EventEgo3D: 3D Human Motion Capture from Egocentric Event Streams —Supplementary Material—

Christen Millerdurai[1,2] Hiroyasu Akada[1] Jian Wang[1]
Diogo Luvizon[1] Christian Theobalt[1] Vladislav Golyanik[1]
[1]MPI for Informatics, SIC [2]Saarland University, SIC

This supplementary document first provides detailed information about our datasets in Section 1. Section 2 discusses the architecture of EE3D and the real-time implementation of our framework. Next, we offer in Section 3 a thorough evaluation of our method alongside competing approaches on the test-set of EE3D-S and conduct an ablation study to analyse various dataset training strategies. We also show visualisations of the predictions generated by our method, along with intermediate representations such as human body masks, confidence maps, and 2D joint heatmaps. Please check our video for more visualisations[1].

1. Dataset Details

1.1. EE3D-S (Synthetic)

Given the difficulty of capturing a large amount of training data for our egocentric setting, we create a synthetic dataset, EE3D-S for pre-training our method. We generate EE3D-S by rendering sequences of human motions using a virtual replica of our HMD. For each motion sequence, we first fit the 3D human model SMPL [8] to the egocentric observations of the HMD wearer. Subsequently, we animate the human model by sampling from the CMU MoCap dataset [1]. Next, we obtain RGB frames and human body masks by rendering the scene from the viewport of the virtual HMD. Additionally, the 3D body joint positions of the wearer are obtained by transforming the SMPL body joint positions in the world coordinate frame to the coordinate frame of the HMD. Finally, the rendered RGB frames are passed to VID2E [6] which converts an RGB frame sequence to an event stream, resulting in the EE3D-S dataset. To represent background events within the event stream generation process, we model the background by randomly selecting images from the LSUN dataset [14].

Scene Modelling. Following Xu *et al.* [13], we build our dataset on top of the large-scale synthetic human dataset, SURREAL [12] using Blender [3]. SMPL [8] provides the proxy geometry of the HMD wearer, and body textures are randomly sampled from the texture set provided by the SURREAL dataset. The background is modelled with a $26m^2$ sized plane with textures randomly sampled from the LSUN dataset [14]. We illuminate the scene with four point light sources with random positions within a five-meter radius from the HMD to create variously illuminated scenes.

Human Animation. The motions of the 3D human model are sampled from the CMU MoCap [1] dataset. While generating events, it is essential to ensure that VID2E has enough temporal information from the motion sequence, as highlighted in Gehrig *et al.* [6]. Hence, we sample the motions at high frame rates. Therefore, the SMPL body parameters obtained from SURREAL are linearly interpolated to sample motions at 480Hz.

Rendering and Event Stream Generation. We render the scenes using the fish-eye camera from the virtual replica of the HMD. The position of the fish-eye camera is obtained by offsetting the nose vertex position of the SMPL body aligning it closely with the event camera mounted on the real HMD, i.e. the offset is determined by visually aligning the position of the real event camera with respect to the wearer. To set the intrinsic parameters of the fish-eye camera, we calibrate our real event camera using the omnidirectional camera calibration toolbox OCamCalib [10]. The obtained intrinsic parameters are then set for the virtual fisheye camera. Due to the different head sizes and HMD movements during its operation, the camera position with respect to the head can change slightly. To account for this effect, we add random perturbations to the position of the virtual fisheye camera. We generate the RGB frames and the human body masks using image and mist render layers in Blender's Cycles renderer [3]. The rendered RGB frames are then processed by VID2E [6] to generate the event streams. In total, we synthesise 946 motion sequences with $6.21 \cdot 10^6$ 3D human poses and $1.419 \cdot 10^{11}$ events.

1.2. EE3D-R (Real)

To evaluate our method and reduce the domain gap between synthetic and real scenarios, we collect the EE3D-R dataset.

Dataset composition. The EE3D-R dataset mainly in-

[1]https://www.youtube.com/watch?v=jatNH0s_k_E

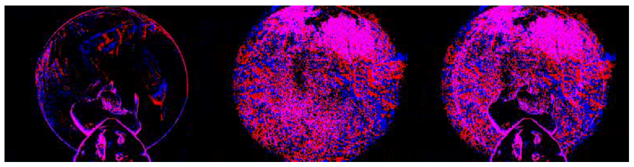


Figure 1. Exemplary events from EE3D-S (left), background events (middle) and background events augmented with the events from EE3D-S (right).

cludes regular body movements, featuring a wide range of (natural and unrestricted) motions and inherent differences in their execution among the participants. We ask twelve subjects—persons with different body shapes and skin tones—to wear our HMD and perform different motions (e.g. fast) in a multi-view studio with 29 RGB cameras recording at 50 fps. Each sequence encompasses the following motion types: Walking, crouching, push-ups, boxing, kicking, dancing, interaction with the environment, crawling, sports and jumping. In the sports category, participants perform specific activities—playing basketball, participating in tug of war and playing golf. Meanwhile, in the interaction with the environment category, the subjects perform actions such as picking up objects from a table, sitting on a chair, and moving the chair. We obtain in total $4.64 \cdot 10^5$ poses spanning $\approx$155 minutes.

Human Body Mask Generation. The human body mask is generated in two steps. First, we track an SMPL mesh on the wearer of the HMD using open-source software EasyMoCap [5]. Subsequently, we take the tracked SMPL mesh, transform it into the local coordinate system of the HMD, and then project it onto the egocentric view.

1.3. Event Augmentation

Motions and data recorded in the multi-view studio would not allow satisfactory generalisation to some in-the-wild scenes with different backgrounds. Hence, we propose an event-wise augmentation technique for the background events and apply it to EE3D-S and EE3D-R; see Fig. 1.

We capture sequences of both outdoor and indoor scenes without humans with a handheld event camera *i.e.,* background event stream. The event-wise augmentation augmentation is done in two steps. First, we convert the background event stream to LNES frames, each represented as $\mathbf{L_B}$ with a time window of size T. Now, we take the LNES frame $\mathbf{L_q}$ and its corresponding human segmentation mask from either EE3D-S or EE3D-R. We perform element-wise multiplication between $\mathbf{L_B}$ and the inverse of the human segmentation mask, resulting in $\mathbf{L_A}$. The inverted human segmentation mask functions as the background mask, effectively removing the human. Finally, we perform element-wise addition of $\mathbf{L_A}$ with $\mathbf{L_q}$, which serves as the input to our network.

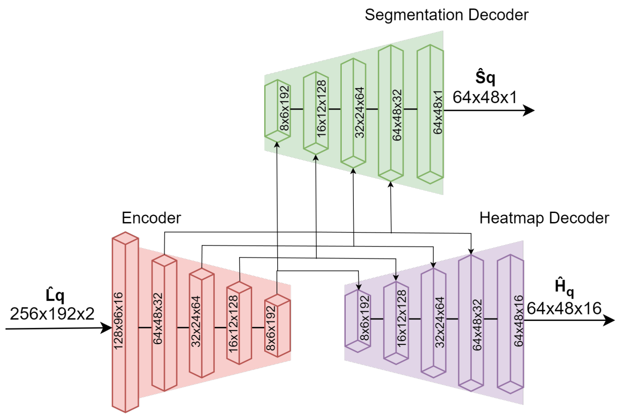


Figure 2. Network architecture of the Encoder (bottom left), Heatmap Decoder (bottom right) and Segmentation Decoder (top).

2. Implementation Details

2.1. Architectures

The Encoder, Heatmap Decoder, and Segmentation Decoder are constructed using Blaze blocks [2]; see Fig. 2. The Confidence Decoder and Heatmap-to-3D (HM-to-3D) Lifting Block are built using standard convolution filters.

The Confidence Decoder is a four-layer fully convolutional neural network. In each convolution layer, filters with a kernel size of three are utilised, followed by the PReLU activation function [7]. We apply appropriate padding to maintain the spatial dimensions at each layer of the network, hence the output of the network has the same dimensions as the input of the network.

The HM-to-3D Lifting block as shown in Fig. 3 is a six-layer network with three convolution layers and three dense layers. Each convolution layer consists of filters with a kernel size of four, followed by batch normalisation and RELU activation function. Subsequently, we perform average pooling and flatten the features outputted by the convolution layers. Finally, these features are passed to the dense layers to estimate the positions of the joints, denoted as $\hat{J}$.

2.2. Real-time Inference

3D Viewer. Our method runs locally on the laptop housed in the backpack. We visualise the results using a separate device running a 3D viewer [5]. The predictions generated by our method are transmitted to the 3D viewer through network sockets. Note that the data transmission causes a slight lag in the visualisations, especially during complex and fast motions for which the pose update frequency is very high.

Temporal Stability. We reduce the jitter generated by our method for real-time inference with the 1€ filter [4]. For a fair comparison, the same procedure is also performed for all the methods we evaluate.

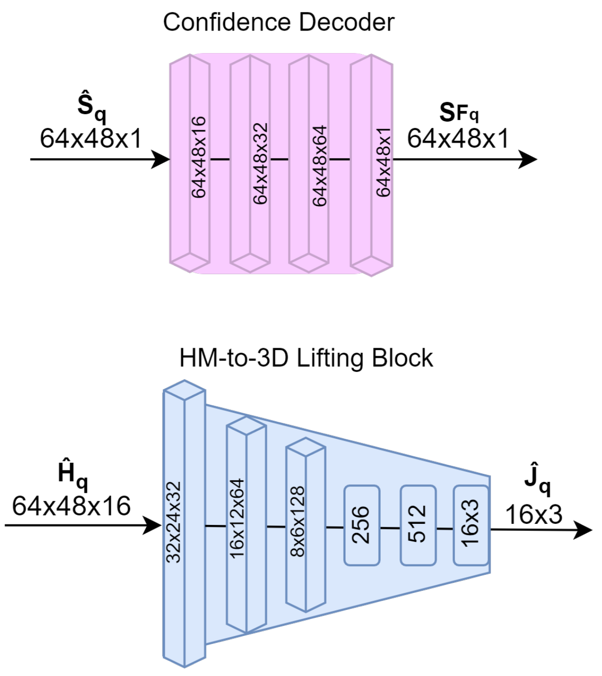


Figure 3. The network architecture of the Confidence Decoder (top) and Heatmap-to-3D (HM-to-3D) lifting block (bottom).

3. Additional Experiments

	MPJPE ↓	PA-MPJPE ↓
Tome *et al.* [11]	172.14	124.62
Rudnev *et al.* [9]	217.05	136.05
Xu *et al.* [13]	196.39	99.07
EventEgo3D (Ours)	**124.85**	**92.58**

Table 1. Quantitative Evaluation on EE3D-S.

Evaluation on EE3D-S. Table 1 quantitatively evaluates our approach and competing methods on EE3D-S. In this experiment, all the methods are pre-trained on EE3D-S and fine-tuned on EE3D-R. We then compare the methods on the test set of EE3D-S. Overall, our method achieves the lowest MPJPE, outperforming Tome *et al.* [11] and Xu *et al.* [13] by 27.47% and 36.42% on MPJPE, respectively. Rudnev *et al.* [9] perform the worst in our testing achieving an MPJPE of 217.05mm.

Ablation on Dataset Training Strategy. Table 2 summarises the quantitative evaluation of our method on the EE3D-R dataset using different training strategies. Training our method solely on EE3D-S without fine-tuning on EE3D-R yields the poorest performance. Pre-training our method on EE3D-S and subsequently fine-tuning it on EE3D-R results in a lower MJPJE (denoted as "Ours w/o Aug") compared to training our method exclusively on EE3D-R (denoted as "Ours with EE3D-R"). Specifically, this approach improves the MJPJE by 2.16%. Furthermore, augmenting the events with background data (refer to Sec. 1.3) in conjunction with fine-tuning leads to the best MJPJE of 107.30mm.

Additional Visualisations. We provide additional visualisations for our method, showcasing intermediate representations produced by each module (refer to Table 3). The input frame, denoted as $\hat{\mathbf{L}}_q$, is computed based on the current LNES frame $\mathbf{L}_q$ and the previous input frame $\hat{\mathbf{L}}_{q-1}$. The segmentation decoder first estimates the segmentation mask, which then serves as input for the confidence decoder to generate the confidence map. Simultaneously, the heatmap decoder estimates the heatmaps of the human body joints. These heatmaps are then input into the HM-to-3D lifting block, resulting in the regression of the 3D joint locations.

References

[1] Cmu graphics lab motion capture database. 1

[2] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking. *arXiv preprint arXiv:2006.10204*, 2020. 2

[3] Blender Online Community. *Blender - a 3D modelling and rendering package, Version: 2.82 (sub 7)*. Blender Foundation, Blender Institute, Amsterdam, 2020. 1

[4] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *SIGCHI Conference on Human Factors in Computing Systems*, 2012. 2

[5] EasyMoCap - Make human motion capture easier. `https://github.com/zju3dv/EasyMocap`, 2021. 2

[6] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, 2015. 2

[8] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, 2015. 1

[9] Viktor Rudnev, Vladislav Golyanik, Jiayi Wang, Hans-Peter Seidel, Franziska Mueller, Mohamed Elgharib, and Christian Theobalt. Eventhands: Real-time neural 3d hand pose estimation from an event stream. In *International Conference on Computer Vision (ICCV)*, 2021. 3

[10] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006. 1

Method	Metric	Walk	Crouch	Pushup	Boxing	Kick	Dance	Inter. with env.	Crawl	Sports	Jump	Avg. (σ)
Ours with EE3D-S	MPJPE	326.17	319.17	239.92	254.62	318.03	274.43	279.79	327.40	317.49	316.91	297.39 (32.29)
	PA-MPJPE	180.83	173.56	134.47	130.56	181.62	146.33	162.28	162.90	153.61	154.39	158.05 (17.76)
Ours with EE3D-R	MPJPE	87.21	163.80	101.40	132.50	111.68	98.72	102.07	135.71	107.53	106.27	114.69 (22.73)
	PA-MPJPE	69.16	110.14	78.92	103.18	94.30	77.49	72.59	104.63	81.75	82.75	87.49 (14.48)
Ours w/o Aug	MPJPE	80.02	127.62	97.68	119.92	118.06	130.22	107.27	93.78	132.21	115.37	112.21 (17.25)
	PA-MPJPE	60.04	95.74	76.33	95.54	89.71	103.02	88.22	74.07	94.72	85.77	86.32 (12.82)
Ours	MPJPE	**70.88**	163.84	**97.88**	136.57	**103.72**	**88.87**	**103.19**	**109.71**	**101.02**	**97.32**	**107.30** (25.78)
	PA-MPJPE	**52.11**	**99.48**	**75.53**	104.66	**86.05**	**71.96**	**70.85**	**77.94**	**77.82**	**80.17**	**79.66** (14.83)

Table 2. Numerical comparisons on the EE3D-R dataset with different dataset training strategies. Fine-tuning our method on EE3D-R after pre-training on EE3D-S yields an improvement of 62.26% on the MPJPE.

[11] Denis Tome, Patrick Peluse, Lourdes Agapito, and Hernan Badino. xr-egopose: Egocentric 3d human pose from an hmd camera. In *International Conference on Computer Vision (ICCV)*, 2019. 3

[12] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[13] Weipeng Xu, Avishek Chatterjee, Michael Zollhoefer, 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 Transactions on Visualization and Computer Graphics*, 25(5):2093–2101, 2019. 1, 3

[14] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 1

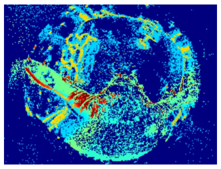
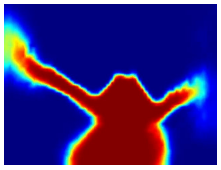
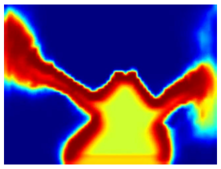
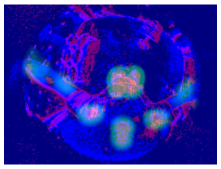
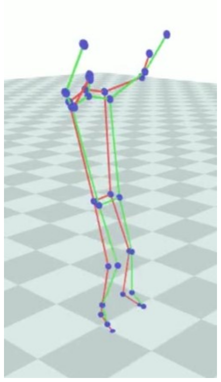
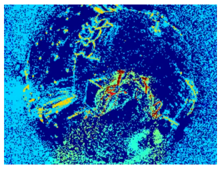
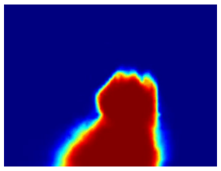
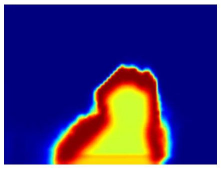
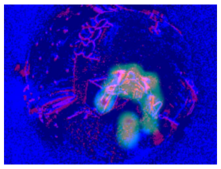
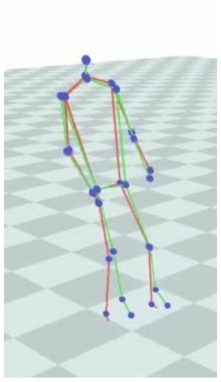
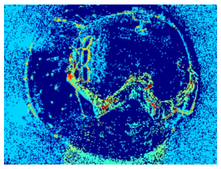
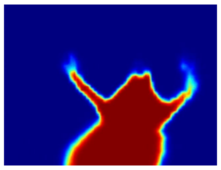
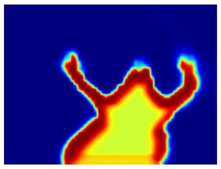
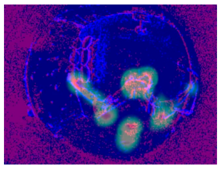
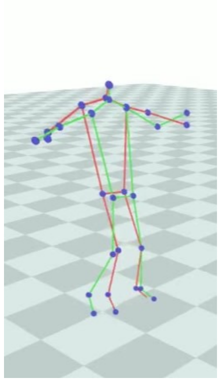
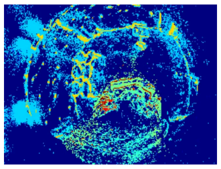
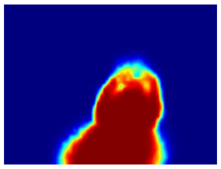
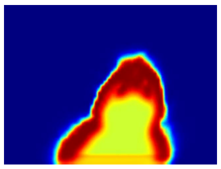
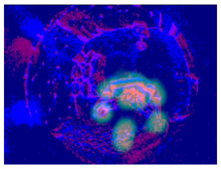
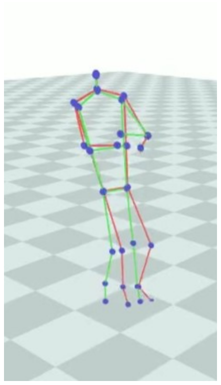
Input frame $\hat{\mathbf{L}}_q$	Human Body Mask	Confidence map	Heatmap	Prediction
				
				
				
				

Table 3. Additional visualisations of EE3D along with the visualisations of input frames $\hat{\mathbf{L}}_q$, human body masks, confidence maps, heatmaps and the 3D predictions. The 3D poses depicted in green represent the ground truth, while those in red signify the predictions by our method. The colour scheme for the input frames $\hat{\mathbf{L}}_q$, human body masks and confidence maps can be found in Fig. 3 of the main paper.