VINECS: Video-based Neural Character Skinning

Supplementary Material

Zhouyingcheng Liao[1,2] Vladislav Golyanik[1] Marc Habermann[1,3] Christian Theobalt[1,3]
[1]Max Planck Institute for Informatics, Saarland Informatics Campus [2]The University of Hong Kong
[3]Saarbrücken Research Center for Visual Computing, Interaction and AI

In the following, we show quantitative and qualitative comparisons against ARAH [18] and TAVA [9] (Sec. A). Then, we provide more details concerning our method (Sec. C-F). We also provide additional information about how the competing approaches are trained and evaluated (Sec. G). Moreover, we provide more qualitative results that compare our design choice of using NeuS [17] for template generation with classical multi-view stereo reconstruction [2] (Sec. I). Lastly, we demonstrate that our method can pose template meshes of varying resolution without re-training (Sec. J).

A. Additional Comparisons

We compare VINECS against ARAH [18] and TAVA [9], which are animatable human volume rendering methods, on the subjects D2 and D5. We provide the quantitative results in Tab. 1 and the qualitative results in Fig. 1. The evaluation protocol is the same one as described in the main manuscript. Their reconstruction error is significantly higher than the one of VINECS. Interestingly, their error is also higher compared to using initial skinning weights. We found two major reasons for this: 1) The DynaCap dataset [6] is significantly more challenging than the datasets used for evaluation in their works, and their methods seem to not scale well to more complex conditions (motions, lighting). 2) For TAVA, we found that they use a density-based representation for which it is hard to extract accurate geometry. In contrast, our approach can scale well and achieves a higher accuracy.

We would also like to highlight that their setting and goal is different from ours. Their goal is to obtain an implicit animatable human model for volume rendering, whilst ours is to obtain pose-dependent skinning for explicit human animation.

B. Normal Consistency

We evaluate normal consistency and show results in Table 2. We highlight that oversmoothed results have higher normal consistency than high-frequency results that are slightly misaligned in image space. This explains the less pronounced improvement compared to SCANimate (requiring 4D scans). Nonetheless, results are coherent with the observations in the main paper, i.e., our method demonstrates a clear improvement.

C. Human Parsing Labels

To obtain the per-vertex human parsing labels, we first apply a 2D human parsing method [8] pre-trained on the LIP dataset on renderings from multiple views. More specifically, we render the template mesh for one frame, animated by the initial skinning weights colored by the texture obtained by NeuS [17]. We apply ambient lighting for the rendering. We found for certain views, especially views from the top, the human parsing method often failed. Thus, we discarded such views. Then, we run the human parsing method for the rendering of the remaining views. For each view, we can obtain a label for each vertex by finding the label of the nearest 2D pixel from its projection, and we perform max-voting to obtain the final label. Note that if the label with the most votes is the background, we select the label with the second most votes instead. After that, we iteratively run a mode filter within one-ring neighborhood until no vertex is labeled as the background. Originally, the method of Li *et al.* [8] predicts 20 classes. We merge these classes and only keep two classes for our training, *i.e.,* the skin and the clothes.

D. Network Architectures

All the neural networks used in this paper, namely SkinNet, AlbedoNet, and ReflectanceNet, are based on the coordinate-based multi-layer perceptrons (MLP) sharing the same set of hyper-parameters. The network contains five layers with $256, 256, 128, 256$ and 256 neurons in them, respectively. There is a skip connection from the input to the third layer. Inspired by [5], we use SoftPlus as the activation function. When the query point is fed into the network, we compute its positional encoding [12] and concatenate it with the 3D coordinate. In addition, we re-parameterize the network weights using the weight normalization [14].

For SkinNet, a SoftMax layer is applied on the output of the MLP, along the dimension of joints, to ensure that the output skinning weights satisfy the partition of unity. For ReflectanceNet and for AlbedoNet, there is no processing of the output during training, while during inference, we clip the values so that they are in the range $[0; 1]$. As for the output of ReflectanceNet, we compute its exponential as the final scalar multiplier.

Quantitative Geometry Comparison						
Subject	D2			D5		
Method	**Chamfer↓**	**M2S↓**	**S2M↓**	**Chamfer↓**	**M2S↓**	**S2M↓**
Initial weights [3]	3.760	2.162	1.599	5.077	2.811	2.267
ARAH [18]	3.086	1.791	1.294	4.900	2.746	2.153
TAVA [9]	5.369	3.017	2.351	6.823	3.714	3.109
Ours	**3.034**	**1.746**	**1.288**	**4.512**	**2.442**	**2.070**

Table 1. We further compare our method to ARAH [18] and TAVA [9] on D2 and D5. As a reference, we also show the results with initial skinning weights obtained from Pinocchio [3]. Our method clearly outperforms both approaches and the baseline as TAVA [9] leverages a density-based surface representation that usually fails to model high quality geometry, and both cannot scale to the more difficult DynaCap dataset, which contains significantly more frames and pose variations as the dataset used in their work.

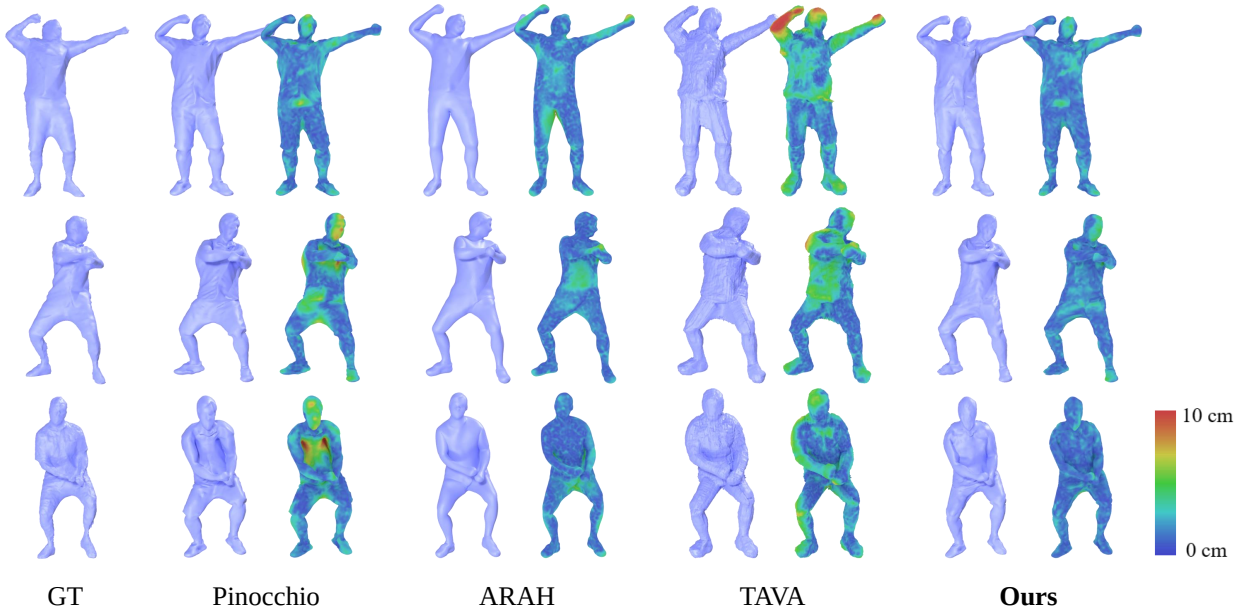

Figure 1. Qualitative comparison. For each method, we visualize the recovered posed geometry as well as the per-vertex error map when comparing the ground truth in terms of M2S. Note that our method consistently shows the lowest error and also has the least visual artifacts.

Subject	D2	D5	V6
Pinocchio	0.516	0.354	0.561
SCANimate*	0.543	0.401	0.599
RigNet	0.499	0.319	0.544
Ours	0.554	0.405	0.588
SCAnimate	0.565	0.439	0.641

Table 2. Normal consistency. Higher values mean better consistency.

E. Silhouette Loss

It is a bidirectional loss, which aims to align the projected boundary of the mesh to the foreground mask $\mathbf{M}$:

$$\mathscr{L}_{silh} = \sum_{c=1}^{C} \Big(\sum_{i \in B_c} \| \, d_c(\pi_c(\mathbf{x}_i)) \, \|^2 + \\ + \sum_{\mathbf{p} \in \{\mathbf{u} \in \mathbb{R}^2 \| d_c(\mathbf{u}) = 0\}} \| \, \pi_c(\mathbf{x_p}) - \mathbf{p} \, \|^2 \Big). \quad (1)$$

Specifically, the first term pushes boundary vertices B_c to the boundary of the foreground mask, where the distance transform value d_c equals zero. In the second term, we minimize the distance between every pixel $\mathbf{p}$ on the zero contour of the distance transform map and its closest projected vertex $\mathbf{x_p}$.

F. Training Details

We implement our method using TensorFlow [1]. All experiments are performed on a single NVIDIA A40 GPU (48GB). Our training consists of four stages. During the first stage, we train SkinNet alone without the rendering loss for 50000 iterations. Next, we train AlbedoNet for 5000 iterations and ReflectanceNet for 5000 iterations. Lastly, SkinNet is refined with the pre-trained appearance field for 20000 iterations. The whole training takes around 18 hours.

For all training stages, the network weights are optimized using Adam [7]. We clip the gradient values to $[-1, 1]$. The learning rate is 0.001 and the batch size is 4.

G. Competing Methods

Pinocchio [3]. We re-implement Pinocchio using Python. Since in our work, the skeleton is obtained from [15], we only use its skin attachment part to compute the skinning weights based on the skeleton. We use the library SciPy [16] to solve the sparse linear system for the heat equilibrium equation in Pinocchio.

SCANimate [13]. SCANimate requires the registered SMPL [10] pose and shape parameters for all training scans. As we already had the pose tracking of the training sequence, we can animate our template mesh using the initial skinning weights, and the animated meshes can roughly align the scans. Thus, we manually label 30 correspondence points between our template mesh and SMPL template and optimize SMPL parameters by fitting to the correspondence points on our animated template mesh. With the paired scan and SMPL parameter, we train SCANimate for each of our characters.

SCANimate* [13]. Since the focus of our paper is to learn the skinning weights, instead of the pose-aware shape, we test SCANimate without the pose-aware shape. More specifically, during inference, we input the canonical pose parameter to the pose-dependent geometry module to obtain a canonical shape, which we keep constant for all poses. We use this canonical shape as the query for the forward skinning network to obtain the skinning weights, which then animates the canonical shape by LBS.

RigNet [19]. We ran RigNet pre-trained on "ModelsResource-RigNetv1" dataset on our template mesh to obtain the skinning weights. Similar to Pinocchio, we only use the skinning prediction module, which takes a mesh and the aligned skeleton and predicts the skinning weights.

ARAH [18]. We train ARAH using the same hyperparameter settings as in their public codes. It is trained for 1.5 days on 4 NVIDIA A40 GPUs, which is much more expensive than the training of VINECS. We extract the mesh from the SDF of ARAH using Marching Cube [11] with the resolution of 256^3.

Figure 2. We apply the same checkerboard texture [4] to our results of different poses. It can be seen clearly that our results well preserve the geometric fidelity of the mesh.

TAVA [9]. We train TAVA following the same setting as in their public code. The whole training takes 30 hours on a single NVIDIA A40 GPU. The mesh is extracted from the density field using Marching Cube with the resolution of 256^3.

H. Rendering with Checkerboard Texture

To better visualize the surface deformation, we additionally render our results with a checkerboard texture. We manually unwrap the canonical mesh into the UV space and apply the checkerboard texture. Then, we keep the UV unwrapping and apply the same checkerboard texture to different poses. As shown in Figure 2, our results well preserve the geometric fidelity of the mesh, ensuring that the checkerboard texture remains uniformly distributed across the surface without noticeable distortion.

I. MVS vs. NeuS

We choose NeuS instead of classical multi-view stereo reconstruction to extract the template mesh because we found in most cases NeuS generates more high-frequency details while introducing less noise (see Fig. 3).

J. Multi-resolution Results

Our method supports multi-resolution character skinning because SkinNet is an implicit function and can take arbitrary 3D positions as input. During training, we only input the vertices of the template mesh, which has a fixed resolution of around 10K vertices, to SkinNet, because our supervision requires an explicit mesh. However, even though only trained on a fixed resolution, our method generalizes well to different resolutions. We re-sample the original mesh generated by NeuS to different numbers of vertices (1K, 5K, 10K, 50K). Then, they are fed into the same pre-trained model and animated. All meshes deform naturally and have low 3D errors (Fig. 4).

Thus, in practice, we query the spatial point at vertex level, which suffices for supervising our skinning weights

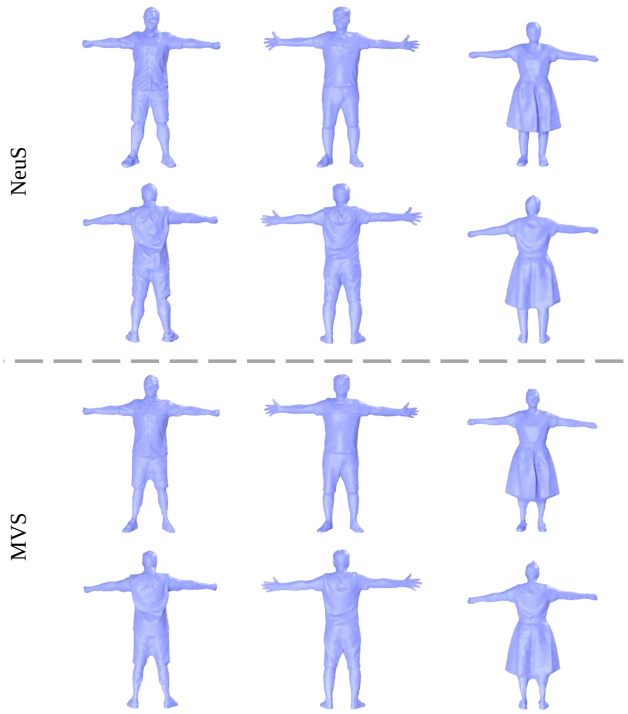

Figure 3. NeuS [17] (top) vs. Multi-view stereo reconstruction [2] (bottom). In most cases, NeuS generates more high-frequency details (cloth wrinkles) while introducing less noise (e.g., around the calf).

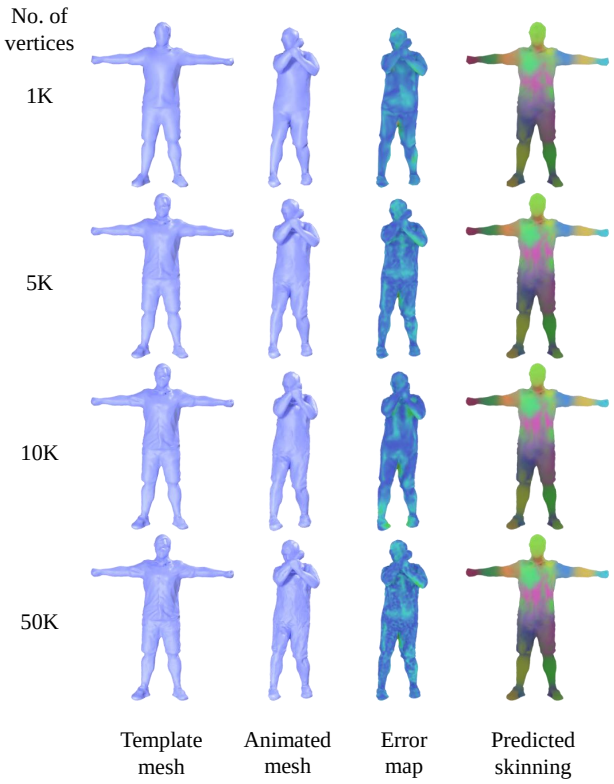

Figure 4. Multi-resolution results of our method. Even though trained on the fixed resolution (10K), our method generalizes well to other resolutions. Note that for different resolutions, the predicted skinning weights are very similar and the error always stays low.

field *during training*. However, we would like to emphasize that our model is invariant to specific mesh connectivity and resolution *during inference*, i.e. we do not require transferring per-vertex skinning weights from one specific mesh to another one as skinning weights can be continuously queried due to the field-based formulation.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 3

[2] Agisoft. PhotoScan. http://www.agisoft.com, 2016. 1, 4

[3] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3), 2007. 2, 3

[4] NewTek Forums. UV map - Generator. https://forums.newtek.com/threads/uv-map-generator.167156, 2024. 3

[5] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of Machine Learning and Systems 2020*, pages 3569–3579. 2020. 1

[6] Marc Habermann, Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Real-time deep dynamic characters. *ACM Trans. Graph.*, 40(4), 2021. 1

[7] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014. 3

[8] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. Self-correction for human parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1

[9] Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhofer, Jurgen Gall, Angjoo Kanazawa, and Christoph Lassner. Tava: Template-free animatable volumetric actors. 2022. 1, 2, 3

[10] Matthew Loper, Naureen Mahmood, Javier Romero, Ger-

ard 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. 3

[11] William Lorensen and Harvey Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21:163–, 1987. 3

[12] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Computer Vision – ECCV 2020*, pages 405–421, Cham, 2020. Springer International Publishing. 1

[13] Shunsuke Saito, Jinlong Yang, Qianli Ma, and Michael J. Black. SCANimate: Weakly supervised learning of skinned clothed avatar networks. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[14] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016. 1

[15] TheCaptury. The Captury. http://www.thecaptury.com/, 2020. 3

[16] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 3

[17] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 1, 4

[18] Shaofei Wang, Katja Schwarz, Andreas Geiger, and Siyu Tang. Arah: Animatable volume rendering of articulated human sdfs. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, page 1–19, Berlin, Heidelberg, 2022. Springer-Verlag. 1, 2, 3

[19] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *ACM Trans. on Graphics*, 39, 2020. 3