Supplementary Materials for Princeton365: A Diverse Dataset with Accurate Camera Pose

Karhan Kayan*, Stamatis Alexandropoulos*, Rishabh Jain, Yiming Zuo, Erich Liang, Jia Deng
Princeton University
{karhan,sa6924,jainr,zuoym,el1685,jiadeng}@princeton.edu

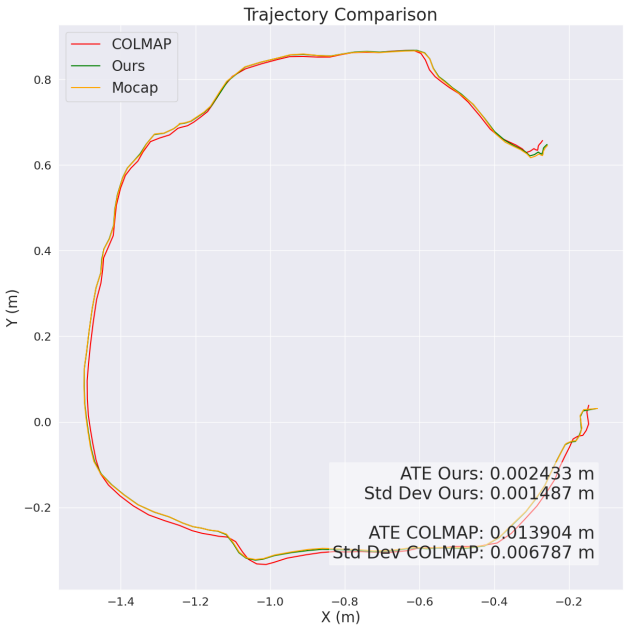


Figure 1. Comparison of our ground truth trajectory with COLMAP and Vicon MoCap system. Our ground-truth has better accuracy than COLMAP.

1. Sensor Setup Details

To record the RGB and depth data, the ZED X camera requires hardware that can support high data throughput. We utilize the NVIDIA Jetson Orin NX along with a GMSL2 Fakra cable to serve this purpose. The NVIDIA Jetson requires a dedicated power source to read data from the ZED X camera. However, the camera rig also needs to be portable and functional in a variety of indoor and outdoor environments, making it infeasible to keep the NVIDIA Jetson plugged into a power outlet at all times during recording. As a result, we used an external, rechargeable battery pack and soldered together a new power adapter to connect it to the NVIDIA Jetson. We then secured this setup, along with an external SSD, to a laser cut acrylic board to prevent components becoming loose while recording. We placed the acrylic board into a backpack with a small opening at the top for the GMSL2 Fakra cable to connect to the camera outside the backpack. See Main Paper Fig. 3 for a picture of the setup.

2. Bundle Rig PnP

Formally, Bundle Rig PnP minimizes the sum of reprojection errors across all cameras and time as

$$\min_{\mathbf{T}^o_{B_i},\mathbf{T}^o_{rig,t},\mathbf{T}^c_{rig}} \sum_{t=1}^{T}\sum_{c}\sum_{d}(\pi(\mathbf{T}^c_{rig}\mathbf{T}^{rig}_{o,t}\mathbf{T}^o_{B_i}p^{(d)}_{B_i},\boldsymbol{\theta_c})-u^{(d)}_{c,t})^2$$

where $\mathbf{T}^o_{rig,t}$ is the pose of the rig at time t, and $\mathbf{T}^c_{rig}$ is the relative pose between the rig and camera c, which does not change with respect to time.

This is a general formulation that can handle multiple cameras. In our case, we just need to estimate the pose between two different views. Therefore, we directly optimize the relative pose between the ground-truth view and the user-view, which is equivalent to taking the rig to be the ground-truth view and having only one camera, which is the user-view. See the Sec. 3 for this particular optimization objective we use.

Note that we also calibrate the relative pose between the 360° camera and the stereo camera using Bundle Rig PnP.

3. Bundle Rig PnP for two cameras

As promised, we give the full expression for the Bundle Rig PnP optimization objective for the case of two cameras. We first remind that the general expression minimizes the reprojection error over all cameras while optimizing over the relative pose between the cameras and the rig:

$$\min_{\mathbf{T}^o_{B_i},\mathbf{T}^o_{rig,t},\mathbf{T}^c_{rig}} \sum_{t=1}^{T}\sum_{c}\sum_{d}(\pi(\mathbf{T}^c_{rig}\mathbf{T}^{rig}_{o,t}\mathbf{T}^o_{B_i}p^{(d)}_{B_i},\boldsymbol{\theta_c})-u^{(d)}_{c,t})^2$$

Taking the rig coordinate system to be the ground-truth view (camera 1) coordinate system, we have the optimization objective for the special case

$$\min_{\mathbf{T}^{o}_{B_i},\mathbf{T}^{o}_{gt,t},\mathbf{T}^{u}_{gt}} \sum_{t=1}^{T} \Bigg[\sum_{d} (\pi(\mathbf{T}^{gt}_{o,t}\mathbf{T}^{o}_{B_i}p^{(d)}_{B_i}, \boldsymbol{\theta_{gt}}) - u^{(d)}_{gt,t})^2 + \sum_{d} (\pi(\mathbf{T}^{u}_{gt}\mathbf{T}^{gt}_{o,t}\mathbf{T}^{o}_{B_i}p^{(d)}_{B_i}, \boldsymbol{\theta_{u}}) - u^{(d)}_{u,t})^2 \Bigg], \tag{1}$$

which gives us the relative pose between the user view and the ground truth view denoted $\mathbf{T}^{u}_{gt}$. We use this optimization objective when we estimate the relative pose between the ground truth views and the user views. We run the the optimization for each rendered user view.

4. Ablation Study

4.1. Camera Pose Initialization

Here we explain the three camera pose initialization methods shown in Main Paper Tab. 7. For all of these, we assume that the pose graph is already constructed, and the question is how to use multiple boards seen in a single frame. See supplemental for more details.

- Method 0: For each board in the frame, we compute the relative pose between the camera and that board using PnP. We return the pose of the camera as the pose computed from the closest board.
- Method 1: We apply a confidence-weighted fusion approach. Since the camera sees several boards at once, we assign a confidence score to each detected board pose based on reprojection error and the number of markers detected. Lower reprojection error signals a more accurate pose. At frame t, we weight each board's pose by its re-projection error.
- Method 2: We use the idea of *multiboard-PnP* described in Main Paper 4.3.2. We estimate each frame's camera pose by combining all the visible checkerboards, transforming their inner 3D points into a common reference using the pose graph, and then performing a PnP estimation with multiple points.

4.2. Complete Ablation Study

In Tab. 1, we report the complete ablation study, which shows the effectiveness of each of the components of our Princeton365 ground truth method and externally validates it with respect to MoCap ground truth. We observe that the optimal configuration for our ground truth system is when pose graph optimization and snapping are enabled, the final pose is multi-board PnP, Bundle PnP is applied and the board pose graph is constructed from a video filmed at close distance. Thus, we choose this configuration to calculate the ground truth for the Princeton365 benchmark.

In this experiment, we estimate the camera trajectory using both our Princeton365 method and a Vicon Motion Capture system in order to measure the accuracy of the Princeton365 method with different configurations enabled. The MoCap system estimates the pose of an object coordinate system built from the positions of reflective markers in view of the MoCap cameras. Thus, we place reflective MoCap markers on our camera rig in a fixed position. We then calibrate the relative pose between the marker object and the camera coordinates using Perspective-n-Point where we film a Vicon Active Wand with both the MoCap cameras and the 360-camera rig. This allows us to estimate the relative pose between the wand and the 360-camera, as well as the pose between the wand and the MoCap coordinates. Thus, we obtain the pose of the 360-camera in MoCap coordinate system. We measure the accuracy of our ground truth based on the Average Trajectory Error (ATE) with respect to the MoCap trajectory.

We observe that the filming distance to calibration boards matters considerably in terms of the accuracy of our ground truth. In practice, this is the distance of the camera rig to the ground since we set up the boards on the ground. We observe that the board pose graphs built from videos that are filmed at a close distance (40-70cm) have lower ATE than those that are filmed at a medium distance (70 -150cm). For instance, in the 16 board setting where Snapping, Pose Graph Optimization and Multi-Board PnP are enabled, the close filmed pose graph - medium distance filmed frame has an ATE of 2.6 mm whereas the medium distance filmed pose graph - medium distance filmed frame has an ATE of 3.2 mm. Note that when collecting the Princeton365 benchmark, we film the calibration boards at a close distance to construct the pose graph, whereas we film the actual video that SLAM and COLMAP methods are evaluated on at varying distances to increase the diversity while preserving accuracy.

Pose graph optimization significantly improves the accuracy of our ground truth. This configuration makes the global board poses consistent with the relative measured poses between them. It almost halves the ATE when compared the same settings with pose graph optimization disabled. For example, in the 16 board C-C filming the ATE goes from 7.1 mm accuracy to 3 mm accuracy when pose graph optimization is enabled. Thus, we always enable pose graph optimization when obtaining the ground truth for the Princeton365 benchmark.

Snapping improves accuracy when the coplanarity assumption holds. This technique snaps the poses of the calibration boards to the same plane under the assumption that the boards are coplanar. For instance, with 16 boards and C-M filming and optimal configurations, the ATE goes from 7.0 mm to 6.5 mm when snapping is enabled. Note that we ensured that the boards are actually coplanar using a bubble level in this experiment. In general, we only enable snapping in Princeton365 benchmark when calibration boards

are coplanar.

Final Pose indicates the method we use to calculate the ground-truth pose of the frame when constructing the Princeton365 trajectory, before applying the Bundle PnP. At most frames, the ground truth view of the 360-camera will see multiple boards.

- Method 0 calculates the pose of the frame only with respect to the closest board using PnP.
- Method 1 performs PnP with respect to each board and combines the estimated poses using a weighted average where the weights are determined by the reprojection errors. Specifically, we apply a confidence-weighted fusion approach. Since the camera sees several boards at once, we assign a confidence score to each detected board pose based on reprojection error and the number of markers detected. Lower reprojection error signals a more accurate pose. At frame t, we weight each board's pose by its re-projection error:

$$W_{reproj}^{(i)} = \frac{1}{reprojection\_error_{(i,t)}} \tag{2}$$

Additionally, we set the confidence proportional to the number of markers detected on the board. We compute confidence as:

$$W_{num}^{(i)} = \frac{Number\ of\ Markers\ Detected\ on\ Board_i}{Total\ Numbers\ of\ Markers} \tag{3}$$

Combining 2 and 3 we define the total confidence as:

$$W_{total}^{(i)} = W_{reproj}^{(i)}\, W_{num}^{(i)} \tag{4}$$

In order to determine the final camera pose, we calculate a weighted average of each detected board pose based on their confidence scores. Since the pose of a camera is represented by a translation vector t and a rotation matrix R, we can compute them separately. The final translation t_{final} is given by:

$$t_{final} = \sum_i W_{total}^{(i)} T_i \tag{5}$$

For the final rotation, we compute the weighted average of quaternions [1, 3].

- Final Pose 2 method identifies the markers on every board, transforms them to a single coordinate system and performs a single multi-board PnP to estimate the pose of the frame. We find this multi-board PnP method to be the most accurate configuration. For instance, the ATE with 8 boards, C-M filming decreases from 8.1mm $\rightarrow$ 6.8mm $\rightarrow$ 5.5mm as we change the configuration closest board $\rightarrow$ weighted average $\rightarrow$ multi-board PnP.

We can further see the effect of these modifications in 2

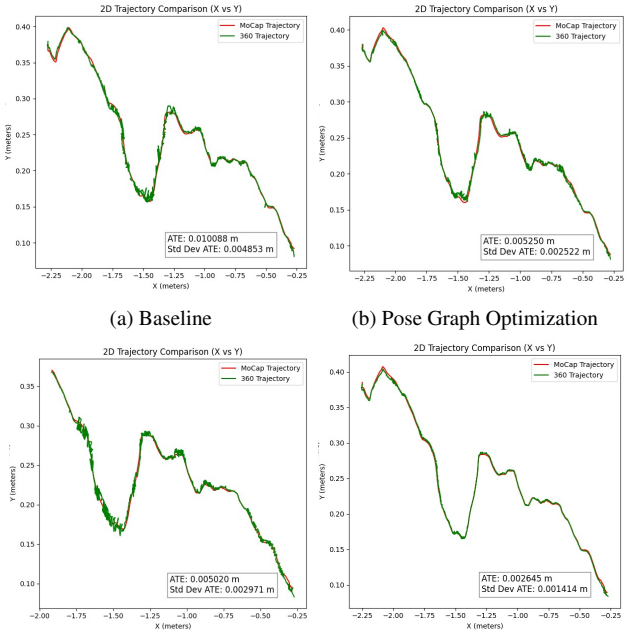
Finally, we get the final camera poses by using the Bundle PnP. This optimization technique further improves the results. In general, more boards lead to a better pose since there are more points detected per frame. As shown in Table 1, even with only 10 boards, applying Bundle PnP leads to a lower ATE (2.1 mm) than using 16 boards without this optimization (2.5mm). In Fig. 6 we can see more results of our ground truth trajectories with the optimal settings compared to MoCap.

# Boards	Filming Distance	Combination	Pose Graph Optimization	Snapping	Final Pose 0	Final Pose 1	Final Pose 2	Bundle PnP	Total ATE (mm)
7	C	C-C			✓				5.1
7	C	C-C		✓	✓				3.1
7	C	C-C		✓		✓			2.6
7	C	C-C		✓			✓		**2.1**
7	M	M-M			✓				22
7	M	M-M		✓	✓				9.5
7	M	M-M		✓		✓			7.2
7	M	M-M		✓			✓		9.6
7	M	C-M			✓				10
7	M	C-M		✓	✓				9.4
7	M	C-M		✓		✓			7.4
7	M	C-M		✓			✓		**4.2**
8	C	C-C			✓				7.1
8	C	C-C		✓	✓				5.2
8	C	C-C		✓		✓			4.4
8	C	C-C		✓			✓		**4.2**
8	M	M-M			✓				13
8	M	M-M		✓	✓				10
8	M	M-M		✓		✓			7.5
8	M	C-M			✓				10
8	M	C-M		✓	✓				8.1
8	M	C-M		✓		✓			6.8
8	M	C-M		✓			✓		**5.5**
16	C	C-C					✓		22
16	C	C-C	✓				✓		7.5
16	C	C-C		✓			✓		7.1
16	C	C-C	✓	✓			✓		3.0
16	M	M-M					✓		8
16	M	M-M	✓				✓		5.5
16	M	M-M		✓			✓		6.7
16	M	M-M	✓	✓			✓		3.2
16	M	C-M					✓		19
16	M	C-M	✓				✓		7.0
16	M	C-M		✓			✓		6.5
16	M	C-M	✓	✓			✓		**2.6**
10	M	C-M	✓	✓			✓		4.4
10	M	C-M	✓	✓			✓	✓	**2.1**

Table 1. Ablation Study. This figure summarizes the effects of various configurations on ATE compared to MoCap. The number of boards is the number of ChArUco boards in the ground truth pose graph. The Filming distance indicates how far the camera rig is from the calibration boards where C stands for close (40-70cm) and M stands for medium (70 -150cm). Combination X-Y means that the pose graph is constructed from a video filmed X distance away and the trajectory is constructed from a video filmed Y distance away. Pose graph optimization makes the global board poses consistent with the relative poses between the boards, snapping encodes an assumption that the boards are all coplanar, and final pose indicates the method we use to get the ground truth pose of a frame.

5. Calibration Boards

Fig. 3 shows the calibration boards used in the Princeton365 benchmark. We evaluated two types of fiducial-based calibration patterns: ChArUco boards and GridBoards. ChArUco boards are a combination of ArUco markers and chessboard, enabling subpixel corner refinement while retaining marker-level identification [7]. GridBoards, on the other hand, are a structured grid of ArUco



(a) Baseline (b) Pose Graph Optimization

(c) Snapping (d) Pose Graph Optimization + Snapping

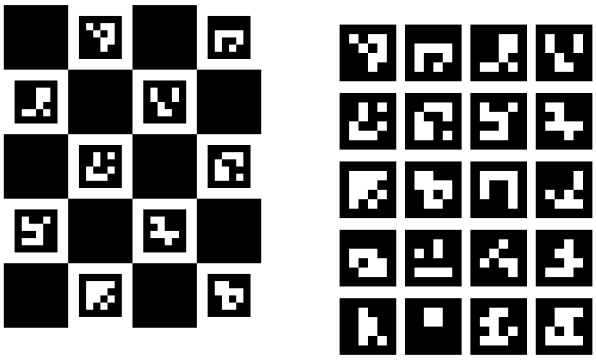
Figure 2. Comparison of our ground truth trajectory with MoCap. The figure illustrates the progressive reduction in ATE and standard deviation, while combining Pose Graph Optimization and/or Snapping. The combined approach achieves the best alignment to the ground truth trajectory.

markers without using the chessboard layout. Thus, using the same paper size, GridBoards can contain larger quantity and size of Aruco markers compared to a Charuco board, offering robust detection across a broader range of viewpoints and lighting conditions, though without subpixel refinement [6].

In order to evaluate the accuracy of both board types, we placed several ChArUco boards and GridBoards side by side and recorded a single trajectory. At the same time we used MoCap system for ground truth. From the same recording, we extracted two separate trajectories—one using only ChArUco board detections and one using only GridBoard detections. Comparing both to the MoCap ground truth, we found that the trajectory estimated with GridBoards resulted in lower ATE as shown in Table 2.

Calibration Board	ATE (mm)
ChArUco Board	1.6
GridBoard	1.1

Table 2. Comparison of ATE for ChArUco and GridBoard using MoCap. GridBoard achieves lower ATE for the same trajectory, indicating higher trajectory accuracy.



(a) **ChArUco Board** (b) **GridBoard**

Figure 3. Two types of fiducial calibration boards used in our benchmark: ChArUco and GridBoard. Each marker has a unique ID for pose estimation and pose graph construction.

6. Depth Distribution

The ZED X stereo camera returns pixel-wise depth values per frame. Since we want to compute the induced optical flow, one option would be to compute the induced flow directly using these depth samples. We instead fit a parametric distribution to these depth samples. This approach has a few advantages. Firstly, it is more robust to outliers, noise, and low amount of samples. More importantly, a parametric distribution allows us to have an analytical expression for the expectation of the induced optical flow. If we wanted to use depth samples directly, we would have to use a Monte Carlo estimate of the expectation.

We fit the parametric depth distribution as follows. We take the depth samples for each frame of a sequence and concatenate them. Then, for every number of components from 1 to 8, we fit mixture of Gaussian and mixture of gamma distributions with that number of components. The reason for including the gamma distribution is because depth distributions are positive and the components tend to be skewed, which are properties satisfied by the Gamma distribution. Since higher the number of components, higher the likelihood, we need to balance the complexity of the distribution with the goodness of the fit. Thus, we take the best fitting distribution as measured by the Bayesian Information Criterion (BIC) [8].

An example of the fitting process for the parametric depth distribution is shown in Fig. 5. We show a histogram of a depth distribution, a comparison of the best mixture of Gaussian and mixture of Gamma fits, as well as the BIC selection process.

7. Pose Graph Optimization

We want construct a pose graph where nodes represent global board poses $\{\mathbf{T}_o^{B_i}\}$ and edges represent measured

relative poses $\mathbf{T}_{B_i}^{B_j}$ between boards i, j. The goal is to estimate a set of global poses $\{\mathbf{T}_o^{B_i}\}$ such that the relative pose computed from them matches the observed local transformations:

$$\mathbf{T}_o^{B_i\,-1}\mathbf{T}_o^{B_j} \approx \mathbf{T}_{B_i}^{B_j} \tag{6}$$

The optimization is the following:

$$\underset{\{\mathbf{T}_o^{B_i}\}}{\arg\min} \sum_{(i,j)\in\mathcal{E}} \left\| \mathrm{Log}\left(\left(\mathbf{T}_{B_i}^{B_j}\right)^{-1} \mathbf{T}_o^{B_i\,-1}\, \mathbf{T}_o^{B_j} \right) \right\|_{\mathbf{\Omega}_{ij}}^2 \tag{7}$$

where $\mathrm{Log} : \mathrm{SE}(3) \rightarrow \mathfrak{se}(3)$ maps a pose to its 6-D Lie-algebra vector, and $\mathbf{\Omega}ij$ is the information matrix for the measurement (i, j). We compute this using g2o[4].

8. Validation of Bundle Rig PnP

In this section, we show that the estimated relative pose between the user view and the ground truth view is accurate. The idea is to move the camera in two different trajectories, run Bundle Rig PnP to estimate a relative pose for each trajectory, and compare the two relative poses to make sure that they are consistent. If Bundle Rig PnP works well, the two estimated poses should be the same since the views are fixed.

We set up 8 different board configurations where the boards are placed so that both views can see boards at the same time. For each configuration, we film two sequences with different trajectories. We render the ground-truth view and the user-view with fixed relative pose for both sequences. Note that for each configuration we render different relative poses used in the benchmark to ensure that all of them are accurate. We then run Bundle Rig PnP to estimate two relative poses, one for each sequence. Finally, we compare the poses to each other in terms of translation and rotation difference.

The results in Tab. 3 show that the rotation difference between the two estimated poses is 0.070° on average, and the translation difference is 0.89 mm on average. Thus, we conclude that the relative pose between the user-view and the ground-truth view is accurate.

Experiment	Rotation Diff. (deg)	Translation Diff. (mm)
1	0.0712	1.04
2	0.0447	0.51
3	0.0543	1.08
4	0.0792	1.04
5	0.0576	0.39
6	0.0894	1.16
7	0.0814	1.32
8	0.0974	0.59
Mean	**0.0705**	**0.891**

Table 3. Relative pose differences for 8 cases using Bundle Rig PnP. It shows that the estimated relative pose between the user view and the ground truth view remains consistent across different trajectories. Thus, we conclude that the relative pose calibration is accurate.

9. Bundle Rig PnP outperforms regular PnP

In this section, we show that Bundle Rig PnP calibration outperforms naive PnP when estimating the relative pose between the user-view and the ground-truth view.

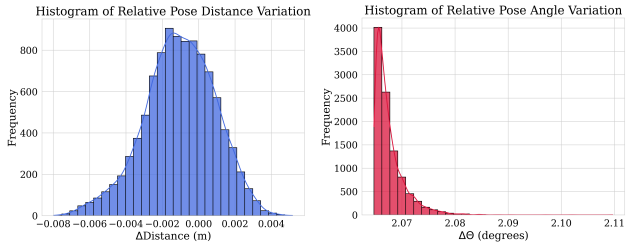
To evaluate naive PnP, we placed reflective markers that can be seen by both MoCap and the user-view of Insta360. We annotate the 2D pixel position of each marker for multiple frames. Running PnP gives us the relative pose between user-view and the markers we have placed. Since the position of the markers can be tracked by MoCap, we obtain the pose of the user-view in MoCap coordinates by chaining poses together. We use this to obtain the fixed relative pose between the Insta360 body (constructed from reflective markers placed on Insta360) and the user-view. We perform a similar process to obtain the relative pose between the ground-truth view and the Insta360 body. Hence, we obtain the relative pose between the user-view and the ground-truth view through just using PnP.

How does this naive approach compare to Bundle Rig PnP? Since we cannot obtain the relative pose between the two views through an external measurement, we have to rely on internal validation for comparison.

We measure the variance of the estimated relative pose between the ground-truth view and the user view of the 360 camera by using a bootstrapping approach. We take all the 2D-3D correspondences used in the PnP calculation of the relative pose, and we resample with replacement to recalculate the relative pose. We construct a histogram of the translation and the rotational distance from the original relative pose performing 10,000 Monte Carlo trials. Fig. 4 shows the that the variance of the poses obtained by naive PnP is significantly higher than the variance of the poses obtained by Bundle Rig PnP. In particular, note that a large portion of the rotations of Naive PnP in Fig. 4 fall higher than 2 degrees in stark contrast to the average rotation difference of 0.07 deg obtained by Bundle Rig PnP shown in Tab. 3.

10. Further GT validation with MoCap

As mentioned in Main Paper Sec. 5, we validate our ground-truth pipeline with a Vicon Vantage V16 MoCap system. However, MoCap does not work outdoors. As a result, all validation sequences in Main Paper Tab. 5 have been recorded indoors. The indoor and outdoor sequences are not significantly different from a ground-truth perspective since the ground-truth view films calibration boards on the ground in both cases where the main difference is the



(a) Distance Variation (b) Angle Variation

Figure 4. Histograms of both distance and angle variation obtained via bootstrapping Naive PnP. We observe that Naive PnP is not as reliable as Bundle Rig PnP when it comes to estimating the relative pose between the user-view and the ground-truth view. The histograms show high variance for the naive PnP estimates. Difference in distance is given in meters and the difference in angle is given in degrees. A total of 10,000 trials were conducted.

indoor vs outdoor lighting. However, we still perform further experiments to make sure that the indoor ground-truth validation generalizes to outdoor sequences as well.

We measure reprojection errors coming from Bundle PnP for both indoor and outdoor sequences. The ground-truth accuracy, as measured by reprojection error, appears consistent across indoor and outdoor environments, where the average indoor reprojection error is 0.7373 px and the average outdoor reprojection error is 0.7394 px. We use the Welch's t-test to test if there is a statistically significant difference between the two cases. The difference between indoor and outdoor averages is 0.0021 px where the 95% confidence interval for the difference is [-0.0175, 0.0217] and the p-value is p=0.832, meaning that the difference measured is not statistically significant. As a result, we conclude that the validation experiments showing that our ground-truth is mm accurate generalize to outdoor sequences as well.

11. Computing Induced Optical Flow

Recall that the IOF metric can be written as

$$\text{IOF} = \frac{1}{AT} \sum_t \sum_{u,v} \int_0^{\infty} \|\mathbf{flow}(t,d,u,v)\|_2 \; p(d) \, dd.$$

We sample (u, v) over a uniform grid of pixels in the image denoted A. After fitting the parametric depth distribution p, we calculate minimum and maximum depth values for this distribution. For the mixture of Gaussians, the d_{min} is calculated as the value 4 standard deviations smaller than the mean of components. We take the smallest value over all components. d_{max} is calculated as the value 4 standard deviations higher than the mean of components. Again, we take the largest value. Thus, we write the IOF metric as

$$\frac{1}{AT} \sum_t \sum_{u,v} \int_{d_{min}}^{d_{max}} \|\mathbf{flow}(t,d,u,v)\|_2 \; p(d) \, dd.$$

Assuming we have calculated $\|\mathbf{flow}(t,d,u,v)\|_2$, the inner integral can be numerically integrated.

Now, we talk about how to calculate $\mathbf{flow}(t,d,u,v)$. We denote the trajectory collected by our ground truth method as $\mathbf{T}_{\text{gt}}^{(t)} \in \mathbf{SE}(3)$ and the trajectory estimated by the SLAM method as $\mathbf{T}_{\text{est}}^{(t)} \in \mathbf{SE}(3)$. Here t denotes the frame or timestamp of the trajectory.

We take $\mathbf{K}$ to be the 4×4 Camera Intrinsic Matrix in Homogeneous Coordinates:

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where f_u, f_v are the focal lengths in the u and v directions (pixels), and c_u, c_v are the principal point coordinates.

We denote a pixel sampled in the ground truth trajectory at frame t as $\mathbf{p}_{\text{gt},(u,v)}^{(t)}$. This can be written in homogeneous coordinates with inverse depth $\rho_{(u,v)}^{(t)}$ as

$$\mathbf{P}_{\text{gt},(u,v)}^{(t)} = \begin{bmatrix} u \\ v \\ 1 \\ \rho_{(u,v)}^{(t)} \end{bmatrix},$$

where the depth comes from the numerical integration over the depth distribution.

The re-projection of $\mathbf{P}_{\text{gt},(u,v)}^{(t)}$ to frame t of the estimated trajectory can be calculated as:

$$\mathbf{P}_{\text{est},(u,v)}^{(t)} = \mathbf{K} \, \mathbf{T}_{\text{est}}^{(t)} \, \mathbf{T}_{\text{gt}}^{(t)-1} \, \mathbf{K}^{-1} \, \mathbf{P}_{\text{gt},(u,v)}^{(t)},$$

where we unproject the pixel, transform it using the relative pose between the estimated and ground truth frames, and reproject it to the estimated frame. The induced flow is the difference between inhomogeneous coordinates:

$$\mathbf{flow}(t,d,u,v) = \mathbf{p}_{\text{est},(u,v)}^{(t)} - \mathbf{p}_{\text{gt},(u,v)}^{(t)}.$$

12. Trajectory Alignment Details

In this section, we denote frames with i to avoid confusion. Decomposing the 6 DoF pose $\mathbf{T}^{(i)}$ into the rotation $\mathbf{R}^{(i)} \in \mathbf{SO}(3)$ and translation $\mathbf{t}^{(i)} \in \mathbb{R}^3$, we can write the initial $\mathbf{Sim}(3)$ alignment of the translation part as

$$\min_{s \in \mathbf{R}^+, \; \mathbf{R} \in \text{SO}(3), \; \mathbf{t} \in \mathbb{R}^3} \sum_{i=1}^{T} \left\| s \, \mathbf{R} \, \mathbf{t}_{\text{est}}^{(i)} + \mathbf{t} - \mathbf{t}_{\text{gt}}^{(i)} \right\|^2.$$

We can write the optimization objective for the secondary alignment on the orientation of the estimated trajectory as

$$\min_{\mathbf{R}_{\text{align}} \in \text{SO}(3)} \sum_{i=1}^{T} \left\| \log \left(\mathbf{R}_{\text{align}} \, \mathbf{R}_{\text{est}}^{(i)} \, (\mathbf{R}_{\text{gt}}^{(i)})^{\top} \right)^{\vee} \right\|^2 .$$

We perform both optimizations using SVD.

13. Coverage and Composite Score

A lot of SLAM methods do not output poses for the whole sequence. For example, ORB-SLAM [5] will take a while to initialize. This means that first couple of hundred frames will not have a pose. Also, SLAM methods might fail on certain sequences, meaning that they will not output any pose for that sequence.

This creates a precision and recall issue. If we report Flow AUC computed only from frames that have a pose, then models that output only a few poses with high confidence are rewarded, punishing robust methods.

What if we forced the methods to output the same exact number of frames as our ground-truth, a strategy followed by benchmarks like KITTI [2]? In that case, a method would have to pad its gaps with pose guesses. For instance, some methods fill in the gaps with the closest available estimated pose. This strategy will cause disproportionately low Flow AUC when poses for certain frames are missing. Also, the strategy does not handle the case when a method completely fails on certain sequences.

Our solution is to create a composite score from Induced Optical Flow Metric, and tracking coverage. Tracking coverage is roughly for what percent of frames there is an estimated pose. It can be thought of as a measure of recall. In particular, we compute

$$\text{coverage} = \frac{\#\text{frames predicted}}{\#\text{frames total}}.$$

Note that coverage includes failed sequences as well. Furthermore, coverage has the same range as Flow AUC where both metrics range from 0% to 100%. This allows us to easily combine the two scores.

The composite score, then, is the harmonic mean (f-score) of Average Flow AUC and coverage, which is similar to f-score computed from precision and recall:

$$\text{composite} = \frac{2}{\frac{1}{\text{AUC}_{\text{avg}}} + \frac{1}{\text{coverage}}}$$

Thus, we sort our leaderboard using the composite score by default. This rewards methods that are both accurate and robust.

14. Experimental Setup

In our evaluation Table Main Paper Tab. 6, we ran DPVO and DPV SLAM with their default settings, processing all sequences at stride 1. As for LEAPVO, we also used the default configuration for the scanning sequences at stride 1. However, we used stride 10 for the indoor and outdoor scenes since increasing the buffer size for longer sequences still resulted in failures. Lastly, we ran COLMAP at stride 1 with default parameters and sequential match, without intrinsic refinement. We assumed that if a sequence needs for processing more than 2days and 16h will be considered as failure.

15. Rig Construction

Regarding the rig construction, we mounted the Insta 360 camera and the ZED camera using a smallrig Clamp along with a 9.8-inch Adjustable Friction Power Articulating Magic Arm. This setup allowed us to adjust the alignment of both cameras, so as the Insta360's user view closely matches with that of the ZED one. Once we finalized an acceptable configuration we also applied epoxy glue to the joints, in order to ensure that the relative pose between the ZED and 360 camera remained stable. Epoxy is a high-strength structural adhesive, which provides extra stability beyond the clamp itself. The complete rig setup can be seen in Main Paper Fig. 3.

16. NVIDIA Jetson Optimizations

16.1. Hardware Tuning

Before achieving maximal data recording frequencies for RGB or IMU data by themselves, we needed to perform additional hardware-related optimizations for the NVIDIA Jetson. These optimizations are crucial, as they introduce about a 3x frequency increase just by themselves.

The first optimization involves `nvpmodel`, which forces the NVIDIA Jetson to draw maximal power from its power source. This ensures the camera is receiving the most amount of power it is designed to handle, and thus can record at maximum frequency.

The second optmization involves `jetson_clocks`, which sets static max frequency to CPU, GPU, and EMC clocks on the NVIDIA Jetson. This ensures that the Jetson's compute resources are being utilized to their maximum potential when recording data.

By combining these different hardware optimizations together, we are able to achieve 60 FPS video recording by itself, or 400 Hz IMU data recording by itself.

16.2. Software Parallelization

Although RGB video and IMU data can be collected at their respective maximum frequencies when recording by them-

selves, new difficulties arise when trying to record both RGB and IMU together. We looked towards techniques in parallel computing to try and preserve maximal data recording frequency. The primary methods we looked into are multiprocessing and multithreading.

Both multiprocessing and multithreading strive to improve total processor performance and therefore position themselves to decrease the processing time for any application that exposes concurrent software threads for execution. The two technologies however take different approaches in the hardware to address these goals and will subsequently offer different levels of success for any particular example of software code.

- **Multithreading** refers to the ability of a processor to execute multiple threads concurrently, where each thread runs a process.
- **Multiprocessing** refers to the ability of a system to run multiple processors in parallel, where each processor can run one or more threads.

Multithreading is beneficial for IO-bound tasks, such as reading files from a network or database, because it allows each thread to concurrently execute these processes. In contrast, multiprocessing is more suitable for CPU-bound tasks that require substantial computational resources, as it can utilize multiple processors, mimicking the efficiency of multicore systems.

There is a clear distinction between concurrency and parallelism: concurrency involves executing multiple tasks in an interleaved manner, one at a time, whereas parallelism involves executing multiple tasks simultaneously.

Python's global interpreter lock (GIL) restricts multithreading to executing only one thread at a time, meaning that it only offers concurrency, not parallelism, particularly for IO-bound processes. However, multiprocessing enables parallel execution.

Using multithreading for CPU-bound tasks can degrade performance, due to the limited execution capabilities under the GIL and the overhead associated with managing multiple threads.

Although multiprocessing can be applied to IO-bound processes, it generally incurs greater overhead than multithreading. Yet, it can lead to increased CPU usage, which is expected given that multiple CPU cores are engaged by the application.

Throughout our experimentation with the ZED X camera to optimize data synchronization and processing, we found out that although multiprocessing allowed parallel data handling, it introduced blocking issues that reduced IMU performance. Multithreading improved responsiveness and IMU capture rate by reducing overhead, offering a more continuous data flow.

17. Estimation of Distance Covered

In Main Paper Tab. 3, we provide estimated total distance covered and distance covered with ground-truth pose. We calculate posed distance by summing up the length of the ground-truth trajectories. To estimate the total distance covered, we need an estimation of the distance covered for frames that do not have a pose. Since the IMU uncertainty compounds quickly it is an unreliable estimate. Thus, we assume an average walking speed of 1.4 m/s and multiply that with the total duration of frames without pose. This assumption roughly holds in practice since for almost all sequence portions without pose, we travel with the camera at walking speed.

18. Dynamic Objects

Examples of dynamic objects in our sequences include people walking and cycling, cars passing by, snow, water fountains, rotating chairs, deformable objects, doors opening, etc. We blur the faces of people as well as the license plates of cars.

References

[1] Jonathan Blow. Understanding slerp, then not using it. *Game Developer Magazine*, 2004. 3

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 7

[3] Claus Gramkow. On averaging rotations. *Journal of Mathematical Imaging and Vision*, 15(1):7–16, 2001. 3

[4] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. 5

[5] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 7

[6] OpenCV Team. OpenCV: Detection of aruco boards. https://docs.opencv.org/4.x/db/da9/tutorial_aruco_board_detection.html, 2024. Accessed: 2024-11-15. 4

[7] OpenCV Team. OpenCV: Detection of charuco boards. https://docs.opencv.org/3.4/df/d4a/tutorial_charuco_detection.html, 2024. Accessed: 2024-11-15. 3

[8] Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461 – 464, 1978. 4

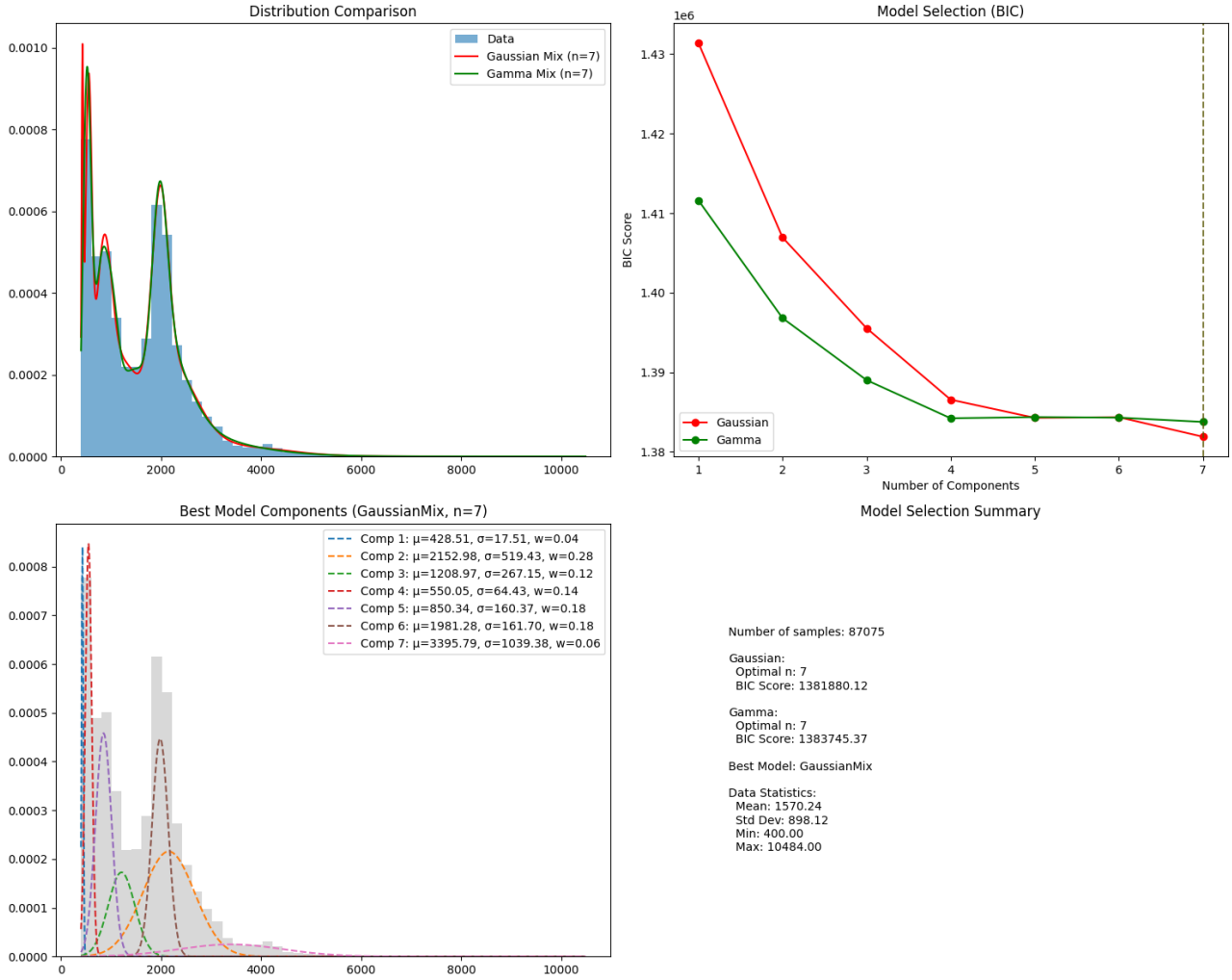
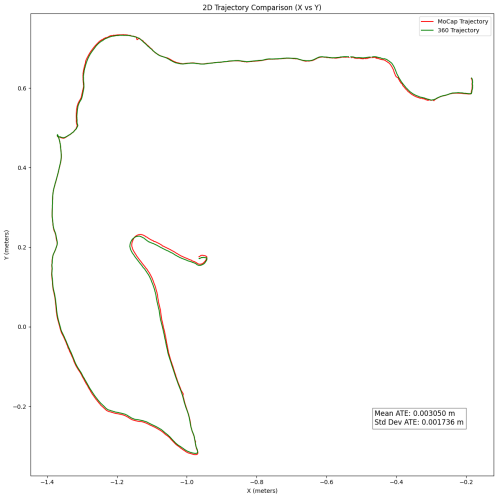
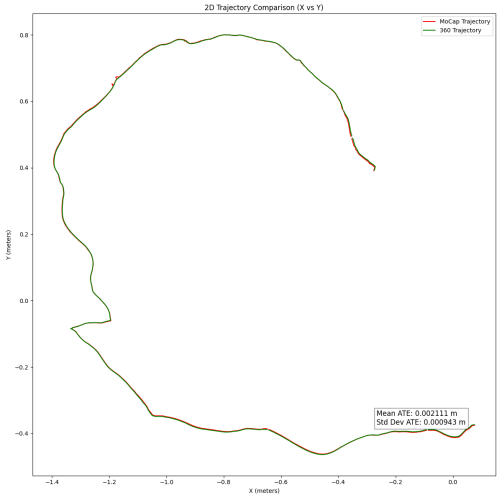


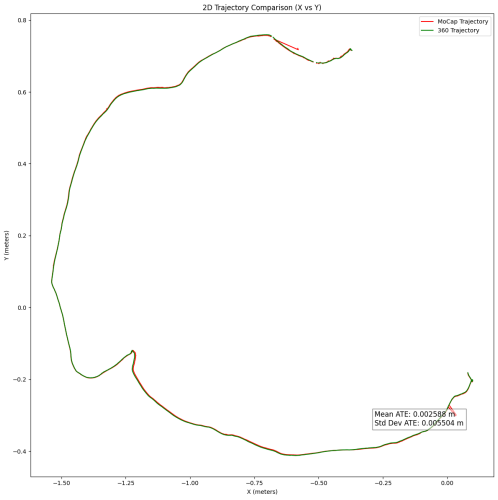
Figure 5. An illustration of the parametric model we fit to the depth data. The figure shows fitted distributions, BIC scores for model selection, individual components, and summary statistics of the depth data.



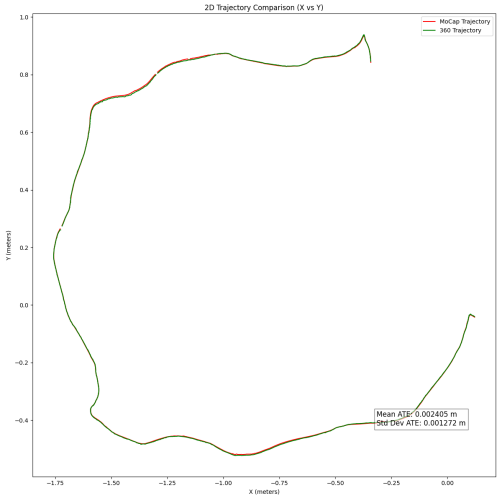
(a)



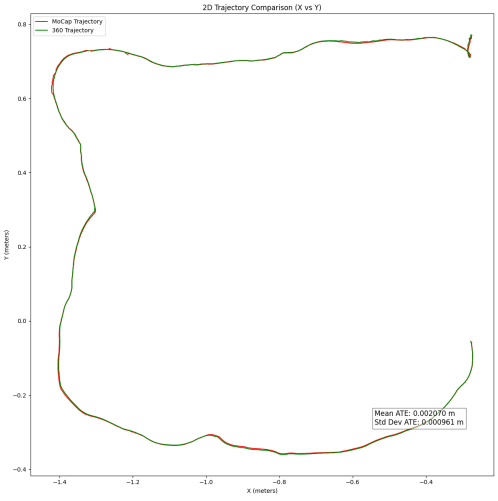
(b)



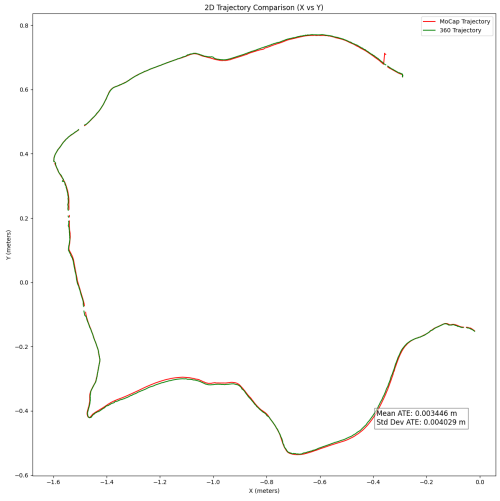
(c)



(d)



(e)



(f)

Figure 6. X and Y comparison of six different trajectories as measured by MoCap and the 360 camera.