ECE 276A Project 1: Orientation Tracking

Unay Shah PID: A59015777

Keywords: orientation tracking, IMU, VICON, image rotation, spherical coordinates, lambert azimuthal coordinates, panorama, gradient descent, quaternions

Data Source: *https://drive.google.com/drive/folders/1-Ot6Yr_zCEMKqYN5hIXYfysUy09qx_G3*

Abstract:
This project attempts to use data from IMU and cameras to first improve the angular velocity measurements and get orientation of the sensor setup. These values are used to get a rotation matrix, which is mapped into world coordinate frame using Lambert Azimuthal projection and cylindrical projection. The images are compiled into a larger image corresponding to the pixel values.

Introduction

The problem requires extraction of rotations in terms of quaternions from IMU data and then using this data in conjunction with images taken by a camera to create a panorama of the area around the setup. There are 3 sensors on the unit: an IMU sensor which captures the angular velocity and linear acceleration of the setup, a VICON sensor, which provides the true rotation matrix by measuring the orientation of the sensor in 3D space, and the camera, which captures images corresponding to the orientation of the setup. All 3 sensors are accompanied by the corresponding timestamp which shows the time at which the sensor was triggered to capture the value.

Using the IMU data to get the angular velocity and linear acceleration, we need to calculate the rotation matrix at each timestamp. We estimate the orientation in terms of quaternions, which are calculated using the angular velocity and difference in timestamp with the motion model. The motion model helps predict the next quaternion measurement using the current value. We also use the observation model, which predicts the next sensor measurements using the current state [1]. The acceleration values are used to build the observation model. As the body is undergoing pure rotation, it feels an acceleration of $[0, 0, -g]$. Hence we can predict the acceleration from the orientation of quaternion.

These values are then optimized using a gradient descent. The predicted value of acceleration is referenced with the IMU data acceleration and this correction is smoothed using the difference between consecutive orientation values. This allows us estimate the orientation trajectory based on the motion and observation models. This orientation trajectory then allows us to calculate the rotation matrix for the setup and using this rotation matrix, we can map the scene being viewed by the camera into an image.

The image viewed by the camera spans 60° horizontally and 45° vertically in its frame. To map it to an image, it needs to be distorted and corrected into a different coordinate system, which is where we use the rotation matrix. This helps find the projection coordinate of each pixel of an image to a flat plan, hence sewing the set of images into a single larger image.

This is an important problem to solve as it helps imaging systems accurately reconstruct their environments using sensor data and cameras. Noise in the IMU data can be minimized using a combination of the angular velocity and linear acceleration readings. Once this is done, the rotation matrix comes out to be more accurate and a better scene can be reconstructed.

Problem Formulation

The IMU sensor gives the angular velocity at time t: ω_t and the angular acceleration at time t: α_t. We use quaternion q_t to represent the body-frame orientation at time t. Using these, and the difference in consecutive timestamps τ_t, we can predict the next quaternion q_{t+1} using the motion model:

$$q_{t+1} = f(q_t, \tau_t \omega_t) \coloneqq q_t \circ e^{\left[0, \frac{\tau_t \omega_t}{2}\right]} - 1$$

For q_0, we assume the sensor is in its initial state and take the reading as $[1, 0, 0, 0]$. We can then use these orientation changes with the expected acceleration to predict the linear acceleration:

$$a_t = h(q_t) = q_t^{-1} \circ [0, 0, 0, -g] \circ q_t \quad -2$$

With this acceleration, we can now try to get more accurate orientation and trajectory by carrying out a gradient descent on the quaternions we obtained. We are assuming that there is some noise in the angular velocity of the IMU data, and trying to remove it using the linear acceleration data from the IMU. The cost is a combination of the losses of consecutive quaternions and the difference between actual IMU acceleration values and the predicted values from the observation model. We define the cost for all timestamps as follows:

$$c(q_{1:t}) = \frac{1}{2}\sum_{t=0}^{T-1} ||2log\left(q_{t+1}^{-1} \circ f(q_t, \tau_t \omega_t)\right)||_2^2 + \frac{1}{2}\sum_{t=1}^{T} || a_t \circ h(q_t)||_2^2 \quad -3$$

We then calculate the gradient of this cost function and minimize it to make it a constraint optimization problem. Using gradient descent, we try to optimize the value of q_t:

$$q_{k+1} = q_k - \alpha \nabla c \quad -4$$

where q_k denotes the value of q after k iterations.

Technical Approach

Cleaning sensor data

We start by removing the bias term from the readings of the IMU sensor. Initially, I tried to dynamically find the index till which the setup was stationary by checking the VICON data. In case of no motion, the rotation matrix would be an identity matrix. Considering sensor noise, a threshold value was set such that: $threshold * I \geq I - R$. But the threshold needed to be 0.017 for dataset 1 to consider the first 800 values as stationary, while the same value gave 0 stationary values to other datasets. Similarly, a threshold of 0.1 was required to take the first 600 values as stationary for dataset 8. Moreover, the testing dataset would not have VICON data, so this was not a feasible way to account for initial conditions. As a result, the index was set to a fixed value of 400, an average of all angular velocity and linear acceleration was taken till this index, and that value was subtracted from all the values of the corresponding sensor data.

The acceleration is measured in gravity units, so $g = 9.81$ is multiplied with all the bias corrected acceleration data. As the object is undergoing pure rotation, the only acceleration observed is downwards due to gravity. So g is added to the z-coordinate acceleration. The x and y axis are multiplied by -1 to correct the directional difference of the sensor.

Calculating quaternions and acceleration

The value of the exponential of product of angular velocity and time duration is calculated and this is further used to calculate the quaternion values from the corrected angular velocity data using equation 1. These quaternion values are used to further calculate the acceleration values as seen in equation 2. The Euler angles are then calculated from the quaternion values and plotted against the Euler angles obtained from the VICON data for datasets 1 to 9. *Transforms3D* library is used to convert the angles from rotation matrix and from the quaternions. These results, along with the acceleration results are plotted.

Gradient Descent

The cost function, as described in equation 3 is used and values of the quaternions are brought closer to the value of sensor's acceleration. The function is vectorised so that 100 iterations are completed within 3 seconds. These corrected quaternions are again used to calculate the acceleration and Euler angles. If there is no VICON data provided in the dataset, the Euler angles are used to generate them and store for use in panorama creation.

Creating Panorama

Given images from the camera along with timestamps at which they were captured, we can use data from the rotation matrix to arrange the images in world-frame and create

a panorama. We are told that the camera sensor has a field of view of 60° horizontally and 45° vertically and the images provided are 240 pixels in height and 320 pixels in width. We can create a spherical coordinate system by creating a 2D matrix with the corresponding point values.

This is obtained by dividing the range of values from $90 - \left(\frac{60}{2}\right) = 60$ and $90 + \left(\frac{60}{2}\right) = 90$ into 320 parts horizontally, and $90 - \left(\frac{45}{2}\right) = 67.5$ and $90 + \left(\frac{45}{2}\right) = 67.5$ into 240 parts. These spherical coordinates need to be transformed into Cartesian coordinates using the following mapping:

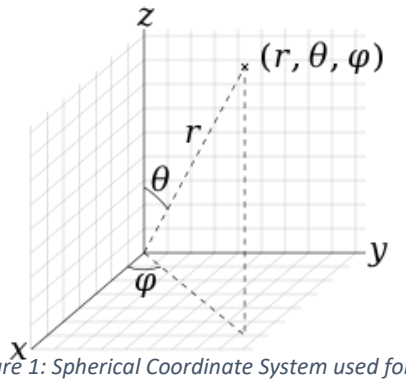


Figure 1: Spherical Coordinate System used for panorama [Source: Wikipedia]

$$x = \rho \sin\varphi \cos\theta$$

$$y = \rho \sin\varphi \sin\theta$$

$$z = \rho cos\varphi \text{AA}$$

After mapping these coordinates to Cartesian coordinates, we can use the list of rotation matrices to find values that maps each image to its world frame coordinates.

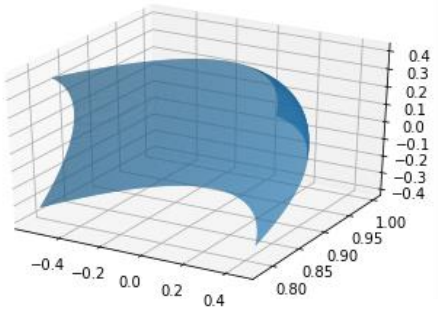


Figure 2: Mapping the rotation matrix to the world frame

I have used two systems of projections: Cylinder coordinates (as discussed in the lecture slides) and Lambert azimuthal equal-area projection. The conversion from world frame to this Lambert azimuthal equal-area can be done using the equations:

$$X = \sqrt{\frac{2}{1-z}}x$$

$$Y = \sqrt{\frac{2}{1-z}}y$$

After converting the projection into the lambert system, iterate over each coordinate

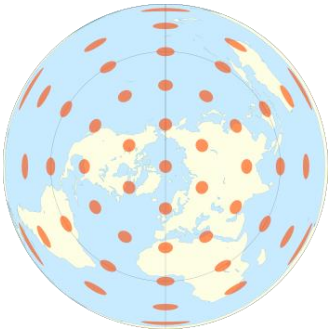


Figure 3: Expected results from lambert azimuthal equal-area projection [Source: Wikipedia]

map, use its X-Y coordinates with the resultant image and assign pixel values from the camera data.

For the cylindrical coordinates, the world frame is first converted to spherical coordinate system using the following mapping:

$$\rho = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \arctan\frac{y}{x}$$

$$\varphi = \arccos\frac{z}{\sqrt{x^2 + y^2 + z^2}}$$

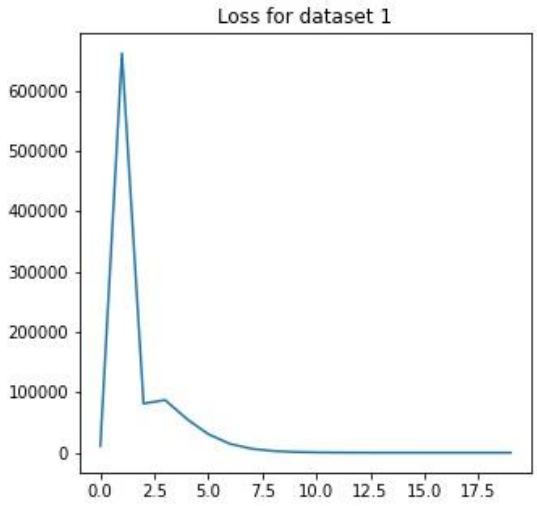
The final image is taken to be of dimension 1920×1080. So a scaling factor is calculated to transform the θ and φ values into the Cartesian coordinates. The ρ value obtained during this conversion is 1.0 for all values, implying that the image plane is present at unit distance from the point of observation. The coordinates corresponding to θ can be calculated by multiplying $1080/\pi$ and that for φ can be found by multiplying $1920/\pi$ to their values. An additional π needed to be added to the longitudes due to the Python function `np.arctan2` being used, which returns values between $-\pi$ and π. After these manipulations, the image can be plotted.

Results

When calculating the quaternions, and observing them, Euler angles obtained from initial data followed a general trend of the Euler angles calculated from the VICON data,

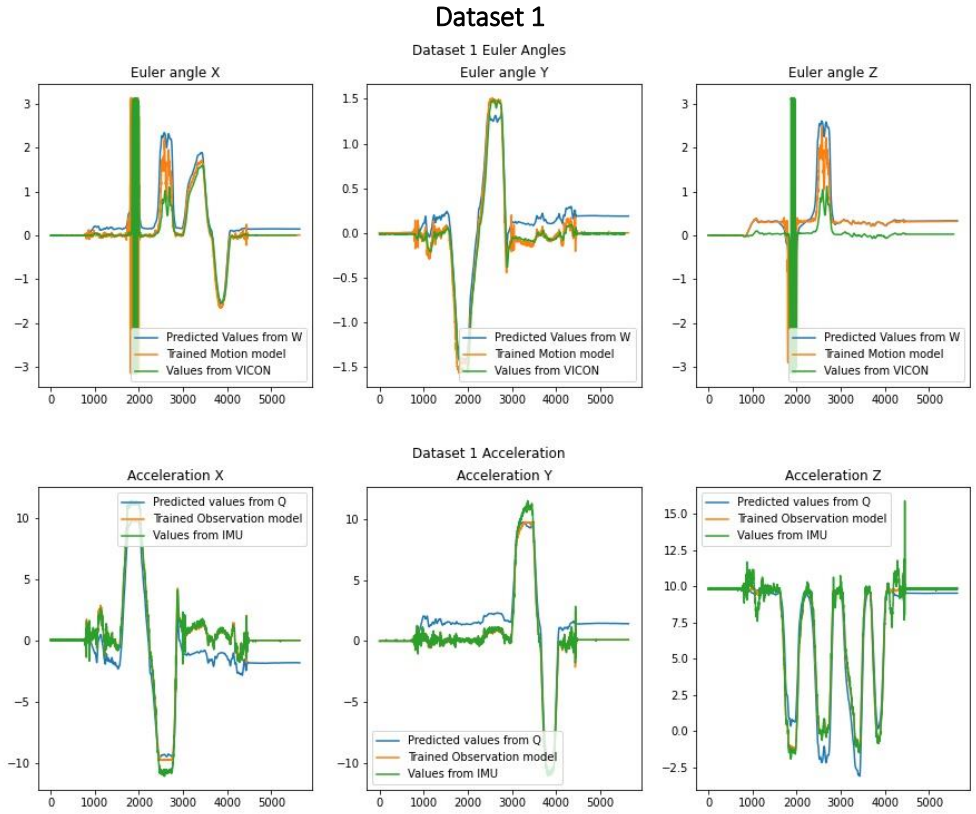
but there was huge error between the values. Acceleration values had relatively lesser error even before optimization (except in dataset 9). Using gradient descent to optimize the quaternions with acceleration values from the

IMU and quaternions from the next timestamp resulted in Euler angles which were much closer to the VICON data angles. The acceleration graphs almost overlap in all cases along X and Y axes. But along the Z axis, there is a significant offset every time. This can be attributed to most motion taking place along this axis and hence the noise increasing in the reading as we integrate.

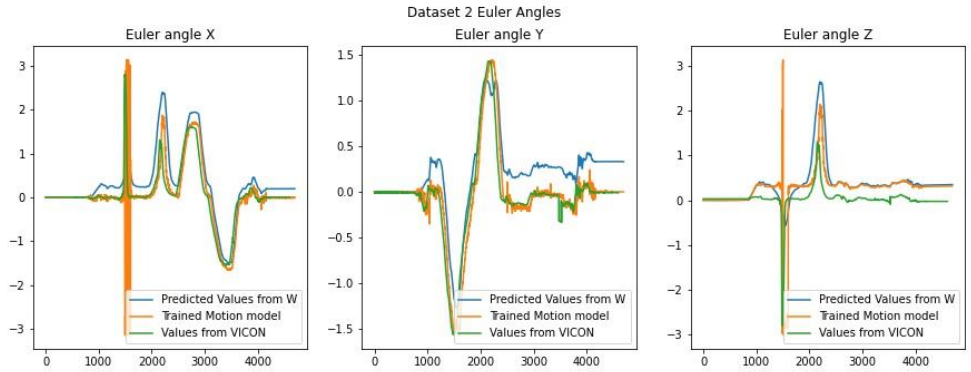


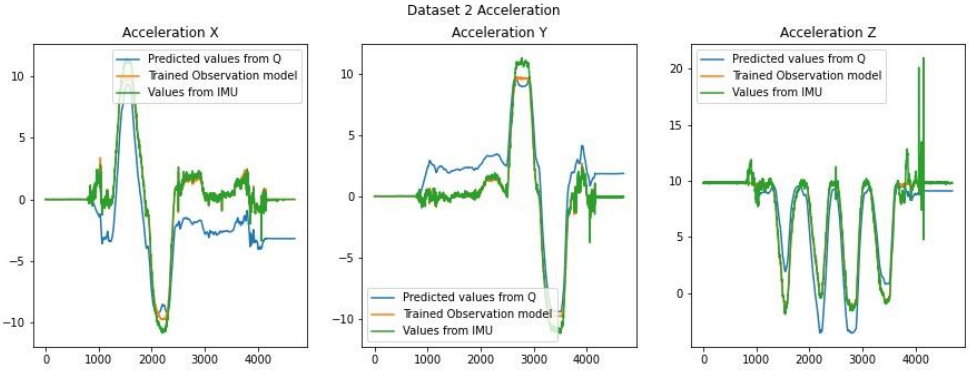
Plots:

Motion and Observation models

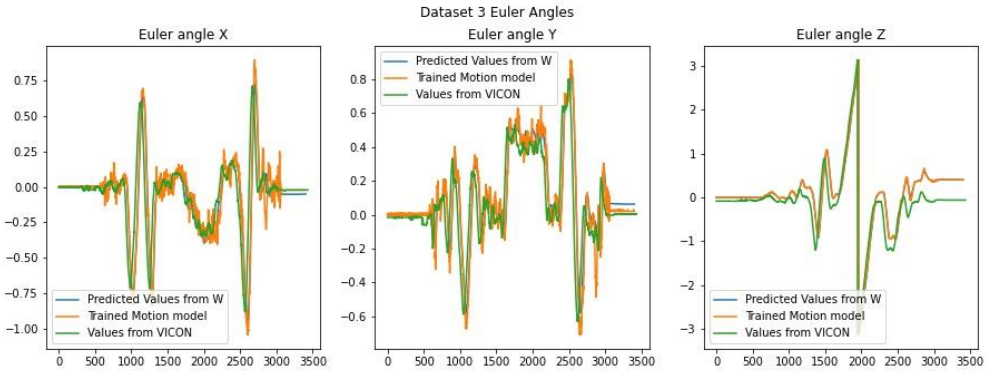


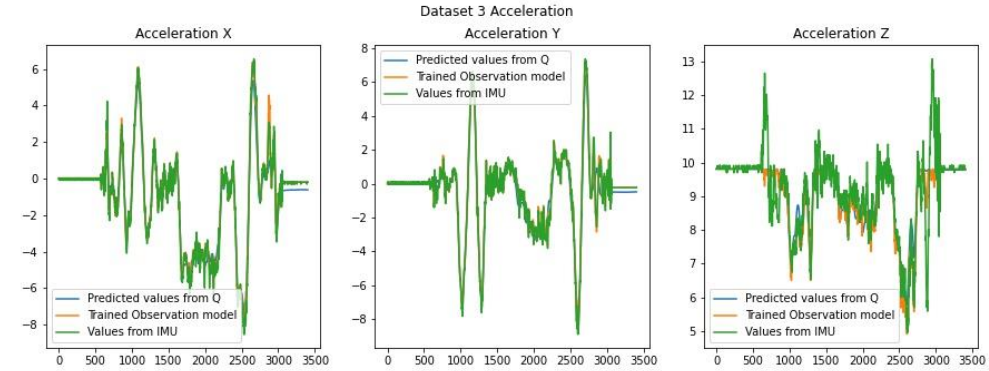
Dataset 2



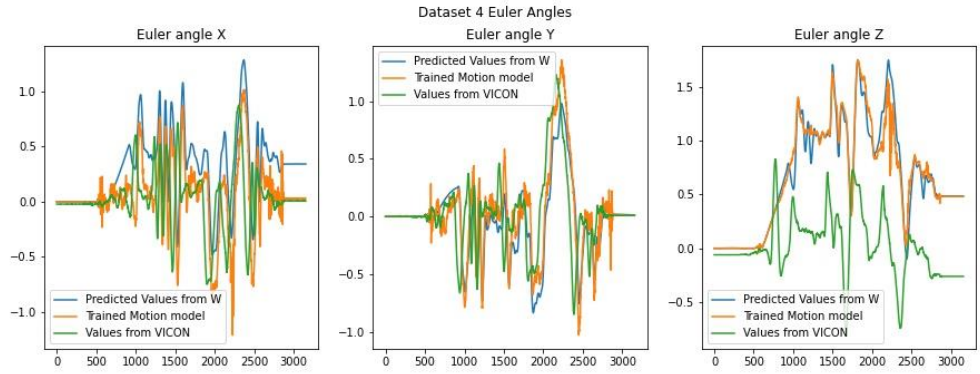


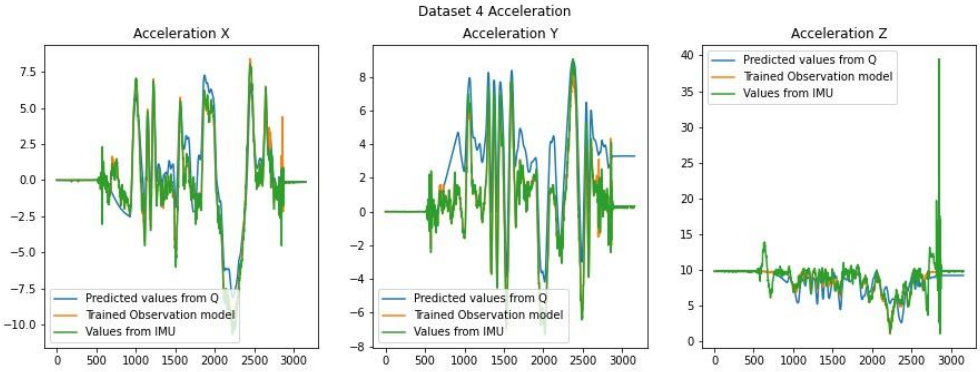
Dataset 3



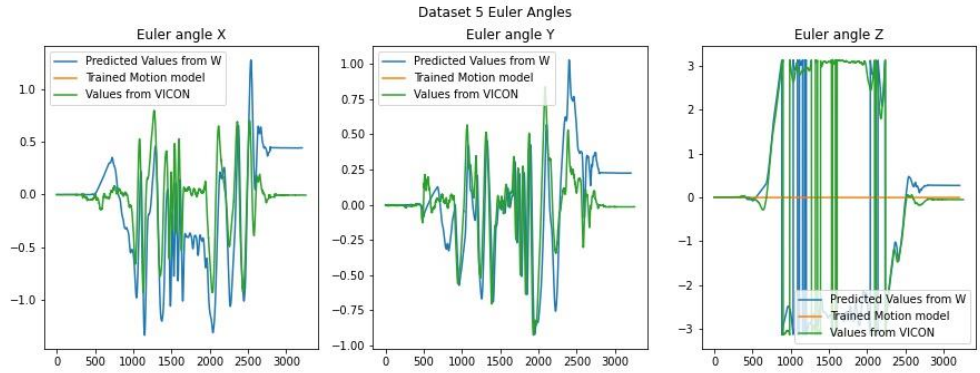


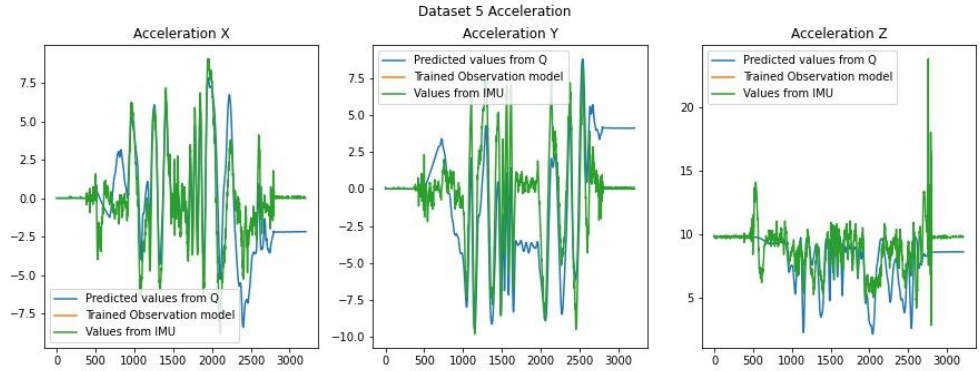
Dataset 4



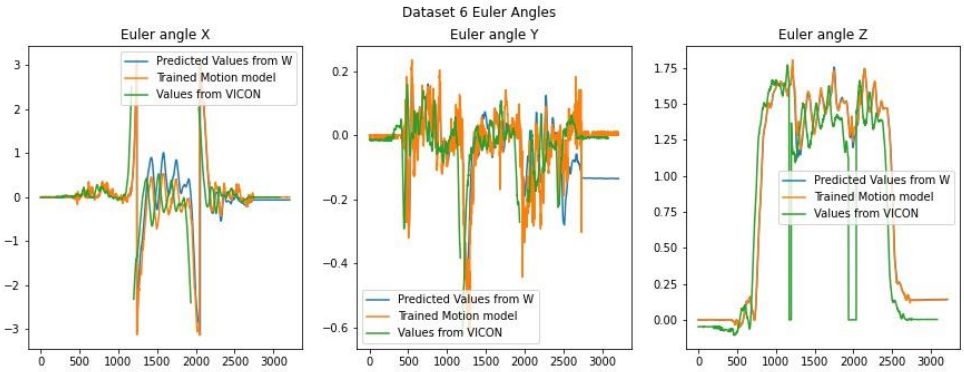


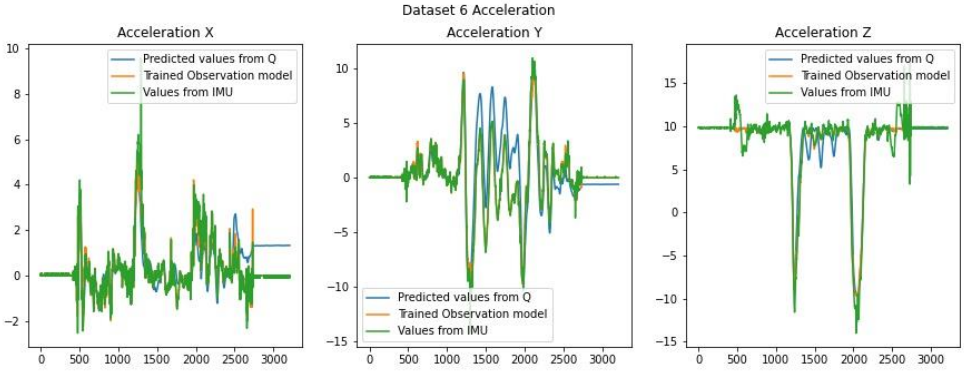
Dataset 5



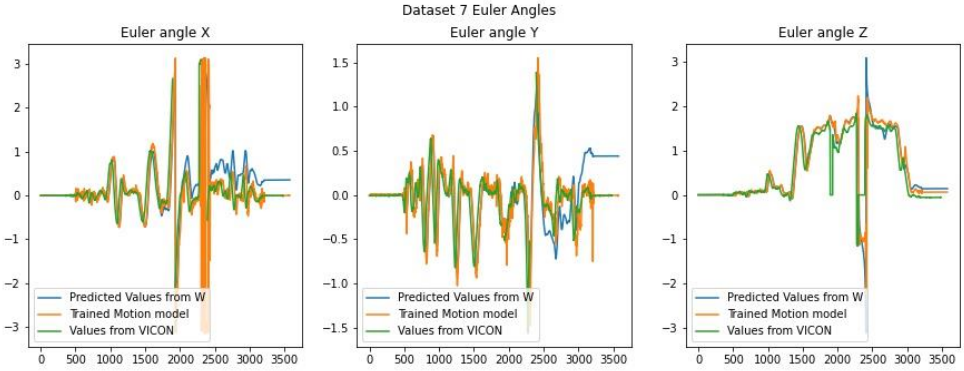


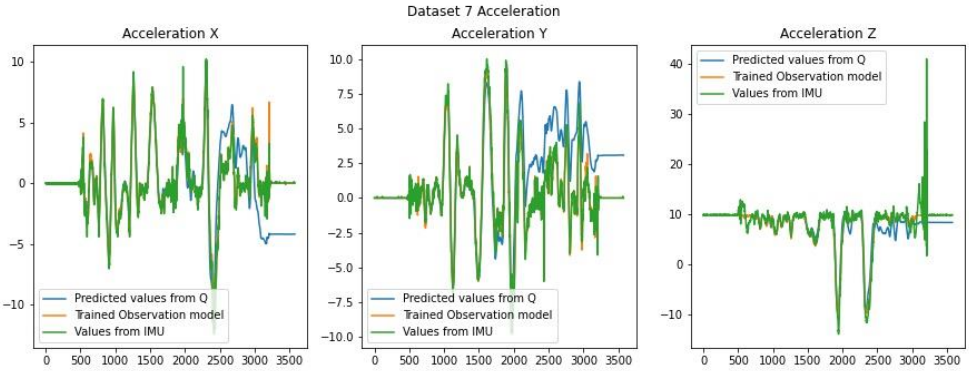
Dataset 6



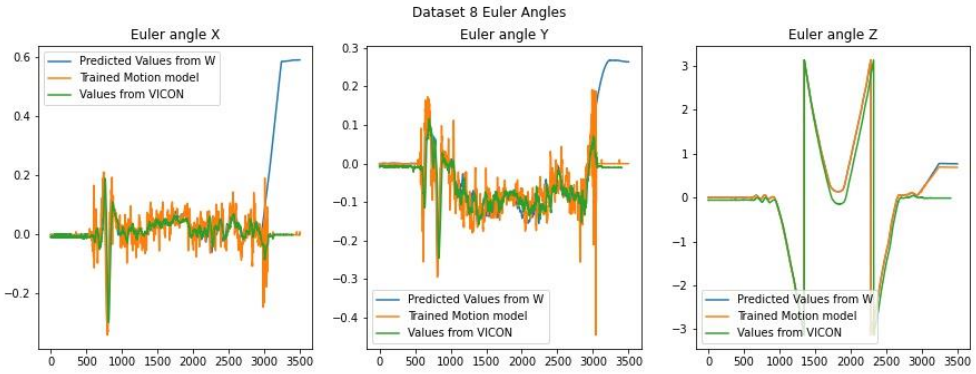


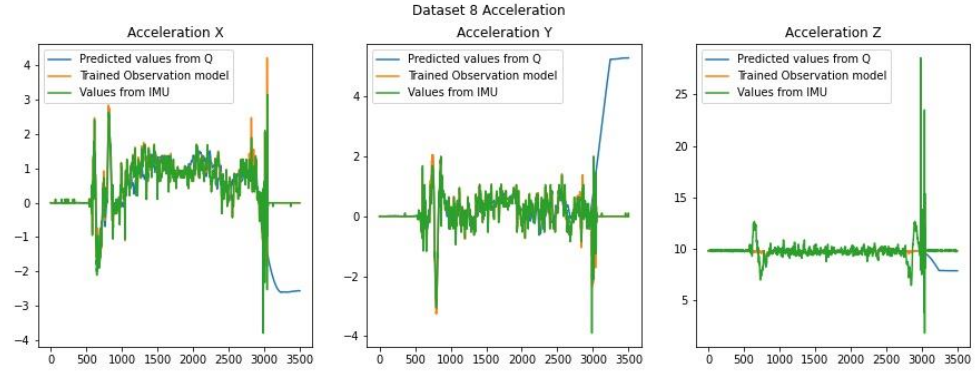
Dataset 7



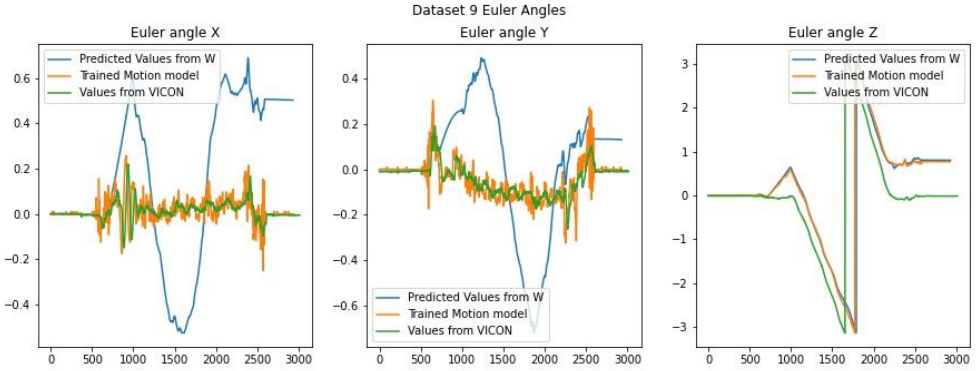


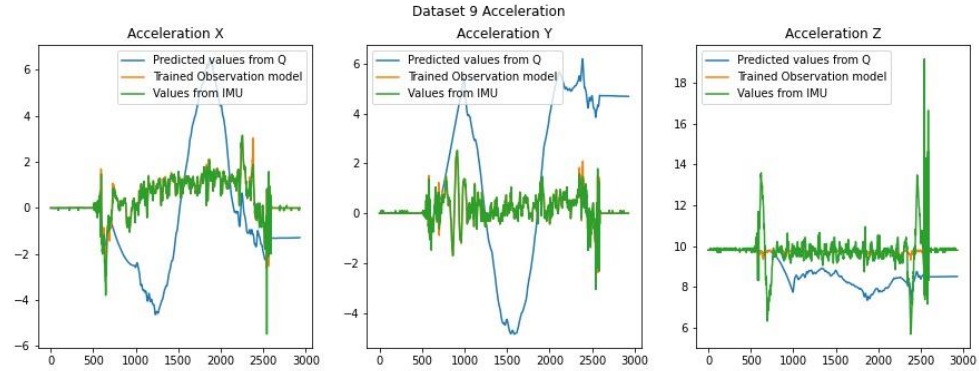
Dataset 8



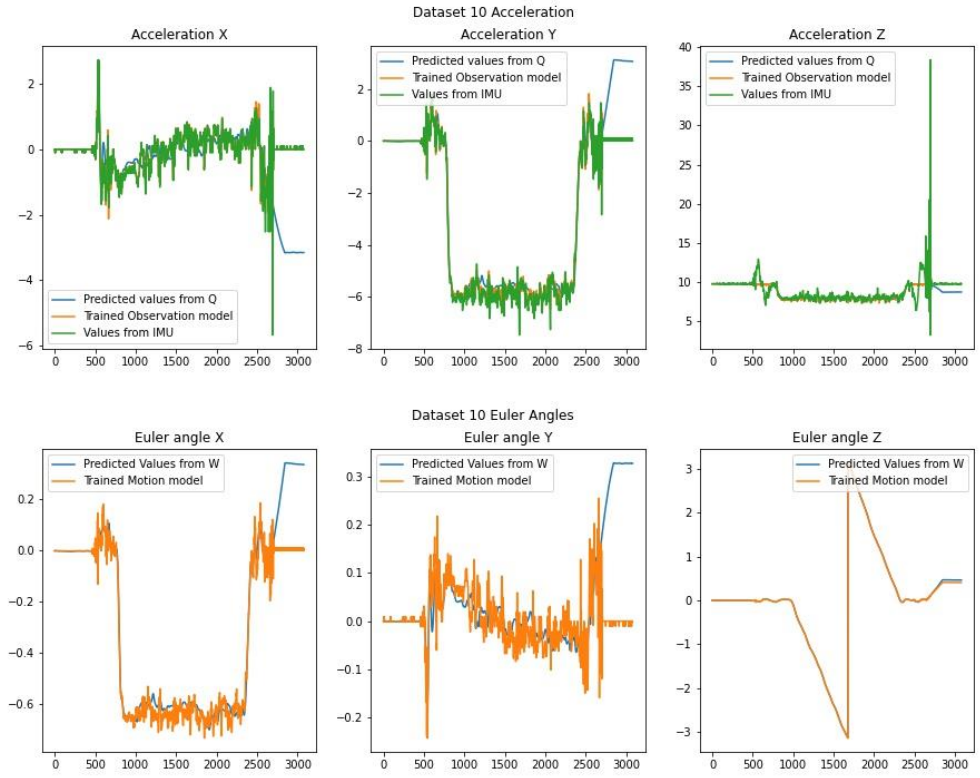


Dataset 9

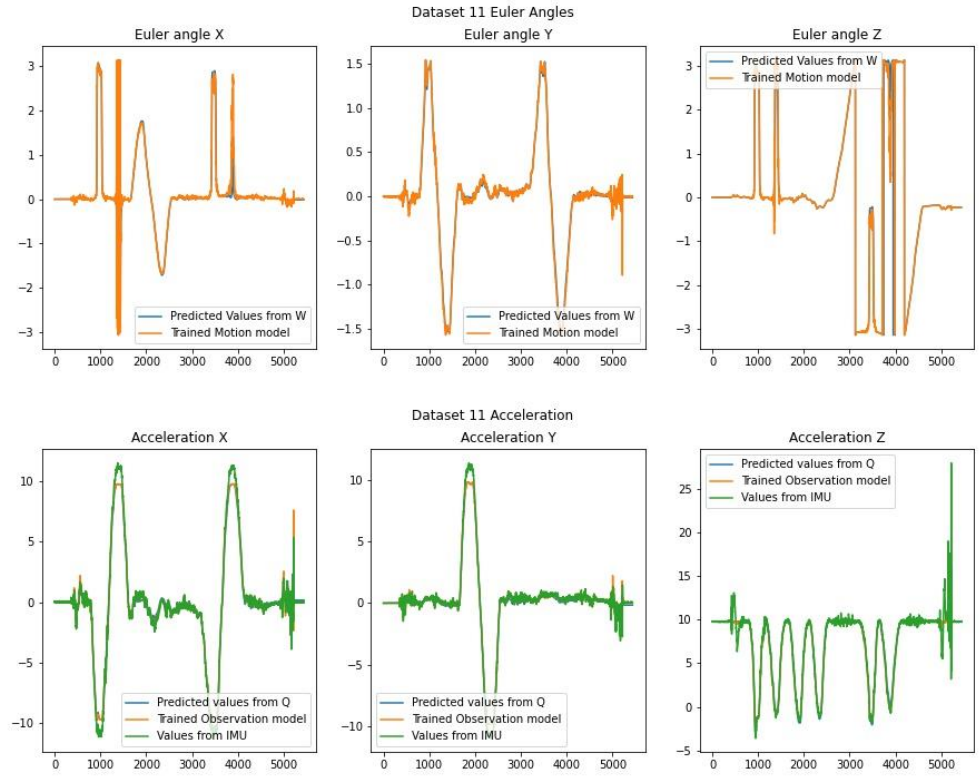




Dataset 10



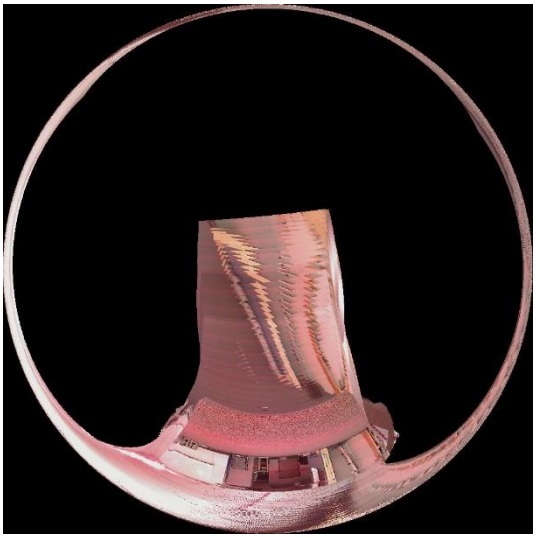
Dataset 11



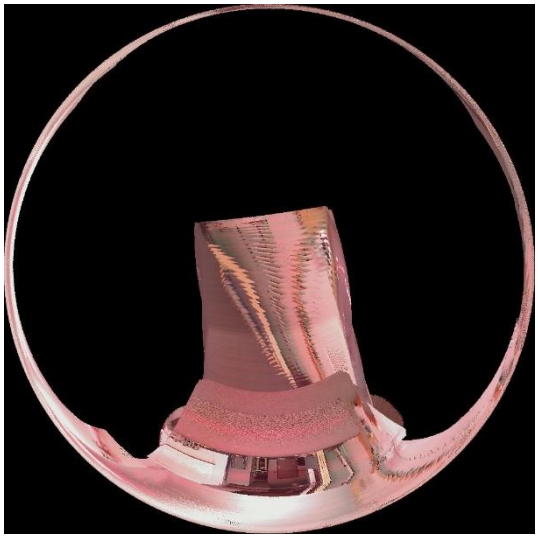
Panoramas: Lambert Azimuthal Projection

These projections provide a top down view of the room, as if a wide angle lens was installed at the top and the floor was being viewed. This results in a ring around at the corners and a relatively narrower field as we move down and towards the middle of the image.

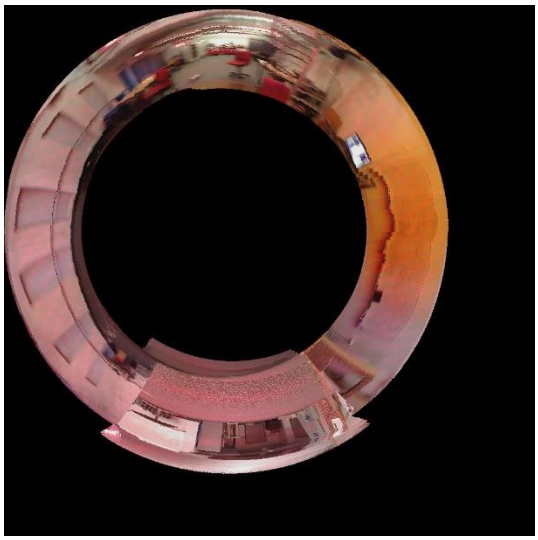
Dataset 1



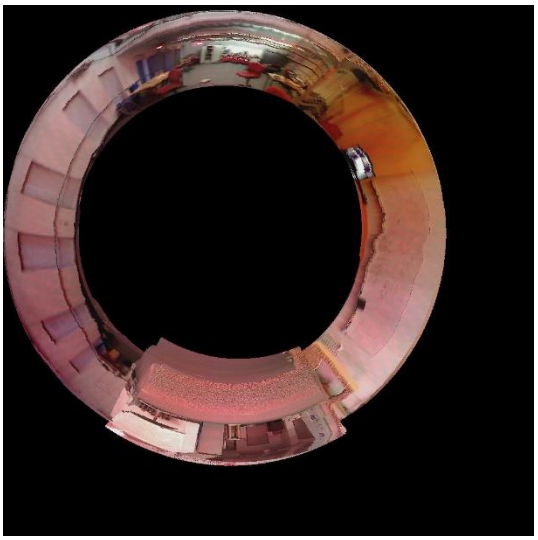
Dataset 2



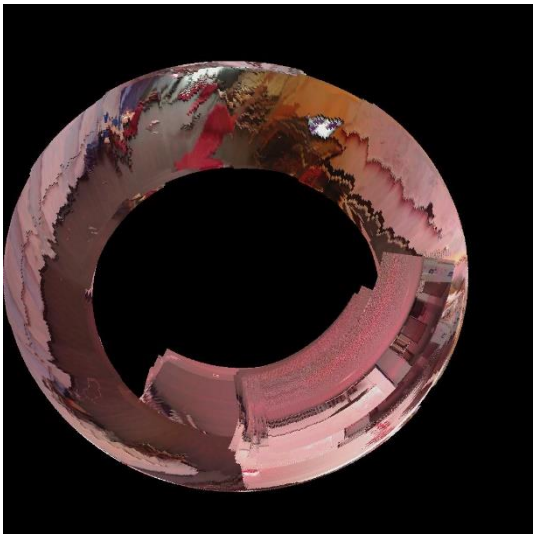
Dataset 8



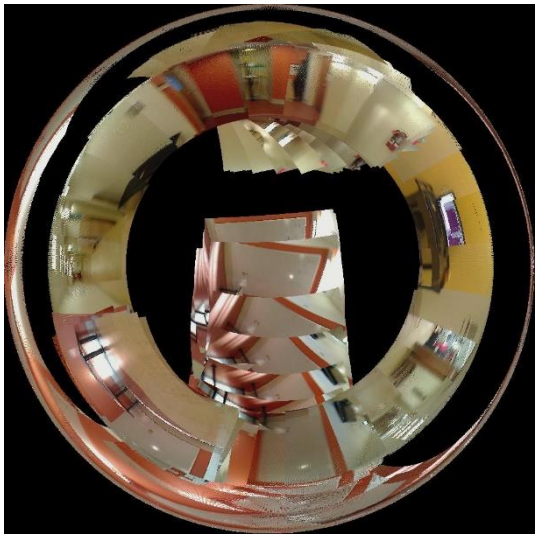
Dataset 9



Dataset 10

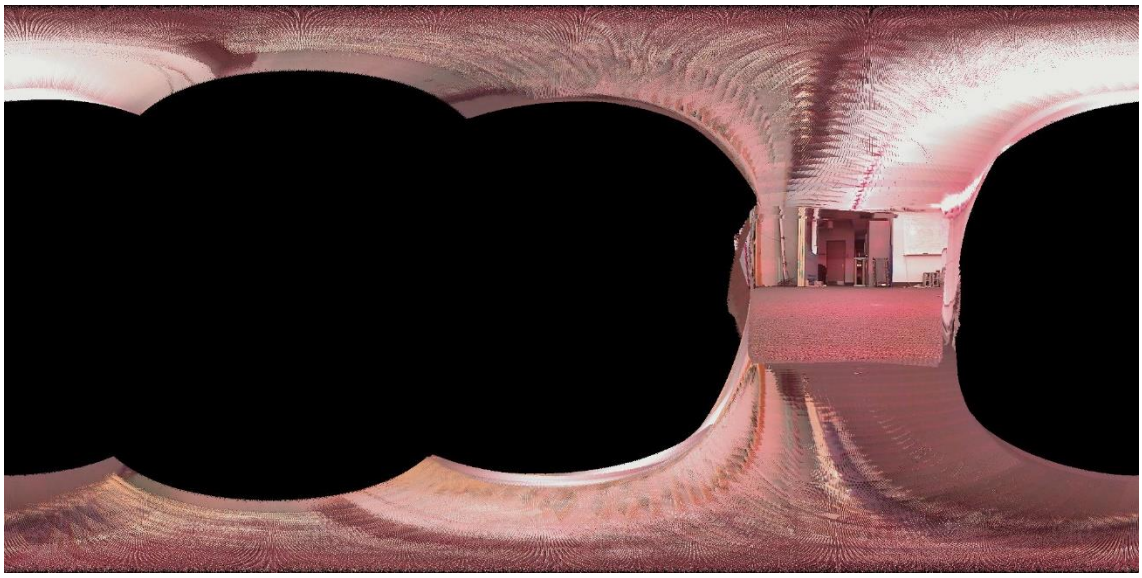


Dataset 11

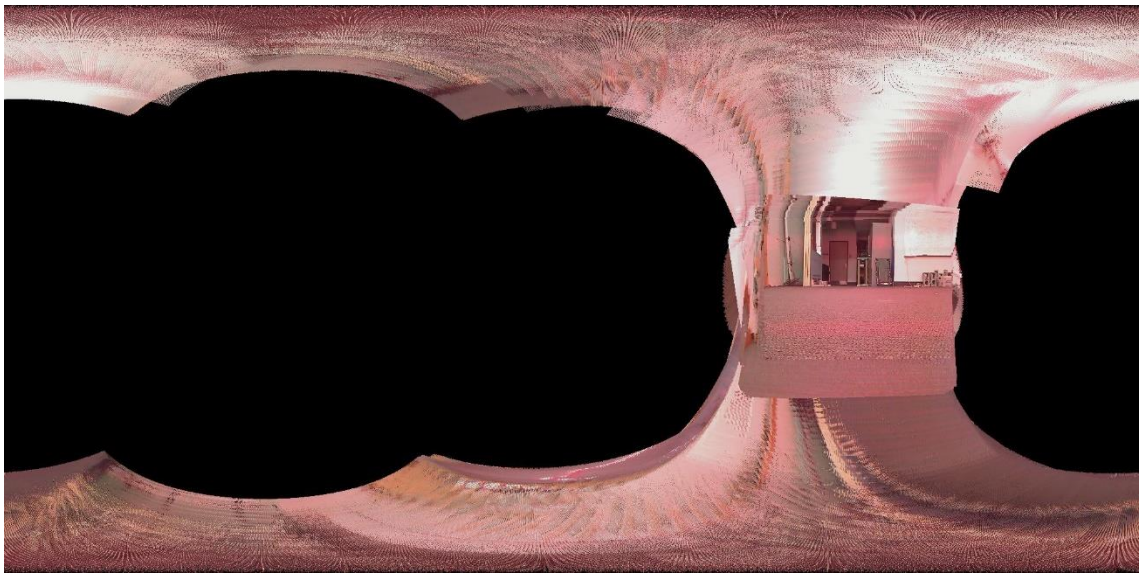


Panoramas: Cylindrical Coordinates

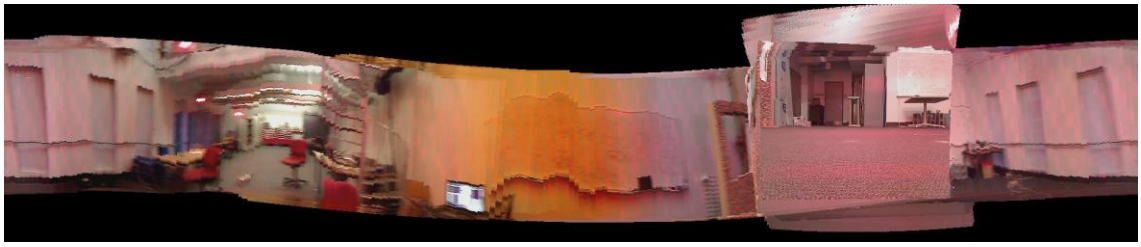
Dataset 1



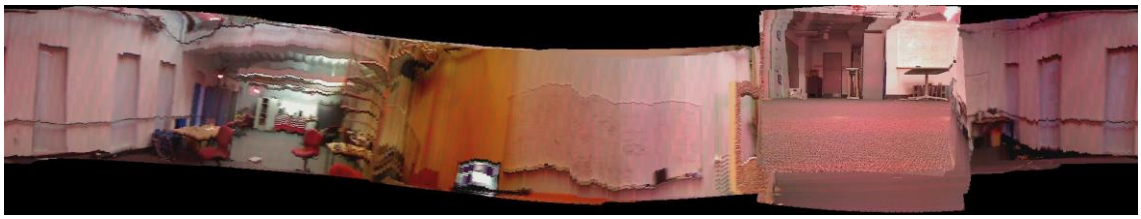
Dataset 2



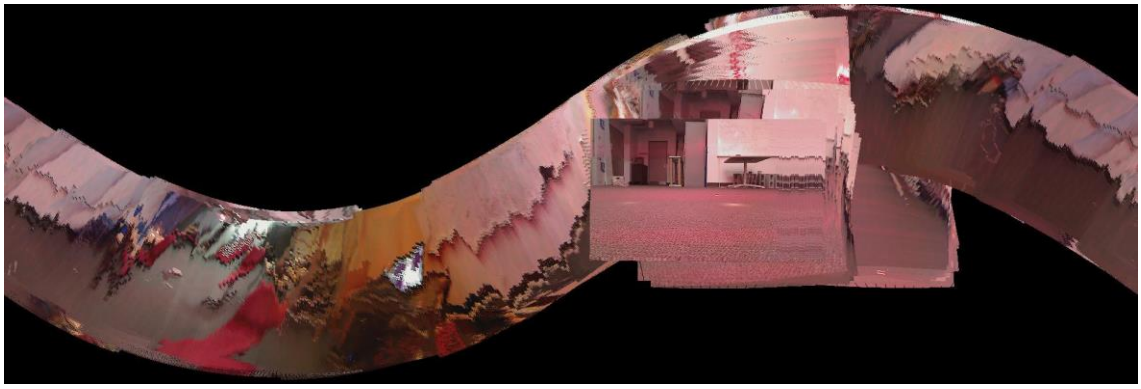
Dataset 8



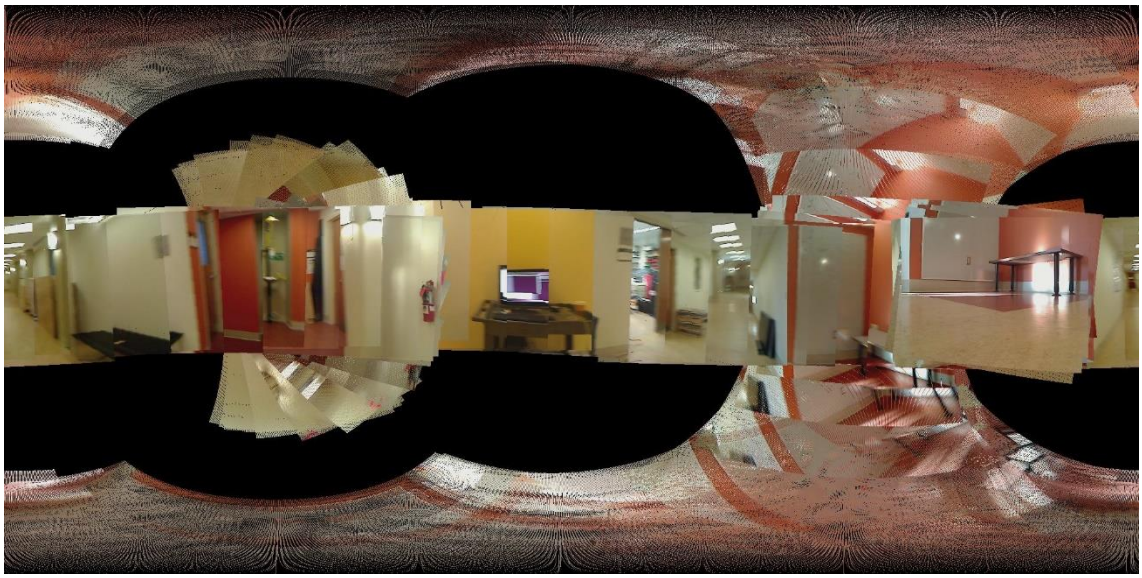
Dataset 9



Dataset 10

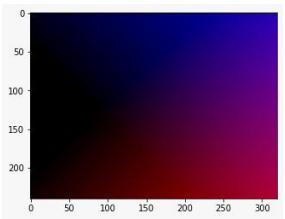


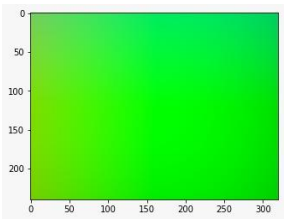
Dataset 11

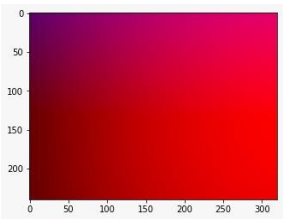


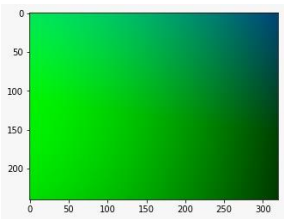
Rotation Matrices with World Frame Projection

When creating the panoramas, a good measure of the world coordinate mapping is displaying the matrices obtained as images. The world frame is an $240 \times 320 \times 3 \times N$ and this maps well to an RGB image. The resulting world frame gives some gradient images that signify the magnitude of rotation along the 3 axes. Some examples are here:









References:

1. Understanding observation model: https://automaticaddison.com/how-to-derive-the-observation-model-for-a-mobile-robot/
2. Formulae for coordinate conversions: https://math.libretexts.org/Bookshelves/Calculus/Book%3A_Calculus_(OpenStax)/12%3A_Vectors_in_Space/12.07%3A_Cylindrical_and_Spherical_Coordinates
3. Lambert azimuthal equal-area projection: https://en.wikipedia.org/wiki/Lambert_azimuthal_equal-area_projection
4. Lecture slides: https://natanaso.github.io/ece276a/schedule.html
5. Discussed Lambert Azimuthal projection with my classmate Prithwiraj Paul