

# A Pixel-Based Framework for Data-Driven Clothing

Ning Jin<sup>1</sup>, Yilin Zhu<sup>1</sup>, Zhenglin Geng<sup>1</sup>, and Ronald Fedkiw<sup>1,2</sup>

<sup>1</sup>Stanford University, <sup>2</sup>Industrial Light & Magic

{njin19, yilinzhu, zhenglin}@stanford.edu, fedkiw@cs.stanford.edu

## Abstract

*With the aim of creating virtual cloth deformations more similar to real world clothing, we propose a new computational framework that recasts three dimensional cloth deformation as an RGB image in a two dimensional pattern space. Then a three dimensional animation of cloth is equivalent to a sequence of two dimensional RGB images, which in turn are driven/choreographed via animation parameters such as joint angles. This allows us to leverage popular CNNs to learn cloth deformations in image space. The two dimensional cloth pixels are extended into the real world via standard body skinning techniques, after which the RGB values are interpreted as texture offsets and displacement maps. Notably, we illustrate that our approach does not require accurate unclothed body shapes or robust skinning techniques. Additionally, we discuss how standard image based techniques such as image partitioning for higher resolution, GANs for merging partitioned image regions back together, etc., can readily be incorporated into our framework.*

## 1. Introduction

Virtual clothing has already seen widespread adoption in the entertainment industry including feature films (*e.g.*, Yoda [12], Dobby [13], Monsters, Inc. [7]), video games (*e.g.*, [21, 37, 39, 40, 57, 58]), and VR/AR and other real-time applications (*e.g.*, [31, 53, 68, 70]). However, its potential use in e-commerce for online shopping and virtual try-on is likely to far surpass its use in the entertainment industry especially given that clothing and textiles is a three trillion dollar industry<sup>1</sup>. Whereas games and real-time applications can use lower quality cloth and films have the luxury of a large amount of time and manual efforts to achieve more realistic cloth, successful e-commerce clothing applications demand high quality predictive clothing with fast turnaround, low computational resource usage, and good scalability.

<sup>1</sup><https://fashionunited.com/global-fashion-industry-statistics>

Although there have been many advances in cloth simulation, the ability to match real cloth of a specific material, especially with highly detailed wrinkling, hysteresis, etc. is rather limited. Moreover, contact and collision approaches typically lack physical accuracy due to unknown parameters dependent on a multitude of factors even including body hair density and garment thread friction. Thus, while embracing simulation and geometric techniques wherever possible, we pursue a new paradigm approaching clothing on humans in a fashion primarily driven by data at every scale. This is rather timely as 3D cloth capture technology is starting to seem very promising [17, 61, 63].

Motivated by a number of recent works that view cloth deformations as offsets from the underlying body [26, 59, 61, 71] as well as the recent phenomenal impact of convolutional neural networks for image processing [28, 42, 51, 62, 64, 66], we recast cloth deformation as an image space problem. That is, we shrink wrap a cloth mesh onto the underlying body shape, viewing the resulting shrink-wrapped vertex locations as pixels containing RGB values that represent displacements of the shrink-wrapped cloth vertices from their pixel locations in texture and normal coordinates. These cloth pixels are barycentrically embedded into the triangle mesh of the body, and as the body deforms the pixels move along with it; however, they remain at fixed locations in the pattern space of the cloth just like standard pixels on film. Thus, cloth animation is equivalent to playing an RGB movie on the film in pattern space, facilitating a straightforward application of CNNs. Each cloth shape is an image, and the animation parameters for joint angles are the choreography that sequences those images into a movie of deforming cloth.

Although we leverage body skinning [5, 36, 38, 52, 54] to move the cloth pixels around in world space, we are not constrained by a need to ascertain the unclothed body shape accurately as other authors aim to [59, 61]. Of course, an accurate unclothed body shape might reduce variability in the cloth RGB image to some degree, but it is likely that CNN network efficacy will advance faster than the technology required to obtain and subsequently accurately pose unclothed body shapes. Even if consumers were willing to provide more accurate unclothed body data or inferences

of their unclothed body forms improve, it is still difficult to subsequently pose such bodies to create accurate shapes governed by animation parameters such as joint angles. In contrast, we demonstrate that CNNs can learn the desired clothing shapes even when unclothed body shapes are intentionally modified to be incorrect, thus providing some immunity to problematic skinning artifacts (*e.g.*, candy wrapper twisting [33, 36, 38]).

## 2. Related Work

**Skinning:** Linear blend skinning (LBS) [46, 54] is perhaps the most popular skinning scheme used in animation software and game engines. Although fast and computationally inexpensive, LBS suffers from well-known artifacts such as candy wrapper twisting, elbow collapse, etc., and many works have attempted to alleviate these issues, *e.g.*, spherical blend skinning (SBS) [38], dual-quaternion skinning (DQS) [36], stretchable and twistable bones skinning (STBS) [33], optimized centers of rotations [47], etc. Another line of works explicitly model pose specific skin deformation from sculpted or captured example poses. For example, pose space deformation (PSD) [48] uses radial basis functions to interpolate between artist-sculpted surface deformations, [44] extends PSD to weighted PSD, and [4] uses  $k$ -nearest neighbor interpolation. EigenSkin [43] constructs compact eigenbases to capture corrections to LBS learned from examples. The SCAPE model [5] decomposes pose deformation of each mesh triangle into a rigid rotation  $R$  from its body part and a non-rigid deformation  $Q$  and learns  $Q$  as a function of nearby joints, and Blend-SCAPE [32] extends this expressing each triangle’s rigid rotation as a linear blend of rotations from multiple parts. [52] learns a statistical body model SMPL that skins the body surface from linear pose blendshapes along with identity blendshapes. More recently, [6] uses neural networks to approximate the non-linear component of surface mesh deformations from complex character rigs to achieve real-time deformation evaluation for film productions. Still, skinning remains one of the most challenging problems in the animation of virtual characters; thus, we illustrate that our approach has the capability to overcome some errors in the skinning process.

**Cloth Skinning and Capture:** A number of authors have made a library of cloth versus pose built primarily on simulation results and pursued ways of skinning the cloth for poses not in the library. [68] looks up a separate wrinkle mesh for each joint and blends them, and similarly [70] queries nearby examples for each body region and devises a sensitivity-optimized rigging scheme to deform each example before blending them. [39] incrementally constructs a secondary cloth motion graph. [21] learns a linear function for the principal component coefficients of the cloth shape, and [27] runs subspace simulation using a set of adaptive

bases learned from full space simulation data. Extending the SCAPE model to cloth, [26] decomposes per-triangle cloth deformation into body shape induced deformation  $D$ , rigid rotation  $R$ , and non-rigid pose induced deformation  $Q$ , and applies PCA on  $D$  and  $Q$  to reduce dimensionality. Whereas [26] treats the cloth as a separate mesh, [59] models cloth as an additional deformation of the body mesh and learns a layered model. More recently [61] builds a dataset of captured 4D sequences and retargets cloth deformations to new body shapes by transferring offsets from body surfaces. The aforementioned approaches would all likely achieve more realistic results using real-world cloth capture as in [45, 61] as opposed to physical simulations.

**Networks:** Some of the aforementioned skinning type approaches to cloth and bodies learn from examples and therefore have procedural formulas and weights which often require optimization in order to define, but here we focus primarily on methods that use neural networks in a more data-driven as opposed to procedural fashion. While we utilize procedural methods for skinning the body mesh and subsequently finding our cloth pixel locations, we use data-driven networks to define the cloth deformations; errors in the procedural skinning are simply incorporated into the offset function used to subsequently reach the data. Several recent works used neural networks for learning 3D surface deformations for character rigs [6] and cloth shapes [20, 45, 71]. In particular, [6, 71] input pose parameters and output non-linear shape deformations of the skin/cloth, both using a fully connected network with a few hidden layers to predict PCA coefficients. [20] takes input images from single or multiple views and uses a convolutional network to predict 1000 PCA coefficients. [45] takes a hybrid approach combining a statistical model for pose-based global deformation with a conditional generative adversarial network for adding details on normal maps to produce finer wrinkles.

**Faces:** Face deformations bear some similarities to body skinning except there are only two bones with a single joint connecting the skull and the jaw, and most of the parameters govern shape/expression. We briefly mention the review paper on blendshapes [49] and refer the reader to that literature for more discussions. However, the recently proposed [23] has some similarities with our approach. They use texture coordinates similar to ours, except that they store the full 3D positions as RGB values, whereas our cloth pixels derive their 3D positions from the surface of a skinned body mesh while storing offsets from these 3D positions as RGB values. Extending our approach to faces, our pixels would follow the shape of the skinned face as the jaw opens and closes. The RGB values that we would store for the face would only contain the offsets from the skinned cranium and jaw due to blendshaped expressions. We would not need to learn the face neutral (identity) shape or the

skinning, and the offset function would simply be identically zero when no further expressions were applied, reducing the demands on the network. Essentially, their method is what computational mechanics refers to as “Eulerian” where the computational domain is fixed, as opposed to a “Lagrangian” method with a computational data structure that follows the motion of the material (*e.g.*, using particles). Our approach could be considered an Arbitrary Lagrangian-Eulerian (ALE [55]) method where the computational domain follows the material partially but not fully, *i.e.*, our cloth pixels follow only the deformation captured by body skinning.

### 3. Pixel-Based Cloth

We start by creating a texture map for the cloth mesh, assigning planar UV coordinates to each vertex. For illustration, we take the front side of a T-shirt mesh as an example, see Figure 1a. Using UV space as the domain, each vertex stores a vector-valued function of displacements  $\mathbf{dx}(u, v) = (\Delta u, \Delta v, \Delta n)$  representing perturbations in the texture coordinate and normal directions. This can be visualized by moving each vertex by  $\mathbf{dx}$ , see Figure 1b. These displacements can be conveniently interpreted as RGB colors stored at vertex locations in this pattern space; thus, we will refer to these vertices as *cloth pixels*, see Figure 1c. Note that the RGB colors of the cloth pixels may contain values not in the visible range using HD image formats, floating point representations, etc. This framework allows us to leverage standard texture mapping [11, 16, 29] as well as other common approaches, such as using bump maps [10] to perturb normal directions and displacement maps [19] to alter vertex positions; these techniques have been well-established over the years and have efficient im-

plementations on graphics hardware enabling us to hijack and take advantage of the GPU-supported pipeline for optimized performance.

### 4. Cloth Images

As can be seen in Figure 1a and 1c, the cloth pixels are located at vertex positions and are connected via a triangle mesh topology. CNNs exploit spatial coherency and such methods can be applied here using graph learning techniques [14, 15, 22, 30, 56], see in particular [67]. Alternatively, since our cloth pixels have fixed UV coordinates in the two dimensional pattern space, we may readily interpolate to a uniform background Cartesian grid of pixels using standard triangle rasterization ([24]) with some added padding at the boundaries to ensure smoothness (see Figure 2), thus facilitating more efficient application of standard CNN technologies especially via GPUs.

Note that we convert all our training data into pixel-based cloth images and train on those images directly, so that the networks learn to predict 2D images, not 3D cloth shapes. If one wanted to connect animation parameters to cloth vertex positions in a more fully end-to-end manner, then the interpolatory approach back and forth between the triangle mesh vertices and the pixels on the Cartesian grid would potentially require further scrutiny. For example, the fluid dynamics community takes great care in addressing the copying back and forth of data between particle-based data structures (similar to our cloth pixels in Figure 1c) and background grid degrees of freedom (similar to our cloth image in Figure 2). Most notable are the discussions on PIC/FLIP, see *e.g.* [34].

Quite often one needs to down-sample images, which creates problems for learning high frequency details. In-

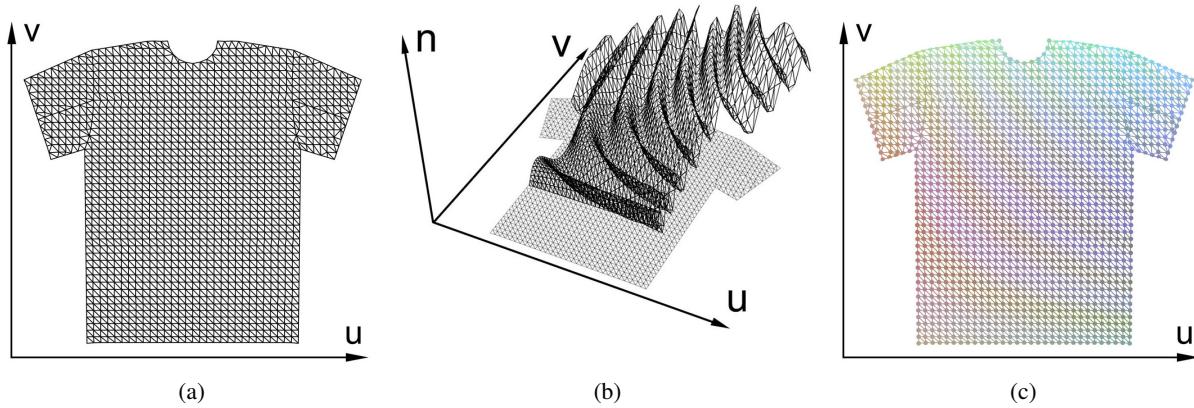


Figure 1: Left: Triangle mesh depicted in texture space using the vertices’ UV coordinates. Middle: depiction of the displacement via  $(u, v, 0) + \mathbf{dx}$  for each vertex. Right: visualization of the displacement field  $\mathbf{dx}$  converted into RGB values normalized to the visible  $[0, 255]$  range and plotted at each vertex.

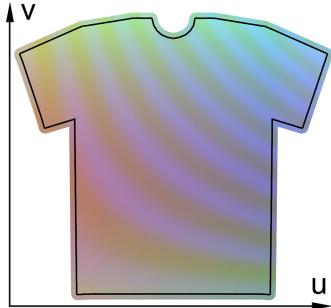


Figure 2: Standard uniform Cartesian grid of pixels for our cloth image. We add some padding to ensure smoothness on the boundaries for convolutional filters.

stead, we use a support “cage” to divide the cloth mesh into smaller patches to aid the learning process, see Figure 3. This notion of a cage and patch based cloth is quite powerful and is useful for capture, design, simulation, blendshape systems, etc. (see Appendix C for more discussions). While cloth already exhibits spatially invariant physical properties making it suitable for convolutional filters and other spatially coherent approaches, further dividing it into semantically coherent individual patches allows a network to enjoy a higher level of specialization and performance. The only caveat is that one needs to take care to maintain smoothness and consistency across patch boundaries, but this can be achieved using a variety of techniques such as GANs [25, 50], image inpainting [8, 72], PCA filtering, etc.

## 5. Skinning Cloth Pixels

While the cloth pixels have fixed UV locations in their 2D pattern space, their real-world 3D positions change as the body moves. We generate real-world positions for the

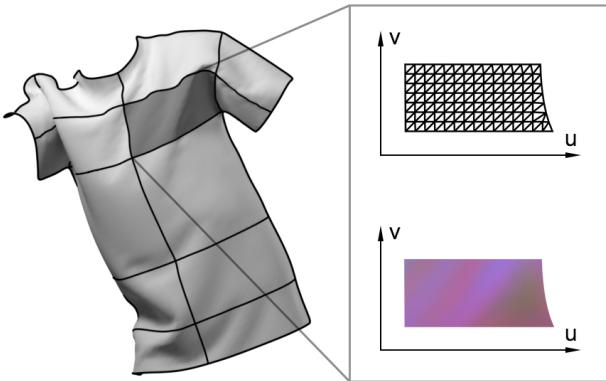


Figure 3: Left: front side of a T-shirt mesh divided into patches by a “cage” (depicted as black edges). Right: the triangulated cloth pixels and corresponding RGB cloth image for the highlighted patch.

cloth pixels by barycentrically embedding each of them into a triangle of the body mesh. Then as the body mesh deforms, the real-world locations of the cloth pixels move along with the triangles they were embedded into. Figure 4 top row shows the pixel RGB values from Figure 1c embedded into the rest pose and a different pose. Applying the  $\mathbf{dx}$  offsets depicted in Figure 1b to the real-world pixel locations in Figure 4 top row yields the cloth shapes shown in Figure 4 bottom row. In Figure 5, we show the process reversed where the cloth shape shown in Figure 5 left is recorded as  $\mathbf{dx}$  displacements and stored as RGB values on the cloth pixels embedded in the body mesh, see Figure 5 middle. These pixel RGB values in turn correspond to a cloth image in the pattern space, see Figure 5 right.

In order to obtain barycentric embeddings of the cloth pixels to the triangles of the body mesh, we start in a rest pose and uniformly shrink the edges of the cloth mesh making it skin-tight on the body. Since this preprocessing step is only done once, and moreover can be accomplished on a template mesh, we take some care in order to achieve a good sampling distribution of the body deformations that drive our cloth image. Note that our formulation readily allows for more complex clothing (such as shirts/jacket collars) to be embedded on the body with overlapping folds in a non-one-to-one manner, *i.e.*, the inverse mapping from



Figure 4: Top: the cloth pixels are shown embedded into body triangles with RGB values copied over from Figure 1c in the rest pose (top left) and a different pose (top right). Bottom: The final cloth shapes obtained by adding displacements  $\mathbf{dx}$  depicted in Figure 1b to the cloth pixel locations in the top row.



Figure 5: Left: part of a 3D cloth shape. Middle: cloth pixels embedded on the body mesh storing displacements  $\mathbf{dx}$  as RGB values. Right: corresponding cloth image in the two dimensional pattern space.

the body texture coordinates to the cloth texture coordinates does not need to exist. See Appendix A for more details.

One might alternatively skin the cloth as discussed in Section 2 to obtain a candidate cloth shape, and embed our cloth pixels into the real-world skinned cloth shape, learning offsets from the skinned cloth to the simulated or captured cloth. The difficulty with this approach is that much of the example-based cloth can behave in quite unpredictable ways making it difficult for a network to learn the offset functions. Thus we prefer to embed our pixels into the body geometry which deforms in a more predictable and smooth manner. Moreover, this allows us to leverage a large body of work on body skinning as opposed to the much smaller number of works that consider cloth skinning.

An interesting idea would be to learn the cloth shape in a hierarchical fashion, first obtaining some low resolution/frequency cloth as offsets from the human body using our image-based cloth, and then embedding pixels in that resulting cloth mesh, subsequently learning offsets from it for higher resolution. We instead prefer analyzing the result from our image based cloth using a number of techniques including compression [35] to see where it might require further augmentation via for example data based wrinkle maps. That is, we do not feel that the same exact approach should be applied at each level of the hierarchy, instead preferring more specialized approaches at each level using domain knowledge of the interesting features as well as the ability to incorporate them.

## 6. Network Considerations

Given input pose parameters, we predict cloth images on the Cartesian grid of pixels in the pattern space. These images represent offsets  $\mathbf{dx}$  from the pixels embedded to follow the body as opposed to global coordinates so that one does not need to learn what can be procedurally incorporated via body skinning (as discussed in Section 2 in regards to faces). Moreover,  $\mathbf{dx}$  is parameterized in local geodesic coordinates  $u$  and  $v$  as well as the normal direction  $n$  in

order to enable the representation of complex surfaces via simpler functions, e.g., see Figure 6; even small perturbations in offset directions can lead to interesting structures.

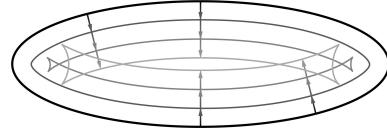


Figure 6: An ellipse with simple constant function offsets in the normal direction, for three different constant values. (well-known swallowtail structure<sup>2</sup>)

Although fully connected networks have been a common choice for generating dense per-vertex 3D predictions such as in [6, 71], coalescing a 2D triangulated surface into a 1D vector forgoes potentially important spatial adjacency information and may lead to a bigger network size as pointed out in [23]. A commonly employed remedy resorts to linear dimensionality reduction methods such as PCA to recover some amount of spatial coherency and smoothness in the output, as the regularized network predicts a small number of PCA coefficients instead of the full degrees of freedom. Alternatively, our pixel-based cloth framework leverages convolutional networks that are particularly well-suited for and have demonstrated promising results in tasks in the image domain where the filters can share weights and exploit spatial coherency; our convolutional decoder takes a 1D vector of pose parameters and gradually upsamples it to the target resolution via transpose convolution operations. As a side note, in Appendix E.3, we illustrate that our cloth pixel framework offset functions can be approximated via a lower dimensional PCA basis, which is amenable to training and prediction via a fully connected network.

Our base loss is defined on the standard Cartesian grid image pixels weighted by a Boolean mask of the padded UV map. One can use different norms for this loss, and empirically we find that while  $L_1$  leads to slightly better quantitative metrics than  $L_2$ , their visual qualities are roughly similar. Noting that normal vectors are important in capturing surface details, we experiment with incorporating an additional loss term on the per-vertex normals.

## 7. Experiments

**Dataset Generation:** For the T-shirt examples, we generate 20K poses for the upper body by independently sampling rotation angles along each axis for the 10 joints from a uniformly random distribution in their natural range of motion, and then applying a simple pruning procedure to remove invalid poses, e.g., with severe nonphysical self-penetrations. We divided the dataset into a training set (16k

<sup>2</sup>See for example page 21 of [65].

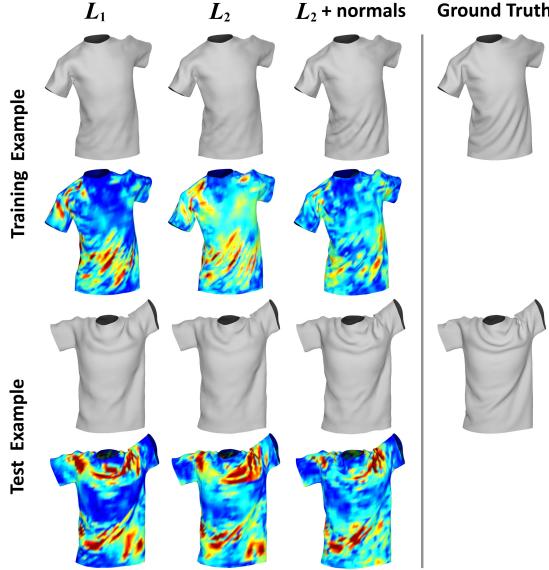


Figure 7: Network predictions/errors (blue = 0, red  $\geq 1$  cm) from models trained with different loss functions. While  $L_1$  and  $L_2$  loss on the pixels behave similarly, adding a loss term on the normals yields better visual quality. From left to right:  $L_1$  on the pixels;  $L_2$  on the pixels;  $L_2$  on the pixels and cosine on the normals; ground truth.

samples), a regularization set (2K samples to prevent the network from overfitting), and a test set (2K samples that the optimization never sees during training). The test set is used for model comparisons in terms of loss functions and network architectures, and serves as a proxy for generalization error. See Appendix D for more details.

**Architecture, Training, and Evaluation:** Our convolutional decoder network takes in  $1 \times 1 \times 90$  dimensional input rotation matrices, and applies transpose convolution, batch normalization, and ReLU activation until the target output size of  $256 \times 256 \times 6$  is reached, where 3 output channels represent offset values for the front side of the T-shirt and 3 channels represent those of the back. The models are

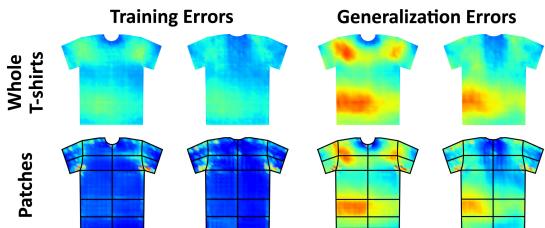


Figure 8: Dataset average per cloth pixel errors on the front/back side of the T-shirt. Top row: model trained on whole T-shirts (training/generalization error is 0.37 cm/0.51 cm). Bottom row: models trained on patches (training/generalization error is 0.20 cm/0.46 cm).

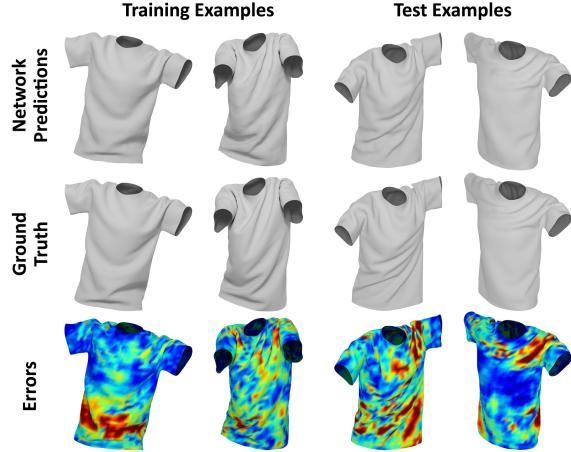


Figure 9: Network predictions and errors on training set and test set examples using our best loss model.

trained using the Adam optimizer [41] with  $10^{-3}$  learning rate. Our implementation uses the PyTorch [60] platform, and the code will be made publicly available along with the dataset. The best visual results we obtained were from models that used additional losses on the normals, see Figure 7 for comparison. Figure 9 shows more examples in various poses from both the training and the test set and their error maps using our best loss model. It is interesting to observe that the quantitative error metrics may not directly translate to visual quality since slight visual shift of folds or wrinkles can introduce big numerical errors. Figure 8 shows the average per cloth pixel model prediction errors on the training and test set. Unsurprisingly, the biggest errors occur near the sleeve seams and around the waist, where many wrinkles and folds form as one lifts their arms or bends. Finally, to see how well our model generalizes to new input data, we evaluated it on a motion capture sequence from [1], see Figure 10 and accompanying video.

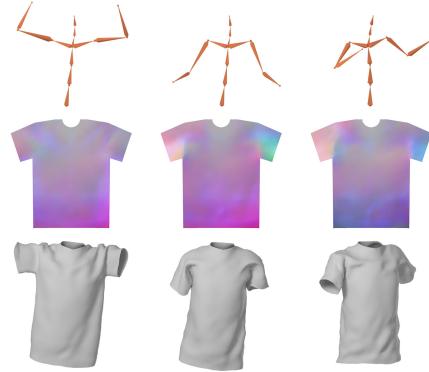


Figure 10: Evaluation on motion capture. Top: skeletal poses. Middle: predicted cloth images. Bottom: predicted cloth shapes.

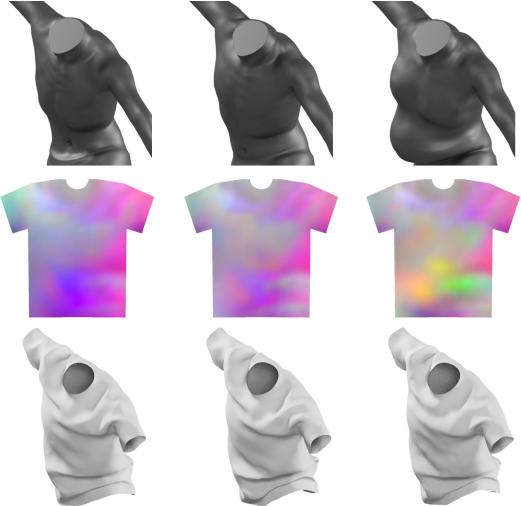


Figure 11: Training the network on unclothed body shapes that are too thin (left column) or too thick (right column) does not hinder its ability to predict cloth shapes, as compared to the ground truth (middle column). The cloth images (middle row) readily compensate for the incorrect unclothed body shape assumptions leading to similar cloth shapes (bottom row) in all three cases.

**Modified Body Shapes:** The inability to obtain accurate unclothed body shapes is often seen as a real-world impediment to e-commerce clothing applications. Our approach embeds cloth pixels in the unclothed form and leverages procedural body skinning techniques to move those cloth pixels throughout space. This embedding of cloth pixels provides spatial coherency for the CNN and alleviates the need for the network to learn body shape (identity) and deformation. However, similar body shapes would tend to deform similarly, especially if the dimensions aren't too different. Thus, we intentionally modified our unclothed body shape making it too thick/thin in order to represent inaccuracies in the assumed body shape of the user. For each modified body shape, we use the same training data for cloth shapes noting that this merely changes the values of  $\mathbf{dx}$  and thus the cloth image stored for each pose. As compared to the high variance in  $\mathbf{dx}$  caused by folds and wrinkles, changing the body shape makes lower frequency modifications that are not too difficult for the network to learn. Surprisingly, the erroneously modified too thick/thin body shapes had almost no effect on the network's prediction ability indicating that our approach is robust to inaccuracies in the unclothed body shape. See Figure 11 and Figure 13 left.

**Skinning Artifacts:** Whether using an accurate unclothed body shape or not, clearly body skinning is not a solved problem; thus, we modified our skinning scheme to intentionally create significant artifacts using erroneous bone weights. Then, we trained the CNN as before noting

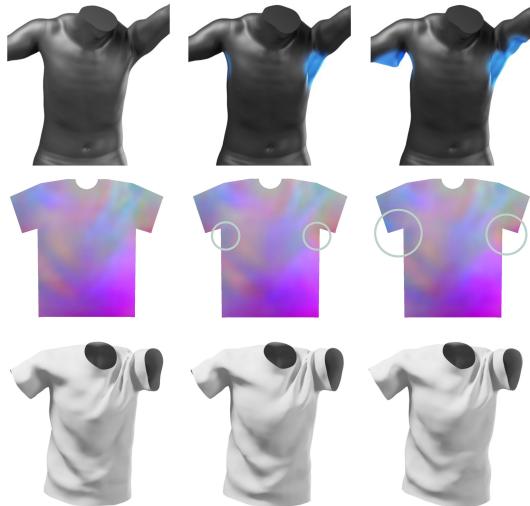


Figure 12: Training the network using a body skinning method that contains artifacts (shown in blue) does not hinder its ability to predict cloth shapes as compared to the ground truth (left column). The cloth images (middle row) readily compensate (see regions annotated by circles) for the skinning artifacts leading to similar cloth shapes (bottom row) in all three cases.

that the cloth training images will be automatically modified whenever skinning artifacts appear. The erroneous skinning artifacts had almost no effect on the network's prediction ability indicating that our approach is robust to inaccuracies in the body skinning. See Figure 12 and Figure 13 right.

**Cloth Patches:** As mentioned in Section 4, we can segment the cloth mesh into smaller semantically coherent pieces, and then train separate networks on these individual patches to achieve better results. Figure 8 shows that the models trained on the patches yield lower errors. See Figure 14 for visual comparison. One can use a variety of methods to achieve visually continuous and smooth results across the patch boundaries. For example, one can precompute the PCA bases of the whole mesh on the training samples, and then project the stitched mesh onto a sub-



Figure 13: The CNN predicts the correct cloth shape even when the unclothed shapes are so erroneous that they penetrate the clothing. In these cases, the network merely predicts offsets in the negative normal direction. Left: dressed version of Figure 11 right. Right: dressed version of Figure 12 right.

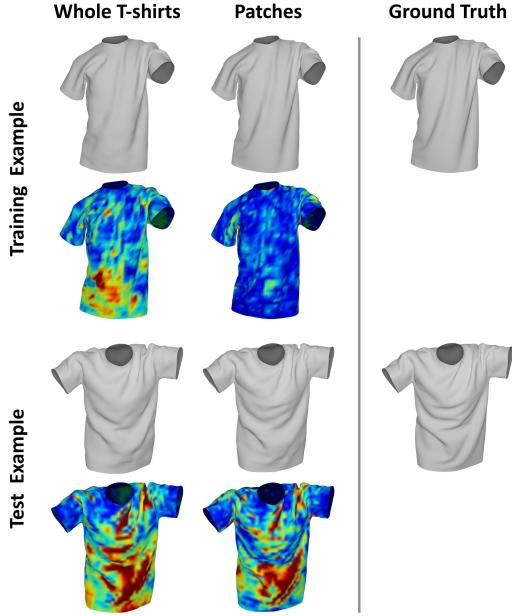


Figure 14: Comparison of network predictions/errors from model trained on whole T-shirts versus models trained on patches. The latter can better capture folds and wrinkles.

set of those bases. Since the simulation/captured data do not have kinks at patch boundaries, the PCA bases also will not have kinks at boundaries unless one gets into ultra-high frequency modes that represent noise; thus, reconstructing the network predicted results using a not too high number of PCA bases acts as a filter to remove discontinuities at patch boundaries. In our experiments, using 2048 components leads to the best filtering results, see Figure 15.

**Necktie:** For generality we also show a necktie example, which unlike the T-shirt, exhibits much larger deformation as the body moves; the maximum per-vertex offset value can be over 50 centimeters. See Figure 16, and Appendix F for more details.

## 8. Conclusion and Future Work

In conclusion, we have introduced a new flexible pixel-based framework for representing virtual clothing shapes as offsets from the underlying body surface, and further illustrated that this data structure is especially amenable to learning by convolutional neural networks in the image space.

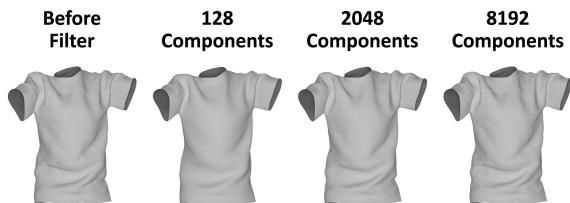


Figure 15: PCA filtering on a stitched mesh from predicted patches (an example from the test set).

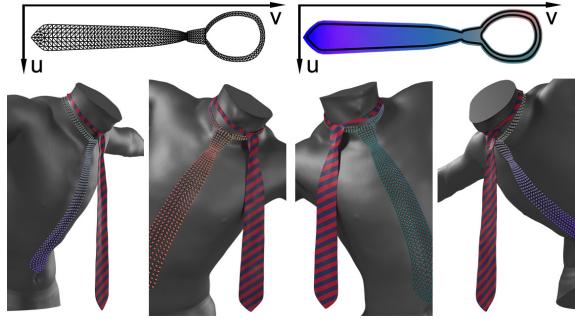


Figure 16: Top left: triangle mesh of necktie in pattern space. Top right: a necktie image. Bottom: network predictions of neckties in different poses (also, necktie pixels are shown embedded on the skinned body mesh).

Our preliminary experiments show promising results with CNNs successfully predicting garment shapes from input pose parameters, and we are optimistic that the results could be further improved with better and more advanced network technologies.

For future work, we would like to leverage real-world captured cloth data and generalize our approach to a larger variety of garment types and materials as well as body types. We would also like to explore alternative network architectures, loss functions, and training schemes to enhance the visual quality of the predictions. In addition, while our evaluation on the motion capture sequence already appears quite smooth in time, we would like to experiment with techniques such as 3D CNNs and recurrent neural networks to achieve better temporal coherency.

## Acknowledgements

Research supported in part by ONR N000014-13-1-0346, ONR N00014-17-1-2174, ARL AHPCRC W911NF-07-0027, and generous gifts from Amazon and Toyota. In addition, we would like to thank Radek Grzeszczuk for initiating conversations with Amazon and those interested in cloth there, Andrew Ng for many fruitful discussions on cloth for e-commerce, and both Reza and Behzad at ONR for supporting our efforts into machine learning. Also, we greatly appreciate the remarkable things that Jen-Hsun Huang (Nvidia) has done for both computer graphics and machine learning; this paper in particular was motivated by and enabled by a combination of the two (and inspirations from chatting with him personally). NJ is supported by a Stanford Graduate Fellowship, YZ is supported by a Stanford School of Engineering Fellowship, and ZG is supported by a VMWare Fellowship. NJ would also like to personally thank a number of people who helped contribute to our broader efforts on data-driven cloth, including Davis Rempe, Haotian Zhang, Lucy Hua, Zhengping Zhou, Daniel Do, and Alice Zhao.

## Appendices

### A. Cloth/Body Texture Space

It is important to note that we do not assume a one-to-one mapping between the cloth texture coordinates and the body texture coordinates; rather, we need only a mapping from the cloth texture space to the body texture space (invertibility is not required). This allows for the ability to handle more complex real-life clothing such as the collars of shirts and jackets, which would naturally be embedded to the shoulder/chest areas on the body causing them to overlap with other parts of the same garment (and/or other garments). See for example Figure 17.



Figure 17: Collars such as this one are more naturally associated with the chest than the neck. Our approach can handle such a non-invertible many-to-one mapping from the cloth texture space to the body texture space.

### B. Image Editing

Our pixel-based cloth framework enables convenient shape modification via image editing. Since the displacement maps represent offsets from the locations of the embedded cloth pixels on the skinned body surface, we can achieve easy and rather intuitive control by manipulating their RGB values in the image space. For example, adjusting the brightness of the texture coordinates channels (red and green) induces shifting of the cloth shape, whereas adjusting the normal directions channel (blue) leads to shrinking or inflation. Moreover, one can add features to the cloth shape by painting in image space, especially using a blue

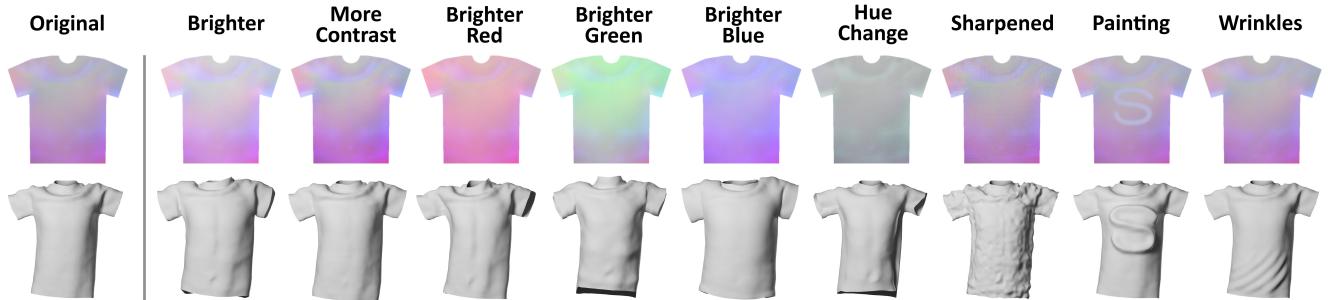
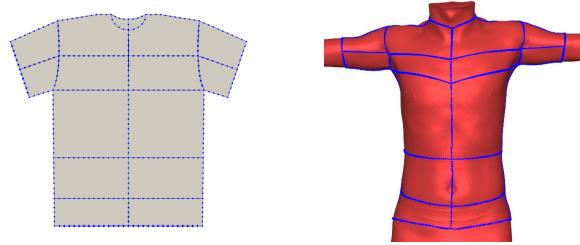


Figure 18: Various image editing operations applied to a given cloth image (top row) and their corresponding modified cloth shapes (bottom row). Note that although the wrinkle lines blended into the image in the last column are hard to see, the resulting wrinkles are clearly visible.

brush that changes the offset values in the normal directions. Furthermore, one can transfer features from another cloth image by selective image blending, *e.g.*, adding wrinkle lines. See Figure 18 for a set of modified cloth shapes resulting from image editing operations.

### C. Cage and Patch Based Cloth

Given a cloth mesh, we can create a wire “cage” that defines a support structure for its overall shape, *e.g.*, by tracing its intrinsic seams, characteristic body loops (*e.g.*, chest, waist, hip, arms), etc. See Figure 19a. The cage structure conveniently divides the cloth surface into patches bound by boundary curves, and this cage and patch based computational structure affords a hierarchical data-driven framework where different specialized methods can be applied at each level. Note that the same cage structure is also defined on the body surface to facilitate correspondences, see Figure 19b.



(a) Cage structure defined on a T-shirt mesh. (b) Corresponding cage defined on the body.

Figure 19: The cage is defined on the cloth mesh and the body surface as a lower-dimensional support structure.

To obtain the shape of the cage when the clothing is dressed on a person, one can interpolate from a set of sparse marker/characteristic key points. That is, given the locations of the key points, one can reconstruct the cage. This can be represented as a constrained optimization problem

to find a smooth curve that passes through the constraint points. Specifically, one can interpolate the points with a piecewise cubic spline curve while attempting to preserve the geodesic lengths between each pair of neighboring points. Alternatively, one could train a neural network to learn to recover the cage from the sparse points.

One can use the reconstructed cage as a boundary condition to fill in the surface patches using a variety of methods. In particular, one can build a blendshapes basis for each patch and select blendshape weights based on the shape of the boundary cage. A cage vertex's basis function can be computed, for example, by solving a Poisson equation on the patch interior with boundary conditions identically zero except at that vertex where the boundary condition is set to unity. Then, any perturbation of the cage can be carried to its interior. For example, given the offsets of the cage from its position on the skinned body in texture and normal coordinates, one can evaluate the basis functions to quickly compute offsets for the interior of the patch.

For a simple illustration, the boundary perturbation in Figure 20a is extended to the patch interior using the Poisson equation basis functions to obtain the result shown in Figure 20b. To achieve more interesting deformations, one could use anisotropic Poisson equations to construct the basis functions. Figure 20c shows the boundary perturbation in Figure 20a evaluated using anisotropic basis functions. Also, see Figures 21, 22, and 23. One could also create basis functions via quasistatic simulations.

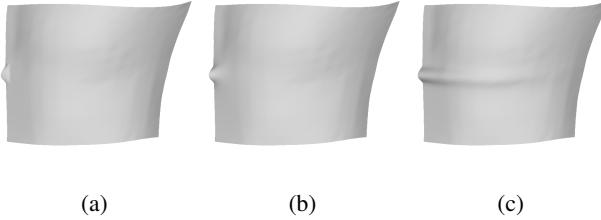


Figure 20: An input boundary perturbation (a) can be used in a blendshape basis to obtain interior patch deformations: isotropic (b), anisotropic (c).

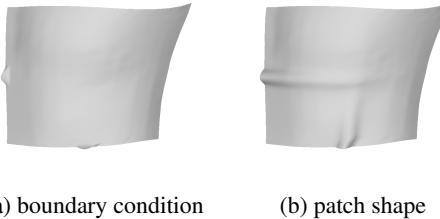


Figure 21: Two small perturbations on the boundary yields two folds coming together.



Figure 22: A sine wave perturbation on the boundary yields smooth wrinkles.



Figure 23: An S-shaped boundary yields an overturning wave shape.

Moreover, one can use this cage structure as an intermediary for designing and dressing garments onto the body leveraging the correspondence to body curves shown in Figure 19.

## D. Dataset Generation

### D.1. Body Modeling

We use a commercial solution<sup>3</sup> to scan a person in the T-pose. The initially acquired mesh is manually remeshed to a quad mesh, and then rigged to the skeleton shown in Figure 24 using Blender [9].

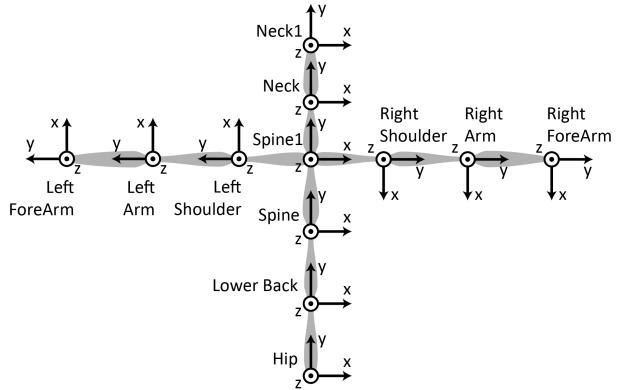


Figure 24: Skeleton structure, bone name, and axis orientation definition.

<sup>3</sup><https://www.artec3d.com/>

## D.2. Intentionally Modified Body Shapes

In order to demonstrate the resilience of our network predictions to errors due to an incorrect assumption of the underlying body shape, we manually sculpted the scanned body and generated a number of intentionally incorrect body shapes. With the normal body shape designated 0 and the manually sculpted body shape designated 1, we create a shape change parameter that ranges from  $-1$  to  $2$  as seen in Figure 25. The plot shows data points for 7 of our trials: the points at zero represent the original body shape, and the other 6 pairs of points represent the results obtained by training the network on the correct cloth shapes using incorrect unclothed body shape assumptions.

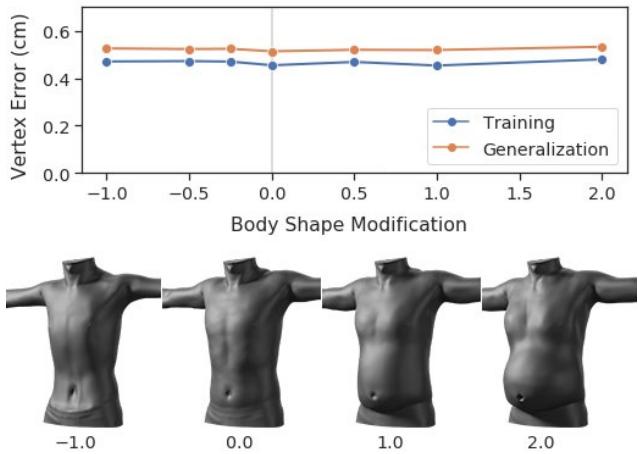


Figure 25: Training and generalization average per-vertex prediction errors (top plot) of models trained on offsets computed from different underlying body shapes (bottom row). As the body shape deviates from the true body shape (0 on the x-axis), the performance of the trained models stay roughly constant.

Also, note that the two versions of skinning with artifacts used in the paper were created on the original rigged body by manually painting weights of upper arm on the torso, and painting weights of upper arm on both the torso and the opposite arm, respectively.

## D.3. Pose Sampling

While one could sample from an empirical distribution learned from motion capture data (*e.g.*, [1]), we prefer an alternative sampling scheme in order to better cover the entire space of possible poses that can affect the T-shirt shape. Since we only focus on the T-shirt interaction with the human body, we define the skeleton structure only for the upper body, as shown in Figure 24. We fix the position and rotation for the hip (root) joint, since we are mainly interested in static poses as a first step. We set the joint limits according to [69], where each joint angle has both a posi-

tive limit and a negative limit for each rotation axis relative to the rest T-pose. For the bones on the vertical center line in Figure 24 (lower back, spine, spine1, neck, and neck1), we sample the rotation angles for each axis from a mixture of two half-normal distributions, each accounting for one direction of the rotation. Since we don't have such a strong prior for shoulder and arm bones, their x-axis rotation angles (azimuth) are uniformly sampled first, their z-axis rotation angles (altitude) are then uniformly sampled in the transformed space of the sine function, and finally their y-axis rotation angles are also uniformly sampled. The rotations are applied in the order of x, z, and y. Finally, a simple pruning procedure is applied to remove poses with severe nonphysical self-penetrations. This is accomplished by selecting 106 vertices from both arm parts as shown in Figure 26 and testing if any of these vertices is inside the torso. The distributions of the sampled joint angles are shown in Figure 27.



Figure 26: The body is segmented into three overlapping parts (left arm, right arm, and torso). The vertices selected for collision detection are shown as light gray dots.

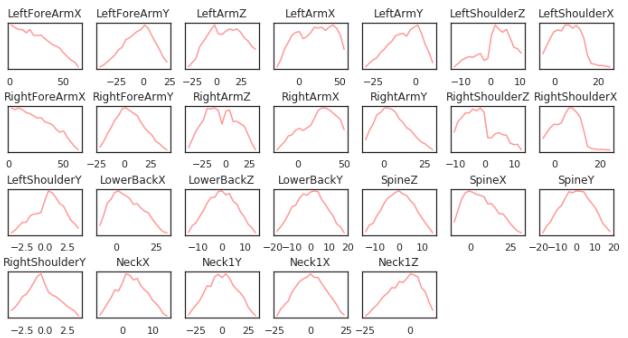


Figure 27: Plots of joint angle distributions in our dataset.

## D.4. Mesh Generation

We carefully construct the rest state of our simulation mesh to be faithful to real-world garments by employing a reverse engineering approach where we cut up a garment along its seam lines, scan in the pieces, and then digitally stitch them back together, as shown in Figure 28. The T-shirt triangle mesh is 67 cm long and contains 3K vertices. Although we try to cut the clothing into pieces such that

each piece is as flat as possible to approximate flat design patterns, this may not always be achievable and the flattened versions thus obtained would not be in their intrinsic rest states leading to errors in the simulation input and distortions in the vertex UV map. However, such issues would be largely alleviated if one could obtain the original flat patterns from fashion designers.

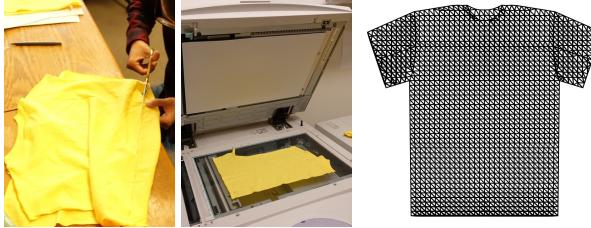


Figure 28: Illustration of the garment mesh generation process. Left: a T-shirt is cut into pieces. Middle: the pieces are scanned in. Right: the digitally stitched T-shirt mesh.

## D.5. Skinning the T-shirt

To shrink wrap the T-shirt onto the body, we first define the cage structure on both the body and the T-shirt as shown in Figure 19, and then compute displacements on the T-shirt cage vertices that would morph them to the body cage; these displacement values are used as boundary conditions to solve a set of Poisson equations (see e.g. [3, 18]) for displacements on T-shirt interior vertices. A level set is built for the body for collision detection [13], and any T-shirt vertices that are inside the body are detected and pushed out to their closest points on the body surface.

Since the morphed T-shirt mesh can exhibit rather large and non-uniform distortion, we run a simulation using a mass-spring system to reduce distortion and achieve a better set of barycentric embedding weights for the T-shirt vertices, see Figure 29. This is done in an iterative manner. At each step, while constraining the T-shirt mesh to stay on the body surface, for each vertex  $v$  we compute the average ratio  $\bar{\alpha}_v = (1/\deg(v)) \sum_{e \in E(v)} (l_e / \bar{l}_e)$  of the current edge length  $l_e$  to the rest edge length  $\bar{l}_e$  over its incident edges  $E(v)$ . Then for each edge  $e$  with endpoints  $a$  and  $b$ , its target edge length is set to  $(1/2)(\alpha_a + \alpha_b)\bar{l}_e$ . This process essentially tries to equalize the amount of distortion for the edges incident to the same vertex, and is repeated until convergence.

## D.6. Simulation

We simulate the T-shirt mesh on each sampled pose using a physical simulator [2] with gravity, elastic and damping forces, and collision, contact, and friction forces until static equilibrium is reached. To make our simulation robust



Figure 29: Shrink wrapping a T-shirt mesh onto a body in the rest pose. Left: shrink wrapped T-shirt mesh obtained by solving a Poisson equation that uses a guide “cage” (in blue) as a boundary condition to morph the T-shirt mesh onto the body. Middle: this initial version has area distortion, where red indicates stretching and blue indicates compression. Right: after simulation, the distortion has been reduced and more uniformly spread out so that the cloth pixels can be embedded at better locations. Note that since the T-shirt is constrained to be on the body surface, distortion is not fully eliminated.

to skinning artifacts that lead to cloth self-interpenetrations especially in the armpit regions, we decompose the body into three parts: the left arm, the right arm, and the torso (see Figure 26), and associate each cloth mesh vertex with one body part as its primary collision body.

After running the simulations, we further run analysis on the resulting cloth meshes and remove any shape that exhibits large distortion to reduce noise in the function to be learned. Specifically, if the area of any triangle in a sample compresses by more than 75% or expands by more than 100%, then we discard that sample. Figure 31 shows that the amount of face area distortion is moderate (top), and the amount of self-interpenetrations is very small in the dataset (bottom). Figure 32 shows that the cleaned dataset contains a similar distribution of poses as the original one. In line with Appendix B and Figure 18, one could also use image analysis on the cloth images in order to identify and prune samples that are deemed undesirable.

This leads to a total of 20,011 samples that we use to train and evaluate our models, see Figure 30 for some examples. We create a separate UV map for the front side and the back side of the T-shirt.

## D.7. Patches

There are 28 patches on the front and the back side of the T-shirt (14 each). Whereas we train on  $256 \times 256$  cloth images for the whole T-shirt, for each patch we make a  $160 \times 160$  crop from a higher resolution  $512 \times 512$  cloth image centered at the center of its axis-aligned bounding box. The cropped patch contains 16 pixels outside of the patch to capture the surrounding context, and the loss is computed on this enlarged patch.

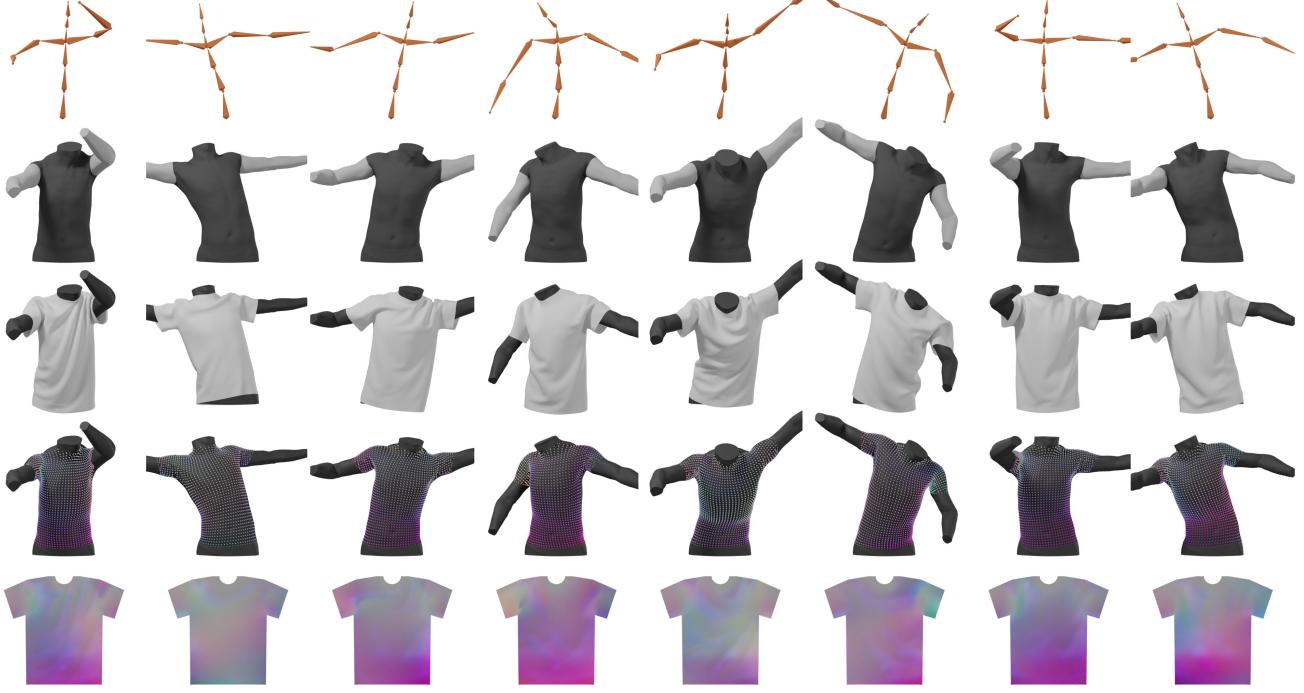


Figure 30: Random samples from our generated dataset. First row: skeletal poses. Second row: three overlapping collision bodies. Third row: simulated T-shirts. Fourth row: cloth pixels. Fifth row: cloth images (front side).

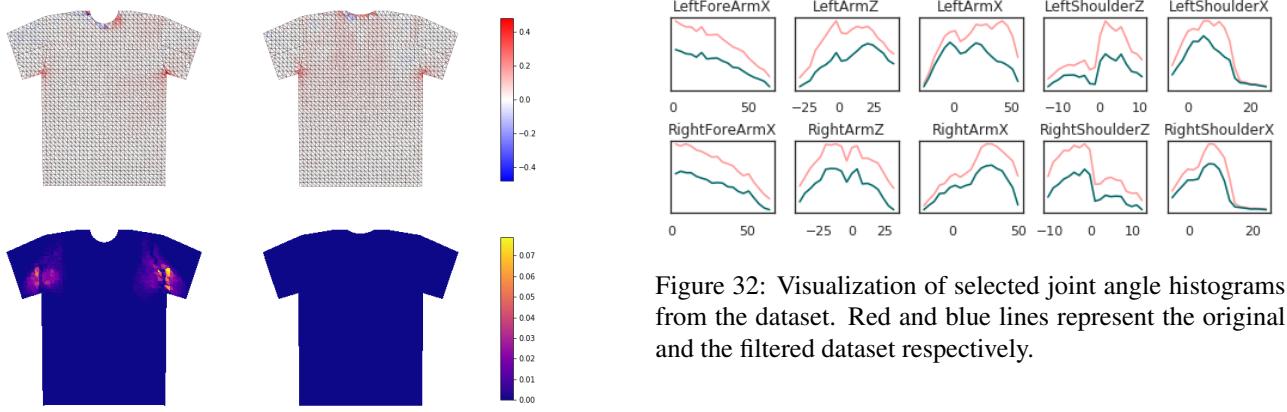


Figure 31: Mesh statistics of the simulated T-shirts. Top: front and back side average face area distortion, measured as the ratio of the current area to the rest area minus one. Bottom: front and back side per-face self-interpenetrations, measured as fraction of samples with self-interpenetrations in the dataset.

## E. Networks

### E.1. Architecture

For predicting cloth images of the whole T-shirt, we start with the 90 dimensional input pose parameters and first ap-

ply transpose convolution followed by ReLU activation to obtain an initial  $8 \times 8 \times 384$  dimensional feature map. Then we successively apply groups of transpose convolution (filter size  $4 \times 4$  and stride 2), batch normalization, and ReLU activation until we reach the output resolution of  $256 \times 256$ . Each time the spatial resolution doubles and the number of channels halves. Finally, a convolution layer (filter size  $3 \times 3$  and stride 1) brings the number of channels to 6. The network contains 3.79 million parameters.

We use the same network architecture for all 28 patches. We start with the 90 dimensional input pose parameters, and first apply a linear layer to obtain a  $5 \times 5 \times 512$  dimensional feature map. Then similar to the network for the whole T-shirt, we successively apply groups of transpose convolu-

tion (filter size  $4 \times 4$  and stride 2), batch normalization, and ReLU activation until we reach the target resolution of  $160 \times 160$ . Again, a final convolution layer (filter size  $3 \times 3$  and stride 1) brings the number of channels to 3. The network contains 3.96 million parameters.

## E.2. Loss Functions

The base loss term for grid pixel values is

$$\mathcal{L}_{\text{grid-pix}}(\mathbf{I}^{pd}, \mathbf{I}^{gt}) = \frac{\sum_{i,j} W(i,j) \|\mathbf{I}^{pd}(i,j) - \mathbf{I}^{gt}(i,j)\|}{\sum_{i,j} W(i,j)}, \quad (1)$$

where  $\mathbf{I}^{gt}$  denotes ground truth grid pixel values,  $\mathbf{I}^{pd}$  denotes predicted grid pixel values,  $W$  denotes the Boolean padded mask of the UV map, and  $i, j$  are indices into the image width and height dimensions.

The additional loss term for the normal vectors is

$$\mathcal{L}_{\text{normal}}(\mathbf{I}^{pd}, \mathbf{I}^{gt}) = \frac{1}{N_v} \sum_v (1 - \mathbf{n}_v^{pd}(\mathbf{I}^{pd}) \cdot \mathbf{n}_v^{gt}), \quad (2)$$

where we compute a predicted unit normal vector  $\mathbf{n}_v^{pd}$  on each vertex  $v$  using the predicted grid pixel values  $\mathbf{I}^{pd}$  (by first interpolating them back to cloth pixels and adding these per-vertex offsets to their embedded locations to obtain predicted vertex positions) and use the cosine distance to the ground truth unit normal vector  $\mathbf{n}_v^{gt}$  as the loss metric.  $N_v$  is the number of vertices.

Table 1 shows the average per-vertex prediction errors and the normal vector errors from our convolutional decoder network trained with different loss terms on our training set and test set. The weight on the loss on normal vectors is set to 0.01.

Table 1: Average per-vertex position error (in cm) and unit normal vector error (cosine distance) of our convolutional decoder network trained with different loss functions.  $L_1$  and  $L_2$  refer to the loss function used on the Cartesian grid pixels. N refers to normal loss.

Loss	Training Error		Generalization Error	
	Vertex	Normal	Vertex	Normal
$L_1$	0.33	0.020	0.44	0.027
$L_2$	0.35	0.017	0.47	0.028
$L_2 + N$	0.37	0.0075	0.51	0.029

## E.3. Fully Connected Networks

We illustrate that our cloth pixel framework provides for offset functions that can be approximated via a lower dimensional PCA basis, and that a fully connected network can be trained and subsequently generalized to predict cloth shapes. Furthermore, we compare functions of offsets represented in different spaces, as well as functions of positions

in the root joint frame. See Table 2 and Figure 33. We train a fully connected network with two hidden layers each with 256 units and ReLU activation for all the functions. The networks trained to predict PCA coefficients indeed have better visual quality and deliver better training and generalization errors compared to the networks trained to directly predict per-vertex values. Our experiments also show that ReLU activation leads to faster convergence and similar results compared to the Tanh activation used in [6].

Table 2: Average per-vertex position error (in cm) of the fully connected network trained with and without PCA in different spaces. “Off. Loc.” refers to offsets represented in local tangent-bitangent-normal frames. “Off. Root.” refers to offsets represented in the root joint frame. “Pos. Root.” refers to positions in the root frame.

Model	Training Error	Generalization Error
Off. Loc. Direct	0.65	0.67
Off. Loc. 128 PC	0.50	0.55
Off. Root. Direct	0.69	0.72
Off. Root. 128 PC	0.53	0.58
Pos. Root. Direct	0.63	0.68
Pos. Root. 128 PC	0.58	0.65

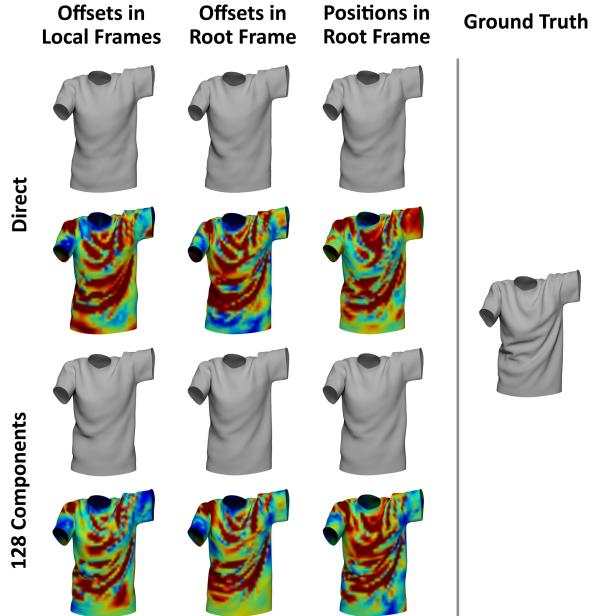


Figure 33: Comparison of fully connected network predictions and errors from models trained on different functions defined on our cloth pixels.

## F. Neckties

Similar to the T-shirt dataset, we generate 9,999 poses by randomly sampling rotation angles on 4 joints along the center line (lower back, spine, neck, and neck1), *i.e.*, our input pose parameters are only 36 dimensional. The dataset is divided into a training set of 7,999 poses, a regularization set of 1,000 poses, and a test set of 1,000 poses. In this example, we use one UV map for the entire mesh, and since the necktie has a much narrower UV map in the texture space, we modify our network architecture to predict a rectangular image with aspect ratio 1 : 4 and size  $64 \times 256$  containing 3 channels.  $L_1$  loss is used for the Cartesian grid pixels. The weight on the normal loss is set to 0.1. We further add an  $L_1$  loss term on the edge lengths with weight 0.1 to ensure a smooth boundary:

$$\mathcal{L}_{\text{edge\_len}}(\mathbf{I}^{pd}) = \frac{1}{N_e} \sum_e ||l_e^{pd}(\mathbf{I}^{pd}) - l_e^{gt}||, \quad (3)$$

where we compute a predicted edge length  $l_e^{pd}$  for each edge  $e$  using the predicted grid pixel values  $\mathbf{I}^{pd}$  (also by first interpolating them back to cloth pixels and adding these per-vertex offsets to their embedded locations to obtain predicted vertex positions) and compare to the ground truth edge lengths  $l_e^{gt}$ .  $N_e$  is the number of edges in the mesh. We represent the offsets  $\mathbf{dx}$  in the root joint frame, *i.e.*,  $(\Delta x, \Delta y, \Delta z)$ , instead of the local tangent-bitangent-normal frames  $(\Delta u, \Delta v, \Delta n)$ . This is more natural for the neckties, because unlike the T-shirts, they have a much larger range of displacements from the body surface while also exhibiting few high frequency wrinkles.

Since the neckties contain less high frequency variation and the output image size is smaller, a smaller network is used to learn the necktie images. Starting from the 36 dimensional input pose parameters, we first apply a linear layer with 128 hidden units and then apply another linear layer to obtain a  $8 \times 8 \times 64$  dimensional feature map. After that, we successively apply groups of transpose convolution, batch normalization, and ReLU activation as above until we reach the target resolution of  $64 \times 256$ . Then, a final convolution layer (filter size  $3 \times 3$  and stride 1) brings the number of channels to 3. The network contains 2.16 million parameters.

## References

- [1] Cmu graphics lab motion capture database. <http://mocap.cs.cmu.edu/>. 6, 11
- [2] Physbam: physically based animation. <http://physbam.stanford.edu/>. 12
- [3] D. Ali-Hamadi, T. Liu, B. Gilles, L. Kavan, F. Faure, O. Palombi, and M.-P. Cani. Anatomy transfer. *ACM Trans. Graph.*, 32(6), Nov. 2013. 12
- [4] B. Allen, B. Curless, and Z. Popović. Articulated body deformation from range scan data. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 612–619. ACM, 2002. 2
- [5] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 408–416, New York, NY, USA, 2005. ACM. 1, 2
- [6] S. W. Bailey, D. Otte, P. Dilorenzo, and J. F. O'Brien. Fast and deep deformation approximations. *ACM Trans. Graph.*, 37(4):119:1–119:12, July 2018. 2, 5, 14
- [7] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 862–870, New York, NY, USA, 2003. ACM. 1
- [8] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 417–424, New York, NY, USA, 2000. 4
- [9] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2018. 10
- [10] J. F. Blinn. Simulation of wrinkled surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '78, pages 286–292, New York, NY, USA, 1978. ACM. 3
- [11] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Commun. ACM*, 19(10):542–547, Oct. 1976. 3
- [12] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.*, 21(3):594–603, July 2002. 1
- [13] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 28–36, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. 1, 12
- [14] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *arXiv:1611.08097*, 2016. 3
- [15] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014. 3
- [16] E. E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, 1974. AAI7504786. 3

- [17] X. Chen, B. Zhou, F. Lu, L. Wang, L. Bi, and P. Tan. Garment modeling with a depth camera. *ACM Trans. Graph.*, 34(6):203:1–203:12, Oct. 2015. 1
- [18] M. Cong, M. Bao, J. L. E. K. S. Bhat, and R. Fedkiw. Fully automatic generation of anatomical face simulation models. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA ’15, pages 175–183, New York, NY, USA, 2015. ACM. 12
- [19] R. L. Cook. Shade trees. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’84, pages 223–231, New York, NY, USA, 1984. ACM. 3
- [20] R. Danecek, E. Dibra, A. C. Öztireli, R. Ziegler, and M. Gross. Deepgarment : 3d garment shape estimation from a single image. *Computer Graphics Forum (Proc. Eurographics)*, (2), 2017. 2
- [21] E. de Aguiar, L. Sigal, A. Treuille, and J. K. Hodgins. Stable spaces for real-time clothing. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH ’10, pages 106:1–106:9, New York, NY, USA, 2010. ACM. 1, 2
- [22] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS’16, pages 3844–3852, USA, 2016. 3
- [23] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, pages 557–574, 2018. 2, 5
- [24] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics (2Nd Ed. In C): Principles and Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996. 3
- [25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. 4
- [26] P. Guan, L. Reiss, D. Hirshberg, A. Weiss, and M. J. Black. DRAPE: DRessing Any PErsom. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 31(4):35:1–35:10, July 2012. 1, 2
- [27] F. Hahn, B. Thomaszewski, S. Coros, R. W. Sumner, F. Cole, M. Meyer, T. DeRose, and M. Gross. Subspace clothing simulation using adaptive bases. *ACM Trans. Graph.*, 33(4):105:1–105:9, July 2014. 2
- [28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015. 1
- [29] P. S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, Nov 1986. 3
- [30] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv:1506.05163*, 2015. 3
- [31] A. Hilsmann and P. Eisert. Tracking and retexturing cloth for real-time virtual clothing applications. In *Computer Vision/Computer Graphics Collaboration Techniques*, pages 94–105. Springer Berlin Heidelberg, 2009. 1
- [32] D. A. Hirshberg, M. Loper, E. Rachlin, and M. J. Black. Coregistration: Simultaneous alignment and modeling of articulated 3d shape. In *Computer Vision – ECCV 2012*, pages 242–255. Springer Berlin Heidelberg, 2012. 2
- [33] A. Jacobson and O. Sorkine. Stretchable and twistable bones for skeletal shape deformation. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA ’11, pages 165:1–165:8. ACM, 2011. 2
- [34] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. The affine particle-in-cell method. *ACM Trans. Graph.*, 34(4):51:1–51:10, July 2015. 3
- [35] N. Jin, W. Lu, Z. Geng, and R. P. Fedkiw. Inequality cloth. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA ’17, pages 16:1–16:10, New York, NY, USA, 2017. ACM. 5
- [36] L. Kavan, S. Collins, J. Žára, and C. O’Sullivan. Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D ’07, pages 39–46. ACM, 2007. 1, 2
- [37] L. Kavan, D. Gerszewska, A. W. Bargteil, and P.-P. Sloan. Physics-inspired upsampling for cloth simulation in games. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH ’11, pages 93:1–93:10, New York, NY, USA, 2011. ACM. 1
- [38] L. Kavan and J. Žára. Spherical blend skinning: A real-time deformation of articulated models. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D ’05, pages 9–16. ACM, 2005. 1, 2
- [39] D. Kim, W. Koh, R. Narain, K. Fatahalian, A. Treuille, and J. F. O’Brien. Near-exhaustive precomputation of secondary cloth effects. *ACM Trans. Graph.*, 32(4):87:1–87:8, July 2013. 1, 2
- [40] T.-Y. Kim, N. Chentanez, and M. Müller-Fischer. Long range attachments - a method to simulate inextensible clothing in computer games. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA ’12, pages 305–310. Eurographics Association, 2012. 1
- [41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6
- [42] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 1
- [43] P. G. Kry, D. L. James, and D. K. Pai. Eigenskin: Real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA ’02, pages 153–159. ACM, 2002. 2
- [44] T. Kurihara and N. Miyata. Modeling deformable human hands from medical images. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA ’04, pages 355–363. Eurographics Association, 2004. 2
- [45] Z. Lähner, D. Cremers, and T. Tung. Deepwrinkles: Accurate and realistic clothing modeling. In *Computer Vision – ECCV 2018*, pages 698–715, 2018. 2
- [46] J. Lander. Skin them bones: Game programming for the web generation. *Game Developer Magazine*, May 1998. 2

- [47] B. H. Le and J. K. Hodgins. Real-time skeletal skinning with optimized centers of rotation. *ACM Trans. Graph.*, 35(4):37:1–37:10, July 2016. 2
- [48] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’00, pages 165–172, 2000. 2
- [49] J. P. Lewis, K. ichi Anjyo, T. Rhee, M. Zhang, F. H. Pighin, and Z. Deng. Practice and theory of blendshape facial models. In *Eurographics*, 2014. 2
- [50] Y. Li, S. Liu, J. Yang, and M.-H. Yang. Generative face completion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 4
- [51] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1
- [52] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015. 1, 2
- [53] N. Magnenat-Thalmann, F. Cordier, H. Seo, and G. Papagianakis. Modeling of bodies and clothes for virtual environments. In *2004 International Conference on Cyberworlds*, pages 201–208, Nov 2004. 1
- [54] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface ’88*, pages 26–33, 1988. 1, 2
- [55] L. Margolin. Introduction to “an arbitrary lagrangian-eulerian computing method for all flow speeds”. *J. Comput. Phys.*, 135(2):198–202, Aug. 1997. 3
- [56] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, ICCVW ’15, pages 832–840. IEEE Computer Society, 2015. 3
- [57] M. Müller and N. Chentanez. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA ’10, pages 85–92, Goslar Germany, Germany, 2010. Eurographics Association. 1
- [58] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *J. Vis. Commun. Image Represent.*, 18(2):109–118, Apr. 2007. 1
- [59] A. Neophytou and A. Hilton. A layered model of human body and garment deformation. In *Proceedings of the 2014 2Nd International Conference on 3D Vision - Volume 01*, 3DV ’14, pages 171–178, Washington, DC, USA, 2014. IEEE Computer Society. 1, 2
- [60] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 6
- [61] G. Pons-Moll, S. Pujades, S. Hu, and M. J. Black. Clothcap: Seamless 4d clothing capture and retargeting. *ACM Trans. Graph.*, 36(4):73:1–73:15, July 2017. 1, 2
- [62] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 39(6):1137–1149, June 2017. 1
- [63] N. Robertini, E. De Aguiar, T. Helten, and C. Theobalt. Efficient multi-view performance capture of fine-scale surface detail. In *Proceedings - 2014 International Conference on 3D Vision, 3DV 2014*, pages 5–12, 02 2015. 1
- [64] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. 1
- [65] J. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 1999. 5
- [66] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv 1409.1556*. 09 2014. 1
- [67] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia. Variational autoencoders for deforming 3d mesh models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3
- [68] H. Wang, F. Hecht, R. Ramamoorthi, and J. F. O’Brien. Example-based wrinkle synthesis for clothing animation. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH ’10, pages 107:1–107:8, New York, NY, USA, 2010. ACM. 1, 2
- [69] M. Whitmore, J. Boyer, and K. Holubec. Nasa-std-3001, space flight human-system standard and the human integration design handbook. 2012. 11
- [70] W. Xu, N. Umentani, Q. Chao, J. Mao, X. Jin, and X. Tong. Sensitivity-optimized rigging for example-based real-time clothing synthesis. *ACM Trans. Graph.*, 33(4):107:1–107:11, July 2014. 1, 2
- [71] J. Yang, J.-S. Franco, F. Hetroy-Wheeler, and S. Wuhrer. Analyzing clothing layer deformation statistics of 3d human motions. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1, 2, 5
- [72] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 4