IMU Based Orientation Estimation and Panorama Construction

Charvi Shukla
Computer Science and Engineering Department
University of California, San Diego
cshukla@ucsd.edu

***Abstract*—In this project, we address the problem of estimating the 3D orientation of a body rotating in-place using measurements from an Inertial Measurement Unit (IMU), and then using those estimations to build a panoramic image. We formulate this estimation as a constrained optimization problem over the set of unit quaternions. We use the angular velocity and linear acceleration measurements from the gyroscope and the accelerometer respectively. The optimal orientation trajectory is found using projected gradient descent. It is an iterative minimization algorithm that minimizes our cost function. We formulate the cost function as a combination of the kinematic motion model and a gravity based observation model. To validate the estimated trajectory and to show a potential use case of this estimation, we construct a panoramic image by stitching together RGB camera images using equirectangular projection. Our results demonstrate that roll and pitch estimates closely track VICON ground truth due to gravity corrections. However, yaw is prone to drifting. A qualitative analysis of the created panoramas shows that one can recognize boundaries and objects in the constructed panoramas, but there is stretching and distortions due to estimation noise.**

***Index Terms*—Robotics, Orientation Estimation, IMU**

I. Introduction

In recent years, there has been a sharp rise in the development of autonomous robots and vehicles. Autonomous robots are being used in industry to automate repetitive workflows (for example, moving boxes, cleaning warehouse floors etc.), and autonomous vehicles such as Waymo and Tesla have made their way into our daily lives. A critical component of such autonomous sytsems is the ability to estimate a body's orientation, as it allows the system to create a map of the environment or make decisions about the environment.

External motion capture systems (such as VICON) are accurate but are expensive, stationary and impractical for non-controlled environments. Thus, the problem of using on-board, low-cost sensors such as IMUs to detect the orientation of objects is very important for autonomous system applications. They are used in devices from iPhone to even drones. However, IMUs are prone to noise and drift, making raw measurements insufficient for estimating orientation.

In this project, we address the problem of estimating the 3D orientation of a rotating body using an Inertial Measurement Unit (IMU). This project has the two following parts:

- **Part 1:** Using an optimization based approach to estimate a trajectory of unit quaternions $q_{1:T}$ that minimizes rotation and observation errors
- **Part 2:** Demonstrating the practical utility of this estimation by constructing a panoramic image using the trajectory of estimated body orientations.

II. Problem Formulation

Our goal is to estimate the 3-D orientation of a rotating body overtimes, denoted as $\mathbf{q}_{1:T} = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T\}$ using measurements from an (Inertial Measurement Unit) IMU. Furthermore, we want to construct a panoramic image by stitching RGB camera images over time based on the body orientation $\mathbf{q}_{1:T}$

A. State and Sensor Variables

We represent the state at time t as the unit quaternion $\mathbf{q}_t \in \mathbb{H}_*$, which represents the orientation of the rotating body in the body frame $\{B\}$ with respect to the world frame $\{W\}$

$$\mathbf{q}_t = \begin{bmatrix} q_s & q_1 & q_2 & q_3 \end{bmatrix}^\top \in \mathbb{R}^4$$

, where the set of unit quaternions is defined as

$$\mathbb{H}_* := \{\mathbf{q} \in \mathbb{H} | q_s^2 + \mathbf{q}_\mathbf{v}^\top \mathbf{q}_\mathbf{v} = 1\}$$

The IMU's gyroscope provides the angular velocity of the body. This is a 3-dimensional vector in the body frame {B},

$$\omega_t = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^\top \in \mathbb{R}^3$$

Furthermore, the accelerometer provides the linear acceleration acting on the body (i.e. measured in the body frame). This is,

$$a_t = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^\top \in \mathbb{R}^3$$

To convert these measurements into pure quaternions, we turn these into 4D vectors and make the first element 0. So, the angular velocity is $\omega_t = [0, \omega_x, \omega_y, \omega_z]$ (no scalar part) and the linear acceleration is $a_t = [0, a_x, a_y, a_z]$

B. Motion Model

A motion model is used to predict the next orientation $\mathbf{q}_{t+1}$ using the angular velocity measurement $\omega_t \in \mathbb{R}^3$, the previous orientation $\mathbf{q}_t$, and the timestep τ_t. It relates the current state of the body and the control input with the change in the state. The kinematics of the orientation are given by:

$$\mathbf{q}_{t+1} = f(\mathbf{q}_t, \tau_t \omega_t) := \mathbf{q}_t \circ \exp\left(\left[0, \frac{\tau_t \omega_t}{2}\right]\right)$$

where $\circ$ is quaternion multiplication and exp is quaternion exponential.

C. Observation Model

Assuming the robot is only rotating, the accelerometer should only measure the gravity vector, $g = 9.81m/s^2$. The observation model relates the unknown state q_t to the known measurement a_t. In the World Frame $\{W\}$, the gravity vector is constant. This is represented as:

$$g_t^W = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^\top$$

The accelerometer measures this gravity vector, but rotated into the Body Frame $\{B\}$. The expected measurement, $h(\mathbf{q}_t)$, is the rotation of the world g vector into the sensor's body frame (in gravity units) is:

$$h(\mathbf{q}_t) = [0, a_t] = q_t^{-1} \circ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^\top \circ q_t$$

Here, $[0, a_t]$ is the accelerometer measurement represented as a pure quaternion.

D. Orientation Estimation Cost Function

We define the cost of the estimation to be as follows:

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

The first term measures the rotation error between $\mathbf{q_{t+1}}$ and the motion prediction using the motion model. Here, the operation $2\log(\cdot)$ extracts the axis-angle vector of the relative rotation error, and the norm penalizes the magnitude of this rotation angle. The second term measures the Euclidean error between the measured acceleration quaternion $[0, \mathbf{a}_t]$ and the *predicted* gravity vector.

To have the best estimation, we want these errors to be as low as possible. Thus, we solve for the orientation trajectory $q_{1:T}^*$ by minimizing the cost function for all orientations:

$$\min_{\mathbf{q}_1:T} c(\mathbf{q}_{1:T}) \quad \text{such that } ||\mathbf{q}_t||_2 = 1 \quad \forall t = 1, ..., T$$

Since we need the quaternions to have unit norm, this orientation estimation problem becomes a constrained optimization problem. The following sections talk about how we might solve such a problem.

E. Panorama Stitching

Once we have the orientation estimation from A-D, we want to use the estimated rotations to stitch together a sequence of RGB camera images $I_1, \ldots, I_N$ captured by the rotating body. So, we have RGB images $\{I_t\}$, with $I_t \in \mathbb{R}^{H \times W \times 3}$, and ($H$, W) are the height and width of the image, respectively. Furthermore, we are also given the horizontal field of view, $\text{FOV}_h = 60°$ and the vertical field of view, $\text{FOV}_v = 45°$. With our estimated rotations $\mathbf{q}_{1:T}$, or with given rotation matrices R_t in the VICON dataset, we want to find the *panoramic image*:

$$P \in \mathbb{R}^{\hat{H} \times \hat{W} \times 3}$$

Note: $\hat{H}$ and $\hat{W}$ are the height and width of the panoramic image, which we qualitatively set as 1024 and 2048 respectively.

III. Technical Approach

A. IMU Data Calibration

The raw IMU measurements are provided as a $7 \times N$ array, where N is the total number of samples. Each column has a single measurement with its timestamp, and the rows are ordered as follows:

$$\begin{bmatrix} t \\ a_x \\ a_y \\ a_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \in \mathbb{R}^{7 \times N}$$

Here, t denotes the timestamp (in unix time), $\mathbf{a} = [a_x, a_y, a_z]^T$ are the raw accelerometer readings, and $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ are the raw gyroscope readings, all expressed in 10-bit ADC units. IMUs have an inherent bias that must be corrected for. The first 5 seconds of the IMU data is stationary. During this time, the angular velocity measured by the gyroscope should be 0 rad/s and the true linear acceleration should be only due to g. We use the following equation to convert from raw A/D values to physical units,

$$\text{value} = (\text{raw - bias }) \times \text{scale factor}$$

$$\alpha = \text{scale factor} = \text{Vref}/1023/\text{sensitivity}$$

Here, we also have that the gyroscope sensitivity is 3.33 mV/deg/sec = $3.33 \times 180/\pi$ mV/rad/sec and the accelerometer sensitivity is 330 mV/g. The bias for the gyroscope is the average of the readings measured in the first 5 seconds. The gyroscope should read 0 rads/sec. So the average of the readings in the first 5 seconds is the bias:

$$\mathbf{b}_\omega = \frac{1}{5} \sum_{t=0}^{5} \omega_t^{\text{raw}}$$

We also compute the bias in the accelerometer as the difference between the average raw reading and the expected gravity vector. This is because when the body is stationary, the accelerometer should only read the gravity vector (in gravity units):

$$\mathbf{b}_a = \left(\frac{1}{5} \sum_{t=0}^{5} \mathbf{a}_{raw,t} \right) - \begin{bmatrix} 0 \\ 0 \\ 1/\alpha_{accel} \end{bmatrix}$$

Finally, we apply these measured biases to our data based on the equation for "value" shown above:

$$\omega_t = (\omega_t^{\text{raw}} - \mathbf{b}_\omega) \times \alpha_{gyro}$$

$$a_t = (\mathbf{a}_t^{\text{raw}} - \mathbf{b}_a) \times \alpha_{accel}$$

We get the following biases:

```
Acceleration bias:
[-510.39 -500.99 505.084]

Gyroscope bias:
[373.686 372.466 369.338]
```

Next, we use the corrected data for our orientation estimation cost optimization.

B. Orientation Estimation Cost optimization: Projected Gradient Descent

We use projected gradient descent to minimize the cost $c(\mathbf{q}_{1:T})$. Projected gradient descent moves the answer in the direction of steepest descent, $-\nabla c(\mathbf{q}_{1:T})$, and then projects the result back into the best element from the set that satisfies our constraint. This happens in two steps: (i) descent step, and (ii) projection step.

We initialize $\mathbf{q}_0 = [1, 0, 0, 0]^\top$, and compute $\mathbf{q}_{1:T}$ purely using the motion model. This "dead recoking", is our initial estimation for the first iteration, $k = 0$, and acts as a reasonable estimate to start the optimization. While this is not very accurate, it acts as a good starting point to begin our optimization. Figure. 1 shows this dead reckoned trajectory, and our goal is to improve it with our optimization.

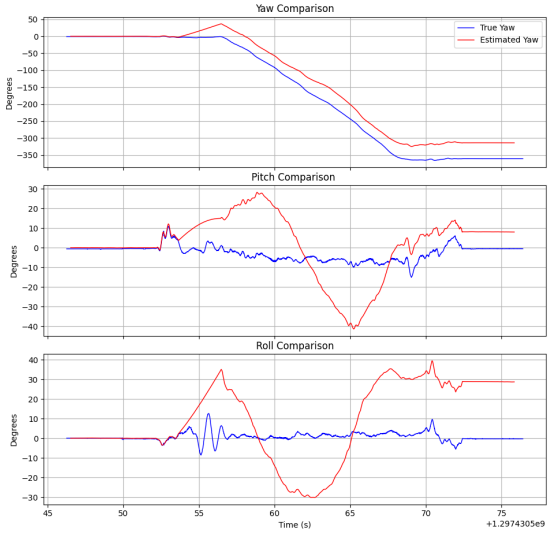


Fig. 1. Plot of Dead Reckoned (only motion model) orientation estimates for **Dataset 9** compared with VICON

Then, for iterations $k = 1, ..., N$, we update the orientation trajectory

$$\mathbf{q}_{1:T}^{(k+1)} \leftarrow \mathbf{q}_{1:T}^{(k)} - \alpha \nabla \delta_k$$

where $\delta_k = \nabla c(\mathbf{q}_{1:T}^{(k)})$ is the gradient of our cost function for the *entire* trajectory at iteration k. The gradient δ_k is computed using auto differentiation using PyTorch's autograd. Therefore, we are able to get exact computation for the derivatives efficiently and easily.

The update step $\mathbf{q}_t^{(k+1)}$ may violate our unit-norm constraint. That is, we may have ($\|\mathbf{q}_t\|_2 \neq 1$). To enforce the constraint $q_t \in \mathbb{H}_*$, we project each quaternion into $\mathbb{H}_*$ by normalizing it:

$$\mathbf{q}_t^{(k+1)} = \Pi_{\mathbb{H}_*}(\mathbf{q}_t^{(k+1)}) := \frac{\mathbf{q}_t^{(k+1)}}{\|\mathbf{q}_t^{(k+1)}\|_2}$$

This projection finds the closest unit quaternion to the unconstrained update in terms of Euclidean distance.

In our case, we use the hyperparamers: $N = 150$ iterations and $\alpha = 0.01$ learning rate.

Figure. 2 depicts the cost function decreasing every iteration. Projected gradient descent finds more optimal $\mathbf{q}_{1:T}$ and iteratively decreases $c(\mathbf{q}_{1:T})$ over 150 iterations.

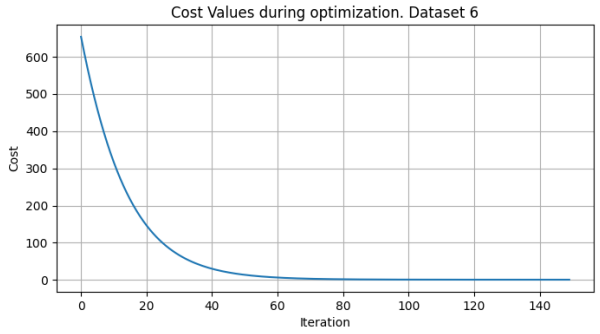


Fig. 2. Orientation estimation cost decreases steadily and smoothly

C. Panorama Building Technical Approach

We construct the panoramic image using equirectangular projection. The IMU and the VICON system have different sampling rates. Thus, we must associate each image I_t to an orientation. We have two cases:

- **VICON:** We find the rotation matrix R_t (from the provided $3 \times 3 \times N$ array) whose timestamp minimizes $|t_{\text{cam}} - t_{\text{VICON}}|$
- **Estimated:** We find the quaternion q_t from our optimized estimate, whose timestamp minimizes $|t_{\text{cam}} - t_{\text{IMU}}|$.

We construct the panoramic image using Equirectangular Projection, which assumes a spherical model. We treat each pixel (i, j) from the source image I as a point on a unit sphere (radius, $r = 1$) and map it to a 3D unit vector in the camera frame. We map camera images to this sphere, convert from the camera frame to the world frame using the rotation matrix R, and then use equirectangular projection to project the 3D world coordinates to a 2D rectangle. The flow goes like this:

$$(i, j) \rightarrow (\lambda, \phi, r) \rightarrow (x, y, z)_{world} \rightarrow (u, v)_{panorama}$$

We compute the latitude (λ) and longitude (ϕ) using $\text{FOV}_h = 60°$ and $\text{FOV}_v = 45°$ and height H and width W of the image:

$$\lambda = \left(0.5 - \frac{j}{W}\right) \cdot FOV_h, \quad \phi = \left(0.5 - \frac{i}{H}\right) \cdot FOV_v$$

The 3D unit vector $\mathbf{x}_{\text{cam}}$ is:

$$\mathbf{x}_{camera} = \begin{bmatrix} \cos\phi \cos\lambda \\ \cos\phi \sin\lambda \\ \sin\phi \end{bmatrix}^\top$$

Next, we rotate the 3D unit vector to the world frame using R. Based on the setting, R could come from VICON rotation matrices or from our estimated trajectory in the form of quaternions $\mathbf{q}_{1:T}$. Following is the conversion from camera frame coordinates into the world frame coordinates:

$$\mathbf{x}_{world} = R\mathbf{x}_{cam}$$

we convert the Cartesian coordinates to spherical world coordinates:

$$\lambda = \tan^{-1}(y, x) \in [-\pi, \pi] \quad \phi = \sin^{-1}(z) \in [-\frac{\pi}{2}, \frac{\pi}{2}]$$

Finally, we map the spherical angles to pixel coordinates (u, v) on the panorama image P

$$u = \frac{\lambda + \pi}{2\pi} \cdot \hat{W}, \quad v = \frac{\frac{\pi}{2} - \phi}{\pi} \cdot \hat{H}$$

Here, $\hat{H}$ and $\hat{W}$ are the height and width of the panorama. We apply a boundary mask to make sure that all coordinates fall within the dimensions $\hat{W} \times \hat{H}$. The continuous coordinates (u, v) are discretized into integer pixel indices using `.astype(int)`, and the RGB color data from the source image $I_t(i, j)$ is mapped to the destination $P(v, u)$. This gives us our panorama.

In the code, we use a step size of 2 (i.e. using every other image) to make the panorma smoother. Furthermore, we use $\hat{H} = 1024$ and $\hat{W} = 2048$.

IV. Orientation Estimation Results

A. Orientation Estimation: Training Set

The training set included datasets 1-9, (Figure 3, Figure 4, Figure 5, Figure 6, Figure 7, Figure 8, Figure 9, Figure 10, Figure 11). Across all the datasets we can see that the roll and pitch curves closely follow the ground truth VICON curves. On the other hand, we see that while the yaw follows the general shape of the curve, there is a large deviation in the yaw values.

The roll and pitch curves are *better* because of the corrections done by the observation model in our cost function. The cost penalizes deviations between the estimated gravity direction and the linear accelerometer measured by the accelerometer. Any tilt immediately changes the projection of gravity in the sensor frame, and the "observation model" term of the cost can do a correction.

On the other hand, yaw deviation is higher because we are doing pure rotation about the gravity axis. The accelerometer reading does not change much (it remains $[0, 0, g]$). As a consequence, the cost function receives little to no gradient information to correct errors in the Z-axis. Hence, the yaw is mostly dependent on dead reckoning, leading to a larger accumulated drift.

The yaw deviation in Dataset 4 is larger than normal. Upon consulting the professor, it was found that there may have been a glitch while collecting the data, explaining the larger deviation.

Furthermore, in dataset 6, there is a Roll discontinuity at $t = -13$. We use Numpy' `np.unwrap` function to turn non-continuous ($-180°$ to $180°$ jumps) into continuous lines for the plot. This may also occur if there are large jumps in angles. This explains why the ground truth data looks discontinuous in the graph for dataset 6. However, the estimation is likely still correct as we can note that the shape of the VICON data and estimated orientation curves exactly match. This is entirely a plotting error for the ground truth data. Similar issues occur when plotting the VICON data in Dataset 7. Yet, the estimated trajectory (shown in green) remains smooth.

The black parts of the panorama are the areas that were not captured by the camera.

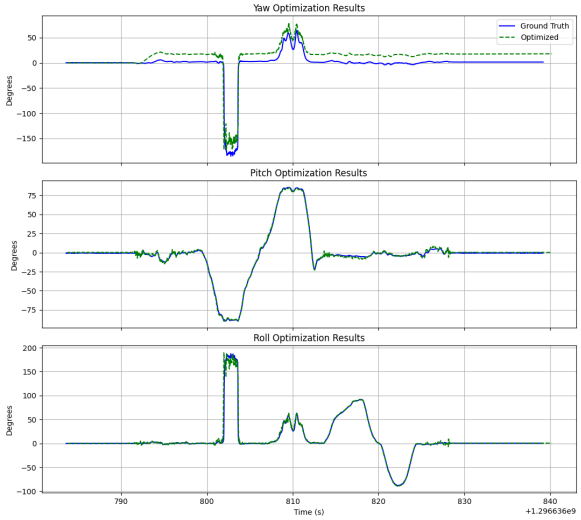


Fig. 3. Plot of estimated orientation (green) and VICON ground truth (blue) orientation on **Dataset 1** (Train set). The figure shows orientation in the form of Yaw (top), Pitch (middle), Roll (bottom)

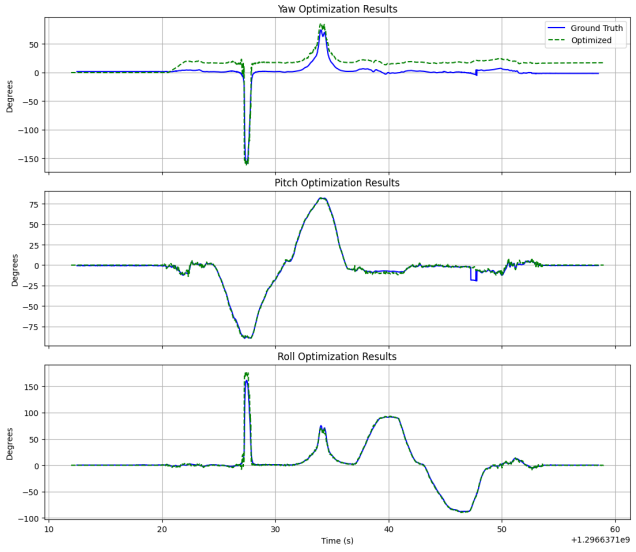


Fig. 4. Plot of estimated orientation (green) and VICON ground truth (blue) orientation on **Dataset 2** (Train set). The figure shows orientation in the form of Yaw (top), Pitch (middle), Roll (bottom)

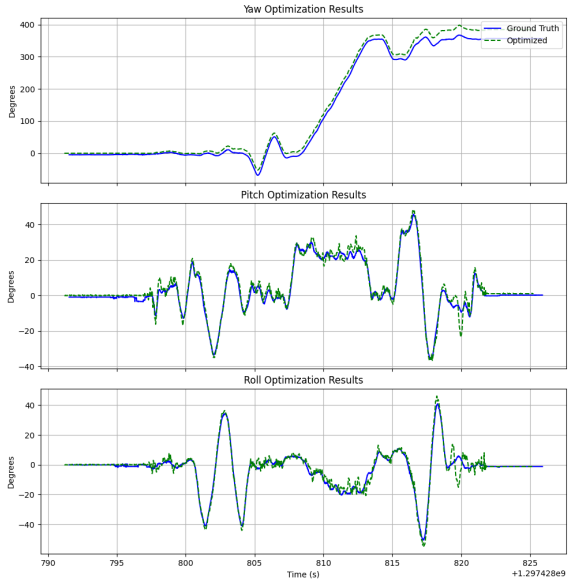


Fig. 5. Plot of estimated orientation (green) and VICON ground truth (blue) orientation on **Dataset 3** (Train set). The figure shows orientation in the form of Yaw (top), Pitch (middle), Roll (bottom)

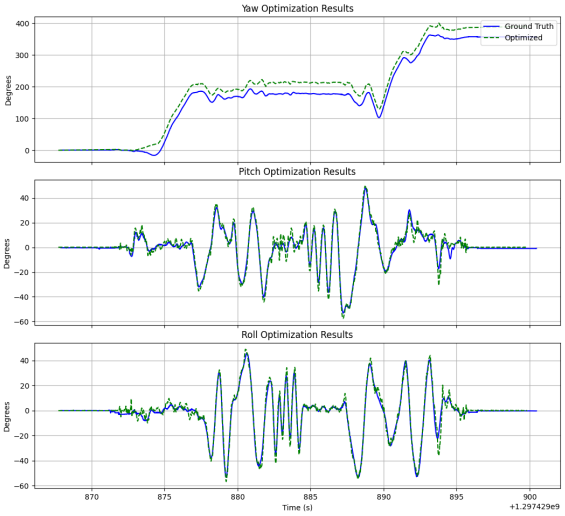


Fig. 7. Plot of estimated orientation (green) and VICON ground truth (blue) orientation on **Dataset 5** (Train set). The figure shows orientation in the form of Yaw (top), Pitch (middle), Roll (bottom)

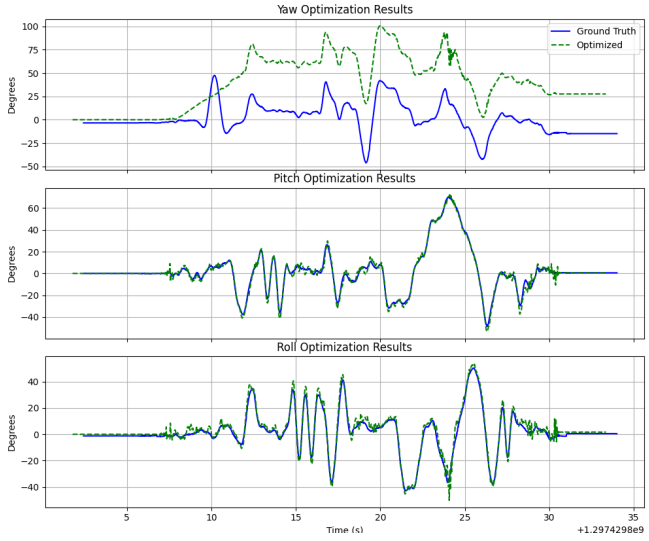


Fig. 6. Plot of estimated orientation (green) and VICON ground truth (blue) orientation on **Dataset 4** (Train set). The figure shows orientation in the form of Yaw (top), Pitch (middle), Roll (bottom)

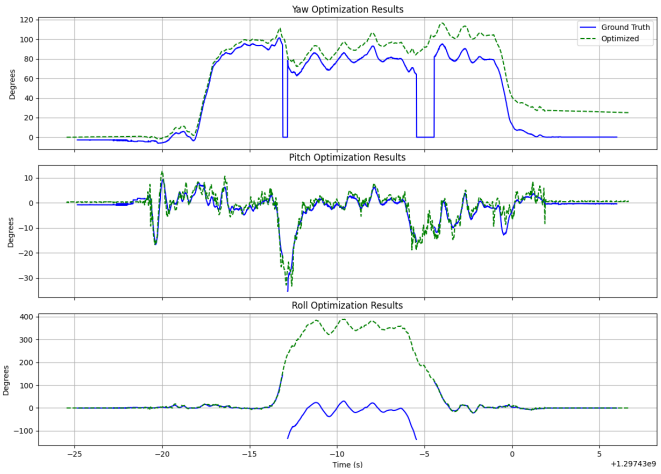


Fig. 8. Plot of estimated orientation (green) and VICON ground truth (blue) orientation on **Dataset 6** (Train set). The figure shows orientation in the form of Yaw (top), Pitch (middle), Roll (bottom). Discontinuity in ground truth curve due to np.unwrap shifting the data by 360 degrees. The green curve (est) should still show the correct trajectory as this is simply a cosmetic issue.

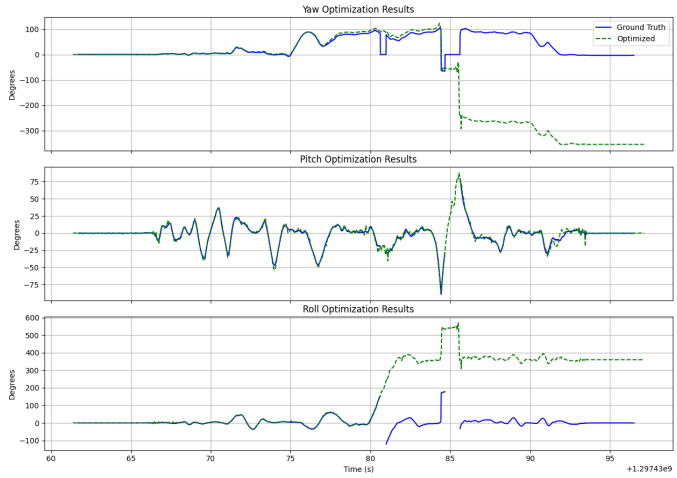


Fig. 9. Plot of estimated orientation (green) and VICON ground truth (blue) orientation on **Dataset 7** (Train set). The figure shows orientation in the form of Yaw (top), Pitch (middle), Roll (bottom). Discontinuity in ground truth curve due to np.unwrap shifting the data by 360 degrees. The green curve (est) should still show the correct trajectory as this is simply a cosmetic issue.

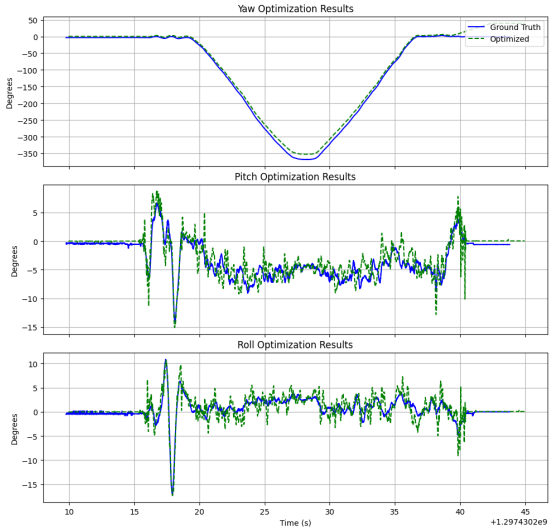


Fig. 10. Plot of estimated orientation and VICON ground truth orientation on **Dataset 8** (Train set)

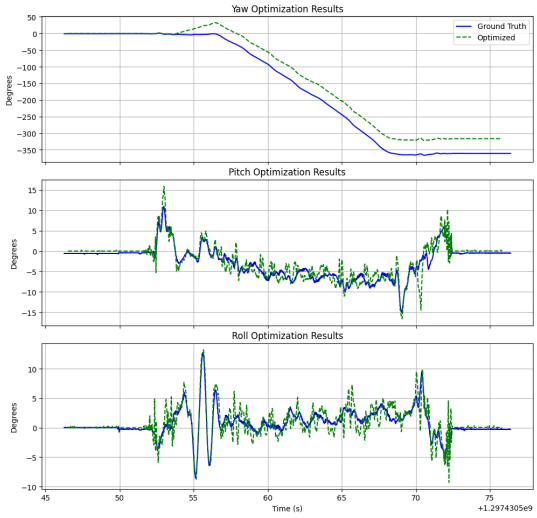


Fig. 11. Plot of estimated orientation and VICON ground truth orientation on **Dataset 9** (Train set)

B. Orientation Estimation: Test Set

Figure. 12, and 13 depict our orientation estimation for Datasets 10 and 11 respectively. Since no ground truth data were provided for these datasets, we do not have a direct comparison. In each case, there is a lot of movement, as represented by the sharp spikes in the Roll and Pitch plots for both runs. Given the previous trend in our roll, pitch and yaw graphs, it is highly likely that there is some deviation in the yaw, but the roll and the pitch follow the ground truth closely.

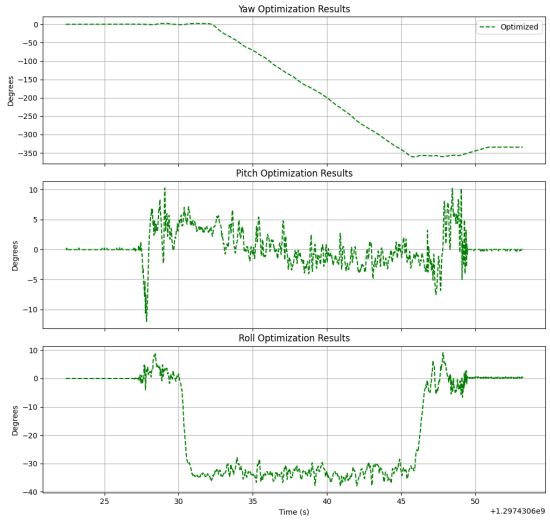


Fig. 12. Plot of estimated orientation on **Dataset 10** (Test set)

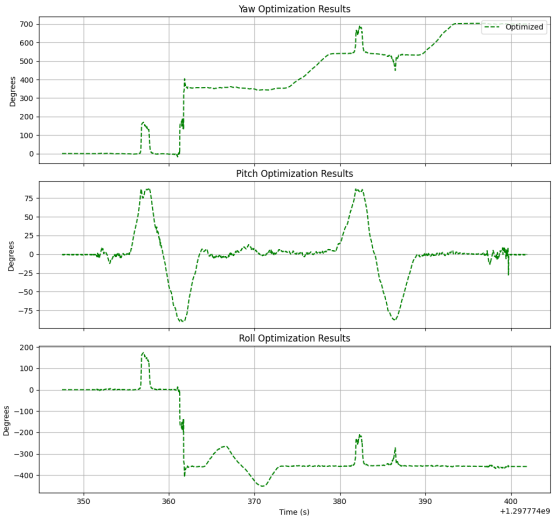


Fig. 13. Plot of estimated orientation on **Dataset 11** (Test set)

V. Panorama Results

A. Panorama: Train Set

We have camera data for datasets $1, 2, 8$ and 9. Figure 14, 16, 18, 20 show us the the panorama created with estimated orientation. Figure 15, 17, 19, 21 show us the panorama created with ground truth VICON data. In each case we have a panorama created with the VICON data (i.e. ground truth rotation matrices) and the estimated orientations generated by our projected gradient descent optimization. When comparing both these versions, we note that the VICON panoramas are much smoother. Parts of walls, doors and furniture from different sub-images are accurately aligned with eachother in the final panoramic image. On the other hand, panoramas generated with our orientation estimates are much more noisy. For instance, figure (created from dataset 10) has distortions due to horizontal misalignment. It is noticeable that the rota-

tion of one image does not seamlessly align with the beginning of the subsequent image. In addition, the red chair on the left hand side seems stretched out. This could be due to lag in time steps or just simple errors in estimations, and leads to things looking strethed out or appear double. We can also notice more jitter in our estimations compared to the VICON measurements.

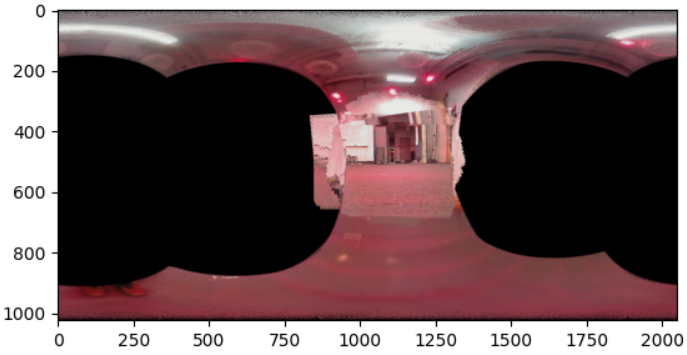


Fig. 14. Dataset 1 Panorama with estimated orientation

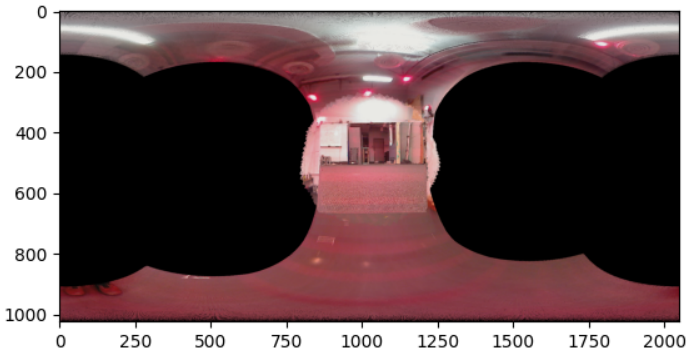


Fig. 15. Dataset 1 Panorama with VICON Data

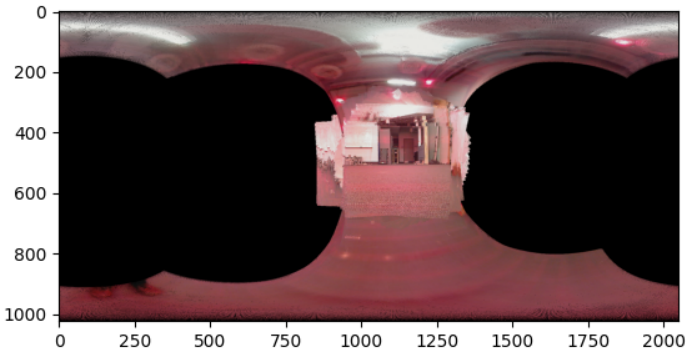


Fig. 16. Dataset 2 Panorama with estimated orientation

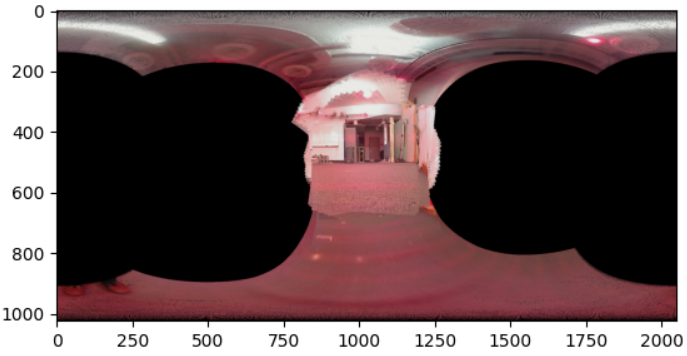


Fig. 17. Dataset 2 Panorama with VICON Data

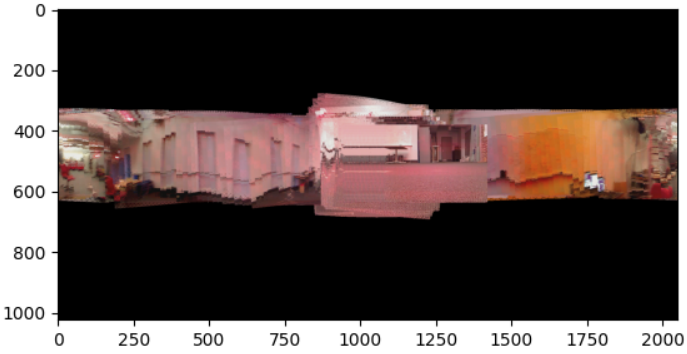


Fig. 18. Dataset 8 Panorama with estimated orientation

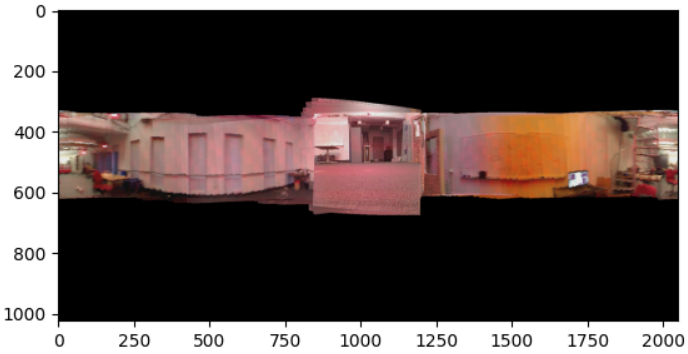


Fig. 19. Dataset 8 Panorama with VICON Data

B. Panorama: Test Set

Figures 22 an 23 depict the panoramic image created using the orientation estimation and camera data for the test data: Dataset 10 and Dataset 11. These results are similar to what we have seen for panoramas created with estimated orientations on the training data.

In general, we are able to identify structures such as doors, hallways and boundary lines between walls. We are also able to identify objects like tables and chairs. They are slightly stretched and distorted due to the errors and noise in our

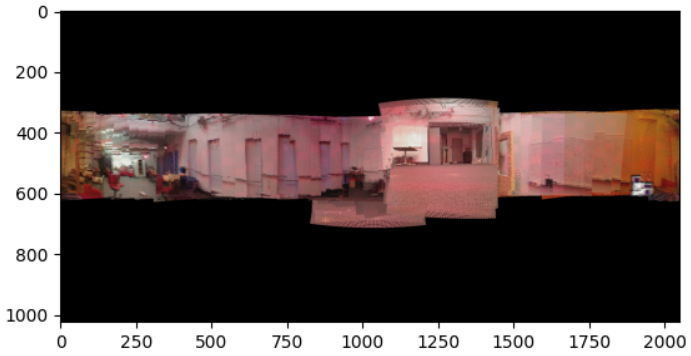


Fig. 20. Dataset 9 Panorama with estimated orientation

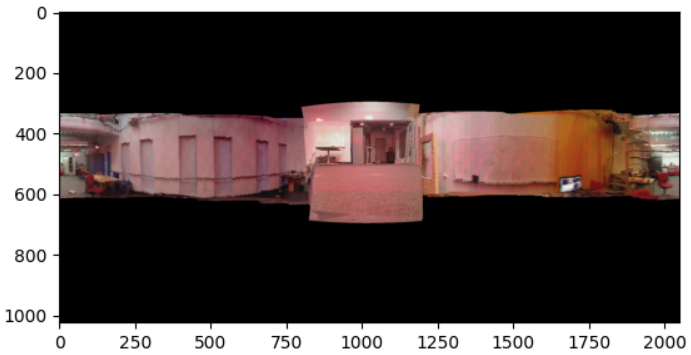


Fig. 21. Dataset 9 Panorama with VICON Data

estimation. Qualitatively, the generated panoramas are *good*. This also helps us verify that our orientation estimations are approximately correct.

Furthermore, The large black areas at the top and bottom of the images represent the areas that were not seen by the camera. This can be explained by the limited field of view of the camera. In Figure 22, we can notice that the top part is *jagged*. This is, perhaps, because the camera was moving up and down (pitch changing). This could change the horizon level of each image.

In Figure 23, and in some other images from the test data section, we see that the top parts are stretched and curved. This is a characteristic of equirectangular projection and is a by-product of us flattenting a sphere.

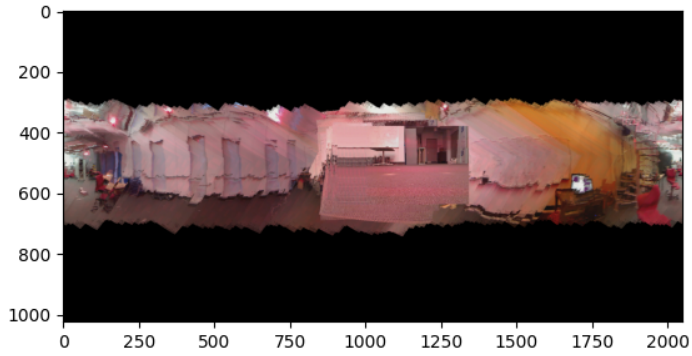


Fig. 22. Dataset 10 (Test Data) Panorama with estimated orientation

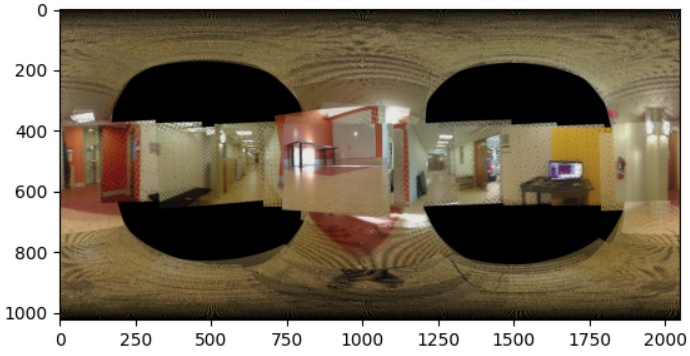


Fig. 23. Dataset 11 (Test Data) Panorama with estimated orientation

VI. Conclusion

This concludes our project, and we have discussed:

- The motion model, observation model, cost function and optimization strategy for estimating the orientation of a body undergoing pure rotation.
- Qualitatively verified our results by using the estimated orientations, camera images and equirectangular projection to construct a panoramic image.

References

[1] Unconstrained Optimization Lecture: https://natanaso.github.io/ece276a/ref/ECE276A_2_UnconstrainedOptimization.pdf.
[2] Rotations Lecture: https://natanaso.github.io/ece276a/ref/ECE276A_3_Rotations.pdf
[3] Motion and Observation Models Lecture: https://natanaso.github.io/ece276a/ref/ECE276A_4_MotionAndObservationModels.pdf
[4] Pytorch Autograd https://docs.pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html
[5] Numpy Unwrap Function https://numpy.org/doc/stable/reference/generated/numpy.unwr
[6] Piazza Questions: https://piazza.com/class/mhr7oh6rr7c3k8/post/71, https://piazza.com/class/mhr7oh6rr7c3k8/post/70, https://piazza.com/class/mhr7oh6rr7c3k8/post/66
[7] Equirectangular Projection: https://en.wikipedia.org/wiki/Equirectangular_projection - ”Reverse” section