Supplementary Material: Ego-Body Pose Estimation via Ego-Head Pose Estimation

Jiaman Li C. Karen Liu† Jiajun Wu†
Stanford University
{jiamanli,karenliu,jiajunwu}@cs.stanford.edu

The supplementary material includes details on the model architecture, implementation, baselines, SLAM analysis, and synthetic dataset. We also encourage readers to watch our video demo for more qualitative results.

1. Model Architecture

HeadNet. Our HeadNet is a transformer-based model [12] consisting of two self-attention blocks, each of which has a multi-head attention layer followed by a position-wise feed-forward layer. In our implementation, we set the number of heads in each block $n_{head} = 4$. We illustrate the operation of each self-attention block in Figure 1 where $\boldsymbol{x}_t$ denotes the input feature vector and $\boldsymbol{x}'_t$ denotes the updated feature vector in time step t.

Specifically, given $\boldsymbol{X} \in \mathbb{R}^{T \times D_{input}}$, we first embed the input sequence into three matrices, namely keys $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K$, queries $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^Q$, and values $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V$, where $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$ are matrices with dimension $T \times D_K$, $T \times D_Q$ and $T \times D_V$. We split each matrix into multiple heads where each head is $T \times D_i$, with $D_K = \sum_i D_i$.

For each head, we compute the features using scaled-dot attention as

$$\boldsymbol{Y}_i = \text{Softmax}\left(\frac{\boldsymbol{Q}_i \boldsymbol{K}_i^T}{\sqrt{D_i}}\right)\boldsymbol{V}_i. \tag{1}$$

We concatenate $\boldsymbol{Y}_i$ to form $\boldsymbol{Y}$ with size $T \times D_V$ and input it to two position-wise 1D convolution layers followed by layer normalization. We denote the dimension of the output feature vector in each block as D_{output}. Obtaining the features $T \times D_{output}$ from transformer blocks, we append an MLP to project features to predicted angular velocity and distance scalar values. In our implementation, we set the dimensions $D_{input}, D_{output}, D_K, D_Q, D_V$ to 256.

GravityNet. Our GravityNet shares the same model architecture as HeadNet.

Conditional Diffusion Model. Our conditional diffusion model for full-body pose generation from the head pose de-

† indicates equal contribution.

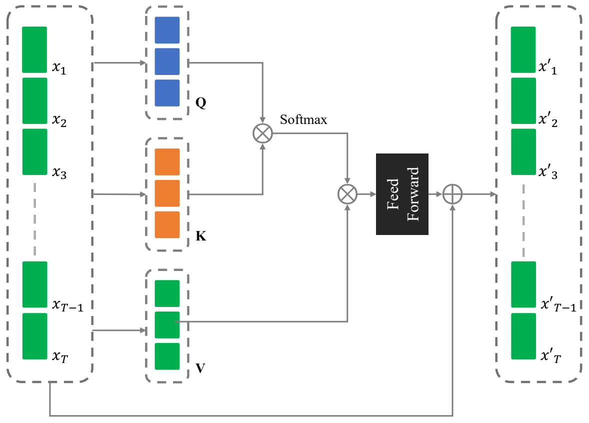

Figure 1. Illustration of the self-attention block used in our transformer-based model architecture.

ploys a similar transformer-based architecture in each denoising step. We use four self-attention blocks and the number of heads in each block is $n_{head} = 4$. The window length T is 120. And the dimensions $D_{input}, D_{output}, D_K, D_Q, D_V$ are set to $512, 512, 256, 256, 256$.

Sliding Window Strategy for Long Sequence Generation. To generate full-body poses given head poses for a long sequence (more than 120 frames), we propose a sliding-window strategy. Suppose we have full-body pose predictions $\boldsymbol{p}_0^k, \boldsymbol{p}_1^k, ..., \boldsymbol{p}_T^k$ of k_{th} window, and we set the overlap steps to 10, when we estimate full-body poses for the $(k+1)_{th}$ window, we use $\boldsymbol{p}_{T-9}^k, \boldsymbol{p}_{T-8}^k, ..., \boldsymbol{p}_T^k$ to replace $\boldsymbol{p}_0^{k+1}, \boldsymbol{p}_1^{k+1}, ..., \boldsymbol{p}_9^{k+1}$ during the denoising step of each noise level.

Implementation Details. Our implementation uses PyTorch [9]. During the training of HeadNet and GravityNet, we start with a learning rate of 0.0001 and use the AdamW optimizer [4] with batch size 8. The training takes about 20 hours for HeadNet and 15 hours for GravityNet to converge using a single NVIDIA Titan RTX GPU. For our conditional diffusion model, we start with a learning rate of 0.0001 and use the Adam optimizer [3] with batch size 32. The training takes about 24 hours to converge using a single NVIDIA Titan RTX GPU.

2. Details of Baselines

AvatarPoser. We use the official code released by Jiang et al. [1]. AvatarPose uses a transformer-based model architecture similar to our denoising network in the conditional diffusion model. The number of self-attention blocks is 3, and the number of heads in each block is 8. We use their default learning rate of 0.0001 and the Adam optimizer. The training takes about one day to converge using a single NVIDIA Titan RTX GPU.

Kinpoly-Head. We use the official code released by Luo et al. [5]. We modify Kinpoly to a setting that takes the head pose as the only input. Kinpoly-Head consists of a ContextRNN and an ActionRNN, as shown in Figure 2. The ContextRNN takes head velocity $\boldsymbol{V} \in \mathbb{R}^{T\times 6}$ as the only input, estimates the first body pose $\boldsymbol{X}_1$ by averaging features from all the time steps denoted as $\bar{\boldsymbol{F}}$ and provides context features $\boldsymbol{F}_1, \boldsymbol{F}_2, ..., \boldsymbol{F}_T$ as input for ActionRNN. In each step, they feed the context features $\boldsymbol{F}_t$, pose state in previous step $\boldsymbol{X}_{t-1}$ and head pose $\boldsymbol{H}_t$ to ActionRNN, and predict the body pose state $\boldsymbol{X}_t$. We use their default learning rate of 0.0001 and the Adam optimizer. The training takes about three days to converge using a single NVIDIA Titan RTX GPU.

PoseReg. We use the official code released by Ye et al. [13]. PoseReg consists of a Bidirectional LSTM model with a hidden dimension of 1024. We use their default learning rate of 0.0001 and the Adam optimizer. The training takes about one day to converge using a single NVIDIA Titan RTX GPU.

Kinpoly-OF. We use the official code released by Luo et al. [5]. We modify Kinpoly so that it takes optical flow features as the only input. The model architecture is the same as Kinpoly-Head. We use their default learning rate of 0.0001 and the Adam optimizer. The training takes about three days to converge using one NVIDIA Titan RTX GPU.

3. SLAM Analysis

Our hybrid approach for head pose estimation relies on the head translation predicted by DROID-SLAM. If DROID-SLAM fails to predict head pose for a given egocentric video, our approach cannot produce accurate head pose prediction results and decent full-body predictions. To eliminate the effects of DROID-SLAM on our evaluations, we compute the upper bound for all the head trajectories extracted using DROID-SLAM and discard the sequences in which DROID-SLAM failed (the upper bounds have large errors). We set threshold $\mathbf{O}_{head} = 0.9$, $\mathbf{T}_{head} = 300$, and remove the sequences that have an upper-bound $\mathbf{O}_{head}$ or $\mathbf{T}_{head}$ larger than the threshold. The number of sequences in our original testing datasets is 300, 165, and 83 for ARES, Kinpoly-MoCap, and GIMO, respectively. We show the statistics of the selected sequences in Table 1.

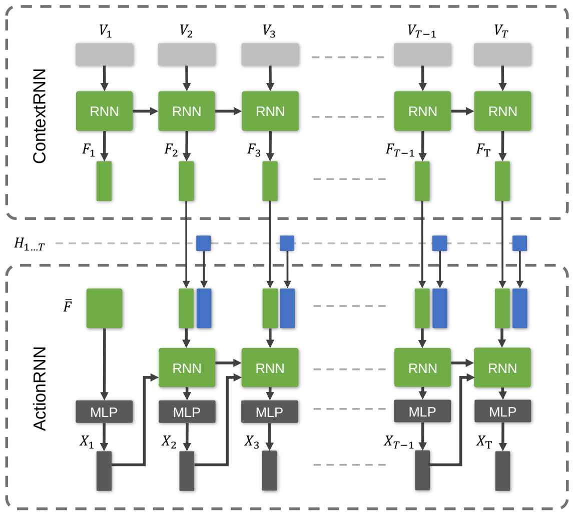

Figure 2. Model architecture of the baseline Kinpoly-Head.

	# of sequences	$\mathbf{O}_{head}$	$\mathbf{T}_{head}$
ARES	215	0.45	31.94
Kinpoly-MoCap	75	0.57	121.42
GIMO	34	0.72	62.08

Table 1. DROID-SLAM's upper bound on three test sets.

4. More Details of Synthetic Dataset (ARES)

Our synthetic dataset (ARES) combines the motion from AMASS [6] and 3D scenes from Replica [10] and uses the AI Habitat [7, 11] for fast and realistic rendering.

SDF Processing. Because the meshes in Replica are not watertight, we first make the scene meshes watertight using Poisson reconstruction [2]. Subsequently, we convert the meshes to signed distance function (SDF) using a public repository [8] to facilitate the calculation of penetration loss with a human mesh.

Efficient Synthesis Strategy. For each motion sequence from AMASS, we randomly place and orient the root trajectory in the scene under the condition that the human mesh at the beginning of the trajectory is in contact with the floor. As described in the paper, we compute penetration loss for each human mesh of the sequence with the 3D scene. If the number of penetration-free meshes is less than 60, we discard the current sequence and try another random rotation and placement. We employ an efficient data generation strategy that we only experiment 10 times for each motion sequence. We showcase more examples from our synthetic dataset in Figure 3 and Figure 4.

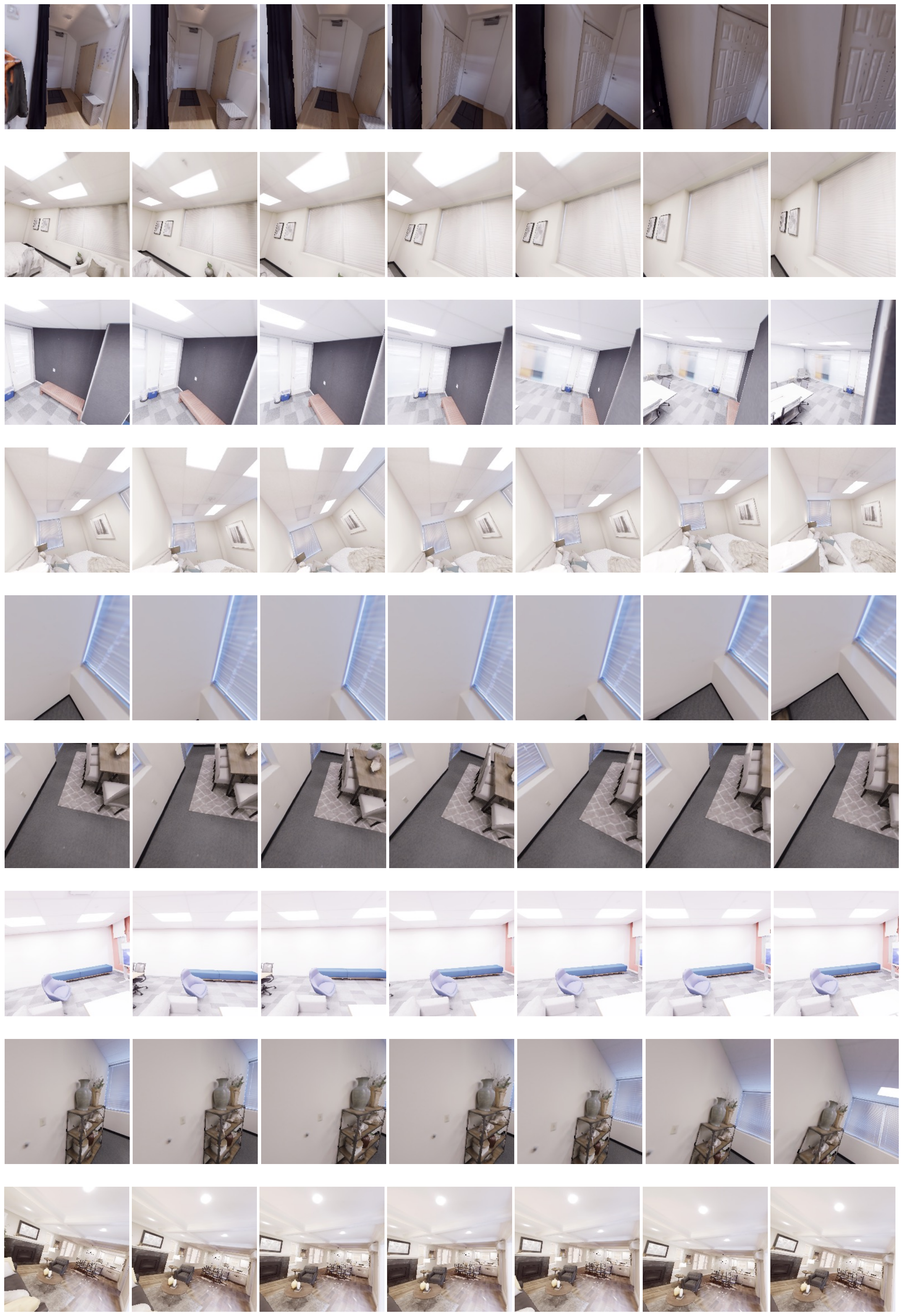

Figure 3. More examples from ARES. Each row represents down-sampled frames from a different sequence.

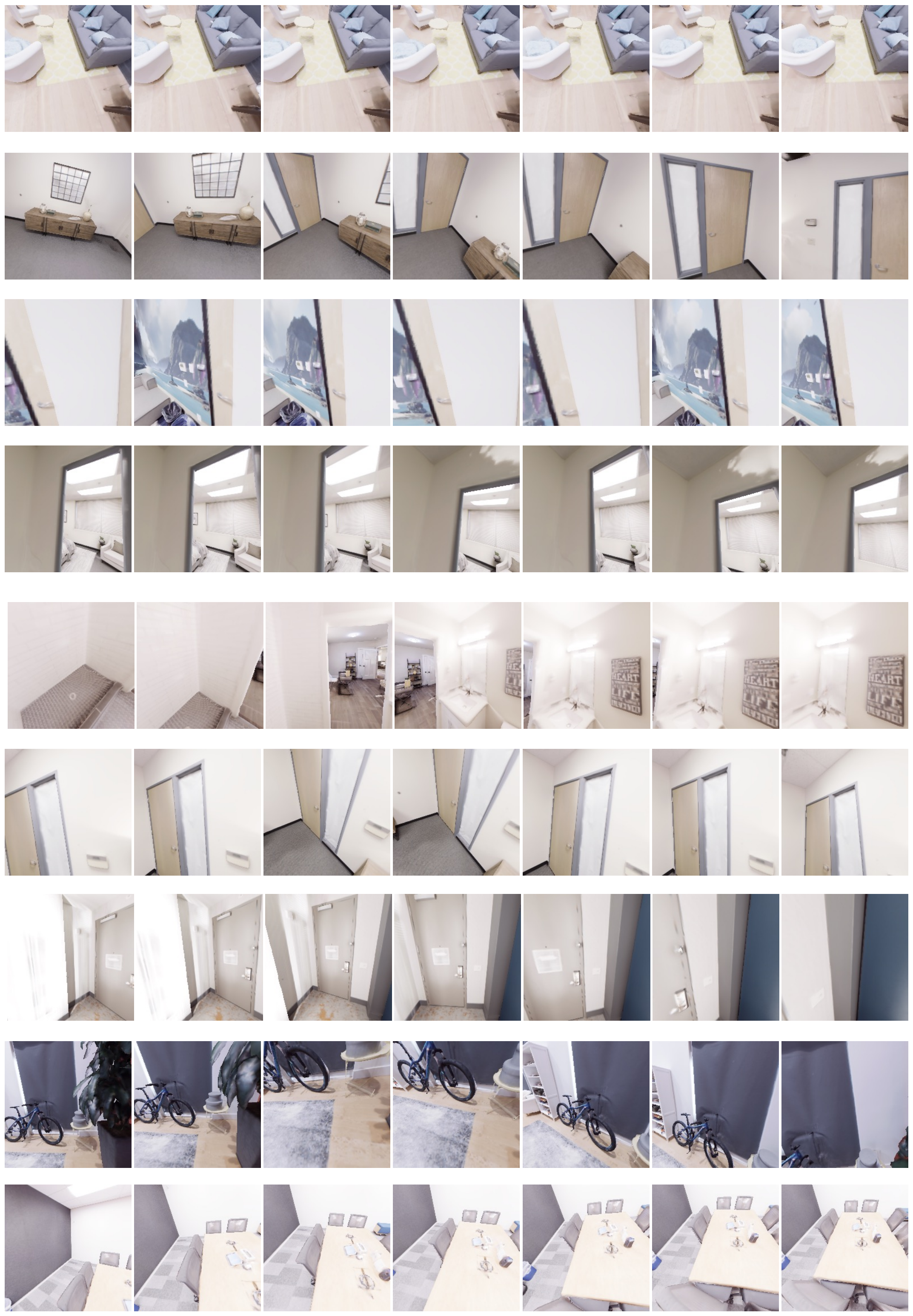

Figure 4. More examples from ARES. Each row represents down-sampled frames from a different sequence.

References

[1] Jiaxi Jiang, Paul Streli, Huajian Qiu, Andreas Fender, Larissa Laich, Patrick Snape, and Christian Holz. AvatarPoser: Articulated full-body pose tracking from sparse motion sensing. In *ECCV*, 2022. 2

[2] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006. 2

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1

[5] Zhengyi Luo, Ryo Hachiuma, Ye Yuan, and Kris Kitani. Dynamics-regulated kinematic policy for egocentric pose estimation. In *NeurIPS*, 2021. 2

[6] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. AMASS: Archive of motion capture as surface shapes. In *ICCV*, 2019. 2

[7] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *ICCV*, 2019. 2

[8] Mesh to SDF, 2020. https://github.com/marian42/mesh_to_sdf. 2

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 1

[10] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 2

[11] 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 *NeurIPS*, 2021. 2

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1

[13] Ye Yuan and Kris Kitani. Ego-pose estimation and forecasting as real-time PD control. In *ICCV*, 2019. 2