EGOACT4D: ESTIMATING BIMANUAL 4D HAND MOTION FROM EGOCENTRIC VIDEOS

Anonymous authors
Paper under double-blind review

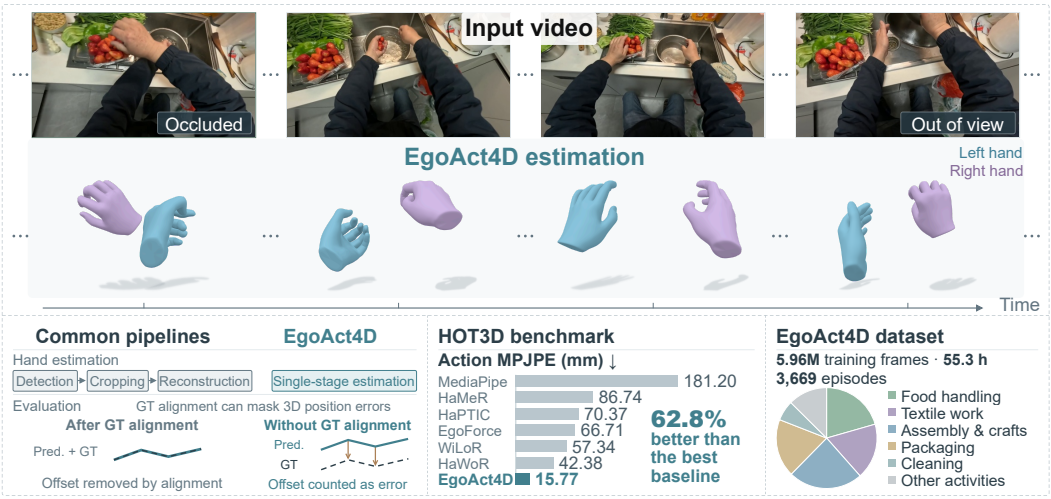


Figure 1: **EgoAct4D overview. Top:** Video frames and schematic bimanual motion in each window's first-frame camera coordinates. **Bottom left:** Simpler estimation pipeline and direct evaluation compared to common pipelines. **Bottom center:** Action MPJPE on the HOT3D benchmark. **Bottom right:** EgoAct4D dataset for studying full-pipeline data scaling.

ABSTRACT

Turning egocentric video into action supervision requires accurate and efficient hand motion estimation. Multi-camera capture and wearable sensing provide accurate motion supervision but are costly to scale. Monocular reconstruction could enable large-scale supervision from abundant Internet videos, but existing pipelines require multiple stages and incur motion errors of several centimeters. To address these limitations, we present EgoAct4D, a simpler pipeline for more accurate bimanual 4D hand motion estimation from monocular egocentric RGB videos. Its single-stage hand estimator predicts bimanual states from short clips, including occluded and out-of-view poses, while a separate estimator directly predicts metric camera poses. The pipeline reuses hand predictions and stitches overlapping camera trajectories to express hand motion in each window's first-frame camera coordinates and pair it with the starting observation. Since existing alignment-based evaluations can mask position and motion errors, we measure camera-space joint error (Absolute MPJPE) and hand-trajectory error (Action MPJPE) in sliding windows without ground-truth alignment. EgoAct4D achieves state-of-the-art results on both metrics on the HOT3D benchmark. We then build the EgoAct4D dataset and demonstrate full-pipeline data scaling beyond HOT3D's training scale. On its test set, EgoAct4D achieves 7.95 mm Absolute MPJPE and 10.98 mm Action MPJPE at an end-to-end throughput of 49.3 FPS. This accuracy and efficiency can support reusable motion supervision for vision-language-action pretraining and human-to-robot motion retargeting.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Large language models have benefited from large-scale training on Internet text (Brown et al., 2020; Kaplan et al., 2020). Abundant egocentric videos on the Internet offer a similar opportunity for robot learning through diverse human demonstrations. Using these videos for action-supervised robot learning (Kim et al., 2025) requires pairing visual observations with action targets. Recovering hand motion provides such supervision from human videos (Macrodata Labs, 2026) and supports human-to-robot motion retargeting (Wang et al., 2026b). This requires accurate motion labels that can be generated efficiently from long videos.

Monocular reconstruction can recover hand motion from existing egocentric videos without the dedicated hardware required by multi-camera capture or wearable sensing (Banerjee et al., 2025; Hoque et al., 2026). However, existing monocular methods can accurately recover hand articulation yet still incur several-centimeter errors in 3D hand placement, especially in depth (Millerdurai et al., 2026; Ye et al., 2026), which can limit their value for precise manipulation. Common pipelines use hand detection, cropping, and reconstruction (Pavlakos et al., 2024; Potamias et al., 2025), often followed by tracking, temporal completion, camera estimation, and separate scale recovery (Zhang et al., 2025). These stages increase computational cost and system complexity when estimating hand motion at scale.

Motivated by these limitations, we present EgoAct4D, a simpler pipeline for more accurate bimanual 4D hand motion estimation from monocular egocentric videos (Figure 1). To reduce depth errors, our hand estimator combines a Depth Anything 3 encoder pretrained for metric depth (Lin et al., 2026) with a structured temporal decoder (Figure 2(a)). The decoder uses a pair of query vectors per frame to represent the left and right hands. These queries read the corresponding frame's image features and exchange information across time and between hands. Given short RGB video clips, the hand estimator predicts camera-space 3D joint positions and wrist orientations for both hands, even when a hand is occluded or out of view. A separate camera estimator (Figure 2(b)) directly predicts metric poses from short clips, without requiring long video sequences or separate scale estimation.

To process continuous video, EgoAct4D reuses hand predictions across overlapping windows, avoiding repeated inference (Figure 2(c)). Camera stitching transforms overlapping local trajectories into a common reference frame at fixed scale and fuses them into a dense trajectory. When available, extrinsics from devices such as Apple Vision Pro or Meta Quest 3 can supply the camera trajectory directly, bypassing camera estimation and stitching. This can further reduce camera-motion errors. Using these camera poses, we express bimanual 3D joint trajectories and wrist orientations in each output window's first-frame camera coordinates (Macrodata Labs, 2026). Each window pairs its starting observation with the estimated hand motion.

Existing evaluations often measure local hand articulation and trajectory similarity after pose and scale alignment (Zhang et al., 2025; Zeng et al., 2026; Sun et al., 2026). This can mask errors in absolute 3D placement and motion. We therefore evaluate sliding windows without prediction-to-ground-truth alignment. Absolute MPJPE (Millerdurai et al., 2026) measures camera-space hand accuracy, while Action MPJPE (Macrodata Labs, 2026), adapted to these windows, measures hand trajectories and captures camera-estimation errors. Under this protocol, we evaluate on the HOT3D benchmark. To study full-pipeline data scaling beyond HOT3D's training scale, we construct the EgoAct4D dataset with paired hand and camera annotations. Our contributions are:

- We propose EgoAct4D, a simpler pipeline for more accurate bimanual 4D hand motion estimation from monocular egocentric videos. It reuses hand predictions and stitches camera trajectories, achieving an end-to-end throughput of 49.3 FPS on one RTX PRO 6000 GPU.
- We introduce a sliding-window evaluation protocol using Absolute and Action MPJPE without ground-truth alignment to assess position and motion errors relevant to action supervision. EgoAct4D achieves state-of-the-art results on both metrics on the HOT3D benchmark under this protocol.
- We construct the EgoAct4D dataset and demonstrate full-pipeline data scaling beyond HOT3D's training scale. On its test set, EgoAct4D achieves 7.95 mm Absolute MPJPE and 10.98 mm Action MPJPE with predicted camera poses, yielding approximately 1 cm motion accuracy. This accuracy can support reusable motion supervision for vision-language-action pretraining and human-to-robot motion retargeting.

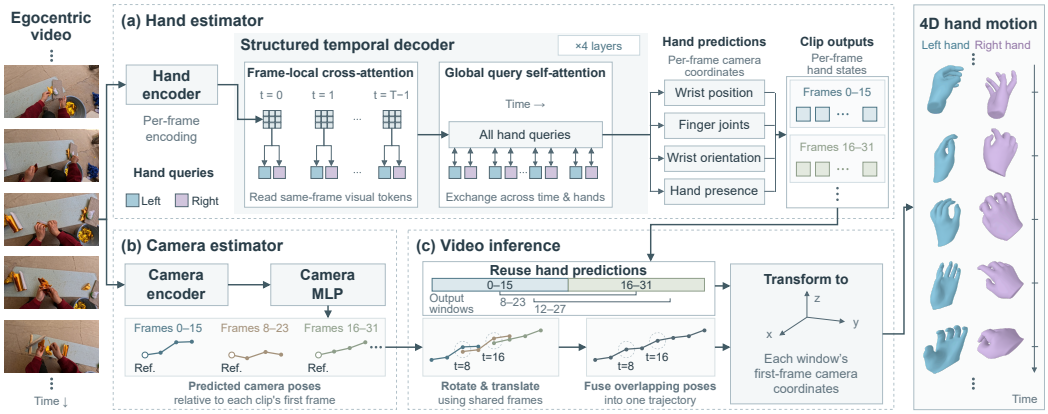


Figure 2: **EgoAct4D architecture and video inference. (a)** The hand estimator predicts per-frame bimanual states from short clips. **(b)** The camera estimator predicts metric poses relative to each clip's first frame. **(c)** Video inference reuses hand predictions and stitches overlapping camera trajectories. Each output window pairs its starting image with hand motion expressed in that image's camera coordinates. The right column shows a schematic of 4D hand motion.

2 Related Work

Datasets and hand motion supervision. Egocentric datasets pair videos with hand-pose annotations obtained through different capture systems. AssemblyHands (Ohkawa et al., 2023) and H2O (Kwon et al., 2021) use multi-view imagery, HOT3D (Banerjee et al., 2025) uses OptiTrack motion capture, and EgoDex (Hoque et al., 2026) uses Apple Vision Pro cameras and ARKit tracking. These annotations support training and evaluating monocular hand-motion estimators. Both annotated and reconstructed hand motion can also provide supervision for robot learning. Macrodata (Macrodata Labs, 2026) extracts camera-anchored hand actions, while Ego2Robot (Wang et al., 2026b) retargets reconstructed or annotated hand poses to robot end-effector trajectories.

Hand pose and motion estimation. HaMeR and WiLoR reconstruct 3D hands after hand detection and cropping (Pavlakos et al., 2024; Potamias et al., 2025), while EgoForce (Millerdurai et al., 2026) uses forearm context for absolute camera-space positioning. For videos, HaWoR (Zhang et al., 2025) combines hand and camera reconstruction with motion infilling. HaPTIC (Ye et al., 2026) uses spatial and temporal attention to infer 4D trajectories in camera coordinates, while StableHand (Zeng et al., 2026) uses quality-aware flow matching to refine and complete reconstructed trajectories.

Camera pose estimation. Camera poses allow hand states from different frames to be expressed in a common coordinate system for action supervision. Sequence-based optimization combines tracking with geometric refinement. DROID-SLAM (Teed & Deng, 2021) estimates camera poses and depth through dense bundle adjustment, while HaWoR (Zhang et al., 2025) combines monocular SLAM with metric depth to recover translation scale. Feed-forward models such as VGGT (Wang et al., 2025) and the any-view models of Depth Anything 3 (Lin et al., 2026) predict cameras and scene geometry directly from input images. Their scale-ambiguous predictions require additional scale recovery for metric use, for example by fitting predicted depth to metric-depth estimates. MapAnything (Keetha et al., 2026) directly predicts metric geometry and cameras. Macrodata (Macrodata Labs, 2026) uses windowed camera estimation and stitching to generate hand-motion supervision.

3 Method

EgoAct4D estimates bimanual 4D hand motion from monocular egocentric RGB videos using separate hand and camera estimators. The hand estimator predicts per-frame 3D joint positions, wrist orientations, and hand presence, while the camera estimator predicts metric poses for coordinate conversion. The video pipeline assembles their short-clip predictions into motion expressed in each output window's first-frame camera coordinates (Figure 2).

3.1 PROBLEM FORMULATION

Given a monocular egocentric RGB clip of T frames, our goal is to estimate bimanual hand states and camera poses. We use the MANO skeleton (Romero et al., 2017) augmented with five fingertips, giving 21 hand joints: one wrist joint and 20 finger joints. For each hand $h \in \{\mathrm{L}, \mathrm{R}\}$ and frame t, we predict 3D joint positions (in meters) and wrist orientation in that frame's camera coordinates. We also predict the probability $p_{t,h} \in [0, 1]$ that the hand is in frame. Temporal context enables 3D state prediction even when the hand is out of view. For each output window, we express hand motion in the camera coordinate system C_s of its first frame s. For a joint at frame t, let $\mathbf{x}_t$ denote its position in camera frame C_t and $\mathbf{y}_{s,t}$ its position in C_s, omitting hand and joint indices. The relative pose $\mathbf{T}_{s,t}$ maps C_t to C_s with rotation $\mathbf{R}_{s,t}$ and translation $\mathbf{d}_{s,t}$:

$$\mathbf{y}_{s,t} = \mathbf{R}_{s,t}\mathbf{x}_t + \mathbf{d}_{s,t}. \tag{1}$$

Wrist orientations are rotated by the same $\mathbf{R}_{s,t}$. Each output window pairs its starting RGB frame with the estimated bimanual hand states across the window, expressed in the starting frame's camera coordinates. We retain timestamps and predicted hand-presence probabilities for action supervision.

3.2 HAND ESTIMATION

Our single-stage hand estimator (Figure 2(a)) combines an encoder initialized from DA3Metric-Large, the monocular metric-depth variant of Depth Anything 3 (DA3) (Lin et al., 2026), with a four-layer structured temporal decoder. The encoder processes uncropped RGB frames independently. A learned projection converts each frame's dense features into visual tokens $\mathbf{M}_t$. The temporal decoder uses two hand-state queries per frame, one per hand. To distinguish queries across frames and between hands, we add learned frame and hand embeddings to each query $\mathbf{q}_{t,h}$. The decoder then updates these queries in each layer through frame-local cross-attention (CA), followed by global query self-attention (SA):

$$\widetilde{\mathbf{Q}}_t = \mathrm{CA}(\mathbf{Q}_t, \mathbf{M}_t), \qquad \mathbf{Q}' = \mathrm{SA}(\widetilde{\mathbf{Q}}). \tag{2}$$

In Eq. (2), $\mathbf{Q}_t$ contains the two queries for frame t. Cross-attention allows these queries to read only that frame's visual tokens $\mathbf{M}_t$. Its outputs are stacked as $\widetilde{\mathbf{Q}}$. Self-attention updates all $2T$ queries to obtain $\mathbf{Q}'$, using temporal and bimanual context when local visual evidence is weak. From the final queries, four separate heads predict wrist position, 20 finger-joint positions, wrist orientation, and hand presence. Positions and orientations are expressed in the current frame's camera coordinates. More details on the architecture and queries are given in Appendix A.1.

3.3 CAMERA ESTIMATION

The camera estimator (Figure 2(b)) initializes a separate encoder from DA3-Large, the any-view DA3 model. The encoder processes each short clip using attention within individual frames and across frames. We fine-tune this encoder with a camera MLP that reads one token per frame to directly predict rotation and metric translation. The estimator requires neither long video sequences nor separate scale estimation. We express the predicted poses relative to each input clip's first camera frame. Appendix A.1 describes the camera MLP outputs and translation normalization in more detail.

3.4 TRAINING AND LOSSES

We train the hand and camera estimators separately for two epochs and one epoch, respectively. We fine-tune the encoders with LoRA (Hu et al., 2022), while fully training the hand decoder, prediction heads, and camera MLP.

Hand supervision. We supervise joint positions, wrist orientations, and hand presence with

$$\begin{aligned}\mathcal{L}_{\mathrm{hand}} = \mathcal{L}_{\mathrm{wrist}} + 2\mathcal{L}_{\mathrm{finger}} + \mathcal{L}_{\mathrm{relative}} \\ + 0.5\mathcal{L}_{\mathrm{rotation}} + \lambda_{\mathrm{proj}}\mathcal{L}_{\mathrm{proj}} + 0.05\mathcal{L}_{\mathrm{presence}}.\end{aligned} \tag{3}$$

In Eq. (3), $\mathcal{L}_{\mathrm{wrist}}$ and $\mathcal{L}_{\mathrm{finger}}$ supervise camera-space wrist and finger positions, respectively, while $\mathcal{L}_{\mathrm{relative}}$ supervises finger-to-wrist offsets to better preserve hand articulation. The remaining terms supervise wrist orientation ($\mathcal{L}_{\mathrm{rotation}}$), valid 2D joint reprojections ($\mathcal{L}_{\mathrm{proj}}$), and in-frame hand presence ($\mathcal{L}_{\mathrm{presence}}$). The reprojection weight λ_{proj} rises from 0 to 0.2 over the first 10% of training.

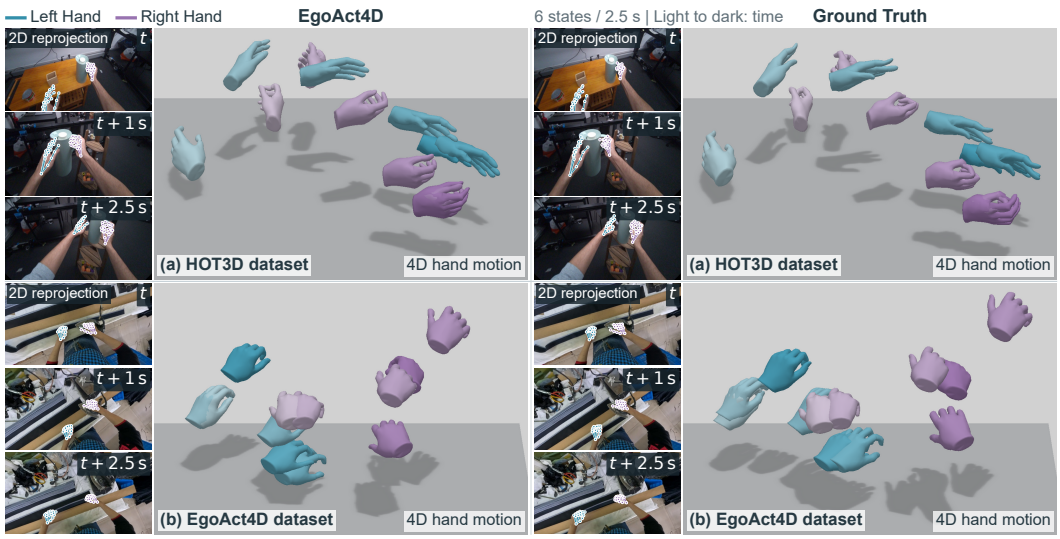


Figure 3: Visual comparison of EgoAct4D predictions (left) and ground truth (right) on the HOT3D **(a)** and EgoAct4D **(b)** datasets. Selected 2D joint reprojections accompany six hand-motion states sampled at 0.5 s intervals over 2.5 s. Blue and purple denote left and right hands; darker shades indicate later times in the 4D views. Predictions use estimated camera poses without GT alignment.

Camera supervision. We train the camera estimator to predict each frame's position and orientation relative to the clip's first frame. We also supervise the camera motion between consecutive frames to constrain local motion. These two terms use first-frame pairs $\mathcal{E}_0 = \{(0, t)\}_{t=1}^{T-1}$ and adjacent-frame pairs $\mathcal{E}_1 = \{(t-1, t)\}_{t=1}^{T-1}$, respectively:

$$\mathcal{L}_{\text{camera}} = \mathcal{L}_{\text{pose}}(\mathcal{E}_0) + 0.25\mathcal{L}_{\text{pose}}(\mathcal{E}_1) + 0.05\mathcal{L}_{\text{FoV}}. \tag{4}$$

In Eq. (4), $\mathcal{L}_{\text{pose}}$ supervises relative camera translation and rotation for each pair, while $\mathcal{L}_{\text{FoV}}$ supervises the predicted field of view using targets derived from camera intrinsics. Appendix A.2 provides the detailed loss definitions, normalization, and optimization settings.

3.5 Efficient video inference

Independently processing overlapping output windows repeats computation, while camera predictions from different input clips use different reference frames. We therefore reuse hand predictions and stitch camera trajectories to estimate motion efficiently over continuous video (Figure 2(c)). The hand estimator runs at stride 16, reusing its per-frame predictions across output windows. The camera estimator uses overlapping clips at stride 8. Using predicted poses at shared frames, we rotate and translate local trajectories into a common reference frame at fixed metric scale. We fuse overlapping estimates into a dense trajectory $\{\mathbf{G}_t\}$, where $\mathbf{G}_t$ maps C_t to this reference frame.

For a window of $L = 16$ frames starting at s, we map C_t to its first camera frame C_s via

$$\mathbf{T}_{s,t} = \mathbf{G}_s^{-1}\mathbf{G}_t, \qquad t = s, \ldots, s + L - 1. \tag{5}$$

Here, $\mathbf{G}_s^{-1}$ maps the common reference frame back to C_s. We use Eq. (5) in Eq. (1) to express hand motion in C_s. Calibrated device extrinsics can replace camera estimation and stitching. Inference details and stride ablations are given in Appendix B.1 and Section 4.4.

4 Experiments

4.1 Experimental setup

Datasets. We compare methods on HOT3D's Project Aria recordings (Banerjee et al., 2025) and study full-pipeline accuracy and data scaling on the EgoAct4D dataset. Figure 3 compares predictions with ground truth on both datasets through time-matched 2D reprojections and 4D views.

Table 1: Motion estimation on the HOT3D benchmark. Lower is better; best results are **bold** and second-best underlined.

	Position (mm)			Temporal ($\mathrm{m/s^2}$)	Wrist rotation (deg)	
Method	Abs.	Act.	PA	Accel.	Abs.	Act.
HaWoR	42.296	42.382	**6.526**	16.212	10.593	10.590
WiLoR	57.327	57.335	7.284	14.934	14.791	14.783
EgoForce	66.697	66.707	7.023	27.418	20.255	20.238
HaMeR	86.978	86.740	9.494	35.284	37.079	37.060
HaPTIC	70.365	70.373	9.917	11.101	35.580	35.566
MediaPipe	181.246	181.202	19.945	55.656	–	–
EgoAct4D	**15.558**	**15.773**	6.962	**5.830**	**6.288**	**6.290**

Metrics. We evaluate hand motion in 16-frame windows. Absolute MPJPE (Abs.) measures 3D joint-position error in each frame's camera coordinates over all 16 frames (Millerdurai et al., 2026). Action MPJPE (Act.) measures hand-trajectory error over the subsequent 15 frames, using the first frame's camera coordinates as the reference (Macrodata Labs, 2026). Predictions and ground truth (GT) use their respective camera trajectories for this coordinate conversion. Neither metric uses wrist centering or prediction-to-GT alignment. Both errors are in mm, averaged equally over eligible hand–window pairs. PA-MPJPE (PA) evaluates local hand-pose accuracy after aligning each predicted hand to GT in rotation, translation, and scale, rather than its absolute 3D placement. We also report acceleration error (Accel.) and wrist rotation errors (degrees): Abs. uses each frame's camera coordinates; Act. uses the starting frame's camera coordinates over the subsequent 15 frames. Appendix B.2 provides detailed definitions of all evaluation metrics. Appendices D.1 and D.2 additionally report W-MPJPE, WA-MPJPE, and reprojection error.

Video evaluation. We evaluate the complete continuous-video pipeline using 16-frame inputs. The hand estimator runs at stride 16 to reuse predictions; the camera estimator runs at stride 8, retaining eight shared frames for stitching. Section 4.4 supports this setting, showing higher throughput with similar position and motion accuracy. We then score every valid 16-frame window at stride 1 using the assembled predictions, without rerunning either estimator.

We report two evaluation settings. The full set includes a hand if at least two GT joints are in front of the camera and project inside the image in at least two frames of the window. All its valid annotated states are scored, including occluded and out-of-view states. This setting tests motion accuracy and recovery when visual evidence is missing. Following the in-frame selection principle of HaWoR's released HOT3D evaluation (Zhang et al., 2025), we define an In-frame subset that retains windows in which both hands meet the two-joint condition in every frame. Occluded states remain included. We report both settings on HOT3D and the full set on the EgoAct4D dataset.

4.2 Comparison on the HOT3D benchmark

We benchmark bimanual hand-motion estimation on HOT3D under the video evaluation protocol described in Section 4.1. We compare EgoAct4D with HaWoR (Zhang et al., 2025), WiLoR (Potamias et al., 2025), EgoForce (Millerdurai et al., 2026), HaMeR (Pavlakos et al., 2024), HaPTIC (Ye et al., 2026), and MediaPipe Hands (Zhang et al., 2020) on HOT3D. Among these baselines, only HaWoR includes a camera-motion estimation module. However, its released HOT3D evaluation uses GT-derived hand bounding boxes and aligns predicted SLAM segments to GT camera trajectories. To ensure a fair comparison without GT-assisted inference, we pair all methods' hand predictions with the same trajectory from our HOT3D-trained camera estimator. This lets us compare hand-motion accuracy using the same camera estimates and stride-1 evaluation windows. Appendix B.1 details baseline processing and hand-track completion.

In-frame evaluation. Table 1 shows that EgoAct4D achieves the lowest Absolute and Action MPJPE, at 15.558 and 15.773 mm, respectively. HaWoR is the strongest baseline on both metrics, at 42.296 and 42.382 mm, so our method reduces these errors by 63.2% and 62.8%. HaWoR achieves lower PA-MPJPE (6.526 versus 6.962 mm), which removes translation, rotation, and scale errors before scoring. This difference illustrates how aligned evaluations can mask errors in 3D

Table 2: Motion estimation on the EgoAct4D benchmark. End-to-end throughput is measured in FPS on one RTX PRO 6000 GPU.

Method	Position (mm)			Temporal (m/s^2)	Wrist rotation (deg)		End-to-end FPS
	Abs.	Act.	PA	Accel.	Abs.	Act.	
EgoAct4D	7.951	10.975	3.367	2.703	3.095	3.140	49.26

hand placement and motion that matter for action supervision. Complete In-frame results are given in Table 11 in Appendix D.2.

Full-set evaluation. We next evaluate the broader set that also includes valid annotated out-of-view hand states. Table 10 in Appendix D.2 reports the complete comparison. EgoAct4D achieves 19.396 mm Absolute MPJPE and 19.606 mm Action MPJPE, compared with 53.306 and 53.300 mm from HaWoR, again the strongest baseline on both metrics. These correspond to error reductions of 63.6% and 63.2%, respectively. The remaining baselines have Action MPJPE ranging from 71.146 to 208.906 mm. Our advantage over HaWoR in Action MPJPE grows from 26.609 mm on the In-frame subset to 33.694 mm on the full set. Our method thus retains lower motion error when out-of-view hand states are included in the evaluation.

4.3 Motion estimation accuracy and data scaling

We construct the EgoAct4D dataset to study full-pipeline data scaling beyond HOT3D's training scale. It pairs egocentric RGB videos with annotations of 3D joints, wrist orientations, and camera poses across a range of everyday manipulation activities. All training subsets share a 2.9-hour test set, allowing comparison across training scales on the same evaluation data. Appendix C.1 provides dataset statistics and examples.

Motion estimation benchmark. The test set and our evaluation protocol provide a benchmark for comparing future methods for bimanual 4D hand motion estimation. With both estimators trained on the full training split, the default video pipeline achieves 7.95 mm Absolute MPJPE and 10.98 mm Action MPJPE on the EgoAct4D benchmark at an end-to-end throughput of 49.3 FPS on one RTX PRO 6000 GPU (Table 2). These results show that the pipeline estimates hand motion with near-centimeter accuracy while processing video efficiently. Section 4.4 examines how the hand estimator, camera source, and inference settings contribute to this performance.

Data scaling. We train on four strictly nested subsets of 3.3, 8.5, 21.7, and 55.3 hours, evenly spaced on a logarithmic scale. The smallest approximately matches HOT3D's training frame count; the largest is the full training set. At each scale, both estimators use the same subset. Training settings, architecture, preprocessing, test data, and video evaluation remain fixed. Both primary errors decrease at every scale (Figure 4). From 3.3 to 55.3 hours, Absolute MPJPE falls by 41.6% and Action MPJPE by 35.1%, demonstrating full-pipeline data scaling. The consistent improvements across all four subsets show that both frame-wise hand placement and motion trajectories benefit from additional training data. Table 9 reports complete video results. These gains highlight the potential of the EgoAct4D pipeline to improve motion estimation with more training data.

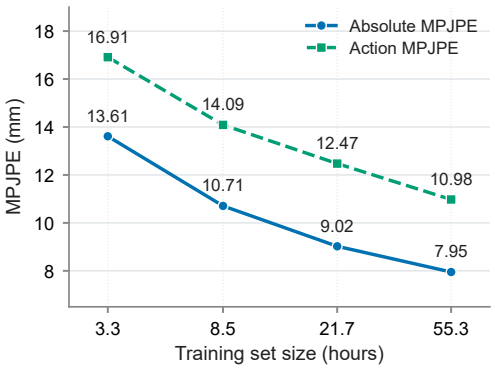


Figure 4: Full-pipeline accuracy versus training set size (logarithmic x-axis).

4.4 Ablations

Hand-estimator design. Our goal is to estimate bimanual hand motion in a single stage, making the choice of encoder and decoder central to the design. We therefore examine how encoder pretrain-

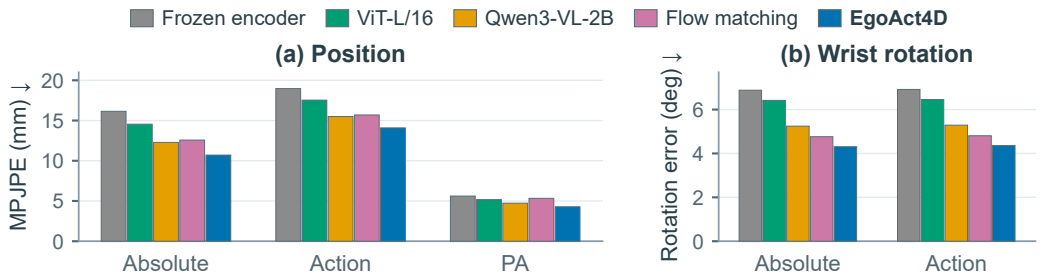


Figure 5: Hand-estimator variants trained on 8.5 hours of the EgoAct4D dataset and evaluated with the same camera trajectory. (a) Position errors. (b) Wrist-rotation errors. Lower is better.

ing, task-specific adaptation, and decoder design affect estimation accuracy. We train all variants on the 8.5-hour subset for two epochs and evaluate them with the same camera trajectory (Figure 5). We compare four alternatives: freezing the DA3Metric-Large hand encoder to test the need for task-specific adaptation; replacing it with an ImageNet-21k-pretrained ViT-L/16 (Dosovitskiy et al., 2021) while retaining our structured temporal decoder; using Qwen3-VL-2B (Bai et al., 2025) as a video encoder; and replacing direct prediction with a flow-matching decoder. Appendix D.3 gives implementation details and complete numerical results (Table 12). The default estimator achieves the lowest position and wrist-rotation errors. Action MPJPE is 14.087 mm with our default design, 18.970 mm with a frozen hand encoder, and 17.536 mm with ViT-L/16. These results support adapting the encoder to hand estimation and using geometry-pretrained features. Despite its larger model size, Qwen3-VL-2B improves only acceleration error (2.800 versus 3.439 $\mathrm{m/s^2}$), possibly benefiting from its video encoder's temporal modeling. This advantage does not translate into more accurate 3D hand estimates: all position and wrist-rotation errors remain higher than those of our default design. Flow matching yields 15.692 mm Action MPJPE and 7.237 $\mathrm{m/s^2}$ acceleration error, favoring direct prediction in this setting.

Wrist position estimation beyond pretrained depth. Our hand encoder is initialized from DA3Metric-Large. We test whether our trained estimator estimates 3D wrist positions more accurately than using this pretrained model's depth maps directly (Table 3). For this baseline, we read depth at GT 2D wrist locations and use camera intrinsics to recover their 3D positions. We use wrists because finger joints are often occluded. Some wrists may also be occluded, so we additionally examine cases where depth-based reconstruction is most accurate. We sort its predictions by 3D wrist error against GT and report the best 25%, 50%, and 75% alongside the full set. Even its best 25% has 27.67 mm 3D wrist error and 25.26 mm depth MAE, whereas our estimator achieves 8.06 and 5.58 mm on the full set. Our estimator therefore predicts wrist positions and depths more accurately than directly using the pretrained depth maps.

Table 3: Wrist position estimation on the EgoAct4D test set (mm, lower is better). The depth baseline uses GT 2D wrist locations; percentages select the samples with its lowest 3D wrist errors.

	EgoAct4D	DA3Metric-Large			
Metric	Full	Best 25%	Best 50%	Best 75%	Full
3D error	**8.065**	27.670	48.732	67.496	93.502
Depth MAE	**5.582**	25.263	45.114	62.767	87.541

Camera-source comparison. Camera errors propagate into hand motion when predictions are transformed across frames. To assess how camera estimates affect hand-motion accuracy, we fix hand predictions from the estimator trained on the full EgoAct4D training split and vary only the camera source on its test set (Table 4). All sources use 16-frame input clips, camera stride 8, and the same scale-fixed stitching. DA3-Large (Lin et al., 2026) and VGGT-Ω (Wang et al., 2026a) predict camera poses up to scale. We recover metric scale for both using depth estimates from DA3Metric-Large. In contrast, MapAnything (Keetha et al., 2026) and our estimator predict metric camera poses directly. Cam. trans. and Cam. rot. measure camera translation and rotation errors relative to each

Table 4: Comparison of camera sources on the EgoAct4D dataset, with the same hand estimator and other settings. End-to-end throughput is measured in FPS on one RTX PRO 6000 GPU.

Camera source	Cam. trans. (mm)↓	Cam. rot. (deg)↓	Action MPJPE (mm)↓	Accel. ($\mathrm{m/s^2}$)↓	End-to-end FPS↑
MapAnything	19.547	0.762	21.408	6.498	18.51
VGGT-Ω + DA3Metric-Large	7.852	0.627	12.989	2.832	5.72
DA3-Large + DA3Metric-Large	8.706	0.612	13.259	3.217	9.51
EgoAct4D camera estimator	**5.062**	**0.328**	**10.975**	**2.703**	**49.26**

output window's first-frame camera coordinates. Our camera estimator achieves the lowest errors across all four metrics. Its camera translation and rotation errors are 5.062 mm and 0.328°. With identical hand predictions, our camera estimator achieves 10.975 mm Action MPJPE, compared with 12.989 mm for VGGT-Ω, the strongest alternative on this metric. Acceleration error is also lower: 2.703 versus 2.832 $\mathrm{m/s^2}$. The resulting pipeline also achieves an end-to-end throughput of 49.26 FPS, compared with 5.72–18.51 FPS for the other sources. Thus, the camera estimator improves both motion accuracy and end-to-end throughput under the same video inference settings.

Hand prediction reuse and camera stitching. Table 5 compares inference configurations using the same trained models and stride-1 evaluation windows. Independent-window inference uses neither hand prediction reuse nor camera stitching. All other configurations reuse hand predictions at stride 16. Relative to independent-window inference, hand prediction reuse roughly doubles throughput. Adding camera stitching reduces Action MPJPE to 10.972 mm, potentially by averaging noisy overlapping pose estimates. Increasing camera stride to 8 maintains similar Action MPJPE (10.975 mm) while raising end-to-end throughput to 49.26 FPS, a $9.27\times$ speedup over independent-window inference. Similar Absolute and Action MPJPE under independent-window inference support the default pipeline's accuracy while reducing repeated computation (Appendix D.4). Device-provided extrinsics can replace camera estimation and stitching to further reduce computation. Appendix D.5 details end-to-end throughput measurement.

Table 5: Hand prediction reuse and camera stitching on the EgoAct4D dataset. End-to-end throughput is measured in FPS on one RTX PRO 6000 GPU; speedup is relative to independent-window inference.

Configuration	Hand stride	Camera stride	Camera stitching	Action MPJPE (mm)↓	End-to-end FPS↑	Speedup
Independent windows	1	1	No	11.027	5.31	$1.00\times$
Hand reuse without stitching	16	1	No	11.095	10.90	$2.05\times$
Hand reuse with stitching	16	1	Yes	**10.972**	10.84	$2.04\times$
Larger camera stride	16	4	Yes	10.977	32.60	$6.13\times$
Default pipeline	16	8	Yes	10.975	**49.26**	$\mathbf{9.27}\times$

5 Conclusion

We present EgoAct4D, a simpler pipeline for more accurate bimanual 4D hand motion estimation from monocular egocentric RGB videos. For continuous videos, the pipeline reuses hand predictions and stitches overlapping camera trajectories to avoid repeated computation. We introduce a sliding-window evaluation protocol that measures hand position and motion errors without GT alignment. Under this protocol, EgoAct4D achieves state-of-the-art results on the HOT3D benchmark. We also construct the EgoAct4D dataset and demonstrate full-pipeline data scaling beyond HOT3D's training scale, reaching approximately 1 cm motion accuracy with an end-to-end throughput of 49.3 FPS on its test set. This accuracy and efficiency can support reusable motion supervision for vision-language-action pretraining and human-to-robot motion retargeting. Future work will explore unifying hand and camera estimation in a single model and scaling training with larger, more diverse datasets to further improve accuracy and generalization.

AI USE STATEMENT

We used AI tools to search the literature, review related work, explore research ideas, plan experiments, and interpret results. AI also assisted with evaluation and analysis code development, manuscript drafting, text editing, and figure preparation. Generative image tools supported preliminary teaser design. We checked reported numerical results against experiment outputs and take responsibility for all AI-assisted content and the conclusions presented in this paper.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A specify the model architectures, loss functions, and training settings. Appendix B describes video inference, evaluation settings, metric definitions, and baseline processing. Dataset statistics, supplementary results, and timing procedures are provided in Appendices C and D. We will share anonymized code with reviewers and area chairs during the discussion period and publicly release the code and EgoAct4D dataset upon acceptance.

ETHICS STATEMENT

The EgoAct4D dataset was collected to support research on hand-motion estimation and learning from human demonstrations. All participants were willing to take part in data collection. We protected participants' privacy throughout data collection and took measures to prevent the video recordings from disclosing sensitive information about participants or other personnel at the data collection site.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Shangchen Han, Fan Zhang, Linguang Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, Jakob Julian Engel, and Tomas Hodan. HOT3D: Hand and object tracking in 3D from egocentric multi-view videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7061–7071, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Ryan Hoque, Peide Huang, David J. Yoon, Mouli Sivapurapu, and Jian Zhang. EgoDex: Learning dexterous manipulation from large-scale egocentric video. In *International Conference on Learning Representations*, 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal feed-forward metric 3D reconstruction. In *International Conference on 3D Vision (3DV)*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025.

Taein Kwon, Bugra Tekin, Jan Stühmer, Federica Bogo, and Marc Pollefeys. H2O: Two hands manipulating objects for first person interaction recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10138–10148, 2021.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. In *International Conference on Learning Representations*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Macrodata Labs. Turning egocentric video into robot actions. Blog post, 2026. URL https://macrodata.co/blog/turning-egocentric-video-into-3d-hand-actions. Accessed September 9, 2026.

Christen Millerdurai, Shaoxiang Wang, Yaxu Xie, Vladislav Golyanik, Didier Stricker, and Alain Pagani. EgoForce: Forearm-guided camera-space 3D hand pose from a monocular egocentric camera. In *ACM SIGGRAPH Conference Papers*, pp. 1–12, 2026. doi: 10.1145/3799902.3811047.

Takehiko Ohkawa, Kun He, Fadime Sener, Tomas Hodan, Luan Tran, and Cem Keskin. AssemblyHands: Towards egocentric activity understanding via 3D hand pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12999–13008, 2023.

Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3D with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9826–9836, 2024.

Rolandos Alexandros Potamias, Jinglei Zhang, Jiankang Deng, and Stefanos Zafeiriou. WiLoR: End-to-end 3D hand localization and reconstruction in-the-wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12242–12254, 2025.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12179–12188, 2021. doi: 10.1109/ICCV48922.2021.01196.

Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics*, 36(6):245:1–245:17, 2017.

Zhihao Sun, Tong Wu, Ruirui Tu, Daoguo Dong, and Zuxuan Wu. UniHand: A unified model for diverse controlled 4D hand motion modeling. *arXiv preprint arXiv:2602.21631*, 2026.

Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In *Advances in Neural Information Processing Systems*, volume 34, pp. 16558–16569, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5294–5306, 2025.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026a.

Ye Wang, Pei Lin, Xiong-Hui Chen, Haoqi Yuan, Zhixuan Liang, Yiyang Huang, Anzhe Chen, Zixing Lei, Jie Zhang, Tao Zhang, Haoyang Li, Tong Zhang, Chenxi Xiao, Ziyuan Jiao, and Qin Jin. Ego2Robot: Scalable robot data synthesis from egocentric human data. *arXiv preprint arXiv:2608.02580*, 2026b.

Yufei Ye, Yao Feng, Omid Taheri, Haiwen Feng, Shubham Tulsiani, and Michael J. Black. Predicting 4D hand trajectory from monocular videos. In *International Conference on 3D Vision (3DV)*, pp. 860–870, 2026.

Huajian Zeng, Chaohua Yao, Yuantai Zhang, Jiaqi Yang, Rolandos Alexandros Potamias, and Xingxing Zuo. StableHand: Quality-aware flow matching for world-space dual-hand motion estimation from egocentric video. *arXiv preprint arXiv:2605.18553*, 2026.

Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. MediaPipe Hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020. Presented at the CVPR Workshop on Computer Vision for Augmented and Virtual Reality.

Jinglei Zhang, Jiankang Deng, Chao Ma, and Rolandos Alexandros Potamias. HaWoR: World-space hand motion reconstruction from egocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1805–1815, 2025.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019. doi: 10.1109/CVPR.2019.00589.

A Implementation and Training

A.1 Architectures and outputs

Hand estimator. We use intermediate DPT features from DA3Metric-Large for hand estimation (Ranftl et al., 2021). Four pre-normalized Transformer layers (Vaswani et al., 2017) read projected features and update hand-state queries as in Section 3.2; Table 6 gives dimensions and optimization settings.

Joint positions are predicted independently in right-handed OpenCV camera coordinates (x right, y down, z forward), in meters. Training-set means and standard deviations normalize each hand, joint, and coordinate. For wrist orientation, the model predicts a correction to a reference six-dimensional rotation representation derived from the training set. We convert the resulting representation into a rotation matrix (Zhou et al., 2019). A sigmoid gives the in-frame probability.

Camera estimator. DA3-Large jointly processes 16 frames. Its MLP maps each frame's 2048-dimensional token to translation, a quaternion, and two field-of-view values; depth and ray heads are unused. Translations are denormalized with a fixed training-set scale. Predicted camera poses are expressed relative to the first frame of each input clip. No test-time GT scale fitting is used.

Table 6: Architecture and optimization settings. The two learning rates apply to encoder LoRA parameters and fully trained decoders and heads, respectively.

Setting	Hand estimator	Camera estimator
Encoder initialization	DA3Metric-Large	DA3-Large
Decoder / camera MLP	4 layers, 8 heads, width 256	2 layers, width 2048
Decoder FFN width / dropout	1024 / 0.1	–
LoRA rank / scaling	8 / 16	8 / 16
Encoder / head learning rate	10^{-4} / 10^{-3}	10^{-5} / 10^{-4}
Epochs / batch size	2 / 32	1 / 32

A.2 Training and losses

AdamW (Loshchilov & Hutter, 2019) trains the estimators separately, with gradient norm clipped at 1. LoRA (Hu et al., 2022) updates attention and feed-forward projections in all 24 encoder blocks. The hand projection, decoder, heads, and camera MLP are fully trained (Table 6). Normalizers are fitted on each training split or subset and fixed at evaluation. Table 7 reports the GPU configurations and wall-clock training times for the default estimators.

Table 7: Training hardware and wall-clock time for the default hand and camera estimators on HOT3D and EgoAct4D. EgoAct4D uses the full 55.3-hour training set.

Dataset	Estimator	GPU configuration	Time (h)
HOT3D	Hand	2× RTX PRO 6000	39.7
HOT3D	Camera	2× RTX PRO 6000	23.1
EgoAct4D	Hand	8× NVIDIA H200	51.3
EgoAct4D	Camera	8× NVIDIA H200	36.9

Hand losses. The hand objective is given in Eq. (3). In Eq. (3), $\mathcal{L}_{\text{wrist}}$ and $\mathcal{L}_{\text{finger}}$ supervise wrist and finger positions using Smooth L1 with transition 0.1 after training-set standardization. $\mathcal{L}_{\text{relative}}$ compares finger-to-wrist offsets using the same penalty after scaling by $0.1\,\text{m}$. $\mathcal{L}_{\text{rotation}}$ measures wrist-rotation geodesic error, and $\mathcal{L}_{\text{presence}}$ uses binary cross-entropy for in-frame presence. $\mathcal{L}_{\text{proj}}$ compares projected joints with valid in-image GT in normalized image coordinates, using Smooth L1 with transition 0.01. Its weight λ_{proj} ramps from zero to 0.2 over the first 10% of training, then stays fixed. Each term is averaged over its supervised elements. Hands in frame for at least two frames receive pose supervision for all valid states, including out-of-view states, independently of predicted presence.

Camera losses. For a $T = 16$-frame clip, let $\mathcal{E}_0$ contain first-frame pairs $(0, t)$ and $\mathcal{E}_1$ contain adjacent pairs $(t-1, t)$, for $t = 1, \ldots, T-1$. The camera objective is given in Eq. (4). Here, $\mathcal{L}_{\mathrm{FoV}}$ supervises the two predicted field-of-view values using targets derived from the training intrinsics. Let $(\mathbf{R}_{i,j}, \mathbf{d}_{i,j})$ map camera frame j to frame i. The pose term in Eq. (4) is

$$\mathcal{L}_{\mathrm{pose}}(\mathcal{E}) = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \left[\rho\left(\frac{\hat{\mathbf{d}}_{i,j} - \mathbf{d}_{i,j}}{s_{\mathrm{tr},\mathcal{E}}} \right) + \rho\left(\frac{d_{\mathrm{SO}(3)}(\hat{\mathbf{R}}_{i,j}, \mathbf{R}_{i,j})}{s_{\mathrm{rot},\mathcal{E}}} \right) \right]. \tag{6}$$

In Eq. (6), $\mathcal{E}$ is either pair set and $|\mathcal{E}|$ is its number of frame pairs. Hats denote predictions; unhatted rotations and translations are GT. $d_{\mathrm{SO}(3)}$ is the geodesic rotation angle, and ρ is Smooth L1 with transition 0.1, averaged over coordinates for vector inputs. The translation and rotation normalization scales $s_{\mathrm{tr},\mathcal{E}}$ and $s_{\mathrm{rot},\mathcal{E}}$ are fitted on the training set separately for each pair type and fixed at evaluation.

B Evaluation Protocol and Metrics

B.1 Video inference and scoring

For HOT3D, we follow the training and test split used by HaWoR (Zhang et al., 2025).

Camera stitching. We join adjacent camera clips using their predicted poses at shared frames. Each shared frame provides a candidate rigid transformation between the two clips. After rejecting inconsistent candidates using median-based translation and rotation residuals, we estimate translation by the component-wise median and rotation by averaging the retained rotation matrices and projecting the result onto a valid rotation matrix using SVD. We compose these transformations to express all clips in a common coordinate system while keeping metric scale fixed. For frames covered by multiple clips, we fuse the transformed poses using triangular weights that increase toward the clip center. Translations are averaged with these weights; rotations are combined by projecting their weighted matrix average onto a valid rotation matrix. The final input is shifted to cover the episode tail. We score all valid 16-frame windows at stride 1 within each episode, requiring consecutive source frames and valid annotations. Eqs. (5) and (1) express each output in its first-frame camera coordinates. Full and In-frame selection follow Section 4.1. Both settings use the same predictions. HOT3D has 84,048 Full and 57,140 In-frame windows (151,184 and 114,280 hand-windows). EgoAct4D has 310,587 Full windows from 192 episodes and 313,467 unique frames, totaling 621,174 hand-windows.

Baseline processing. HaWoR's released HOT3D evaluation uses GT-derived bounding boxes for hand reconstruction. Its mask-generation code renders the resulting predicted meshes, making hand-mask generation indirectly dependent on GT localization. The evaluation code also aligns each predicted SLAM segment to the corresponding GT camera trajectory before joining the segments, fitting rotation and translation while keeping scale fixed. These GT-assisted steps introduce additional test-time information and would make the inference conditions unequal across methods. We therefore use detector-predicted boxes for HaWoR and a shared predicted EgoAct4D camera trajectory for all methods, without GT camera alignment. Some baselines require detected hand boxes and cannot reconstruct hands when detections are missing due to occlusion or out-of-view motion. We fill gaps between detected boxes by linear interpolation and copy the nearest available box to frames before the first or after the last detection. The completed boxes are then used for hand reconstruction.

B.2 Metric definitions

Each output window contains $L = 16$ frames, with $J = 21$ joints per hand. Indices t and j identify the frame within the window and the joint, respectively. Eq. (1) maps joint positions $\mathbf{x}_{t,j}$ in frame t's camera coordinates to $\mathbf{y}_{t,j}$ in the window's first-frame camera coordinates, both in meters. Hats denote predictions and unhatted quantities denote GT; each uses its respective camera trajectory. Absolute MPJPE (e_{abs}) and Action MPJPE (e_{act}) are

$$e_{\mathrm{abs}} = \frac{1000}{LJ} \sum_{t=0}^{L-1} \sum_{j=0}^{J-1} \|\hat{\mathbf{x}}_{t,j} - \mathbf{x}_{t,j}\|_2, \tag{7}$$

$$e_{\mathrm{act}} = \frac{1000}{(L-1)J} \sum_{t=1}^{L-1} \sum_{j=0}^{J-1} \|\hat{\mathbf{y}}_{t,j} - \mathbf{y}_{t,j}\|_2. \tag{8}$$

Here, $\|\cdot\|_2$ is Euclidean distance, and the factor 1000 converts meters to millimeters. Eqs. (7) and (8) score all frames and future offsets 1–15, respectively, without wrist centering or GT alignment. Errors in mm are averaged equally over eligible hand-windows, including repeated states in overlapping windows.

Supplementary metrics. Absolute wrist rotation measures geodesic error in camera coordinates; Action rotation uses first-frame coordinates from each source's camera trajectory and scores offsets 1–15. Both are in degrees. PA-MPJPE first aligns the predicted 3D joints to the GT joints independently for each hand in each frame, fitting rotation, translation, and uniform scale. It then measures the mean Euclidean distance between corresponding joints in millimeters, assessing hand-pose accuracy after alignment. For W-MPJPE and WA-MPJPE (Zhang et al., 2025), we express each hand trajectory in the window's first-frame camera coordinates and fit a single similarity transform for the entire window. W-MPJPE estimates this transform from the first two frames, whereas WA-MPJPE uses all 16 frames. Both apply the fitted transform to all 16 frames and report the mean joint-position error in millimeters.

Acceleration error uses first-frame-coordinate trajectories and recorded timestamps. Let Δ_t be the time interval in seconds from frame $t-1$ to t. Joint acceleration $\mathbf{a}_{t,j}$ is estimated as

$$\mathbf{a}_{t,j} = \frac{2}{\Delta_t + \Delta_{t+1}} \left(\frac{\mathbf{y}_{t+1,j} - \mathbf{y}_{t,j}}{\Delta_{t+1}} - \frac{\mathbf{y}_{t,j} - \mathbf{y}_{t-1,j}}{\Delta_t} \right). \tag{9}$$

We apply Eq. (9) to prediction and GT and average their Euclidean difference over joints and 14 interior frames, then over eligible hand-windows, in $\mathrm{m/s^2}$. Reprojection error measures pixel distance using resized-image intrinsics and GT joints with positive depth inside the image, including occluded joints; predicted projection depth is clamped below at 10^{-3} m.

C EgoAct4D Dataset

C.1 Dataset organization, splits, and training subsets

Dataset organization. EgoAct4D organizes its recordings into 160 subtasks across nine activity categories, covering food preparation, packaging, textile work, crafts and assembly, cleaning, organization and storage, harvesting, personal care, and repair. Each subtask groups episodes of related activities. Figure 6 counts each label once per episode; episodes can have multiple labels.

Training and test splits. We use an episode-level split with 3,669 training episodes totaling 55.3 hours and 192 test episodes totaling 2.9 hours. Training covers all 160 subtasks, while the test set covers 129. Subtasks with too few episodes are retained entirely in training. Training and test episodes contain non-overlapping source-video frames. All experiments on EgoAct4D use the same test set.

Training subsets. We construct four nested training subsets containing 3.3, 8.5, 21.7, and 55.3 hours of video (Table 8), using sampling proportional to each subtask's available training data. All four subsets retain the 160 subtasks and are spaced logarithmically in size. The smallest matches HOT3D's training frame count approximately. Architecture, preprocessing, and training epochs remain fixed (Table 6), and normalization statistics are computed from each training subset. Figure 7 shows training clips of rinsing a board, threading beads, and folding a garment, with projected hand-joint annotations.

Table 8: Training subset statistics. Windows are counted at stride 1. HOT3D refers to the Project Aria training split used in our experiments.

Dataset	Video (h)	RGB frames	Windows
HOT3D	3.3	358,723	344,938
EgoAct4D	3.3	358,556	355,406
	8.5	914,094	907,029
	21.7	2,335,702	2,315,210
	55.3	5,961,837	5,906,766

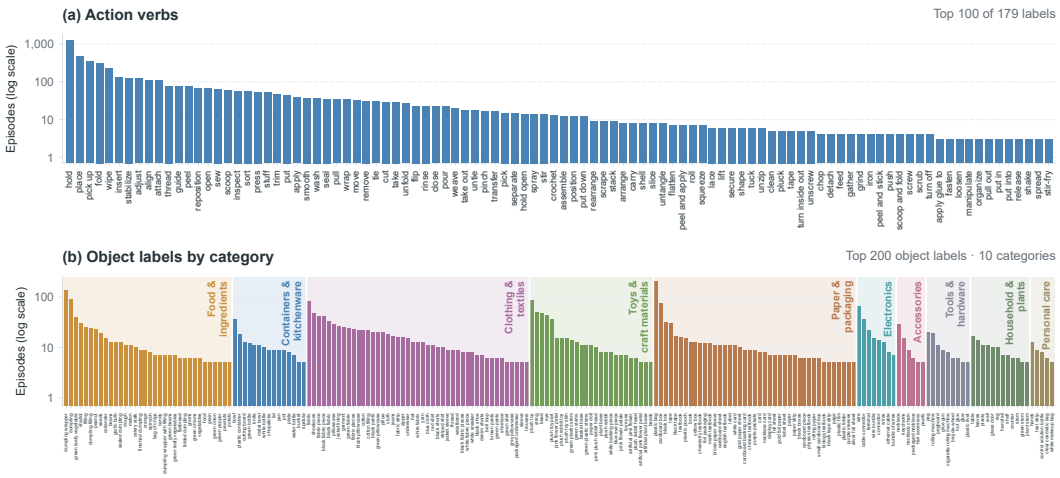


Figure 6: EgoAct4D label distributions. **(a)** Top 100 action verbs by episode count. **(b)** Top 200 object labels in ten categories, sorted by count within each category. Vertical axes are logarithmic; each label is counted once per episode.

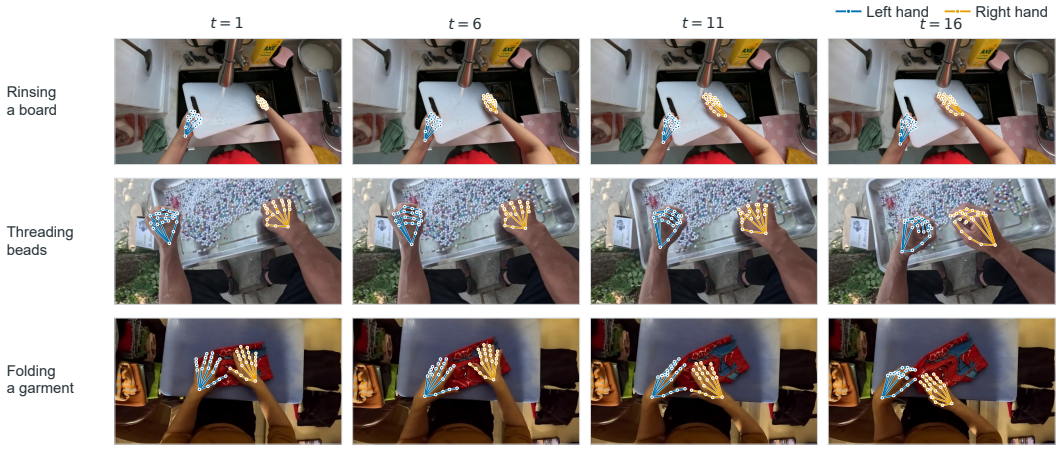


Figure 7: EgoAct4D dataset examples. Each row shows four frames from a 16-frame clip, with projected joint annotations for the left (blue) and right (orange) hands.

D Supplementary Results and Ablations

D.1 Additional data-scaling results

Table 9 reports the complete results for the four training scales under full-set video evaluation. All scales use the default video inference pipeline and the same test set. The GT-camera reference keeps hand predictions fixed and replaces the predicted camera trajectory with GT.

D.2 Additional HOT3D results

Tables 10 and 11 report additional HOT3D results using the same shared predicted camera trajectory as the main comparison.

D.3 Hand-estimator ablation details

This section details the hand-estimator variants compared in Figure 5. Each variant is trained on the same 8.5-hour subset for two epochs (56,688 updates) and evaluated on the same targets.

Table 9: EgoAct4D data scaling under full-set video evaluation. Units: position (mm), reprojection (pixels), acceleration ($\mathrm{m/s^2}$), rotation (degrees). * replaces only camera predictions with GT. Best and second-best results are **bold** and underlined, excluding the GT-camera reference.

Data (h)	Abs.	Act.	PA	W	WA	Reproj.	Accel.	Abs. rot.	Act. rot.
3.3	13.613	16.912	5.310	13.508	8.965	3.553	3.939	5.705	5.752
8.5	10.708	14.087	4.281	11.679	7.636	2.711	3.439	4.306	4.358
21.7	9.024	12.473	3.698	10.570	6.791	2.298	2.883	3.524	3.573
55.3	**7.951**	**10.975**	**3.367**	**9.494**	**6.134**	**2.027**	**2.703**	**3.095**	**3.140**
55.3*	7.951	7.951	3.367	7.032	5.082	2.027	2.662	3.095	3.095

Table 10: Motion estimation on the HOT3D benchmark under full-set evaluation. Lower is better; best results are **bold** and second-best underlined.

Method	Abs.	Act.	PA	W	WA	Reproj.	Accel.	Abs. rot.	Act. rot.
HaWoR	53.306	53.300	**6.829**	28.367	14.945	125.510	17.167	12.947	12.935
WiLoR	71.176	71.146	7.724	37.206	18.715	25.182	22.793	18.548	18.542
EgoForce	93.378	93.443	7.390	51.992	22.979	510.224	42.687	24.764	24.761
HaMeR	172.699	172.914	9.832	85.680	25.052	109.135	151.135	42.116	42.111
HaPTIC	83.846	83.795	10.337	50.126	23.137	19.039	16.350	39.949	39.942
MediaPipe	208.900	208.906	20.197	100.352	40.107	226.115	54.086	–	–
EgoAct4D (ours)	**19.396**	**19.606**	7.137	**18.771**	**11.287**	**6.208**	**7.427**	**7.416**	**7.405**

Table 11: Motion estimation on the HOT3D benchmark under in-frame evaluation. Lower is better; best results are **bold** and second-best underlined.

Method	Abs.	Act.	PA	W	WA	Reproj.	Accel.	Abs. rot.	Act. rot.
HaWoR	42.296	42.382	**6.526**	20.042	11.819	153.139	16.212	10.593	10.590
WiLoR	57.327	57.335	7.284	26.106	15.272	24.438	14.934	14.791	14.783
EgoForce	66.697	66.707	7.023	37.584	18.509	454.958	27.418	20.255	20.238
HaMeR	86.978	86.740	9.494	44.678	20.198	17.886	35.284	37.079	37.060
HaPTIC	70.365	70.373	9.917	35.511	19.300	17.800	11.101	35.580	35.566
MediaPipe	181.246	181.202	19.945	89.218	37.791	35.770	55.656	–	–
EgoAct4D (ours)	**15.558**	**15.773**	6.962	**13.494**	**9.417**	**5.909**	**5.830**	**6.288**	**6.290**

Frozen encoder. We freeze the DA3Metric-Large encoder and train only the projection, temporal decoder, and prediction heads. **ViT-L/16 encoder.** We replace DA3Metric-Large with an ImageNet-21k-pretrained ViT-L/16 and apply rank-eight LoRA to all 24 encoder blocks. The projection, temporal decoder, and prediction heads are fully trained. **Qwen3-VL-2B.** We use Qwen3-VL-2B-Instruct as a video encoder with rank-eight LoRA. We add 32 learned queries, one per hand per frame, and predict hand states through a projection and MLP heads. **Flow matching.** We retain the adapted DA3Metric-Large encoder and replace direct prediction with a four-layer Transformer flow-matching decoder, using four Euler steps at inference and random seed 42.

Table 12 reports all six evaluation metrics for these variants and our default estimator.

D.4 Additional video-inference results

Table 13 compares independent-window inference, hand prediction reuse without camera stitching, and the default pipeline. All configurations use models trained on the full 55.3-hour training set and are evaluated on the same 310,587 stride-1 windows from 192 test episodes. For independent-window inference, both estimators process each window separately, without hand prediction reuse or camera stitching. Absolute and Action MPJPE remain close across the three configurations. Independent-window inference has lower acceleration error (2.133 versus 2.703 $\mathrm{m/s^2}$) and Action wrist-rotation error (3.104° versus 3.140°) than the default pipeline, which achieves the higher throughput reported in Table 5.

Table 12: Hand-estimator variants trained on 8.5 hours of the EgoAct4D dataset and evaluated with the same camera trajectory. Lower is better; best results are **bold**.

	Position (mm)			Temporal ($\mathrm{m/s^2}$)	Wrist rotation (deg)	
Method	Abs.	Act.	PA	Accel.	Abs.	Act.
Frozen hand encoder	16.148	18.970	5.614	3.919	6.882	6.917
ViT-L/16 encoder	14.544	17.536	5.169	3.732	6.418	6.456
Qwen3-VL-2B	12.282	15.501	4.730	**2.800**	5.244	5.289
Flow matching	12.570	15.692	5.332	7.237	4.763	4.807
EgoAct4D (default)	**10.708**	**14.087**	**4.281**	3.439	**4.306**	**4.358**

Table 13: Independent-window and video inference on the EgoAct4D dataset using the same trained models and full-set evaluation windows. Lower is better; best results are **bold** and second-best underlined.

	Position (mm)			Temporal ($\mathrm{m/s^2}$)	Wrist rotation (deg)	
Configuration	Abs.	Act.	PA	Accel.	Abs.	Act.
Independent windows	<u>7.956</u>	<u>11.027</u>	**3.367**	**2.133**	<u>3.097</u>	**3.104**
Hand reuse without stitching	**7.951**	11.095	**3.367**	2.708	**3.095**	3.150
Default pipeline	**7.951**	**10.975**	**3.367**	<u>2.703</u>	**3.095**	<u>3.140</u>

D.5 End-to-end throughput measurement

We measure end-to-end throughput on four 512-frame segments using one RTX PRO 6000 GPU, four CPU threads, batch size one, and 16-frame inputs. After 64 warm-up frames, we report the median throughput in FPS over three repeats. Throughput is computed by dividing the 2,048 input video frames by the total processing time for all four segments. Timing includes warm-cache data reads, preprocessing, transfers, fresh inference, scale recovery, stitching, and output assembly. DA3Metric-Large runs for every camera window requiring scale recovery. For independent-window inference, both estimators run at stride 1, producing fresh predictions for each window without hand prediction reuse or camera stitching.