Egocentric Global Pose Estimation
Supplementary Material

Jian Wang[1,2] Lingjie Liu[1,2] Weipeng Xu[3] Kripasindhu Sarkar[1,2] Christian Theobalt[1,2]
[1]MPI Informatics [2]Saarland Informatics Campus [3]Facebook Reality Labs
{jianwang,lliu,ksarkar,theobalt}@mpi-inf.mpg.de xuweipeng@fb.com

1. Implementation Details

1.1. Sequential VAE

The input pose sequence with n frames is firstly reshaped to $(3 \times 15, n)$ and fed into the encoder with 45 input channels. The encoder has five 1D conv blocks with 64, 64, 128, 256 and 512 output channels. Each conv block contains one 1D conv layer (kernel size=3, stride=1 and padding=1), one batch norm layer and one leaky relu layer with negative slope=0.01. The output of the encoder is sent into two linear layers giving $\mu, \sigma \in \mathbb{R}^{2048}$. The latent vector z is sampled with μ, σ with the reparameterization trick.

For the decoder, the sampled latent vector z is firstly fed into a linear layer with output dimension $n \times 512$, and five 1D de-conv blocks with 256, 128, 64, 64 and 64 output channels. Each block contains one 1D de-conv layer, one batch norm layer and one leaky relu layer with the same hyper-parameters as the encoder. The output vector with 45 channels is obtained from a final conv layer (kernel size=3, stride=1 and padding=1). The output vector is eventually reshaped to $(n, 15, 3)$, representing a pose sequence as the input.

During training, the weight of reconstruction loss and KL divergence loss is set to 1 and 5×10^{-3} respectively. The code will be released in the future.

1.2. Optimization Details

In local and global pose optimization frameworks, we optimize the latent vector z by using a PyTorch implementation and the Limited-memory BFGS optimizer (L-BFGS) [3] with strong Wolfe line search. We use a learning rate of 2.0 with 30 maximum iterations. We initialise z using the results of the single-frame egocentric pose estimation network $z = f_{enc}(\mathcal{P}_{seq})$. After optimization, the output pose sequence is reconstructed from the optimized z with a VAE decoder $f_{dec}(z)$.

For each long sequence, we firstly split it into several overlapping segments with length B and process each segment independently. After two adjacent segments are processed, we merge the overlapping part between these segments in a linear combination way. For a segment with length $B = 10$, the local pose optimizer, running on 10-frame segments, takes 120.0 ms per segment while the global pose optimizer takes 75.7 ms per segment. Our optimization process is time-efficient thanks to the simple VAE network and GPU-based optimization algorithm. All aforementioned time is measured on a computer with Xeon 6144 CPU and Tesla V100 GPU.

2. Details and Comparisons of the Test Dataset

Our test set has 12200 frames in total, which consists of 5 sequences of 2 actors wearing different clothes performing 13 types of actions (see Table 2 and Table 3). Compared with the Mo^{2}Cap^2 test set and the xR-egopose test set (unreleased), our test set contains more types of actions and more data with global motions. The Mo^{2}Cap^2 test set contains 5591 frames (2 actors performing 8 types of actions). The xR-egopose test set has 10k frames (3 actors performing 6 types of actions).

3. Comparisons on Different Types of Motions

In Table 1 of our main paper, we show the quantitative comparison between our method and the state-of-the-art methods: Mo^{2}Cap^2 and xR-egopose. In order to further compare the performance on different types of motions, we show the quantitative comparison on Mo^{2}Cap^2 [6] in Table 1. The comparisons on different motions of our test dataset is shown in Table 2 and Table 3. In these tables, we also show the BA-MPJPE results of the smoothed global pose of Mo^{2}Cap^2 and xR-egopose to give a fair comparison. In the aforementioned results, our method outperforms all of the baselines on every type of motion.

4. The Structure of RNN-based VAEs

In the Sec. 4.4 of the main paper, we compared the performance of our CNN-based sequential VAE with the MLP-based VAE and RNN-based VAEs in VIBE [1] and MEVA [2]. We will describe the implementation details of the aforementioned VAEs in this section.

Method	walking	sitting	crawling	crouching	boxing	dancing	stretching	waving	total (mm)
Mo^{2}Cap^2+SLAM	38.41	70.94	94.31	81.90	48.55	55.19	99.34	60.92	61.40
Mo^{2}Cap^2+SLAM+Smooth	37.35	64.45	87.41	69.68	45.19	54.76	90.89	49.41	58.25
Mo^{2}Cap^2+Ours	**35.39**	**60.83**	**75.45**	**63.15**	**40.14**	**53.05**	**84.96**	**44.10**	**52.90**
xR-egopose+SLAM	39.69	63.64	64.90	61.22	47.87	58.37	84.64	53.99	55.43
xR-egopose+SLAM+Smooth	38.68	63.20	63.84	60.49	46.53	57.20	84.19	52.58	54.03
xR-egopose+Ours	**33.66**	**60.34**	**62.33**	**55.67**	**44.24**	**51.29**	**82.63**	**46.81**	**50.52**

Table 1. The BA-MPJPE of different types of motions on the indoor sequence of Mo2Cap2 dataset [6]. When based on the local poses estimated by Mo^{2}Cap^2, our approach improves the Mo^{2}Cap^2 [6] results by 13.8% (8.5 mm); when based on the local poses estimated by xR-egopose [5], our method improves the xR-egopose results by 8.9% (4.9 mm).

Method	Mo^{2}Cap^2 +SLAM	Mo^{2}Cap^2 +SLAM +Smooth	Mo^{2}Cap^2 +Ours
walking	69.68	66.68	**57.30**
running	77.88	74.14	**66.78**
crouching	63.28	60.76	**56.05**
boxing	79.37	75.59	**67.57**
dancing	82.65	76.88	**61.43**
stretching	117.7	114.9	**107.5**
waving	53.14	49.31	**42.77**
playing balls	60.95	57.69	**53.30**
open door	55.88	53.33	**46.27**
play golf	113.8	104.4	**94.17**
talking	53.93	50.65	**48.16**
shooting arrow	67.07	62.82	**57.58**
sitting	83.24	78.70	**50.89**
total (mm)	74.46	70.84	**62.07**

Table 2. The BA-MPJPE of different types of motions on our test set. When based on Mo^{2}Cap^2 [6], our approach outperforms Mo^{2}Cap^2 results by 16.6% (12.4 mm).

Method	xR-egopose +SLAM	xR-egopose +SLAM +Smooth	xR-egopose +Ours
walking	84.20	82.96	**60.72**
running	76.78	74.43	**64.92**
crouching	96.86	96.53	**75.11**
boxing	85.74	83.67	**63.45**
dancing	94.23	92.42	**64.78**
stretching	119.9	119.7	**116.3**
waving	72.66	71.83	**46.38**
playing balls	95.30	93.94	**58.49**
open door	71.70	70.80	**45.86**
play golf	94.41	92.58	**83.25**
talking	78.10	75.84	**46.90**
shooting arrow	76.75	74.82	**57.86**
sitting	69.10	63.89	**55.97**
total (mm)	87.20	84.70	**64.31**

Table 3. The BA-MPJPE of different types of motions on our test set. When based on xR-egopose [5], our method outperforms xR-egopose results by 26.2% (22.9 mm).

RNN-Based VAE in VIBE The VIBE [1] explored the performance of RNN-based VAE as a loss term in the training of the VIBE network. At time step t, the body pose $\mathcal{P}_t$ with shape (15, 3) is firstly flattened and put in the encoder. The encoder gives the $\mu_t, \sigma_t \in \mathbb{R}^{2048}$ and the latent vector z_t is sampled from them. The latent vector z_t is put into the decoder and reconstructs the body pose $\mathcal{P}_t$ at time step t. The encoder and decoder are two-layer GRU networks with 512 hidden dimensions.

RNN-Based VAE in MEVA The structure of RNN-based VAE is shown in [2] and the code is released in https://github.com/ZhengyiLuo/MEVA. We directly use their implementation in our experiment.

Note that the structure of RNN-based VAE in MEVA is different from the VAE in VIBE. The VAE in VIBE get different μ and σ for each time step and use the different latent vector z as the decoder input for each time step. In the VAE of MEVA, the latent vector is obtained with a pooling layer and works as the first input of the RNN-based decoder.

MLP-Based VAE The input pose sequence with n frames is firstly reshaped to a vector with length $n \times 15 \times 3$ and fed into the encoder with $n \times 15 \times 3$ input dimensions. The encoder has five 1D fully connected blocks with 512, 512, 1024, 2048 and 2048 output dimensions. Each fully connected block contains one fully connected layer, one batch norm layer and one leaky relu layer with negative slope=0.01. The output of the encoder is sent into two linear layers giving $\mu, \sigma \in \mathbb{R}^{2048}$. The latent vector z is sampled with μ, σ with the reparameterization trick.

For the decoder, the sampled latent vector z is firstly fed into a linear layer with 2048 output dimension, and five 1D fully connected blocks with 2048, 2048, 1024, 512 and 512 output channels. Each block contains one fully connected layer, one batch norm layer and one leaky relu layer with the same hyper-parameters as the encoder. The output vector is obtained from a final fully connected layer. The output vector is eventually reshaped to $(n, 15, 3)$, representing a pose sequence as the input.

5. Design Choices of Our Method

In this section, we explore some design choices of our method.

With VAE Prior Loss Different from previous optimization schemes based on priors captured by VAE [4] or normalizing flow [7], we do not use any prior error $E_{prior} = \|z\|_2$ to maximize the probability of the latent vector in the Gaussian-distributed latent space. That is because such a prior error encourages the latent vector z closer to 0, which would make the pose stay close to a single mean pose, thus incurring unnecessary errors. To validate this analysis, we add the prior error with several different weights w_{vae} in our energy function and show the MPJPEs in the 3rd to 5th row of Table 4. From the experimental result, we can see that all the three errors rise as we increase the prior weight and that the errors converge to our proposed method when the prior weight approaches 0. This verifies our claim that the VAE prior error is harmful to our optimization algorithm.

Optimize P_{seq} In our optimization algorithm, we optimize the latent vector of VAE z and get the final prediction P_{seq} with the VAE decoder f_{dec}. An alternative optimization strategy is to optimize pose sequence P_{seq} directly, calculate the latent vector z with VAE encoder f_{enc} and incorporate prior term with $E_{prior} = \|z\|_2$. To compare these approaches, we report the direct optimization result in the 6th row of Table 4. It shows a better result is achieved when the optimization is performed in the latent space, which is consistent with the conclusion [7] in previous research.

Method	Global MPJPE	PA-MPJPE	BA-MPJPE
Mo^{2}Cap^2	141.8	102.3	74.46
w_{vae}=1e-3	176.3	105.2	70.42
w_{vae}=5e-4	136.5	89.20	64.86
w_{vae}=1e-5	121.1	83.40	62.53
optimize P_{seq}	128.1	92.32	68.10
Ours	**119.5**	**82.06**	**62.07**

Table 4. The quantitative results of different design choices.

6. Limitation of Our Method

As a common limitation for the SLAM methods, our global camera pose estimation requires an environment with rich visual features. Featureless scenes such as white walls and green screens can lead to unreliable camera poses. Although our method fails in the featureless background, it performs well for real-life scenarios as shown in the supplementary video. We encourage the reader to watch our supplementary video (starting from #5:55) for our global pose estimation for various motions and daily life activities in various environments.

References

[1] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. VIBE: video inference for human body pose and shape estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5252–5262, 2020. 1, 2

[2] Zhengyi Luo, S. Alireza Golestaneh, and Kris M. Kitani. 3d human motion estimation via motion compression and refinement. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020. 1, 2

[3] Jorge Nocedal and Stephen Wright. *Numerical optimization*. 2006. 1

[4] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10975–10985, 2019. 3

[5] Denis Tomè, Patrick Peluse, Lourdes Agapito, and Hernán Badino. xr-egopose: Egocentric 3d human pose from an HMD camera. In *IEEE International Conference on Computer Vision*, pages 7727–7737, 2019. 2

[6] Weipeng Xu, Avishek Chatterjee, Michael Zollhöfer, Helge Rhodin, Pascal Fua, Hans-Peter Seidel, and Christian Theobalt. Mo^{2}cap^2: Real-time mobile 3d motion capture with a cap-mounted fisheye camera. *IEEE Trans. Vis. Comput. Graph.*, 25(5):2093–2101, 2019. 1, 2

[7] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, William T. Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Weakly supervised 3d human pose and shape reconstruction with normalizing flows. In *European Conference on Computer Vision*, volume 12351, pages 465–481, 2020. 3