + \frac{pi}{3}\left(\frac{column}{imagewidth - 1}\right)$$

Step 3: For this step we first transform the above phi theta values to Cartesian coordinates using the standard conversion equations.

Step 4: Now that we have the data in the cartesian coordinates of the body from we need to rotate these coordinates to the world from using the following equation with S_w being the point in the world frame, R being the rotation matrix, S_b being the point in the body from and p being the positional offset which in our case p = [0,0,0,0.1].

$$S_w = R * S_b + p$$

Step 5: In this step we will being going in reverse through the previous steps to convert the above world frame points to the new image frame. Therefore, in this step we will convert the cartesian points S_w back to spherical points.

Step 6: In this step we convert the spherical points back to image row/column values using the following equations.

$$column = \frac{w-1}{2 * pi} * theta$$

$$row = \frac{h-1}{pi} * (phi + \frac{pi}{2})$$

Note: (w = panoramic image width) and (h = panoramic image height)

Step 7: Now that we have all the pixel locations, we will assign the pixel values of our original image to the calculated locations in the panoramic image.

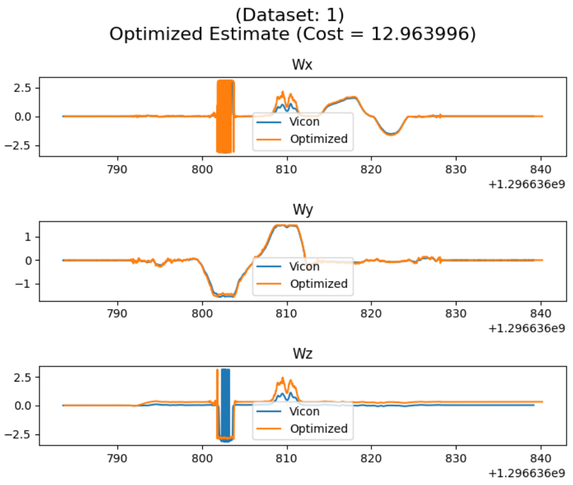
Step 8: Loop over the steps 4-7 for each of the rotation matrices completing your panoramic mapping of the images.

Task Challenges: The challenges for this section were twofold. The first being code optimization again as the data objects for this task were no longer matrixes but rather tensors. The second challenge was making sure that all the transformations between coordinate types and spaces aligned properly as a simple mistake could alter the result for some datasets while possibly not affecting the others.

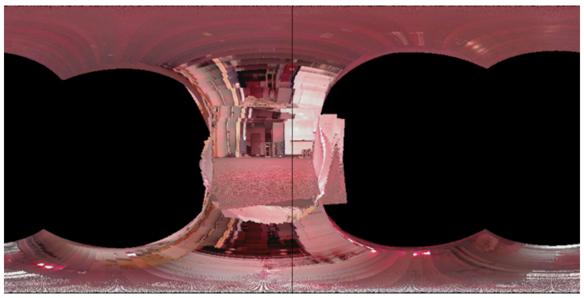
Results

In this section we will be going through each dataset iteratively displaying and discussing the results.

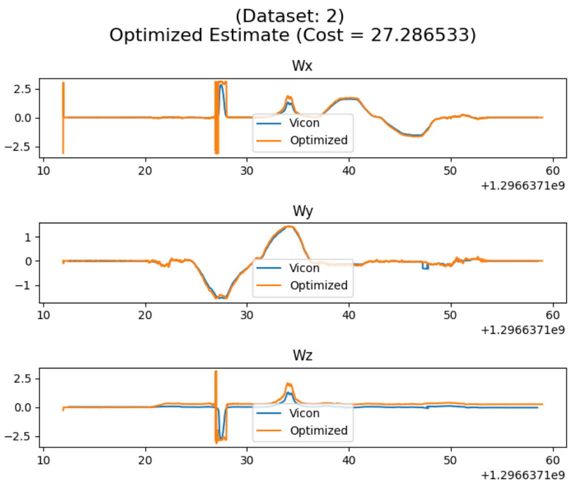
Dataset 1: For this dataset we were able to accurately predict the ideal Euler rotations for Wx and Wy with a slight difference when modeling Wz.



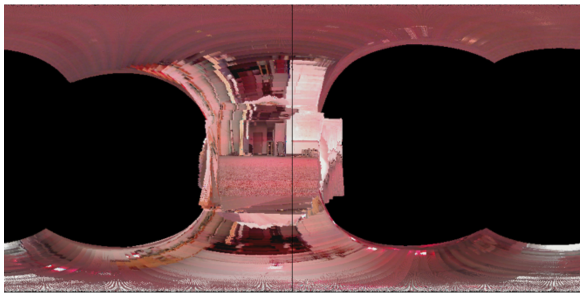
The Panoramic image representation of the above optimized model broke down due to the rotation becoming gimbal locked. This occurred when the Wy reached 90 degrees.



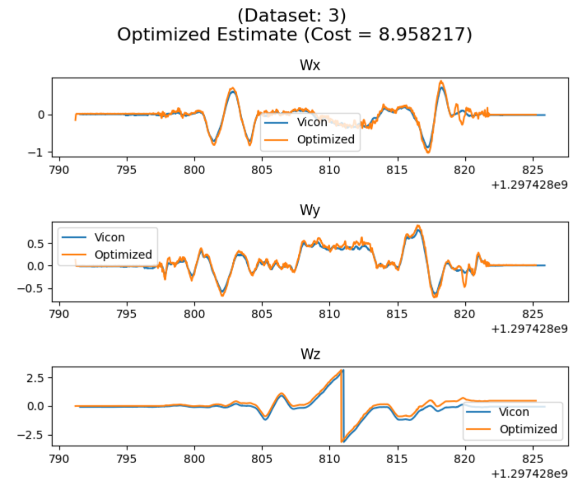
Dataset 2: For this dataset we accurately modeled the angular accelerations for all the rotation axes.



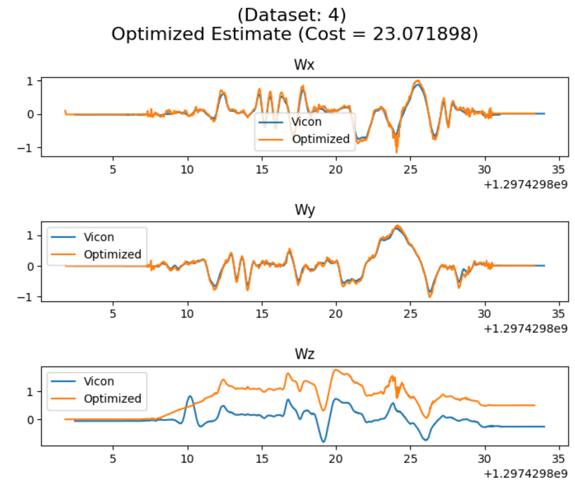
This image also becomes locked similar to dataset 1.



Dataset 3: This dataset seems to give the most accurate rotation estimates only deviating slightly in the Wz towards the end of the dataset.

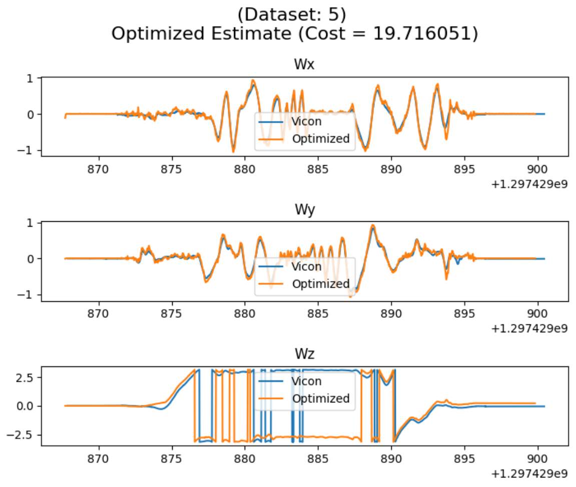


Dataset 4*:* The method used on this dataset worked well in both the Wx and Wy direction but failed to accurately predict the Wz value. This is likely due to a reset pin being triggered during rotation and throwing off the estimation.

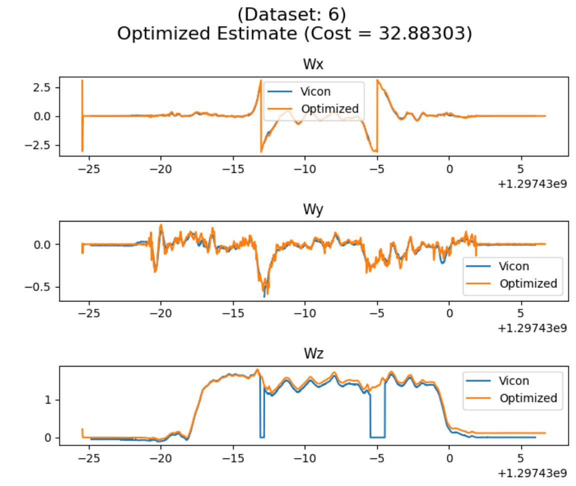


Dataset 5: For this dataset our model did

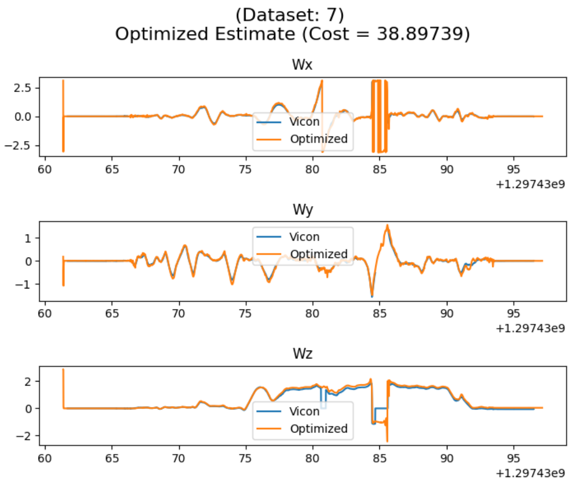
handle the large amount of variation in the Wx, Wy and Wz direction with the reason that it looks off on the data being that the top of the graph is equal to the bottom due to it being a rotation.



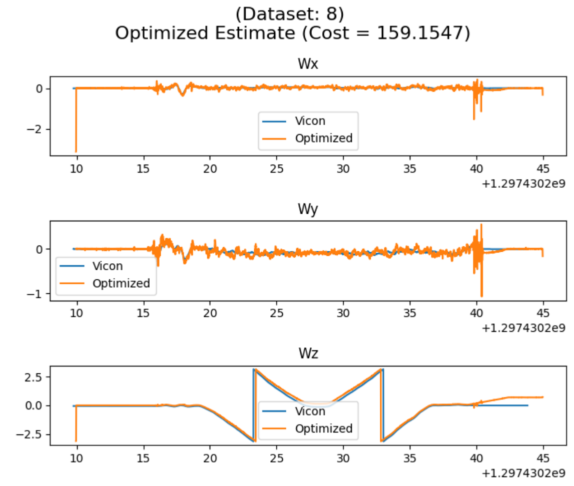
Dataset 6: This dataset handles the estimation well in all, but the Wz direction as it was not able to accurately account for the sharp dips in acceleration.



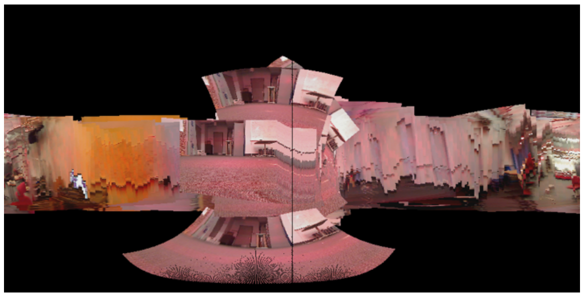
Dataset 7: This similar to the other datasets the method had trouble estimating the Wz values but in general it handles the dataset well.



Dataset 8: This dataset look visually very close along the angular accelerations however has a high-cost value due to the static in the measurements and estimations.

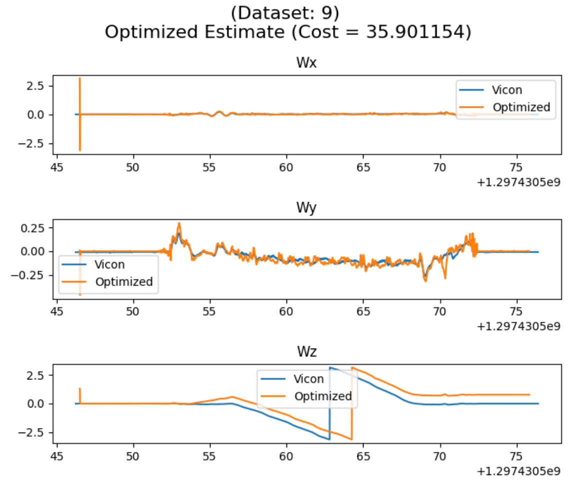


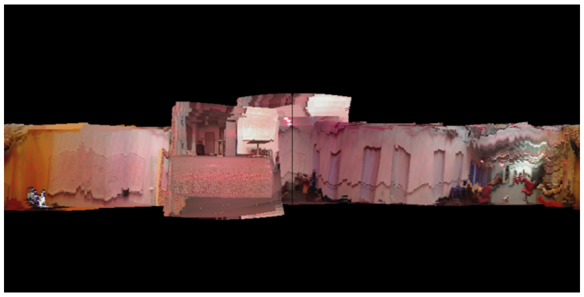
The constructed panoramic image was reconstructed well despite the high cost in our optimization.



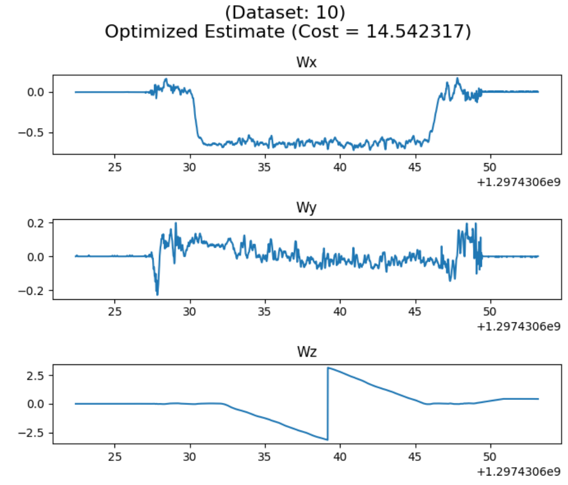
Dataset 9: This dataset had almost no rotation in the Wx or Wy and was off in the Wz however the resultant image turned out

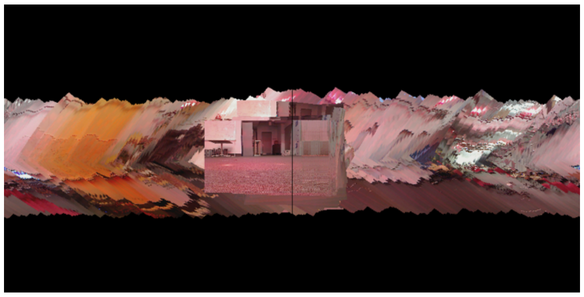
well and constructed a sufficiently clean panoramic image.



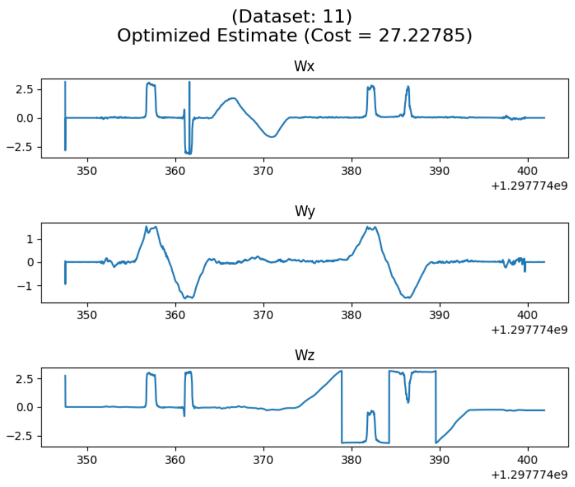


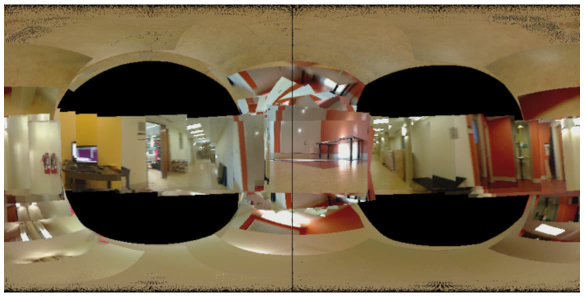
Dataset 10: For the test set show here the image did not come out as expected this is likely due to an inaccuracy in predicting the Wx variable as it looks like the estimates are just slightly off in there Wx rotations consistently.





Dataset 11: For this dataset we can see from the image that we have accurately mapped how the image rotates first in the Ay direction then rotates in the Az before rotating in the Ay again. We can also see that it accurately handles the Wx rotation.





Conclusion

The orientation tracking and panorama reconstruction approach presented in this project demonstrates the capability of accurately estimating the orientation of a rotating body using IMU measurements and generating panoramic images from camera images. Although the model didn’t estimate the yaw angle as precisely as the others and the images are not perfectly precise with further refinement and optimization of the algorithms this can be resolved. For these reasons this method of orientation tracking and panoramic image construction are effective choices for future projects and applications.