Abstract
We propose a novel deep learning based method, called mesh super-resolution (MSR), to enrich low resolution (LR) cloth meshes with wrinkles. A pair of low and high resolution (HR) meshes are simulated, with the simulation of the HR mesh tracks that of the LR mesh. The frame data is converted into geometry images and used as training dataset. A residual network, called SRResNet, is employed to train an image synthesizer that super-resolves an LR image into an HR one. Once the HR image is converted back to an HR mesh, it is abundant in wrinkles compared to its coarse counterpart. The synthesizing is very efficient and is 24 times faster than a full high-resolution simulation. We demonstrate the performances of MSR with various simulation scenes.

Keywords: cloth animation, deep learning, data-driven, geometry image, wrinkle synthesis

1 Introduction
Clothing plays an important role in animation, as it contributes to the style and personality of virtual characters with realistic and complex detailed folding patterns. Although physical simulation systems can compute the deformation of cloth at a remarkable level of realism and detail, they incur an extremely high computational cost. In computer graphics, one common approach to accelerate computations of high-resolution models is to find a suitable low-resolution subspace that captures the large-scale behavior, and define a mapping from the low-resolution subspace to the high-resolution domain. It usually relies on precomputed data and data-driven techniques to build the mapping. This approach has been followed for cloth-wrinkle animation too, and various algorithms have been put forward, including subspace simulation methods [1] and pose space deformation methods [2] and others.

From another point of view, the detail enhancement for clothing is essentially a super-resolution operation, turning low-resolution meshes into high-resolution ones. Super-resolution (SR) is also a technique investigated by the computer vision community, for producing a high-resolution image from its low-resolution counterpart. By using deep convolutional neural networks (CNN), this ill-posed problem has achieved great breakthroughs in recent years. CNN is by far the most powerful machine learning tool and is particularly useful for data-driven applications involving images, speech, and natural language processing. Due to the irregular structure of meshes, it is not straightforward to use CNN to process meshes. If a mesh can be represented as a regular 2D array as an image, many successful image SR networks may be adapted to mesh SR. Fortunately, geometry image [3] is a technique that bridges the gap between images and meshes. Meshes can be converted into or from images, and a
CNN based image super-resolution can be applied. To enhance an LR mesh with details, it is first converted into an LR image, and then super-resolved into an HR image, and converted back into an HR mesh in the end.

A geometry image of a mesh captures geometry as a simple 2D array of quantized points, whose 3D coordinates are stored as RGB values. Due to the connectivity of the mesh, the corresponding image is rather smooth, i.e., lacking of high frequency information. This makes our dataset largely different from other photorealistic image datasets. As a result, some SR networks may not work well on our dataset. In this paper, we explore two networks, namely super-resolution residual networks (SRResNet) [4] and deeply-recursive convolutional network (DRCN) [5], for mesh super-resolution. Experiments show that SRResNet is more suitable for our goal. To the best of our knowledge, this is the first work that uses deep learning technique for cloth wrinkle synthesis.

2 Related Work

2.1 Data-driven Cloth Synthesis

Data-driven methods have drawn much attention as they offer faster cloth animation creation. These methods can be roughly classified into two groups. One line of work combines simulation on a coarse base mesh with pre-computed data for adding geometric details. Feng et al. [6] describe an approach which decomposes a high-res simulation into mid- and fine-scale deformations. For the mid-scale deformations, the mesh is decomposed into a set of bone clusters for which skinning weights are fit in a way similar to [7], while fine-scale details are added based on a principal component analysis of residual vectors. Both mid- and fine-scale details are then driven by a coarse scale simulation, which is fast enough to yield real-time rates. Focussing on fitted clothing, Wang et al. [8] present an example-based approach that augments coarse simulations with pose-dependent detail meshes obtained from a wrinkle database. The wrinkle database stores per-joint wrinkle meshes that are pre-computed from high-res simulations and merged together at run time. Targeting the more general case of free-flowing cloth, Kavan et al. [9] describe a method for learning linear up-sampling operators from high resolution simulations. With similar goals, Zurdo et al. [2] combine multi-resolution and pose space deformation (PSD) techniques in order to augment coarse simulations with example-based wrinkles. Hahn et al. [1] present a subspace simulation method using low-dimensional linear subspace with temporally adaptive bases. It exploits full-space simulation training data in order to construct a pool of low-dimensional bases distributed across pose space.

Another stream of work exploits pre-computed data to avoid runtime simulations altogether. De Aguiar et al. [10] present a technique for learning a linear conditional cloth model that can be trained with data from physics-based simulations. The method achieves very fast computation, but it primarily targets at low complexity cloth with little folding. The method of Guan et al. [11] factors clothing deformations into components due to body shapes and poses. A linear model is learned in order to quickly dress different body shapes and poses without run-time simulations. A different way of exploiting pre-computed data is suggested by Kim et al. [12], who create an exhaustive set of secondary motion to accompany a given primary motion graph. Kim and Vendrovsky [13] make use of pre-computed data to drive the deformation of clothing using the animated underlying model of the character wearing it.

2.2 CNN-based Image Super-resolution

Single image super-resolution, without any prior information, is a hard and ill-conditioning problem. With enough training data, CNN-based methods have achieved great progress recently. Wang et al. [14] encode a sparse representation prior into their feed-forward network architecture based on the learned iterative shrinkage and thresholding algorithm (LISTA). Dong et al. [15] uses bicubic interpolation of LR images as input and applies a simple 3-layers CNN to generate HR images. Kim et al. [5] propose to use DRCN to improve the performance of CNN by using a network as deep as 20 layers and using a recursive structure to decrease the number of parameters. To accelerate the computation and to
add more layers, many works use LR images, as opposed to bicubically interpolated HR images, as input. They upscale the feature maps into high resolution in the last few layers. For example, fast super-resolution convolutional neural network [16] has a transposed convolutional layer (also named deconvolutional layer), and efficient sub-pixel convolutional neural network [17] has a sub-pixel convolutional layer to solve the upscale problem. The evaluation of the quality of an algorithm is also an important problem. The optimization target is often the minimization of the mean squared error (MSE) between the recovered HR image and the ground truth. As minimizing MSE also maximizes the peak signal-to-noise ratio (PSNR), which is a common measure used to evaluate and compare SR algorithms. However, the ability of MSE and PSNR to capture perceptually relevant differences is very limited, as they are defined based on pixel-wise image differences [18]. As a result, very highest PSNR does not necessarily reflect perceptually good SR result. Ledig et al. [4] try to solve this problem with generative adversarial network.

2.3 CNN for 3D

Compared with 2D images, 3D shapes are more difficult to be processed by convolutional neural networks, mainly due to their irregular connectivity. Nevertheless, some effort was made in recent years. For 3D object recognition, Su et al. [19] represent 3D shapes using multi-view projections or converting them to panoramic views and utilize 2D CNNs. Li et al. [20] analyze a joint embedding space of 2D images and 3D shapes. For 3D shape synthesis, Wu et al. [21] use deep belief networks to generate voxelized 3D shapes. Girdhar et al. [22] combine an encoder for 2D images and a decoder for 3D models to reconstruct 3D shapes from 2D input. Yan et al. [23] generate 3D models from 2D images by adding a projection layer from 3D to 2D. Choy et al. [24] propose a novel recurrent network to map images of objects to 3D shapes. Wu et al. [25] exploit the power of the GAN with a voxel CNN. In addition to voxel representation, Sinha et al. [26] propose to combine SRRResNet and geometry images to synthesis 3D models. Li et al. [27] and Nash and Williams [28] propose to use neural networks for encoding and synthesizing 3D shapes based on pre-segmented data. Different parameterization methods emerged, such as authalic parametrization to a spherical domain [29] and global seamless parameterization to a planar flat-torus [30]. For animation creation, Chu [31] uses CNN to synthesize high-res smoke by encoding the similarity between LR and HR fluid patches. Our work tries to enhance low-res cloth meshes with high-res details using CNN.

3 The Method

3.1 Dual-resolution Cloth Meshes

A piece of cloth is initially defined by a closed curve in 2D space, and it is triangulated into a mesh for physical simulation. To generate training data, we triangulate a cloth patch into two meshes of different resolutions. They are simulated to create two sequences of frame data. Yet certain correspondence should be maintained between two meshes, so that in the simulation each pair of meshes have identical or similar large-scale folding behavior, but differ only in the fine-level wrinkles. To do so the initial high resolution mesh is created by subdividing the low resolution mesh, with all the vertices of the LR mesh retaining their positions in the HR counterpart. These vertices will be called feature vertices. This subdivision is achieved by the adaptive remeshing method in [32]. To create animation sequences, the LR mesh is simulated first. When simulating the HR mesh, the feature vertices are updated by assuming their positions from the LR mesh, and they are constraints in updating other vertices according to the dynamics. The TRACKS algorithm [33] gives more details on how to achieve this. Although an elegant averaging scheme is given in that paper, we find the interpolating scheme is simple and good enough for our application.

The mesh pairs from the animation sequences are converted into LR/HR image pairs, and are fed into the CNN for training. At the synthesizing stage an LR image, corresponding to an input LR mesh, is super-resolved into an HR image, which is then converted to an HR mesh. The pipeline of our approach, called mesh super-resolution (MSR), is illustrated in Fig. 1.
3.1.1 SRResNet

With physical simulation, we get a set of LR meshes \( \{ M_1^l, M_2^l, \ldots \} \) and a set of HR meshes \( \{ M_1^h, M_2^h, \ldots \} \). They are converted into LR images \( \{ I_1^l, I_2^l, \ldots \} \) and HR images \( \{ I_1^h, I_2^h, \ldots \} \), respectively. Each image, before being fed into the training network, is split into patches. An HR image is split into patches of \( 96 \times 96 \) each with stride 48. An LR image is split into patches of \( 24 \times 24 \) each with stride 12. Given the training data, our goal is to find a mapping function \( f(x) \) that minimizes the loss between the predict values \( I_s \) and ground truth \( I_h \). A common objective function is the MSE of the predicted and the ground-truth images:

\[
I_{\text{MSE}}^2 = \sum_{i} \sum_{j} (I_{i,j}^h - I_{i,j}^s)^2,
\]

where \( i, j \) are the indexes of pixels.

3.1.2 DRCN

Before choosing SRResNet, we tried DRCN [5], a once state-of-the-art SR algorithm for photorealistic images. After test-driving, we find the result less satisfactory, and a comparison between SRResNet and DRCN is given in the next...
DRCN uses bicubic interpolation to upscale LR images to the same size as the target, and feed them to the network as input. The highlight is the deep recursive layer in inference network and the prevention scheme of exploding/vanishing gradients. Each training image is split into $61 \times 61$ patches with stride 31, and 128 patches are used in each mini-batch. We use 16 recursions and the learning rate is initialized to be $10^{-4}$ and is halved if validation MSE does not decrease for 4 epochs. The training terminates when learning rate is less than $10^{-7}$. The experimental results however show that DRCN is not suitable for our application.

### 3.2 Mesh-image Conversion

Now we go back and discuss the conversion between meshes and images. It is realized by the technique of geometry image [3]. We let the resolution be $96 \times 64$ for LR images and $384 \times 256$ for HR images, as the upscaling factor is $4 \times$ for both SR algorithms. The factor for a pair of meshes are made to be as close as $4 \times$. For example, for the flag model the LR mesh has 58 vertices and the HR mesh has 930 vertices, the latter being roughly 16 times as many as the former.

**Mesh to image.** For the initial flat mesh, we sample $m \times n$ points uniformly in two-dimensional parameter space. Then for each sample point, we find the triangle where it is located, and compute its barycentric coordinates. These coordinates are unchanged even if the mesh deforms, thus the 3D coordinates of a sample point can be recovered from the three triangle vertices. The 3D coordinates of sample points, after being normalized to the range of $[0, 255]$, can be interpreted as RGB values of a $m \times n$ image $I$. We find that using integer values for RGB causes precision loss in reconstruction so floating point numbers are used instead as input for the convolutional neural network. A geometry image, unlike a photo-realistic image, is very smooth and does not convey much high frequency information (see Fig. 1).

**Image to mesh.** Converting a super-resolved HR image to a mesh can be done in two different manners. The common choice, is to respect the original HR mesh topology, and only restore vertex positions, by bilinearly interpolating four nearest sample points (pixels). Another choice, is to take advantage of property of geometry images and reconstruct a new connectivity. As the essence of geometry image is to sample surfaces on regular grid, it is straightforward to reconstruct the surface to quadrilateral meshes, of various resolutions. We could span each $2 \times 2$ quad of grid points (pixels) using two triangles, by splitting along any of the two diagonals. We could also lower the resolution of the reconstructed mesh, by spanning each $t \times t$ quad, with $t > 2$.

**Non-rectangle shapes.** When converting a mesh of non-rectangular shape, parameterization is needed to transform the mesh to a rectangle before sampling. There are multiple parameterization approaches to choose from. The as-rigid-as-possible (ARAP) method [34] is employed by us.

**Image boundary.** As each image is split into patches and each pixel is covered by multiple patches, it is desired that each pixel has equal significance in the training data. This requires each pixel to be covered by the same number of patches. Therefore, pixels on or close to the boundary need special attention. We expand the image by padding extra rows or columns beyond the four boundaries of the image. We find padding zeros is not a good choice, as it introduces noises to the image as well as the resulted mesh. We also tried to mirror the pixels inside the boundary outside, but it still does not work well. Directly replicating boundary pixels multiple times actually gives the best result.
4 Results

To evaluate the performance of our MSR method, we construct several datasets. We simulate a piece of cloth of rectangular shape and generate two datasets. The first one is from the simulation of fluttering flag on the wind. The second is a collection of frame data from simulation of multiple scenes. In addition to a mesh of rectangular shape, a mesh of irregular shape is also used to create the third dataset. Both LR and HR meshes are simulated with ARCSim [35], an open source simulator.

![Images of LR, Bicubic, SRResNet, and HR (ground truth) meshes]

Figure 4: Enhancing the flag with wrinkles.

The Flag Dataset. The simulation of the flag mesh generates 1000 frames of data. From the 1000 pairs of LR/HR meshes, we randomly select 800 pairs for training and leave the rest 200 pairs for testing. As we mentioned before, images are split into patches for training. In each mini-batch we randomly choose 16 patches. The learning rate is $10^{-4}$ and the procedure is terminated by 300 epoches. We compare our SR results to three other settings: 1) an LR mesh; 2) a mesh from bicubically interpolated LR image; 3) a simulation of HR mesh that tracks the LR simulation (tracked) (see Fig. 4). We provide quantitative evaluation in Table 1, including image PSNR and mesh VMSE (vertex-wise mean square error), which is computed as per vertex $L^2$ error averaged over all vertices and frames [2]. Although bicubic image interpolation has higher PSNR and lower VMSE values than our MSR method, but its visual appearance contradicts these values, due to sharp folds and lack of wrinkles. This is because geometry images do not have much high frequency information so bicubic interpolation can easily gain high score without learning the structural features of HR meshes. Data-driven MSR method performs better in learning features of HR meshes so proper wrinkles can be superimposed.

![Images of Scene1, Scene2, Scene3, and Scene4]

Figure 5: Four simulation scenes. The training data is from the first three scenes.

The Dataset of Multiple Scenes. To evaluate the generalization of our method we augment the dataset with simulation data of multiple scenes. There are four scenes: the cloth fluttering as a flag (Scene1), the cloth swinging with the right corner pinned (Scene2), the cloth interacting with a ball (Scene3), and the cloth swinging with the left corner pinned (Scene4) (see Fig. 5). Each scene simulation produces 1000 pairs of frames. For the training data, we randomly select 800 pairs from each of the first three scenes. The other 200 pairs of each scene, together with all pairs of Scene4 are used as testing data. We initialize the training with the trained model from the flag dataset and fine-tune it with a learning rate set to $10^{-5}$. In Fig. 6 and Fig. 7, we show two super-resolved meshes, one is from Scene3 and the other from Scene4.

The Dataset of Non-rectangular Mesh. The ultimate goal of our MSR algorithm is to be applied to garment meshes, which could be of any shape. An image, by nature, is a 2D array and is usually perceived as a rectangle. We would like to test the effectiveness of our method for meshes of an irregular shape. We chop off two corners of the flag mesh, making it a non-
Figure 6: A frame from Scene3, which is one of the three scenes in the training data.

rectangle. The ARAP method [34] is employed to parameterize the mesh onto a rectangle domain before sampling. Vertices on the mesh boundary are mapped to the boundary of the rectangle, with user designating four vertices to be the corners of the rectangle. Again we use 800 images for training and 200 images for testing. Parameterization distorts triangles so that the sampling is not uniform. Fig. 8 shows the results of SRResNet, along with LR and HR simulation results. Our MSR approach superimposes folds and wrinkles onto the LR mesh and creates a natural looking result.

**SRResNet vs DRCN.** We also have tried DRCN on the flag dataset and done a comparison against SRResNet. We find the super-resolved meshes from DRCN are not as good as that of SRResNet. Existing works that use DRCN for photo-realistic images usually evaluate their performances in terms of quantitative measures, such as MSE or PSNR. Although DRCN has good MSE and PSNR scores, it does not necessarily produce high quality images. In addition to using the bicubic interpolation as did in [5], we further try two other upscale filters. The first is dense sampling of an LR mesh, so that the sampled image has the same resolution as the HR image. The second is sampling of a much smoother mesh, resulted from the loop subdivision [36] of the LR mesh. This is to make the input to be very smooth so that there is enough space for accepting wrinkles. Yet we are a bit disappointed that in both cases the SR mesh is almost identical to the input, few details have been learned and superimposed. The reason, we believe, is that the upscaling for DRCN is outside the network. Once the upscaled input leads to a small initial value of loss function, the training terminates very quickly and no much wrinkle information is learned. The SRResNet, however, does the upscaling inside the network, and the wrinkle information is being employed.

**Running Time.** Increasing the resolutions of LR and HR flag meshes to 200 and 3,215 vertices, respectively, we create a new set of data through simulation. The resolutions of the LR and HR geometry images remain unchanged. We fine-tune the previously trained flag model with the new data to get a new synthesizer. The super-resolved results are shown in Fig. 9. The training and image synthesizing of all examples run on a NVIDIA Tesla P4 GPU with Pytorch 0.2.0. The running time for coarse simulation, tracked simulation and mesh-image conversion are collected on a 2.50GHz Core 4 Intel processor. The timing for two sets of flag examples is shown in Table 2. Using SRResNet, our algorithm is $24 \times$ faster than tracked simulation. In the testing, super-resolving an image using SRResNet takes 0.0039 sec/frame, which is $38 \times$ faster than using DRCN (0.145 sec/frame).
Table 1: PSNR and VMSE for 4× upscaling factor for all experiments

<table>
<thead>
<tr>
<th>dataset</th>
<th>scale factor</th>
<th>Bicubic</th>
<th>SRResNet</th>
<th>DRCN</th>
<th>#frames for test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR/VMSE</td>
<td>PSNR/VMSE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>flag</td>
<td>4×</td>
<td>58.34/0.00058</td>
<td>57.77/0.00068</td>
<td>58.07/0.00058</td>
<td>200</td>
</tr>
<tr>
<td>multi-scene</td>
<td>4×</td>
<td>58.31/0.00059</td>
<td>57.70/0.00068</td>
<td>-</td>
<td>200</td>
</tr>
<tr>
<td>Scene1</td>
<td>4×</td>
<td>60.72/0.00035</td>
<td>59.97/0.00038</td>
<td>-</td>
<td>200</td>
</tr>
<tr>
<td>Scene2</td>
<td>4×</td>
<td>59.59/0.00044</td>
<td>58.70/0.00061</td>
<td>-</td>
<td>200</td>
</tr>
<tr>
<td>Scene3</td>
<td>4×</td>
<td>58.38/0.00070</td>
<td>-</td>
<td>-</td>
<td>1000</td>
</tr>
<tr>
<td>non-rectangle</td>
<td>4×</td>
<td>41.62/0.01700</td>
<td>43.08/0.01700</td>
<td>-</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 2: Statistics and timing (sec/frm) of two sets of flag examples.

<table>
<thead>
<tr>
<th>dataset</th>
<th>#verts LR</th>
<th>#verts HR</th>
<th>tracked sim.</th>
<th>coarse sim.</th>
<th>mesh ↔ image conversion</th>
<th>synthesizing (GPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SRRResNet</td>
</tr>
<tr>
<td>flag</td>
<td>58</td>
<td>930</td>
<td>0.48</td>
<td>0.0152</td>
<td>0.00127</td>
<td>0.0039</td>
</tr>
<tr>
<td>flag2</td>
<td>200</td>
<td>3215</td>
<td>1.2</td>
<td>0.0616</td>
<td>0.00118</td>
<td>0.0040</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DRCN</td>
</tr>
</tbody>
</table>

Figure 8: Super-resolving a mesh of irregular shape.

Figure 9: Flags with increased mesh resolutions: 200 vertices for the LR mesh and 3,215 vertices for the HR and SR meshes.

5 Limitations and Future Work

So far our research has not reached state of the art simulation approaches like [37]. There are a number of issues to be explored in the future. First, for a garment model, which is an assembly of multiple pieces, it is tricky on how to parameterize and sample patches into geometry images that guarantee continuity at the seam line. Second, the scale factor of state-of-art image SR is only of 4×. Although this does not directly translate to the scale factor for meshes, networks that have higher scale factor are definitely welcome. Third, when building a training dataset, the full simulation at high resolution needs to track the low resolution simulation. This has a side effect that the large-scale behavior of the HR mesh is prescribed by the LR mesh. A new
tracking mechanism that given more degrees-of-freedom is to be discovered. Fourth, collisions could happen to super-resolved meshes, and should be properly solved. This is an important issue but has not been addressed in this article. Fifth, a geometry image is just a sampling of a polygonal surface, and certain information is missing during the sampling, so additional information including normal and curvature may be considered. Moreover, other approaches for transforming a polygonal mesh into a regular representation (e.g. Laplacian matrix) are worth trying.

6 Conclusion

We have presented a CNN based MSR method for generating high resolution cloth meshes. Experiments demonstrate the potential of our method. Our approach uses CNN to synthesize high-resolution triangle meshes with a large up-scaling factor (4×). Once trained, the speed of the synthesizing meets the demand of real-time animation for moderate mesh resolutions.

Acknowledgements

This work is supported by grants from China’s 863 Program (#2015AA016401), NSFC (#61502471 and #61502490) and Chinese Guangdong’s S&T project (2017B090912001). The authors would like to thank Yong Zhang and Zheng Lian for proof-reading.

References


