– Supplementary Material –
Learning 3D Human Pose Estimation from Dozens of Datasets using a Geometry-Aware Autoencoder to Bridge Between Skeleton Formats

István Sárándi Alexander Hermans Bastian Leibe
RWTH Aachen University
{sarandi,hermans,leibe}@vision.rwth-aachen.de

Abstract

In this supplementary material, we provide additional qualitative results, as well as details about implementation, data processing and evaluation. We also provide two additional ablations (training length and batch norm configuration) and a derivation for the claim that ℓ_1 regularization of the affine-combining autoencoder leads to a reduction in negative weights and hence near-convex combinations.

S1. Additional Qualitative Results

Similar to the qualitative results shown in the main paper, Figures S2, S3, and S4 show further predictions for a variety of images. It can clearly be seen that the model with separate heads without consistency regularization creates rather inconsistent skeleton predictions, whereas fine-tuning with our ACAE regularization significantly improves the consistency. Furthermore, these figures show that the resulting models display excellent in-the-wild performance, even on challenging poses, or in suboptimal lighting conditions.

S2. Training Details

Learning Rate. Our learning rate schedule is shown in Fig. S1. The learning rate starts at 2.12e-4 and exponentially decays by a factor of 3 over 92% of training, then drops by a factor of 10 and then further decays exponentially by a factor of 3 until the end of the initial training.

The fine-tuning phase uses two different learning rates. We perform a warm restart on the last layer (the prediction head) in order to ensure that the regularization loss can take effect, without disrupting the already mostly converged weights of the backbone. For the head, we follow a similar recipe as in the initial training, but we perform the large learning rate drop at 50% of the fine-tuning phase. For the backbone, we repeat the last, decaying segment of the initial schedule.

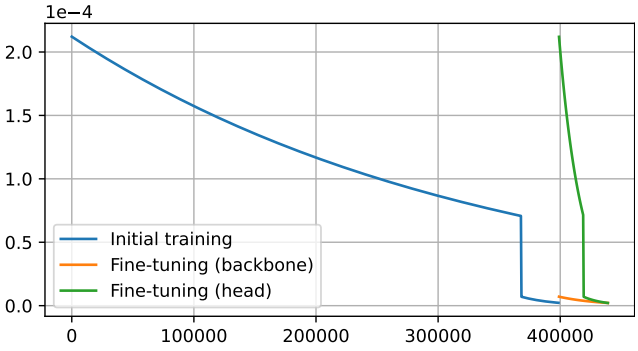

Figure S1: Learning rate schedule.

Loss Details. We make minor adjustments to the MeTRAbs model [5], which we use as the basis of our experiments. In [5], the authors perform internal supervision on the output of a 2D heatmap head and a 3D heatmap head. Instead, we simplify this and only use the absolute pose output for supervision, *i.e.*, the 2D projection loss and the mean-relative loss are computed on this single, absolute output. This makes the implementation cleaner when more losses are added for consistency regularization or student-teacher latent matching, etc., since the model can be treated as a single-output black box.

The weak supervision loss for 2D-annotated examples only consists of the 2D projection loss. For this, we do not specifically predict skeletons according to the skeleton formats of the 2D datasets. Instead, the prediction is derived by averaging the corresponding 3D joint predictions for every output skeleton format. In other words, for the calculation of the 2D weak loss, we consider our prediction for the left shoulder to be the average of all the left shoulder joints in every skeleton format that we use.

We use $\lambda_{\text{proj}} = 1$ and $\lambda_{\text{abs}} = 0.1$ and scale the weak-supervision-loss by a factor of 0.2.

The absolute loss $\mathcal{L}_{\text{abs}}$ is only turned on after 5000 steps (also in fine-tuning, for consistency), similarly with the teacher loss. In some datasets the absolute distance to the person can be very large (*e.g.*, JTA, SAIL-VOS, ASPset).

Here the absolute loss would overwhelm the total loss, so we scale down the absolute Z component to a maximum effective distance of 10 m for loss computation.

Batch Composition. In Tab. S1, we specify the number of examples from each dataset per batch. This is based on the total number of examples in each dataset but not linearly, as we oversample smaller datasets compared to their size, in order to provide more diverse supervision to the model. For batch generation, we set up one queue per dataset that iterates over epochs of that dataset, then we interleave the streams and chunk it into batches (as opposed to independently sampling each batch).

Initialization Details. We initialize with ImageNet-pretrained weights. For the RN50 experiment in Table 4 (SOTA), we use ResNet50V1.5 as implemented in PyTorch, ported to TensorFlow, along with the ImageNet weights, which we found to be superior to the ones provided with TensorFlow.

We precisely control the random seeds, which guarantees that bitwise equal batches are fed to each training run, improving comparability.

Implementation Details. We use TensorFlow 2.9 with Keras, CUDA 11.4 and CuDNN 8.2.4 for the implementation. Training takes about 2 days with the EffV2-S backbone and about 6 days with EffV2-L on a single Nvidia A40 GPU (48 GB) in mixed FP16/FP32 precision.

S3. Data Processing Details

Where missing, we obtain person bounding boxes with YOLOv4 [1] and person segmentation with DeepLabv3 [2]. Examples with implausible bone lengths are removed to avoid training on erroneous annotations. We use all cameras of 3DHP, and all HD cameras of CMU-Panoptic (and all sequences with labels). We further calibrated all cameras of BML-MoVi that did not have calibration provided in the dataset, and use all of them in training (based on pose predictions from an earlier version of our model). We use 200k composited images for MuCo-3DHP, generated with the official Matlab script.

S4. Evaluation Details

We evaluate all 24 SMPL joints for 3DPW, and all 17 joints for 3DHP and MuPoTS. In case of 3DPW, the entire dataset is used for testing. For 3DHP we use the official split, for H36M the most common split from the literature, *i.e*. subjects S9 and S11 are used for testing.

For MuPoTS, we evaluate the matched poses. As we use the same YOLOv4 detector in all our experiments, we have 94.6% recall in all of our experiments (hence the matched-pose results are directly comparable). For our main evaluations, in each benchmark, we simply calculate the average metrics over all metric-scale poses.

Table S1: Batch composition for the experiments with the three different levels of dataset combinations. Each mini-batch consists of 96 examples with 3D labels and 32 with 2D labels.

Dataset name	Small	Medium	Full
Real images with markerless MoCap			
MuCo-3DHP	32	9	6
CMU-Panoptic	–	9	7
AIST-Dance++	–	9	6
HUMBI	–	7	5
MPI-INF-3DHP	–	5	3
RICH	–	7	4
BEHAVE	–	–	3
ASPset	–	–	4
3DOH50K	–	–	3
IKEA ASM	–	–	2
Real images with marker-based MoCap			
Human3.6M	32	9	4
TotalCapture	–	5	3
BML-MoVi	–	–	5
Berkeley-MHAD	–	–	3
UMPM	–	–	2
Fit3D	–	–	2
GPA	–	–	4
HumanSC3D	–	–	1
CHI3D	–	–	1
Human4D	–	–	1
MADS	–	–	2
Synthetic images			
SURREAL	32	8	5
3DPeople	–	6	4
JTA	–	5	3
HSPACE	–	5	3
SAIL-VOS	–	7	5
AGORA	–	5	3
SPEC	–	–	2
Real images with 2D annotations (weak supervision)			
COCO	8	8	8
MPII	8	8	8
PoseTrack	8	8	8
JRDB	8	8	8

In Table 4 (SOTA comparison) of the main paper, we use the more complex standard evaluation metrics. That is, for MuPoTS, here we use bone rescaling, normalized skeletons, and averaging is performed first per sequence and the final value is the average of per-sequence averages. In this, and also other details, we follow the same protocols as [5] (*e.g*., which joints to evaluate).

S5. Additional Ablations

Training Length. In Tab. S2, we study the effect of the length of training on the final model performance. Clearly, longer training can further improve the results and especially the correct pose score improves. Do note that every new line doubles the number of training steps, so this is expensive. Further, with long trainings we noted that training in 16-bit floating point (FP16) precision is unstable, as the activations tend to grow out of the representable range. The cause of this was that the convolutional kernels of the backbone grow in scale during gradient descent, since. Since they are always followed by BatchNorm in EfficientNetV2, the weight scale has no impact on the network output (though it has an effect on the effective learning rate [4]). We mitigated the problem by applying a max-norm constraint on the convolutional kernels to keep them from growing without bound. Some numerical instability remains in case of very long trainings, the cause of which needs more investigation. Tab. S3 shows that further extending the fine-tuning phase can bring minor performance benefits. For reasons of practicality, we chose 400k training steps and 40k fine-tuning step as the default setting for all of our experiments in the main paper, albeit one could achieve slightly better results with longer schedules.

Ghost BatchNorm. In Tab. S4 we show an ablation on using Ghost BatchNorm [3, 6]. We compare three options: normal BatchNorm, Ghost BN where the 96 3D annotated examples are normalized as one group and the 32 2D-labeled ones as another, and Ghost BN with ghost batch size 16. While the differences are not very large, the Ghost BN options tend to perform better. This is probably due to the discrepancies in BatchNorm statistics among datasets.

Inference-Mode BatchNorm Fine-Tuning. Furthermore, Tab. S4 also demonstrates that, when using Ghost BN, it is important to fine tune the network at the end in inference mode. By inference-mode fine-tuning, we mean that the BN layers use the stored, fixed statistics for normalization instead of the usual training mode of using the statistics of the current minibatch. In Ghost BN, the stored statistics may be suboptimal, since they are updated based on parts of the batch, instead of the overall batch statistics. A final fine-tuning in "inference mode" allows the network to fine-tune its weights to the setting that it will be used in during inference (*i.e.*, to adapt the weights to work well with the stored statistics).

S6. Effects of L1 Regularization in the ACAE

We point out in the main paper that using ℓ_1 regularization on the weight matrices of the affine-encoding autoencoder results both in sparsity and fewer negative weights. Here we elaborate on this connection. Since the weights produce affine combinations, they sum to one as specified in Eq. 1-2 in the main paper.

$$\sum_{j=1}^{J} w_{l,j}^{\text{enc}} = 1 \tag{1}$$

$$\sum_{l=1}^{L} w_{j,l}^{\text{dec}} = 1. \tag{2}$$

We can partition the weights to negative and non-negative ones.

$$w_{l,+}^{\text{enc}} = \sum_{j\,:\,w_{l,j}^{\text{enc}} \geq 0} w_{l,j}^{\text{enc}} \tag{3}$$

$$w_{l,-}^{\text{enc}} = \sum_{j\,:\,w_{l,j}^{\text{enc}} < 0} w_{l,j}^{\text{enc}} \tag{4}$$

$$w_{l,+}^{\text{enc}} + w_{l,-}^{\text{enc}} = 1, \tag{5}$$

and analogously for the decoder weights. Now, the ℓ_1 penalty (sum of absolute values) can be written as

$$\ell_1(w_{l,\cdot}^{\text{enc}}) = \sum_{j=1}^{J} \left| w_{l,j}^{\text{enc}} \right| = \tag{6}$$

$$= w_{l,+}^{\text{enc}} - w_{l,-}^{\text{enc}} = \tag{7}$$

$$= \left(1 - w_{l,-}^{\text{enc}}\right) - w_{l,-}^{\text{enc}} = \tag{8}$$

$$= 1 - 2 \cdot w_{l,-}^{\text{enc}} = \tag{9}$$

$$= 1 + 2 \cdot \left| w_{l,-}^{\text{enc}} \right|. \tag{10}$$

This means that the ℓ_1 penalty is equivalent to penalizing the absolute sum of the negative weights.

When all weights are non-negative, we get convex combinations. In other words, the ℓ_1 regularization in the ACAE encourages constructing close-to-convex combinations besides sparsity.

References

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[2] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[3] E. Hoffer, I. Hubara, and D. Soudry. Train longer, generalize better: Closing the generalization gap in large batch training of neural networks. In *NIPS*, 2017.

[4] T. van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

[5] I. Sárándi, T. Linder, K. O. Arras, and B. Leibe. MeTRAbs: metric-scale truncation-robust heatmaps for absolute 3D human pose estimation. *IEEE Trans. Biometr. Behav. Ident. Sci. (T-BIOM)*, 3(1):16–30, 2021.

[6] C. Summers and M. J. Dinneen. Four things everyone should know to improve batch normalization. In *ICLR*, 2019.

Table S2: Ablation for the length of training.

	MuPoTS-3D				3DPW				MPI-INF-3DHP				Human3.6M			
	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑
Initial, separate-skeleton model																
100k	88.6	62.4	67.8	60.0	65.9	47.0	81.5	65.3	66.3	51.2	82.2	71.7	47.7	37.6	91.9	86.6
200k	87.3	60.3	68.3	65.1	63.1	44.6	82.6	69.5	63.1	47.4	84.4	77.1	46.8	36.2	93.1	88.7
400k (default)	84.6	59.0	70.1	66.0	61.8	43.4	**83.8**	71.1	59.6	44.1	86.6	81.8	44.7	34.3	94.3	90.1
800k	82.9	57.8	70.5	69.8	61.7	42.6	83.7	73.1	58.8	42.9	87.3	83.0	43.0	33.2	94.8	91.4
1.6M	**81.6**	**56.7**	**71.6**	**72.8**	**61.5**	**41.9**	**84.4**	**74.2**	**58.6**	**41.3**	**87.8**	**86.3**	**41.5**	**32.3**	**95.5**	**92.1**
Fine-tuned with consistency regularization for 40k steps																
100k	85.5	60.6	70.3	66.6	65.0	46.0	81.8	66.9	63.6	48.7	83.9	74.1	46.7	36.4	92.5	87.6
200k	84.2	59.2	70.8	70.4	63.4	44.3	82.7	69.9	61.4	46.3	85.4	79.1	46.7	35.1	93.3	88.6
400k (default)	81.8	57.8	72.5	72.9	61.5	43.0	84.0	71.9	59.2	43.6	86.6	82.7	45.2	33.3	94.4	90.1
800k	80.5	56.8	72.7	74.4	61.3	42.1	84.5	73.3	**57.7**	42.2	87.7	84.3	42.0	31.9	95.3	91.4
1.6M	**79.7**	**56.1**	**73.3**	**76.6**	**60.6**	**41.7**	**84.7**	**74.5**	58.6	**41.1**	**87.8**	**86.6**	**41.1**	**31.2**	**95.7**	**92.2**

Table S3: Ablation for the length of consistency-regularized fine-tuning with an initial training length of 400k steps.

	MuPoTS-3D				3DPW				MPI-INF-3DHP				Human3.6M			
	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑
20k	82.0	58.0	72.2	72.1	61.7	43.0	83.9	71.3	59.3	43.6	86.5	82.1	44.9	33.5	94.3	89.8
40k (default)	81.8	**57.8**	**72.5**	72.9	61.5	43.0	84.0	71.9	59.2	43.6	86.6	82.7	45.2	**33.3**	94.4	**90.1**
80k	**81.6**	**57.8**	72.4	**73.2**	**61.4**	**42.9**	**84.1**	**72.1**	**58.4**	**43.2**	**87.2**	**82.9**	**44.5**	**33.3**	**94.6**	**90.1**

Table S4: Ablation for Ghost Batch Normalization and inference-mode fine-tuning for 1000 steps.

	MuPoTS-3D				3DPW				MPI-INF-3DHP				Human3.6M			
	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑	MPJPE↓	PMPJPE↓	PCK_{100}↑	CPS_{200}↑
*Initial, separate-skeleton model, **with** fine-tuning at the end with BN in inference mode*																
Normal BN	84.5	59.2	70.0	65.9	62.6	43.6	83.6	**71.9**	61.3	**43.7**	85.7	**82.0**	46.3	34.6	94.2	89.9
Ghost BN (3D/2D)	**83.6**	**58.7**	**70.4**	**70.0**	62.8	43.5	83.2	71.2	61.8	45.1	85.7	80.3	46.6	34.7	94.1	**90.5**
Ghost BN 16	84.6	59.0	70.1	66.0	**61.8**	**43.4**	**83.8**	71.1	**59.6**	44.1	**86.6**	81.8	**44.7**	**34.3**	**94.3**	90.1
*Initial, separate-skeleton model, **without** fine-tuning at the end with BN in inference mode*																
Normal BN	84.2	59.1	70.4	66.4	62.6	43.6	83.5	71.9	60.4	43.4	86.1	82.3	45.9	34.4	94.2	89.8
Ghost BN (3D/2D)	88.2	63.5	66.7	62.0	67.3	47.5	80.9	68.0	66.3	49.0	81.7	77.4	50.8	40.4	90.7	87.7
Ghost BN 16	85.8	60.4	69.0	63.5	63.4	44.7	83.2	70.8	59.8	44.3	86.3	81.8	45.4	35.7	93.6	89.6
*Fine-tuned with consistency regularization for 40k steps, **with** fine-tuning at the end with BN in inference mode*																
Normal BN	83.3	58.5	70.9	73.2	63.1	43.7	83.4	71.8	60.5	**43.4**	85.9	**82.7**	46.0	33.5	**94.4**	89.9
Ghost BN (3D/2D)	**81.2**	**57.7**	**72.5**	**74.0**	62.2	**43.0**	83.7	**72.3**	60.5	44.4	86.1	80.9	46.2	33.7	94.1	**90.6**
Ghost BN 16	81.8	57.8	**72.5**	72.9	**61.5**	**43.0**	**84.0**	71.9	**59.2**	43.6	**86.6**	**82.7**	**45.2**	**33.3**	**94.4**	90.1
*Fine-tuned with consistency regularization for 40k steps, **without** fine-tuning at the end with BN in inference mode*																
Normal BN	83.1	58.7	71.3	72.9	62.8	43.5	83.5	71.7	59.6	43.2	86.4	82.9	45.5	33.4	94.5	89.9
Ghost BN (3D/2D)	89.1	64.6	66.4	62.1	68.5	49.4	79.6	65.2	73.4	52.9	76.4	70.9	58.4	43.4	85.7	82.7
Ghost BN 16	84.0	60.0	70.7	69.7	63.3	45.3	82.7	69.4	63.2	45.7	83.8	80.1	49.0	36.6	92.2	88.3

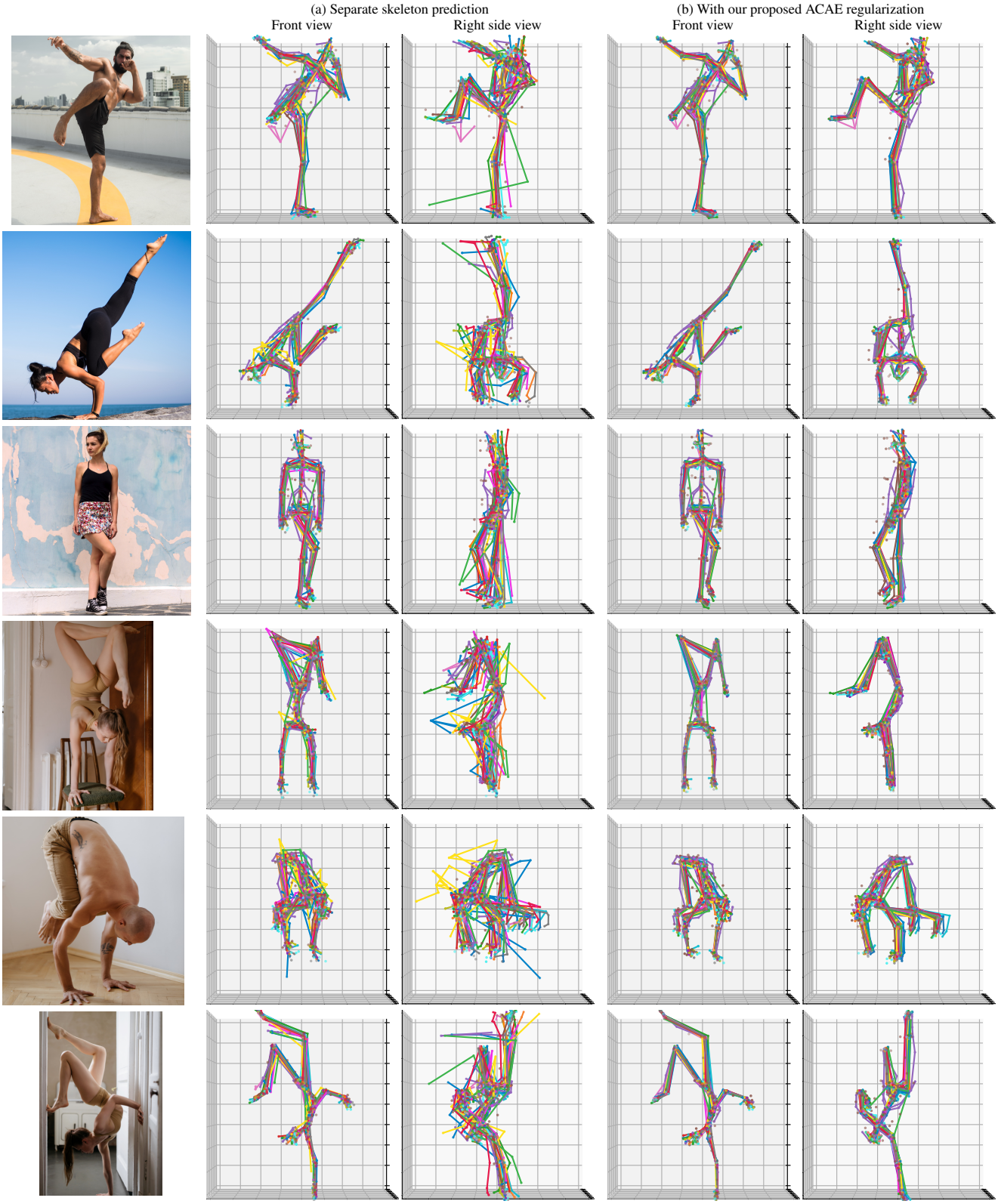

Figure S2: A qualitative result comparison between a model trained without (a) and with our ACAE regularization (b). It can clearly be seen that our regularization leads to improved skeleton consistency.

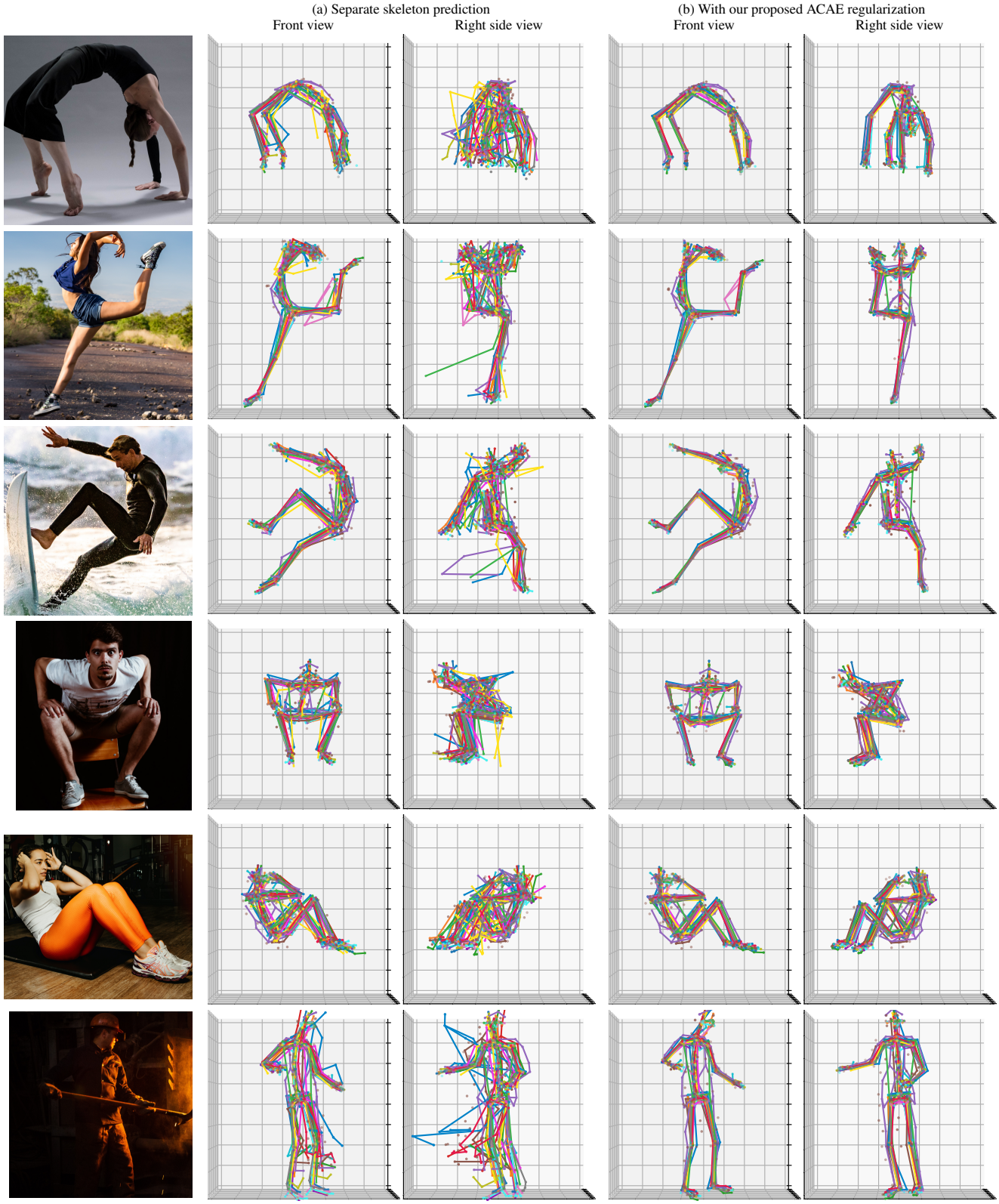

Figure S3: A qualitative result comparison between a model trained without (a) and with our ACAE regularization (b). It can clearly be seen that our regularization leads to improved skeleton consistency.

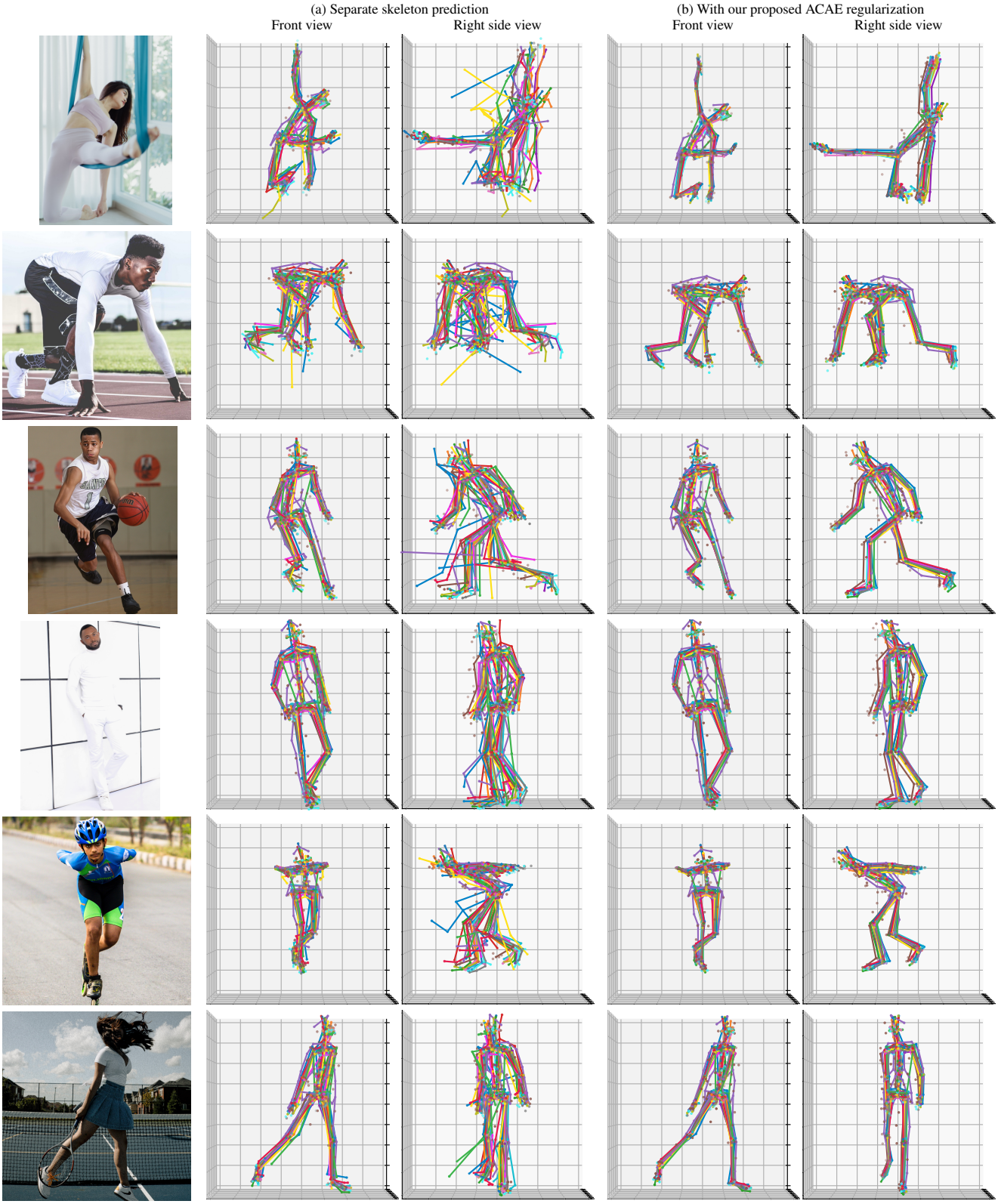

Figure S4: A qualitative result comparison between a model trained without (a) and with our ACAE regularization (b). It can clearly be seen that our regularization leads to improved skeleton consistency.