Learning to Fit Morphable Models

Vasileios Choutas[1, 2, †] Federica Bogo[2,*] Jingjing Shen[2] Julien Valentin[2]

[1]Max Planck Institute for Intelligent Systems, Tübingen, Germany, [2]Microsoft
vchoutas@tue.mpg.de, fbogo@fb.com,
{jinshen, valentin.julien}@microsoft.com

1 Social impact

Accurate tracking is a necessary pre-requisite for the next generation of communication and entertainment through virtual and augment reality. Learned optimizers represent a promising avenue to realize this potential. However, it can also be used for surveillance and tracking of private activities of an individual, if the corresponding sensor is compromised.

2 Errors per iteration

Figure 2 shows the metric values per iteration, averaged across the test set, for our fitter on the task of fitting SMPL+H to HMD head and hand signals. Different to the main paper, this figure corresponds to the full visibility scenario, i.e. the hands are always visible. The learned fitter aggressively optimizes the target data term and quickly converges to the minimum.

3 Update rule

In addition to the update rule described in Eq. 1 of the main paper, we investigated two other alternatives, based on the convex combination of the network update and gradient descent. The first is a simple re-formulation of Eq. 1, with $\boldsymbol{\lambda} \in [0, 1]$, selecting either the network update or the gradient descent direction. In the second, we first compute a convex combination between the normalized network update and gradient descent, i.e. selecting a direction, and then scale the computed direction according to $\boldsymbol{\gamma}$.

$$
\begin{aligned}
u(\Delta\boldsymbol{\Theta}_n, \boldsymbol{g}_n, \boldsymbol{\Theta}_n) &= \boldsymbol{\lambda}\Delta\boldsymbol{\Theta}_n + (1 - \boldsymbol{\lambda})\,(-\boldsymbol{\gamma}\boldsymbol{g}_n) \\
u(\Delta\boldsymbol{\Theta}_n, \boldsymbol{g}_n, \boldsymbol{\Theta}_n) &= \boldsymbol{\gamma}\left[\boldsymbol{\lambda}\left(\frac{\Delta\boldsymbol{\Theta}_n}{\|\Delta\boldsymbol{\Theta}_n\|}\right) + (1 - \boldsymbol{\lambda})\left(\frac{-\boldsymbol{g}_n}{\|\boldsymbol{g}_n\|}\right)\right] \\
\boldsymbol{\lambda} &= \sigma\left(f_\lambda(\mathcal{R}(\boldsymbol{\Theta}_n), \mathcal{R}(\boldsymbol{\Theta}_n + \Delta\boldsymbol{\Theta}_n)), \boldsymbol{\lambda} \in \mathbb{R}^{|\boldsymbol{\Theta}|}\right)
\end{aligned}
\tag{1}
$$

[†]Work performed at Microsoft. * Now at Meta Reality Labs Research.

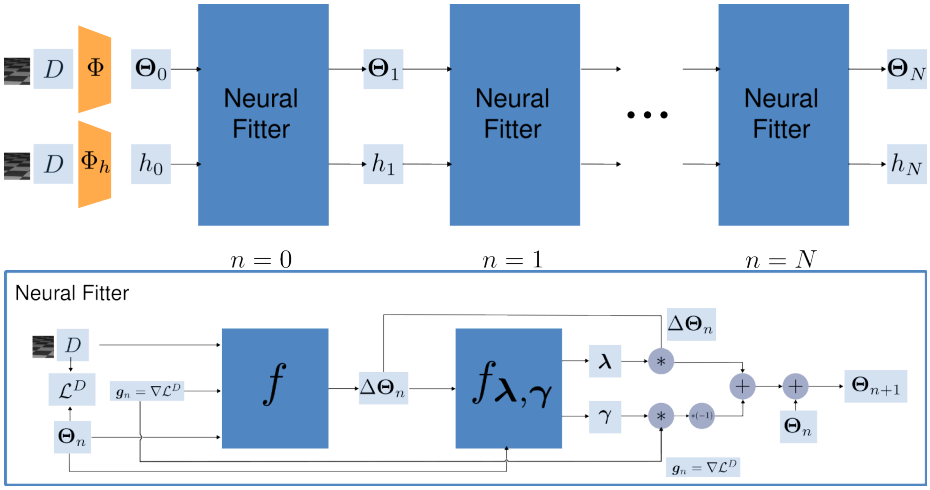

Fig. 1: Top: the general fitting process described in Alg. 1. Bottom: A schematic representation of our update rule, described in Eq. 1, 2 of the main paper.

Here, $\sigma()$ is the sigmoid function: $\sigma(x) = \frac{1}{1+\exp(-x)}$. The learning rate of the gradient descent term is the same as the main text:

$$\boldsymbol{\gamma} = f_{\gamma}(\mathcal{R}(\boldsymbol{\Theta}_n), \mathcal{R}(\boldsymbol{\Theta}_n + \boldsymbol{\Delta\Theta}_n)), \boldsymbol{\gamma} \in \mathbb{R}^{|\boldsymbol{\Theta}|} \tag{2}$$

We empirically found that the performance of these two variants is inferior to the proposed update rule, but we nevertheless list them for completeness.

4 Additional ablation

Table 1 contains an additional ablation experiment, where we compare different options for the type of variable for $\boldsymbol{\lambda}$, $\boldsymbol{\gamma}$, namely whether to use a scalar or a vector variable, and and whether to use a common network predictor for $\boldsymbol{\lambda}$, $\boldsymbol{\gamma}$. We use the problem of fitting SMPL to 2D keypoint predictions, evaluating our results using the 3DPW test set.

5 Qualitative comparisons

We present a qualitative comparison of the proposed learned optimizer with a classic optimization-based method in Fig. 3. Without explicit hand-crafted constraints, the classic approach cannot resolve problems such as ground-floor penetration. Formulating a term to represent this constraint is not a trivial process. Furthermore, tuning the relative weight of this term to avoid under-fitting the data term is not a trivial process. Our proposed method on the other hand can learn to handle these constraints directly from data, without any heuristics.

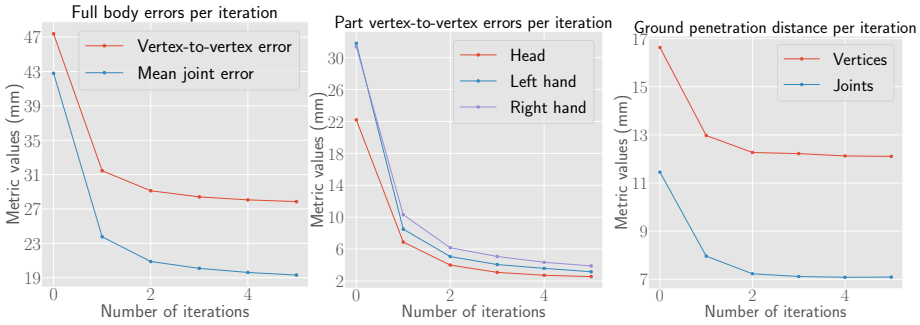

Fig. 2: Errors per iteration when fitting SMPL+H to HMD data, assuming that the hands are always visible. From left to right: 1) Full body vertex and joint errors, 2) head, left and right hand V2V errors and 3) vertex and joint ground distance, computed on the set of points below ground.

Table 1: Predicting vector values for $\boldsymbol{\lambda}, \boldsymbol{\gamma}$ is always better than scalars. This is expected, since each variable to be optimized has different scale and the learned fitter must adapt its predicted updates accordingly. Having a shared network for $\boldsymbol{\lambda}, \boldsymbol{\gamma}$ improves performance and lowers the number of parameters of the learned fitter.

Vector $\boldsymbol{\lambda}$	Vector $\boldsymbol{\gamma}$	Shared network for $\boldsymbol{\lambda}, \boldsymbol{\gamma}$	PA-MPJPE (mm)
✓	✗	✗	52.8
✗	✓	✗	52.7
✓	✓	✗	52.3
✓	✓	✓	**52.2**

6 Training details

6.1 GRU formulation

All our recurrent networks are implemented with Gated Recurrent Units (GRU) [3], with layer normalization [1]:

$$
\begin{aligned}
z_n &= \sigma_g\left(LN(W_z x) + LN(U_z h_{n-1})\right) \\
r_n &= \sigma_g\left(LN(W_r x) + LN(U_r h_{n-1})\right) \\
\hat{h}_n &= \phi_h\left(LN(W_h x) + LN(U_h\left(r_n \odot h_{n-1}\right))\right) \\
h_n &= (1 - z_n) \odot h_{n-1} + z_n \odot \hat{h}_n, \quad h_0 = \Phi_h\left(D\right)
\end{aligned}
\tag{3}
$$

We also tried replacing the GRUs with LSTMs [6], but did not observe significant performance benefit. Hence we chose the computationally lighter GRUs.

6.2 Training losses

We apply a loss on the output of every step of our network:

$$\mathcal{L}(\{\boldsymbol{\Theta}_n\}_{n=0}^N, \{\hat{\boldsymbol{\Theta}}_n\}_{n=0}^N; D) = \sum_{i=0}^N \mathcal{L}_i(\boldsymbol{\Theta}_i, \hat{\boldsymbol{\Theta}}_i; D) \tag{4}$$

The loss $\mathcal{L}_i$ contains the following terms:

$$\mathcal{L}_i = \lambda_M \mathcal{L}_i^M + \lambda_{\mathcal{E}} \mathcal{L}_i^{\mathcal{E}} + \lambda_T \mathcal{L}_i^T + \lambda_{\boldsymbol{\theta}} \mathcal{L}_i^{\boldsymbol{\theta}} \tag{5}$$

$$\mathcal{L}_i^M = \|\hat{M} - M\|_1 \tag{6}$$

$$\mathcal{L}_i^{\mathcal{E}} = \sum\nolimits_{(i,j)\in\mathcal{E}} \|(\hat{M}_i - \hat{M}_j) - (M_i - M_j)\|_1 \tag{7}$$

$$\mathcal{L}_i^T = \sum\nolimits_{j=1}^J \|\hat{T}_j - T_j\|_1 \tag{8}$$

$$\mathcal{L}_i^{\boldsymbol{\theta}} = \|\hat{R}_{\boldsymbol{\theta}} - R_{\boldsymbol{\theta}}\|_1 + \|\hat{\boldsymbol{t}} - \boldsymbol{t}\|_1 \tag{9}$$

M represents the mesh vertices deformed by parameters $\boldsymbol{\Theta}$. $\mathcal{E}$ is the set of vertex indices of the mesh edges. T denotes the transformations in world coordinate while $R_{\boldsymbol{\theta}}$ denotes the rotation matrices (in the parent-relative coordinate frame) computed from the pose values $\boldsymbol{\theta}$. $\boldsymbol{t}$ is the root translation vector. We use the following values for the weights of the training losses: $\lambda_M = 1000$, $\lambda_{\mathcal{E}} = 1000$, $\lambda_T = 100$, $\lambda_{\boldsymbol{\theta}} = 1$, $\lambda_{\boldsymbol{t}} = 100$.

6.3 Datasets

For body fitting from HMD signals, we use a subset of AMASS [8] to train and test our method. Specifically, we use CMU [2], KIT [9] and MPI_HDM05 [11], adopting the same pre-processing and training, test splits as [4]. An important difference is that we fit the neutral SMPL+H to the gendered SMPL+H data found in AMASS, to preserve correct contact with the ground and avoid the use of heuristics [13]. We attach random hand poses from the MANO [14] training set to simulate hand articulation. In all our experiments that involve SMPL+H, we use the ground-truth shape parameters $\boldsymbol{\beta}$. Future work could include estimating a subset of the shape parameters corresponding to height from the position of the headset. For the learned fitter that estimates body parameters from 2D joints, we use the data, augmentation and evaluation protocol of Song et al. [15]. To be more precise, we use AMASS [8] to train the fitter and evaluate the resulting model on 3DPW [10], which contains sequences of subjects in complex poses in outdoor scenes, along with SMPL parameters captured using RGB cameras and IMUs.

For face fitting from 2D landmarks, we use the face model proposed in [16] to generate a synthetic face dataset by sampling 50000 sets of parameters from the model space. For each sample, we vary pose, identity and expression. We use a perspective camera with focal length $(512, 512)$ and principal point $(256, 256)$ (in pixels) to project the 3D landmarks onto the image for 2D landmarks. Afterwards, we randomly split this by 80/20 into training and testing sets.

6.4 Training schedule

We implement our model in [12] and train it with a batch size of 512 on 4 GPUs using Adam [7]. We anneal the learning rate by a factor of 0.1 after 400 epochs. We apply dropout with a probability of $p = 0.5$ on the hidden states of the GRUs. We initialize the weights of the output linear layer of Eq. 3 with a gain equal to 0.01 [5].

6.5 Edge loss

We empirically observed that the loss between the 3D edges of the predicted and ground-truth meshes helps training converge faster.

6.6 Runtimes

We measure time on the 2D keypoint fitting problem on a Quadro P5000 GPU and with a batch size of 512 data points. Our extra networks and update rule add 6 (ms) per iteration to LGD's [15] runtime. Using a common network for $\boldsymbol{\gamma}$ and $\boldsymbol{\lambda}$ reduces this to 4 (ms).

6.7 Number of iterations

Similar to LGD [15], we observe limited gains beyond 5 iterations. Training with more iterations, e.g. 10 or 20, leads to similar performance, at the cost of increased training time. Picking a random number of iterations during training, e.g. 5 to 20, does not affect the final result.

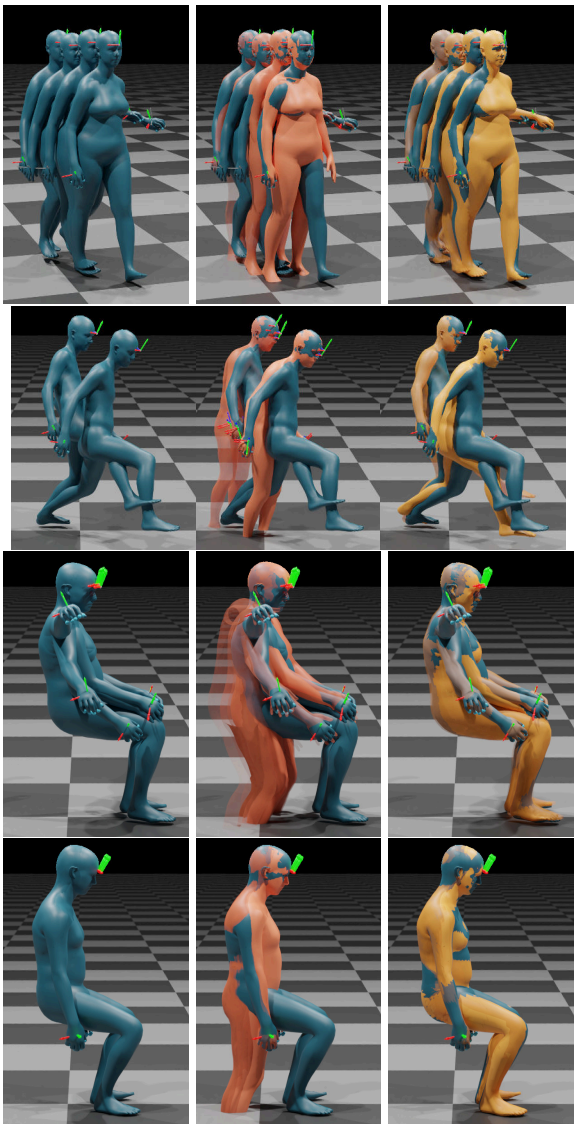

Fig. 3: Comparison of our learned fitter with a Levenberg-Marquardt based optimization method. Left to right: 1) Input HMD data and Ground-Truth mesh (blue), 2) LM solution (orange) overlayed on the GT, 3) our solution (yellow) overlayed on the GT. While the classic LM optimization successfully fits the input data, it still needs hand-crafted priors to prevent ground floor penetration. In contrast, our proposed fitter learns from the data to avoid such penetrations.

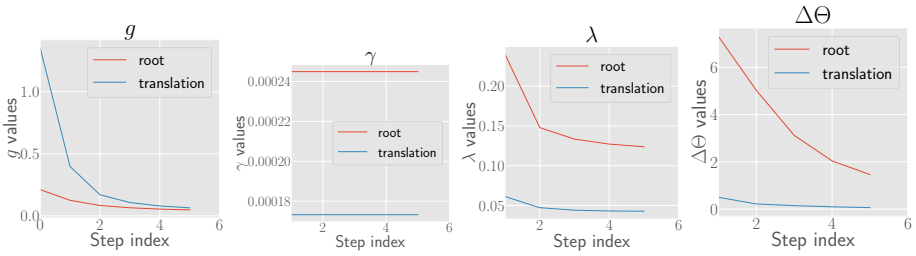

Fig. 4: Average norm for (left to right) 1) $\|\boldsymbol{g}_n\|_2$, 2) $\|\boldsymbol{\gamma}\|_2$, 3) $\|\boldsymbol{\lambda}\|_2$ and 4) $\|\boldsymbol{\Delta\Theta}_n\|_2$, computed across the test set, for the root rotation and translation. The learned optimizer slows down as it approaches a minimum of the target data term.

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
2. Carnegie Mellon University: CMU MoCap Dataset, http://mocap.cs.cmu.edu
3. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1179, https://aclanthology.org/D14-1179
4. Dittadi, A., Dziadzio, S., Cosker, D., Lundell, B., Cashman, T.J., Shotton, J.: Full-Body Motion From a Single Head-Mounted Device: Generating SMPL Poses From Partial Observations. In: International Conference on Computer Vision (ICCV). pp. 11687–11697 (October 2021)
5. 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. JMLR Workshop and Conference Proceedings (2010)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015), http://arxiv.org/abs/1412.6980
8. Mahmood, N., Ghorbani, N., Troje, N.F., Pons-Moll, G., Black, M.J.: AMASS: Archive of motion capture as surface shapes. In: International Conference on Computer Vision (ICCV). pp. 5442–5451 (October 2019)
9. Mandery, C., Terlemez, O., Do, M., Vahrenkamp, N., Asfour, T.: The KIT whole-body human motion database. In: 2015 International Conference on Advanced Robotics (ICAR). pp. 329–336 (Jul 2015). https://doi.org/10.1109/ICAR.2015.7251476
10. von Marcard, T., Henschel, R., Black, M., Rosenhahn, B., Pons-Moll, G.: Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera. In: European Conference on Computer Vision (ECCV). pp. 614–631 (September 2018)
11. Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., Weber, A.: Documentation mocap database HDM05. Tech. Rep. CG-2007-2, Universität Bonn (Jun 2007)
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
13. Rempe, D., Birdal, T., Hertzmann, A., Yang, J., Sridhar, S., Guibas, L.J.: HuMoR: 3D Human Motion Model for Robust Pose Estimation. In: International Conference on Computer Vision (ICCV). pp. 11468–11479 (October 2021)
14. Romero, J., Tzionas, D., Black, M.J.: Embodied hands: Modeling and capturing hands and bodies together. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **36**(6) (November 2017)

15. Song, J., Chen, X., Hilliges, O.: Human Body Model Fitting by Learned Gradient Descent. In: European Conference on Computer Vision (ECCV). pp. 744–760 (August 2020)
16. Wood, E., Baltrušaitis, T., Hewitt, C., Dziadzio, S., Johnson, M., Estellers, V., Cashman, T.J., Shotton, J.: Fake It Till You Make It: Face Analysis in the Wild Using Synthetic Data Alone. In: International Conference on Computer Vision (ICCV). pp. 3681–3691 (October 2021)