Action Motifs: Self-Supervised Hierarchical Representation of Human Body Movements
Supplementary Material

Genki Kinoshita[1] Shu Nakamura[1] Ryo Kawahara[1] Shohei Nobuhara[2]
Yasutomo Kawanishi[3] Ko Nishino[1]
[1]Kyoto University [2]Kyoto Institute of Technology [3]RIKEN
https://vision.ist.i.kyoto-u.ac.jp/

A. Implementation Details

A.1. Pose Representation

For the input to A4Mer, we use the same pose representation as HumanML3D [7], where body shapes are normalized and poses are expressed in a local coordinate system aligned with the gravity direction. A pose is defined as $(\boldsymbol{c}, \dot{r}^a, \dot{\boldsymbol{r}}^v, \boldsymbol{j}, \boldsymbol{j}^v, \boldsymbol{j}^r)$, where $\boldsymbol{c} \in \{0,1\}^4$ denotes the binary indicators of the ground contacts for left and right heels and toes; $\dot{r}^a \in \mathbb{R}$ is the pelvis angular velocity along the gravity direction; $\dot{\boldsymbol{r}}^v \in \mathbb{R}^2$ is the pelvis velocity on the horizontal plane; $\boldsymbol{j} \in \mathbb{R}^{J\times 3}$, $\boldsymbol{j}^v \in \mathbb{R}^{J\times 3}$, and $\boldsymbol{j}^r \in \mathbb{R}^{J\times 6}$ represent joint positions, velocities, and rotations, respectively, where J is the number of joints. The joint rotations $\boldsymbol{j}^r$ are parameterized using the 6D continuous rotation representation [21]. For comparison with existing methods in our experiments, we use their original pose representations exactly as specified in the respective papers.

A.2. Action Atom Segmentation

To determine the boundaries between Action Atom segments based on nonlinear changes in joint trajectories, A4Mer starts by linearly predicting local joint positions $\boldsymbol{j}_t \in \mathbb{R}^{J\times 3}$ at frame t. With least-squares fitting to the past w frames $\{\boldsymbol{j}_{t-w}, \ldots, \boldsymbol{j}_{t-1}\}$, a linear prediction of the joint trajectory at $\boldsymbol{j}_t$ is computed. The prediction error is computed as the L_1 difference $\mathcal{E}_{j,t} \in \mathbb{R}^3$ for each joint j. To eliminate joint-specific variation in the motion range, each joint's error is normalized with the mean μ_j and the standard deviation σ_j computed over all frames within the same video. The normalized errors are averaged over joints and coordinates to obtain a scalar error signal e_t for each frame. To capture movement transitions from this error signal, the first-order temporal difference $\Delta e_t = e_t - e_{t-1}$ and the second-order difference $\Delta^2 e_t = \Delta e_t - \Delta e_{t-1}$ are computed. While Δe_t is effective at capturing large movement transitions, it may fail to respond when the transition is sharp yet low in magnitude. The second difference $\Delta^2 e_t$ compensates for this by responding strongly to abrupt accelerations in the error signal. Based on these quantities, candidate boundary frames are detected using the logical condition

$$\left(\Delta e_t > \tau_1 \ \wedge \ \Delta^2 e_t > 0\right) \vee \left(\Delta^2 e_t > \tau_2\right), \qquad \text{(A)}$$

where $\tau_1 = 0.005$ and $\tau_2 = \mathrm{mean}_t(|\Delta^2 e_t|)$. The first term selects frames where the error is not only increasing beyond a threshold but also accelerating, indicating the onset of a new movement. The second term captures abrupt events, even when their first-order differences remain small. To prevent excessive fragmentation of segments, once a boundary is identified, we suppress further boundary detection for the subsequent 0.5 seconds, under the assumption that human movements typically preserve coherence for at least this duration. Note that this boundary-detection procedure is applied to the 30 fps streams, in contrast to the 5 fps used as input to the representation learning models.

A.3. Action Motif Segmentation

The Generalized Sequential Pattern (GSP) algorithm [13] is used to find frequent patterns shared across sequences of Action Atoms discretized into categorical codes by k-means. GSP is grounded in the Apriori principle, *i.e.*, any supersequence of an infrequent subsequence must also be infrequent, and discovers frequent patterns by iteratively expanding the current set of frequent patterns into longer pattern candidates. At each iteration, candidates are generated by concatenating pairs of frequent patterns obtained from all previous iterations. Candidates whose occurrence counts exceed a threshold o are then registered as new frequent patterns. This iterative generate-and-prune procedure enables GSP to efficiently extract recurring Action Atom patterns, *i.e.*, Action Motifs.

For an efficient matrix-based implementation, we set the maximum pattern length $p_{\max}\,(= 20)$. The minimum occurrence threshold is linearly decreased from $o_{\max}\,(= 15)$ to

Table A. Architectural configuration of A4Mer and task-specific heads.

Module	Submodule	Layers	Token Dim.	Attn. Heads
A4Mer	Encoder	4	256	4
	LatentFormer	4	256	4
	predictor g_ϕ	3	256	4
Recognition Head	–	1	256	4
Decoding Head	–	4	256	4
Motion Prediction Head	–	4	64	4
Motion Interpolation Head	–	4	256	4

$o_{\min}$ $(= 5)$ as the pattern length increases. To mitigate the risk of missing patterns due to intervening trivial or minor movements, two candidate patterns are determined as being equivalent when their Hamming distance does not exceed h, and their occurrences are summed when counting. The Hamming-distance threshold h is linearly increased from $h_{\min}$ $(= 0)$ to $h_{\max}$ $(= 4)$ as the pattern length grows. Furthermore, contiguous runs of identical categorical codes are treated as continuations of the same action, and such runs—regardless of whether their length exceeds $p_{\max}$—are also considered valid patterns.

A.4. Network Architectures

The task-specific heads and A4Mer including the predictor g_ϕ used for JEPA [1] are built upon Transformer [17] architectures. All self-attention layers in these networks employ the rotary positional embeddings (RoPE) [15]. The architectural hyperparameters of each network are summarized in Tab. A.

A4Mer. Encoder consists of a stack of Transformer-decoder layers, each composed of segment-wise self-attention and cross-attention from latent tokens z_k to their corresponding input segments. Here, we detail these two attention mechanisms. We represent the attention operator as

$$\mathrm{Attn}(\mathrm{Q},\mathrm{K},\mathrm{V}) = \mathrm{softmax}\left(\frac{(\mathrm{Q}W_{\mathrm{Q}})(\mathrm{K}W_{\mathrm{K}})^{\top}}{\sqrt{d}}\right)(\mathrm{V}W_{\mathrm{V}})\,, \tag{B}$$

where $W_{\mathrm{Q}}, W_{\mathrm{K}}, W_{\mathrm{V}}$ denote learnable projection matrices for queries Q, keys K, and values V, respectively, and d denotes the dimensionality of the projected queries and keys.

For convenience, we restate the notation from the main paper: the input sequence $X = (x_1, \ldots, x_T)$ is partitioned into segments $X_k = \{x_t \mid s(t) = k\}$, where $s(t)$ returns the segment index that frame t belongs to. Each layer first applies self-attention within each segment $k \in \{1, \ldots, K\}$:

$$\mathrm{SelfAttn}_k = \mathrm{Attn}(X_k, X_k, X_k)\,, \tag{C}$$

which restricts the receptive field to intra-segment interactions, helping the model consolidate local movement information without responding to coincidental similarities across distant frames.

Each variable-length segment X_k is consolidated into a single latent token through cross-attention:

$$\mathrm{CrossAttn}_k = \mathrm{Attn}(z_k, X_k, X_k)\,. \tag{D}$$

LatentFormer operates on the set of latent tokens $Z = (z_1, \ldots, z_K)$ and performs standard self-attention $\mathrm{Attn}(Z, Z, Z)$ to capture temporal relationships between movements at the abstracted semantic level.

Predictor in JEPA. The predictor g_ϕ takes a latent-token sequence in which the positions indexed by $\mathcal{K} \subset \{1, \ldots, K\}$ are masked out. For each $k \in \mathcal{K}$, a learnable mask token $\mathcal{M}$ (shared across $\mathcal{K}$) is inserted. Formally, we construct $\tilde{Z} = (\tilde{z}_1, \ldots, \tilde{z}_K)$, where

$$\tilde{z}_k = \begin{cases} \mathcal{M}, & k \in \mathcal{K}, \\ z_k, & \text{otherwise}\,. \end{cases} \tag{E}$$

The resulting sequence $\tilde{Z}$ is then processed using self-attention, and the outputs corresponding to the mask tokens are passed through a single linear layer to predict the latent tokens at the masked positions.

A.5. Training Settings

Pretraining of A4Mer. We pretrain A4Mer in two steps: an initial training of the first-stage model, and a subsequent end-to-end training of the first- and second-stage models. Both training steps are conducted for 400 epochs with a batch size of 64. Optimization is performed using AdamW with a learning rate of 1×10^{-3} and weight decay of 0.1. We apply a linear warm-up for the first 2,000 iterations, followed by a cosine-annealing schedule over 100,000 iterations with a minimum learning rate of 1×10^{-6}. For all downstream heads, we adopt the same optimization configuration as above, except that the number of training epochs is reduced to 200. During pretraining, the parameters of the target feature extractor $f_{\bar{\theta}}$ are updated using an exponential moving average with a decay rate of 0.996, and 50% of latent tokens are randomly masked to create the pretext task where the model predicts masked latent tokens from visible ones.

Action Recognition. For action recognition, a 1-layer Transformer-encoder classification head is trained by minimizing the cross-entropy loss $\mathcal{L}_{\mathrm{recog}}$ with class-balanced weighting to mitigate class imbalance:

$$\mathcal{L}_{\mathrm{recog}} = -\sum_{\mathrm{c}=1}^{\mathrm{C}} w_{\mathrm{c}}\, y_{\mathrm{c}} \log p_{\mathrm{c}}\,, \tag{F}$$

where p_{c} is the predicted probability for class $\mathrm{c} \in \{1, \ldots, \mathrm{C}\}$, y_{c} is the one-hot label, and w_{c} is the inverse of the frequency of class c in the dataset.

Decoding. Motion prediction and motion interpolation are realized by decoding predicted or interpolated latent tokens into poses. For this, we train a decoding head with a Transformer-encoder architecture. The decoding head is trained using the objective $\mathcal{L}_{\mathrm{dec}}$, which combines the joint-position reconstruction loss $\mathcal{L}_{\mathrm{pos}}$ and the velocity reconstruction loss $\mathcal{L}_{\mathrm{vel}}$:

$$\mathcal{L}_{\mathrm{dec}} = \lambda_{\mathrm{pos}}\mathcal{L}_{\mathrm{pos}} + \lambda_{\mathrm{vel}}\mathcal{L}_{\mathrm{vel}}\,, \tag{G}$$

$$\mathcal{L}_{\mathrm{pos}} = \frac{1}{JT}\sum_{t=1}^{T}\sum_{j=1}^{J}\left\|\hat{\boldsymbol{j}}_{j,t} - \boldsymbol{j}_{j,t}\right\|\,, \tag{H}$$

$$\mathcal{L}_{\mathrm{vel}} = \frac{1}{JT}\sum_{t=2}^{T-1}\sum_{j=1}^{J}\left\|\hat{\Delta\boldsymbol{j}}_{j,t} - \Delta\boldsymbol{j}_{j,t}\right\|\,, \tag{I}$$

$$\hat{\Delta\boldsymbol{j}}_{j,t} = \hat{\boldsymbol{j}}_{j,t} - \hat{\boldsymbol{j}}_{j,t-1}, \quad \Delta\boldsymbol{j}_{j,t} = \boldsymbol{j}_{j,t} - \boldsymbol{j}_{j,t-1}\,, \tag{J}$$

where $\boldsymbol{j}_{j,t}$ denotes the position of the joint j at frame t and $\hat{\boldsymbol{j}}_{j,t}$ is the predicted one. We set λ_{pos} as 1.5 and λ_{vel} as 1.0.

Long-term Motion Prediction. For long-term motion prediction, we train a prediction head for forecasting the next latent token z_{k+1} from the preceding sequence $z_{1:k}$. In addition to minimizing the latent token prediction error $\mathcal{L}_{\mathrm{token}}$, the prediction head also minimizes the joint-space reconstruction error $\mathcal{L}_{\mathrm{dec}}$ computed by feeding the predicted latent tokens into the pretrained, frozen decoder. This auxiliary objective enables the model to focus its predictions on the meaningful dimensions within the latent space.

Moreover, since future segments are not observable at inference time, the prediction head is trained to predict not only latent tokens but also their segment lengths, and minimizes the segment length error $\mathcal{L}_{\mathrm{segm}}$. Since segment lengths follow a long-tailed distribution where short segments are far more common than long ones (see Fig. A), direct regression on the raw scale might cause the loss to be dominated by rare but large values. To alleviate this imbalance, we apply a logarithmic transformation to the ground-truth segment lengths and train the model by minimizing the prediction error in log-space. This transformation compresses the dynamic range of the target values, reduces the undue influence of rare long segments, and yields a more uniform target distribution. Note that the "ground-truth" segment lengths used here are not manually annotated; rather, they are determined by the pretrained A4Mer.

The final prediction loss $\mathcal{L}_{\mathrm{pred}}$ is represented as

$$\mathcal{L}_{\mathrm{pred}} = \mathcal{L}_{\mathrm{token}} + \mathcal{L}_{\mathrm{dec}} + \mathcal{L}_{\mathrm{segm}}\,, \tag{K}$$

$$\mathcal{L}_{\mathrm{token}} = \sum_{k=1}^{K-1}\mathrm{SmoothL1}\left(z_{k+1} - \hat{z}_{k+1}\right)\,, \tag{L}$$

$$\mathcal{L}_{\mathrm{segm}} = \sum_{k=1}^{K-1}\left|\log l_{k+1} - \mathrm{Softplus}\left(\hat{l}_{k+1}\right)\right|\,, \tag{M}$$

where $\hat{\cdot}$ denotes an estimated value by the prediction head, and l_k is the segment length of the latent token k.

For the baseline methods, we similarly train the next-token prediction head using the representations output by each method as input. However, since these methods except $\mathrm{H_2OT}$ [10] rely on fixed, predefined segment lengths, the segment length loss $\mathcal{L}_{\mathrm{segm}}$ is not applied. $\mathrm{H_2OT}$ does not treat consecutive frames as temporal segments. Instead, it groups feature vectors based on their similarity with DPC-kNN [4], producing clusters that do not correspond to fixed temporal durations. As a result, the segment length is inherently undefined for this method. To perform motion prediction under the same formulation as other methods, we approximate the segment length by dividing the input pose-sequence length by the predefined number of clusters and use this value during both training and inference.

Motion Interpolation. For motion interpolation, we train an interpolation head that infers latent tokens of unobserved frames from the latent tokens extracted by A4Mer from partially observed pose sequences. We insert learnable tokens at the latent positions corresponding to the unobserved frames and feed the resulting sequence to the interpolation head. This head is trained by minimizing a loss $\mathcal{L}_{\mathrm{latent}}$ between the predicted latent tokens and the latent tokens extracted by A4Mer from fully observed input sequences. Although the primary goal is to recover the latent tokens of the unobserved frames, partially masking the input also slightly shifts the distribution of the latent tokens corresponding to the observed frames. Therefore, we compute the loss not only for the latent tokens produced from the learnable tokens but also for the latent tokens of the observed frames, allowing the interpolation head to additionally correct the output distribution. Formally, the loss $\mathcal{L}_{\mathrm{latent}}$ is formulated as:

$$\mathcal{L}_{\mathrm{latent}} = \sum_{k=1}^{K}\mathrm{SmoothL1}\left(z_k - \hat{z}_k\right)\,. \tag{N}$$

In addition, as in motion prediction, we introduce an auxiliary objective in the pose space $\mathcal{L}_{\mathrm{dec}}$ by passing the interpolated latent token sequences through the pretrained, frozen decoder.

Table B. Pretraining time. Although A4Mer requires two steps for pretraining, its overall training time is comparable to or faster than other methods.

Method	Step	Time (hours)
MotionBERT [22]	training	32.86
USDRL [18]	training	13.45
PUMPS [11]	training	2.926
MacDiff [19]	training	10.32
BehaveMAE [14]	training	21.83
A4Mer	1st-stage training	3.096
	segmentation	0.056
	end-to-end training	4.400
	total	7.552

A.6. Training Time

Tab. B reports the training times of A4Mer and other representation learning methods. Although A4Mer involves a two-step pre-training procedure unlike the others, its overall training time remains comparable to, or even faster than theirs.

A.7. Datasets

AMD. All representation learning methods, as well as the decoding, motion-prediction, and motion-interpolation heads, are trained on our proposed AMD, which contains 14.2 hours of footage. We split AMD into 80% for training, 10% for validation, and 10% for testing, ensuring that no subjects overlap across splits. While the scenarios performed by participants are shared across splits, variations exist in the specific actions performed, the order of actions, and the arrangement of furniture, depending on the subject. For evaluation of motion prediction and motion interpolation, we automatically exclude trivial sequences where the subject merely remains stationary by detecting them based on the magnitude of joint velocities.

Humans in Kitchens. We train the action recognition head on the Humans in Kitchens (HiK) dataset [16]. Following the protocol in the original paper, recordings from *Kitchen D* are used exclusively for testing, while data from *Kitchens A, B,* and *C* are split into 90% for training and 10% for validation, respectively. Since HiK contains fine-grained, imbalanced, and occasionally noisy action annotations, multi-label classification is unreasonably challenging across all methods we examined, as verified in Tab. C. Instead, we group semantically related actions into coarser, single-label categories and adopt these custom labels for our recognition experiments. For this, we automatically assign new task class labels through the following procedure:

Table C. Multi-label action recognition on HiK with its original fine-grained labels. Although the Action Motifs extracted by A4Mer attain slightly better performance than the others, the overall scores are consistently low, making this evaluation setting inadequate for assessing the utility of the learned representations. Therefore, in the main text, we instead train and evaluate models using the single-label classes that we annotated following the procedure described in Sec. A.7.

Method	mAP ↑
MotionBERT [22]	0.113
USDRL [18]	0.087
PUMPS [11]	0.057
MacDiff [19]	0.072
BehaveMAE [14]	0.113
H_2OT [10]	0.096
A4Mer	**0.121**

1. We first define a set of task classes as shown in the first column of Tab. D.
2. We then categorize the original action labels into four groups: *main actions*, *sub-actions*, *transition actions*, and *others*.
   - **Main actions:** Actions uniquely associated with a specific task class.
   - **Sub-actions:** Actions that accompany main actions and may appear across multiple task classes.
   - **Transition actions:** Movements associated with transitions between actions, such as *sitting down* or *standing up*.
   - **Others:** Rarely occurring actions and state-like labels (*e.g.*, *sitting*) that do not convey task-specific semantics.
3. We next identify all frames annotated with any main action and assign the corresponding task class label to these frames (the first and second columns of Tab. D).
4. For each frame with task class labels, if sub-actions belonging to the same task class appear in its temporal neighborhood, we propagate the task class label to cover the entire contiguous interval containing these sub-actions (the first and third columns of Tab. D).
5. Finally, among the remaining unlabeled frames, those containing transition actions are assigned the task class label associated with the corresponding transition (the first and fourth columns of Tab. D).

Any frames that remain unlabeled after this procedure are excluded from both training and evaluation, resulting in a curated subset comprising 30.8% of the original data.

For motion prediction and motion interpolation evaluation on HiK, we follow the same criterion as in AMD and exclude trivial sequences in which the subject remains nearly stationary.

Task classes	Main actions	Sub-actions	Transition actions
take dish out of cupboard	open cupboard take dish out of cupboard close cupboard		
drink	drink	carry cup	
phone call	phone call		
use smartphone	use smartphone		
use laptop	use laptop		
fussball	fussball		
draw on whiteboard	draw on whiteboard		
erase on whiteboard	erase on whiteboard		
remove sheet from whiteboard	remove sheet from whiteboard		
eat fruit	eat fruit		
peal fruit	peal fruit		
read paper	read paper		
use dishwasher	open dishwasher place in dishwasher close dishwasher		
take cake out of fridge	open fridge take cake out of fridge put cake in fridge close fridge		
take sth. out of drawer	open drawer close drawer		
pour milk	pour milk take milk	carry cup	
washing hands	washing hands	take water from sink	
clean dish	clean dish	take water from sink	
put cup in microwave	put cup in microwave		
take cup out of microwave	take cup out of microwave		
put teabag in cup	put teabag in cup take teabag		
put water in kettle	put water in kettle	take water from sink take kettle	
pour kettle	pour kettle	take kettle	
make coffee	place cup onto coffee machine press coffee button		
take cup from coffee machine	take cup from coffee machine		
mark coffee	mark coffee		
prepare coffee machine	check water in coffee machine take water tank from coffee machine empty water from coffee machine fill water to coffee machine fill water tank place water tank in coffee machine	take water from sink	
cut cake	take piece of cake place cake on plate cut cake in pieces		
place cake on table	place cake on table		
eat cake	eat cake		
put water in glass	put water in glass	take water from sink	
throw in trash	throw in trash		
sitting down			sitting down
standing up			standing up
leaning down			leaning down
kneeling down			kneeling down

Table D. Our definition of the task classes on Humans in Kitchens. We first categorize the original HiK action labels into *main actions*, *sub-actions*, *transition actions* (the second to fourth columns), and *others*, and then assigned our new task classes considering the context of the actions (the first column). HiK action labels not listed in this table are categorized as *others*. This includes labels describging states rather than actions (*e.g.*, "standing", "leaning"), labels that is annotated to co-occur with other actions (*e.g.*, "start microwave" appears while performing another action), and labels that occur only rarely (*e.g.*, "clean countertop"). See Sec. A.7 for details.

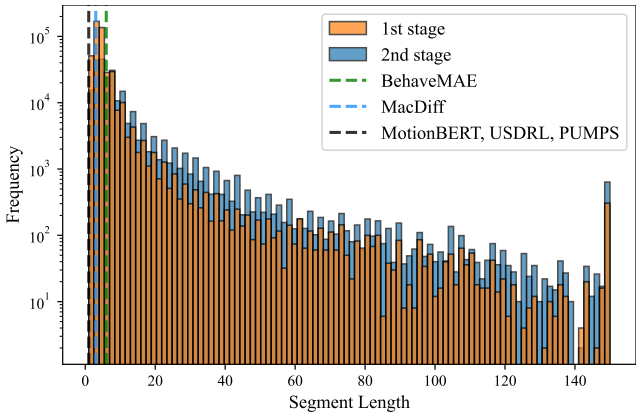

Figure A. Distribution of segment lengths. Compared with the other methods that adopt fixed segment lengths (dashed lines), Action Motif segments are substantially longer. Note that our implementation caps the maximum length at 150, so the rightmost bin has a relatively large count.

B. Additional Experimental Results

B.1. Segment Length Distributions

As shown in Fig. A, our Action Motif segments are substantially longer than those of existing fixed-length representations, indicating a significantly higher compression rate of pose information. Despite this aggressive compression, Action Motifs achieve superior performance across major downstream tasks, as reported in Tab. 2 of the main text, since their segments correspond to semantically meaningful movements.

B.2. Sensitivity to Hyperparameters

We analyze the impact of A4Mer hyperparameters used during pretraining (and, in some cases, during inference) on downstream task performance.

τ_1 in Action Atom Segmentation. Varying τ_1 in Eq. (A) during both pretraining and inference results in different Action Atom segments. As shown in Tab. E, although different choices of τ_1 lead to minor variations in performance, our method generally outperforms existing methods, and achieves comparable performance in the remaining cases.

The Number of Clusters of k-means. During pretraining of A4Mer, we first discretize the Action Atoms output by the pretrained first-stage model using k-means clustering to discover Action Atom patterns. Tab. F reports that $k = 512$ offers consistently solid performance across all tasks, although it is not always the best. Consequently, we adopt $k = 512$ in the main paper. Note that other choices of k achieve performance that is comparable to or exceeds that of competing methods on most tasks.

Table E. Impact of varying τ_1 in the Action Atom segmentation. Across downstream tasks and different values of τ_1, A4Mer outperforms prior methods in most cases and achieves comparable performance otherwise.

τ_1	**Recog** (%) ↑		**Pred** (mm) ↓		**Interp** (mm) ↓	
	k-NN (top-1/-3)	head	AMD	HiK	AMD	HiK
0.01	30.2 / 59.3	39.6	164	128	134	118
0.001	31.1 / 57.1	34.5	155	138	142	124
0.005	31.7 / 59.0	38.1	150	120	126	110
best in priors	31.1 / 46.5	35.6	167	132	137	110

Table F. Effect of varying k in the k-means clustering. Although not always optimal, $k = 512$ offers consistently strong performance across all tasks. We therefore adopt $k = 512$ in the main paper. Other choices of k remain comparable to, and in many cases exceed, prior methods.

k for k-means	**Recog** (%) ↑		**Pred** (mm) ↓		**Interp** (mm) ↓	
	k-NN (top-1/-3)	head	AMD	HiK	AMD	HiK
128	29.1 / 57.7	35.2	163	130	140	125
256	28.0 / 57.1	35.9	155	126	134	117
512	31.7 / 59.0	38.1	150	120	126	110
1024	31.5 / 58.8	38.6	156	121	123	112
best in priors	31.1 / 46.5	35.6	167	132	137	110

Table G. Effect of varying o_{max} and o_{min} in GSP. Although different threshold choices lead to slight variations in downstream task performance, A4Mer outperforms prior methods in most cases and remains competitive otherwise.

o_{max} / o_{min}	**Recog** (%) ↑		**Pred** (mm) ↓		**Interp** (mm) ↓	
	k-NN (top-1/-3)	head	AMD	HiK	AMD	HiK
30 / 5	32.0 / 60.3	38.8	143	116	130	110
20 / 5	31.8 / 60.1	39.2	153	116	128	112
15 / 5	31.7 / 59.0	38.1	150	120	126	110
10 / 5	28.0 / 55.9	41.7	161	126	131	120
15 / 3	32.3 / 61.4	39.2	166	134	131	117
15 / 7	30.9 / 58.8	38.8	167	131	130	114
best in priors	31.1 / 46.5	35.6	167	132	137	110

o_{max} and o_{min} in GSP. As shown in Tab. G, varying the occurrence thresholds o_{max} and o_{min} in GSP leads to slight performance variations, yet A4Mer achieves competitive performance across all threshold settings and, in most cases, outperforms prior methods across downstream tasks.

B.3. Additional Qualitative Results

We show additional qualitative results of motion prediction in Fig. C and motion interpolation in Fig. D.

C. Details of AMD

Here, we describe the details of the SMPL annotation procedure for AMD as well as the contents of the dataset.

Figure B. Foot cameras attached to subjects' feet. These cameras do not restrict the subjects' movements and are small enough to remain invisible to the room-mounted cameras.

C.1. Shape Fitting

Before recording daily activities, each subject performs an A-pose, which is captured by four RGB-D cameras (RealSense L515) separate from the room RGB cameras. We fit a SMPL model to the obtained point cloud [2, 3] and use the optimized shape parameters $\boldsymbol{\beta}$ as ground truth.

C.2. Foot Camera Localization

To robustly annotate SMPL parameters despite occlusions around the legs, which frequently occur in natural indoor environments, we attach tiny cameras (Insta360 GO3) to the feet (Fig. B) and place ChArUco markers [5] on the ceiling and the undersides of tables. To localize the foot cameras with the Perspective-n-Points (PnP) algorithm, we need to know the 3D positions of the markers in the global coordinate system. Before recording subjects, we first calibrated the extrinsic parameters of the room-mounted cameras ('room cameras') using Structure-from-Motion on videos capturing a marker rig moving throughout the room. During this calibration, the ceiling markers observed from the room cameras were also accurately localized. To localize additional markers—including those placed on the underside of tables, which are not observed by the room cameras—we captured supplementary images from diverse viewpoints targeting these markers. The camera poses for these newly captured images were estimated via PnP using the already-localized markers as reference points. We then perform bundle adjustment to localize the previously unobserved markers, while ensuring consistency with the already localized ones. Through this two-stage marker localization, the 3D positions of all markers in the environment were obtained.

During recording subjects, we estimated the pose of each foot camera in every frame by applying the PnP algorithm to the localized markers detected from the foot camera. Note that the room cameras and the foot cameras were time-synchronized using QR codes displayed at the beginning of each session, in the same manner as Ego-Exo4D [6].

C.3. Postprocessing of Foot Camera Pose

When subjects perform rapid or large movements, the images from the foot cameras might suffer from momentary motion blur, causing marker detection to fail. Besides, the PnP algorithm might occasionally produce inaccurate pose estimates. To reduce the number of frames in which the camera pose cannot be localized and to obtain temporally consistent trajectories, we apply the following postprocessing procedure. First, we compare the raw PnP estimates with predictions smoothed by the Rauch–Tung–Striebel (RTS) smoother [12] and identify frames with large discrepancies as outliers, which are subsequently discarded. Next, we apply an exponential moving average to the foot camera rotations and translations to further stabilize the trajectories. Finally, we interpolate the poses of unlocalized frames using Kochanek-Bartels splines [9]. The resulting foot camera poses are illustrated in Fig. E.

C.4. Fitting Objective Derivation

To achieve accurate SMPL fitting by leveraging the foot camera poses, we augment Eq. (7) in the main text with additional constraint terms. To prevent the fitted SMPL model from intersecting the foot cameras, we add a loss term E_u to ensure that the SMPL foot vertices V_{foot} remain below the foot camera plane P_{cam} formed by the width and height axes of the camera coordinate system:

$$E_u = \sum_{\boldsymbol{v} \in V_{\text{foot}}} \max(\boldsymbol{v} \cdot \boldsymbol{n}_{\text{cam}} + d_{\text{cam}}, 0) \ , \tag{O}$$

where $\boldsymbol{n}_{\text{cam}}$ and d_{cam} denote the normal vector and the constant of P_{cam}, respectively. To fully leverage the foot camera information, we also incorporate E_d and E_a into Eq. (7). E_d constrains the SMPL foot dorsum vertices V_{drsm} to lie close to the foot camera plane P_{cam}. E_a penalizes the SMPL foot bone $\boldsymbol{b}$, the vector from the toe to the heel joint, to be in the same direction as a designated axis $\boldsymbol{n}_b$ of the foot camera coordinate frame. This angular constraint is realized by the physical mounting of the foot camera such that one of its coordinate axes is oriented along the toe-to-heel direction of the foot. These loss terms are formulated as:

$$E_d = \sum_{\boldsymbol{v} \in V_{\text{drsm}}} |\boldsymbol{v} \cdot \boldsymbol{n}_{\text{cam}} + d_{\text{cam}}| \ , \tag{P}$$

$$E_a = 1 - \frac{\boldsymbol{b} \cdot \boldsymbol{n}_b}{\|\boldsymbol{b}\|} \ . \tag{Q}$$

In addition to the aforementioned constraints, we also add loss terms, E_R and E_t, which are introduced in the

main text, to preserve the relative pose between each foot camera and the corresponding SMPL foot. We define a SMPL-foot coordinate system whose origin $\boldsymbol{t}_{\text{smpl}}$ is placed at the toe joint. The basis vectors $\boldsymbol{R}_{\text{smpl}}$ are constructed as follows: (i) the first axis is obtained by projecting $\boldsymbol{b}$ onto the plane of the foot dorsum computed from V_{drsm} with cross product and normalizing it; (ii) the second axis is the normal vector of this plane; and (iii) the third axis is defined to be orthogonal to the preceding two axes. From the fitted result of the A-pose videos, we derive a rigid transformation $(\boldsymbol{R}^{\text{cam}}_{\text{smpl}}, \boldsymbol{t}^{\text{cam}}_{\text{smpl}})$ that maps the foot camera pose $(\boldsymbol{R}_{\text{cam}}, \boldsymbol{t}_{\text{cam}})$ to the corresponding SMPL foot pose $(\boldsymbol{R}_{\text{smpl}}, \boldsymbol{t}_{\text{smpl}})$. E_R and E_t enforce that the relative rotation between $\boldsymbol{R}_{\text{cam}}$ and $\boldsymbol{R}_{\text{smpl}}$, and the relative translation between $\boldsymbol{t}_{\text{cam}}$ and $\boldsymbol{t}_{\text{smpl}}$, remain consistent with these transformations:

$$E_R = \left\| \boldsymbol{I} - \text{diag}\{ \left(\boldsymbol{R}_{\text{cam}} \boldsymbol{R}^{\text{cam}}_{\text{smpl}}\right)^\top \boldsymbol{R}_{\text{smpl}} \} \right\| , \quad \text{(R)}$$

$$E_t = \left\| \boldsymbol{t}_{\text{smpl}} - \left(\boldsymbol{R}_{\text{cam}} \boldsymbol{t}^{\text{cam}}_{\text{smpl}} + \boldsymbol{t}_{\text{cam}}\right) \right\| , \quad \text{(S)}$$

where $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ denotes the identity matrix. Combining these terms with Eq. (7), the full optimization objective is given by

$$E(\boldsymbol{\theta}) = \lambda_{\boldsymbol{J}} E_{\boldsymbol{J}} + \lambda_{\text{reg}} E_{\text{reg}} + \lambda_{\text{foot}} E_{\text{foot}} , \quad \text{(T)}$$

$$E_{\text{foot}} = \lambda_u E_u + \lambda_d E_d + \lambda_a E_a + \lambda_R E_R + \lambda_t E_t , \quad \text{(U)}$$

where each λ denotes the weight for the corresponding loss term.

After the SMPL fitting with Eq. (T), we further improve temporal consistency by imposing a smoothness constraint on the SMPL joints $\boldsymbol{J}$ and the pose parameters $\boldsymbol{\theta}$ through the velocity loss $\mathcal{L}_{\text{vel}}$ and acceleration loss $\mathcal{L}_{\text{acc}}$. We dynamically modulate the weights of these loss terms based on the average 2D joint confidence c estimated by ViTPose [20] across all views, assigning larger weights to joints with lower confidence. Furthermore, for frames in which the foot camera positions remain nearly constant, *i.e.*, the foot is effectively stationary, we assign larger weights to the foot joints of these two terms. This second-stage SMPL fitting minimizes the following objective function:

$$E^{\text{2nd}}(\boldsymbol{\theta}) = \sum_t E^t + \lambda_{\text{vel}} E^t_{\text{vel}} + \lambda_{\text{acc}} E^t_{\text{acc}} , \quad \text{(V)}$$

$$E^t_{\text{vel}} = \sum_j \Lambda_j \left(\left\| \boldsymbol{J}^t_j - \boldsymbol{J}^{t-1}_j \right\| + \boldsymbol{\delta}^t_j \right) , \quad \text{(W)}$$

$$E^t_{\text{acc}} = \sum_j \Lambda_j \left(\left\| \boldsymbol{J}^t_j - 2\boldsymbol{J}^{t-1}_j + \boldsymbol{J}^{t-2}_j \right\| + \boldsymbol{\delta}^t_j - \boldsymbol{\delta}^{t-1}_j \right) , \quad \text{(X)}$$

$$\boldsymbol{\delta}^t_j = d_{\text{geo}} \left(\boldsymbol{\theta}^t_j, \boldsymbol{\theta}^{t+1}_j \right) , \quad \text{(Y)}$$

$$\Lambda_j = \frac{1}{c_j} + \lambda_s \mathbb{1} \left\{ \left\| \boldsymbol{t}^t_{\text{cam}} - \boldsymbol{t}^{t-1}_{\text{cam}} \right\| < \epsilon \right\} \quad \text{(Z)}$$

where j and t denote the joint and time indices, respectively, $d_{\text{geo}}(\cdot,\cdot)$ is the geodesic distance, and $\mathbb{1}\{\cdot\}$ is the indicator function. This optimization is performed every 100 frames.

C.5. Evaluation of the Effectiveness of the Foot Camera Constraint Using 2D Masks

To quantitatively evaluate the effectiveness of the foot camera constraint, we compute the IoU between the human masks predicted by OneFormer [8], which we regard as a pseudo ground truth mask, and the mask obtained by rendering the fitted SMPL mesh from the corresponding camera viewpoint, as described in Sec. 4.2 of the main text.

For a more reliable evaluation, we further apply two preprocessing steps to the pseudo ground truth mask: (i) removing regions that extend beyond the body surface (e.g., clothing and hair), and (ii) interpolating missing regions in the mask caused by occlusions.

Removal of Regions Extending Beyond the Body Surface. The human masks predicted by OneFormer include regions corresponding to clothing and hair, which often extend beyond the actual body surface. As a result, these masks tend to be larger than the masks obtained from the SMPL model, which represents only the body surface. To remove such extraneous regions from the pseudo ground truth mask, we apply a morphological erosion operation.

Interpolation of Missing Regions Caused by Occlusion. For the evaluation, we randomly sample 10,000 frames in which the full body is visible. This is determined based on the confidence scores of the body joints predicted by ViTPose [20]. However, even with this sampling, some frames still contain partial occlusions by furniture or other objects, resulting in unobserved, missing regions in the human mask.

To mitigate this issue, we render the SMPL meshes obtained from the fittings with and without the foot camera constraint. We then treat the overlapping regions of the two rendered masks as a reliable human region, since this is consistently supported by both fitting configurations. This region is added to the pseudo ground truth mask predicted by OneFormer.

This procedure not only restores human regions that were occluded, but also compensates for regions that may have been excessively removed by the aforementioned erosion process.

C.6. Contents of AMD

In Tab. H, we summarize the diverse daily actions contained in AMD. We show sample sequences of AMD in Figs. F and G.

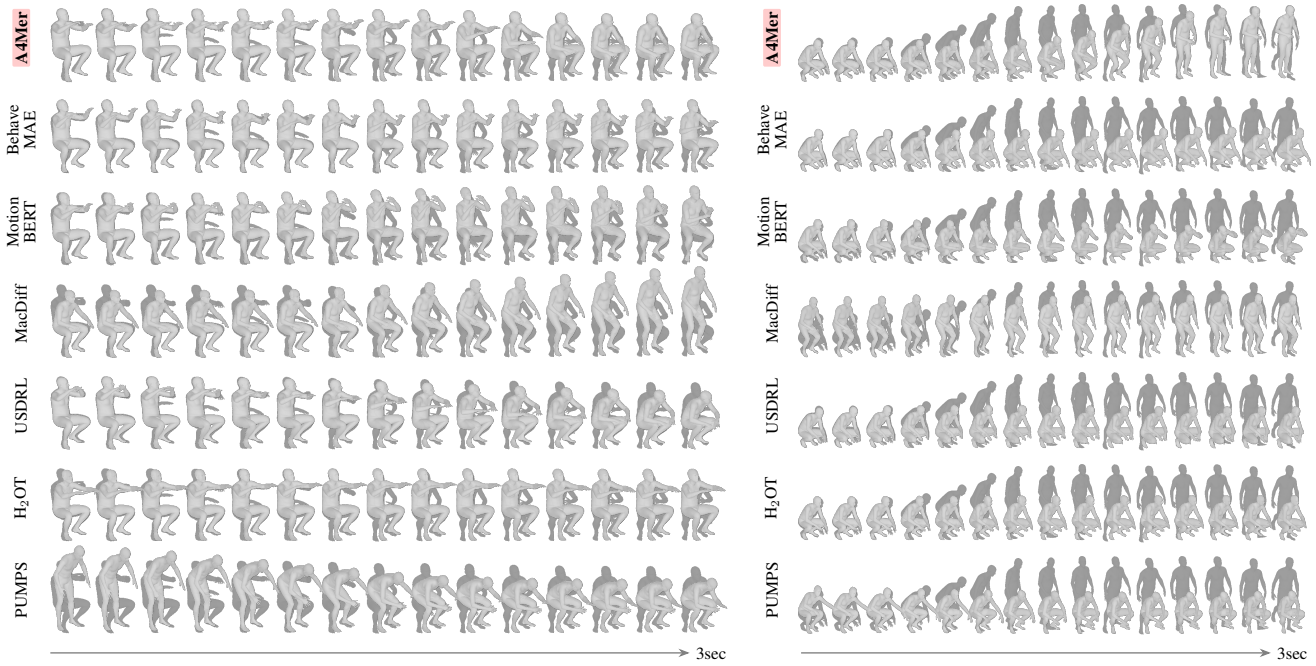

Figure C. Predicted poses on AMD obtained via auto-regressive latent token prediction and decoding. The dark-colored SMPLs indicate the ground-truth.

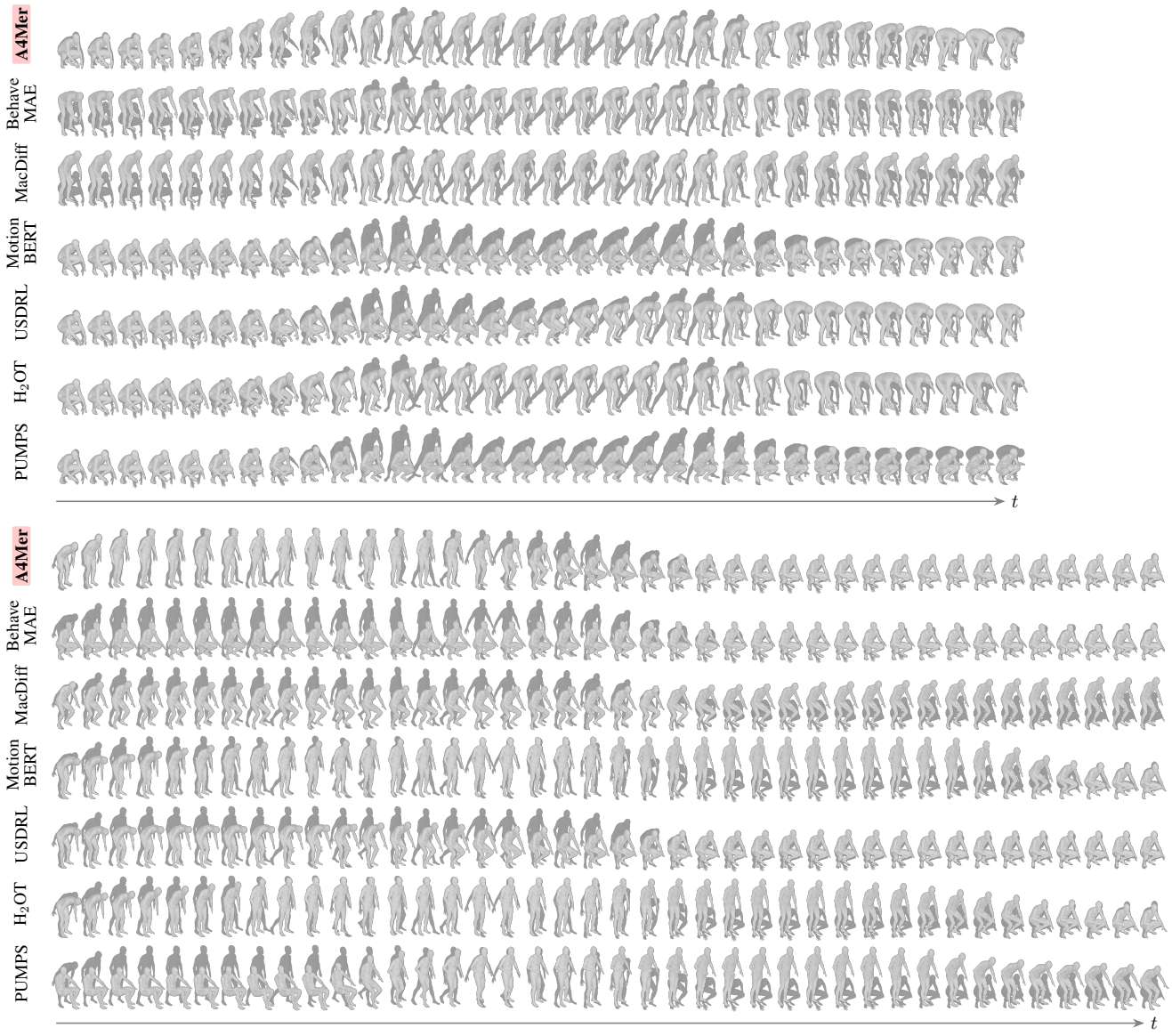

Figure D. Interpolated poses through latent token interpolation and decoding. The SMPLs in dark color are the ground-truth.

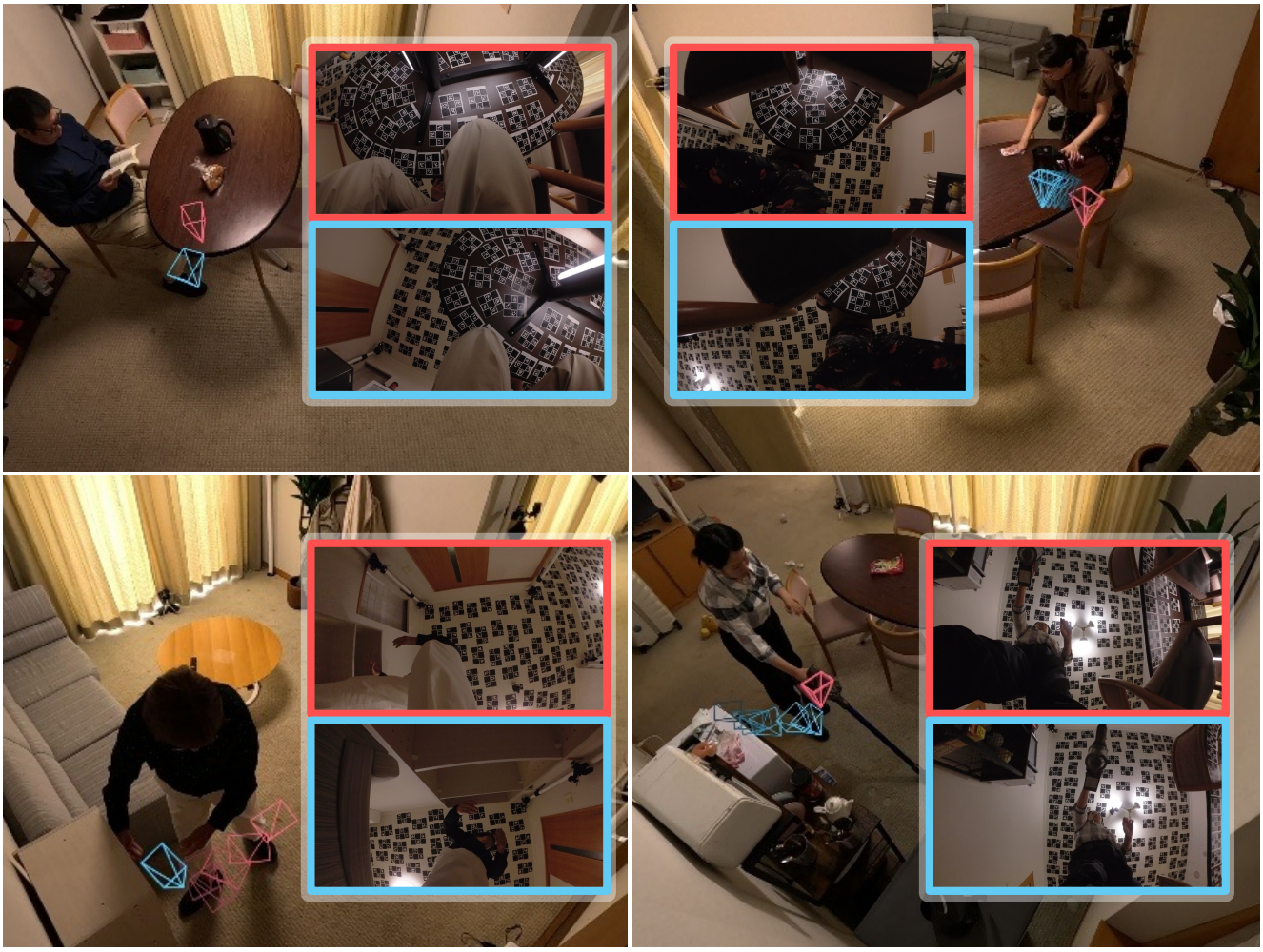

Figure E. Localized foot camera poses and the images captured from these cameras. The wide field of view of the foot cameras enables capture of many markers and accurate pose localization.

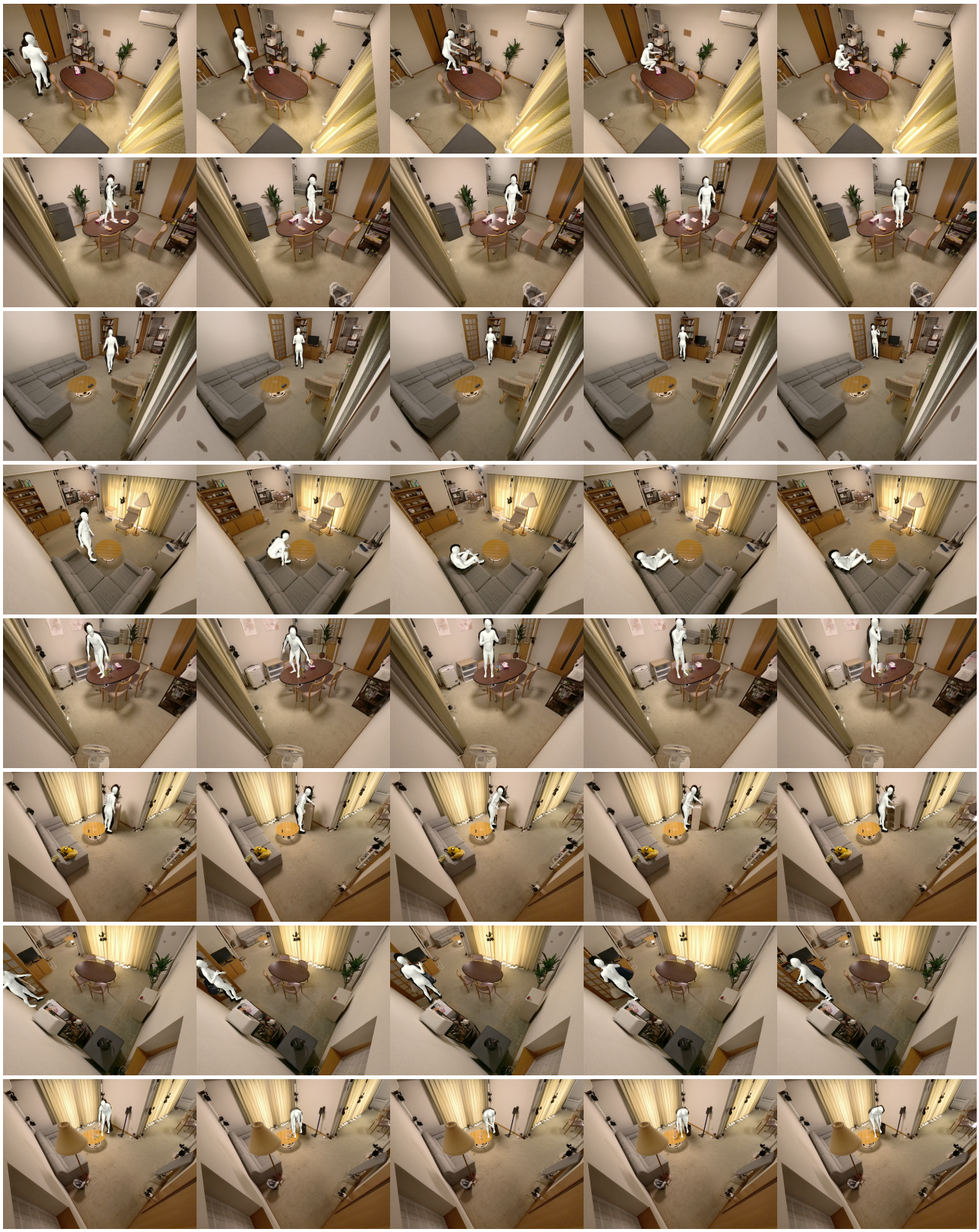

Figure F. Samples of AMD. Our dataset contains videos of people conducting various daily activities with accurate per-frame SMPL annotations.

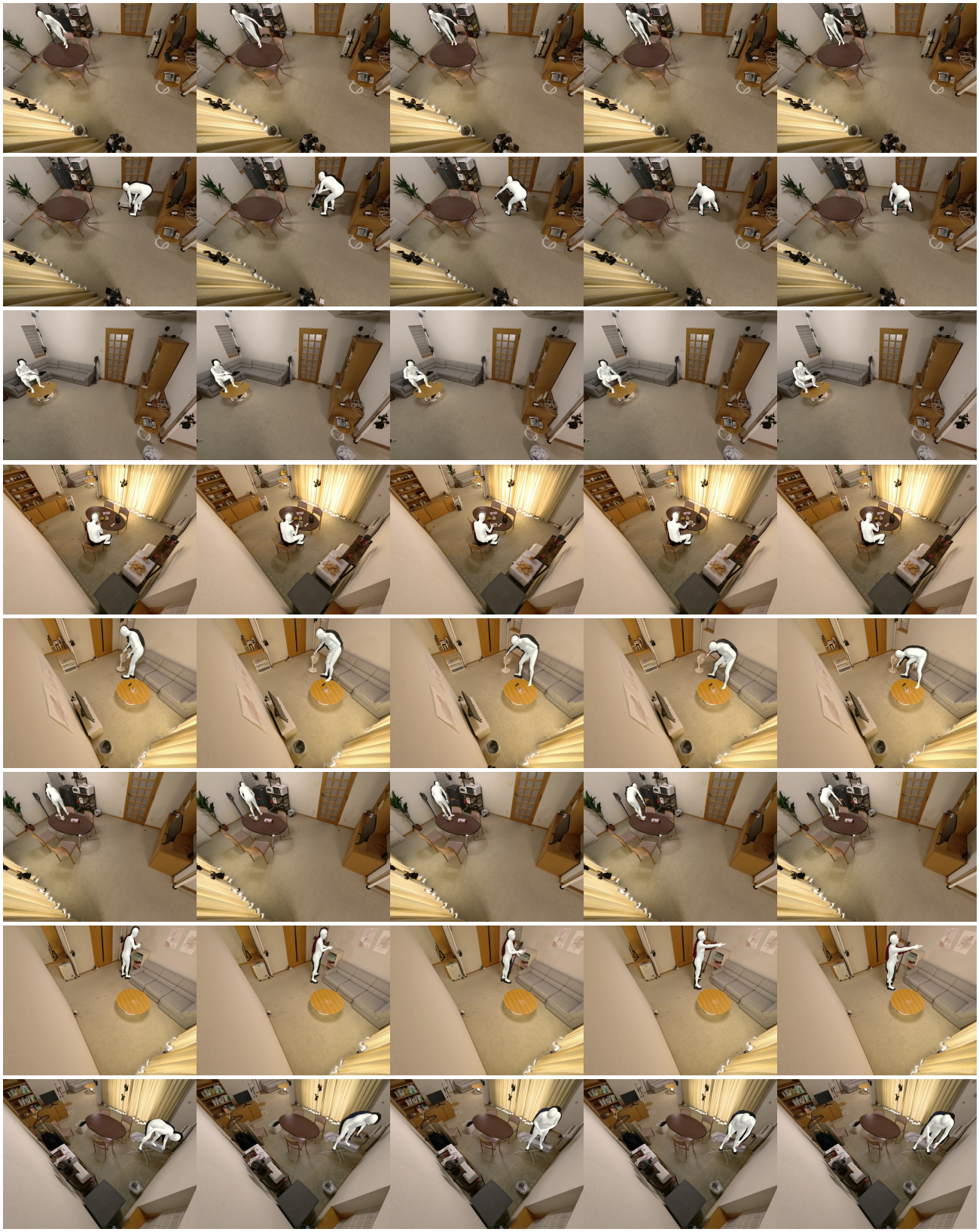

Figure G. More samples of AMD.

Table H. AMD includes 129 daily actions. Actions performed in nearly identical poses are grouped under labels such as "Do sth." to streamline similar movements.

Adjust electric fan direction	Make instant soup	Stand up
Answer the call	Make tea	Stir sth
Assemble shelf	Move sth	Take down laundry
Blow nose	Notice incorrect time	Take down wall clock
Carry sth to swh	Notice ringing phone	Take medicine
Change TV channel	Open door of sth	Take off bag
Check the time	Open up laptop	Take off clothes
Chill sth in fridge	Open up suitcase	Take off glasses
Choose book	Pack suitcase	Take off lid
Choose suitable utensil	Pause sth	Take out required dose of medicine
Clean	Pick up sth	Take sth out of swh
Clip clothes with pegs	Place sth on swh	Tear open packet
Close book	Play game with smartphone	Tidy up
Close door of sth	Plug sth into outlet	Tie plastic bag
Close drawer	Point sth at sth	Toast bread
Close laptop	Pour liquid into cup	Transfer trash to bag
Close up suitcase	Pour powder into cup	Turn on TV
Collect trash	Press button	Turn on air conditioner
Dispose of trash	Pull open sth	Turn on ceiling light
Draw line with ruler	Pull out drawer	Turn on electric fan
Drink sth	Put away sth	Turn on reading lamp
Drop sth	Put lid on sth	Turn off TV
Eat food	Put on bag	Turn off air conditioner
Eat soup	Put on coat	Turn off ceiling light
Empty shelf	Put on glasses	Turn off electric fan
End call	Reach for sth	Turn off reading lamp
Erase with eraser	Read book	Unclip small clothes from pegs
Fold clothes	Read newspaper	Unplug sth from outlet
Fold drying rack	Remove battery	Use lint roller on floor
Fold iron board	Remove hanger from clothes	Use calculator
Fold newspaper	Set the time on clock	Use laptop
Hang clothes on hanger	Set timer	Use mouse
Hang laundry	Set up drying rack	Use phone
Heat food with microwave	Set up iron board	Use fork
Insert battery	Sit down	Use spoon
Insert bookmark	Sit on chair	Vacuum floor
Iron clothes	Sit on sofa	Wait for timer
Lean sth against wall	Sort laundry	Walk
Lie down	Spill liquid	Watch TV
Lie on sofa	Spread out newspaper	Wipe table
Listen to music with headphones	Spread out washed towel	Wipe up liquid
Look for sth	Stand on chair	Wrap dish
Make coffee	Stand sth up	Write sth

References

[1] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-Supervised Learning from Images With a Joint-Embedding Predictive Architecture. In *CVPR*, pages 15619–15629, 2023. 2

[2] Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. Combining Implicit Function Learning and Parametric Models for 3D Human Reconstruction. In *ECCV*, pages 311–329, 2020. 7

[3] Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. LoopReg: Self-Supervised Learning of Implicit Surface Correspondences, Pose and Shape for 3D Human Mesh Registration. In *NeurIPS*, pages 12909–12922, 2020. 7

[4] Mingjing Du, Shifei Ding, and Hongjie Jia. Study on Density Peaks Clustering Based on k-Nearest Neighbors and Principal Component Analysis. *Knowledge-Based Systems*, 99:135–145, 2016. 3

[5] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. 7

[6] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-Exo4D: Understanding Skilled Human Activity from First- and Third-Person Perspectives. In *CVPR*, pages 19383–19400, 2024. 7

[7] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating Diverse and Natural 3D Human Motions From Text. In *CVPR*, pages 5152–5161, 2022. 1

[8] Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One Transformer to Rule Universal Image Segmentation. In *CVPR*, pages 2989–2998, 2023. 8

[9] Doris HU Kochanek and Richard H Bartels. Interpolating Splines with Local Tension, Continuity, and Bias Control. In *ACM SIGGRAPH Conference Papers*, pages 33–41, 1984. 7

[10] Wenhao Li, Mengyuan Liu, Hong Liu, Pichao Wang, Shijian Lu, and Nicu Sebe. H_2OT: Hierarchical Hourglass Tokenizer for Efficient Video Pose Transformers. *IEEE TPAMI*, pages 1–15, 2025. 3, 4

[11] Clinton Ansun Mo, Kun Hu, Chengjiang Long, Dong Yuan, Wan-Chi Siu, and Zhiyong Wang. PUMPS: Skeleton-Agnostic Point-Based Universal Motion Pre-Training for Synthesis in Human Motion Tasks. In *ICCV*, pages 14496–14506, 2025. 4

[12] H. E. RAUCH, F. TUNG, and C. T. STRIEBEL. Maximum likelihood estimates of Linear Dynamic Systems. *AIAA Journal*, 3(8):1445–1450, 1965. 7

[13] Ramakrishnan Srikant and Rakesh Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *EDBT*, pages 1–17, 1996. 1

[14] Lucas Stoffl, Andy Bonnetto, Stéphane d'Ascoli, and Alexander Mathis. Elucidating the Hierarchical Nature of Behavior with Masked Autoencoders. In *ECCV*, pages 106–125, 2024. 4

[15] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. 2

[16] Julian Tanke, Oh-Hun Kwon, Felix B Mueller, Andreas Doering, and Jürgen Gall. Humans in Kitchens: A Dataset for Multi-Person Human Motion Forecasting with Scene Context. In *NeurIPS*, pages 10184–10196, 2023. 4

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *NeurIPS*, 2017. 2

[18] Wanjiang Weng, Hongsong Wang, Junbo Wang, Lei He, and Guo-Sen Xie. USDRL: Unified Skeleton-Based Dense Representation Learning with Multi-Grained Feature Decorrelation. In *AAAI*, pages 8332–8340, 2025. 4

[19] Lehong Wu, Lilang Lin, Jiahang Zhang, Yiyang Ma, and Jiaying Liu. MacDiff: Unified Skeleton Modeling with Masked Conditional Diffusion. In *ECCV*, pages 110–128, 2024. 4

[20] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation. *NeurIPS*, 35:38571–38584, 2022. 8

[21] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the Continuity of Rotation Representations in Neural Networks. In *CVPR*, pages 5745–5753, 2019. 1

[22] Wentao Zhu, Xiaoxuan Ma, Zhaoyang Liu, Libin Liu, Wayne Wu, and Yizhou Wang. MotionBERT: A Unified Perspective on Learning Human Motion Representations. In *ICCV*, pages 15085–15099, 2023. 4