

Dynamic Camera Poses and Where to Find Them: Supplemental Material

Chris Rockwell^{1,2} Joseph Tung³ Tsung-Yi Lin¹
Ming-Yu Liu¹ David F. Fouhey³ Chen-Hsuan Lin¹
NVIDIA¹ University of Michigan² New York University³

The Supplemental material contains experiments and detail that could not be included in the main paper due to length constraints. DynPose-100K results are best viewed in the Supplemental video. Supplemental videos use 12 fps so each frame corresponds to an annotated camera pose, since poses are annotated at 12 fps.

1. DynPose-100K: Dataset Curation Additional Details

We expand upon the method of §3 from the paper.

1.1. Candidate Selection Criteria

Candidate selection criteria fit into three broad categories, which can be broken down into sub-categories. These are visualized in Figure 2 in the main paper and are broken down further in Figure 1. We detail below:

- C1. **Real-world and quality video.** Videos removed for this reason are titled in Figure 1 not real and quality / ethical. Not real videos are described in the main paper and include cartoons and animated videos, video games, computer screen recordings, post-processing resulting in large logos or text appearing on screen or other video appearing side-by-side. Quality reasons include poor lighting and blocked or blurred lens. We select Panda-70M videos [3] with 720p resolution, but in rare cases a lower-resolution video has been up-sampled to 720p. Ethical reasons include children, NSFW and violence.
- C2. **Feasibility for pose prediction.** Videos removed for this reason are labeled not estimable. Subcategories include long focal length, zoom in or out, non-existent or out-of-focus static region, shot change, and ambiguous frame of reference.
- C3. **Dynamic camera and scene.** These correspond to static camera and static scene in Figure 1.

1.2. Candidate Video Selection

We include videos based on scores coming from the following filters. Each filter produces a score between 0 and 1 as to whether the video should be included. The average of these is used as the final score, from which we threshold and take

all videos with scores higher. Thresholds for each filter are tuned on a 1K validation set Panda-Val, collected in a similar manner to the 1K Panda-Test. Validation videos have no overlap with test videos. Also, long videos containing val videos have no overlap with the long videos containing test videos. This is a necessary precaution since Panda-70M videos are clips from long videos in HD-VILA-100M [23].

- 1. *Cartoons and presenting.* We use the classifiers of Hands23 [4], which predict whether a video is likely to have interaction; and whether a video contains children, cartoons or screen recordings, or person sitting in front of a camera. This uses 4 evenly spaced frames in thumbnail size as input to two classifiers, one for if a video is acceptable, *e.g.* no cartoons, etc.; and one for if interaction is likely to occur, *e.g.* not a static scene. Each predicts a confidence score between 0 and 1. We use a minimum threshold for 0.55 for acceptable and 0.20 for interaction, meaning scores are 1 if greater than a threshold and 0 otherwise. Final score is the average of both.
- 2. *Non-perspective distortion.* We use DroidCalib [8] to predict radial distortion at an interval of 6 fps for efficiency. DroidCalib predicts α from the unified camera model [15]. A maximum threshold of 1.00 is used; higher outputs receive score 0 as they are likely distorted.
- 3. *Focal length.* We use WildCamera [26] to compute the frame-wise focal length of videos, applying on frames at 6 fps for efficiency. Focal lengths are predicted on 720p frames. We apply two variance thresholds. First, we check the difference between 90th and 10th percentile focal lengths. If this is greater than 40% of the mean focal length, it receives sub-score 0; 1 otherwise. This is a good check to see if focal length changes over the course of the video, sometimes due to shot change. Second, we check variance over a sliding window of 1 second. If focal length changes by more than 20% over 1 second, it receives sub-score 0 as there is likely zoom or shot change; 1 otherwise. Finally, we apply a mean threshold, giving sub-score 0 if the 80th percentile focal length is greater than 1400; 1 otherwise. This is because extremely long focal lengths are often too focused to clearly see a background, and further have such a small

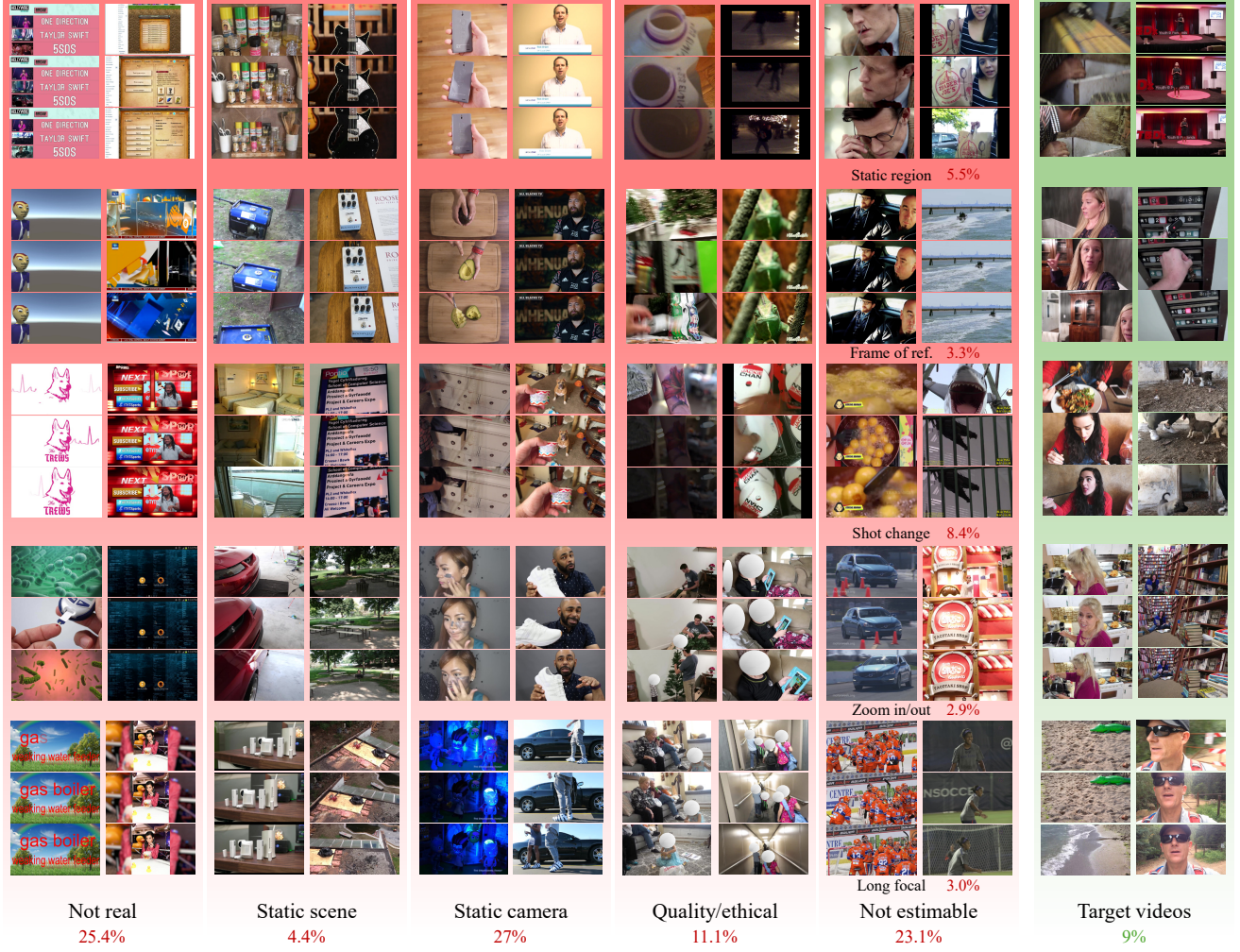


Figure 1. **Panda-Test breakdown.** We add more examples for each category, including further breakdown statistics for non estimable. Stats reflect human labels on the 1K Panda-Test set, detailed in § 2.1.

- field of view localization is hard, even for a human. The final scores is the average of the three sub-scores.
4. *Dynamic object masking.* We apply our masking method (§3 from the paper) to compute the dynamic mask size. For efficiency, instead of applying masking every 0.5 seconds and propagating forward 0.5 seconds, we apply masking every 1 second and propagate forward 1 second. This is useful since propagation is much faster than the other components of masking. Dynamic mask size is computed for each frame, the 90th percentile mask size is then compared to 80% of the frame size. In other words, if the 10% of frames with biggest masks have a mask greater than 80% of frame size, the video receives masking score of 0; 1 otherwise.
 5. *Optical flow.* We compute the average magnitude of sequential optical flow using RAFT [20], this is applied

at 6 fps. Flow bigger than 2.127% of frame size average distance is given sub-score 1. This threshold is chosen to correspond to the 80th percentile of optical flow in the dataset. Low optical flow is removed as it tends to correspond to static scenes and cameras. Next, two sub-components handle shot change. First, maximum sequential-frame flow must be less than the mean plus 4 standard deviations for 1 on the sub-score; else 0. Second, sustained flow must not be too high; maximum average flow over 1 second sliding windows must be less than a threshold of 15% of frame size to receive 1 on the sub-score; else 0. The average of all three components is the final score.

6. *Point tracking.* We apply an abbreviated version of our tracking method over the full video. Instead of repeating tracking, we apply tracking only once for efficiency.

Tracking is done at 3 fps and tracked for 30 frames, meaning entire videos of 10 seconds or less are tracked fully. From Figure 5 in the main paper, over 99% of sequences are less than 10 seconds. In the case of longer videos, the first 10 seconds still provides a good estimate of the tracking metrics. Tracking has three components, one measuring large disappearance of tracks indicating shot change, and two measuring lack of movement indicating static camera or scene. Each respective formula is more complex, but at a high-level, a video scores 0 on shot change if more than 50% of tracks are lost across a single frame, while dynamics scores 0 if median tracks move less than about 5% of frame size.

Component smoothing. A sigmoid is applied to component scores for smoothing. While this doesn’t impact clear positives or negatives in each case, it gives a more informative ranking of borderline scores. For example, an example with average optical flow of 2.126% is marginally less likely to be included than one with average flow of 2.128%. Smoothed scoring is more reflective of this rather than the former scoring 0 and the latter scoring 1.

Generalist VLM. The VLM answers eight questions overviewed in the main paper. Each question indicates a reason a video would not be included, so if any answer is yes, a score of 0 is given. Otherwise, the score is 1.

Filtering efficiency. We find we can often reduce computation of filtering methods compared to pose estimation, as not as much precision is required to get an aggregate filtering score. For example, precisely tracking each frame is important for accurate pose estimation, but to determine whether the camera is static, we need only check if tracks sufficiently moved at a few fps. This efficiency is important considering filtering is run on many more videos (about 3.2M) than pose estimation (about 100K).

Collection details. During filtering, we remove videos with average score below a threshold. Final filter scores are between 0 and 1; the final threshold we use is 0.910, resulting in 137K videos, corresponding to about 4.3% of the full 3.2M videos.

Filtering is applied in several stages to reduce compute cost, as reported in the main paper. After completing a stage, only a subset of all filters are available to attempt to remove unsuitable videos. Experiments indicate this is not as effective as using all filters (Table 1). Nevertheless, this subset is still a good proxy for whether to filter the video. We therefore use an average of these filters, but apply a less strict threshold than the final filter to avoid removing suitable videos. Subset thresholds are chosen empirically based on the evaluation set to avoid removing good candidates.

Filters are applied in the following order for efficiency: Hands23, flow, focal; then distort; then tracking; then masking; and finally VLM. Hands23, flow and focal are

lightweight operations, distort is nearly as lightweight, then tracking, then masking. VLM (GPT-4o [16]) is applied last to minimize OpenAI API costs. Hands23, flow and focal were run jointly early in the project to experiment with filtering on a large set.

After running SfM, we drop trajectories with less than 80% of frames registered. This is a typical threshold for a succeeded trajectory used in evaluation [2, 14]. Early experiments found this to be a good proxy for quality.

1.3. Dynamic Camera Pose Estimation

Dynamic camera pose estimation builds upon ParticleSfM [25]’s calls to TheiaSfM [19] for global bundle adjustment given masks and tracks as input. More detail for dynamic masking and point tracking follow.

Dynamic masking. Dynamic masking combines four complimentary approaches to motion detection: semantics, for common dynamic classes; object interaction, for sometimes-dynamic objects that move when interacted with; motion, for sometimes-dynamic objects that move even when not in contact with humans; and tracking, to smoothly and efficiently propagate detected masks. We apply the former three masks once every six frames and input them into tracking. Tracking then outputs the tracked masks for the current frame and next five frames, before the process is repeated at the sixth new frame. Using frames extracted at 12fps for DynPose-100K, this results in masks produced and propagated forward for 0.5 seconds. An example of dynamic masking in practice can be found in more detail in Figure 2.

Semantic segmentation. We apply OneFormer [10] to mask common dynamic classes such as humans and sports equipment. We use the same classes for dynamic masking as RoDynRF [14]: MS-COCO [12] classes 1 (person), 2-9 (vehicle), 16-25 (animal), 26-33 (accessory), 34-43 (sports), and 88 (teddy bear).

Object interaction segmentation. We use Hands23 [4] hand-object interaction with default parameters. We consider only masks associated with hand-held objects. This corresponds to touch classes 1, 2, 4 and 6. We found masking objects being touched but not held (3, 5) resulted in masking too many objects that were not moving. Movement was highly correlated with the held class.

Motion segmentation. Sampson error [9] is computed both forward and backward on each sequential frame pair based on flow [20]. The maximum of forward and backward is computed for each pixel, after which we normalize and threshold. Like [14], we mask pixels with errors greater than $(frameheight * framewidth) / 8100$.

Mask propagation. We use SAM2 [18] video predictor with sam2-hiera-large checkpoint and config to propagate

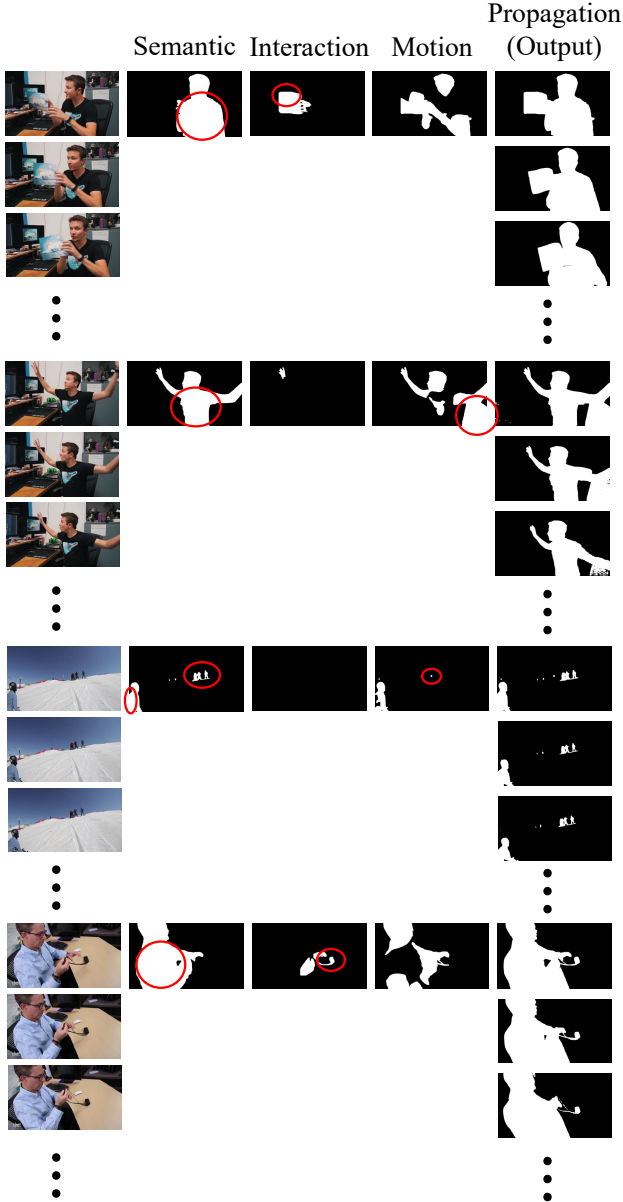


Figure 2. **Masking composition.** Each component of masking contributes to final masks. Unique contributions for semantics, interaction and motion are circled in red. Semantic segmentation handles common dynamic objects such as humans (all examples). Object interaction handles things humans are manipulating such as paper (top) or accessories (bottom). Motion handles things moving not by human hands such as swiveling chairs (second from top) or flying snowballs (third from top). Sometimes objects are partially segmented by one component but complementary components can still give a more complete mask. *E.g.* in the third example, flying snowballs are segmented by semantics but motion helps complete the dynamic mask. All components are combined and tracked smoothly by propagation (right).

the combination of the three former masks. For efficiency, we perform segmentation every six frames and use SAM2 to propagate the mask forward for six frames. Since propagation is several times faster than masking, the result is a substantial overall speedup.

Point tracking. To collect DynPose-100K, we predict tracks using input resolution (256, 256), the default for BootsTAP [6]. We track a grid of $(42, 42) = 1764$ points. We apply tracks every 5/12 second and track for 2.5 seconds. Using frames extracted at 12fps, this corresponds to a stride of 5 frames and tracking length of 30 frames. Early experiments indicated higher resolution did not meaningfully improve results while reducing efficiency. We therefore selected these parameters, considering our goal of large-scale collection.

Repeating tracking later in videos results in sequences where video is shorter than the number of frames to track. We find BootsTAP [6] runs far faster when repeating the same sequence length, as opposed to shortening sequence length. We therefore pad sequences with empty frames as needed to keep same sequence length, finding in early experiments this has negligible impact on results while meaningfully speeding up tracking.

2. DynPose-100K: Dataset Analysis Additional Details

This section adds detail for §4.2 in the paper.

2.1. Filtering Evaluation on Panda-Test

Dataset. Panda-Test consists of 1K randomly selected videos manually categorized into suitable or one of several non-suitable categories. These are visualized in Figure 2 in the main paper and detailed in Figure 1. Manual filtering goes as follows: a video is first checked to see if it is not real. If it passes this check, it is checked for quality / ethics, then if pose is estimable, then if pose is dynamic, then scene dynamic. If it passes these checks it is considered a target video. 90 (9%) are considered suitable.

Results. Table 1 shows filtering evaluation in more detail than Figure 4 from the main paper. Following the Pascal Visual Object Classes challenge [7], Table 1, left shows a smoothed precision-recall curve for increased visual clarity. In particular, for each data point, we use the maximum precision of all data points whose recall is greater than or equal to the current recall. This smooths the curve such that precision is non-increasing as recall increases. Table 1, right reports precision values of the non-smoothed curves. Precision at Recall of 0.40 corresponds to the operating threshold for DynPose-100K.

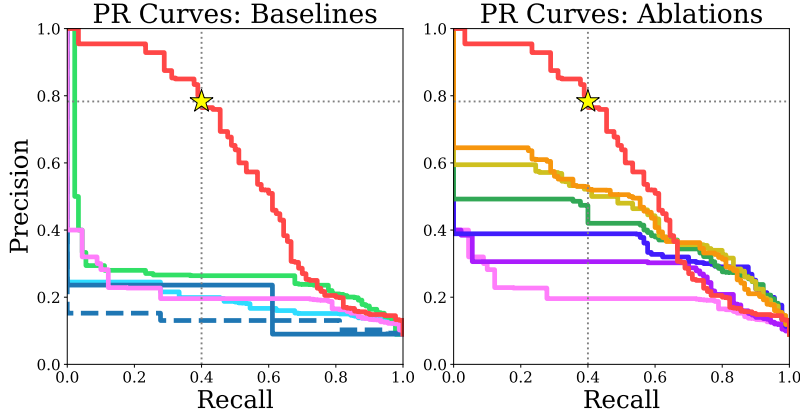


Figure 3. The \star represents DynPose-100K’s operating thresholds. For baselines, we show: ■ Reconstructed points (CamCo [22]), ■ Reprojection error, (solid ■) GPT-4o mini [16]: binary, (dashed ■) GPT-4o mini [16]: score, ■ Hands23 [4], and ■ Ours. For ablations, we begin from ■ Hands23 and add components until we recover ■ Ours. Specifically, we depict: ■ Hands23, ■ +Flow, ■ +Tracking, ■ +Masking, ■ +Focal, ■ +Distort, ■ +VLM (Ours).

Method	Avg. Prec. @ Prec. Rec. >0.40 \uparrow	Prec. @ Prec. Rec. >0.40 \uparrow
Recon. points (CamCo [22])	0.17	0.19
Reprojection error	0.26	0.24
GPT-4o mini [16]: binary	0.18	0.24
GPT-4o mini [16]: score	0.13	0.13
Hands23 [4]	0.19	0.18
Ours	0.58	0.78
Hands23	0.19	0.18
+ Flow	0.27	0.28
+ Tracking	0.34	0.36
+ Masking	0.39	0.47
+ Focal	0.45	0.52
+ Distort	0.46	0.53
+ VLM (Ours)	0.58	0.78

Table 1. **Expanded detail on video filtering on Panda-Test.** Left is Figure 4 from the main paper showing PR curves for baselines and ablations. Right displays average precision for these curves; along with precision at the threshold of 0.40 recall, corresponding to the operating threshold of DynPose-100K.

3. Experimental Details

We expand upon experimental details from §5 in the paper.

3.1. Pose Evaluation on Lightspeed

We run all methods on Lightspeed dataset extracted at full 24 fps due to the short nature of clips, following prior protocol on synthetic data evaluation [2, 14, 25].

Dataset. Additional sequences of Lightspeed are displayed in Figure 4. They have resolution (2560, 1440).

Metrics. We follow [2, 14, 25] in defining failing to register a sequence as registering less than 80% of frames. We fill failed trajectories with random translations so they can be aligned with ground truth trajectories to compute ATE; *i.e.* the identity sequence cannot be transformed to align with another sequence. Unlike Panda-Test, we did not find it necessary to repeat SfM: our method succeeded on all sequences, while repeating SfM on failed sequences for other methods *e.g.* COLMAP made minimal or no difference.

Implementation details. Our pose estimation method predicts tracks using input resolution (480, 854) and a grid of $(56, 32) = 1792$ points, and tracks for 40 frames. This combination is the maximum fitting in 40G memory, and is chosen as a balance of local precision via high-resolution and robustness via long-term tracking. The 24 fps frame-rate means 40 frames is 1.67 seconds, shorter than the 2.5 seconds on DynPose-100K and Panda-Test (detailed in § 1.3). Nevertheless, we find this tracking results in competitive fi-

nal results (Table 2, Figure 8 in the main paper, Figure 8). We use tracking applied both forward and backward, though found this had minimal impact on results compared to tracking only forward.

3.2. Pose Evaluation on Panda-Test

We use the same settings as DynPose-100K and apply masking and tracking upon video frames extracted at 12fps.

Metrics. We select correspondences with small depth (closer to the camera) if possible to make correct pose reprojection error more discernible from incorrect pose estimates. Sample correspondences used for evaluation are plotted in Figure 5. Both human correspondences and corresponding SuperPoint [5]+LightGlue [13] (SP+LG) correspondences are displayed. While human correspondences are annotated by two expert annotators and are reliable at a coarse level, SP+LG allows more precise correspondence. In addition, we require a SP+LG correspondence to exist within 10 pixels (on 720p images) of both human points in correspondence. This is a fail-safe against a missed annotation from the human. In practice, we collect 11,866 pairs, about 86.0% (10,210) of which have agreeing correspondence. We note SP+LG alone would be difficult to annotate correspondence, since correspondences could potentially belong to a dynamic object or be incorrect.

Non-registered predicted frames are replaced by nearby frames, while non-registered sequences are replaced by the identity matrix. Both can result in identity relative pose



Figure 4. **Additional Lightspeed videos.** High resolution (2560, 1440) sequences span indoor and outdoor; light and dark; close-up dynamic object and far-away dynamic object; forward and backward movement.

across image pairs. Identity relative pose results in a singular fundamental matrix, used to compute reprojection error. In this case, instead of reprojection error measuring distance to an epipolar line, distance is computed to the location of the ground truth corresponding point in the opposite image. This can be thought of transforming the correspondence by the identity. We experimented alternatively adding small random perturbations to enable computation of the fundamental matrix, but this resulted in larger error.

Baselines and ablations. We find across methods, SfM occasionally fails on challenging Internet video on the same sequence on which it may succeed. To better analyze performance difference between methods, we therefore repeat failed SfM; defined as registering less than 80% of frames.

Results. In addition to superior pose estimates, we observe our pose estimation method is faster than ParticleSfM [25]. On an A40 GPU, it averages 11 minutes, while ParticleSfM averages 44 minutes, using the same 10 video samples between 48 and 93 frames, with mean length 62.6 (all at 12fps). The speed difference is a result of ParticleSfM’s reliance on propagated dense optical flow for tracking, requiring expensive flow matching to create point tracks. These dense points also result in more correspondences, slowing SfM. Ours takes about 3.3 minutes for tracking, 4.2 minutes for masking and 3.5 minutes for SfM. ParticleSfM takes 26 minutes for tracking, 2 for masking and 16 for SfM. Our quality pose estimates show such dense correspondence is not necessary. A different setup can provide faster speed

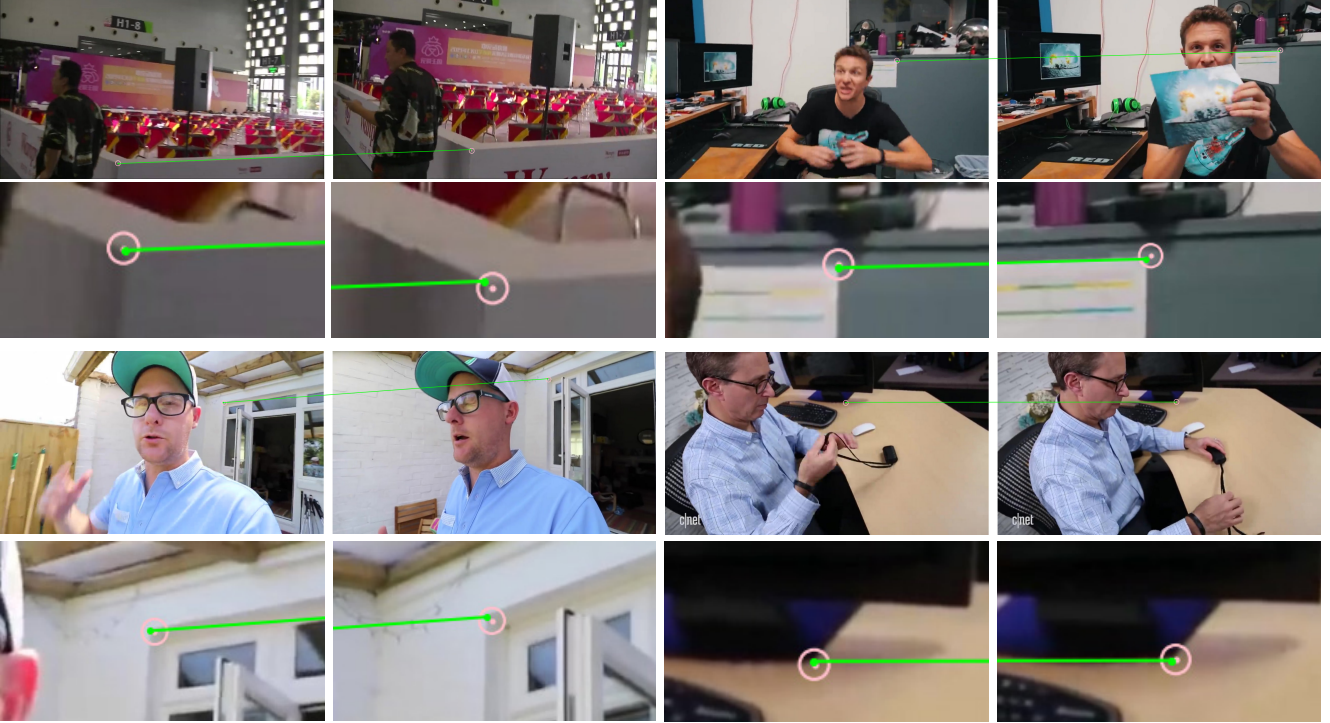


Figure 5. **Panda-Test correspondence annotation.** Human matches (**pink points**) provide coarse accuracy while SuperPoint+LightGlue [5, 13] correspondences (**green points and line**) provide precision. We search for SP+LG matches within 10 pixels of human correspondences on 720p frames (**pink circle**). If no match exists, we do not include the match in testing.

across methods. On an A100 40G GPU in a different infra, we saw our inference in as fast as 4 minutes for a video.

On an A100 40G GPU, filtering one video takes about 0.2min for each of Hands23, flow, focal, VLM, and distort; tracking takes 0.8min and masking takes 1.4min. Filtering is performed sequentially to reduce compute and cost.

Fine-tuning experiment. We compare fine-tuning DUST3R [21] on DynPose-100K against MonST3R [24]. To do so, we follow a similar experimental setup to MonST3R, replacing synthetic data used with DynPose-100K. For depth supervision, we use pixelwise predictions from Depth-Pro [1]. We use median depth values to rescale DynPose-100K translations to meters. We use only 2K of the 100K videos from DynPose-100K for this experiment, finding it produced sufficient performance and highlighted supervision efficiency. We hypothesize using the full dataset could improve performance. We remove videos not containing all registered frames. We also remove those with high reprojection error to improve pose accuracy, inspired by Table 2. We use a reprojection threshold of 1.37, chosen to provide a balance of quantity and quality at about 1.2K videos for training. We train for 250K iterations with batch size 8, which takes about 4 days on 2 A40 GPUs.

4. DynPose-100K: Dataset Analysis Additional Results

We visualize DynPose-100K filtering process and output.

4.1. Filtering Evaluation on Panda-Test

Figure 6 breaks down scores for high and low scoring examples on Panda-Test. The minimum average score for DynPose-100K is 0.91, meaning all filters must score relatively high. Low scoring examples may still have some filters with high scores. This is consistent with the Selection Process from §3.1 in the main paper: only one reason for exclusion is needed for a video to be considered non-target. This also helps visualize why each single filter is not sufficient: R1C6 shows the VLM may not catch zoom-in, R2C2 shows Hands will not handle shot change, R2C5 shows masking will not handle static cameras.

4.2. Dataset Overview

Figure 7 displays sample videos and pose annotations on DynPose-100K. These videos are diverse and face challenges for pose annotation, including varied lighting, movement and apparent dynamic object sizes. Despite this challenge, pose annotations are of high quality.

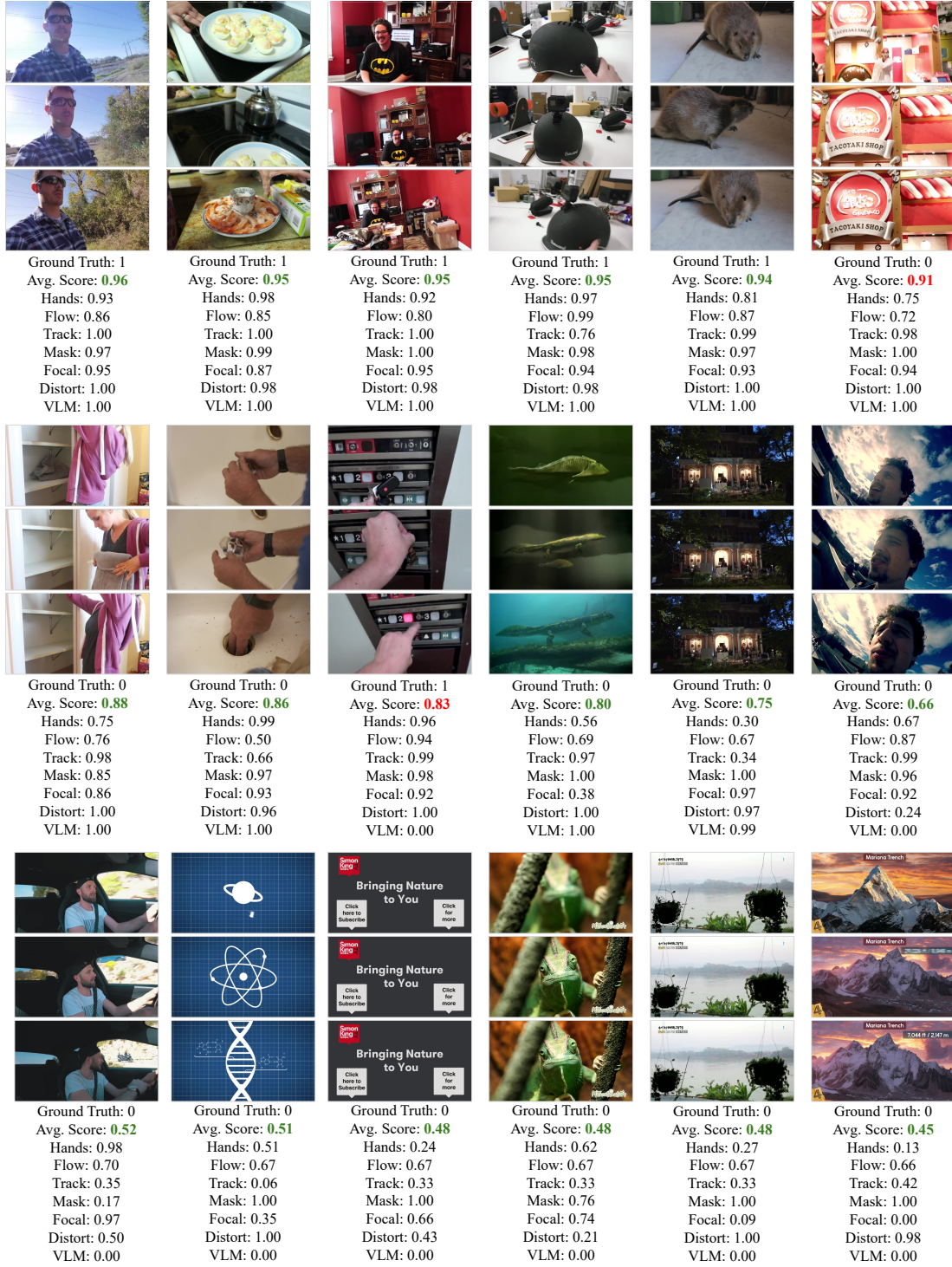


Figure 6. **Panda-Test filtering score samples.** High scoring (top), moderate scoring (middle) and low scoring (bottom) examples from Panda-Test. Ground Truth 1 indicates suitable video, 0 is unsuitable. The minimum average score in DynPose-100K is 0.91, meaning all filters must produce a relatively high score. **G** indicates correct classification based on 0.91 threshold; **R** are sample failure cases. Reasons for exclusion: R1C6: zoom-in, R2C1: static camera, R2C2: shot change, R2C4: not real, R2C5: static camera, R2C6: distortion, R3C1: ambiguous frame of ref, R3C2-C3: not real, R3C4-C5: insufficient clear static region, R2C6: long focal.

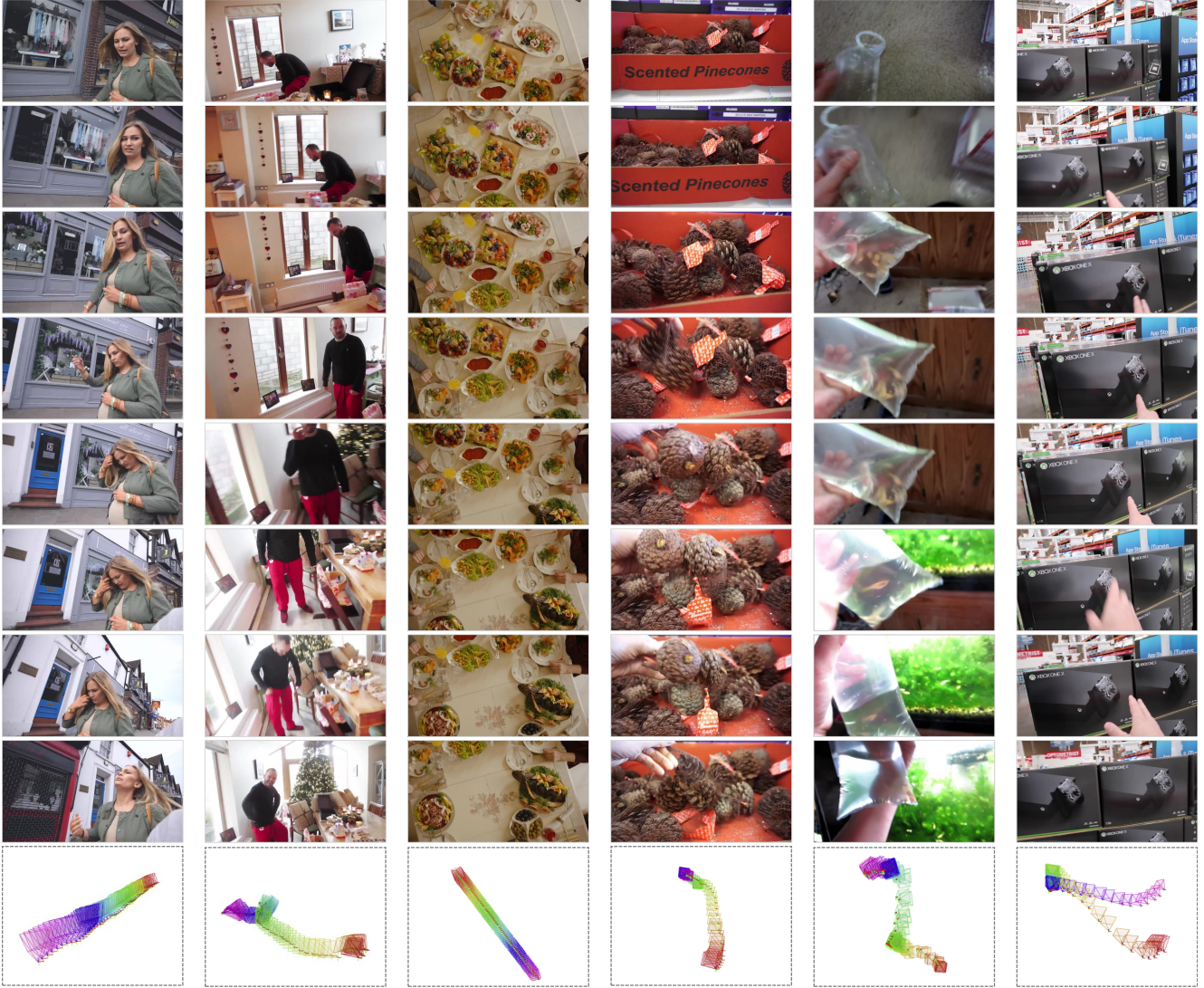


Figure 7. **Sample videos on DynPose-100K.** DynPose-100K collects a diverse set of videos with challenging trajectories and dynamics. It pairs these videos with high-quality pose annotations. The dataset is best viewed via the Supplemental video.

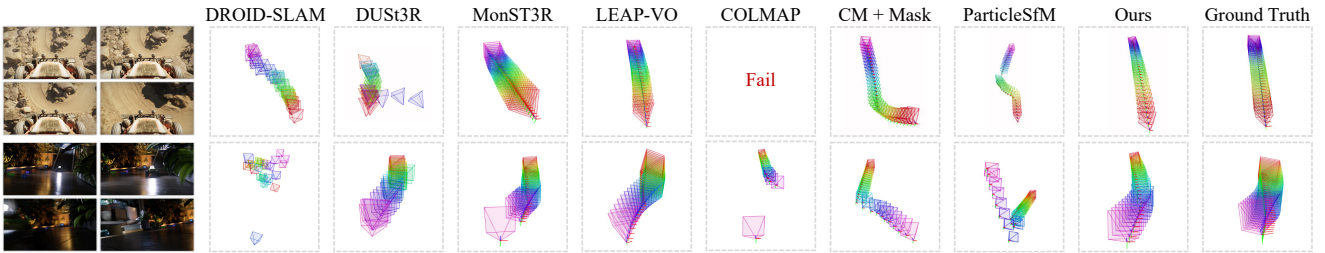


Figure 8. **Additional comparison on Lightspeed.** Ours has more accurate poses than baselines in challenging settings. Top: a dynamic object is static relative to the camera and of similar color to the background. Bottom: dynamic object quickly moves by at night while camera moves and turns. In both cases, baselines are either incorrect or do not have continuous, smooth trajectories. Top: MonST3R curves upward towards the end, while COLMAP+Mask has a large turn.

Reprojection error	%Data \uparrow	% < 5 \uparrow	% < 10 \uparrow	Mean \downarrow
Full test set	100.	72.2	84.4	5.76
Reproj. err. < 1.37	<u>71.1</u>	78.1	<u>84.4</u>	5.04
Reproj. err. < 1.18	41.1	<u>81.1</u>	83.8	<u>4.49</u>
Reproj. err. < 1.00	24.4	81.8	86.4	3.85

Table 2. **Identifying high-quality poses.** Reprojection error is effective in identifying low error videos in Panda-Test. It is useful to produce high-quality subsets of DynPose-100K *e.g.* we use reprojection error to help filter data to fine-tune DUST3R (§ 3.2).

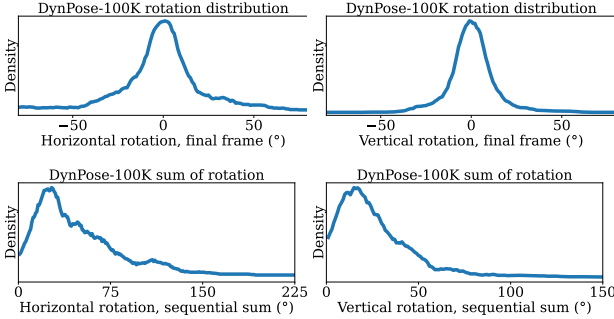


Figure 9. **Dataset statistics.** Top left: distribution of final horizontal rotation. Top right: distribution of final vertical rotation. Bottom left: distribution of sequential sum of absolute value of horizontal rotation. Bottom right: distribution of sequential sum of absolute value of vertical rotation.

4.3. Reprojection Error Analysis

Reprojection error is produced during SfM and is saved for each video in DynPose-100K. It can be used to filter high-quality poses. Table 2 reports pose accuracy results of the proposed method before and after filtering by videos with low reprojection error. Retaining only videos with low reprojection error reduces error significantly. Future users of DynPose-100K may consider filtering by reprojection error.

4.4. Dataset Rotation Analysis

Figure 9 shows video trajectory rotation statistics. Horizontal rotations are particularly diverse, often ending over 30 degrees from the initial rotation with over 75 degrees of total horizontal rotation, measured sequentially.

5. Additional Pose Results

We show additional results on Lightspeed and Panda-Test.

5.1. Pose Evaluation on Lightspeed

Figure 8 displays additional comparisons on Lightspeed. Both examples show all alternatives struggle, while Ours can handle challenges in a large dynamic object that is static relative to the camera (top) and of a dynamic object spinning and kicking up dust (bottom).

Reprojection error	% < 5 \uparrow	% < 10 \uparrow	% < 30 \uparrow	Mean \downarrow
Ours	72.2	84.4	98.9	5.76
+ CoTracker [11]	66.7	78.9	97.8	7.11

Table 3. **Tracking ablation on Panda-Test.** CoTracker produces overall worse camera pose estimates than BootsTAP.

Reprojection error	% < 5 \uparrow	% < 10 \uparrow	% < 30 \uparrow	Mean \downarrow
Ours	72.2	84.4	98.9	5.76
- Semantic	65.6	86.7	<u>97.8</u>	8.14
- Object Interaction	72.2	85.6	97.8	5.79
- Motion	67.8	81.1	94.4	7.95
- Propagation	68.9	86.7	94.4	6.93

Table 4. **Masking ablations on Panda-Test.** Each component is important to final performance: any component removed reduces results on average.

Reprojection error	% < 5 \uparrow	% < 10 \uparrow	% < 30 \uparrow	Mean \downarrow
Ours	72.2	84.4	98.9	5.76
+ GLOMAP [17]	81.1	90.0	95.6	8.86

Table 5. **Bundle adjustment ablation on Panda-Test.** GLOMAP offers competitive precision, but Ours has lower mean error.

5.2. Pose Evaluation on Panda-Test

Figure 11 shows additional comparisons to baselines on Panda-Test. Ours best handles challenges in large dynamic objects and large variations in appearance and lighting. Figure 10 shows additional pose results on Panda-Test. Ours produces reasonable trajectories on a wide variety of videos, agreeing with quantitative results.

Tracking ablation. Table 3 compares our tracking to alternative method CoTracker [11]. We find our tracking produces better pose accuracy.

Masking ablation. Table 4 displays each of the four components to masking. Each component is important to final performance: any component removed reduces results on average. This is also apparent in Figure 2.

Bundle adjustment ablation. Table 5 compares Ours to alternative bundle adjustment method GLOMAP [17]. We find GLOMAP improves precision, but Ours, using Theia-SfM [19], has better mean error. We note in the GLOMAP paper the method does not outperform Theia-SfM across all scenes. Further, the paper measures percentage accuracy within a threshold; our finding on Internet video is GLOMAP failures tend to be more severe than Theia-SfM, which is most clearly reflected in mean error.

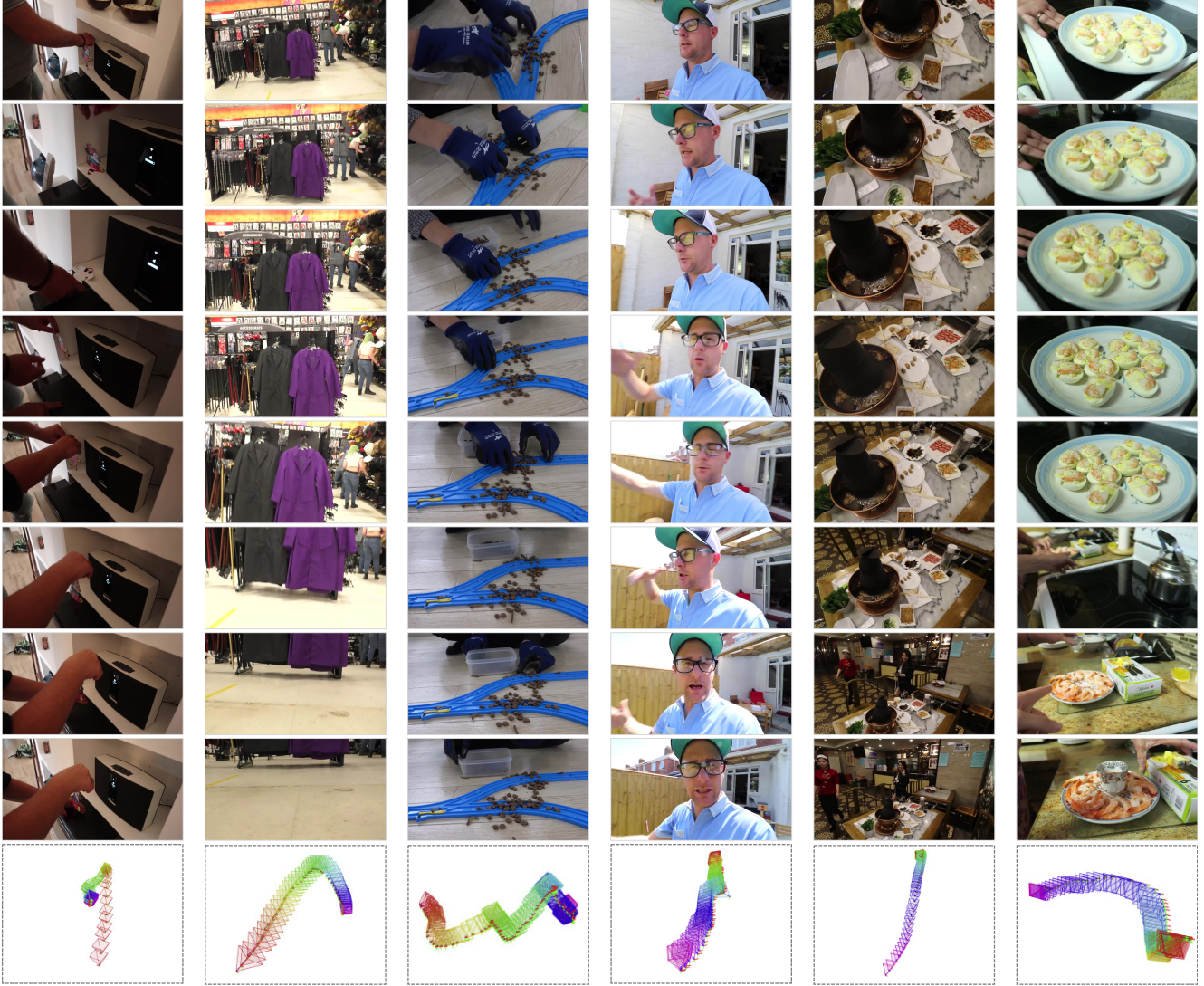


Figure 10. **Additional poses on Panda-Test.** Ours produces sensible poses on a variety of videos, further validating quantitative results.

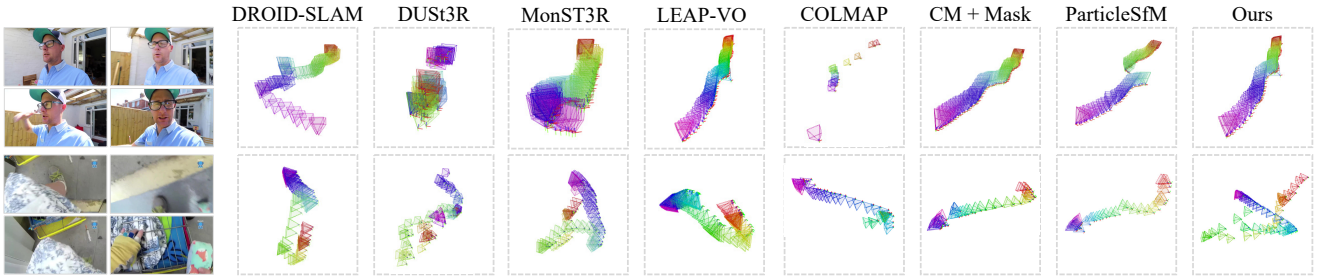


Figure 11. **Additional comparison on Panda-Test.** Ours best handles challenging lighting and large dynamic objects (top), and handling scale variation resulting from moving very close to objects (bottom). In the top sequence, Our trajectory is accurate, while alternatives do not reflect the consistent, fast and mostly straight backward movement of the camera. COLMAP and COLMAP+Mask register most of the bottom sequence, but miss the movement at the end of the trajectory (down and to the left).

References

- [1] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. In *ICLR*, 2025. 7
- [2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 3, 5
- [3] Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, et al. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. In *CVPR*, 2024. 1
- [4] Tianyi Cheng, Dandan Shan, Ayda Hassen, Richard Higgins, and David Fouhey. Towards a richer 2d understanding of hands at scale. *NeurIPS*, 2023. 1, 3, 5
- [5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, 2018. 5, 7
- [6] Carl Doersch, Yi Yang, Dilara Gokay, Pauline Luc, Skanda Koppula, Ankush Gupta, Joseph Heyward, Ross Goroshin, João Carreira, and Andrew Zisserman. Bootstap: Bootstrapped training for tracking-any-point. In *ACCV*, 2024. 4
- [7] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 4
- [8] Annika Hagemann, Moritz Knorr, and Christoph Stiller. Deep geometry-aware camera self-calibration from video. In *ICCV*, 2023. 1
- [9] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3
- [10] Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. Oneformer: One transformer to rule universal image segmentation. In *CVPR*, 2023. 3
- [11] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Co-tracker: It is better to track together. In *ECCV*, 2024. 10
- [12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 3
- [13] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023. 5, 7
- [14] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *CVPR*, 2023. 3, 5
- [15] Christopher Mei and Patrick Rives. Single view point omnidirectional camera calibration from planar grids. In *ICRA*, 2007. 1
- [16] OpenAI. Gpt-4o mini: advancing cost-efficient intelligence, 2024. 3, 5
- [17] Linfei Pan, Dániel Baráth, Marc Pollefeys, and Johannes L Schönberger. Global structure-from-motion revisited. In *ECCV*, 2024. 10
- [18] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv*, 2024. 3
- [19] Chris Sweeney. Theia multiview geometry library: Tutorial & reference. <http://theia-sfm.org>. 3, 10
- [20] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 2, 3
- [21] Shuzhe Wang, Vincent Leroy, Yann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024. 7
- [22] Dejia Xu, Weili Nie, Chao Liu, Sifei Liu, Jan Kautz, Zhangyang Wang, and Arash Vahdat. Camco: Camera-controllable 3d-consistent image-to-video generation. *arXiv*, 2024. 5
- [23] Hongwei Xue, Tiankai Hang, Yanhong Zeng, Yuchong Sun, Bei Liu, Huan Yang, Jianlong Fu, and Baining Guo. Advancing high-resolution video-language representation with large-scale video transcriptions. In *CVPR*, 2022. 1
- [24] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. In *ICLR*, 2025. 7
- [25] Wang Zhao, Shaohui Liu, Hengkai Guo, Wenping Wang, and Yong-Jin Liu. Particlesfm: Exploiting dense point trajectories for localizing moving cameras in the wild. In *ECCV*, 2022. 3, 5, 6
- [26] Shengjie Zhu, Abhinav Kumar, Masa Hu, and Xiaoming Liu. Tame a wild camera: In-the-wild monocular camera calibration. In *NeurIPS*, 2023. 1