Orientation Tracking and Panoramic Image Stitching using IMU Data

Anderson Compalas
Department of Electrical and Computer Engineering
University of California San Diego
La Jolla, California
acompalas@ucsd.edu

***Abstract*—This project implements a projected gradient descent algorithm to track the 3D orientation of a rotating body using inertial measurement unit (IMU) data. The estimated orientation is then used to stitch camera images into panoramic views. The approach combines gyroscope angular velocity measurements with accelerometer observations of gravity direction to optimize a quaternion-based trajectory. The motion model integrates angular velocity through quaternion kinematics, while the observation model enforces consistency with gravity acceleration. Results demonstrate successful orientation tracking on training datasets with close agreement to VICON ground truth, and effective panorama generation on both training and test datasets.**

***Index Terms*—IMU calibration, orientation tracking, quaternions, projected gradient descent, panorama stitching**

I. Introduction

Accurate orientation tracking is fundamental to many robotics and computer vision applications such as autonomous navigation. Inertial Measurement Units (IMUs) provide high-frequency measurements of angular velocity and linear acceleration, making them essential sensors for estimating body orientation.

This project addresses the challenge of estimating 3D orientation from biased IMU data and using these estimates to construct panoramic images from a rotating camera. The problem is important because: (1) orientation tracking enables navigation when GPS is unavailable, (2) panoramic imaging provides wide field-of-view scene understanding, and (3) the fusion of gyroscope and accelerometer data demonstrates sensor fusion principles.

Our approach consists of two main components. First, we calibrate the IMU to remove sensor biases and implement a projected gradient descent optimization algorithm that estimates the orientation trajectory by minimizing errors in both motion prediction and gravity observation. We represent orientation using unit quaternions, which avoid singularities and provide computational efficiency. Second, we use the estimated orientations to transform camera images into a spherical coordinate system, stitching them into panoramic views.

The technical challenges include: handling biased sensor measurements, maintaining quaternion unit-norm constraints during optimization, and aligning camera images with potentially imperfect orientation estimates. We validate our approach on nine training datasets with VICON ground truth and two test datasets.

II. Problem Formulation

A. Orientation Representation

We represent the body orientation at time t using a unit quaternion $q_t \in \mathbb{H}^*$, where $\mathbb{H}^*$ denotes the set of unit quaternions with $\|q_t\|_2 = 1$. A quaternion is defined as $q_t = [w, x, y, z]^T$ where w is the scalar part and $[x, y, z]^T$ is the vector part.

B. Motion Model

The quaternion kinematics motion model predicts orientation change based on angular velocity $\omega_t \in \mathbb{R}^3$ (rad/sec) measured by the gyroscope:

$$q_{t+1} = f(q_t, \tau_t \omega_t) := q_t \circ \exp([0, \tau_t \omega_t / 2]) \tag{1}$$

where τ_t is the time step, $\circ$ denotes quaternion multiplication, and $\exp(\cdot)$ is the quaternion exponential function.

C. Observation Model

Since the body undergoes pure rotation, the acceleration in the world frame should be $[0, 0, -g]^T$ m/s^2 (gravity only). The measured acceleration $a_t \in \mathbb{R}^3$ in the IMU frame should agree with this after transformation:

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

where $[0, 0, 0, 1]$ represents gravity in quaternion form (in gravity units).

D. Optimization Problem

We formulate orientation tracking as a constrained optimization problem over the trajectory $q_{1:T} = \{q_1, q_2, \ldots, q_T\}$:

$$\begin{aligned} \min_{q_{1:T}} \quad & c(q_{1:T}) = \frac{1}{2} \sum_{t=0}^{T-1} \|2 \log(q_{t+1}^{-1} \circ f(q_t, \tau_t \omega_t))\|_2^2 \\ & + \frac{1}{2} \sum_{t=1}^{T} \|[0, a_t] - h(q_t)\|_2^2 \\ \text{s.t.} \quad & \|q_t\|_2 = 1, \quad t = 1, \ldots, T \end{aligned} \tag{3}$$

The first term measures motion model error (rotation angle between predicted and estimated orientation), while the second term measures observation model error (mismatch between measured and predicted acceleration).

III. Technical Approach

A. IMU Calibration

1) Bias Estimation: The first few seconds of each dataset are static (no rotation). During this period:

- Accelerometer should read $[0, 0, 1]^T$ in gravity units
- Gyroscope should read $[0, 0, 0]^T$ rad/sec

We estimate biases by averaging measurements during the static period (first 3 seconds). The calibrated measurements are:

$$\omega_{calibrated} = \omega_{raw} - \text{bias}_\omega \tag{4}$$

$$a_{calibrated} = a_{raw} - \text{bias}_a \tag{5}$$

Accelerometer readings are then normalized to unit vectors to match the observation model.

2) Gyro Integration Verification: To verify calibration, we integrate angular velocity starting from $q_0 = [1, 0, 0, 0]$ using the motion model (Eq. 1). We convert quaternions to roll-pitch-yaw angles and compare with VICON ground truth. Good calibration shows reasonable agreement to ground truth data.

B. Projected Gradient Descent

1) Initialization: We initialize the trajectory using gyroscope integration:

- $q_0 = [1, 0, 0, 0]$ (identity)
- q_t for $t > 0$ computed via Eq. (1) with integrated gyroscope

This provides a good starting point near the optimal solution, improving convergence.

2) Gradient Computation: We use PyTorch's automatic differentiation to compute gradients of the cost function with respect to all quaternions:

$$\nabla_{q_{1:T}} c(q_{1:T}) \tag{6}$$

The quaternion operations (multiplication, inverse, exponential, logarithm, normalization) are implemented as differentiable PyTorch functions.

3) Optimization Loop: For each iteration:

1) Compute cost function $c(q_{1:T})$ using current trajectory
2) Compute gradient $\nabla_{q_{1:T}} c(q_{1:T})$ via autograd
3) Update: $q_{1:T} \leftarrow q_{1:T} - \alpha \nabla_{q_{1:T}} c(q_{1:T})$
4) Project onto unit quaternion manifold: $q_t \leftarrow q_t / \|q_t\|_2$

Hyperparameters: learning rate $\alpha = 0.01$, maximum iterations = 5000, convergence tolerance = 10^{-7}.

4) GPU Acceleration: The implementation uses vectorized operations on GPU (CUDA) for significant speedup. All quaternion trajectories and operations are batched, allowing parallel processing of thousands of time steps.

C. Panorama Stitching

1) Camera Model: The camera has:

- Field of view: 60° horizontal, 45° vertical (total)
- Resolution: 320 × 240 pixels
- Optical axis aligned with IMU x-axis

Focal lengths are computed as:

$$f_x = \frac{W}{2\tan(\text{FOV}_h/2)} \tag{7}$$

$$f_y = \frac{H}{2\tan(\text{FOV}_v/2)} \tag{8}$$

2) Cylindrical Projection: We use cylindrical projection to map camera images onto a panorama. This approach wraps viewing directions around a vertical cylinder, avoiding the distortion at top and bottom that occurs with spherical projections.

For each pixel (u, v) in a camera image:

1) Compute ray in camera frame (optical axis = z):

$$\mathbf{r}_{cam} = \begin{bmatrix} (u - c_x)/f_x \\ (v - c_y)/f_y \\ 1 \end{bmatrix} \tag{9}$$

2) Transform to IMU frame: $\mathbf{r}_{IMU} = R_{cam \to IMU} \cdot \mathbf{r}_{cam}$ where the rotation matrix aligns camera z-axis with IMU x-axis
3) Transform to world frame using quaternion:

$$\mathbf{r}_{world} = q_t \circ [0, \mathbf{r}_{IMU}] \circ q_t^{-1} \tag{10}$$

4) Convert to cylindrical coordinates:

$$\text{azimuth} = \arctan 2(y_{world}, x_{world}) \quad \in [-\pi, \pi] \tag{11}$$

5) Map to panorama image coordinates:

$$u_{pano} = \frac{\text{azimuth} + \pi}{2\pi} \times W_{pano} \tag{12}$$

$$v_{pano} = \frac{1 - (z_{world} + 1)/2}{1} \times H_{pano} \tag{13}$$

where the vertical position is determined directly from the z-coordinate, mapping $z \in [-1, 1]$ to the full panorama height.

3) Timestamp Alignment: For each camera image timestamp, we find the closest-in-the-past IMU orientation estimate. This ensures we use orientation data that was available at image capture time.

4) Image Stitching: We use simple overwriting: later pixel values replace earlier ones at the same panorama location. No blending or color averaging is performed.

IV. Results

A. Training Data Performance

1) IMU Calibration Verification: Gyroscope-only integration shows good agreement with VICON ground truth for roll and pitch angles across all training datasets. Yaw exhibits expected drift due to accumulating integration errors. This validates our bias estimation approach.

Bias estimation results were consistent across all datasets with gyroscope biases around [373.5, 375.4, 369.6] counts and accelerometer biases around [-511, -501, 606] counts. The static calibration window of 300 samples (approximately 3 seconds) proved sufficient for stable bias estimates. Figures 1 through 3 show the Euler angle plots for gyro only integration of the calibrated IMU data versus VICON data for datasets 1-9.

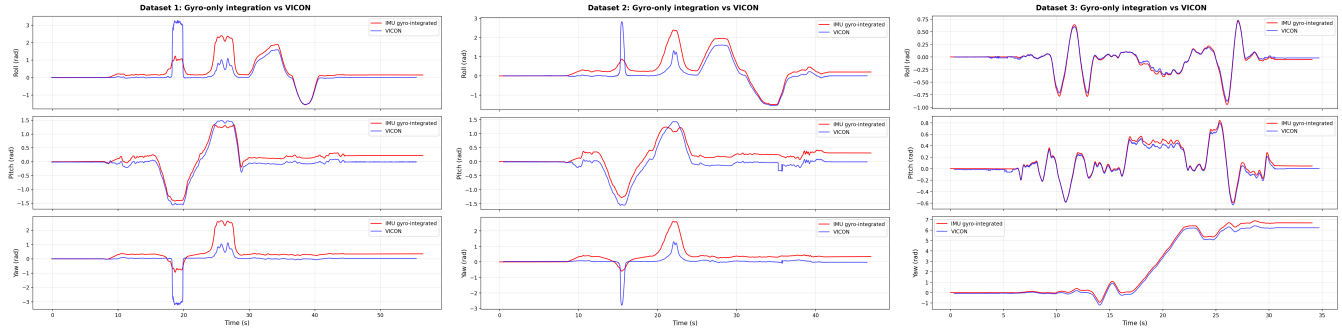

Fig. 1. IMU calibration verification: Gyro-only integration vs VICON ground truth for training datasets 1-3.

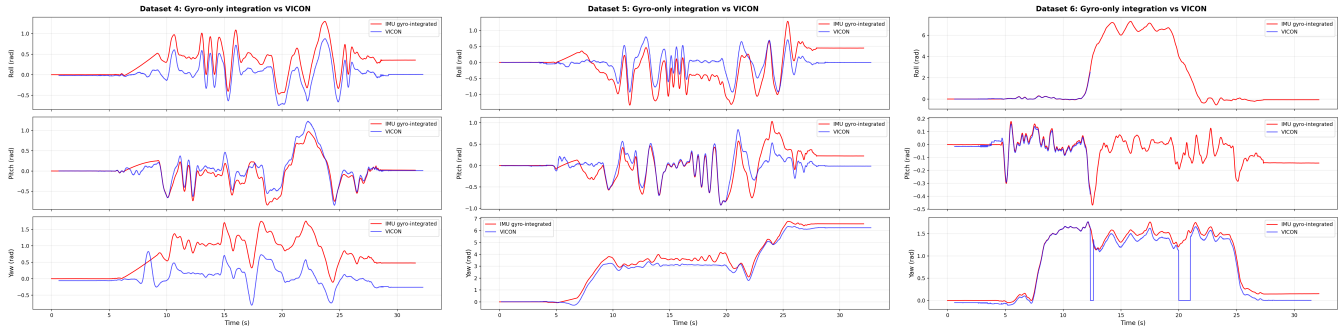

Fig. 2. IMU calibration verification for training datasets 4-6.

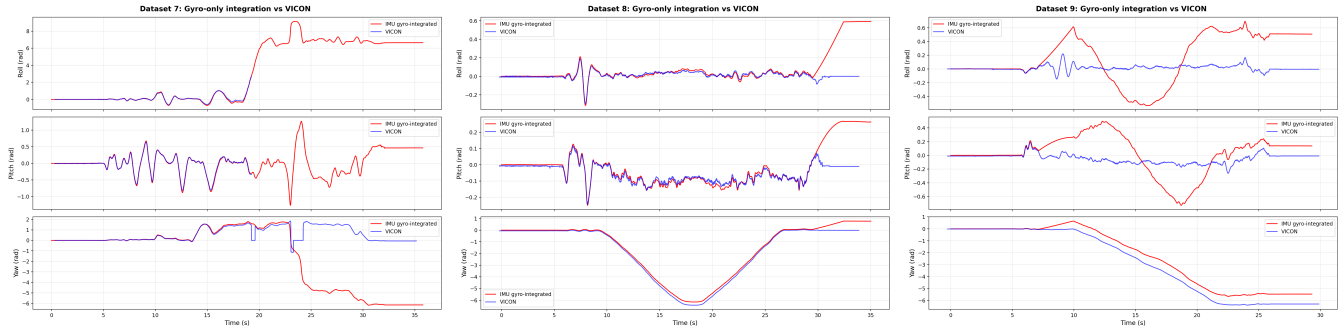

Fig. 3. IMU calibration verification for training datasets 7-9.

2) Optimized Orientation Tracking: After running projected gradient descent optimization incorporating accelerometer data:

- **Roll and Pitch**: Excellent agreement with VICON ground truth, typically within ± 0.1 radians. The accelerometer observation model successfully constrains these angles.
- **Yaw**: Slightly deteriorated compared to gyro-only integration, but this is expected since accelerometer measurements cannot observe yaw rotation (both world and body gravity vectors remain in the z-direction regardless of yaw).

Table I summarizes the optimization performance for all datasets. The optimization was run with a maximum of 5000 iterations, learning rate of 0.01, and convergence tolerance of 10^{-7}. Most training datasets converged well before reaching the iteration limit, demonstrating effective optimization. The initial cost values varied significantly (11.2 to 301.6) depending on the complexity and duration of each trajectory, but all datasets achieved final costs below 1.5, representing cost reductions of 1-3 orders of magnitude.

Key observations from the convergence analysis:

- **Fast Convergence**: Four datasets (1, 6, 7, 11) converged in under 3000 iterations, with dataset 11 achieving convergence in only 712 iterations.
- **Iteration Limit**: Four datasets (2, 3, 4, 5) reached the 5000 iteration limit, though their final cost changes were small (10^{-5} to 10^{-4}), indicating they were close to convergence.
- **Initial Cost Variation**: Initial costs ranged from 11.2 (datasets 3 and 11) to 301.6 (dataset 9), reflecting differences in trajectory complexity and the quality of

TABLE I
OPTIMIZATION CONVERGENCE ANALYSIS

Dataset	Split	Samples	Converged At Iter	Initial Cost	Final Cost
1	Train	5645	2057	161.41	0.434
2	Train	4698	5000*	281.36	0.561
3	Train	3404	5000*	11.17	1.187
4	Train	3156	5000*	142.37	1.089
5	Train	3210	4811	271.05	1.242
6	Train	3211	2776	84.81	0.836
7	Train	3577	1978	230.66	1.466
8	Train	3501	2717	75.55	0.401
9	Train	2931	4117	301.65	0.328
10	Test	3078	3576	37.75	0.315
11	Test	5441	712	11.79	1.182

*Reached maximum iterations without converging to tolerance

gyroscope-based initialization.

- **Consistent Final Performance**: Despite varying convergence rates, all datasets achieved low final costs (0.315-1.466), demonstrating robust optimization across different datasets.

Figures 4 through 6 show optimized Euler angle orientation estimates compared to VICON ground truth for all training datasets.

3) Training Panoramas: Panoramas generated for datasets 1, 2, 8, and 9 show clear room structure with walls, ceiling, and floor visible in proper orientation (floor at bottom, ceiling at top). Black regions indicate unobserved directions, expected due to limited camera FOV during rotation. Some waviness and jitter are present but within reasonable expectations. Figure 7 shows panoramas generated using optimized trajectories.

B. Test Data Performance

1) Orientation Estimates: For test datasets 10 and 11 (no VICON ground truth available), optimization converged successfully with favorable characteristics:

- **Dataset 10**: Converged at iteration 3576 with final cost 0.315, starting from initial cost 37.75. The relatively low initial cost indicates good gyroscope-based initialization.
- **Dataset 11**: Demonstrated the fastest convergence of all datasets, reaching the tolerance threshold in only 712 iterations. Final cost of 1.182 from initial 11.79 represents a 90% reduction.

Both test trajectories show smooth, continuous motion with no divergence or numerical instability. The fast convergence of dataset 11 is particularly noteworthy, as it is the longest test sequence with 5441 samples spanning 54.43 seconds. Figure 8 shows the estimated roll, pitch, and yaw angles for test datasets.

2) Test Panoramas: Test panoramas demonstrate good quality with recognizable room structure. Dataset 10 shows a wide continuous panorama, while dataset 11 provides clear features of the images (Figure 9).

C. Comparison with VICON Panoramas

As an above-and-beyond analysis, we generated panoramas using VICON ground truth orientations for training datasets (Figure 10). Comparison shows that our optimized trajectories produce panoramas qualitatively similar to VICON-based panoramas. The main differences occur in yaw (expected), causing slight horizontal shifts in the panorama features such as those observed in the panorama of data set 2 (see Figure 7.) The overall structure and vertical alignment match well, confirming the accurate estimation of roll and pitch.

D. Computational Performance

GPU acceleration provides significant speedup and enables practical real-time processing:

- **Data Loading**: 4.4-13.9 seconds per dataset (includes file I/O and initial processing)
- **Optimization Time**: 5-50 seconds depending on trajectory length and convergence rate
- **Total Processing**: 1-2 minutes per dataset end-to-end on NVIDIA GPU with CUDA
- **GPU vs CPU**: Approximately 50-100× faster than CPU-only implementation

The vectorized PyTorch implementation with float32 precision on CUDA enables batch processing of thousands of quaternions simultaneously, significantly reducing training time. The automatic differentiation provided by PyTorch eliminates the need for manual gradient derivation while maintaining computational efficiency.

V. DISCUSSION

A. What Worked Well

1) **IMU Calibration**: Simple averaging during static periods effectively removes sensor biases. The short static period (3 seconds) is sufficient for stable bias estimates.
2) **Quaternion Representation**: Unit quaternions avoid gimbal lock and provide smooth, singularity-free optimization. The projection step (normalization) is simple and effective.
3) **Gyroscope Integration Initialization**: Starting near the solution dramatically improves convergence compared to identity initialization for all quaternions.
4) **GPU Vectorization**: Batched PyTorch operations enable accelerated training. Automatic differentiation simplifies gradient computation for complex quaternion operations.
5) **Roll and Pitch Estimation**: Accelerometer observations strongly constrain these angles, achieving near-ground-truth accuracy.

B. Challenges and Limitations

1) **Yaw Drift**: Accelerometer cannot observe yaw rotations about the gravity vector. This fundamental observability limitation could possibly be addressed with magnetometer data.
2) **Panorama Artifacts**: Waviness and jitter result from orientation errors, discrete time sampling, and simple overwriting. More sophisticated stitching could possibly improve visual quality.

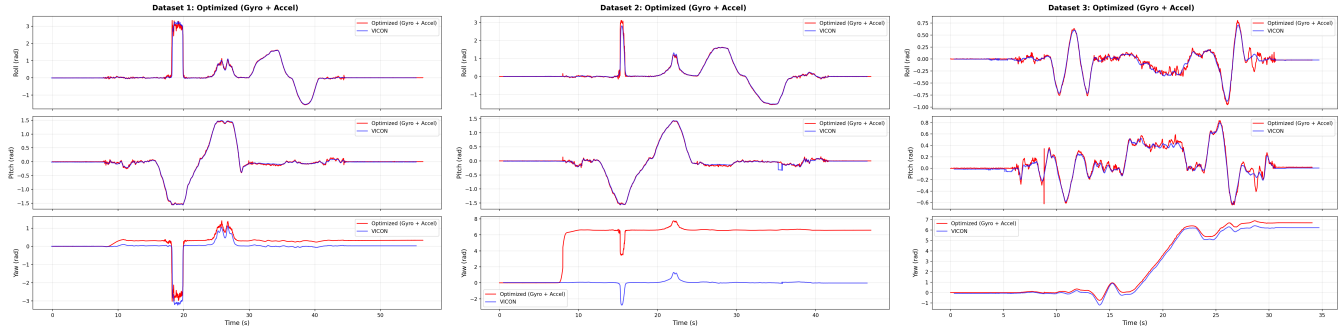

Fig. 4. Optimized orientation (gyro + accelerometer) vs VICON ground truth for training datasets 1-3.

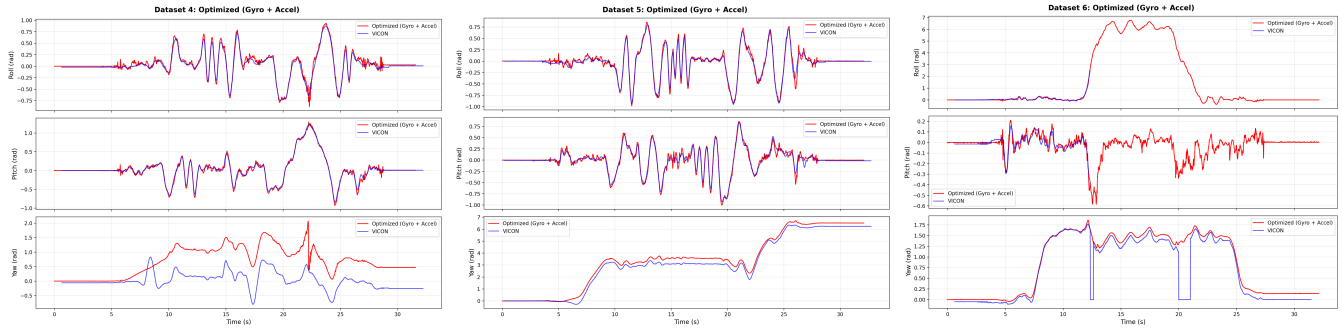

Fig. 5. Optimized orientation vs VICON ground truth for training datasets 4-6.

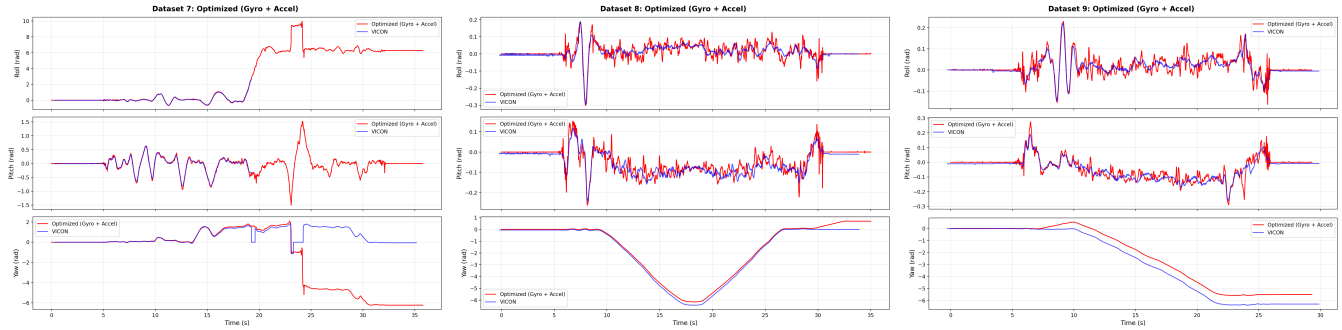

Fig. 6. Optimized orientation vs VICON ground truth for training datasets 7-9.

3) **Hyperparameter Tuning**: Learning rate and iteration count were chosen empirically. More sophisticated hyperparameter search methods could possibly improve robustness.

C. Lessons Learned

1) **Initialization Matters**: Good initialization (gyro integration) is important for non-convex optimization. Starting from all-identity quaternions leads to poor convergence.
2) **Coordinate Frame Consistency**: Careful tracking of coordinate transformations (camera, IMU, world) is essential. We debugged an upside-down panorama issue by correcting world frame z-axis orientation.
3) **Vectorization**: Modern deep learning frameworks (PyTorch) designed for neural networks were excellent for training acceleration with automatic differentiation and parallelization.

VI. Conclusion

This project successfully demonstrated orientation tracking from IMU data using projected gradient descent optimization on quaternion manifolds. The approach achieves accurate roll and pitch estimation by fusing gyroscope kinematics with accelerometer gravity observations. The estimated orientations enable panoramic image stitching that reconstructs wide field-of-view scenes from narrow camera images.

Key achievements include:

- Effective IMU calibration through bias estimation
- Robust optimization with quaternion constraints
- GPU-accelerated implementation for efficient training

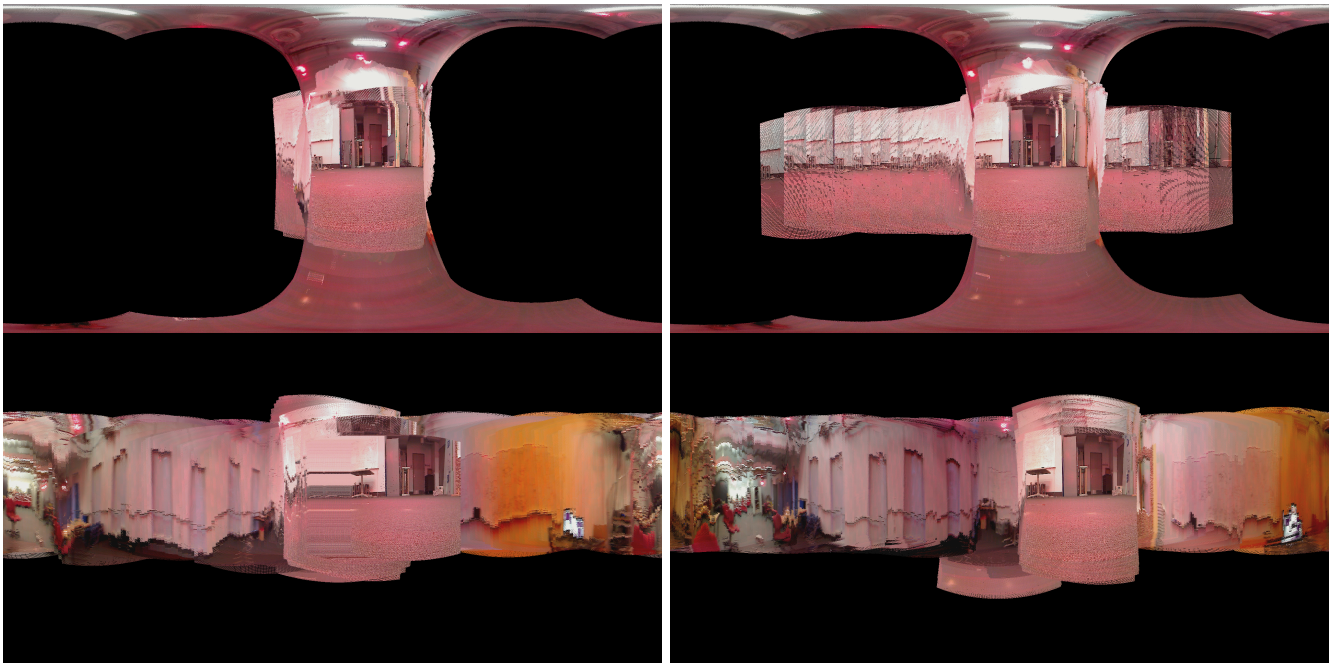

Fig. 7. Panoramas generated using optimized trajectories for training datasets 1, 2, 8, and 9 (top to bottom).

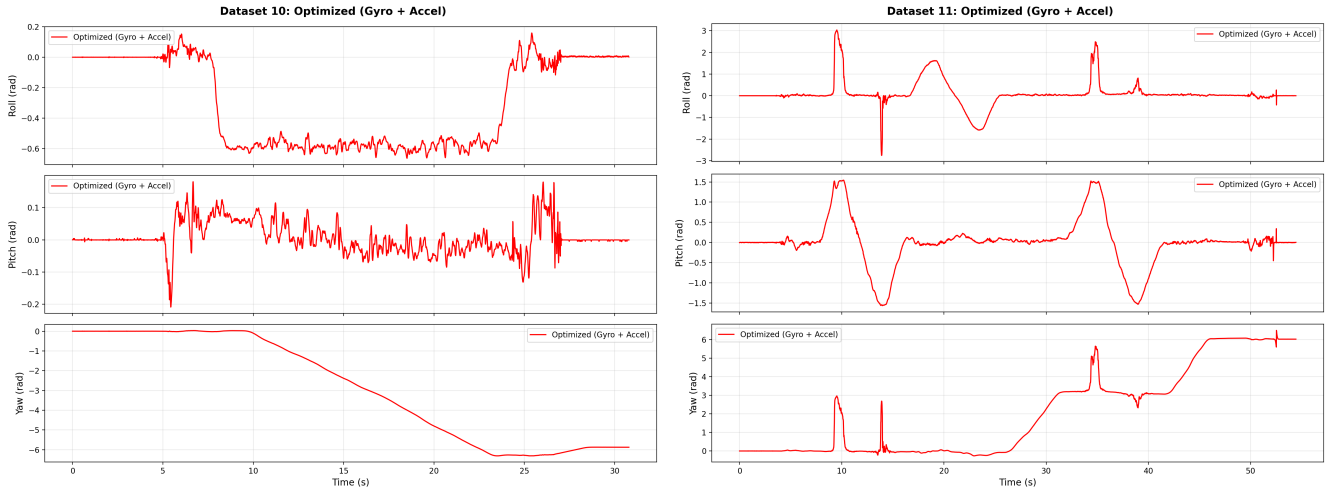

Fig. 8. Estimated orientation trajectories for test datasets 10 (left) and 11 (right). No ground truth available.

Fig. 9. Panoramas generated for test datasets 10 (left) and 11 (right).

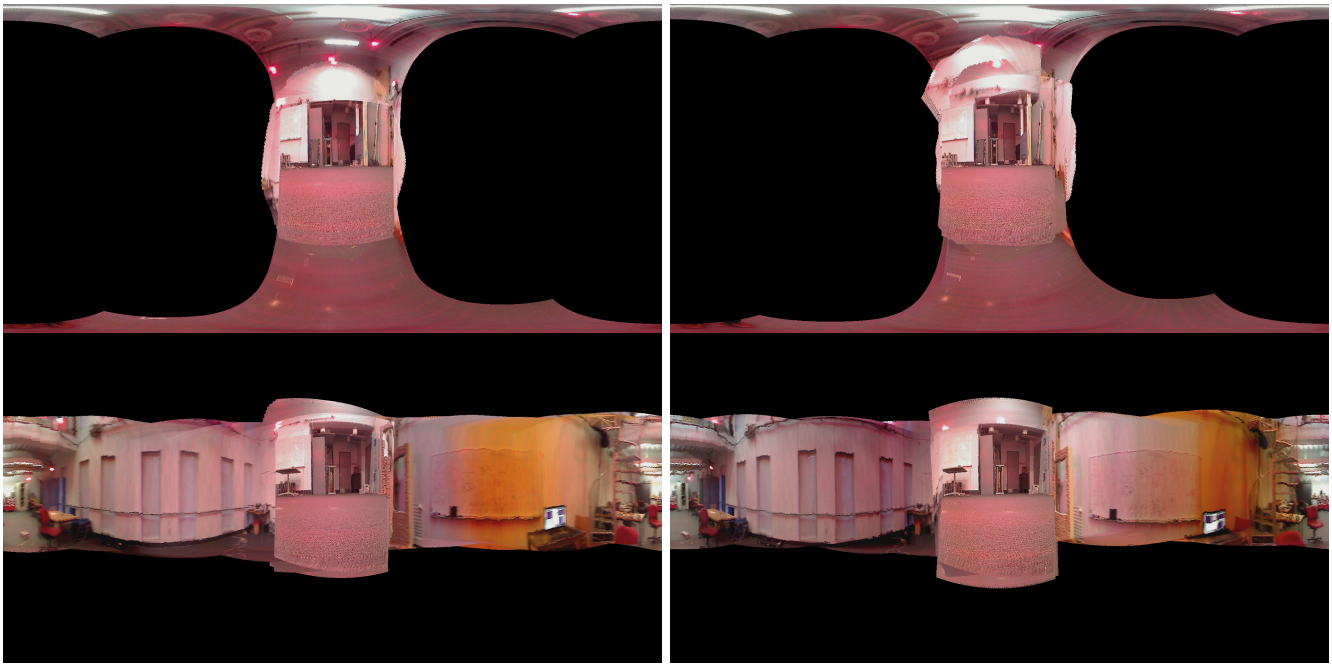

Fig. 10. Panoramas generated using VICON ground truth for training datasets 1, 2, 8, and 9 (top to bottom). Compare with Figure 7.

- Successful panorama generation on both training and test data

ACKNOWLEDGMENTS

We thank Professor Nikolay Atanasov and the ECE 276A teaching assistants for providing the project framework, datasets, and guidance.

REFERENCES

[1] N. Atanasov, "ECE 276A: Sensing & Estimation in Robotics - Project 1," University of California San Diego, 2026.