Supplementary Material: GLAMR: Global Occlusion-Aware Human Mesh Recovery with Dynamic Cameras

Ye Yuan[2*] Umar Iqbal[1] Pavlo Molchanov[1] Kris Kitani[2] Jan Kautz[1]
[1]NVIDIA [2]Carnegie Mellon University
https://nvlabs.github.io/GLAMR

1. Details for the Datasets

AMASS [9] is a large human motion database with 11000+ human motions. We use AMASS to train and evaluate the motion infiller and trajectory predictor. Specifically, we use the Transitions, SSM, and HumanEva [11] subsets for testing and all other subsets for training.

3DPW [15] is an in-the-wild human motion dataset that consists of 60 videos recorded with dynamic cameras in diverse environments. The GT 3D poses are obtained using wearable IMU sensors. Since non-optical sensors are used to obtain GT data, the dataset also provides body pose information when the persons go outside the FoV of the camera. 3DPW also provides the global trajectories of people in the dataset. However, the global trajectories are quite inaccurate since they are estimated from IMU data. Therefore, we do not use 3DPW to evaluate global trajectory reconstruction in the paper. Since we do not use 3DPW for training, we use sequences from the entire 3DPW dataset for visualization. We use the official 3DPW test split to report quantitative results in the paper.

Dynamic Human3.6M is a new benchmark for global human pose estimation with dynamic cameras that we create from the Human3.6M dataset [4]. We simulate dynamic cameras and occlusions by cropping each frame with a view window of 300×600 that horizontally oscillates around the person's bounding box center with a period of 4.8 seconds and a magnitude of 200 pixels. In this way, we synthesize large camera motions and severe occlusions where the person is occluded for almost half of the time, which makes it very challenging for existing 3D human pose and shape estimation methods. Additionally, since Human3.6M provides accurate global human trajectories and human poses, we use Dynamic Human3.6M to evaluate global trajectory reconstruction and pose estimation for occluded frames. We follow the standard protocol [6] and use the official test split (subjects 9 and 11) for evaluation. Please refer to the supplementary video for an example sequence of the Dynamic Human3.6M dataset. Code for generating Dynamic Human3.6M are available here for users who have downloaded the original Human3.6M dataset [4].

2. Implementation Details for Preprocessing

3D Multi-Object Tracking and Re-identification. We use DeepSORT [16] with ResNet-50 [3] in the MMTracking package [1] for 3D multi-object tracking (MOT) and re-identification. We use the GT tracks to evaluate our approach and the baselines, following the standard protocol for human pose estimation.

Initial Human Pose and Shape Estimation. As mentioned in the main paper, we use KAMA [5] or SPEC [8] to provide the initial human pose and shape estimation from the bounding boxes extracted by 3D MOT. We choose these two methods since both KAMA and SPEC estimate 3D human poses in the camera coordinates with absolute root translations, while many state-of-the-art human pose estimation methods do not provide the root translations. We also use HRNet [12] to extract 2D human keypoints from the video, which are used in the proposed global optimization framework.

*Work done during an internship at NVIDIA.

3. Implementation Details for Generative Motion Infiller

Network Architecture. The detailed network architecture of the CVAE-based generative motion infiller is outlined in Fig. 1. For all the Transformer [14] modules, the dimensions for keys, queries, and values are set to 256, the number of transformer blocks is 2, the hidden dimensions of the feedforwards layers are 512, the dropout rate is 0.1, and 8 heads are used for the multi-head attention. The time-based encoding takes the same sinusoidal form as the original positional encoding [14] but replaces the position with the time index. We use two hidden layers (512, 256) with ReLU activations for all the token-wise MLPs. In the prior network, two learnable tokens are used to form queries to produce the mean $\boldsymbol{\mu}_{\boldsymbol{z}}^{p}$ and standard deviation $\boldsymbol{\sigma}_{\boldsymbol{z}}^{p}$ of the prior distribution of the latent code $\boldsymbol{z}$. Similarly, in the posterior network, two learnable tokens are appended to the GT pose sequence $\tilde{\boldsymbol{\Theta}}'$ to output the mean $\boldsymbol{\mu}_{\boldsymbol{z}}^{q}$ and standard deviation $\boldsymbol{\sigma}_{\boldsymbol{z}}^{q}$ of the posterior distribution of the latent code $\boldsymbol{z}$.

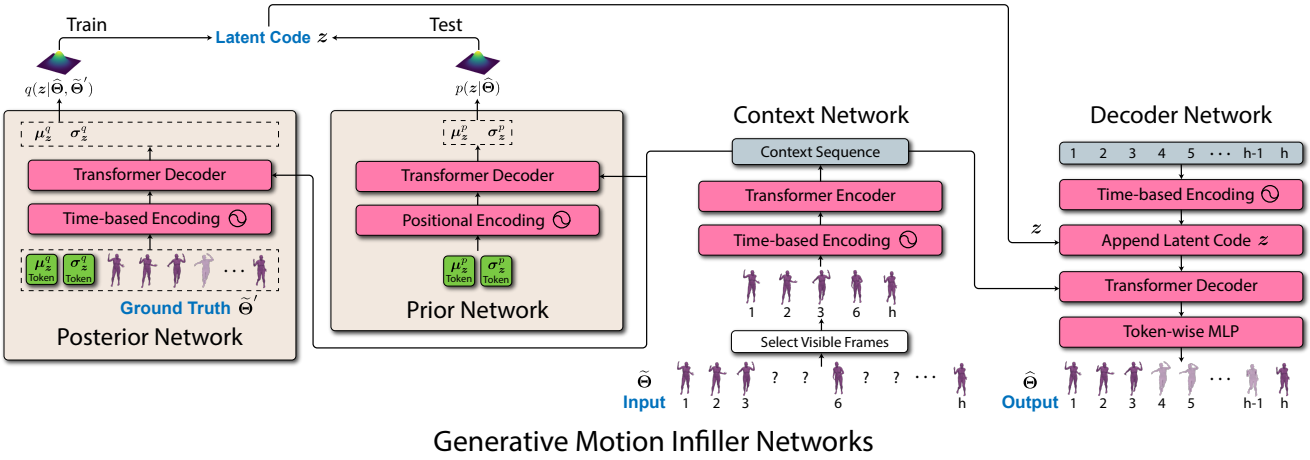

Figure 1. The detailed network architecture of the CVAE-based generative motion infiller. For all the Transformer modules, the dimensions for keys, queries, and values are set to 256, the number of transformer blocks is 2, the hidden dimensions of the feedforwards layers are 512, the dropout rate is 0.1, and 8 heads are used for the multi-head attention. Two hidden layers (512, 256) with ReLU activations are used for all the token-wise MLPs.

Hyperparameters and Training. The dimension of the latent code $\boldsymbol{z}$ is 128. The sliding window size h of the autoregressive motion infilling is 50. Both the number of context frames h_{c} and the number of look-ahead h_{1} frames are 10. When synthesizing occluded motions, for any GT training motion of $h = 50$ frames, we randomly occlude H_{occ} consecutive frames of motion where H_{occ} is uniformly sampled from $[10, 40]$. Note that we do not occlude the first $h_{\text{c}} = 10$ frames which are reserved as context. The KL divergence term in Eq. (2) of the main paper uses a weighting factor of 0.001. We train the networks for 2000 epochs with a batch size of 1024 where each epoch uses a total of 10 million frames of motion. For optimization, we use the Adam optimizer [7] with a learning rate of 0.001 and clip the gradient if its norm is larger than 5. We use PyTorch [10] to implement and train the networks.

4. Implementation Details for Global Trajectory Predictor

Heading Coordinate and Egocentric Trajectory Representation. The heading vector of a person points towards where the person is facing and is parallel to the ground. We obtain the heading vector by aligning the z-axis of the person's root coordinate with the world z-axis and use the resulting y-axis of the aligned root coordinate as the heading vector. This way of obtaining the heading is more stable than using the yaw of the Euler angle representation, which suffers from singularities and can be quite unstable. The heading coordinate is defined by first placing the world coordinate at the root position of the person and then rotating the world coordinate around the z-axis (vertical) to align the y-axis with the heading vector. By definition, representing and predicting human trajectories in the heading coordinate allows the predicted trajectory to be invariant of the person's absolute xy translation and heading. In the egocentric trajectory representation $\boldsymbol{\psi}_t = (\delta x_t, \delta y_t, z_t, \delta\phi_t, \boldsymbol{\eta}_t)$, we use absolute height z_t since the height of a person relative to the ground does not vary a lot and is highly correlated with the body motion of the person. For the local rotation $\boldsymbol{\eta}_t$, we adopt the 6D rotation representation [17] to avoid discontinuity.

Network Architecture. The detailed network architecture of the CVAE-based global trajectory predictor is illustrated in Fig. 2. We use two bidirectional LSTM layers with hidden dimension 256 for all the LSTM blocks in the networks. We use

two hidden layers (512, 256) with ReLU activations for all the token-wise MLPs. For the input poses, we first convert them to 3D joint positions using the SMPL joint function without global rotations and translations. This is because we find that using 3D joint positions leads to better performance than using joint rotations directly. In both the prior and posterior networks, token-wise mean pooling is used to produce a single feature from a sequence of tokens, which is then used to produce the parameters of the prior or posterior distribution of the latent code $\boldsymbol{v}$.

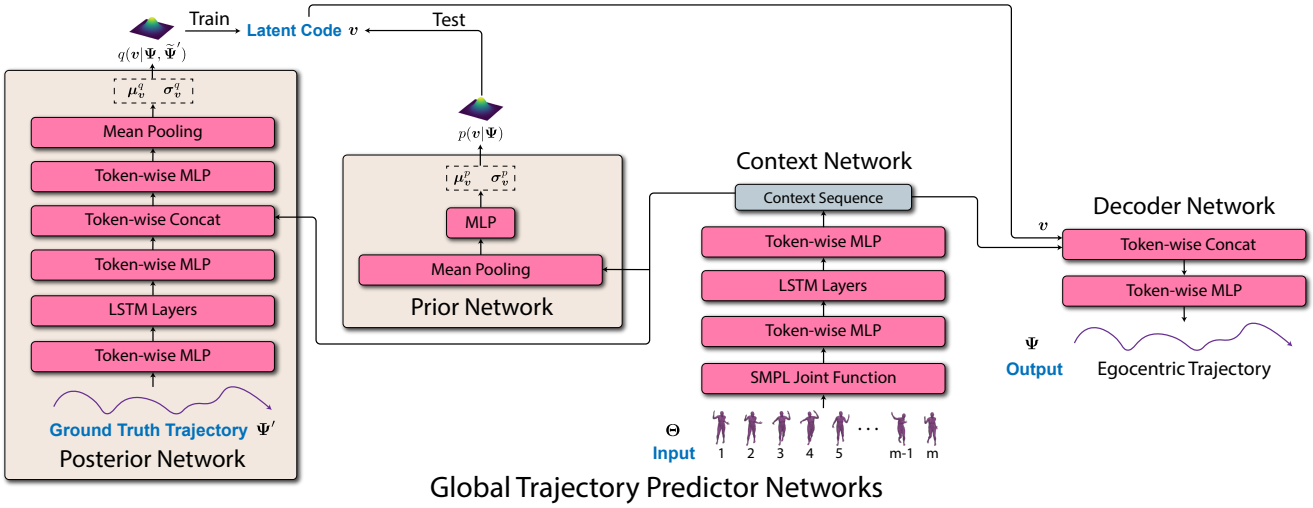

Figure 2. The network architecture of the CVAE-based global trajectory predictor. We use two bidirectional LSTM layers with hidden dimension 256 for all the LSTM blocks, and we use two hidden layers (512, 256) with ReLU activations for all the token-wise MLPs. Token-wise mean pooling is used in the prior and posterior networks to summary sequences into a single feature.

Hyperparameters and Training. The dimension of the latent code $\boldsymbol{v}$ is 128. The KL divergence term in Eq. (8) of the main paper uses a weighting factor of 0.001. We train the networks for 2000 epochs with a batch size of 256 where each epoch uses a total of 2 million frames of motion. The training sequence length is 100 frames For optimization, we use the Adam optimizer [7] with a learning rate of 0.0001 and clip the gradient if its norm is larger than 5. We use PyTorch [10] to implement and train the networks.

5. Implementation Details for Global Optimization

Initialization. We initialize the egocentric trajectories using the output from the global trajectory predictor. For the camera, we approximate the camera intrinsic parameters $\boldsymbol{K}$ using the dimensions of the image $[\mathtt{w}, \mathtt{h}]$ where we assume the principal point is at the image center $[\mathtt{w}/2, \mathtt{h}/2]$. Note that the camera intrinsics are kept fixed during the optimization process. For the camera extrinsic parameters $\boldsymbol{C}$, we initialize them from the persons' global trajectories using the following equations:

$$\boldsymbol{C}_t = \Omega\left(\frac{1}{\sum_{i=1}^{N} V_t^i}\sum_{i=1}^{N} V_t^i \cdot \boldsymbol{P}_t^{i,\mathtt{global}}\boldsymbol{P}_t^{i,\mathtt{cam}^{-1}}\right), \tag{1}$$

where V_t^i is the visibility of person i at frame t, $\boldsymbol{P}_t^{i,\mathtt{global}} \in \mathbb{R}^{4\times4}$ is the person's transformation in the global coordinates based on the predicted global trajectory $(\widehat{\boldsymbol{T}}^i, \widehat{\boldsymbol{R}}^i)$, $\boldsymbol{P}_t^{i,\mathtt{cam}} \in \mathbb{R}^{4\times4}$ is the person's transformation in the camera coordinates based on the estimated trajectory $(\widetilde{\boldsymbol{T}}^i, \widetilde{\boldsymbol{R}}^i)$ by the pose estimator (e.g., KAMA [5]), Ω is a projection operator that projects the matrix into a valid transformation. If no person is visible at frame t, the camera extrinsics $\boldsymbol{C}_t$ is initialized to the camera extrinsics of the most recent frame with visible people. Eq. (1) is the least squares solutions of the following (transposed) linear systems:

$$\boldsymbol{P}_t^{i,\mathtt{global}} = \boldsymbol{C}_t\boldsymbol{P}_t^{i,\mathtt{cam}}, \qquad \forall i, V_t^i = 1\,. \tag{2}$$

Hyperparameters and Optimization. The optimization loss coefficients $(\lambda_{\mathtt{2D}}, \lambda_{\mathtt{traj}}, \lambda_{\mathtt{reg}}, \lambda_{\mathtt{cam}}, \lambda_{\mathtt{pen}})$ are set to (1, 100000, 100, 10000, 100000) for 3DPW and (1, 100000, 100, 10000, 0) for Human3.6M. We do not use the inter-person penetration loss for Human3.6M since it only has one person in each video. The trajectory regularization weighting factor $\boldsymbol{w}_\psi$ is set to

(3,10,10000,5,10000) for each element in the egocentric trajectory $\boldsymbol{\psi}_t = (\delta x_t, \delta y_t, z_t, \delta \phi_t, \boldsymbol{\eta}_t)$, where we use large weights to penalize changes in height z_t and local rotation $\boldsymbol{\eta}_t$. The global optimization is also implemented in PyTorch [10], where we use the Adam optimizer [7] with a learning rate of 0.001 to optimize the global trajectories and camera extrinsics.

Computation Time. The overall processing time for a 1-min scene is around 5 mins with 500 optimization iterations, which is much faster than using OpenSfM ($>$ 30 mins).

6. Evaluation of Global Optimization on 3DPW

We also perform experiments on 3DPW with and without our global optimization framework to study the importance of global optimization when there are multiple people in the video. Although 3DPW does not provide accurate GT human trajectories in the global coordinates, the relative translations and rotations between people in 3DPW are quite accurate. Therefore, we compute the relative translations and rotations between pairs of humans and calculate their errors w.r.t. the ground truth. These metrics, *i.e.*, relative translation and rotation errors, serve as an alternative way to evaluate global reconstruction quality. As shown in Table 1, using global optimization can greatly reduce the relative translation and rotation errors between humans, which means our global optimization framework can greatly help to reconstruct the spatial relationships of humans in the video.

Method	Relative Translation Error	Relative Rotation Error
Ours w/o Global Optimization	1.92	1.07
Ours (GLAMR)	**0.66**	**0.30**

Table 1. Evaluation of our global optimization framework on 3DPW. We evaluate the relative translation error (in meters) and relative rotation error (in angles) between pairs of humans. Here, "relative" denotes the relative spatial relationship between two humans.

7. Effect of Sliding Window Length.

As shown in Fig. 3, when increasing the window length h (with context h_c and look-ahead h_l being $0.2h$), the reconstruction error increases because it is harder for the latent code $\boldsymbol{z}$ to encode a longer window which contains more motion variations than a shorter window. In the meantime, the sample error first drops and then increases since there is a trade-off: a longer window provides more context for better inference, but it also puts more burden on the latent code as indicated by the increasing reconstruction error.

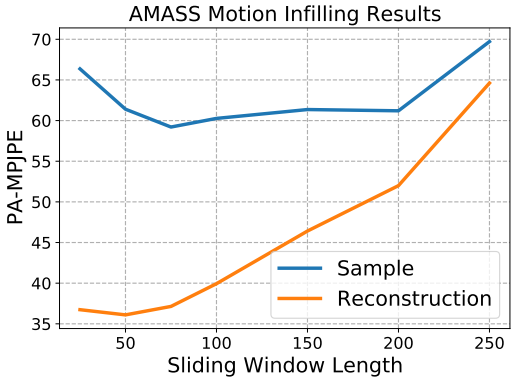

Figure 3. Sample and reconstruction PA-MPJPE *vs.* sliding window length h. The context h_c and look-ahead h_l are always $0.2h$.

Motion Infilling without Visible Pose. In the extreme case, when there is no visible pose ($h_c = h_l = 0$), our motion infiller can still produce plausible motions sampled from the prior learned from the training motion datasets. In this case, the motion infiller essentially becomes an unconditional VAE model.

8. Discussion of Limitations

As the first paper on this new problem, our method has a few limitations that are important for future research to address. First, our approach has five stages that are sequentially dependent. Therefore, errors in early stages can propagate to late stages, which may lead to inaccurate global pose estimation. Future work could integrate these stages together to form an

end-to-end learnable framework. Second, like many works in human mesh recovery, our approach can only recover the SMPL parameters which omit the fine details of human meshes such as clothing. Integrating neural articulated shapes such as [2] into our approach could potentially address this problem. Third, our approach is not real-time due to the batch processing and global optimization. Future work could explore a causal version of our approach where only a small window around the incoming frame is optimized, which could substantially improve computational efficiency. Finally, the generative motion infiller and global trajectory predictor in our approach operate for each person independently. Therefore, the generated motions and trajectories may not capture potentially complex and nuanced interactions between occluded people such as hugging or dancing. Future work could address this limitation by employing new generative models that produce interaction-aware motions of multiple people.

9. Discussion of Potential Negative Impact

With its strong ability to reconstruct global human motions and tackle severe occlusions, our method marks a significant step towards global human mesh recovery in the wild. However, misuse of this technology could lead to potential privacy concerns and the propagation of misinformation. For instance, combined with advanced neural rendering approaches [13], the reconstructed global human motion of our approach could be used to fabricate videos of human actions that are indistinguishable from real ones. To address this issue, future research should continue to study the detection of synthesized videos with realistic human motion.

References

[1] MMTracking Contributors. MMTracking: OpenMMLab video perception toolbox and benchmark. https://github.com/open-mmlab/mmtracking, 2020. 1

[2] Boyang Deng, John P Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa neural articulated shape approximation. In *ECCV*, pages 612–628. Springer, 2020. 5

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[4] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *TPAMI*, 36(7):1325–1339, 2014. 1

[5] Umar Iqbal, Kevin Xie, Yunrong Guo, Jan Kautz, and Pavlo Molchanov. Kama: 3d keypoint aware body mesh articulation. In *3DV*, 2021. 1, 3

[6] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018. 1

[7] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014. 2, 3, 4

[8] Muhammed Kocabas, Chun-Hao P. Huang, Joachim Tesch, Lea Müller, Otmar Hilliges, and Michael J. Black. SPEC: Seeing people in the wild with an estimated camera. In *ICCV*, 2021. 1

[9] 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. 1

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 2, 3, 4

[11] Leonid Sigal and Michael J Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. *Brown Univertsity TR*, 120(2), 2006. 1

[12] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, pages 5693–5703, 2019. 1

[13] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. *arXiv preprint arXiv:2111.05849*, 2021. 5

[14] 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. 2

[15] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *ECCV*, 2018. 1

[16] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, pages 3645–3649. IEEE, 2017. 1

[17] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 2