Supplementary Material for EM-POSE: 3D Human Pose Estimation from Sparse Electromagnetic Trackers

Manuel Kaufmann[1,2] Yi Zhao[2] Chengcheng Tang[2] Lingling Tao[2]
Christopher Twigg[2] Jie Song[1] Robert Wang[2] Otmar Hilliges[1]

[1]ETH Zürich, Department of Computer Science [2]Facebook Reality Labs

In this supplementary material we give more details about the computation of subject-specific offsets (Sec. 1), describe training details of our proposed method (Sec. 2) and the optimization and learning baselines (Sec. 3, Sec. 4), provide details how we compute the accuracy of our EM sensors (Sec. 5), provide more quantitative comparisons (Sec. 6), show additional visualizations and failure cases (Sec. 7), describe the detailed capture protocol of our test set $\mathcal{T}$ (Sec. 8), and finally describe some initial experiments how to tackle denoising of EM data (Sec. 9). For more visualizations, please also refer to the video.

1. Computation of $\boldsymbol{O}_p$

To take into account the subject-specific offsets, we reserve a "calibration sequence" taken from $\mathcal{T}$ for each of our subjects. We use these sequences to extract per-sensor and per-subject offsets $\boldsymbol{O}_p$. The following description holds for each sensor s of subject p. We obtain $\boldsymbol{o}_s$ by first computing $\tilde{\boldsymbol{m}}_s$ on the calibration sequence following Eq. (3). This allows us to solve for $\boldsymbol{R}(t)$ and $\boldsymbol{t}(t)$ in Eq. (4) at every time step t, where we simply replace $\boldsymbol{R}_s^v$ and $\boldsymbol{p}_s^v$ with the actual real measurements.

Because $\boldsymbol{R}(t)$ and $\boldsymbol{t}(t)$ can vary over time, we extract a single estimate as follows. For $\boldsymbol{t}$ we simply compute the mean over all time steps. For $\boldsymbol{R}$ we compute the average rotation over $\boldsymbol{R}(t)$.

$$\boldsymbol{t} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{t}(t) \tag{1}$$

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \text{SVD}(\sum_{t=1}^{T}\boldsymbol{R}(t))$$

$$\boldsymbol{R} = \phi(\boldsymbol{U}, \boldsymbol{V}) \tag{2}$$

where ϕ extracts a valid rotation as follows:

$$\phi(\boldsymbol{U}, \boldsymbol{V}) = \boldsymbol{U} \cdot \text{diag}(1, 1, \text{sign}(\det(\boldsymbol{U}\boldsymbol{V}^T)) \cdot \boldsymbol{V}^T \tag{3}$$

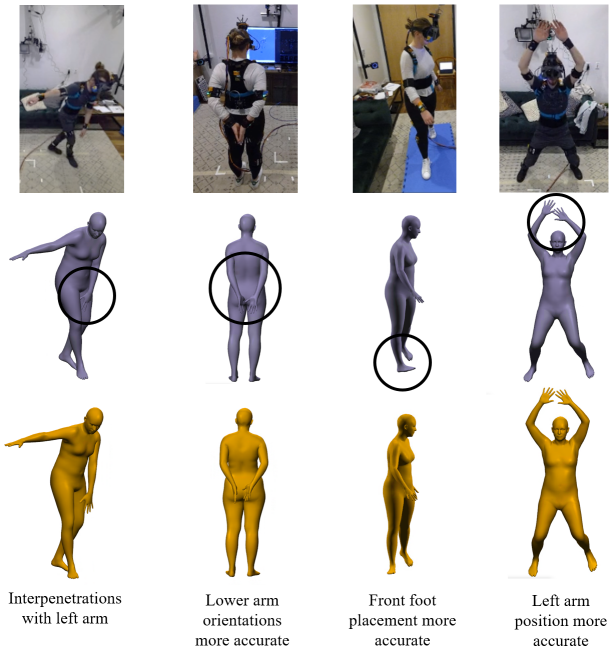

Figure 8: **Reconstructions** with 6 sensors with the BiRNN (middle row) and our best model, LGD RNN (bottom row). Differences are described directly in the figure.

2. Training Details

2.1. Normalization

We normalize our data before feeding it to our method. Since we assume that the root information is given we remove the root translation entirely by setting the SMPL root translation to zero. Then, for every sequence, we normalize the SMPL root orientation as follows (superscript n indicates it is normalized data):

$$\boldsymbol{R}_{root}^n(t) = \boldsymbol{R}_{root}^{-1}(0)\boldsymbol{R}_{root}(t)$$

Since the remaining SMPL pose parameters are all

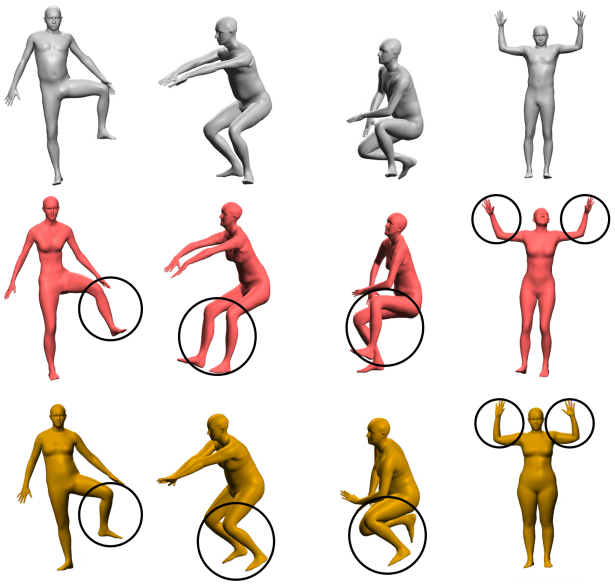

Figure 9: **MoSh++ results** [7] when using 12 sensors as input (middle row) compared to reference poses (top row) and our method (LGD RNN) with 6 sensors (bottom row).

parent-relative, this is the only normalization we perform for SMPL data. We apply the same normalization to the marker data, *i.e.*

$$\boldsymbol{p}_s^n(t) = \boldsymbol{R}_{root}^{-1}(0)(\boldsymbol{p}_s(t) - \boldsymbol{t}_{root}(t))$$
$$\boldsymbol{R}_s^n(t) = \boldsymbol{R}_{root}^{-1}(0)\boldsymbol{R}_s(t)$$

2.2. Offset augmentation

As explained in Sec. 1, the computed offsets $\boldsymbol{R}(t)$ and $\boldsymbol{t}(t)$ can vary over time. We use the translational part of the offsets to introduce some noise for data augmentation purposes during training. To do so, for every sensor and every participant we fit a multi-variate normal distribution to $\boldsymbol{t}(t)$, denoted as $\boldsymbol{\varphi}(\boldsymbol{t})$. When applying the offsets $\boldsymbol{O}_p$ as explained in Sec. 5.3 of the main paper we first draw a vector $\boldsymbol{t} \sim \boldsymbol{\varphi}(\boldsymbol{t})$ which we then use as the translational offset to obtain virtual sensors $\boldsymbol{m}_s^v$.

2.3. Architecture Details and Hyperparameters

The RNN in our proposed method, LGD RNN, consists of two LSTM layers [2] of size 512. The output of the RNN is mapped to pose and shape parameters $\boldsymbol{\Omega}_t^{(0)}$ with a dense layer. The network $\mathcal{N}$ is essentially a multi-layer perceptron (MLP). The MLP first maps its inputs, *i.e.* $\boldsymbol{\Omega}_t^{(n)}$, to the chosen hidden size, which is 512 in our case. The hidden representation is then passed to L (here 5) dense layers whereas each layer maps to the same dimensionality as the size of its inputs (*i.e.* 512). Each dense layer is preceded by a batch normalization layer [4], a PReLU activation function [1], and a dropout layer [11] in this order. The last dense layer maps back to the target dimension and thus produces the next estimate $\boldsymbol{\Omega}_t^{(n+1)}$.

We use the Adam optimizer [5] to train our models. The choice of hyperparameters are listed in Tab. 5. We use PyTorch 1.6 [8] and train all our models on a NVIDIA GeForce GTX 1080Ti, which takes roughly 16 hours.

Hyperparameter	LGD RNN 6	LGD RNN 12
α (LGD step size)	0.1	0.1
Batch size	12	12
Dropout (on inputs)	0.0	0.2
Dropout (inside MLP of $\mathcal{N}$)	0.0	0.2
λ_1 (pose loss weight)	10.0	1.0
λ_2 (shape loss weight)	1.0	1.0
λ_3 (joint loss weight)	0.1	1.0
λ_4 (reconstruction loss weight)	0.01	0.01
Learning rate	0.0005	0.0001
N (number of LGD iterations)	2	4
Number of epochs	50	50
Sequence length (training only)	32	32

Table 5: **Hyperparameters** for LGD RNN.

Hyperparameter	ResNet 6/12	BiRNN 6/12
Batch size	16	16
λ_1 (pose loss weight)	1.0	1.0
λ_2 (shape loss weight)	1.0	1.0
λ_3 (joint loss weight)	10.0	10.0
Learning rate	0.0005	0.0005
Number of epochs	50	50
Sequence length (training only)	128	128

Table 6: **Hyperparameters** for learning baselines.

3. Optimization baseline

Here we explain the details of our optimization baseline mentioned in Sec. 6.2 of the main paper. The objective function we minimize is $\arg\min_{\boldsymbol{\Omega}_t} \mathcal{L}_r(\boldsymbol{x}_t, \boldsymbol{\Omega}_t, \boldsymbol{O}_p)$, *i.e.* essentially the same objective that LGD minimizes. However, to induce a prior we operate directly in the latent space provided by VPoser [9]. This means the body parameters $\boldsymbol{\Omega}_t$ are now split into $(\boldsymbol{z}_t, \boldsymbol{\beta})$ where $\boldsymbol{z}_t$ corresponds to the latent space of VPoser. Furthermore, we add regularizers on pose and shape. The objective function we minimize thus becomes

$$\arg\min_{\boldsymbol{z}_t, \boldsymbol{\beta}} \mathcal{L}_r(\boldsymbol{x}_t, \boldsymbol{z}_t, \boldsymbol{\beta}, \boldsymbol{O}_p) + \rho_1 ||\boldsymbol{z}_t||_2^2 + \rho_2 ||\boldsymbol{\beta}||_2^2 \quad (4)$$

where we choose $\rho_1 = 10^{-6}$ and $\rho_2 = 10^{-2}$. We use PyTorch to run our optimization and use an LBFGS optimizer with a step size of 1.0 and strong Wolfe line search.

Model	MPJPE [mm]	PA-MPJPE [mm]	MPJAE [°]
Ours 6 no $\boldsymbol{t}$	44.0 ± 34.0	32.8 ± 22.3	15.8 ± 10.6
Ours 6 ori only	80.9 ± 81.4	53.5 ± 46.3	18.0 ± 13.6
Ours 6 pos only	38.6 ± 32.0	31.4 ± 25.2	17.7 ± 12.4
Ours 6 no RNN	44.4 ± 33.3	32.8 ± 23.9	16.2 ± 11.3
Ours 6	**35.4 ± 21.3**	**27.0 ± 16.3**	**14.9 ± 10.0**

Table 7: **Ablation studies** on our best performing 6-sensor model.

4. Learning baselines

Here we describe the details of our learning-based baselines, ResNet and BiRNN, as described in Sec. 6.2 of the main paper. Both baselines perform direct body parameter regression, *i.e.* we obtain SMPL pose and shape estimates $\hat{\boldsymbol{\Omega}}_t$ directly from a neural network $\nu(\boldsymbol{x}_t)$. We use the same data augmentation and preprocessing as for LGD RNN. The loss function at time step t is the same in both cases:

$$\mathcal{L}_t = \lambda_1 \mathcal{L}_1(\hat{\boldsymbol{\theta}}_t, \boldsymbol{\theta}_t^{gt}) + \lambda_2 \mathcal{L}_2(\hat{\boldsymbol{\beta}}, \boldsymbol{\beta}^{gt}) + \lambda_3 \mathcal{L}_3(\hat{\boldsymbol{J}}_t, \boldsymbol{J}_t^{gt})$$

where $\mathcal{L}_1, \mathcal{L}_3$ are the MSE and $\mathcal{L}_2$ is the L1 loss. The architectural details are explained in the following and hyperparameters are listed in Tab. 6.

ResNet The ResNet baselines is a frame-wise architecture inspired by [3]. One block consists of a dense layer that maps to the same output size as the size of the inputs, followed by a skip connection and a ReLU activation function. We use 5 such layers of dimension 1024. The output of the last layer is mapped directly to $\boldsymbol{\Omega}_t$.

BiRNN The BiRNN is a simple bidirectional RNN [10] with LSTM cells [2]. We use 2 bidirectional layers of size 256 each. The hidden forward and backward states of the last layer are mapped directly to $\boldsymbol{\Omega}_t$.

5. Computation of EM-Tracking Accuracy

To compare our EM sensors to optical marker-based tracking we glued an Optitrack rigid body to each of our sensors (*c.f.* Fig. 2 of the main paper). Here we explain in detail how we compute the disagreement between Optitrack and our sensors (*c.f.* Sec. 6.1 of the main paper). As a reminder, for every sensor s and every time step t we obtain four measurements: the Optitrack 6D pose, *i.e.* $\boldsymbol{p}_s^O(t)$ and $\boldsymbol{R}_s^O(t)$, and the EM 6D pose, *i.e.* $\boldsymbol{p}_s^M(t)$ and $\boldsymbol{R}_s^M(t)$. All measurements are calibrated to world space and hence, under perfect agreement, a constant rigid transformation $[\boldsymbol{R} \mid \boldsymbol{t}]$ would relate the two. We characterize the agreement by computing this rigid transformation and measuring how much it changes over time as follows. For the positional agreement $e_s^{pos}(t)$ we simply compute the deviation from the mean translational offset.

$$\boldsymbol{t} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{p}_s^M(t) - \boldsymbol{p}_s^O(t)$$
$$e_s^{pos}(t) = ||\boldsymbol{p}_s^M(t) - \boldsymbol{p}_s^O(t) - \boldsymbol{t}||_2$$

For the angular error $e_s^{ang}(t)$ we proceed similarly and solve an orthogonal Procrustes problem to find the constant rotation $\boldsymbol{R}$ that best relates $\boldsymbol{R}_s^M$ and $\boldsymbol{R}_s^O$ as follows:

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \text{SVD}\left(\sum_{t=1}^{T} (\boldsymbol{R}_s^M(t))^T \boldsymbol{R}_s^O(t)\right)$$
$$\boldsymbol{R} = \phi(\boldsymbol{U}, \boldsymbol{V})$$
$$e_s^{ang}(t) = \text{dist}(\boldsymbol{R}_s^M(t)\boldsymbol{R}, \boldsymbol{R}_s^O(t))$$

where ϕ is defined in Eq. (3) and $\text{dist}(\cdot)$ finds the closest angle of rotation between its inputs. To do so we first convert the rotation matrices to quaternions and then use:

$$\text{dist}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \cos^{-1}\left(2\langle \boldsymbol{q}_1(t), \boldsymbol{q}_2(t)\rangle^2 - 1\right)$$

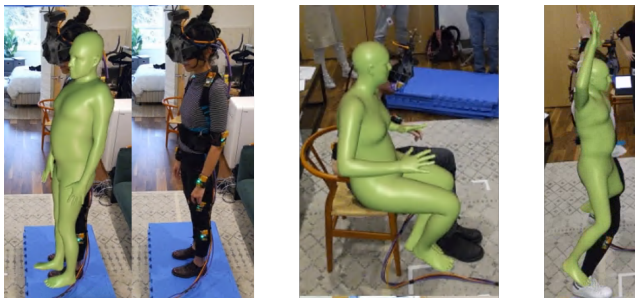

Figure 10: **Failure cases.** Inaccurate shape reconstruction especially around abdomen (left) and challenging lower leg orientations (middle and right).

6. More Quantitative Results

6.1. Comparison to RGB Methods

We compare our method to a state-of-the-art monocular RGB-based pose estimator, VIBE [6]. To do so, we select the camera facing the front of the subject as input to VIBE. The results are shown in Tab. 8. The error is only computed on frames for which VIBE detected a person. As we can see, VIBE does not perform favourably on our data. This is not unexpected because a) our imagery is a real in-the-wild scenario and b) the inputs to VIBE and our method are vastly different. Under these circumstances, VIBE still performs admirably. We see this experiment as further motivation to employ EM-based systems to gather reference data to boost RGB-based methods down the line.

Model	MPJPE [mm]	PA-MPJPE [mm]	MPJAE [°]
VIBE [6]	100.3 ± 79.3	70.1 ± 58.2	24.8 ± 15.7
Ours (LGD-RNN) 6	**36.4 ± 23.7**	**28.1 ± 16.9**	**13.6 ± 8.8**

Table 8: **Comparison to VIBE** on 10 representative test sequences from subjects 1-4.

6.2. Ablation Study with 6 Sensors

In Tab. 7 we provide the same ablation study as in Sec. 6.3 of the main paper but with the 6 instead of the 12 sensor model. Based on this table we can see that the same conclusions hold as already drawn in the main paper.

7. More Visualizations

To highlight the differences between our best model, LGD RNN, and its closest baseline, BiRNN, we compare their performance visually in Fig. 8. We can see that the BiRNN sometimes produces interpenetrations and lacks some accuracy at the end effectors.

Furthermore, we also compare the performance of MoSh++ [7] in Fig. 9. The chosen frames highlight that the shape estimates of MoSh++ are sometimes off by quite a margin. This is because it makes different assumptions about the sensor-to-skin offsets. Furthermore, the orientation of end effector segments often exhibit a high error in the MoSh++ results. This is not unexpected since it only uses 12 positional estimates. In contrast, our best model (*c.f.* bottom row in Fig. 9) produces more accurate limb orientations even with only 6 sensors as it uses both position *and* orientation inputs.

We also show failure cases of our method in Fig. 10. Shape estimation from just a few on-skin measurements is challenging. We sometimes see bulging bellies (*c.f.* Fig. 10) and inaccurate shape in the hip region (*c.f.* bottom right corner in Fig. 9). Getting the lower leg orientation correct in extreme articulations is difficult, too, even with explicit orientation measurements (*c.f.* Fig. 10). In addition, such errors are visually very striking as they can cause foot sliding.

8. Test Set Details

We describe the detailed content and capture protocol of our test set $\mathcal{T}$ in Tab. 9. All participants were guided by an assistant, participated voluntarily and gave written consent to record and publish their data.

9. Denoising Experiments

The data measured by our EM-based capture system can be noisy. Typical sources of noise include dropped frames (due to sensor malfunctions or wireless connections), increased jitter when operating outside the calibrated range, or unexpected magnetic distortion. Our proposed method, LGD RNN, iteratively fits SMPL to the observed EM data. Hence, it is clear that LGD RNN cannot handle certain types of noise, such as dropped sensors. We find pose estima-

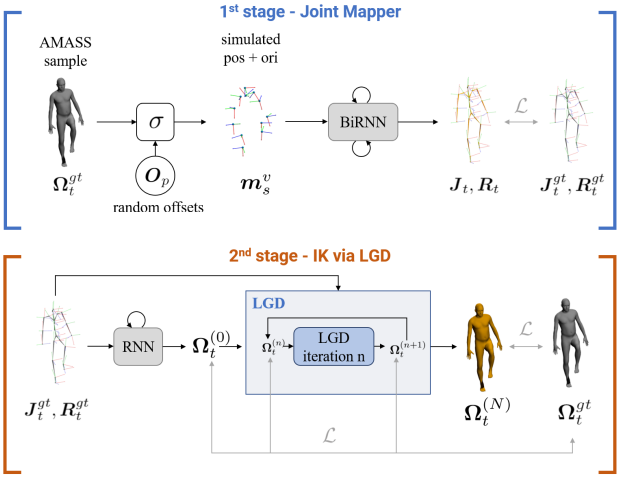

Figure 11: **Denoising architecture overview.** Our architecture to estimate SMPL pose and shape with noisy input measurements works in two stages. The first stage, called joint mapper, maps EM data to 3D SMPL joints $\boldsymbol{J}_t$ and root-relative joint orientations $\boldsymbol{R}_t$. This is a simple two-layer BiRNN which we directly supervise with the ground-truth joint positions and orientations. We randomly remove one or several sensor measurements from the input for half the duration of the given sequence. The second stage is performing IK and uses the LGD framework to do so. The orientations $\boldsymbol{R}_t$ help to disambiguate the orientation of bone segments (especially so for end effectors). At training time we use synthetic EM measurements $\boldsymbol{m}_s^v$ to train the joint mapper. Ground-truth joint positions $\boldsymbol{J}_t^{gt}$ and orientations $\boldsymbol{R}_t^{gt}$ extracted from AMASS are used to train the IK stage. At test time we simply feed the real EM data to the joint mapper and the output of the joint mapper to the IK stage.

Figure 12: **Denoising comparison.** For this frame, the right lower leg sensor is missing in the input. LGD RNN struggles to reconstruct the pose (middle) whereas the two-stage approach does a better job (right). Notice how the missing sensor is affecting the entire pose output for LGD RNN.

Action Type	Description	# Frames	Minutes
Arms ROM	Arm raises, arm swings, cross arms, clap hands front and back with straight arms.	14,109	7.8
Arms Fast	Fast arm swings, pretend to play Beat Saber VR, punches, rotate wrists around each other fast.	7,169	4.0
Calibration	Move head left to right and rotate wrists in T-Pose, move head left to right and rotat rotate wrists when arms stretched in front, one leg raise each.	3,709	2.1
Head and Shoulders	Nod head back and forth, move head left to right, roll head left to right, rotate shoulders forwards and backwards, rotate torso, bend over and move arms around.	9,498	5.3
Jumping Jacks	3-5 jumping jacks.	1,957	1.1
Lower Body	Leg raises left and right, raise leg then rotate outwards, squats, crouching	7,305	4.1
Lunges	Crouching, several lunges with left and right foot in front.	4,714	2.6
Sitting on chair	Grab a chair, sit on chair, move one leg over the other, pretend to sit at a table and interact with PC, keyboard, touch screens.	9,362	5.2
Walking	Walk normally from left to right in capture area, side-stepping from left to right with and without crossing over the legs.	8,391	4.7
Total		**66,214**	**36.8**

Table 9: **Test set $\mathcal{T}$.** Description and length of the sequences in our test set $\mathcal{T}$. Each of the 5 participants performed the actions described here in a single session. Each sequence starts and ends with a T-Pose.

Frames	Model	MPJPE [mm]	PA-MPJPE [mm]	MPJAE [°]
all	LGD RNN 12	**33.3** ± 24.3	**26.2** ± 18.8	**13.4** ± 9.2
	2-stage 12	38.8 ± **22.0**	28.2 ± **17.3**	13.8 ± **9.1**
avail.	LGD RNN 12	**31.8** ± **21.0**	**24.8** ± **16.4**	**13.3** ± 9.2
	2-stage 12	39.1 ± 21.3	28.1 ± 16.7	13.7 ± **9.1**
miss.	LGD RNN 12	45.6 ± 40.6	37.7 ± 30.0	15.0 ± 9.3
	2-stage 12	**36.2** ± **27.0**	**29.1** ± **21.9**	**14.6** ± **9.1**

Table 10: **Denoising experiments** on all frames (*all*), only on frames with*out* missing sensors (*avail.*) and only on frames with at least one missing sensor (*miss.*).

tion in such a noisy data regime an interesting direction for future work and experimented with an initial architecture that can cope with dropped frames and magnetic distortion which we briefly describe in the following.

Although LGD RNN *can* handle some noise (by means of incorporating pose priors), it is not meant to be a denoising architecture per se as it fits SMPL pose and shape to the inputs directly. Hence, our idea is to separate the tasks of denoising and fitting into separate modules and came up with a two-stage architecture. The first stage, also called joint mapper, regresses SMPL 3D joint positions and root-relative joint orientations from the input observations. In this stage we randomly remove sensors from the input to simulate dropped frames. Thus, the joint mapper maps to a proxy representation that is close to SMPL while also denoising the inputs. The second stage then lifts the output of the joint mapper to the final estimate of SMPL pose and shape. This is again an LGD-based iterative fitting procedure. Experimentally we have found that using LGD for this stage outperforms an optimization-based IK step. Both stages are trained independently. For an overview, please refer to Fig. 11.

In our experiments we have found that this two-stage architecture yields good results. The final SMPL poses are smooth and with 12 input sensors 1-2 missing sensors are compensated plausibly. This is also reflected in quantitative comparisons shown in Tab. 10. In this table we compare the two-stage approach to LGD RNN when using 12 sensors. We report its performance on 3 sets of frames: frames that have no missing sensors (*avail.*), frames with at least one missing sensor (*miss.*) and the union of these two sets, which corresponds to all frames in our test set (*all*). We observe that the two-stage approach does not beat LGD RNN on the good frames where no sensor data is missing, but remains competitive. It also trails behind LGD RNN on all the frames, which makes sense since we have many more "good" frames than frames with missing sensors. However, on frames where at least one sensor is missing (*miss.* in Tab. 10), the two-stage architecture shows its potential and clearly outperforms LGD RNN. For a visual example of a denoised frame please refer to Fig. 12.

The two-stage model is not only interesting to fill in missing sensor data. If we assume we have a mechanism to detect magnetic distortion, we can simply suppress the sensor measurement for those time instances where magnetic

distortion is detected. We can then use the same two-stage architecture to remedy the impact of EM interference. We find this an interesting direction for exploration and release code and data to foster future research.

References

[1] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. 2

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2, 3

[3] Daniel Holden. Robust solving of optical motion capture data by denoising. *ACM Trans. Graph.*, 37(4), July 2018. 3

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015. 2

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 2

[6] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3, 4

[7] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. Amass: Archive of motion capture as surface shapes. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019. 2, 4

[8] 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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2

[9] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10975–10985, 2019. 2

[10] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. 3

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 2