Project 1: Orientation Tracking

Prathamesh Saraf (A59015739)
Department of Electrical and Computer Engineering
University of California, San Diego
psaraf@ucsd.edu

***Abstract*—This project aims to track the orientation of a camera sensor mounted on a robot and generate a panoramic image of its surroundings. The project employs the use of IMU sensor data to calculate quaternions through a motion model and an observation model. A cost function incorporating errors from both models is optimized through gradient descent to produce optimized quaternion trajectories. The optimized quaternion trajectory is then transformed into a rotation matrix and used to process the camera data, resulting in the final panoramic image.**

***Index Terms*—Inertial Measurement Unit, quaternions, gradient descent algorithm, orientation tracking**

I. Introduction

The goal of this project is to track the orientation of a camera sensor mounted on a robot and use the collected camera data to construct a panoramic view of its surroundings. To achieve this, the project employs a number of technical methods and processes.

Firstly, the IMU (Inertial Measurement Unit) sensor data, which includes the values from the gyroscope and accelerometer, is used to calculate quaternions. These quaternions are determined through a motion model that incorporates the angular velocity and time step information. With the calculated quaternions, an observation model is established to obtain the acceleration values.

We then build a cost function that incorporates the errors from both the motion model and the observation model. This cost function is minimized through the use of a gradient descent algorithm. The output of the optimization is a set of optimized quaternion trajectories that closely match the reference data from the VICON system.

In the second part of the project, the optimized quaternion trajectory is transformed into a rotation matrix. This matrix is used to perform various transformations and rotations on the camera data, which is then unwrapped to produce the final panoramic image.

In summary, this project combines various fields such as robotics, computer vision, and machine learning to track the orientation of a camera sensor, process the collected data, and generate a panoramic view of the environment.

II. Problem Formulation

We construct the motion model, which represents the trajectory of all the quaternions. The starting quaternion is assumed to be [1, 0, 0, 0], and subsequent quaternions are calculated using Equation 1.

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

We leverage the quaternions obtained from Equation 1 to compute our observation model. The observation model encompasses all of the acceleration values that correspond to each quaternion, which are calculated using Equation 2.

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

To optimize the quaternion trajectory, we first calculate the cost function using the motion model error and the observation model error. The motion model error is determined by comparing the previous quaternions with the newly generated quaternions resulting from the application of gradient descent. Meanwhile, the observation model error is calculated by comparing the IMU accelerometer data with the acceleration output produced by the observation model. The ultimate goal is to minimize both errors in order to obtain the optimal quaternion trajectory after the gradient descent algorithm has been executed. the cost function is given in Equation 3 below:

$$\mathrm{c}(\mathbf{q}_{1:T}) := \tfrac{1}{2} \textstyle\sum_{t=0}^{T-1} \left\| 2 \log\left(\mathbf{q}_{t+1}^{-1} \circ f\left(\mathbf{q}_t, \tau_t, \omega_t\right)\right) \right\|_2^2 + \frac{1}{2} \sum_{t=1}^{T} \left\| \mathbf{a}_t - h\left(\mathbf{q}_t\right) \right\|_2^2 \tag{3}$$

The cost function is then optimized over the constraint that all quaternions have unit norms as shown in equation 4.

$$\begin{aligned} \min_{\mathbf{q}_{1:T}} \quad & c\left(\mathbf{q}_{1:T}\right) \\ \text{s.t.} \quad & \left\|\mathbf{q}_t\right\|_2 = 1, \quad \forall t \in \{1, 2, \ldots T\} \end{aligned} \tag{4}$$

Gradient Descent is an optimization algorithm used in machine learning to minimize a cost function by iteratively adjusting the parameters of a model. The cost function represents the error or difference between the predicted values of the model and the actual values. The goal of the algorithm is to find the set of model parameters that minimize this cost function.

In Gradient Descent, the optimization process is performed by computing the gradient of the cost function with respect to the model parameters. The gradient represents the slope of the cost function and points in the direction of the maximum increase in the cost function. By taking small steps in the opposite direction of the gradient, the algorithm adjusts the parameters in such a way that the cost function is gradually minimized.

$$\mathbf{q}_{t+1} = \mathbf{q} - \alpha \cdot \nabla c(\mathbf{q}) \tag{5}$$

III. Technical Approach

A. Calibration

The IMU data must undergo calibration to eliminate bias from the gyroscope and accelerometer readings. This is achieved by determining the average acceleration and angular velocity values from the IMU when the robot is stationary and not rotating. The length of this stationary period is determined using the VICON data, as the rotation matrix from the VICON readings will be an identity matrix during this time. To find the bias, we calculate the average acceleration and omega values during the stationary period as determined by the VICON data and subtract this bias value from the IMU readings. The result is a corrected and calibrated set of acceleration and angular velocity values. Here, I was getting the acceleration plot to be 180 degrees out of phase with the VICON data. Even after debugging with the help of TA, I wasn't able to figure out the problem. The acceleration plot was matching for z values but not for x and y values. Also, I was getting suspiciously high acceleration values for x and y (around 65000). Hence, I did not change the sign of the x and y acceleration values as mentioned in the doc. After this, the problem was solved and the acceleration data for all 3 axes was matching with the VICON data.

TABLE I
Bias values for each dataset

Dataset	Linear acceleration	Angular velocity
1	[510.80, 500.99, 605.15]	[373.57, 375.37, 369.68]
2	[511, 500., 605.18]	[373.63, 375.37, 369.65]
3	[509.90, 501.08, 607.58]	[373.71, 375.85, 370.02]
4	[512.58, 503.025, 607.85]	[374.24, 376.25, 371.24]
5	[513.02, 501.68, 605.82]	[375.77 , 377.03, 373.58]
6	[510.27, 498.78, 605.57]	[373.37, 375.70, 369.79]
7	[511.98, 500.32, 605.95]	[374.50, 376.74, 369.60]
8	[511.75, 501.10, 607.93]	[373.45, 376.27, 369.68]
9	[513.21 , 501.16, 607.44]	[373.77 , 376.14, 369.27]
10	[511.73, 499.67, 606.21]	[373.52, 376.08, 369.61]
11	[510.18, 500.38 , 605.88]	[373.77, 375.22, 369.74]

B. Motion and Observation Model

1) Motion Model: The motion model generates the quaternion trajectory using the exponential value of angular velocity and timestep, multiplied by the previous timestep quaternion. We start with q = [1, 0, 0, 0] and then calculate the complete trajectory.
2) Observation Model: using the quaternions generated from the motion model, we calculate the acceleration trajectory for the respective quaternion value.

C. Cost Function

The cost function in the optimization process incorporates two error terms: the motion model error and the observation model error. The motion model error is determined by comparing the previous quaternion trajectory and the updated quaternion trajectory obtained through one iteration of gradient descent optimization. The observation model error is computed by calculating the difference between the IMU's unbiased acceleration readings and the observation model's output which is computed at each iteration of gradient descent. The equation for the cost function is presented in (3). However, encountering 'Nan' values for the motion model error term due to certain norm values equaling zero prompted the removal of this term from the optimization process. As a result, the optimization model only optimizes the cost derived from the observation model.

D. Gradient Descent Optimization

The gradient descent algorithm is executed for 20 iterations with the objective of minimizing the cost function in each iteration. The derivative of the cost value with respect to the quaternion trajectory at each step is calculated, yielding the "delta" used in the optimization equation. With a learning rate of 0.001, the optimized quaternion trajectory is calculated for each iteration. This updated trajectory is then supplied to the motion model, observation model, and cost function for further processing, resulting in a new cost value. This new value is differentiated with respect to the quaternions at that iteration and utilized in the gradient descent equation to determine future quaternions. After 20 iterations, the cost function reaches convergence or stability and does not show significant further reduction. Therefore, the gradient descent process is terminated after 20 iterations and the quaternion trajectory generated after the 20th iteration is used. This trajectory is then utilized to generate the rotation matrix, which is discussed in the following section.

E. Image Orientation

We have now optimized the quaternion trajectory through gradient descent optimization and can proceed with constructing the panorama. The rotation matrices are calculated from the optimized quaternion trajectory using the transforms3d library. However, since the camera data was recorded at a different frequency than the IMU data, we must identify and select the quaternion values that are closest to the camera image timestep in order to match the timestep of the camera image. The camera images are represented in a spherical coordinate frame, each with a resolution of 240 x 320 pixels, covering an angular extent of 60 degrees horizontally and 45 degrees vertically on a sphere. Using this information, we calculate the spherical coordinates for each pixel, starting from the top left corner [0,0] and with each pixel separated by a factor. The spherical coordinates are transformed into Cartesian coordinates, and then into the world frame by rotating them using optimized quaternion trajectory rotation matrices. With the image coordinates now in the world frame, we must convert them back to a spherical coordinate system. This conversion is necessary because the RGB values in the camera data are assigned based on spherical coordinates. Thus, in order to bring the spherical coordinates into the world frame, we must first convert them from Cartesian to spherical.

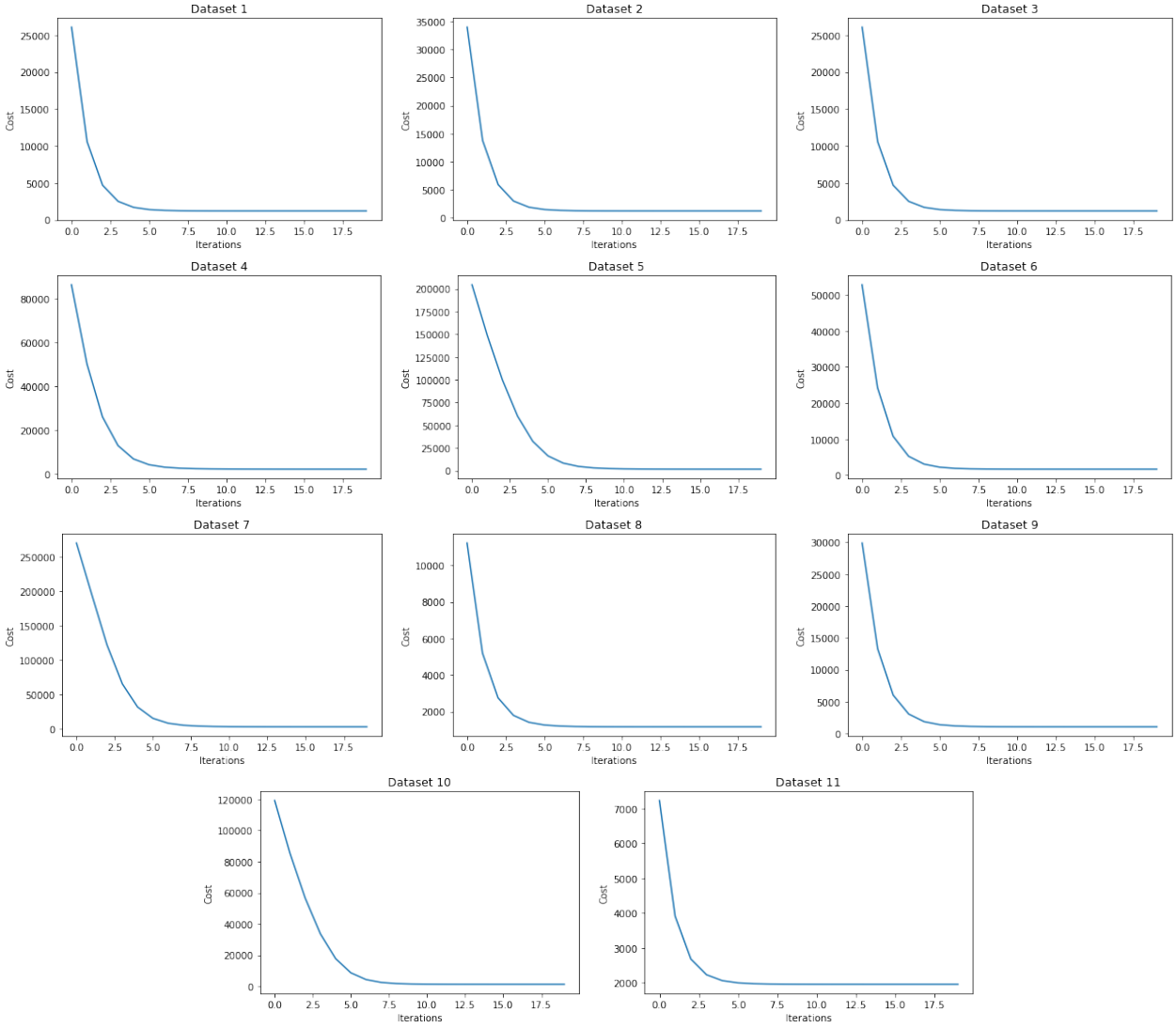

Fig. 1. Cost value per iteration for training and testing dataset

F. Panorama

After assigning all the RGB values to the spherical coordinates in the world frame, we use a cylindrical projection to project them onto a cylindrical surface. This is achieved using the Mercator projection method. In this projection, we are left with only the [x, y] coordinate values for each pixel. We then map the corresponding RGB values from the camera data onto the cylindrical surface for each pixel. Once all the assignments are complete, we unfold the cylinder to obtain the final panoramic image.

IV. Results

This section presents the plots for:

1) The cost value per iteration for all datasets, Fig 1.
2) The panorama generated for the respective training datasets, Fig 2.
3) The roll, pitch, and yaw values for the optimized trajectory vs the ground truth data (VICON) data, Fig 3.

V. Acknowledgement

I collaborated with Mihir Kulkarni (A59018127) for the assignment where we helped in debugging each other's code.

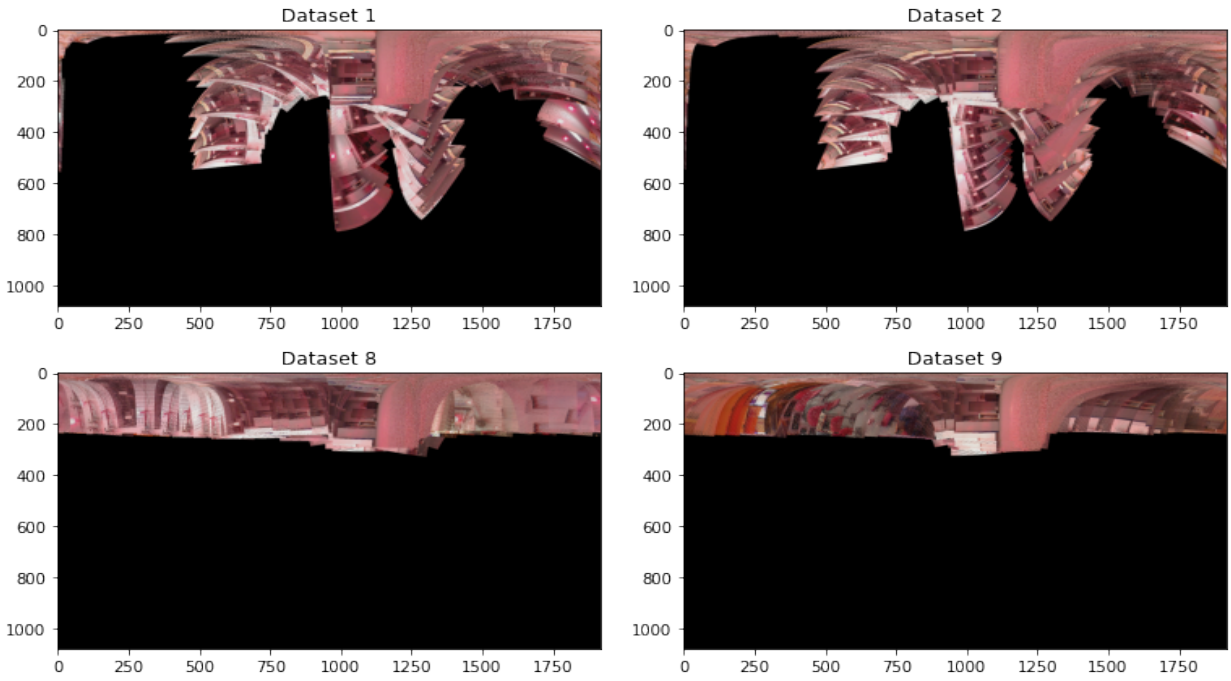

Fig. 2. Panorama generated for training and testing datasets


Roll Dataset 1
Pitch Dataset 1
Yaw Dataset 1
Roll Dataset 2
Pitch Dataset 2
Yaw Dataset 2
Roll Dataset 3
Pitch Dataset 3
Yaw Dataset 3
Roll Dataset 4
Pitch Dataset 4
Yaw Dataset 4
Roll Dataset 5
Pitch Dataset 5
Yaw Dataset 5
Roll Dataset 6
Pitch Dataset 6
Yaw Dataset 6
Roll Dataset 7
Pitch Dataset 7
Yaw Dataset 7
Motion Model Roll
VICON Roll
Optimised Roll
Motion Model Pitch
VICON Pitch
Optimised Pitch
Motion Model Yaw
VICON Yaw
Optimised Yaw
Roll
Pitch
Yaw
time

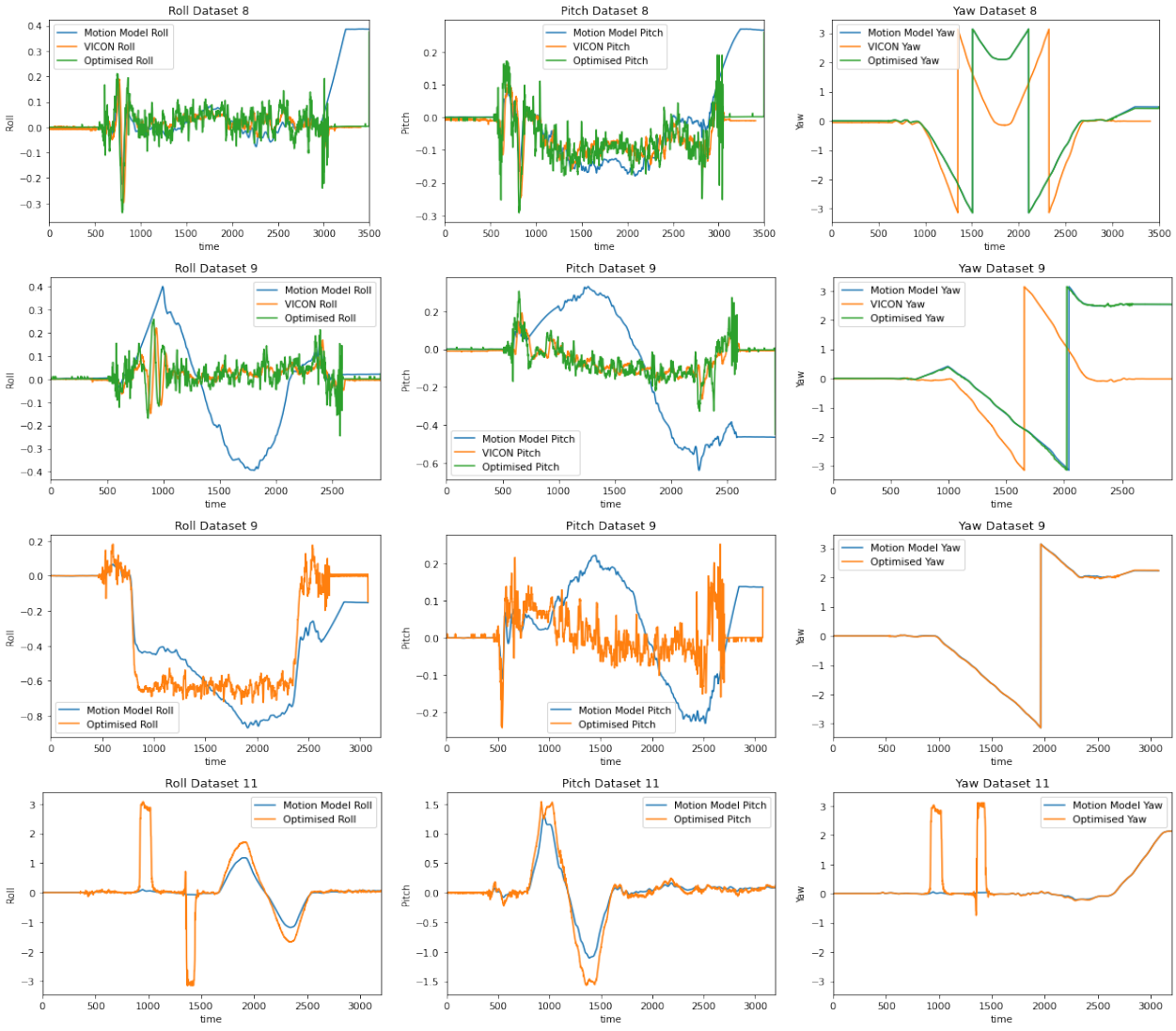

Fig. 3. Roll, Pitch, and Yaw values for all datasets