Orientation Tracking

Narayanan Elavathur Ranganatha
University of California San Diego
nelavathurranganatha@ucsd.edu

***Abstract*—The following report formulates the problem of orientation estimation using data provided by a inertial measurement unit (IMU). The orientations are estimated as quaternions and *Projected Gradient Descent* is used to estimate the orientation for all time steps. These estimated orientations are then used to create a panorama of the scene.**

I. Introduction

A. Problem Statement

The objective of the problem is to estimate the orientation trajectory of a body that is kept on a rotating platform using the IMU sensor placed on the body. Then using this orientation trajectory we create a panorama by localising the images collected by the camera.

B. Motivation

This problem of trajectory estimation is very essential as a robot always needs to be aware of its surroundings. This gives the robot the power to estimate its state with respect to the world around it, and can therefore also measure changes that were not caused or were not meant to be caused by its control input, which therefore allows it to adapt its control input by taking into account any changes that might have happened in the world. The solution proposed in the report does not do online trajectory prediction but only after the entire trajectory is known, but extensions are possible to make the trajectory prediction online and causal in nature.

C. Approach

A gradient based approach is used for estimating the trajectory. The cost function stated in Eq 12 is used to find the gradients with respect to the trajectory.The cost is formulated using a motion and observation model to enforce consistency, and then projected gradient descent is used to reach the optimal trajectory that fits the data best based on the cost formulated. Once the optimal trajectory is estimated, we use spherical, cartesian and cylindrical coordinates to estimate the final panaromic image.

II. Problem Formulation

The objective of the problem is to estimate the orientation-trajectory of a body that is being rotated using the IMU that is attached to it. Let the unit quaternion $\mathbf{q}_t \in \mathbb{H}_*$ represent the body-frame orientation at time t. Our aim is to find the orientation-trajectory represented as follows:

$$\mathbf{q}_{1:T} = \mathbf{q}_1, \mathbf{q}_2, \dots \mathbf{q}_T \tag{1}$$

given the following information:

$$\boldsymbol{\omega}_{1:T} = \boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \dots \boldsymbol{\omega}_T \tag{2}$$

$$\mathbf{a}_{1:T} = \mathbf{a}_1, \mathbf{a}_2, \dots \mathbf{a}_T \tag{3}$$

where $\boldsymbol{\omega}_t$ represents the angular velocity of the body in the body-frame at time t and $\mathbf{a}_t$ represents the acceleration of the body in the body-frame at time t. These values are obtained from the IMU.

Then use the trajectory $\mathbf{q}_{1:T}$ to rotate each image accordingly and map it to a cylinder which is bounded to $[-\pi, \pi]$ horizontally and $[\frac{-\pi}{2}, \frac{\pi}{2}]$ and unwrap this to create a panoramic image.

III. Technical Approach

A motion and observation model is created using some priors that are known about the setup.

A. Motion Model

In this case we define motion model as a function f that relates the current state $\mathbf{x}_t$ and control input $\mathbf{u}_t$ to its state change depending on how much time has passed(τ_t). The current state and control input of the body is defined as follows:

$$\mathbf{x}_t = \begin{bmatrix}\mathbf{q}_t\end{bmatrix} \tag{4}$$

$$\mathbf{u}_t = \begin{bmatrix}\boldsymbol{\omega}_t\end{bmatrix} \tag{5}$$

Even though $\boldsymbol{\omega}_t$ is not actually the control input, it can still be abstracted as one as we are trying to predict its affect on the orientation.

Therefore we can write the motion model as follows:

$$\mathbf{x}_{t+\tau_t} = f(\mathbf{x}_t, \mathbf{u}_t, \tau_t) = f(\mathbf{q}_t, \boldsymbol{\omega}_t, \tau_t) = \mathbf{q}_t \circ exp(\begin{bmatrix}0, \tau_t \boldsymbol{\omega}_t\end{bmatrix}) \tag{6}$$

where $\circ$ is quaternion multiplication and $exp()$ is the exponential map for quaternions given below:

$$\mathbf{q} \circ \mathbf{p} = \begin{bmatrix} q_s p_s - \mathbf{q}_v^T \mathbf{p}_v & q_s \mathbf{p}_v + p_s \mathbf{q}_v + \mathbf{q}_v \times \mathbf{p}_v \end{bmatrix} \tag{7}$$

$$exp(\mathbf{q}) = e^{q_s} \begin{bmatrix} cos(||\mathbf{q}_v||) & \frac{\mathbf{q}_v}{||\mathbf{q}_v||} sin(||\mathbf{q}_v||) \end{bmatrix} \tag{8}$$

The motion model in this case is just assuming that we are rotating at a constant angular velocity $\boldsymbol{\omega}_t$(obtained via IMU) for a time τ_t(Time difference between two consecutive IMU measurements).

B. Observation Model

The observation model is a function h that relates the current state $\mathbf{x}_t$ to the environment m. The fact that the body's motion is only rotational and not translational is used to create the observation model. Therefore our absolute acceleration from the perspective of the environment should always be the same. This means that the acceleration reported by the IMU in the world frame should always be $\begin{bmatrix}0 & 0 & -g\end{bmatrix}$. Therefore if this vector is rotated using the inverse of the quaternion representing the current pose of the body-frame, it should match the acceleration reported by the IMU. This can be written mathematically as follows:

$$\mathbf{a}_t = h(\mathbf{q}_t) = \mathbf{q}_t^{-1} \circ \begin{bmatrix}0 & 0 & -g\end{bmatrix} \circ \mathbf{q}_t \tag{9}$$

C. Cost Function

Now that both the observation and motion models have been computed, these can be used to build a cost function. The cost function should ensure that the deviation from both these models is as less as possible and find a trajectory that is able to best fit both the models simultaneously.

1) Motion Model Cost: This term will calculate the error between our estimated orientation($\mathbf{q}_{t+1}$) and the prediction made using the motion model($f(\mathbf{q}_t, \boldsymbol{\omega}_t, \tau_t)$). We do this by multiplying the predicted quaternion by the inverse of the estimated quaternion, this should ideally give us the quaternion $\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}$. This quaternion is then converted to the angle axis parametrization using the $log(.)$ map, the norm of which given us the angle(should ideally be 0). This angle can be used as the error metric. Mathematically this can be written as follows:

$$Cost_m = \frac{1}{2}\sum_{t=0}^{T-1} ||2log(\mathbf{q}_{t+1}^{-1} \circ f(\mathbf{q}_t, \boldsymbol{\omega}_t, \tau_t))||_2^2 \tag{10}$$

We assume that $\mathbf{q}_0 = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}$ and we go only till $T-1$ as we don't have a estimate for the quaternion at $\mathbf{q}_{T+1}$.

2) Observation Model Cost: This term will calculate the error between the acceleration we get after transforming the acceleration $\begin{bmatrix}0 & 0 & -g\end{bmatrix}$ to the body-frame using $\mathbf{q}_t$ $(h(\mathbf{q}_t))$ and the acceleration measured by the IMU $\mathbf{a}_t$. We just take the $L_2 - norm$ between the two as the error. It is shown as follows:

$$Cost_o = \frac{1}{2}\sum_{t=1}^{T} ||\mathbf{a}_t - h(\mathbf{q}_t)||_2^2 \tag{11}$$

3) Final Cost: The final cost is just a summation of the above two costs:

$$Cost = c(\mathbf{q}_{1:T}) = Cost_m + Cost_o \tag{12}$$

D. Projected Gradient Descent

The aim is to minimize the above cost. This is a constrained optimization problem as the norm of a quaternion should always be 1. This can written mathematically as follows:

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

Normal gradient descent cannot be applied as that would not enforce the contraint of the quaternions being unit-norm. So therefore we use projected gradient descent.

Let $\mathbf{X}_k$ be the orientation trajectory after k steps of projected gradient descent optimization. The mathematical formulation for projected gradient descent is as follows:

$$\Pi_{\mathbb{H}_*}(\mathbf{x}) = arg\min_{y \in \mathbb{H}_*} ||y - x|| \tag{13}$$

$$\mathbf{X}_{k+1} = \Pi_{\mathbb{H}_*}(\mathbf{X}_k - \alpha \nabla c(\mathbf{X}_k)) \tag{14}$$

The function $\Pi_{\mathbb{H}_*}$ essentially ends up being equivalent to finding the unit vector in $\mathbb{R}^4$ that points in the same direction as the input. So essentially just normalizing the input vector. It can be visualized that the vector on the $4D$ hypersphere that is closest to any vector in $\mathbb{R}^4$ is just the unit-norm normalization of that vector.

E. Panorama

Once the orientation trajectory has been estimated, this can then be used to create the panorama.

Here is a summary of the pipeline followed by a elaboration of each part.

- Convert Image Coordinates to Spherical coordinates assuming radius 1
- Convert the Spherical Coordinates to Cartesian Coordinates
- Rotate the Cartesian coordinates using the orientation estimated for that image
- Convert the new rotated Cartesian coordinates to Spherical coordinates.
- Convert the Spherical Coordinates to Cylindrical Cordinates.
- Unwrap the Cylinder to form the panorama image.

The axis orientations are assumed as shown in Fig 1 while doing the conversion.

1) Image to Spherical Coordinates: This is the part where we convert the image to as if it lied on a sphere of radius r. This can be done by estimating 2 angles. We consider the vector joining the origin and the point on the sphere. The angle λ represents the angle that the projection of this vector on the $xy - plane$ makes with the $x - axis$ as shown in Fig 2. The angle ϕ is the angle that the vector makes with the z-axis as shown in the Fig 2. Each pixel is converted into its corresponding spherical coordinate $(\lambda, \phi, 1)$. We assume the horizontal Field-of-View(FOV) is $\frac{\pi}{3}$ and vertical FOV is $\frac{\pi}{4}$. We have the image height and width $(H, W) = (240, 320)$. Let's say we are looking at pixel (u, v), So we have:

$$\lambda = v\frac{\pi}{3 * 320} - \frac{\pi}{6} \tag{15}$$

$$\phi = u\frac{\pi}{4 * 240} - \frac{\pi}{8} \tag{16}$$

2) Spherical to Cartesian Coordinates: The Spherical coordinates can be converted to cartesian coordinates while keeping in mind the convention stated above using the following

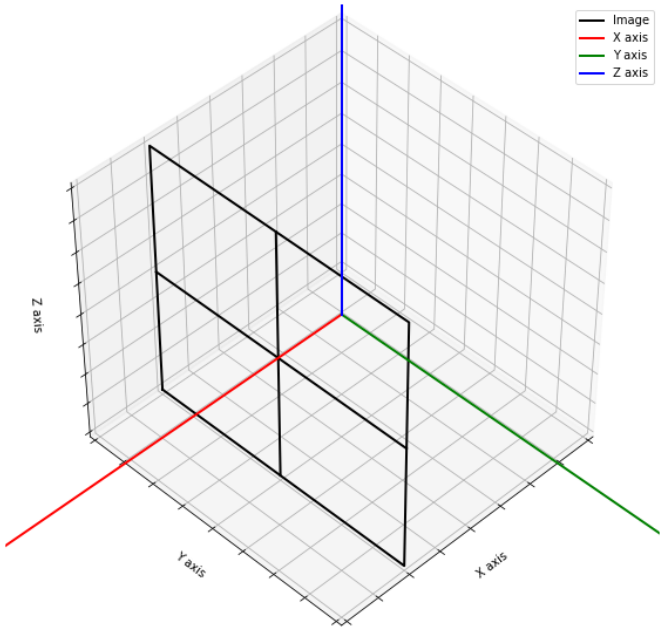

Fig. 1. Axis convention followed for coordinate frame conventions

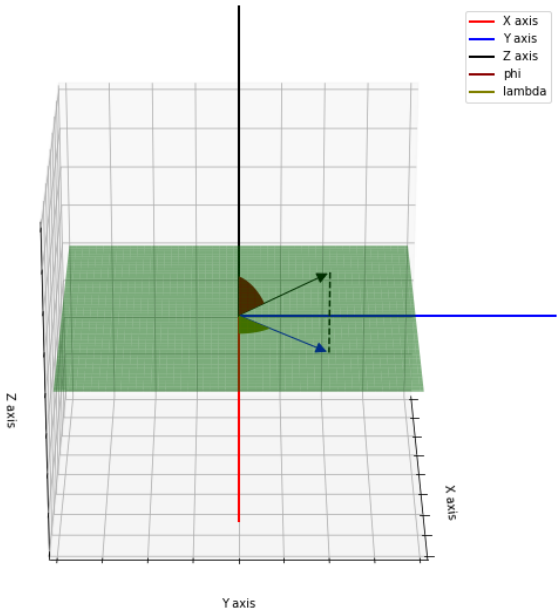

Fig. 2. λ and ϕ visualized

formulas. Spherical coorrdinates (λ, ϕ, r) can be written as cartesian coordinates (x, y, z) with:

$$x = rcos(\phi)cos(\lambda) \tag{17}$$

$$y = -rcos(\phi)sin(\lambda) \tag{18}$$

$$z = -rsin(\phi) \tag{19}$$

3) Rotate the Cartesian Coordinates with quaternion $\mathbf{q}_t$: So if we have the cartesian coordinates $\mathbf{x} = (x, y, z)$, we can rotate it to $\mathbf{x}^{`} = (x^{`}, y^{`}, z^{`})$ as:

$$[0, \mathbf{x}^{`}] = \mathbf{q}_t \circ [0, \mathbf{x}] \circ \mathbf{q}_t^{-1} \tag{20}$$

4) Cartesian to Spherical Coordinates: If we have the cartesian coordinates (x, y, z), we can get the spherical coordinates $(\lambda, \phi, 1)$(following the axis convention mentioned) with the following computation:

$$\lambda = tan^{-1}(\frac{y}{x}) \tag{21}$$

$$\phi = tan^{-1}(\frac{-z}{\sqrt{x^2 + y^2}}) \tag{22}$$

5) Spherical to Cylindrical: Now fortunately, (λ, ϕ) are a sufficient representation in cylindrical coordinates where ϕ represents the height of the pixel in the image and λ represents the location of the pixel horizontally. λ goes from $-\pi$ to π and ϕ goes from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. Depending on the number of pixels desired in the output, we can discretise this and map to pixels.

F. IMU Calibration

The first few seconds of each IMU file is used to find the bias of the IMU. We apply the following calculations to calculate the bias.

$$bias = (Actual - Expected) \tag{23}$$

where expected value is $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ for the gyroscope and $\begin{bmatrix} 0 & 0 & -g \end{bmatrix}$ for the acceleration outputted by the IMU. The IMU reference provided is used to calculate the Actual values.

IV. Results

A. Qualitative Results - Training Set

The graphs for the initial, estimated and ground-truth(GT)(obtained via VICON) are plotted to see the match and improvement after applying projected gradient descent from Fig 3 - Fig 11.

The panoramas can be found from Fig 12 - Fig 19. The results contain panoramas generated by both VICON data and out estimated data.

The panoramas for 8 and 9 have many bad features and are much less smoother than the VICON ones. An explanation for why this might be the case is listes in Observations.

B. Quantitative Results - Training Set

The relative rotation error also known as the Geodesic Distance between two rotation matrices can be used to validate the quaternions by comparing them to the closest timestamped VICON rotation matrix. The formula for the following is as follows for two rotation matrices P and Q:

$$RRE = arccos(\frac{trR - 1}{2}) \tag{24}$$

$$R = PQ^T \tag{25}$$

The results can be found in the Table I. The error is in radians.

File	RRE (radians)
1	0.2506
2	0.2585
3	0.2684
4	0.6797
5	0.3141
6	0.4254
7	0.2722
8	0.5129
9	0.5160

TABLE I
RELATIVE ROTATION ERROR FOR ALL TRAIN FILES

C. Qualitative Results - Test Set

The graphs for the initial and estimated trajectories are plotted to see the match and improvement after applying projected gradient descent from Fig 20 - Fig 21.

The panoramas can be found from Fig 22 - Fig 23. The results contain panoramas generated by the estimated data.

The test set panoramas seem good in general. But it is difficult to say without VICON data for the same.

D. Observations

- Changing the weights between the motion model and observation model costs does not affect the result. Many weights were tried and all converged to similar results. Removing one cost completely on the other hand does lead to suboptimal results.
- The panorama creation is very sensitive to even tiny mistakes in coordinate conversion. The entire result is thrown off if you miss one "−" sign.
- At $\alpha >= 0.5$ the cost begins to explode rather than reduce which is expected as the updates are too large and so really big steps are taken on the quaternion hypersphere.
- The RRE for file 9 is very high and when the observations are analyzed via rotplot, it is seen that the initial motion is really jerky, and leads to a certain constant offset being introduced in the yaw. It is as if the predicted yaw is always playing catch-up to real one but is never able to reach it. It can also therefore be seen that the panorama difference between the one generated using VICON and one generated using estimates is also very large and may be due to this initial jerky motion. This can be explained by very fast changes in ω_t within the τ_t time taken to get the next measurement. Therefore the data obtained from the IMU is not a true reflection of the actual angular velocity.
- Vectorizing the code is very important, otherwise the code is extremely slow.

V. FUTURE IMPROVEMENTS

- The model can be made causal in nature, such that only $t-1$ steps of data is available when predicting $\mathbf{q}_t$. This would make this algorithm practically applicable on a real robot online.

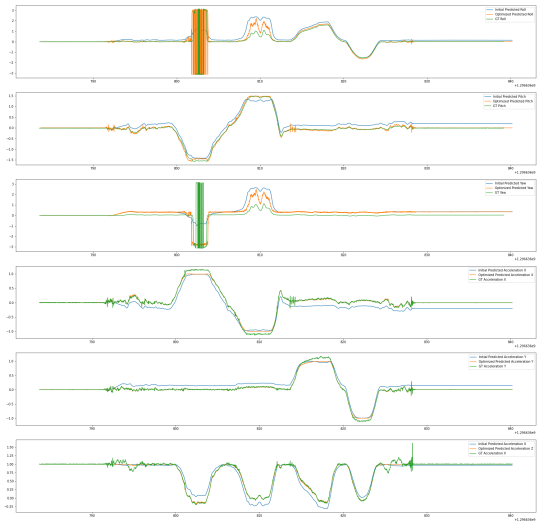

Fig. 3. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial, estimated, GT orientation-trajectories for 1

- Pixel averaging can be done. Right now the code just overwrites the pixel with the latest value that it receives. Averaging might lead to a more cleaner panorama.

VI. ACKNOWLEDGEMENTS

I would like to thank the TAs and Professor Nikolay for their guidance throughout the project. I would also like to mention that I discussed ideas and compared panorama results with Dwait Bhatt

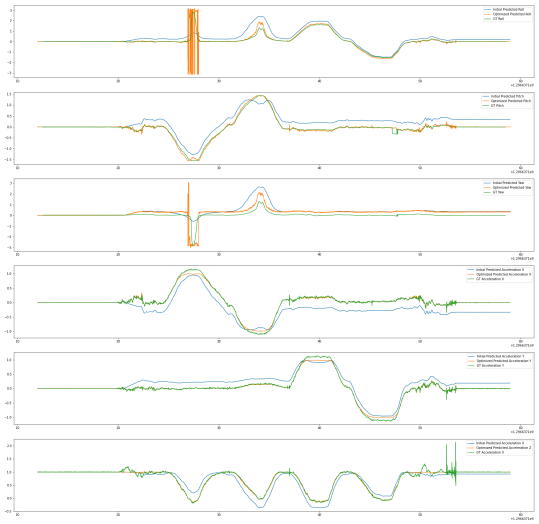

Fig. 4. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial, estimated, GT orientation-trajectories for 2

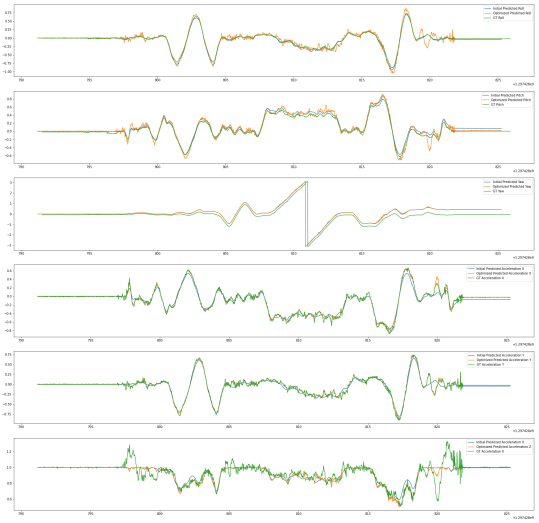

Fig. 5. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial, estimated, GT orientation-trajectories for 3

Fig. 6. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial, estimated, GT orientation-trajectories for 4

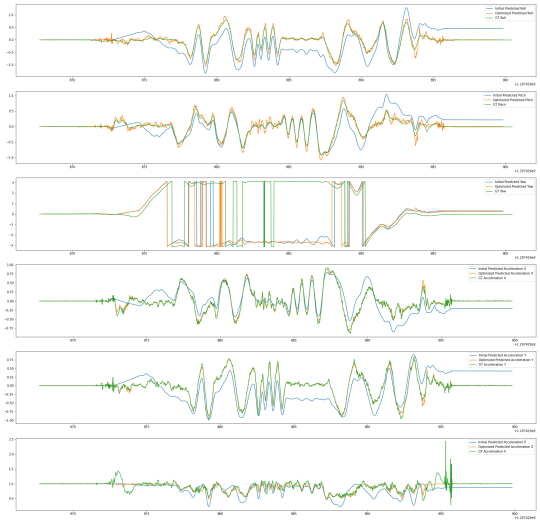

Fig. 7. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial, estimated, GT orientation-trajectories for 5

Fig. 8. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial, estimated, GT orientation-trajectories for 6

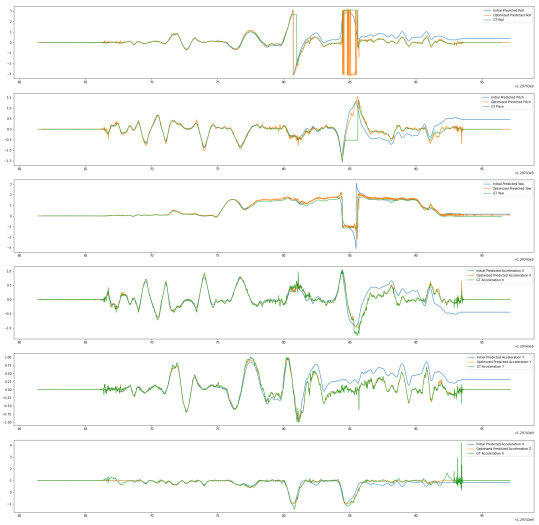

Fig. 9. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial, estimated, GT orientation-trajectories for 7

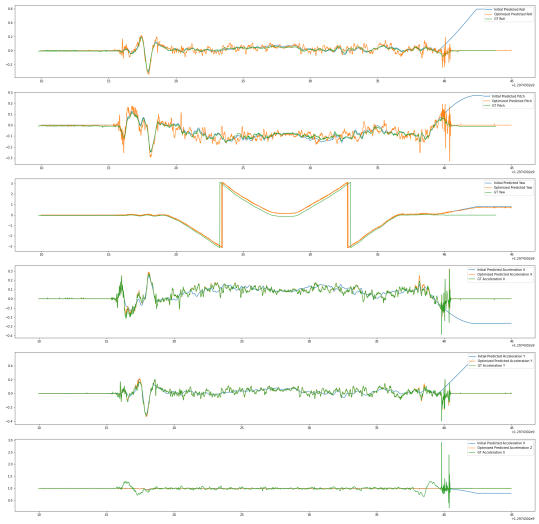

Fig. 10. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial, estimated, GT orientation-trajectories for 8

Fig. 11. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial, estimated, GT orientation-trajectories for 9

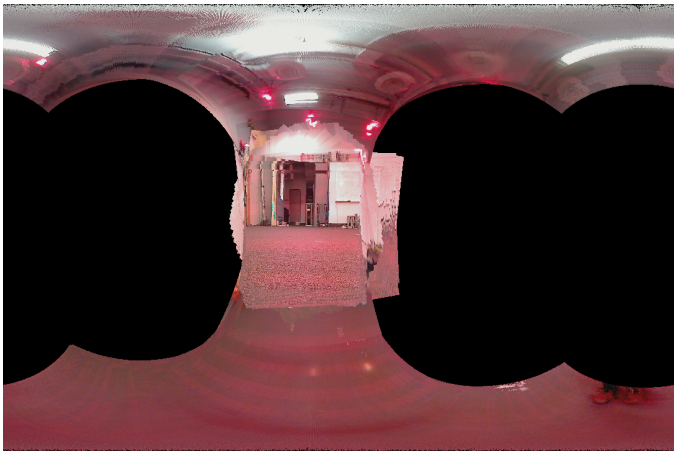

Fig. 12. panorama generated using estimated quaternions for file 1

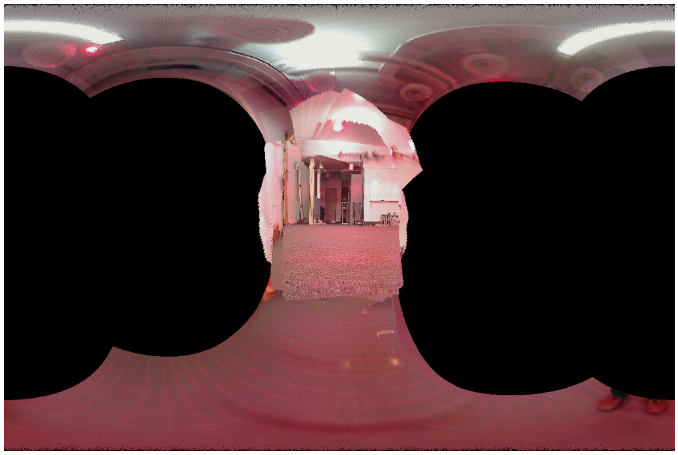

Fig. 15. panorama generated using VICON measurements for file 2

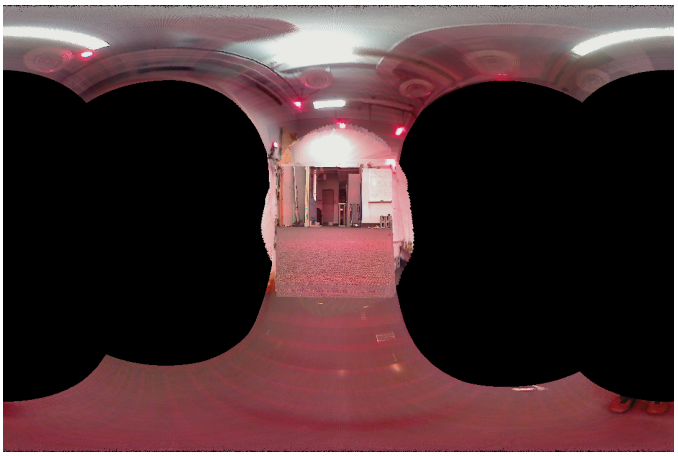

Fig. 13. panorama generated using VICON measurements for file 1

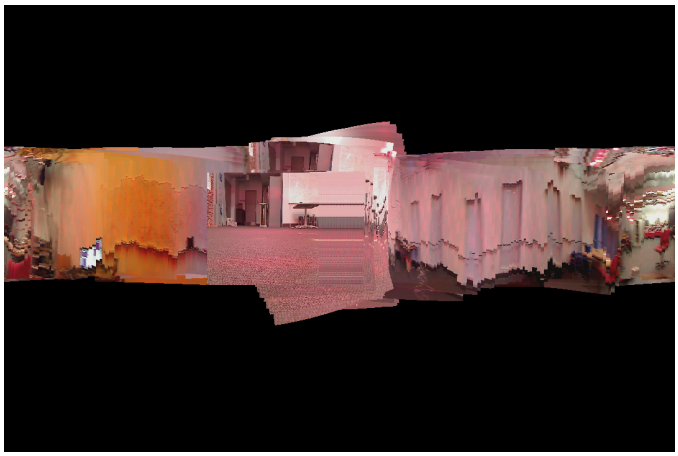

Fig. 16. Panorama generated using estimated quaternions for file 8

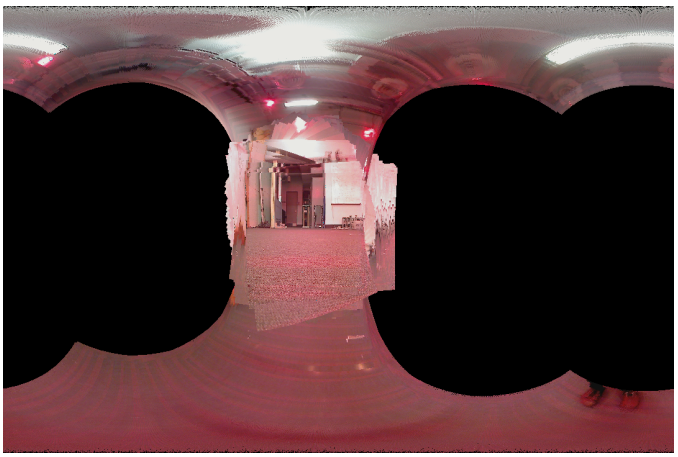

Fig. 14. panorama generated using estimated quaternions for file 2

Fig. 17. Panorama generated using VICON measurements for file 8

Fig. 18. Panorama generated using estimated quaternions for file 9

Fig. 20. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial and estimated trajectories for 1 - Test Set

Fig. 19. Panorama generated using VICON measurements for file 9

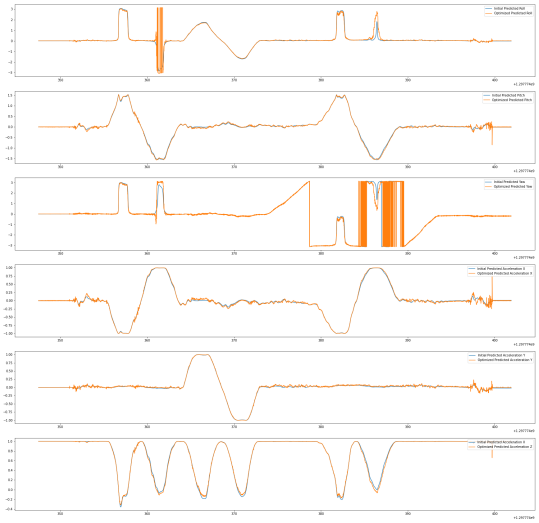

Fig. 21. Roll, Pitch, Yaw, Acceleration along X, Y and Z axis for initial and estimated trajectories for 2 - Test Set

Fig. 22. Panorama generated using estimated quaternions for file 1 - Test Set

Fig. 23. Panorama generated using estimated quaternions for file 2 - Test Set