Cross-Attention of Disentangled Modalities for 3D Human Mesh Recovery with Transformers
— Supplementary Material —

Junhyeong Cho[1] Kim Youwang[2] Tae-Hyun Oh[2,3]

[1]Department of CSE [2]Department of EE [3]Graduate School of AI
Pohang University of Science and Technology (POSTECH), Korea
{junhyeong99, youwang.kim, taehyun}@postech.ac.kr
https://github.com/postech-ami/FastMETRO

In this supplementary material, we present more implementation details (Section A), quantitative evaluations (Section B) and qualitative evaluations (Section C), which are not included in the main paper due to its limited space.

A Implementation Details

PyTorch [14] is used to implement our FastMETRO. We employ ResNet-50 [4] or HRNet-W64 [15] as our CNN backbone, where each backbone is initialized with ImageNet [2] pre-trained weights. For the initialization of our transformer, we use Xavier Initialization [3]. Given an input image of size $224 \times 224 \times 3$, our CNN backbone produces the image features $\mathbf{X}_I \in \mathbb{R}^{H \times W \times C}$, where $H = W = 7$ and $C = 2048$. The hidden dimension size of the camera token is $D = 512$, which is also same for each joint token $\mathbf{t}_i^J$ and vertex token $\mathbf{t}_j^V$. Following [8, 9], the number of joint tokens is $K = 14$ and that of vertex tokens is $N = 431$. The number of vertices in the fine mesh output $\hat{\mathbf{V}}'_{3D}$ is $M = 6890$, which is same with that of SMPL [10]. The number of heads in multi-head attention modules is 8. We employ two linear layers with a ReLU activation function for the MLPs in our transformer layers. Regarding the 3D coordinates regressor or camera predictor, we use a linear layer. To retain spatial information for the flattened image features $\mathbf{X}_F$, we use fixed sine positional encodings as in [1]. Note that the pre-computed matrix for mesh upsampling and the adjacency matrix for attention masking are sparse matrices in our implementation; only about $25K$ elements are non-zeros in the upsampling matrix and about $3K$ elements are non-zeros in the adjacency matrix. We leverage these sparse matrices for memory-efficient implementation.

We use AdamW optimizer [11] with the learning rate of 10^{-4}, weight decay of 10^{-4}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For stable training of our transformer, we apply gradient clipping and set the maximal gradient norm value to 0.3. The loss coefficients are $\lambda_{3D}^V = \lambda_{2D}^J = 100$ and $\lambda_{3D}^J = 1000$. We train our FastMETRO with a batch size of 16 for 60 epochs, which takes about 4 days on 4 NVIDIA V100 GPUs (16GB RAM). Note that METRO [8] and Mesh Graphormer [9] are trained with a batch size of 32 for 200 epochs, which takes about 5 days on 8 NVIDIA V100 GPUs (32GB RAM). During training, we apply the standard data augmentation for this task as in [6, 8, 9, 13].

Table B1. Ablation study of our FastMETRO on Human3.6M [5]. The effects of different components are evaluated. The default model is FastMETRO-S-R50.

Model	#Params	Human3.6M	
		MPJPE ↓	PA-MPJPE ↓
w/ baseline setting	51.9M	59.0	41.8
w/ learnable positional encodings	32.8M	56.2	39.7
w/ camera token in transformer decoder	**32.7M**	56.1	39.5
w/ camera prediction using estimated mesh	32.8M	58.0	39.6
FastMETRO–S–R50	**32.7M**	**55.7**	**39.4**

B Quantitative Evaluations

Baseline. To validate the effectiveness of our method, we evaluate a naïve transformer encoder-decoder architecture. Following encoder-based methods [8,9], we employ a 3D human mesh for input tokens and learnable layers for mesh upsampling, and do not perform attention masking. For the construction of joint and vertex tokens, we linearly project the 3D coordinates of joints and vertices in the human mesh. To simplify this model, we do not progressively reduce hidden dimension sizes in the transformer. This baseline shows lower regression accuracy as shown in the first row of Table B1, although it demands much more parameters. This demonstrates that our FastMETRO effectively improves the accuracy and reduces the number of parameters required in the architecture.

Positional Encodings. Following [1], we employ fixed sine positional encodings for retaining spatial information in our transformer. When we use learnable embeddings as positional encodings, we obtain similar results but this demands more parameters as shown in the second row of Table B1.

Weak-Perspective Camera Parameters. We estimate these parameters using a camera token in our transformer encoder. On the other hand, GraphCMR [7] estimates the camera parameters from the vertex features obtained by graph convolutional layers, and encoder-based transformers [8,9] predict the camera parameters from the 3D coordinates of mesh vertices estimated by transformer layers. As shown in the third and fourth rows of Table B1, we also evaluate our FastMETRO using different methods to predict the camera parameters. When we employ a camera token in our transformer decoder, we obtain similar results. When we predict the camera parameters using the 3D mesh estimated by transformer layers, the regression accuracy drops.

C Qualitative Evaluations

Comparison with Encoder-Based Transformers. Figure C1 shows the qualitative comparison of transformer encoders [8,9] with our method. As shown in Figure C1, FastMETRO-L-H64 achieves competitive results, although our model requires only about 25% of the parameters in the transformer architecture compared with the encoder-based transformers. Note that FastMETRO captures more detailed body pose especially for knees and ankles.

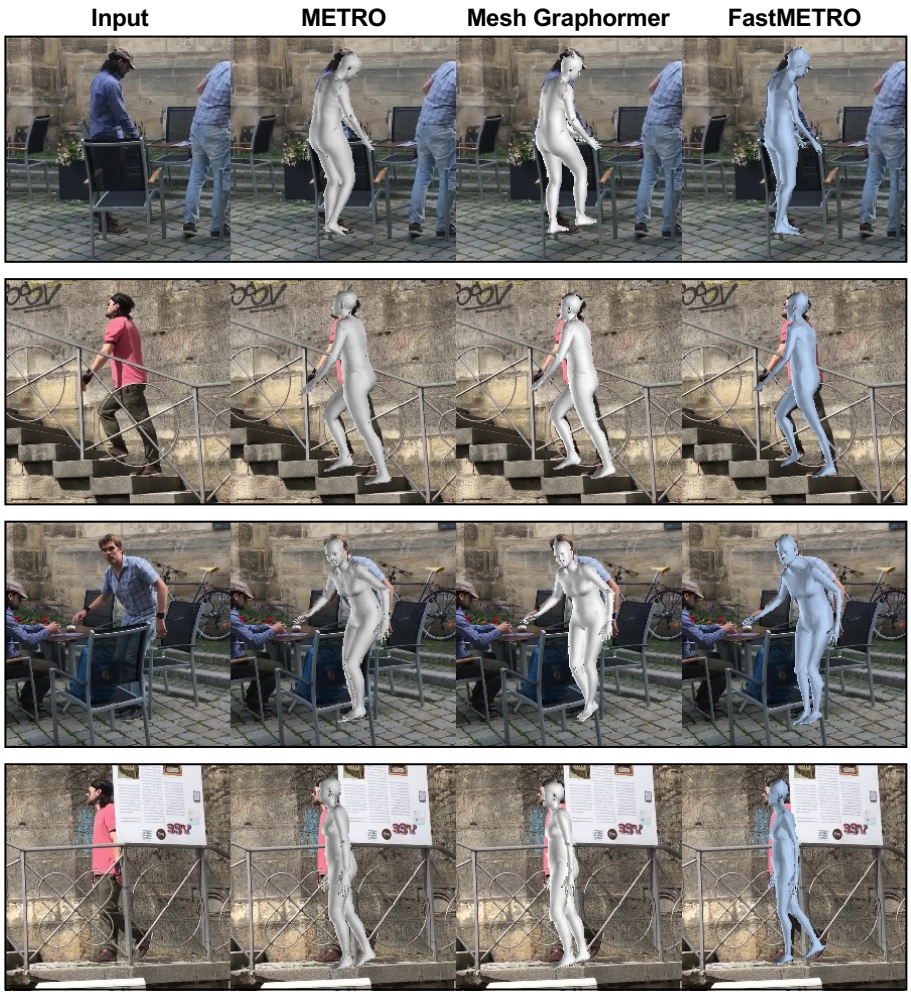

Fig. C1. Comparison with encoder-based transformers [8, 9] and FastMETRO-L-H64 on 3DPW [12]. Our model achieves competitive results using much fewer parameters, and shows more favorable body pose especially for knees and ankles.

SMPL Parameters from Estimated Mesh. Following [7,13], we can optionally regress SMPL [10] parameters from the output mesh estimated by our model. To be specific, we first regress the 3D coordinates of human mesh vertices via our model, then predict SMPL pose and shape coefficients via a SMPL parameter regressor which takes the estimated 3D mesh vertices as input. Following [7, 13], we employ fully connected layers with skip connections as the SMPL parameter regressor. In this way, we can reconstruct 3D human mesh using the predicted SMPL parameters. Figure C2 shows the visualization of the estimation results obtained by our FastMETRO and the SMPL parameter regressor.

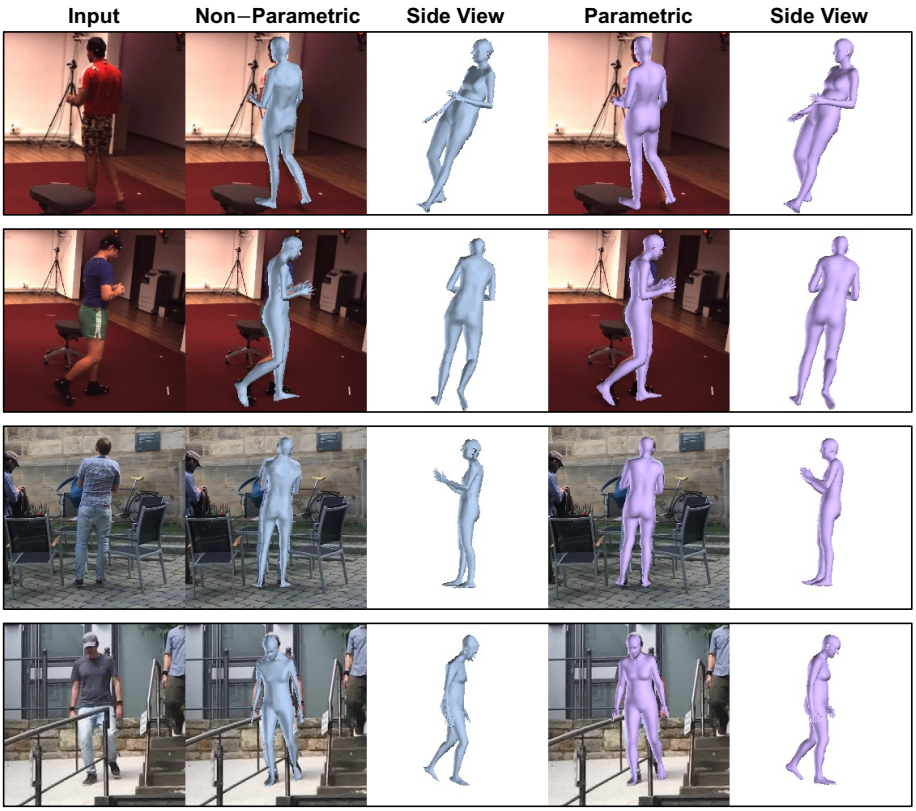

Fig. C2. Qualitative results of FastMETRO-L-H64 on Human3.6M [5] and 3DPW [12]. We can optionally learn to regress SMPL [10] parameters from the 3D coordinates of mesh vertices estimated by our FastMETRO.

References

1. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-End Object Detection with Transformers. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 213–229 (2020) 1, 2
2. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009) 1
3. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 249–256 (2010) 1
4. He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian: Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016) 1
5. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments.

IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **36**(7), 1325–1339 (2014) 2, 4
6. Kolotouros, N., Pavlakos, G., Black, M.J., Daniilidis, K.: Learning to Reconstruct 3D Human Pose and Shape via Model-fitting in the Loop. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019) 1
7. Kolotouros, N., Pavlakos, G., Daniilidis, K.: Convolutional Mesh Regression for Single-Image Human Shape Reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 2, 3
8. Lin, K., Wang, L., Liu, Z.: End-to-End Human Pose and Mesh Reconstruction with Transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1954–1963 (2021) 1, 2, 3
9. Lin, K., Wang, L., Liu, Z.: Mesh Graphormer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 12939–12948 (2021) 1, 2, 3
10. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A Skinned Multi-Person Linear Model. ACM Transactions on Graphics (SIGGRAPH Asia) **34**(6), 248 (2015) 1, 3, 4
11. Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization. In: International Conference on Learning Representations (ICLR) (2019) 1
12. Marcard, T.v., Henschel, R., Black, M.J., Rosenhahn, B., Pons-Moll, G.: Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018) 3, 4
13. Moon, G., Lee, K.M.: I2L-MeshNet: Image-to-Lixel Prediction Network for Accurate 3D Human Pose and Mesh Estimation from a Single RGB Image. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020) 1, 3
14. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: Advances in Neural Information Processing Systems (NIPS) (2017) 1
15. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., Xiao, B.: Deep High-Resolution Representation Learning for Visual Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2019) 1