Orientation Tracking and Image Reconstruction

Charles Lee
Electrical and Computer Engineering
La Jolla, CA
CharLee@ucsd.edu

I. Introduction

In this project, we aim to estimate the 3D orientation of a rotating body over time. We also use our 3D orientation estimates with images collected by a camera on the rotating body to construct a panorama. 3D orientation estimation is an important problem to solve since the localization of a robot in space allows for more information for planning a robot's next move and combining it with information collected by other sensors. Constructing a panorama is also important because mapping the space the robot is in allows the robot to understand information about the space it exists in, so it can better plan its next move. We estimate 3D orientation by using readings from an inertial measurement unit (IMU) and minimizing the error of our orientation estimates given by a motion model and observation model. We construct the panorama by using images captured by the camera and rotating it into the correct place using our 3D orientation estimates.

II. Problem Formulation

A. Orientation Tracking

Given IMU angular velocity $\boldsymbol{\omega}_t$ and linear acceleration $\mathbf{a}_t$ measurements on a rotating body, estimate the orientation of the body over time, denoted by $\mathbf{q}_t \in \mathbb{H}_*$

Let the time differences between consecutive time stamps be τ_t.

Using the motion model

$$\mathbf{q}_{t+1} = f(\mathbf{q}_t, \tau_t, \omega_t) \coloneqq \mathbf{q}_t \circ \exp([0, \tau_t \omega_t / 2]), \tag{1}$$

and the observation model

$$\mathbf{a}_t = h(\mathbf{q}_t) \coloneqq \mathbf{q}_t^{-1} \circ [0, 0, 0, -g] \circ \mathbf{q}_t \tag{2}$$

we define the error of our orientation estimates as

$$\begin{aligned} c(\mathbf{q}_{1:T}) = \frac{1}{2} \sum_{t=0}^{T-1} \| 2 \log(\mathbf{q}_{t+1}^{-1} \circ f(\mathbf{q}_t, \tau_t, \boldsymbol{\omega}_t)) \|_2^2 + \\ \frac{1}{2} \sum_{t=1}^{T} \| \mathbf{a}_t - h(\mathbf{q}_t) \|_2^2 \end{aligned} \tag{3}$$

where $\mathbf{q}_{1:T} \coloneqq \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_T$ is the orientation trajectory of the rotating body over time.

$$\min_{\mathbf{q}_{1:T}} c(\mathbf{q}_{1:T}) : \|\mathbf{q}_t\|_2 = 1, \forall t \in \{1, 2, \ldots, T\} \tag{4}$$

B. Panorama Reconstruction

Given body orientation $\mathbf{q}_{1:T}$ and RGB camera images $\mathbf{I}_{1:T}$, devise a function $\mathbf{P}(\mathbf{q}_{1:T}, \mathbf{I}_{1:T})$ which maps the images onto a singular panoramic image that represents a 360° view of the rotating body's environment.

III. Technical Approach

A. Orientation Tracking

a) Calibration: We first calibrate the IMU sensor data by converting its raw ADC values to physical units. The sensor data is formatted as $(\mathrm{a}_x, \mathrm{a}_y, \mathrm{a}_z, \omega_z, \omega_x, \omega_y)$. The data follow the conventions seen in Figure 1. We convert the raw units into g for the accelerations and $\frac{rad}{s}$ for the angular velocities using the following equation:

$$\begin{aligned} \text{value} &= (\text{raw} - \text{bias}) * \text{scale\_factor} \\ \text{scale\_factor} &= \mathrm{V}_{\text{ref}} / 1023 / \text{sensitivity} \end{aligned} \tag{5}$$

where V_{ref} is given to be 3.3V and sensitivity can be read off of the IMU datasheet. We calculate the bias by averaging the first several seconds of the raw data to calibrate. This is defined as

$$\text{bias} = \frac{1}{K} \sum_{t=0}^{K} \frac{\text{raw}_t - \text{expected\_value}}{\text{scale\_factor}} \tag{6}$$

where K is the number of seconds of data we want to use as a baseline. For this project, we used $K = 2$. The expected values are chosen to be

$$\begin{pmatrix} \mathrm{a}_x \\ \mathrm{a}_y \\ \mathrm{a}_z \\ \omega_z \\ \omega_x \\ \omega_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{7}$$

because, aside from gravity in the z direction, there should be no acceleration or angular velocity during the first few K seconds of calibration.

b) Initial Estimation: We now make our initial estimation on the orientation trajectory of the rotating body, $\mathbf{q}_{1:T}^{(0)}$. We use the motion model (1) along with initial estimate $\mathbf{q}_0 = [1, 0, 0, 0]$, denoting the identity orientation, to compute the values $\mathbf{q}_{1:T}^{(0)}$.

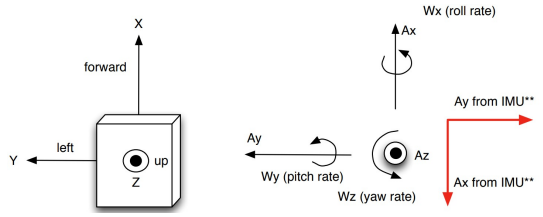


Fig. 1. Body Reference Frame *(provided in project documentation)*

c) Optimizing Estimates: Once we have our initial estimates $\mathbf{q}_{1:T}^{(0)}$, we minimize the error (3) by solving the minimization problem given by (4). This constrained minimization problem can be solved by using a projected gradient descent algorithm,

$$\hat{\mathbf{q}}_{1:T}^{(k+1)} = \mathbf{q}_{1:T}^{(k)} - \alpha \nabla c(\mathbf{q}_{1:T}^{(k)}) \tag{8}$$

$$\mathbf{q}_{1:T}^{(k+1)} = \Pi(\hat{\mathbf{q}}_{1:T}^{(k+1)}) \tag{9}$$

$$\Pi(\mathbf{q}) = \frac{\mathbf{q}}{\|\mathbf{q}\|_2} \tag{10}$$

We minimize the error of our estimated orientations in (8), while (9) ensures that our estimated orientations stay within the constraints $\|\mathbf{q}_t\|_2 = 1$. We continue to iterate until we converge onto an estimated orientation with locally minimal error, by this update rule:

while $\|\mathbf{q}_{1:T}^{(k)} - \mathbf{q}_{1:T}^{(k-1)}\|_2 > \epsilon$ **do**
  $\mathbf{q}_{1:T}^{(k+1)} \leftarrow \Pi(\mathbf{q}_{1:T}^{(k)} - \alpha \nabla c(\mathbf{q}_{1:T}^{(k)}))$
end while

For this project, we used $\epsilon = 10^{-4}$ for the stopping criterion and $\alpha = 10^{-3}$ for the learning rate.

B. Panorama

a) Matching Timestamps: After estimating the orientation of the body, we match the timestamps of the estimated orientations with the timestamps of the images captured by the camera. Since we have more estimated orientations over time compared to images captured over time, we align the closest-in-the-past timestamp of orientation estimates to each camera image timestamp, and discard the orientation estimates that are not matched.

b) Construction: We use the assumption that the image lies on a sphere of radius 1 meter away from the rotating body. We also use the assumption that the camera has a 60° horizontal and 45° field of view. Given these assumptions, we can map the pixel coordinates into spherical coordinates by

$$S(x, y) = (\frac{22.5}{160}x + 67.5, -\frac{1}{4}y + 30, 1) \tag{11}$$

We convert to radians by

$$S^{rad}(\phi, \theta, 1) = (\phi\frac{\pi}{180}, \theta\frac{\pi}{180}, 1) \tag{12}$$

Then we convert the spherical coordinates to cartesian coordinates by

$$C(\phi, \theta, 1) = (\sin\phi\cos\theta, \sin\phi\sin\theta, \cos\phi) \tag{13}$$

We then use our estimate orientation to rotate the coordinates in cartesian space into the correct location by

$$C_r(x, y, z, \mathbf{q}_t) = R(\mathbf{q}_t)C \tag{14}$$

After rotation, we now convert back to spherical coordinates

$$S_{rot}(x, y, z) = (\arccos(z)\frac{180}{\pi}, \arctan(\frac{y}{z})\frac{180}{\pi}, 1) \tag{15}$$

Then we project the coordinates from the sphere onto a circumscribing cylinder

$$C_{cyl}(\phi, \theta, 1) = (\cos(\phi\frac{\pi}{180}), (-\theta + 180) \mod 360) \tag{16}$$

Finally, we change the cylindrical coordinates back to image coordinates

$$P(z, \theta) = (-314z, \frac{1706}{360}\theta) + (314, 0) \tag{17}$$

We have devised the function that maps images onto a singular panoramic image

$$\mathbf{P}(\mathbf{q}_{1:T}, \mathbf{I}_{1:T}) = P(C_{cyl}(S_{rot}(C_r(C(S^{rad}(S(\mathbf{I}_{1:T})), \mathbf{q}_{1:T})))) \tag{18}$$

We stitch the images together by having images later in time overwrite images earlier in time. Although this is not the most robust method, this is sufficient for the purposes for this project.

IV. Results

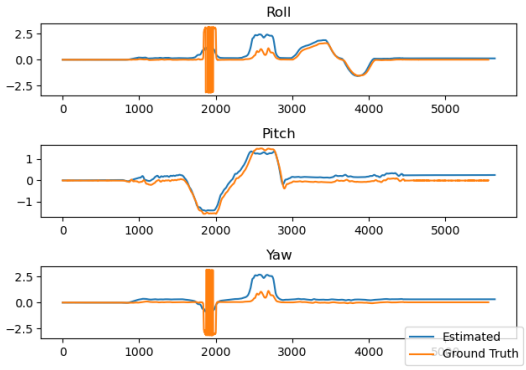
A. Orientation Tracking

a) Training Data: The following images display a graph of ground truth and estimated orientations in the form of roll, pitch, and yaw, for every dataset. The initial estimate and the optimized estimate graphs are displayed one after another for easier reference of before and after optimization. The cost of the orientation trajectory (3) is displayed in the title of the graphs. We can see that over all training datasets, the cost is decreased after the projected gradient descent algorithm is run. An interesting takeaway is that all of the training datasets costs decreased after the optimization algorithm was run. This is interesting because the projection (9) does not necessarily result in a orientation trajectory that has a lower cost that the predicted orientation trajectory in the previous iteration of the algorithm. i.e.

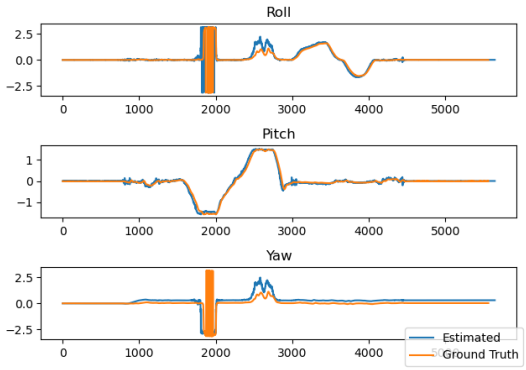
$$c(\Pi(\hat{\mathbf{q}}_{1:T}^{(k+1)})) \leq c(\mathbf{q}_{1:T}^{(k)}) \tag{19}$$

may not necessarily be true. We can also visually inspect the before and after optimization images, and we can see that the estimated orientations curves of all datasets approached the ground truth curves.

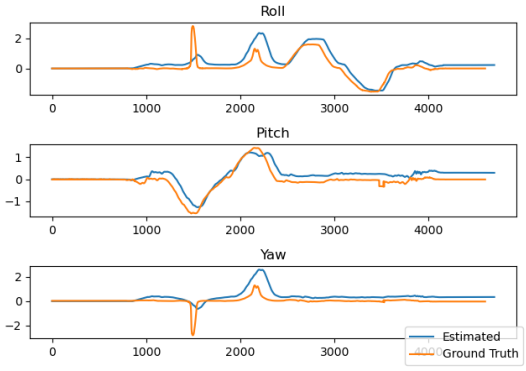
Initial Estimate in Radians: Data 1, Cost=177.63



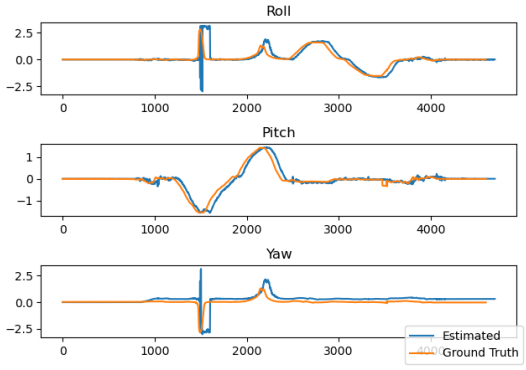
Optimized Estimate in Radians: Data 1, Cost=12.61



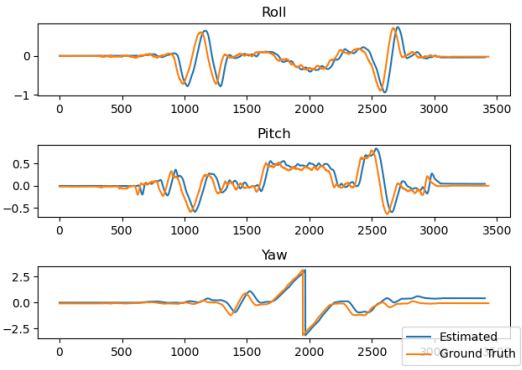
Initial Estimate in Radians: Data 2, Cost=296.00



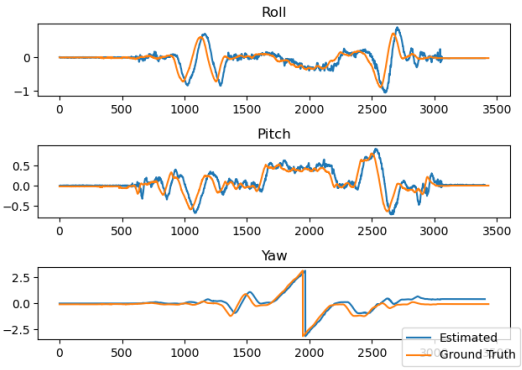
Optimized Estimate in Radians: Data 2, Cost=13.07



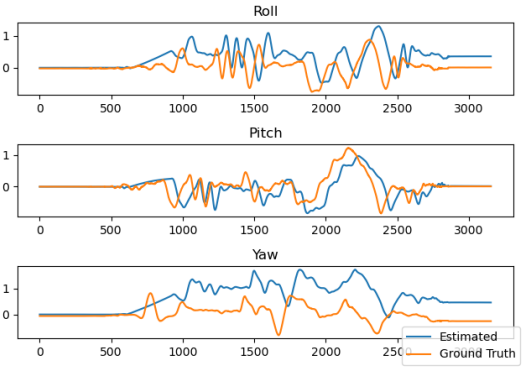
Initial Estimate in Radians: Data 3, Cost=19.01



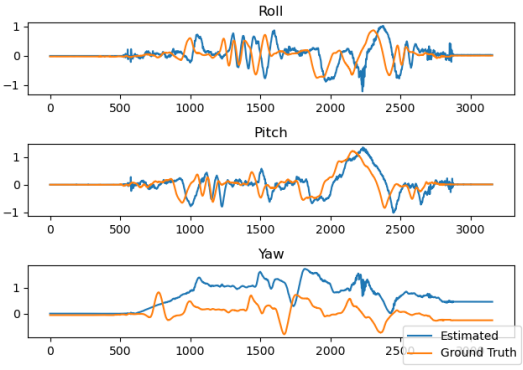
Optimized Estimate in Radians: Data 3, Cost=8.58

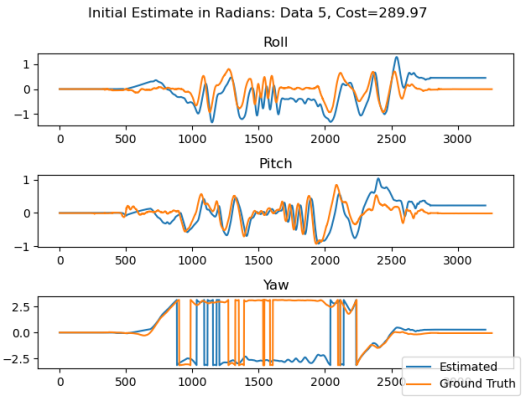


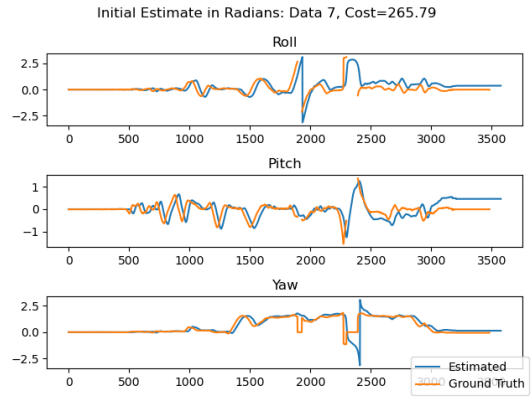
Initial Estimate in Radians: Data 4, Cost=164.65

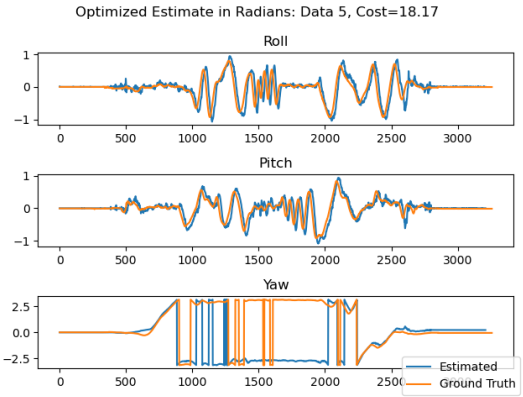


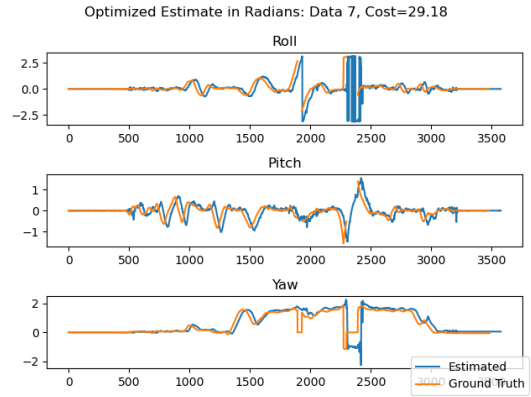
Optimized Estimate in Radians: Data 4, Cost=22.40

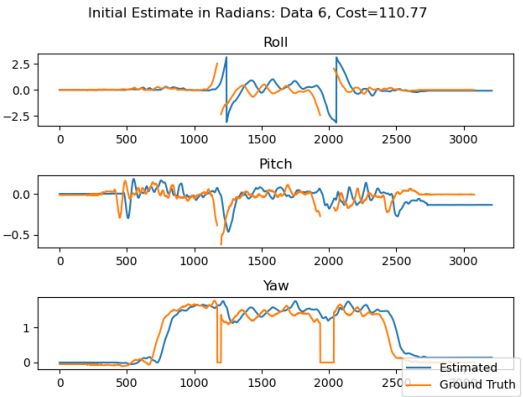


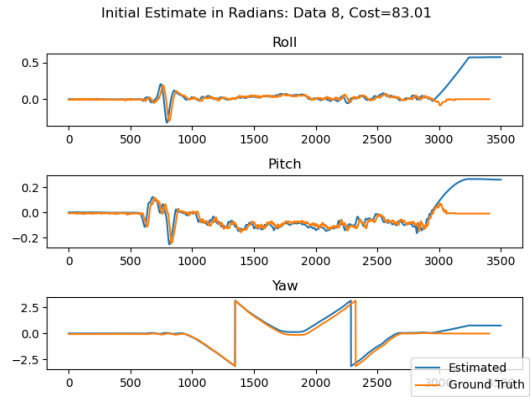

Initial Estimate in Radians: Data 5, Cost=289.97
Roll
Pitch
Yaw
Estimated
Ground Truth

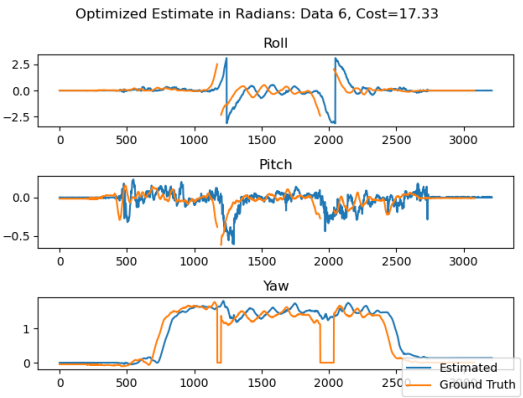

Initial Estimate in Radians: Data 7, Cost=265.79
Roll
Pitch
Yaw
Estimated
Ground Truth

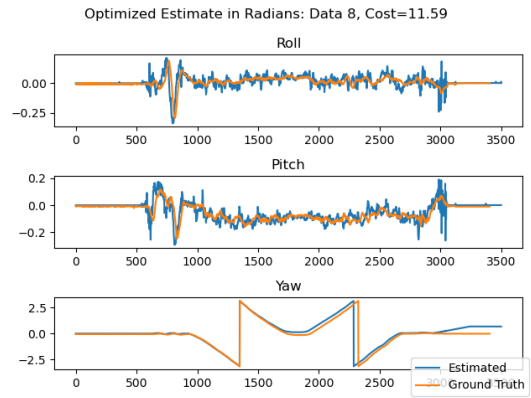

Optimized Estimate in Radians: Data 5, Cost=18.17
Roll
Pitch
Yaw
Estimated
Ground Truth

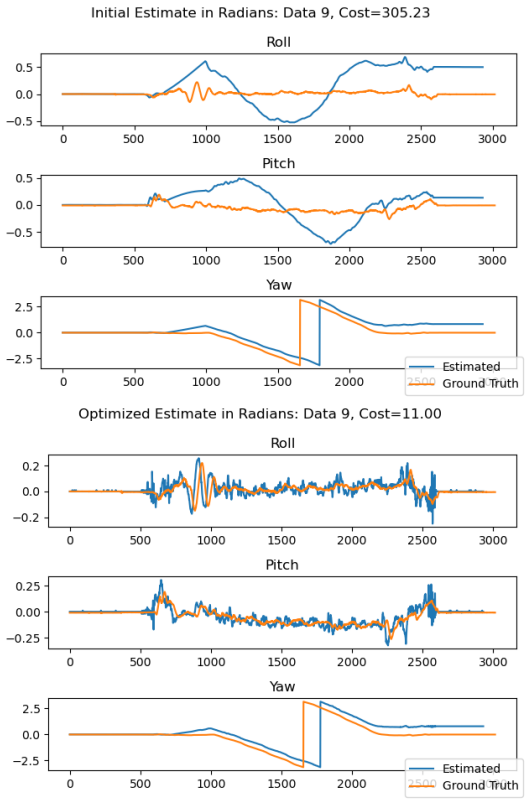

Optimized Estimate in Radians: Data 7, Cost=29.18
Roll
Pitch
Yaw
Estimated
Ground Truth


Initial Estimate in Radians: Data 6, Cost=110.77
Roll
Pitch
Yaw
Estimated
Ground Truth

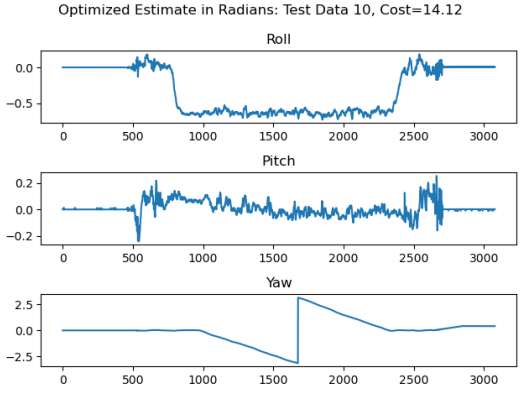

Initial Estimate in Radians: Data 8, Cost=83.01
Roll
Pitch
Yaw
Estimated
Ground Truth

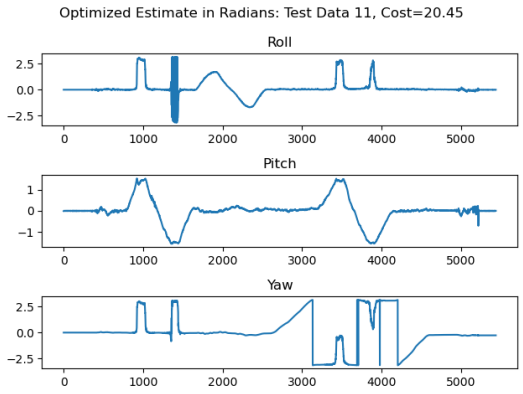

Optimized Estimate in Radians: Data 6, Cost=17.33
Roll
Pitch
Yaw
Estimated
Ground Truth


Optimized Estimate in Radians: Data 8, Cost=11.59
Roll
Pitch
Yaw
Estimated
Ground Truth



b) Testing Data: We can see that our estimated orientation optimization algorithm worked successfully because the algorithm generalized to the testing data very well. The cost of the optimized orientation trajectory in the test data is about the same as the cost values seen in the training data. As a sanity check, we can also simply look at the curves of the roll, pitch, and yaw in the testing data and see that the curves move at a reasonable value and rate of a rotating body.





B. Panorama

a) Training Data: The following images show the constructed panoramas. The images make sense for the orientations that were given by the ground truth. There are the occasional repeat images that show up, and this is due to the inaccuracies that occur in our estimation of the orientation. Also, our method of stitching up the images are not very robust, resulting in these inaccuracies.

Looking at Figure 2, we can see that the images end up in the center of the image, and distorted images at the top and bottom stretched left and right. We can match this up with the ground truth data from the plots for Data 1 earlier. The rotation is mostly only roll and pitch. This results in the camera only seeing in front, above, and below. This lines up with the panorama only having images in the center. The top and bottom of the image only gets stretched out left and right because of the projection from a sphere onto a cylinder. This is similar to our world map, where the north and south poles are distorted to seem larger than they really are. This distortion comes from the fact that we cannot unroll a globe into a flat sheet.

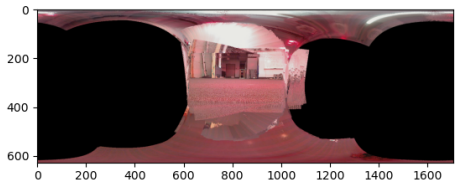


Fig. 2. Panorama: Data 1

Another image worth focusing on is Figure 4. In this panorama, we can see that the image spans the entire length. We can compare this with the ground truth plots for Data 8 from earlier. We can see that the rotation is mostly on the yaw. This results in camera only seeing around it, left and right, but

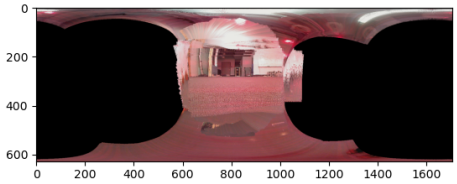


Fig. 3. Panorama: Data 2

not above or below. This matches up with the panorama, where we do not see any image data appearing at the top or bottom of the panorama, and only see images along the length. We do not see as much distortion due to the fact that when unrolling a globe onto a flat sheet, the equator is mostly preserved.

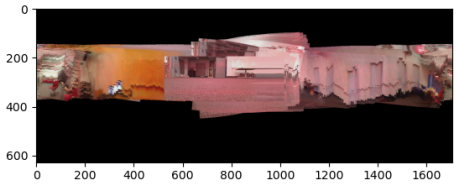


Fig. 4. Panorama: Data 8

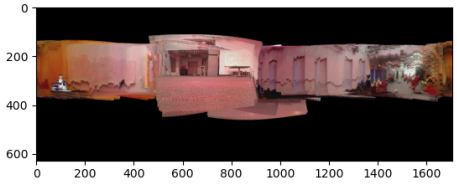


Fig. 5. Panorama: Data 9

b) Testing Data: We can see that our algorithm generalized fairly well onto the testing data. As a sanity test, simply

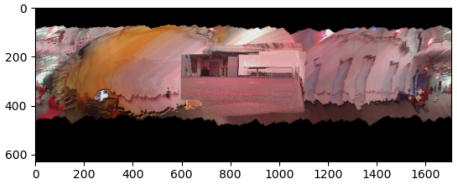


Fig. 6. Test Panorama: Data 10

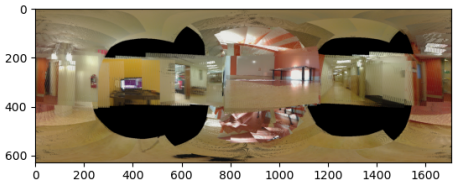


Fig. 7. Test Panorama: Data 11

observing the panorama seems fairly believable. Looking at Figure 6, the image spans the entire length. Comparing this with our orientation estimate for Data 10 from earlier, we can see that the rotation is mostly yaw. This means the camera sees left and right around it, which matches the panorama. Looking at Figure 7, the image covers almost the entire figure. This is unlike any of our other panoramas seen thus far. Comparing this with our orientation estimation for Data 11, we can see that the rotation includes a pitch and yaw. This means the camera sees above, below, in front, and left and right. This covers most of the globe space, with a few missed spots. This can be reflected in the panorama where there are a few black spots. Overall, we can see that our methodology was able to work on training data as well as test data.