A Pixel-Based Framework for Data-Driven Clothing: Appendices

A. Cloth/Body Texture Space

It is important to note that 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 1.

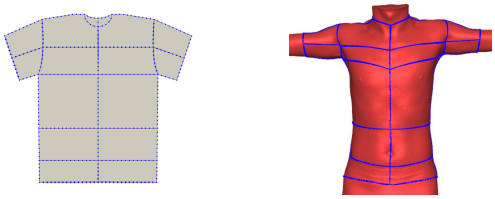
Figure 1: Collars such as this one are more naturally associated with the chest than the neck. Our approach can handle such a non-invertible many-to-one mapping from the cloth texture space to the body texture space.

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 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 2 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 3a. 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 3b.



(a) Cage structure defined on a T-shirt mesh.

(b) Corresponding cage defined on the body.

Figure 3: 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 loca-

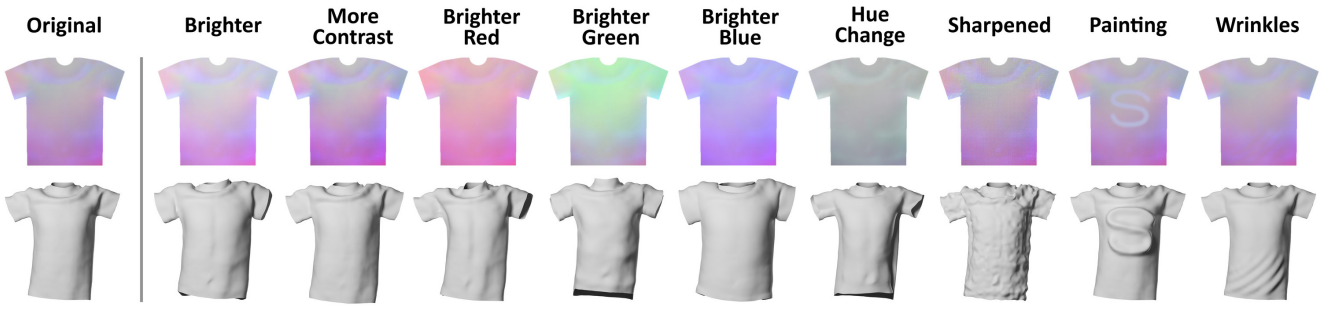


Figure 2: Various image editing operations applied to a given cloth image (top row) and their corresponding modified cloth shapes (bottom row). Note that although the wrinkle lines blended into the image in the last column are hard to see, the resulting wrinkles are clearly visible.

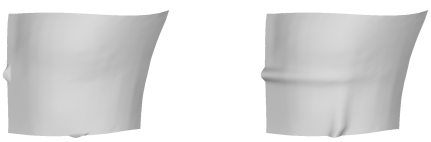
tions of the key points, one can reconstruct the cage. This can be represented as a constrained optimization problem to find a smooth curve that passes through the constraint points. Specifically, one can interpolate the points with a piecewise cubic spline curve while attempting to preserve the geodesic 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 4a is extended to the patch interior using the Poisson equation basis functions to obtain the result shown in Figure 4b. To achieve more interesting deformations, one could use anisotropic Poisson equations to construct the basis functions. Figure 4c shows the boundary perturbation in Figure 4a evaluated using anisotropic basis functions. Also, see Figures 5, 6, and 7. One could also create basis functions via quasistatic simulations.

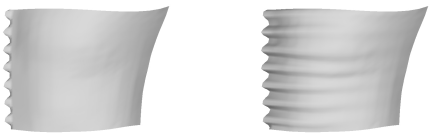
(a) (b) (c)

Figure 4: An input boundary perturbation (a) can be used in a blendshape basis to obtain interior patch deformations: isotropic (b), anisotropic (c).



(a) boundary condition (b) patch shape

Figure 5: Two small perturbations on the boundary yields two folds coming together.



(a) boundary condition (b) patch shape

Figure 6: A sine wave perturbation on the boundary yields smooth wrinkles.

(a) boundary condition (b) patch shape

Figure 7: 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 3.

D. Dataset Generation

This section aims to provide more details on the dataset generation process.

D.1. Body Modeling

The initially acquired mesh from scanning is manually remeshed to a quad mesh, and then rigged to the skeleton shown in Figure 8.

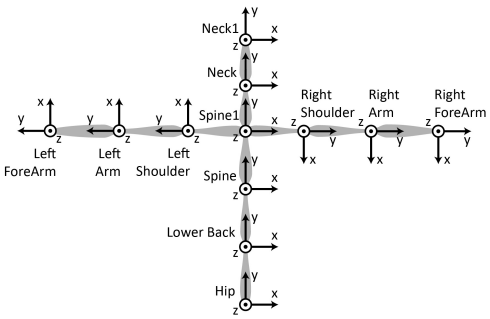


Figure 8: Skeleton structure, bone name, and axis orientation definition.

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 9. 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.

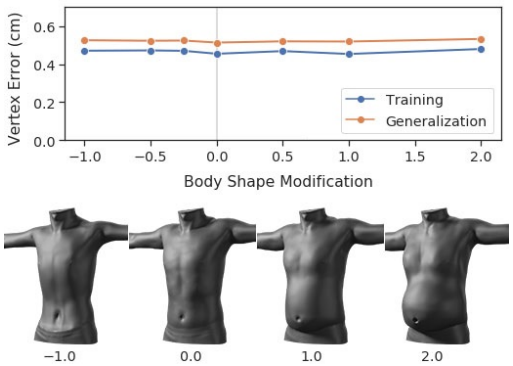


Figure 9: Training and generalization average per-vertex prediction errors (top plot) of models trained on offsets computed from different underlying body shapes (bottom row). As the body shape deviates from the true body shape (0 on the x-axis), the performance of the trained models stay roughly constant.

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.

Figure 10 shows that the CNN can predict the correct cloth shape even when the unclothed shapes are so erroneous that they penetrate the clothing.

Figure 10: Left: predicted T-shirt on a body that has been modified to be too thick. Right: Left: predicted T-shirt on a body that has skinning artifact on the upper arm. In these cases, the network merely predicts offsets in the negative normal direction.

D.3. Pose Sampling

While one could sample from an empirical distribution learned from motion capture data (*e.g.*, [cmu]), 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 8. 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 [WBH12], where each joint angle has both a positive 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 8 (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 11 and testing if any of these vertices is inside the torso. The distributions of the sampled joint angles are shown in Figure 12.

Figure 11: 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.

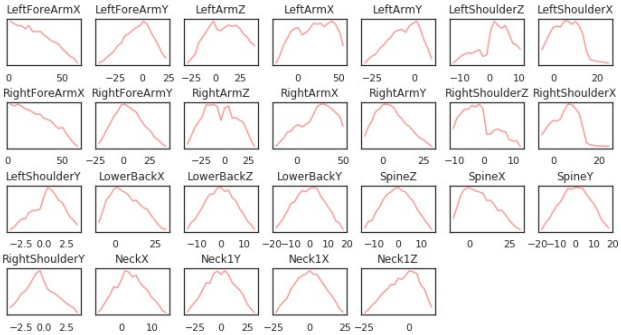


Figure 12: Plots of joint angle distributions in our dataset.

D.4. T-shirt Mesh Generation

To obtain the T-shirt mesh in its rest state, we cut up a T-shirt along its seam lines, scan in the 2D pieces, and then digitally stitch them back together, see Figure 13 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.

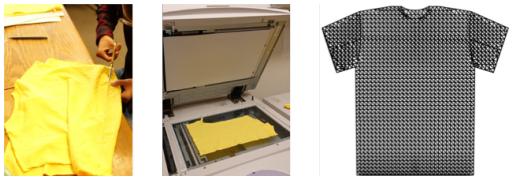


Figure 13: Illustration of the garment mesh generation process. Left: a T-shirt is cut into pieces. Middle: the pieces are scanned. Right: the digitally stitched T-shirt mesh.

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 3, 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.* [AHLG*13, CBE*15]) for displacements on T-shirt interior vertices. A level set is built for the body for collision detection [BMF03], and any T-shirt vertices that are inside the body are detected and pushed out to their closest points on the body surface.

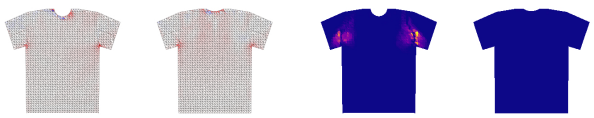
Since the morphed T-shirt mesh can exhibit rather large and non-uniform distortion, we run a simulation using a mass-spring system to reduce distortion and achieve a better set of barycentric embedding weights for the T-shirt vertices, see Figure 14. This is done in an iterative manner. At each step, 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.

Figure 14: 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.

D.6. Simulation

To make our simulation robust 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 11), and associate each cloth mesh vertex with one body part as its primary collision body.

After the simulation, we run a post-processing step to remove shapes with large distortion. 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 15 shows that the amount of face area distortion is moderate (left), and the amount of self-interpenetrations is very small (right) in the dataset. Figure 16 shows that the cleaned dataset contains a similar distribution of poses as the original one. In line with Section B and Figure 2, one could also use image analysis on the cloth images in order to identify and prune samples that are deemed undesirable.



(a) Front and back average face area distortion, measured as the ratio of the current area to the rest area minus one. Red = 40%, white = 0, blue = -40%.

(b) Front and back per-face self-interpenetrations, measured as the fraction of samples with self-interpenetrations. Blue = 0, yellow = 8%.

Figure 15: Mesh statistics of the simulated T-shirts.

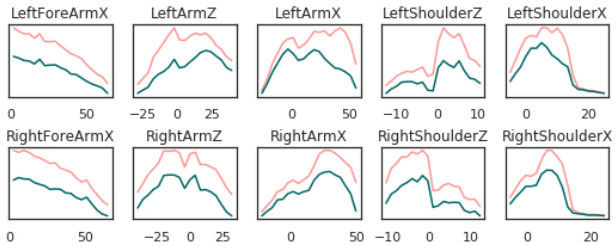


Figure 16: Visualization of selected joint angle histograms from the dataset. Red and blue lines represent the original and the filtered dataset respectively.

This leads to a total of 20,011 samples that we use to train and evaluate our models. 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 256×256 cloth images for the whole T-shirt, for each patch we make a 160×160 crop from a higher resolution 512×512 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.

E. Networks

E.1. CNN Architecture Details

For predicting cloth images of the whole T-shirt, we start with the 90 dimensional input pose parameters and first apply 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 4×4 and stride 2), batch normalization, and ReLU activation until we reach the output resolution of 256×256. Each time the spatial resolution doubles and the number of channels halves. Finally, a convolution layer (filter size 3×3 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 convolution (filter size 4×4 and stride 2), batch normalization, and ReLU activation until we reach the target resolution of 160×160. Again, a final convolution layer (filter size 3×3 and stride 1) brings the number of channels to 3. The network contains 3.96 million parameters.

E.2. Quantitative Comparison of Loss Functions

Table 1 compares the 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 17.

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

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 [BODO18].

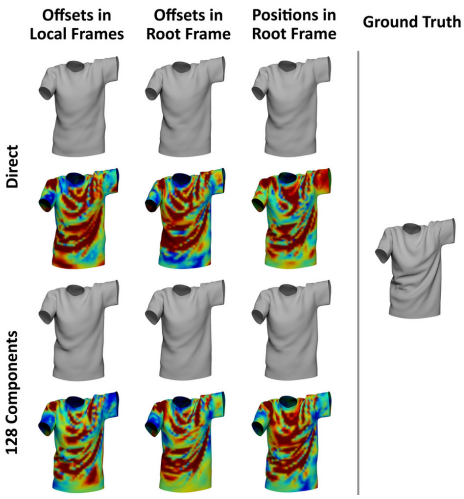


Figure 17: Comparison of fully connected network predictions and errors from models trained on different functions defined on our cloth pixels.

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 64×256 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:

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

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 64×256. Then, a final convolution layer (filter size 3×3 and stride 1) brings the number of channels to 3. The network contains 2.16 million parameters.

References

[AHLG*13] ALI-HAMADI D., LIU T., GILLES B., KAVAN L., FAURE F., PALOMBI O., CANI M.-P.: Anatomy transfer. *ACM Trans. Graph. 32*, 6 (Nov. 2013). 4

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 28–36. 4

[BODO18] BAILEY S. W., OTTE D., DILORENZO P., O'BRIEN J. F.: Fast and deep deformation approximations. *ACM Trans. Graph. 37*, 4 (July 2018), 119:1–119:12. 5

[CBE*15] CONG M., BAO M., E J. L., BHAT K. S., FEDKIW R.: Fully automatic generation of anatomical face simulation models. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2015), SCA '15, ACM, pp. 175–183. 4

[cmu] Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu/. 3

[WBH12] WHITMORE M., BOYER J., HOLUBEC K.: Nasa-std-3001, space flight human-system standard and the human integration design handbook. 3