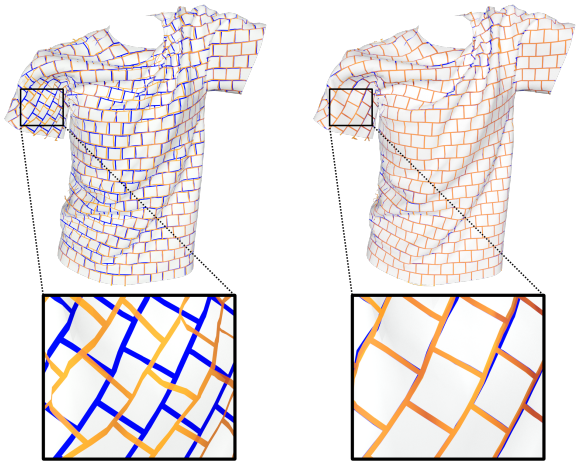
Recovering Geometric Information with Learned Texture Perturbations

Yongxu Jin (yxjin@stanford.edu)
Advisors: Prof. Ron Fedkiw, Jane Wu



(a) inferred cloth (b) texture sliding

Figure 1: Texture coordinate perturbations (texture sliding) reduce shape inference errors: ground truth (blue), prediction (orange).

1. Introduction

Since machine learning models (like neural networks) are trained to generalize to unseen data, regularization is important for reducing overfitting, see *e.g.* [41]. However, regularization also removes some of the high variance characteristic of much of the physical world. Even though high-quality ground truth data can be collected or generated to reflect the desired complexity of the outputs, regularization will inevitably smooth model predictions. Rather than attempting to directly infer high-frequency features, we alternatively propose to learn a low-frequency space in which such features can be embedded.

We focus on the specific task of adding high-frequency wrinkles to virtual clothing, noting that the idea of learning a low-frequency embedding may be generalized to other tasks. Because cloth wrinkles/folds are high-frequency features, existing deep neural networks (DNNs) trained to infer cloth shape tend to predict overly smooth meshes [1, 22, 25, 29, 40]. Rather than attempting to amend such errors directly, we perturb texture so that the rendered cloth mesh *appears* to more closely match the ground truth. See Figure 1. Then given texture perturbations from at least two unique camera views, 3D geometry can be accurately reconstructed [19] to recover high-frequency wrinkles. Similarly, for AR/VR applications, correcting visual appearance from two views (one for each eye) is enough to allow the viewer to accurately discern 3D geometry.

Our proposed texture coordinate perturbations are highly dependent on the camera view. Thus, we demonstrate that one can train a separate machine learning model, like support vector regression (SVR), and texture sliding neural network (TSNN), for each of a finite number of cameras laid out into an array and use nearby models to interpolate results valid for any view enveloped by the array. The input of our machine learning models is the joint angles of the human skeleton pose θ (similar to [22]), and the output of our machine learning models is the per-vertex (or per-pixel) displacement of texture coordinate $d(\theta)$.

Although an approach similar in spirit might be pursued for various lighting conditions, this limitation is left as future work since there are a great deal of applications where the light is ambient/diffuse/non-directional/etc. In such situations, this further complication may be ignored without significant repercussion.

2. Related Work

While physically-based cloth simulation has matured as a field over the last few decades [8, 9, 42], data-driven methods are attractive for many applications. There is a rich body of work in reconstructing cloth from multiple views or 3D scans, see *e.g.* [7, 15, 44]. More recently, optimization-based methods have been used to generate higher resolution reconstructions [20, 35, 46, 48]. Some of the most interesting work focuses on reconstructing the body and cloth separately [6, 30, 49, 51].

With advances in deep learning, one can aim to reconstruct 3D cloth meshes from single views. A number of approaches reconstruct a joint cloth/body mesh from a single RGB image [1, 4, 29, 39], RGB-D image [50], or video [2, 3, 18, 47]. To reduce the dimensionality of the output space, DNNs are often trained to predict the pose/shape parameters of human body models such as SCAPE [5] or

SMPL [26] (see also [33]). [1, 2, 3] predict SMPL model parameters along with per-vertex offsets to add details, and [4] refines the mesh using the network proposed in [21]. [18, 29, 43] leverage predicted pose information to infer shape. Estimating shape from silhouettes given an RGB image has also been explored [12, 13, 29]. When only the garment shape is predicted, a number of recent works output predictions in UV space to represent geometric information as pixels [11, 22, 25], although others [17, 40] define loss functions directly in terms of the 3D cloth vertices.

Cloth realism can be improved by introducing wrinkles and folds. In the graphics community, researchers have explored both procedural and data-driven methods for generating wrinkles [28, 38, 45]. Other works add real-world wrinkles as a postprocessing step to improve smooth captured cloth: [36] extracts the edges of cloth folds and then applies space-time deformations, [37] solves for shape deformations directly by optimizing over all frames of a video sequence. Recently, [25] used a conditional Generative Adversarial Network [27] to generate normal maps as proxies for wrinkles on captured cloth.

3. Dataset Generation

Given an input pose θ, we can generate the cloth mesh of this pose either by physically-based simulation [14] or data-driven method [22]. We call the simulation cloth mesh 'ground truth' and the data-driven cloth mesh 'inferred'. Then we generate our dataset by sliding the texture coordinate on the inferred cloth to make the inferred cloth visually resembles ground truth from a specific camera view. Specifically, the dataset is generated for supervised learning. The input feature is the rotation matrix of each joint in the skeleton pose (denote as θ), and the output variable is the texture coordinate displacement of the cloth after texture sliding from camera view v_p (denote as $d_{v_p}(\theta)$). We formulate our dataset generation process below.

Let $C = \{X, T\}$ be a cloth triangulated surface with n vertices $X \in \mathbb{R}^{3n}$ and texture coordinates $T \in \mathbb{R}^{2n}$. We assume that mesh connectivity remains fixed throughout. The ground truth cloth mesh $C_G(\theta) = \{X_G(\theta), T_G\}$ depends on the pose θ. Given a pre-trained DNN (we use the network from [22]), the inferred cloth $C_N(\theta) = \{X_N(\theta), T_G\}$ is also a function of the pose θ. Our objective is to replace the ground truth texture coordinates T_G with perturbed texture coordinates $T_N(\theta, v)$, *i.e.* to compute $C'_N(\theta, v) = \{X_N(\theta), T_N(\theta, v)\}$ where $T_N(\theta, v)$ depends on both the pose θ and the view v. Even though $T_N(\theta, v)$ is in principle valid for all v using interpolation, training data $T_N(\theta, v_p)$ is only required for a finite number of camera views v_p. For each camera p, we also only require training data for finite number of poses θ_k, *i.e.* we require $T_N(\theta_k, v_p)$, which is computed from T_G using $X_G(\theta_k)$, $X_N(\theta_k)$, and v_p.

We project texture coordinates to the inferred cloth vertices $X_N(\theta_k)$ from the ground truth cloth mesh $C_G(\theta_k)$ using ray intersection. For each inferred cloth vertex in $X_N(\theta_k)$, we cast a ray from camera p's aperture through the vertex and find the first intersection with the ground truth mesh $C_G(\theta_k)$; subsequently, T_G is barycentrically interpolated to the point of intersection and assigned to the inferred cloth vertex as its $T_N(\theta_k, v_p)$ value. See Figure 2. Rays are only cast for inferred cloth vertices that have at least one incident triangle with a nonzero area subregion visible to camera p. Also, a ground truth texture coordinate value is only assigned to an inferred cloth vertex when the point of intersection with the ground truth mesh is visible to camera p. We store and learn texture coordinate displacements $d_{v_p}(\theta_k) = T_N(\theta_k, v_p) - T_G$. After this procedure, any remaining vertices of the inferred cloth that have not been assigned $d_{v_p}(\theta_k)$ values are treated as occluded.

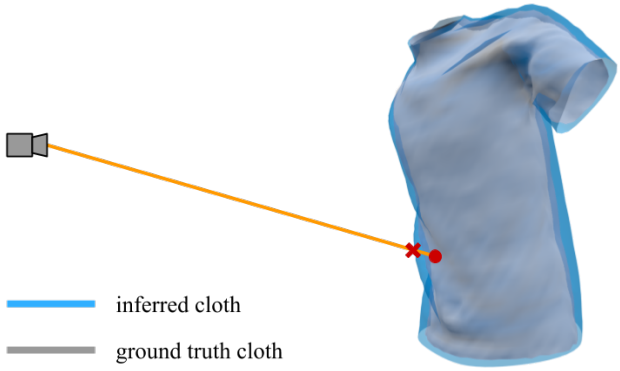


Figure 2: Illustration of the ray intersection method for transferring texture coordinates to the inferred cloth from the ground truth cloth. Texture coordinates for the inferred cloth vertex (red cross) are interpolated from the ground truth mesh to the point of ray intersection (red circle).

Some vertices of the inferred cloth mesh remain unassigned with $d_{v_p}(\theta_k) = 0$ after executing the algorithm above. This creates a discontinuity in $d_{v_p}(\theta_k)$ which excites high frequencies that require a more complex machine learning model to capture. In order to alleviate demands on the model, we smooth $d_{v_p}(\theta_k)$ as follows. First, we use the Fast Marching Method on triangulated surfaces [23] to generate a signed distance field. Then, we extrapolate $d_{v_p}(\theta_k)$ normal to the distance field into the unassigned region, see *e.g.* [31]. Finally, a bit of averaging is used to provide smoothness, while keeping the assigned values of $d_{v_p}(\theta_k)$ unchanged. Alternatively, one could solve a Poisson equation as in [10] while using the assigned $d_{v_p}(\theta_k)$ as Dirichlet boundary conditions.

The inferred cloth data we chose to correct are predictions of the T-shirt meshes from [22], each of which contains 2969 vertices. The dataset spans 9161 different poses

generated from a scanned garment using physically-based simulation [14]. To improve the resolution, we up-sampled each cloth mesh by subdividing each triangle into four sub-triangles. Notably, our texture sliding approach can be used to augment the results of any dataset for which ground truth and inferred training examples are available. Moreover, it is trivial to increase the resolution of any such dataset simply by subdividing triangles. Note that perturbations of the subdivided geometry are unnecessary, as we merely desire more sample points. Finally, we applied an 80-10-10 training-validation-test set split.

4. Methods

A separate machine learning model is trained for each camera p; thus, we drop the v_p notation in this section. We reiterate our input and output of machine learning models here for clarity: The input feature is the rotation matrix of each joint in the skeleton pose (denote as θ), and the output variable is the texture coordinate displacement of the cloth after texture sliding from a specific camera view (denote as $d(\theta)$), so it is a regression problem. We trained two different machine learning regressors. One is multivariate Support Vector Regression (SVR) with Gaussian kernel, and the other is Texture Sliding Neural Network (TSNN). We introduce these two methods below in detail.

4.1. Support Vector Regression

We first try a traditional machine learning model: Support Vector Machine (SVM), and use SVM for regression problems. We call it Support Vector Regression (SVR). The loss function we use is similar to hinge loss for classification. We define a bandwidth with width equals to ε. Any target variables that lie in the bandwidth of trained regressor will have zero loss. Otherwise a linear loss will be added. This loss is called ε-insensitive loss and is defined over all poses θ_k in the training set, formulated as

$$\mathcal{L} = \sum_{\theta_k} \begin{cases} 0, & \text{if } \|d(\theta_k) - \hat{d}(\theta_k)\|_2 \leq \varepsilon \\ \|d(\theta_k) - \hat{d}(\theta_k)\|_2 - \varepsilon & \text{otherwise} \end{cases} \tag{1}$$

to minimize the difference between the desired displacements $d(\theta_k)$ and predicted displacements $\hat{d}(\theta_k)$.

We also add kernel tricks to the regressor to add nonlinearity. Gaussian function is used as our kernel function. L2 regularization is also added to prevent overfitting.

For SVR, the input vector θ is the flattened 3×3 rotation matrices of 10 joints, which is a 90 dimensional vector. The output texture coordinate displacement $d(\theta)$ is represented as a per-vertex fashion, i.e., storing the UV displacements of each vertex. Thus, the output is a 2969×2 matrix, where 2969 is the number of vertices, and 2 is the displacements in U and V coordinate respectively. We flatten the matrix into a 5938 dimensional vector, so our multivariate SVR maps 90 dimensional input to a 5938 dimensional output.

Note that for the multivariate SVR, each element in the output is trained independently (having its own parameters w, b). It is very similar to a fully connected neural network with no hidden layer, where the output data has no spatial coherency. Thus, as shown in the Section 5.2, multivariate SVR performs very poorly in predicting texture coordinate displacement (where spatial coherency is very important).

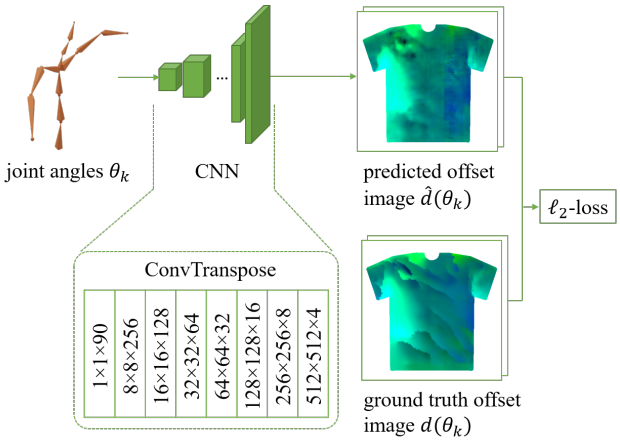


Figure 3: Texture sliding neural network (TSNN) architecture.

4.2. Texture Sliding Neural Network

To deal with spatial coherency and get better accuracy, we design a neural network called Texture Sliding Neural Network (TSNN) to do prediction. We use mean squared loss as our loss function. Similar to SVR, the loss is defined over all poses θ_k in the training set

$$\mathcal{L} = \sum_{\theta_k} \left\| d(\theta_k) - \hat{d}(\theta_k) \right\|_2 \tag{2}$$

to minimize the difference between the desired displacements $d(\theta_k)$ and predicted displacements $\hat{d}(\theta_k)$.

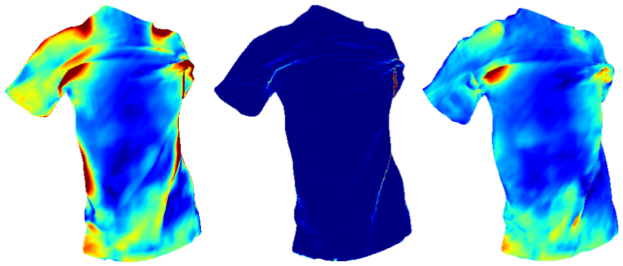
We adopt the paradigm similar to [22] to rasterize displacements on vertices into image, making it possible to use CNN to do predictions on mesh structure. The displacements $d(\theta_k)$ are stored as pixel-based cloth images for the front and back sides of the T-shirt, though we still output per-vertex texture coordinate displacements in UV space. See Figure 3 for an overview of the network architecture. Given input 90 dimensional vector, TSNN applies a series of transpose convolution, batch normalization, and ReLU activation layers to upsample the input to $512 \times 512 \times 4$. The first two dimensions of the output tensor represent the predicted displacements for the front side of the T-shirt, and the remaining two dimensions represent those for the back side.

5. Experiments and Results

We experiment and evaluate our method in three parts: in Section 5.1, we evaluate our dataset generation method by proving the efficacy of texture sliding approach; in Section 5.2, we evaluate our machine learning models, giving quantitative and qualitative results; in Section 5.3, we present one promising application of our ML model: 3D reconstruction of cloth with high frequency geometry.

5.1. Dataset Generation Evaluation

We measure texture coordinate errors in a per-pixel fashion comparing between the ground truth and inferred cloth at the center of each pixel. Figure 4a shows results typical for cloth inferred using the network from [22], and Figure 4b shows the highly improved results obtained on the same inferred geometry using our texture sliding approach (with 1 level of subdivision). Note that the vast majority of the errors in Figure 4b occur near the wrinkles where the non-linearities are most prevalent. In Figure 4c, we deform the vertices of the inferred cloth mesh so that they lie exactly on the ground truth mesh in order to mimic a two-step approach [40]. Note how our error metric captures the still rather large errors in the material coordinates (and thus cloth vertex positions) in spite of the mesh in Figure 4c appearing to have the same wrinkles and folds of the ground truth mesh.



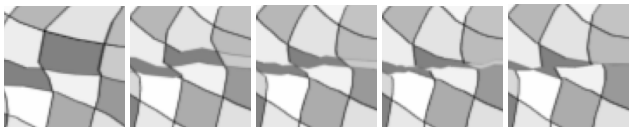
(a) (b) (c)

Figure 4: Per-pixel texture coordinate errors before (a) and after (b) applying texture sliding to the inferred cloth output by the network of [22]. The result of a two-step process (c) may well match the ground truth in a visual sense, whilst still having quite large errors in material coordinates. Blue $= 0$, red ≥ 0.04.

Figure 5 illustrates the efficacy of subdividing the cloth mesh to get more samples for texture sliding. The particular ground truth cloth wrinkle shown in Figure 5e is not captured by the inferred cloth geometry shown in Figure 5a. The texture sliding result shown in Figure 5b better represents the ground truth cloth. Figures 5c and 5d show how subdividing the inferred cloth mesh one and two times (respectively) progressively alleviates errors emanating from the linearity assumption. Table 1 shows quantitative results comparing the inferred cloth to texture sliding with and without subdivision.

Method	SqrtMSE ($\times 10^{-3}$)
Inferred	24.496 ± 6.9536
TS	5.2662 ± 2.2320
TS + subdivision	3.5645 ± 1.6617

Table 1: Per-pixel square root of mean squared error (SqrtMSE) for the entire dataset.



(a) (b) (c) (d) (e)

Figure 5: As the inferred cloth mesh (a) is subdivided, texture sliding (b-d) moves the appearance of the inferred mesh closer to the ground truth (e).

5.2. ML Model Evaluation

For machine learning model evaluation, we include both qualitative and quantitative results. For qualitative results, we show the texture sliding results in generated dataset, the texture sliding predicted by our machine learning models, and the texture coordinate error between model prediction and ground truth cloth. For quantitative results, we use the error metric same as equation 2, which computes the square-root mean squared error (SqrtMSE) between model prediction and ground truth.

We first evaluate our traditional SVR model. We use the built-in Support Vector Regression model in Sklearn [34] to do regression. Gaussian kernel and L2 Regularization is used. For Gaussian kernel, we set $\sigma = 0.1$. The regularization coefficient C is set to 10^3. The bandwidth ε in the loss function is set to 0.1. As mentioned above, due to no spatial coherency, SVR peforms very poorly on predicting displacements. The second row of Table 2 shows that the per-pixel SqrtMSE loss is even higher than the inferred cloth (with zero texture coordinate displacement). By seeing the qualitative result produced by SVR, we notice that the texture is totally disrupted, leading to a very high error. Due to the space limit, we do not include the bad qualitative results of SVR in this report.

We then evaluate our TSNN. The network was trained using the Adam optimizer [24] with a 10^{-3} learning rate in PyTorch [32]. We trained multiple networks to test the performance of TSNN in different scenarios. One thing we

experimented is subdivision on the dataset. We generated three sets of dataset, with no subdivision, 1 time subdivision and 2 times subdivision respectively, and trained networks on these different datasets. The other thing we experimented is the output layer of the TSNN. We changed the output layer of TSNN from $512\times512\times4$ to $1024\times1024\times4$, and did some modification to the network architecture. Then we trained this new network on 1 time subdivision dataset and 2 times subdivision dataset to see if there is any improvement. Table 2 shows quantitative results comparing the inferred cloth to TSNN results with different levels of subdivision and different output resolution. Figure 6 shows some typical qualitative results on test set, including the per-pixel errors in predicted texture coordinates. We can see that while the TSNN is able to recover the majority of the shirt, it struggles near wrinkles.

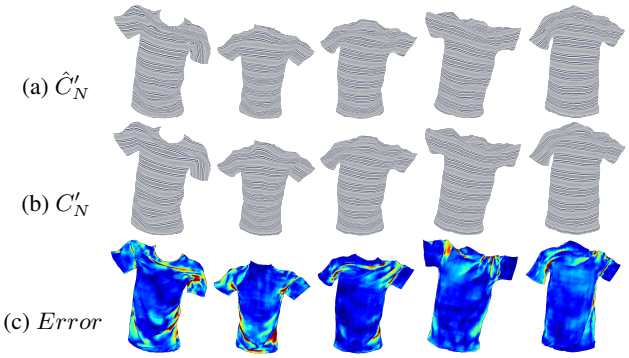


Figure 6: Some typical test set example predictions. The per-pixel errors are shown in (c) (blue $= 0$, red ≥ 0.04).

ML Model	SqrtMSE ($\times10^{-3}$)
Inferred	24.871 ± 7.0613
SVR	51.848 ± 4.1035
TSNN	13.335 ± 4.2924
TSNN + subdivision	13.591 ± 4.5194
TSNN + $2\times$ subdivision	13.506 ± 4.6745
TSNN (1024) + subdivision	16.243 ± 6.1372
TSNN (1024) + $2\times$ subdivision	13.418 ± 4.7268

Table 2: Per-pixel SqrtMSE for the test set. SVR performs very poorly on prediction. For TSNN, multiple neural networks including different output resolution and different times subdivision are trained and tested.

5.3. 3D Reconstruction

In this section, we present a promising application of our trained TSNN: 3D reconstruction. Given TSNN trained on at least two camera views, we can reconstruct the high-frequency 3D cloth geometry by triangulation [19] using texture sliding information. In order to reconstruct the 3D position of a vertex of the ground truth mesh, we take the usual approach of finding rays that pass through that vertex and the camera aperture for a number of cameras. Then given at least two rays, one can triangulate a 3D point that is minimal distance from all the rays. After triangulation, the mesh is still noisy, so we apply a physically-based post-processing [16] to the mesh to smooth the result. Figure 7 shows the 3D reconstruction of four test set examples using TSNN, compared to the ground truth and inferred cloth mesh.

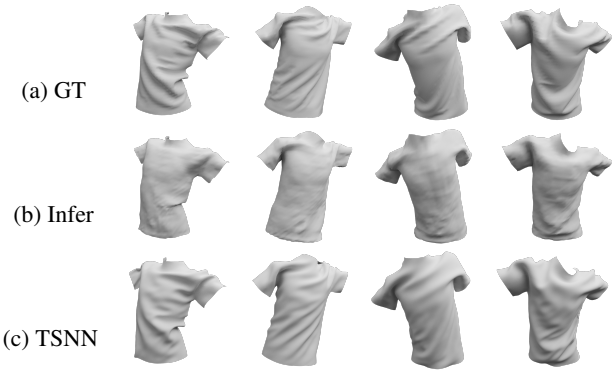


Figure 7: Comparison of the ground truth cloth (a) and inferred cloth (b) to the 3D reconstructions obtained using TSNN (c). To remove reconstruction noise generated by network inference errors in (c), we used the postprocess from [16]; although, there are many other smoothing options in the literature that one might also consider.

6. Conclusion

We present a technique called texture sliding that can recover high frequency information (which is lost due to regularization/interpolation of machine learning models) from overly smoothed inferred cloth mesh. We propose two different machine learning models: SVR and TSNN, to learn texture sliding from data. From experiments, we found that TSNN can recover the majority of the shirt with low error. We also used triangulation to reconstruct 3D high frequency cloth geometry from texture sliding information predicted from TSNN.

As future work, we plan on continuing experimenting with the network architecture, the size of the image used in the CNN, the smoothing methods near occlusion boundaries, the amount of subdivision, etc. In addition, it would be interesting to consider more savvy multiview 3D reconstruction methods (particularly ones that employ DNNs; then, one might train the whole process end-to-end).

7. Contribution and Code

Most of the work in this report is implemented by Yongxu, except two: the Fast Marching Method in dataset generation is implemented by Jane Wu, and the 3D reconstruction in the experiment section is implemented by Zhenglin Geng. The source code of this project can be found at: `https://github.com/jinyx728/CS229`. The Arxiv version of this work will release soon. I would like to thank Prof. Ron Fedkiw, Jane Wu and Zhenglin Geng for their guidance on this project.

References

[1] Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single rgb camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1175–1186, 2019. 1, 2

[2] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Detailed human avatars from monocular video. In *2018 International Conference on 3D Vision (3DV)*, pages 98–109. IEEE, 2018. 1, 2

[3] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Video based reconstruction of 3d people models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8387–8397, 2018. 1, 2

[4] Thiemo Alldieck, Gerard Pons-Moll, Christian Theobalt, and Marcus Magnor. Tex2shape: Detailed full human body geometry from a single image. In *Proceedings of the International Conference on Computer Vision (ICCV)*. IEEE, 2019. 1, 2

[5] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM transactions on graphics (TOG)*, volume 24, pages 408–416. ACM, 2005. 1

[6] Alexandru O Bălan and Michael J Black. The naked truth: Estimating body shape under clothing. In *European Conference on Computer Vision*, pages 15–29. Springer, 2008. 1

[7] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. In *ACM Transactions on Graphics (TOG)*, volume 27, page 99. ACM, 2008. 1

[8] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Transactions on Graphics (ToG)*, volume 21, pages 594–603. ACM, 2002. 1

[9] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 28–36. ACM, 2003. 1

[10] Matthew Cong, Michael Bao, Jane L E, Kiran S Bhat, and Ronald Fedkiw. Fully automatic generation of anatomical face simulation models. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 175–183. ACM, 2015. 2

[11] R Daněřek, Endri Dibra, Cengiz Öztireli, Remo Ziegler, and Markus Gross. Deepgarment: 3d garment shape estimation from a single image. In *Computer Graphics Forum*, volume 36, pages 269–280. Wiley Online Library, 2017. 2

[12] Endri Dibra, Himanshu Jain, Cengiz Öztireli, Remo Ziegler, and Markus Gross. Hs-nets: Estimating human body shape from silhouettes with convolutional neural networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 108–117. IEEE, 2016. 2

[13] Endri Dibra, Himanshu Jain, Cengiz Oztireli, Remo Ziegler, and Markus Gross. Human shape from silhouettes using generative hks descriptors and cross-modal neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4826–4836, 2017. 2

[14] Pradeep Dubey, Pat Hanrahan, Ronald Fedkiw, Michael Lentine, and Craig Schroeder. Physbam: Physically based simulation. In *ACM SIGGRAPH 2011 Courses*, page 10. ACM, 2011. 2, 3

[15] Jean-Sébastien Franco, Marc Lapierre, and Edmond Boyer. Visual shapes of silhouette sets. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 397–404. IEEE, 2006. 1

[16] Zhenglin Geng, Dan Johnson, and Ronald Fedkiw. Coercing machine learning to output physically accurate results. *arXiv preprint arXiv:1910.09671*, 2019. 5

[17] Erhan Gundogdu, Victor Constantin, Amrollah Seifoddini, Minh Dang, Mathieu Salzmann, and Pascal Fua. Garnet: A two-stream network for fast and accurate 3d cloth draping. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8739–8748, 2019. 2

[18] Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Livecap: Real-time human performance capture from monocular video. *ACM Transactions on Graphics (TOG)*, 38(2):14, 2019. 1, 2

[19] Richard I Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997. 1, 5

[20] Peng Huang, Margara Tejera, John Collomosse, and Adrian Hilton. Hybrid skeletal-surface motion graphs for character animation from 4d performance capture. *ACM Transactions on Graphics (ToG)*, 34(2):17, 2015. 1

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2

[22] Ning Jin, Yilin Zhu, Zhenglin Geng, and Ronald Fedkiw. A pixel-based framework for data-driven clothing. *arXiv preprint arXiv:1812.01677*, 2018. 1, 2, 3, 4

[23] Ron Kimmel and James A Sethian. Computing geodesic paths on manifolds. *Proceedings of the national academy of Sciences*, 95(15):8431–8435, 1998. 2

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[25] Zorah Lahner, Daniel Cremers, and Tony Tung. Deepwrinkles: Accurate and realistic clothing modeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 667–684, 2018. 1, 2

[26] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):248, 2015. 2

[27] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[28] Matthias Müller and Nuttapong Chentanez. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 85–92. Eurographics Association, 2010. 2

[29] Ryota Natsume, Shunsuke Saito, Zeng Huang, Weikai Chen, Chongyang Ma, Hao Li, and Shigeo Morishima. Siclope: Silhouette-based clothed people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4490, 2019. 1, 2

[30] Alexandros Neophytou and Adrian Hilton. A layered model of human body and garment deformation. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 171–178. IEEE, 2014. 1

[31] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, New York, 2002. 2

[32] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 4

[33] 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

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 4

[35] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J Black. Clothcap: Seamless 4d clothing capture and retargeting. *ACM Transactions on Graphics (TOG)*, 36(4):73, 2017. 1

[36] Tiberiu Popa, Quan Zhou, Derek Bradley, Vladislav Kraevoy, Hongbo Fu, Alla Sheffer, and Wolfgang Heidrich. Wrinkling captured garments using space-time data-driven deformation. In *Computer Graphics Forum*, volume 28, pages 427–435. Wiley Online Library, 2009. 2

[37] Nadia Robertini, Edilson De Aguiar, Thomas Helten, and Christian Theobalt. Efficient multi-view performance capture of fine-scale surface detail. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 5–12. IEEE, 2014. 2

[38] Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles. In *ACM Transactions on Graphics (TOG)*, volume 29, page 157. ACM, 2010. 2

[39] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the International Conference on Computer Vision (ICCV)*. IEEE, 2019. 1

[40] Igor Santesteban, Miguel A Otaduy, and Dan Casas. Learning-based animation of clothing for virtual try-on. In *Computer Graphics Forum*, volume 38, pages 355–366. Wiley Online Library, 2019. 1, 2, 4

[41] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001. 1

[42] Andrew Selle, Jonathan Su, Geoffrey Irving, and Ronald Fedkiw. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE transactions on visualization and computer graphics*, 15(2):339–350, 2008. 1

[43] Gul Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. Bodynet: Volumetric inference of 3d human body shapes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36, 2018. 2

[44] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. In *ACM Transactions on Graphics (TOG)*, volume 27, page 97. ACM, 2008. 1

[45] Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F O'Brien. Example-based wrinkle synthesis for clothing animation. In *Acm Transactions on Graphics (TOG)*, volume 29, page 107. ACM, 2010. 2

[46] Chenglei Wu, Kiran Varanasi, and Christian Theobalt. Full body performance capture under uncontrolled and varying illumination: A shading-based approach. In *European Conference on Computer Vision*, pages 757–770. Springer, 2012. 1

[47] Weipeng Xu, Avishek Chatterjee, Michael Zollhöfer, Helge Rhodin, Dushyant Mehta, Hans-Peter Seidel, and Christian Theobalt. Monoperfcap: Human performance capture from monocular video. *ACM Transactions on Graphics (ToG)*, 37(2):27, 2018. 1

[48] Jinlong Yang, Jean-Sébastien Franco, Franck Hétroy-Wheeler, and Stefanie Wuhrer. Estimation of human body shape in motion with wide clothing. In *European Conference on Computer Vision*, pages 439–454. Springer, 2016. 1

[49] Jinlong Yang, Jean-Sébastien Franco, Franck Hétroy-Wheeler, and Stefanie Wuhrer. Analyzing clothing layer deformation statistics of 3d human motions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 237–253, 2018. 1

[50] Tao Yu, Zerong Zheng, Yuan Zhong, Jianhui Zhao, Qionghai Dai, Gerard Pons-Moll, and Yebin Liu. Simulcap: Single-view human performance capture with cloth simulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[51] Chao Zhang, Sergi Pujades, Michael J Black, and Gerard Pons-Moll. Detailed, accurate, human shape estimation from clothed 3d scan sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4191–4200, 2017. 1