EgoGen: An Egocentric Synthetic Data Generator

Supplementary Material

S1. Related Work

EgoGen addresses the gap in egocentric synthetic data generation specifically tailored for head-mounted devices, situated at the intersection of three key areas: 1) General synthetic data generation; 2) Egocentric simulation for embodied agents; 3) Human-related synthetic data generation. For a more detailed understanding of the distinctions between *EgoGen* and existing methods, refer to Tab. S1, where these three areas are clearly outlined within distinct blocks.

In particular, VirtualHome [16, 17] also provides rendered egocentric views from head-mounted cameras. However, their egocentric videos lack fluctuating patterns due to the absence of natural human motion; instead, they display robotic-like patterns as Habitat 2.0 [23]. We advance closer to synthesizing more realistic data for head-mounted devices. In addition, they lack a generative human motion model, whereas ours can generate more diverse human motion and trajectories. A very recent work Habitat 3.0 [18] introduced virtual humans to robotic simulation. However, their human locomotion is synthesized by cyclically replaying a walking motion clip from MoCap data along a precalculated path with rigid rotations to transition to the next walking direction. Both VirtualHome and Habitat 3.0 have a limited number of human agents and fall short in representing diverse human characters, with limitations in body shapes, ethnic variation, and clothing options. UnrealEgo [1] is a large-scale naturalistic dataset for egocentric 3D human motion capture, which employs downward-facing cameras and replies on mocap data. It is used to estimate the human body pose of the camera wearer. In contrast, we employ front-facing cameras and perform generative human motion synthesis for autonomous virtual humans.

Our synthetic data generation achieves increased diversity by incorporating a walking path-free generative human motion model, diverse body shapes, various body textures, and varied 3D textured clothing.

S2. Ego-Sensing Driven Motion Synthesis

S2.1. Egocentric Sensing Calculation

As a compact and cheap-to-compute representation of depth maps, egocentric sensing resembles the calculation of depth information but is simplified into 2D.

As shown in Fig. S4, the location of the egocentric camera is the midpoint of two eyeballs, and the viewing direction $\vec{\mathbf{v}}$ is visualized as the red arrow. N rays are cast from the location of the egocentric camera, with the central direction of these rays determined by the 2D projection of $\vec{\mathbf{v}}$. The starting points of these rays are identical, while their endpoints form a semicircle in front of the virtual human, representing the field of view $[\theta_{min}, \theta_{max}]$ to the human. Each ray has the potential to extend infinitely. In our implementation, $N = 32$, $\theta_{min} = -90°$, $\theta_{max} = 90°$.

The 2D collision detection of rays leverages the 2D layout of the 3D scene. For illustration purposes, we simplify the obstacles in 3D scenes with grey rectangles and visualize the collision detection of rays in Fig. S1. The egocentric sensing encodes the simplified depth of obstacles.

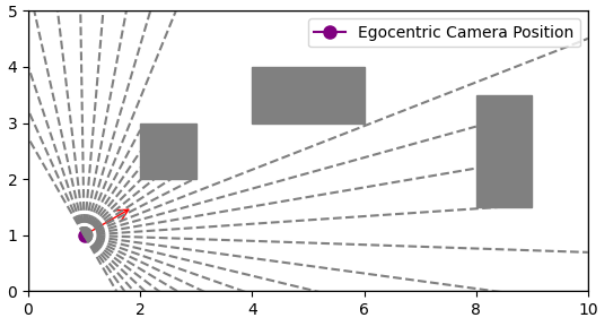


Figure S1. The 2D projection of the egocentric camera location is represented by the purple point, while the 2D projection of the viewing direction $\vec{\mathbf{v}}$ is indicated by the red arrow. The field of view changes due to the head pose.

S2.2. Reward, Weighting, and Training Detail

In our motion primitive environment, we design an intuitive set of rewards to encourage the agent to perform realistic human motions (all vectors are normalized in the following equations):

Attention reward that encourages the human to look at the goal:

$$r_{attention} = \frac{\langle \vec{\mathbf{v}}, g - h \rangle + 1}{2}, \tag{1}$$

where $\vec{\mathbf{v}}$ denotes the viewing direction, g and h denote the goal location and current head location and $\langle \cdot \rangle$ denotes the inner product.

Foot contact reward $r_{cont.}$ contains two components: Foot floor distance reward and foot skating reward.

$$r_{cont.} = r_{floor} + r_{skate} \tag{2}$$

$$r_{floor} = e^{-(|\min_{x \in F} x_z| - 0.02)_+}, \tag{3}$$

$$r_{skate} = e^{-(\min_{x \in F} \|x_{vel}\|_2 - 0.075)_+}, \tag{4}$$

Table S1. Comparison of existing synthetic datasets or generators. (Sec. S1)

	Domain	Egocentric	Head-mounted	Multi-Camera rigs	Virtual Humans	Automated Clothing Simulation	Generative Human Motion Synthesis
Kubric [10]	Scattered Objects	✗	✗	✗	✓	✗	✗
PointOdyssey [31]	Point Tracking	✓	✓	✗	✓	✗	✗
InfiniGen [19]	Natural	✗	✗	✗	✗	✗	✗
RoboGen [25]	Robotics	✗	✗	✗	✗	✗	✗
UniSim [27]	Real-world Interaction	✓	✓	✗	✗	✗	✗
UnityPerception [8, 24]	Object Detection Pose Estimation	✗	✗	✗	✓	rigged clothing	✗
uHumans2 [20]	Scene Graphs	✓	✗	✗	✓	✗	✗
Carla [7]	Driving	✓	✗	✗	✓	rigged clothing	✗
VirtualHome [16, 17]	Household Simulation	✓	✓	✗	✓	rigged clothing	✗
VRKitchen [9]	Cooking Simulation	✓	✓	✗	✗	✗	✗
Habitat 2.0 [23]	Embodied Robots	✓	✗	✗	✗	✗	✗
Habitat 3.0 [18]	Human-robot Interaction	✓	✗	✗	✓	✗	✗
UnrealEgo [1]	Ego-pose Estimation	✓	✓	✓	✓	rigged clothing	✗
GTA-Human [4]	Pose Estimation	✗	✗	✗	✓	✗	✗
BEDLAM [3]	Pose Estimation	✗	✗	✗	✓	✗	✗
SynBody [28]	Pose Estimation	✗	✗	✗	✓	✗	✗
ADT [14]	Digital Twin	✓	✓	✓	✗	✗	✗
EgoGen (ours)	Head-mounted Devices	✓	✓	✓	✓	✓	✓

where F denotes foot markers, x_z denotes the marker height, x_{vel} denotes the marker velocity, and $(\cdot)_+$ denotes clipping negative values. There are tolerance thresholds of 0.02m for foot-floor distance and 0.075m/s for skating.

Goal Distance reward that encourages the agent to get closer to the goal at each step:

$$r_{dist} = d^{t-1} - d^t, \tag{5}$$

Here d^t denotes the body-goal distance at step t.

Body orientation reward that encourages the body forward direction to be aligned with the goal location direction:

$$r_{ori} = \frac{\langle o_b, g - p \rangle + 1}{2}, \tag{6}$$

where o_b denotes the body forward orientation, g and p denote the goal location and current pelvis location, and $\langle \cdot \rangle$ denotes inner product. Different from the *attention reward* that drives the head motion, this penalizes backward movement toward the goal.

Penetration reward that penalizes the intersection of the human body and obstacles. We use different penetration rewards in different settings.

When training in sparse scenes, e.g., a single static box obstacle, penetration detection is simplified into 2D to accelerate calculation:

$$r_{pene}^{sparse} = \begin{cases} 0.05, & |\mathcal{M}_0 \cap \mathcal{B}_{xy}(X)| < thres \\ 0, & otherwise \end{cases} \tag{7}$$

where $\mathcal{M}_0$ denotes the non-walkable cells on the ground plane, $\mathcal{B}_{xy}(.)$ denotes the 2D bounding box of the body markers X, $\cap$ denotes their intersection, and $|\cdot|$ denotes the number of non-walkable cells within the bounding box of the human. $thres$ is set to 3 and the cell dimension is $0.1m \times 0.1m$.

When training in crowded scenes, we use the signed distance field (Ψ_O) for precise penetration detection:

$$r_{pene}^{crowded} = e^{-\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{|V|}|(\Psi_O(\mathbf{v}_{ti}))_-|} \tag{8}$$

where $|V|$ denotes the number of SMPL-X mesh vertices, T denotes the number of frames in our motion primitive ($T = 20$), $\mathbf{v}$ denotes SMPL-X mesh vertex, and $(\cdot)_-$ denotes clipping positive values. The penetration reward penalizes body vertices with negative SDF values within a motion primitive.

Pose reward that penalizes generating unrealistic human poses using VPoser [15] body pose prior:

$$r_{pose} = \begin{cases} 0.05, & \|VP\|_2 < thres \\ 0, & otherwise \end{cases} \tag{9}$$

where $\|\text{VP}\|_2$ denotes the pose embedding inferred by the VPoser encoder $\mu(\cdot)$, where $\|\text{VP}\|_2 = |\mu(\theta)|_2$. θ denotes the SMPL-X body pose parameter representation. The VPoser pose prior learns a probabilistic pose distribution where vectors closing to zero have a high probability and correspond to realistic human poses. In our observation, $\|\text{VP}\|_2 > 15$ produces unrealistic human poses. $thres$ is set to 11.

Success reward for reaching the goal location:

$$r_{succ} = \begin{cases} 1, & d < thres \\ 0, & otherwise \end{cases} \tag{10}$$

where d is the body-goal distance. $thres$ is set to 0.1.

The weights for each reward are listed in Tab. S2. The weighting of each reward is determined according to the reward value. For example, the goal distance reward measures the distance change in one motion primitive spanning 0.5s, which is approximately 10 times smaller than other rewards. As a result, its weight is 10 times bigger than others. We observe high foot skating weight helps to reduce foot skating. Higher success rewards encourage the agent to reach the goal. But on the other hand, the weight can not be too big. Because we did not do reward normalization, too large values may lead to big errors in value estimation and training instabilities.

Reward	Weight
Foot floor distance	0.1
Foot skating	0.3
Goal distance	1
Body orientation	0.1
Attention	0.3
Penetration pretraining	1
Penetration finetuning	0.1
Pose	0.1
Success	0.5

Table S2. Reward weights.

Penetration Termination. We terminate an episode due to penetration using different criteria. In sparse scenes, an episode is terminated if $r_{pene}^{sparse} = 0$.

As mentioned in Sec. 3.2 in the main paper in crowded scenes, we employ a two-stage RL training scheme. In stage I, we pretrain the policy with a penetration weight of $w_{r_{pene}} = 1$ to more effectively encourage the virtual human to avoid obstacles and explicitly **not** perform penetration termination. After convergence, in stage II, we proceed to fine-tune the policy with a strict penetration termination using a reduced penetration weight of $w_{r_{pene}} = 0.1$. Penetration detection involves considering the maximum number of body vertices in penetration within a motion primitive. An episode is terminated if:

$$\max_t \sum_{i=1}^{|V|} |(\Psi_O(\mathbf{v}_{ti}))_-| \geq thres \tag{11}$$

where $thres$ is set to 40.

This design has several reasons: 1) Our action space is an unbounded Gaussian, direct training with strict termination can lead the policy to explore unreasonable spaces and produce unrealistic human poses, see Fig. S8 for illustration. 2) Reducing penetration weight during fine-tuning can amplify the significance of the goal-reaching weight, encouraging goal-reaching behaviors.

S2.3. PPO

Our PPO implementation is based on Tianshou [26]. We list the hyperparameters of PPO in Tab. S3. c_1, c_2 are defined in Sec. 3.2 in the main paper. "Repeat per Collect" is the training iterations with the same collected rollouts.

Param	Value
Learning Rate	3e-4
γ Discount	0.99
PPO Clip Threshold	0.1
Repeat per Collect	1
Value Function Coefficient (c_1)	1
Entropy Coefficient (c_2)	0.01
GAE (λ)	0.95
Max Grad. Norm	0.1

Table S3. PPO hyperparameters.

The majority of the hyperparameters were set to their default values. To note, adopting smaller values of "PPO Clip Threshold" and "Repeat per Collect" will not update parameters too drastically and thus stabilize the training. We use advantage normalization without value function clipping.

Another trick we adopted is that we performed the last policy layer weight scaling, which makes initial actions close to the standard normal distribution, which can boost the performance [2].

The training time is roughly 20 hours on a GeForce RTX 3090 GPU with batch size 256, 20000 steps per epoch.

The key difference between our environment with others is that our action space is not strictly bounded. The motion primitive model $\mathcal{P}$ is based on VAE and is pretrained with a KLD loss w.r.t. a standard normal distribution. As a result, we do not do any action scaling or clipping during training. Due to the nature of our action space, the learned policy can deviate too much from the standard normal distribution. As a result, we select the best model using the best test reward and minimum KL divergence between the learned policy action space and the standard normal distribution.

S2.4. Qualitative Failure Cases

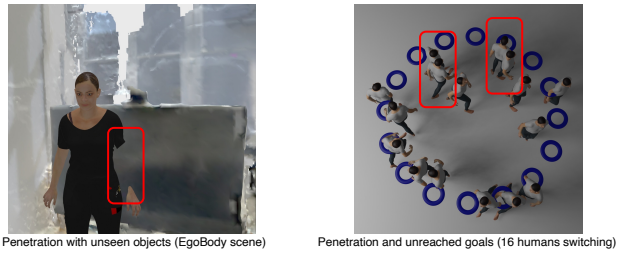


Figure S2. Failure cases.

When evaluation significantly differs from training, failure cases may occur, including penetrations and unreached goals. See Fig. S2.

S3. Egocentric Synthetic Data Generation

S3.1. Embodied Camera Placement

We support various camera placements in *EgoGen*.

For egocentric sensing-driven motion synthesis (Sec. 3 in the main paper), we place one camera at the midpoint of two eyeballs and the viewing direction $\vec{\mathbf{v}}$ is shown in Fig. S4. We use the SMPL-X [15] armature in Blender [5] to calculate $\vec{\mathbf{v}}$. The two eye bones are visualized in Fig. S3.

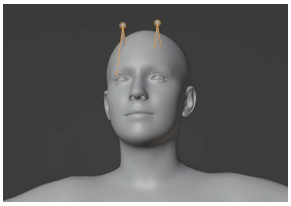


Figure S3. Eye bones are located at the eyes and are highlighted with orange edges.

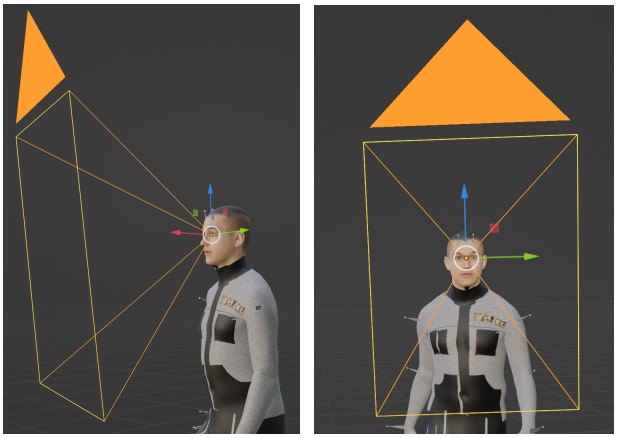


Figure S4. Illustration of embodied camera placement. The camera axes are determined by 1) The blue arrow of the eye bone (from *root* to *tip*); 2) The green arrow from one eyeball to another; 3) The red arrow representing the viewing direction $\vec{\mathbf{v}}$. (Sec. S2.1, Sec. S3.1)

EgoGen also supports multi-camera rigs simulation. With the information about the relative poses of cameras within a rig, we have the flexibility to position the camera at various locations on the head.

To further enhance realism, one may consider various face shapes and physics simulations to place egocentric cameras. We agree that simulating headset placement and shifting can enhance realism. However, first, achieving faithful simulation would involve modeling deformable human facial skin and muscles, which is an ongoing research problem. Second, even with perfect modeling of the camera extrinsic shifting, real-world devices can still have instrumental errors that may offset fine-grained simulation. As an alternative, we considered that people can wear the headset with slightly different placements, and introduced noise augmentation to simulate this (See Sec. S5.2: Egocentric camera tracking). It might be effective to use aleatoric uncertainty to model headset shifting and instrumental errors together in a Bayesian way.

S3.2. Automated Clothing Simulation

As shown in Tab. S1, many prior works resort to generating synthetic data with rigged clothing, with unrealistic clothing deformations. In contrast, BEDLAM [3] and Synbody [28] incorporate physics-based clothing simulation to enhance realism and allow for dressing a diverse range of body shapes in a wide array of clothing. However, their approaches are not scalable for handling arbitrary motion sequences produced by our generative human motion model.

We further automate clothing dynamics simulation with the state-of-the-art clothing simulation network HOOD [11]. HOOD treats each garment as a single graph and predicts graph deformations due to both gravity and collisions with the human body mesh.

First, we perform preprocessing on the 3D clothing mesh from [3] to separate the upper garment and lower garment into distinct clothing meshes because HOOD can not handle disconnected graphs as input. Second, we sample pose blend shapes, shape blend shapes, and average skinning weights from the closest n SMPL-X mesh vertices in A-Pose, where $n = 1$ for tight garments such as pants and $n = 1000$ for loose garments such as dresses. We repose the clothing meshes in A-Pose to match the body pose in the first frame of a synthesized motion sequence. Then, for lower garments, the vertices in the top ring are fixed to the body to prevent dropping due to gravity. Finally, we simulate the upper and lower garments separately using HOOD.

S3.3. More Examples of Available Annotations

In addition to the fisheye cameras shown in the teaser, here we show more ground-truth annotations with perspective cameras, including RGBD, optical flow, bounding boxes, segmentation masks, and surface normals in Fig. S5.

S4. Experiments

S4.1. Test Scenarios in Evaluation of CAMPs

We provide visualizations of how we built test scenarios. Please refer to Sup. Vid. for qualitative results.

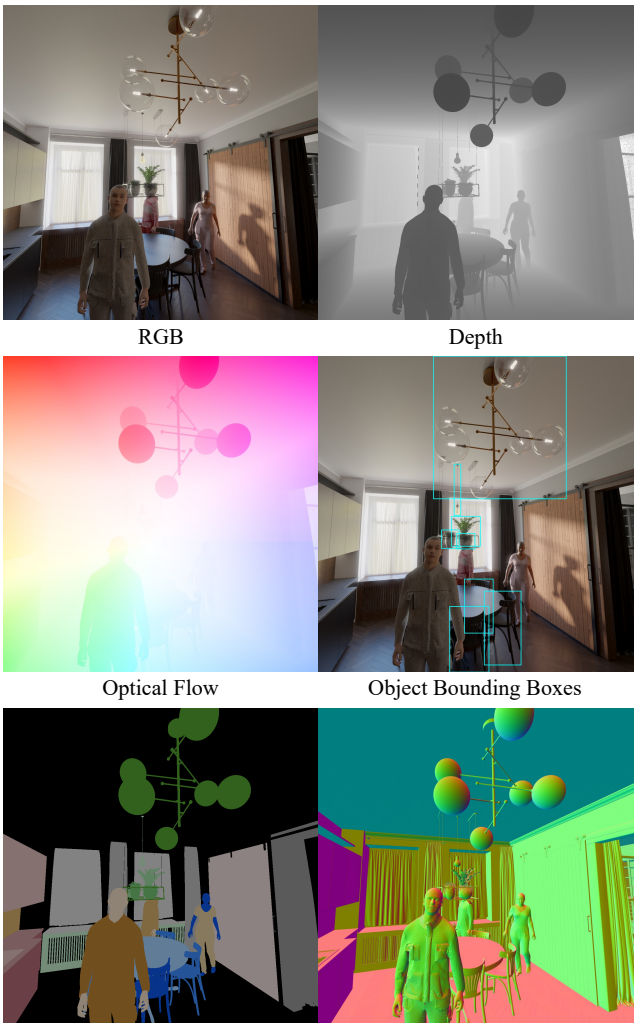


Figure S5. Ground-truth annotations from perspective cameras.

Moving obstacle. Refer to Fig. S6 for an illustration of the evaluation in scenes with moving obstacle. The moving obstacle will move between the human and its goal location.

Multiple humans. To visually demonstrate, as depicted in Fig. S7, we initiate four virtual humans from distinct points in the figure. We require they walk to the opposite location across the origin, either from A to B or from B to A. There are no other obstacles. Please refer to Sup. Vid. for qualitative results.

Path diversity. We assess walking path diversity with a static obstacle fixed at the midpoint between the start and target locations. Similar to Fig. S6, but the obstacle is not moving anymore.

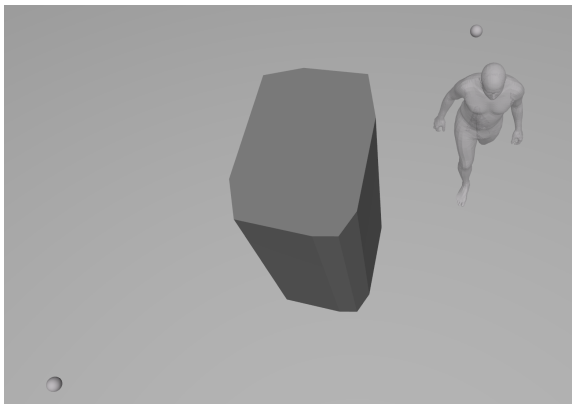


Figure S6. Moving obstacle test scenario illustration.

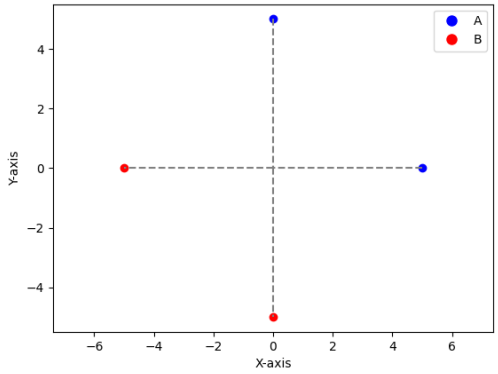


Figure S7. Multiple humans crowd motion test scenario illustration.

S4.2. Evaluation Metrics

The Foot contact metric [30] reaches 1 when there is foot-floor contact and no foot skating, defined as:

$$s_{contact} = e^{-(|\min x_z| - 0.05)_+} \cdot e^{-(\min \|x_{vel}\|_2 - 0.075)_+}, \tag{12}$$

where x_z and x_{vel} denote the marker height and velocity, 0.05 and 0.075 are tolerance thresholds, and $(\cdot)_+$ denotes clipping with the lowerbound of 0.

For moving obstacle scenes, we evaluate human-scene penetration by detecting all frames where the floor plane projections of any body parts and obstacles have intersections. For multiple human scenes, we measure the accurate human-human penetration using the implicit human body occupancy model COAP [12], which predicts the body occupancy given spatial location queries. Since the articulated human bodies are complex and require accurate penetration detection, we detect whether one human collides with other humans by querying its body vertices using the occupancy field of all other humans at that frame and report the occu-

pancy values for human-human penetration.

S4.3. Ablation Studies

No pretraining. Without our two-stage RL training scheme, direct penetration termination in crowded scenes will result in unrealistic predicted human poses. As shown in Tab. 3 in the main paper, where $\|\text{VP}\| = 28.77 > 15$, we provide visualizations of the corresponding unnatural poses in Fig. S8. In contrast, our full model works well. Please refer to Sup. Vid.

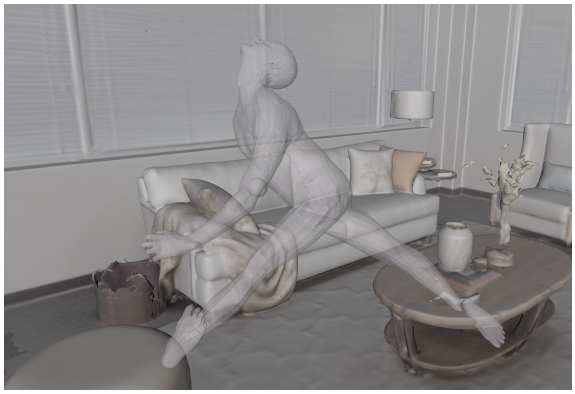


Figure S8. Ablation of our two-stage RL training. Without pretraining, the model can produce unrealistic human poses.

S5. Egocentric Perception Tasks

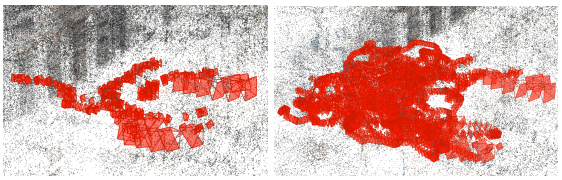
S5.1. Mapping and Localization for AR

As shown in Fig. S9, we can leverage *EgoGen* to explore the large-scale scene, add synthetic egocentric images into the dataset, and build a more complete Structure-from-Motion (SfM) map (Fig. S9b). In our implementation, we randomly set the starting and target locations of virtual humans. Compared with [13] that perturbs real-world cameras with random noise (Fig. S9c) that may result in unrealistic camera poses, *EgoGen* can simulate human trajectories and motion (Fig. S9d).

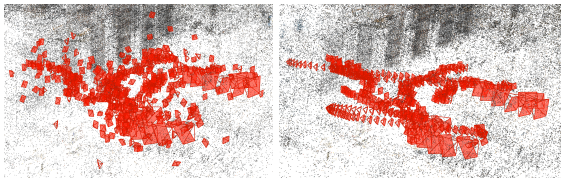
The efficacy of synthetic data for a task relies on the domain gap between synthetic and real-world images. In the SfM pipeline in our experiment, the render-to-real gap can influence the result of the feature extraction. As shown in Fig. S10 on the left, detected feature points with SuperPoint [6] are much noisier in synthetic images due to scene quality, which can make feature matching challenging. In addition, the feature matcher SuperGlue [21] exhibits overfitting behaviors: visually similar images are preferred to be matched first, i.e., it tends to match sim-sim and real-real pairs only. As a result, simply adding synthetic images into the real-world dataset will result in no matches between synthetic and real-world images, making it impossible to improve localization recall.

To ensure valid matching between synthetic and real-world mapping images, during the pair selection process using SuperGlue, we force synthetic images to match with real images only. By implementing this approach, we can achieve a denser SfM map by establishing matches between synthetic and real-world 2D image feature points (see Fig. S10) and thereby triangulating more 3D points.

To enhance the localization performance of real-world query images using the augmented SfM map, we enforce matches with both synthetic and real mapping images for all query images. This ensures that real-world query images can be paired with synthetic mapping images, leveraging a denser SfM map and enhancing localization recall.



(a) Real-world cameras (b) EgoGen synthetic cameras



(c) Perturbing existing cameras (d) EgoGen same # of cams as (c)

Figure S9. EgoGen addresses the issue of sparsity by populating the dataset with synthetic images. In Fig. S9a, the sparsity of real-world mapping images is apparent, where each red object represents a camera and each colored dot represents a triangulated 3D point. After applying EgoGen, mapping images are more densely distributed, resulting in denser 3D triangulated points, as shown in Fig. S9b. In Fig. S9c and Fig. S9d, we augment S9a with the same amount of synthetic images using [13] and EgoGen respectively. EgoGen generates synthetic data with a similar distribution as human trajectories as illustrated in S9d. Results are visualized using Colmap [22]. Note that we only visualize a subset of cameras here.

S5.2. Egocentric Camera Tracking

The egocentric camera tracking task is evaluated using the head rotation error, translation error, and pose error that jointly accounts for both rotation and translation. The head rotation error calculates the Frobenius norm of the difference between the matrix representations of the predicted rotation R_{pred} and the ground truth rotation R_{gt}, which is defined as:

$$e_{rotation} = \|R_{pred} R_{gt}^{-1}\|_2, \tag{13}$$

The head translation error is computed as the mean Euclidean distance of two sequences of head translations. The

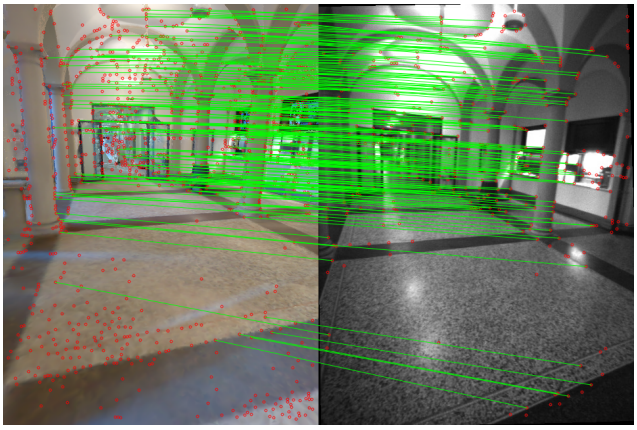


Figure S10. Feature matching visualization for a render-to-real image pair.

results are reported in the unit of millimeter.

The head pose error calculates the Frobenius norm of the difference between the transformation matrix of the predicted head pose and ground truth head pose, which is given by:

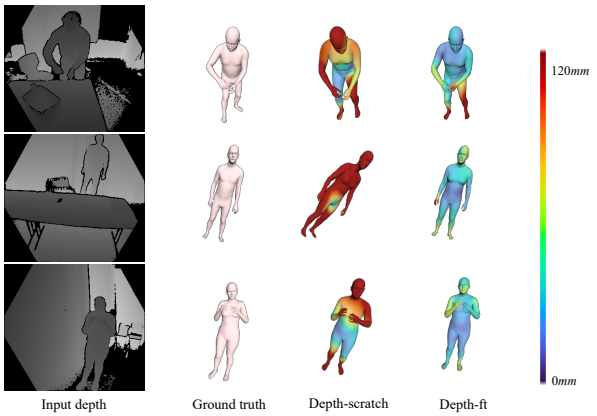
$$e_{pose} = \|T_{pred} T_{gt}^{-1}\|_2, \tag{14}$$

S5.3. Human Mesh Recovery from Egocentric Views

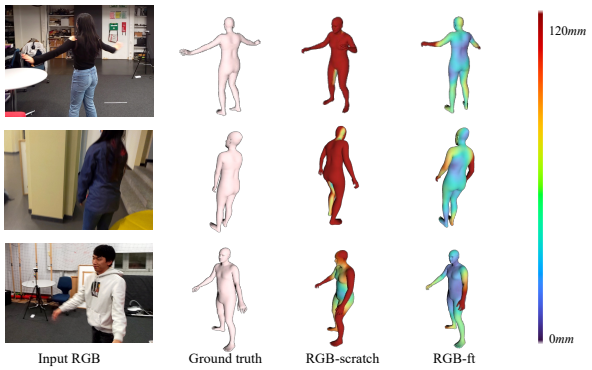
We simulate the data collection process of Egobody [29] and let two virtual humans walk in the scanned scene meshes from Egobody. We randomly sample *gender*, *body shape*, and *initial body pose* and synthesize human motions with our proposed generative human motion model to increase data diversity.

The egocentric camera is attached to both humans and we render the interactee from the camera wearer's egocentric view. Camera intrinsic is set similarly to the real-world camera. For depth data generation, we omit the clothing because the simulated depth sensor noise will remove detail. For RGB data generation, to further increase data diversity and close the sim-real gap, we randomly sample body texture and 3D textured clothing meshes from BEDLAM [3] and perform automated clothing simulation (Sec. S3.2) given arbitrary synthesized human motion sequences from our generative human motion model. In addition, we adopt random lighting in the rendering. In total, we synthesized 105k depth images and 300k RGB images with diverse body shapes, poses, skin textures, and clothing, along with ground-truth SMPL-X annotations. We will release both of our synthetic datasets as a complement to Egobody.

Qualitative results. We visualize our qualitative results for HMR from depth in Fig. S11a and HMR from RGB in Fig. S11b on real-world test data. With large-scale synthetic data from *EgoGen*, we can compensate for the lack of real-world data and improve the performance of current models. "*-scratch" denotes models trained only with limited real-world data. "*-ft" denotes models pretrained with our large-scale synthetic data and then finetuned with real-world data.



(a) Human Mesh Recovery from Depth Images

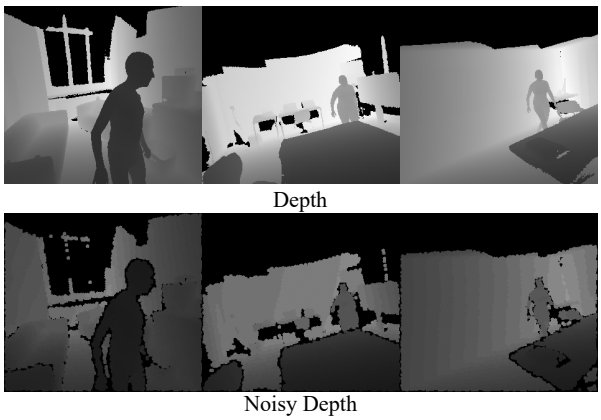


(b) Human Mesh Recovery from RGB Images

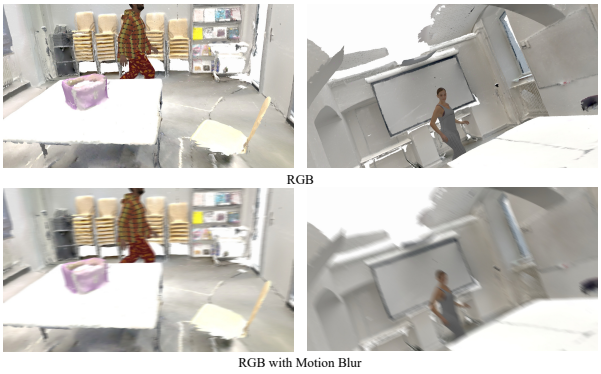
Figure S11. Qualitative results of HMR on EgoBody test set. The body mesh color of the last two columns denotes the per-vertex error between the predicted body and the ground truth.

Synthetic Data Samples. We show some examples of synthetic data from *EgoGen* in Fig. S12.

Synthetic Dataset Statistics. The generated depth dataset consists of 105000 depth images with 47107 male and 57893 female images. The generated RGB dataset consists of 301073 depth images with 147862 male and 153211 female images. Both datasets cover a large range of indoor interaction distances ranging from $0.60m$ to $5.02m$. Fig. S13a shows the distribution of the interaction distance of the depth dataset and Fig. S13b shows the distribution of the RGB dataset.



(a) Our synthetic depth images.

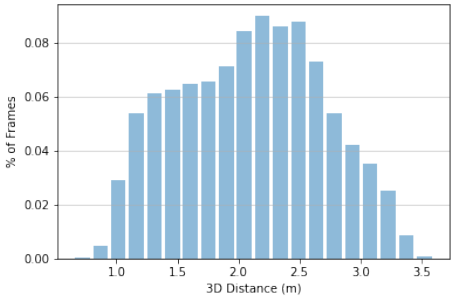


(b) Our synthetic RGB images.

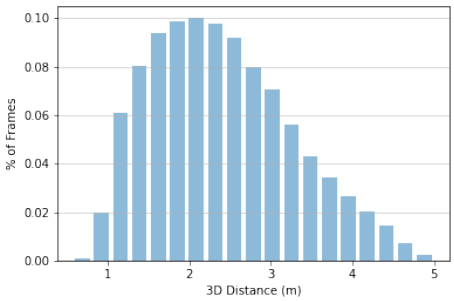
Figure S12. Synthetic data samples from *EgoGen*.

Additionally, we consider two types of "invisibility" of the joints: frame-wise invisibility and joint-wise invisibility ratio. The frame-wise invisibility ratio calculates the percentage of joints that are not on the image plane among all body joints. The joint-wise invisibility ratio calculates the ratio of frames when the joint is out of the image plane among all frames.

An analysis of the invisibility distribution of the depth dataset and the RGB depth distribution can be seen in Fig. S14. Due to the different camera intrinsic of depth and RGB sensor, the invisibility distribution is different. From Fig. S14a, we can see that over 79% of the depth frame contains more than 90% joints. This means most depth images contain the full body. While from Fig. S14c we can see that the RGB dataset yields higher invisibility. The detailed invisibility of the joints is shown in Fig. S14d. It illustrates that even though RGB images have a higher invisibility, the most frequently missing joints are the upper or lower part of people (eyes and toes). In more than 85% of the images, the pelvis joint can be found.



(a) 3d Distance on depth images.



(b) 3d Distance on RGB images.

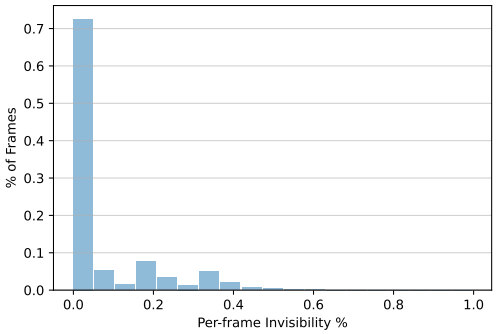
Figure S13. Interaction Distance of synthetic samples from *EgoGen*.

Table S4. Training scheme comparison of HMR. "-scratch": training with limited real-world data only. "-ft": transfer learning. "-Mixed training": training with mixed real and synthetic data together.

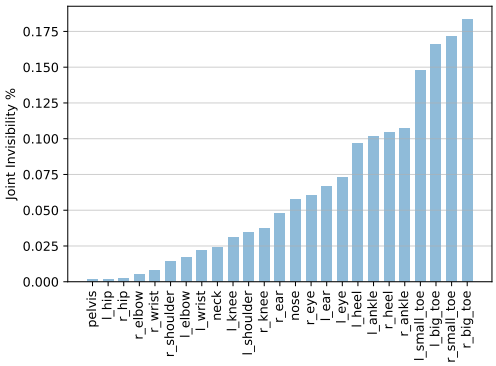
(All units are *mm*)	G-MPJPE ↓	MPJPE ↓	PA-MPJPE ↓	V2V ↓
Depth-scratch	117.7	82.2	54.1	100.6
Depth-ft	**90.7**	**65.2**	**47.3**	**81.0**
Depth-Mixed training	99.8	72.2	51.5	90.7
RGB-scratch	-	90.7	59.9	102.1
RGB-ft	-	**85.3**	**56.2**	**97.2**
RGB-Mixed training	-	85.5	57.3	98.2

Training Details. We use data augmentation on the training dataset besides adding the motion blur. These methods include using different kinds of image compression, brightness and contrast modification, noise addition, gamma, hue and saturation modification, conversion to grayscale, and downscaling techniques. During training, we set the batch size to 64 for the training on the depth dataset and 128 for the training on the RGB dataset. We use the AdamW Optimizer in the training process.

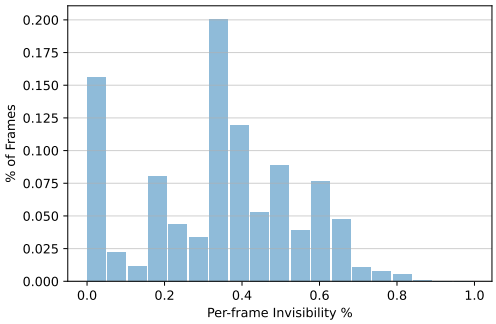
Impact of Training Schemes. We adopted transfer learning to improve generalization by transferring knowledge from pretraining on synthetic data and refining features through real-world data finetuning. Mixed training, as shown in Tab. S4, is less effective than "-ft", but still better than training with limited real data only.



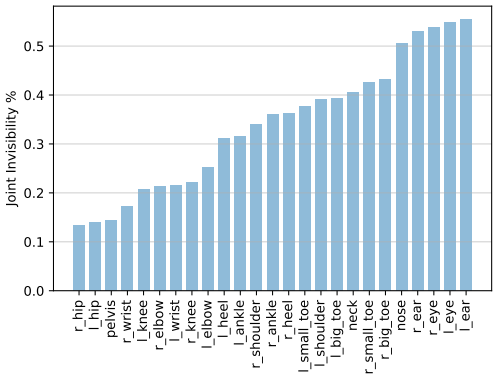
(a) Frame-wise invisibility on depth images.



(b) Joint-wise invisibility on depth images.



(c) Frame-wise invisibility on RGB images.



(d) Joint-wise invisibility on RGB images.

Figure S14. Invisibility Statistics.

References

[1] Hiroyasu Akada, Jian Wang, Soshi Shimada, Masaki Takahashi, Christian Theobalt, and Vladislav Golyanik. Unrealego: A new dataset for robust egocentric 3d human motion capture. In *European Conference on Computer Vision (ECCV)*, 2022. 1, 2

[2] Marcin Andrychowicz, Anton Raichuk, Piotr Stanczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? A large-scale empirical study. *CoRR*, abs/2006.05990, 2020. 3

[3] Michael J. Black, Priyanka Patel, Joachim Tesch, and Jinlong Yang. BEDLAM: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 4, 7

[4] Zhongang Cai, Mingyuan Zhang, Jiawei Ren, Chen Wei, Daxuan Ren, Zhengyu Lin, Haiyu Zhao, Lei Yang, and Ziwei Liu. Playing for 3d human recovery. *arXiv preprint arXiv:2110.07588*, 2021. 2

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 4

[6] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description, 2018. 6

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017. 2

[8] Salehe Erfanian Ebadi, Saurav Dhakad, Sanjay Vishwakarma, Chunpu Wang, You-Cyuan Jhang, Maciek Chociej, Adam Crespi, Alex Thaman, and Sujoy Ganguly. Psp-hdri+: A synthetic dataset generator for pre-training of human-centric computer vision models. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*, 2022. 2

[9] Xiaofeng Gao, Ran Gong, Tianmin Shu, Xu Xie, Shu Wang, and Song-Chun Zhu. Vrkitchen: an interactive 3d virtual environment for task-oriented learning. *arXiv*, abs/1903.05757, 2019. 2

[10] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam H. Laradji, Hsueh-Ti Derek Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, A. Cengiz Öztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 3739–3751. IEEE, 2022. 2

[11] Artur Grigorev, Michael J. Black, and Otmar Hilliges. HOOD: hierarchical graphs for generalized modelling of clothing dynamics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 16965–16974. IEEE, 2023. 4

[12] Marko Mihajlovic, Shunsuke Saito, Aayush Bansal, Michael Zollhoefer, and Siyu Tang. COAP: Compositional articulated occupancy of people. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 5

[13] Tony Ng, Adrian Lopez-Rodriguez, Vassileios Balntas, and Krystian Mikolajczyk. Reassessing the limitations of cnn methods for camera pose regression, 2021. 6

[14] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar M. Parkhi, Richard A. Newcombe, and Carl Yuheng Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception. *CoRR*, abs/2306.06362, 2023. 2

[15] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019. 2, 4

[16] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018. 1, 2

[17] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B. Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-{ai} collaboration. In *International Conference on Learning Representations*, 2021. 1, 2

[18] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, Vladimir Vondrus, Théophile Gervet, Vincent-Pierre Berges, John M. Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots. *CoRR*, abs/2310.13724, 2023. 1, 2

[19] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12630–12641, 2023. 2

[20] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone. 3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Robotics: Science and Systems (RSS)*, 2020. 2

[21] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks, 2020. 6

[22] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 6

[23] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2

[24] Unity Technologies. Unity Perception package. https://github.com/Unity-Technologies/com.unity.perception, 2020. 2

[25] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation, 2023. 2

[26] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267): 1–6, 2022. 3

[27] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023. 2

[28] Zhitao Yang, Zhongang Cai, Haiyi Mei, Shuai Liu, Zhaoxi Chen, Weiye Xiao, Yukun Wei, Zhongfei Qing, Chen Wei, Bo Dai, Wayne Wu, Chen Qian, Dahua Lin, Ziwei Liu, and Lei Yang. Synbody: Synthetic dataset with layered human models for 3d human perception and modeling, 2023. 2, 4

[29] Siwei Zhang, Qianli Ma, Yan Zhang, Zhiyin Qian, Taein Kwon, Marc Pollefeys, Federica Bogo, and Siyu Tang. Egobody: Human body shape and motion of interacting people from head-mounted devices. In *European conference on computer vision (ECCV)*, 2022. 7

[30] Yan Zhang and Siyu Tang. The wanderings of odysseus in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20481–20491, 2022. 5

[31] Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *ICCV*, 2023. 2