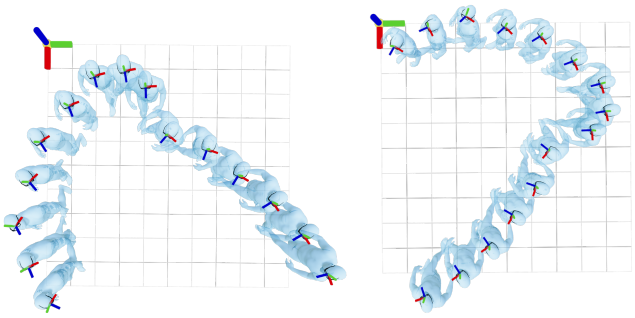
Estimating Body and Hand Motion in an Ego-sensed World

Supplementary Material

A.1. Invariant Conditioning Visualization

As we observe in Table 1, naively training a model using absolute head poses results in poor estimation performance. The absence of spatial invariance (Invariance 1) explains this result. To visualize this, we show in Figure A.1 two renders of the same human motion trajectory. The second render has the same local body motion as the first, but with the world frame re-defined:

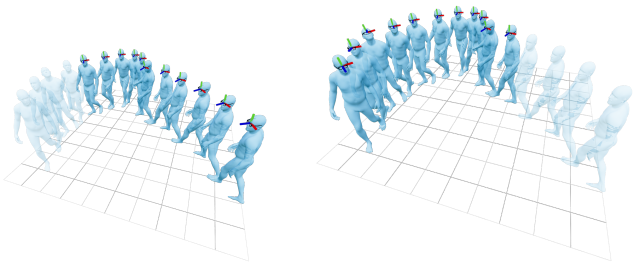


(a) Before (b) After

Figure A.1. **Absolute head pose visualization for a single human motion trajectory, before and after re-defining the world frame.**

Because the world frame location is arbitrarily defined, naively conditioning on these poses hinders generalization. Works like EgoPoser [29] have made similar observations.

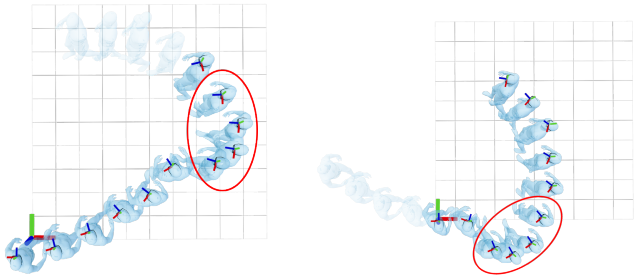
To fix this, prior works have preprocessed sequences by aligning them to a canonical coordinate frame located at the first timestep of each sequence [21, 48, 74]. However, we observe that this is flawed from the perspective of temporal invariance (invariance 2). To visualize this, we render in Figure A.2 two temporal slices of the same body motion, with one slice starting from the beginning of the motion and another starting from the middle:



(a) First slice (b) Second slice

Figure A.2. **Two slices of the same human motion trajectory.**

Next, we consider how the head pose trajectories for each of these slices would look if they were canonicalized by aligning the first timestep. We visualize the resulting head pose trajectories in Figure A.3. Circled in red are four timesteps that are shared between the two slices. Notice that head poses from canonicalized sequences can still differ significantly, even for the same body motion.



(a) First slice (b) Second slice

Figure A.3. **Poses canonicalized by aligning the first timestep.**

To achieve both Invariances 1 and 2, EgoAllo's invariant conditioning parameterization proposes an alternative way to canonicalize head poses. Instead of defining a single canonical coordinate frame for each temporal window, we define a canonical coordinate frame at *every* timestep. The resulting representation couples relative CPF motion $\Delta \mathbf{T}^t_{\text{cpf}}$ with per-timestep canonicalized pose $\mathbf{T}^t_{\text{canonical, cpf}}$. These transformations are visualized in Figure A.4. Notice that the transformations that make up this conditioning approach are invariant both to the world coordinate system and to choices in temporal windowing. This enables significant improvements in estimation accuracy (Table 1).

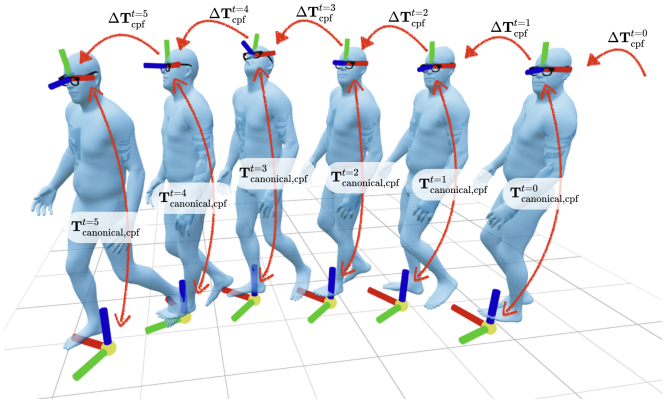


Figure A.4. **Transformations that make up the invariant conditioning used by EgoAllo.**

A.2. Ancillary Results

A.2.1. Sequence length evaluation

At test time, EgoAllo follows MultiDiffusion [3] for extrapolating to arbitary sequence lengths. To validate this choice, we filter out test sequences shorter than 256 frames and then evaluate both EgoAllo and EgoEgo [48] with subsequences of length 32, 128, and 256. We report MPJPE metrics on these sequences in Table A.1. Both EgoAllo and

Seqlen	32	128	256
EgoAllo	149.3	130.3	127.9
EgoEgo	187.7	173.8	184.3

Table A.1. **Effect of sequence length on MPJPE (mm).** EgoAllo is trained with sequences of length 128. EgoEgo [48] is trained with sequences of length 140.

No Hands	EgoAllo-Mono	EgoAllo-Reproj	EgoAllo-Wrist3D
119.7	91.1	78.8	63.1

Table A.2. **Body MPJPEs with hand guidance.** We quantify how hand guidance impacts body MPJPE (mm).

EgoAllo	EgoPoser	EgoPoser w/o Normalization
119.7	127.2	127.4

Table A.3. **MPJPE vs EgoPoser.** We compare against EgoPoser both with and without temporal normalization.

EgoEgo include windowing strategies for handling longer sequences; unlike prior work, however, we find that accuracy improves even after test set sequence lengths surpass the training set sequence length.

A.2.2. Body improvements from hand guidance

Incorporating hand observations into human motion estimation improves overall body MPJPE metrics. We quantify this in Table A.2. For fair evaluation, this experiment uses synthetic hand pose observations on the AMASS test set.

A.2.3. EgoPoser ablation

In Table A.3, we report MPJPE for a variant of EgoAllo trained using the conditioning formulation from EgoPoser. This includes temporally normalized head poses and global frame velocities. We also report MPJPE without EgoPoser's normalization; this is equivalent to *Absolute+Global Deltas* in Table 1. We found that the impact of EgoPoser's normalization is small after adapting to our problem setting. We hypothesize that this is because their temporal normalization (subtraction using the first timestep's position) is a simple linear function, which makes it easier to learn end-to-end.

A.2.4. Additional qualitative results

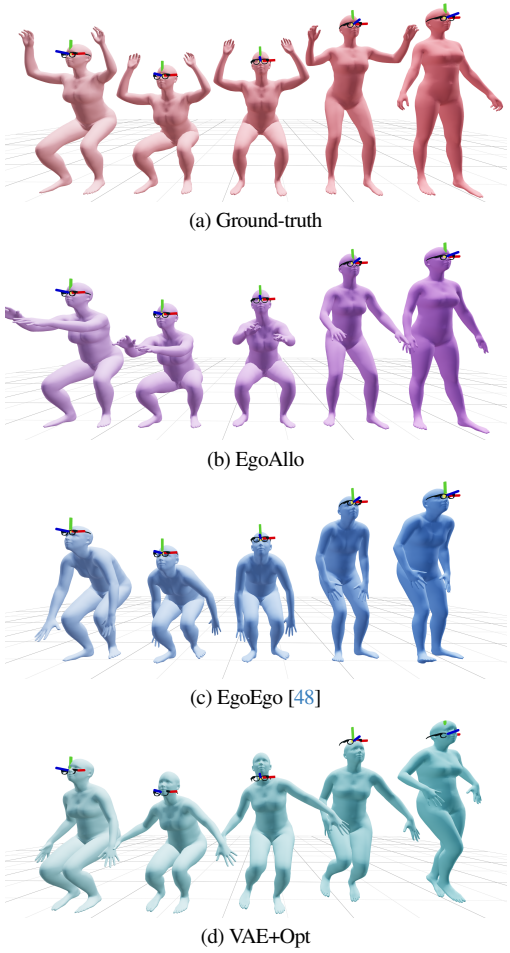
We provide additional qualitative results for the body motion prior in Figure A.5. EgoAllo estimates have the head aligned exactly to input observations and the feet planted realistically on the floor.

A.3. Implementation Details

A.3.1. Training and testing

EgoAllo models are trained and tested using the splits recommended by the official AMASS GitHub repository. We use ADAM and learning rate 1e-4 for 3M steps. Training subsequence lengths in [32,128] are sampled uniformly.

Details on AMASS annotation. To annotate AMASS with central pupil frames (CPFs), we average pupil vertex positions (SMPL-H indices 6260, 6262, 2800, and 2802). CPF orientation is copied from the head joint.



(a) Ground-truth

(b) EgoAllo

(c) EgoEgo [48]

(d) VAE+Opt

Figure A.5. **Head pose-conditioned motion prior results for a squatting sequence.** Spatial shifts are used to visualize different timesteps within the sequence. Hand observations are not used.

Test sequence lengths We evaluate lengths 32 and 128 to compare performance on both our min and max train sequence lengths. For length 32 test sequences, we filter out all AMASS test sequences shorter than 32. We then randomly slice length-32 subsequences from the remaining trajectories. For length 128 test sequences, we filter out all AMASS test sequences shorter than 128. We then randomly slice length-128 subsequences from the remaining trajectories.

A.3.2. Network architecture

EgoAllo uses a transformer [94] architecture with rotary positional embeddings [88] for its denoising model $\mu_\theta(\vec{x}_n, \vec{c}, n)$. Sampling is performed by denoising all timesteps within a temporal window in parallel: we do not sample autoregressively and therefore do not use causal masking. *Encoder details:* latent encodings $\vec{z}_c$ are computed as output from conditioning sequences $\vec{c}$ as input using six transformer blocks, each containing a self-attention layer followed by a 2-layer MLP. *Decoder details:* the denoised output is computed using six additional transformer blocks that take $\vec{x}_n$ as input, while conditioning

on $\vec{z}_c$ via cross-attention. All hidden dimensions are set to 512.

Body shapes. Per-timestep shapes are used for architectural simplicity. At test time, we use the average body shape across timesteps. In practice, we find that it is easy for the model to learn temporal consistency.

Runtime. For a length-128 sequence, each forward pass through EgoAllo's denoising network takes 0.05 seconds on a single RTX 4090. Because we use DDIM [86] for sampling, the number of denoising steps for each sample can be chosen to make tradeoffs between sample quality and speed. All experiments in our paper use 30 DDIM steps.

A.3.3. Guidance optimizer

For guidance, we use a Levenberg-Marquardt optimizer implemented in JAX [5]. Levenberg-Marquardt is an iterative nonlinear least squares algorithm, which requires solving a linearized subproblem at each timestep. We compute the Jacobians needed for this as block-sparse matrices for efficiency, and solve the resulting linear subproblems using a Conjugate Gradient optimizer.

Runtime. The guidance optimizer converges in 0.15∼0.2 seconds on an RTX 4090. We compare our LM optimizer against off-the-shelf PyTorch optimizers in Figure A.6.

A.3.4. Guidance Loss Details

The guidance objective $\mathcal{L}_{\text{guidance}}(\Theta)$ combines three major components as described in the main paper: $\mathcal{L}_{\text{hands}}$, $\mathcal{L}_{\text{skate}}$, and $\mathcal{L}_{\text{prior}}$. All rotation-based error terms are implemented using geodesic distance in the tangent space of SO(3). All weight terms can be found in our open-source code release.

Hand-related Terms:

- *Hand Pose HaMeR Local Alignment*: Aligns local hand joint rotations with HaMeR detections.
- *Hand Reprojection*: Aligns projected hand joints with 2D detections in image space.
- *Wrist Pose Alignment*: Matches wrist positions and orientations with 3D observations.

Contact-related Terms:

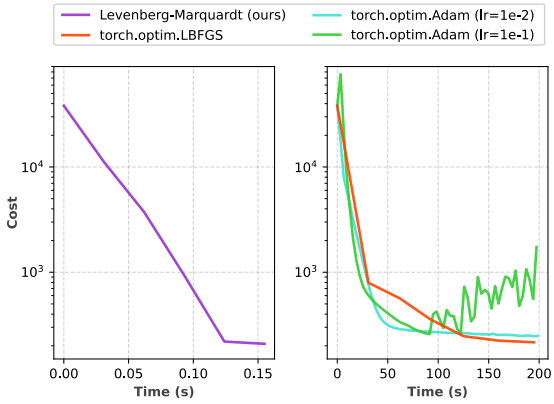
- *Foot Skating Prevention*: Penalizes movement of foot joints predicted to be in contact with the ground, using contact-weighted position differences.

Prior-related Terms:

- *Body Pose Prior*: Regularizes body pose rotations toward denoiser output.
- *Hand Pose Prior*: Regularizes hand poses toward denoiser output.
- *Position Prior*: Regularizes joint positions toward denoiser output.
- *Delta Smoothness*: Encourages smooth deviation from denoiser output.
- *Pose Smoothness*: Encourages smooth body pose transitions.
- *Hand Temporal Smoothness*: Enforces smooth hand motion.
- *Velocity Smoothness*: Penalizes sudden changes in rotational velocity.

A.3.5. Floor height estimation

One requirements of EgoAllo is SLAM poses that can be situated relative to the floor. While floor heights are provided in our training data, they are not directly available on real-world data. We found that a RANSAC-based algorithm works well on real-world data from Project Aria [61]. We filter SLAM points by confidence, then use RANSAC to find a z-value with that best fits a plane. Example floor plane outputs using scenes from the EgoExo4D [17] dataset are shown in Figure A.7.



(a) **Costs over time.** LM converges significantly faster than off-the-shelf PyTorch optimizers for guidance optimization.

Optimizer	Final Cost
Levenberg-Marquardt (ours)	209.05
torch.optim.LBFGS	215.96
torch.optim.Adam (lr=1e-2)	248.01
torch.optim.Adam (lr=1e-1)	1733.30

(b) **Final costs.** We report the final cost for each method in the plot above.

Figure A.6. **Comparing guidance optimizers.**

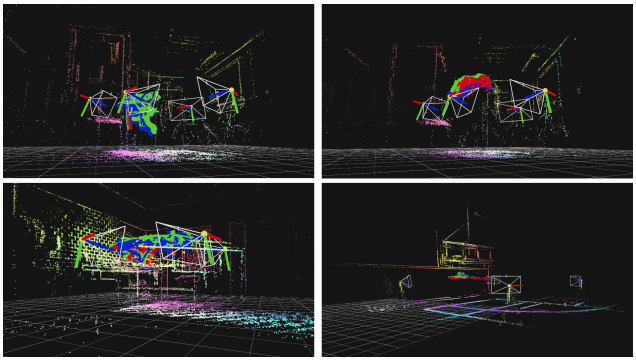


Figure A.7. **Floor height examples.** Point cloud-derived floor height examples on the EgoExo4D dataset.

A.3.6. Biomech57 evaluation details

The majority of our evaluation data (AMASS [56] and RICH [26]) is provided directly using SMPL conventions. Because EgoAllo outputs SMPL-H parameters, this makes computation of joint error metrics straightforward.

The one exception is the Aria Digital Twins dataset [61], which we use for quantitative body metrics. Each device wearer in the Aria Digital Twins dataset is recorded via an Optitrack motion capture system, which records 57 joint locations (30 hand joints, 27 body joints) following the Biomech57 joint template. To evaluate our method on ADT, we match and compare the common major joints between the two templates. We manually corresponded each of the 57 joints between Biomech57 and the standard SMPL-H joint conventions.

While the majority of these have 1:1 correspondences—feet, knees, hips, shoulders, elbows, wrist, and finger joints, for example, are consistently defined—we mask out others like the head and collar bone joints that are misaligned.