Journal of computer science and technology: Instruction for authors. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY

Multi-Feature Super-Resolution Network for Cloth Wrinkle Synthesis

Abstract Existing physical cloth simulators suffer f rom e xpensive c omputation a nd d ifficulties in tun ing mechanical parameters to get desired wrinkling behaviors. Data-driven methods provide an alternative solution. They typically synthesize cloth animation at a much lower computational cost, and also create wrinkling effects t hat h ighly r esemble the much controllable training data. In this paper we propose a deep learning based method for synthesizing cloth animation with high resolution meshes. To do this we first c reate a d ataset f or t raining: a p air o f l ow a nd h igh r esolution meshes are simulated and their motions are synchronized. As a result the two meshes exhibit similar large-scale deformation but different small wrinkles. Each simulated mesh pair are then converted into a pair of low and high resolution "images" (a 2D array of samples), with each image pixel being interpreted as any of three descriptors: the displacement, the normal and the velocity. With these image pairs, we design a multi-feature super-resolution (MFSR) network that jointly train an upsampling synthesizer for the three descriptors. The MFSR architecture consists of shared and task-specific layers to learn multi-level features when super-resolving three descriptors simultaneously. Frame-to-frame consistency is well maintained thanks to the proposed kinematics-based loss function. Our method achieves realistic results at high frame rates: $12 \sim 14$ times faster than traditional physical simulation. We demonstrate the performance of our method with various experimental scenes, including a dressed character with sophisticated collisions.

Keywords cloth animation, deep learning, wrinkle synthesis, multi-feature, super-resolution

1 Introduction

Cloth animation plays an important role in many applications, such as movies, video games, virtual try-on [1, 2]. With the rapid development of physics-based simulation techniques [3, 4, 5, 6], garment animations with remarkably realistic and detailed folding patterns can be achieved. However, these techniques require high resolution meshes to represent fine details, therefore need much computation to solve velocity-updating equations and resolve collisions. Moreover it is labor-intensive to tune simulation parameters for a desired wrinkling behavior. Recently data-driven methods [7, 8, 9] provide alternative solutions for these problems, as they offer fast production and also create wrinkling effects that highly resemble the training data. Relying on precomputed data and data-driven techniques, a high resolution (HR) mesh is either directly synthesized, or super-resolved from a physically simulated low resolution (LR) mesh. Nevertheless, existing data-driven methods either depend on human body poses [7, 10, 11, 9] thus are not suitable for loose garments, or lack of dynamic modeling of wrinkle behaviors [12, 8, 13, 14, 15] for general case of free-flowing cloth.

To tackle these challenges, we propose a framework, synthesizing cloth wrinkles with a deep learning based method. We create datasets, from physics-based simulation, as the training data. The simulation is assumed to be independent of human bodies and not limited to tight garments. This dataset is generated by a pair of LR and HR meshes with synchronized simulations. Given the simulated mesh pairs, we aim to map the LR meshes to the HR domain by a detail enhancement method, which is essentially a super-resolution (SR) operation. Deep SR networks have proven to be powerful and fast machine learning tools for image detail enhancement [16, 17]. Yet for surface meshes which usually have irregular structures, it is not straightforward to apply traditional convolutional operations as for images. Chen *et al.*[13] proposed a method, converting manifold meshes into geometry images [18], to solve

Regular Paper

this issue. Inspired by their work, we design a multi-feature super-resolution network (MFSR) to improve the synthesized results and model the dynamic wrinkle behaviors. The LR and HR image pairs, encoding three features: the displacement, the normal and the velocity, are fed into the network for training. Our MFSR jointly learns upsampling synthesizers with a multi-task architecture, consisting of a shared network and three task-specific networks, instead of combining all features with a single SR network. The proposed spatial and temporal losses also contribute to the generation of dynamic wrinkles and further maintain frame-to-frame consistency. At runtime, with super-resolved geometry images generated by MFSR, we convert them back into HR meshes. As our approach is based on deep neural networks, it reduces the computational cost significantly. In summary, the main contributions of our work are as follows:

- We propose a novel framework for cloth wrinkle synthesis, which is composed of synchronized simulation, mesh-image conversion, and a multi-feature super-resolution network (MFSR).
- We learn both shared and task-specific representations of garment shapes via multiple features.
- We generate dynamic wrinkles and consistent mesh sequences thanks to the spatial and temporal loss functions.

We qualitatively and quantitatively evaluate our method for various cloth types (tablecloths and long skirts) and motion sequences. Experimental results show that the quality of synthesized garments is comparable with that from a physics-based simulation, yet significantly reducing the computation cost. To the best of our knowledge, this is the first approach to employ a multi-feature learning model on 3D dynamic wrinkle synthesis.

2 Related work

2.1 Cloth animation

A physics-based simulation for realistic fabrics includes velocity updating by physical energies [3, 6], time integration [5], collision detection and collision response [4]. These modules are solved separately and time consuming. To improve the efficiency of this system, researchers have exploited many algorithms such as implicit time integration [5], adaptive remeshing [19] and iterative optimization [20]. Nevertheless, these algorithms still cost the expensive computation to produce rich wrinkles and are labor consuming to tune mechanical parameters for desired wrinkling behaviors. Recently data-driven methods have drawn much attention as they offer faster cloth animations than the physics-based methods. Based on precomputed data and data-driven techniques, an HR mesh is either directly synthesized, or super-resolved from a physically simulated LR mesh. In the first stream of work, with precomputed data, researchers have investigated many techniques to accelerate the process for new animations, such as a linear conditional model [11, 21] and a secondary motion graph [22]. Additionally, deep learning-based methods [23, 24, 25] are also used to generate garments on human bodies. In the another line of work, researchers have proposed to combine coarse mesh simulations with learned geometric details from paired mesh databases, to generalize the performance to complicated testing scenes. This stream of methods includes wrinkle synthesis depending on bone clusters [10] or human poses [7] for fitted clothes, and linear upsampling operators [12] or low-dimensional subspace with bases [8, 26] for general case of free-flowing cloth. Inspired by these data-driven methods, we propose a deep learning based approach to synthesize wrinkles on coarse simulated meshes, while our approach is independent with poses or skeletons and not limited with tight garments.

2.2 Representation in 3D learning

To process 3D models for deep learning, there are various representations [27], *e.g.* , voxels, images, point clouds, meshes. Wang *et al.*[28] use voxel grids with octree-based convolutional neural networks (CNNs) for 3D shape analysis. Su *et al.*[29] learn to recognize 3D shapes from multi-view images with 2D-CNNs. Representations based on voxels or multi-view images are extrinsic to the shapes, which are sensitive to isometric deformations, like rotation or translation. Instead of rendered images, recent works [30, 13] use a technique called geometry images [18] encoding features of 3D meshes into 2D domain for 3D object recognition and generation . With a patch-based approach, this technique is easily coped with deep CNNs thus suitable for our mesh super-resolution task. Geometry images require parameterization for non-rectangular meshes, we use a padding scheme to avoid mesh distortion. Recently, some researches [31, 32] directly encode triangle meshes with deformation-based features [33, 34] into latent space with applications to shape embedding and synthesis. These methods focus on the deformation of overall meshes, however, our patch-based algorithm aims at learning local details and is independent of the underlying mesh connectivity.

Feature-based methods aim for proper descriptions of irregular 3D meshes, for synthesizing detailed and realistic objects. Conventional data-driven methods [8] simplify the calculation of wrinkle features, by formulating the strain or stress in an LR mesh. As for deep learning, several algorithms have also investigated robust descriptors for wrinkle deformation. Chen *et al.*[13] and Oh *et al.*[14] use 3D coordinates to augment coarse meshes with synthesized wrinkles. Wang *et al.*[25] learn an autoencoder network for cloth using 3D positions. Instead, Santesteban *et al.*[9] decompose the cloth deformation into two displacements, a global fit displacement and the wrinkle displacements. In addition to the position or the displacement, Lähner *et al.*[15] and Zhang *et al.*[35] learn high frequency details from normal maps. In our approach, we cascade multiple geometric features as shape descriptors embedded in geometry images, including spatial information of the displacement, the normal and temporal information of the velocity.

2.3 Deep CNN-based Super-resolution

In the area of single image super-resolution (SISR), deep learning-based techniques have achieved significant breakthroughs in recent years. Convolutional layers [36] are proposed to be more efficient instead of fully-connected structures in SISR, and later on are extended to deep networks using various upsampling layers, *e.g.* transposed layers [37] and sub-pixel layers [38]. For deep networks, residual learning [16] or dense connections [39] are employed to solve the vanishing gradient problem. Our method likewise uses a deep network with residual dense architecture [17] for its performance and efficiency.

In video super-resolution tasks, how to generate temporal consist results is a vital problem. One way is using consecutive frames as inputs [40] or recurrently using previously predicted outputs [41]. More recently, the recurrent mechanism has influenced the field of cloth animations. *E.g.* , Santesteban *et al.*[9] use recurrent networks based on gated recurrent units to regress garment wrinkles. Recurrent modules need to predict results sequentially, while our technique processes images individually and parallelly, even in arbitrary order. An alternative solution is to synthesize single output with specialized loss terms to constrain the consistency over time. Loss functions using nearby frames help to

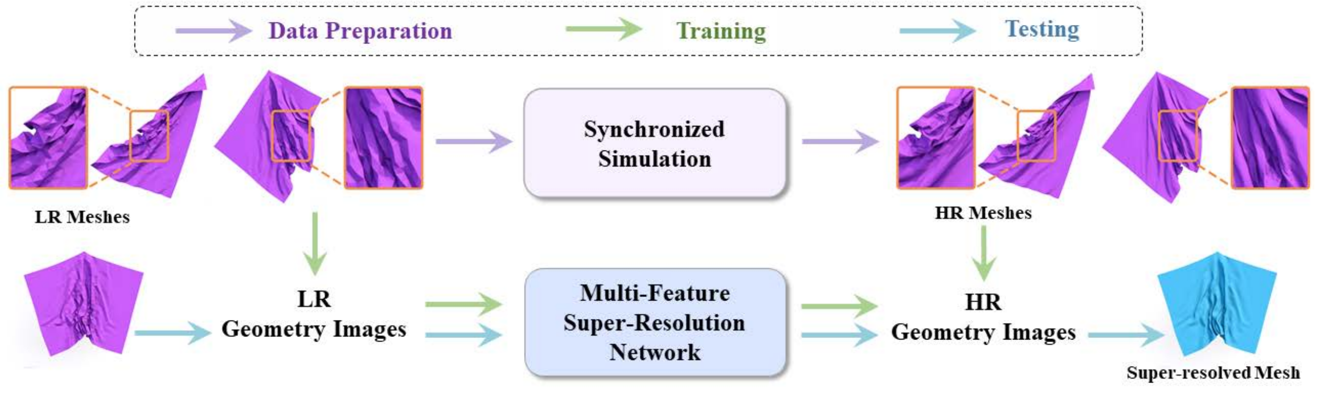

Fig.1. Pipeline of our Multi-Feature Super-Resolution (MFSR) network for cloth wrinkle synthesis. We generate low-resolution (LR) and high-resolution (HR) mesh sequences via synchronized simulation. In the training stage, LR and HR meshes are converted into LR and HR geometry images, respectively, encoding multiple features: the displacement, the normal and the velocity of the sampled points. Then these features are fed into our MFSR network for training. At runtime stage, LR geometry images (converted from the input LR mesh) are super-resolved into HR geometry images, which are converted to a detailed mesh.

alleviate temporal discontinuities in video [42]. In fluid generation, Xie *et al.*[43] utilize a discriminator loss to preserve temporal coherence. For cloth wrinkle synthesis, Lähner *et al.*[15] propose a $L1$ loss between the generated normal map and the ground truth at the previous frame. In our work, the cloth meshes are created by physics-based simulation, thus ground truth motion is available. A kinematic-based loss constraining the estimated velocity and position enables our network to generate realistic wrinkles while keeping the predictions coherent from frame to frame.

3 Overview

Our method takes physical simulated LR meshes as input, to infer realistic and consistent HR cloth animations. The pipeline of our approach is illustrated in Fig. 1. To generate training data, a pair of LR and HR meshes are simulated synchronously by virtual spring constraints and multi-resolution dynamic models (§ 4.2). Thus, the LR and HR meshes are well aligned at the level of large-scale deformation and differ in the wrinkles. Then the simulated mesh pairs are converted into dual-resolution geometry images (§ 4.1), with each sample encoding three features: the displacement, the normal and the velocity. A multi-feature super-resolution network (MFSR) with shared layers and task-specific modules is proposed to super-resolve LR images with details (§ 5). Based on these features, we design the spatial and temporal loss functions (§ 5.2) to train our MFSR for detailed and consistent results. At runtime, the testing LR geometry images (converted from the input LR mesh) are upsampled into HR geometry images, which are then converted to a detailed HR mesh with a refinement step to solve collisions.

4 Data preparation

4.1 Data representation and conversion

Dual-resolution meshes. Before executing cloth simulation for data preparation, we need to set the initial rest state of LR and HR meshes. We obtain the HR mesh by subdividing the edges of the LR one progressively till the desired resolution. In this work, the number of faces in the HR mesh is 16 times as many as the LR mesh. With the rest state LR/HR meshes, we create two sets of dual-resolution frame data via physics-based simulation. The correspondence between them is maintained during the simulation, so that they exhibit similar large-scale folding behaviors but differ in the fine-level wrinkles. More details about the synchronized simulation are given in § 4.2.

Dual-resolution geometry images. We convert the paired meshes to dual-resolution geometry images of 9 channels. The embedded descriptors in our images include the *displacement* $\mathbf{d}$, the *normal* $\mathbf{n}$ and the *velocity* $\mathbf{v}$. Different from the original geometry image paper [18], we encode the displacements instead of positions, since we are only interested in the intrinsic shape of the mesh, not its absolute spatial locations. The displacement is defined as the difference between its position in current frame and that in its starting position. The vertex normal is computed by the area-weighted average normals of the faces adjacent to this vertex. Due to the physics-based simulation with fixed time step, the velocity is naturally calculated using the positions between two frames (The complete calculation of our feature descriptors is provided in the supplemental materials). Since these features are not rotation invariant, we calculate a rigid motion transformation [44] with rotation $\mathbf{R}$ and translation $\mathbf{t}$. Then, we apply $(\mathbf{R}, \mathbf{t})$ to displacement, $\mathbf{R}$ applied to normal and velocity. To reduce the computation cost, we only compute the rigid motion of LR meshes and apply the same $(\mathbf{R}, \mathbf{t})$ to the HR meshes. To release the internal covariate shift [45], these features are normalized into a range of [0, 1].

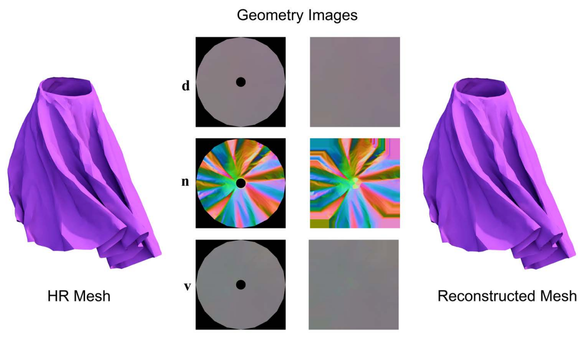

Fig.2. Mesh-image conversion. The HR skirt is converted to geometry images with three descriptors: the displacement $\mathbf{d}$, the normal $\mathbf{n}$ and the velocity $\mathbf{v}$. For irregular garments, the feature values of sample points outside the mesh but inside the bounding box are zero (black pixels in the 2nd column), then are padded with the nearest non-zero values (the 3rd column). The right one is the reconstructed mesh via geometry image of displacement.

Mesh-to-image conversion. For a mesh in its rest state, we find its bounding box in the 2D material space. Inside the bounding box, we then sample an array of $m \times n$ points uniformly. For each sample point inside the mesh, we find the triangle it is located in, and compute its barycentric coordinate (BC) w.r.t. three triangle vertices. BC is unchanged even though a triangle deforms during simulation. When computing features for sample points, BCs are used as weights for interpolating feature values $(\mathbf{d}, \mathbf{n}, \mathbf{v})$ from triangle vertices. For a mesh whose boundary coincides with the bounding box edge, we do the padding operation along boundaries. Otherwise, for sample points outside the mesh but inside the bounding box, their feature values are filled with the nearest non-zero pixels similar to replicate padding. A long skirt example is given in Fig. 2.

Image-to-mesh conversion. After an HR image is synthesized, values in the displacement channels are used to reconstruct the positions of the detailed mesh, while the original topology of that mesh is retained. Due to the padding operation, every vertex in 2D material space has four nearest non-zero sample points. We reconstruct the displacements of vertices by

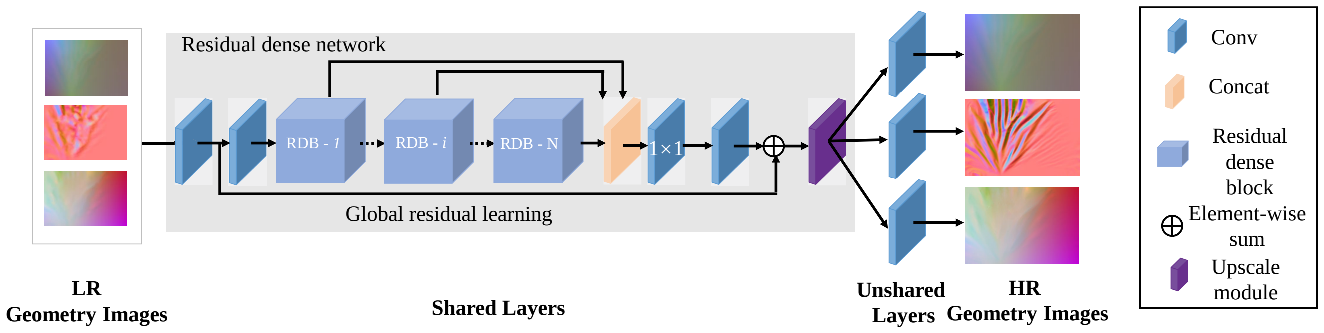

Fig. 4. The architecture of MFSR. The input and the output are LR/HR images where each pixel is represented as a 9-dimensional feature vector enclosing the displacement, the normal and the velocity in order. Conv and Concat refer to convolutional and concatenation layers, respectively. The MFSR upscales the LR features with shared and unshared layers to recover HR features with detailed information.

bilinear interpolation. These computed displacements are added to the positions of subdivided mesh vertices in the rest state to obtain wrinkle-enhanced positions. In the end we apply the inverse of the rigid transformation, computed in the mesh-to-image phase, to new positions. As shown in the right of Fig. 2, almost no visual differences can be seen. In our quantitative experiments, the geometric reconstruction error is smaller than 1e-4 meter, measured by the vertex mean square error (VMSE).

4.2 Synchronized simulation

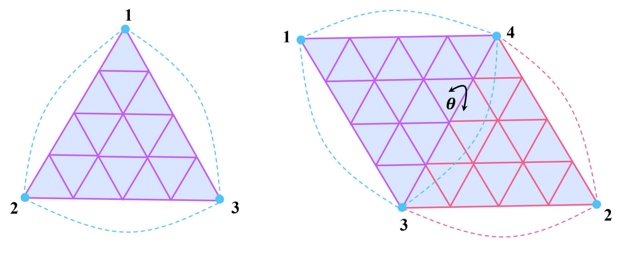

Fig. 3. The multi-resolution dynamic model for tracking. Forces of stretching (left) and bending (right).

The high-quality training dataset is equally important for data-driven approaches. In our case, we need to generate corresponding LR/HR mesh pairs in animation sequences by physics-based simulation. In image super-resolution tasks [16, 36], one way to generate training dataset is down-sampling HR images to obtain their corresponding LR ones. However, down-sampling an HR cloth mesh could cause collisions, even though the HR mesh is collision-free. Therefore, it is preferred that two meshes are simulated individually, with all collisions being properly resolved. However, as mentioned in previous works [12, 8], if there is no constraints between two simulations, they will bifurcate to different b ehaviors b ecause o f a ccumulated h igher frequencies generated by finer meshes and numerical errors. There are several ways to formulate synchronized constraints such as testing functions [46, 12]. And our implementation enforces virtual spring constraints and uses multi-resolution dynamic models to construct synchronized simulation for HR meshes.

Our dual-resolution meshes are well aligned in the initial state, because we only add vertices on the edges without changing the mesh shape. The vertices in an LR mesh, called *feature vertices*, show up in an HR mesh and are used as constraints for synchronized simulation. We first run coarse cloth simulation and record the positions of all feature vertices at total N frames as $\mathbf{p}_k^l$, $k = 1, \cdots, N$, where the superscript l stands for the LR. While simulating an HR mesh at the frame k, virtual springs are added to connect pairs $(\mathbf{p}_k^l, \mathbf{p}_{k-1}^h)$ of feature vertices between LR mesh at the frame k and HR mesh at the frame $k-1$. To pull $\mathbf{p}_{k-1}^h$ towards $\mathbf{p}_k^l$, we define an internal force following Hooke's law as

$$\mathbf{f}_{spring} = -c(\mathbf{p}_k^l - \mathbf{p}_{k-1}^h). \quad (1)$$

where c is a spring stiffness constant that can be adjusted depending on how tight the tracking is desired by the user. A large c results in tight tracking of the feature vertices, but not for other vertices. As a side effect the simulated HR mesh has many annoying "spikes".

Thus, we add a multi-resolution dynamic model to cooperate with virtual springs. Given an HR mesh at level H_0 (shown as the solid lines in Fig. 3), we construct an LR triangle mesh at level H_1 (the dashed triangle in Fig. 3). The mesh in H_1 connects the feature vertices by retaining the topology of the LR mesh. In finite-element simulations, the constitutive model includes internal cloth forces supporting behaviors such as anisotropic stretch or compression [47] and surface bends [6] with damping [19]. For a triangle in the coarse mesh at level H_1, the in-plane stretching forces $\mathbf{f}_1^s = (\mathbf{f}_{11}^s, \mathbf{f}_{12}^s, \mathbf{f}_{13}^s)$ at three vertices are measured by a corotational finite-element approach [47]. While the bending forces for two adjacent triangles are added using a discrete hinge model based on dihedral angles, denoted as $\mathbf{f}_1^b = (\mathbf{f}_{11}^b, \mathbf{f}_{12}^b, \mathbf{f}_{13}^b, \mathbf{f}_{14}^b)$. The triangles in the fine level H_0 have the same force patterns $\mathbf{f}_0^s$ and $\mathbf{f}_0^b$ imposed on all particles (including feature vertices). All stretching and bending forces are added accompanying damping forces. In addition, our two-level dynamic models are independent of the force implementations, and would also work with other triangular finite-element methods. As a result, the feature vertices in multi-resolution dynamic models receive the stretch forces from both $\mathbf{f}_0^s$ and $\mathbf{f}_1^s$, while the same for bending forces. The rest vertices are only imposed on the forces at level H_1. With the two-hierarchy dynamics model, modest virtual spring coefficients can make the HR mesh keep pace with the LR mesh in simulation.

5 Multi-feature super-resolution network

In this section, we introduce our MFSR architecture based on the RDN, as well as the loss functions taking spatial and temporal features into account to improve wrinkle synthesis capability.

5.1 MFSR architecture

We now introduce our MFSR architecture for the image SR tasks of multiple features. With LR/HR images of the form $(\mathbf{d}, \mathbf{n}, \mathbf{v})^l$ and $(\mathbf{d}, \mathbf{n}, \mathbf{v})^h$, our MFSR learns the mappings of different features by image SR networks. One standard methodology is single task learning, which means learning one task at a time. However it ignores a potentially rich source of information available in other tasks. Another option is multi-task learning, which achieves inductive transfer between tasks, with the goal to leverage additional sources to improve the performance of the target task [48]. Our MFSR is a multi-task architecture, consists of two components: a single shared network, and three task-specific networks. The shared network is designed based on the SR task, whilst each task-specific network consists of a set of convolutional modules, which link with the shared network. Therefore, the features in the shared network, and the task-specific networks, can be learned jointly to maximise the generalisation of the shared representation across multiple SR tasks, simultaneously maximising the task-specific performance.

Fig. 4 shows a detailed visualisation of our MFSR based on residual dense blocks (RDB) [17]. In the shared network, the image SR model consists of four parts: *shallow feature extraction*, *basic blocks*, *dense feature fusion*, and finally *upsampling*. We use two convolutional layers to extract shallow features, followed by the RDB [17] as the basic blocks, then dense feature fusion to extract hierarchical features, and lastly one bilinear upsampling layer to upscale the height and width of the LR feature maps by 4 times. Different from general SR tasks, we find that pixel shuffle and deconvolution methods cause apparent checkboard artifacts so we use bilinear method. For basic blocks in our SR network, we employ RDB instead of residual blocks used in [13]. As shown in the left of Fig. 5, a residual block learns a mapping function with reference to its input, therefore can be used to build deep networks to address the problem of vanishing gradients. However, in the residual block a convolutional layer only has direct connection to its precedent layer, neglecting to make full use of all preceding layers. To exploit all the hierarchical features, we choose RDB (see in the right of Fig. 5) that consist of densely connected layers for global feature combination and local feature fusion with local residual learning. More details about RDB are given in [17]. In each task-specific network, we utilize one convolutional layer to map the extracted local and global features to each upsampled descriptor $\mathbf{d}^s, \mathbf{n}^s$, and $\mathbf{v}^s$, respectively.

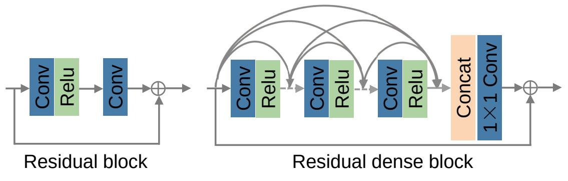

Fig. 5. Two network structures used in image super-resolution. The left is residual block in [16]. The right is residual dense block in [17] used for our MFSR.

5.2 Spatial and temporal losses

In order to learn the spatial details and temporal consistency of the underlying HR meshes, our MFSR is trained by minimizing the following loss functions for mesh features. A baseline mean square error (MSE) reconstruction loss is defined as

$$\mathcal{L}_d = ||\mathbf{d}^h - \mathbf{d}^s||^2, (2)$$

where $\mathbf{d}^h$ and $\mathbf{d}^s$ stand for the ground truth HR and the synthesized SR displacement images, respectively. This displacement loss term is able to obtain a smooth HR result with given low frequency information.

To extend the loss into wrinkle feature space, a novel L_2 loss for normal is introduced:

$$\mathcal{L}_n = ||\mathbf{n}^h - \mathbf{n}^s||^2. \quad (3)$$

where $\mathbf{n}^h$ denotes the ground truth normal image and $\mathbf{n}^s$ denotes the superresolved normal image. The normal feature is directly related to the bending behavior of cloth meshes. This loss term encourages our model to learn the fine-level wrinkle features so that the outputs can stay as close to the ground truth as possible. In our experiments it aids the networks in creating realistic details.

The above two loss terms are utilized to reconstruct high-frequency details exclusively from spatial statistics. To improve the consistency for animation sequences, we should also take the temporal coherence into account. The vertex velocities of every animation frame contribute a velocity loss of the form

$$\mathcal{L}_v = ||\mathbf{v}^h - \mathbf{v}^s||^2, \quad (4)$$

where $\mathbf{v}^h$ denotes the ground truth velocity image and $\mathbf{v}^s$ denotes the synthesized velocity image.

In addition, we minimize a kinematics-based loss in the training stage, to constrain the relationship between synthesized velocities and displacements (please refer to the supplementary material for the detail derivation) as

$$\mathcal{L}_{kine} = \sum_{k=1}^{n} ||\mathbf{R}^{-1}(\mathbf{d}_k^s - (\mathbf{d}^s + (\sum_{j=1}^{k} \mathbf{v}_{k-j}^s) * \Delta t)||^2, \quad (5)$$

where n is the length of frames associated to the input frame, and Δt represents the time step between consecutive frames. $\mathbf{R}$ is the precomputed rotation part in the rigid motion transformation for input frame. This kinematics-inspired loss term can improve the consistency between the generated cloth animations.

The overall loss of our MFSR is defined as

$$\mathcal{L}_{all} = w_d\mathcal{L}_d + w_n\mathcal{L}_n + w_v\mathcal{L}_v + w_{kine}\mathcal{L}_{kine}. \quad (6)$$

which is a linear combination of spatial smoothness, detail similarity, temporal consistency and kinematic loss terms with the weight factors w_d, w_n, w_v, and w_{kine}. As for back propagation, each loss term propagates backwards through the task-specific layer independently. In the shared layers, parameters are updated according to the total loss $\mathcal{L}_{all}$. As a result, the gradient of loss functions from multiple SR tasks will pass through the shared layers directly, and learn a common representation for all related tasks.

The reconstructed meshes converted from super-resolved images by the above network may suffer from penetrations with obstacles or self-collision. For interactions with human or balls, we adopt a fast refinement method [25] to push the cloth vertices colliding with the obstacles outside meanwhile preserving the local wrinkle details. As for self-collision, since the run-time simulation of an LR mesh is collision-free, we interpolate the vertices in the penetrating area between LR meshes and the reconstructed ones to guarantee to resolve all collisions. In implementation, we use the bisection method [49] to search for a close-to-optimal interpolation weight. We do the bisection several times and then take the last collision-free state as the interpolation result (see Fig. 6). It is not necessary to let all vertices of the whole mesh get involved in the position interpolation. Instead, only the vertices involved in the intersections are of our interests. These vertices can be specified by a discrete collision detection process and grouped into impact zones as done in [50]. Position interpolations are performed per zone, and each zone has different interpolation weights. In this way, the synthesized meshes are least affected by the collision handling.

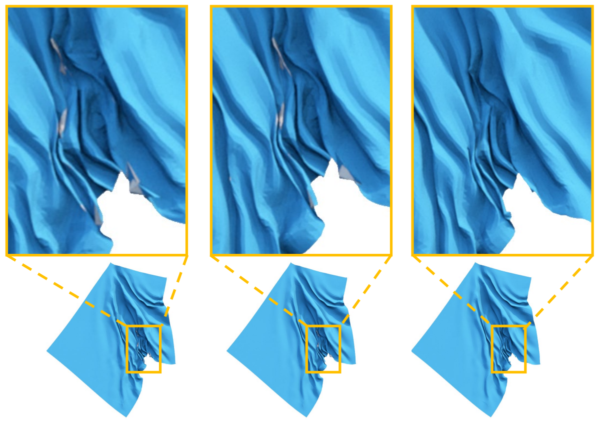
Fig.6. Given a super-resolved cloth with self-collision (the 1st column), the collision solving method is utilized to untangle the intersection regions after two steps (the 2nd and the 3rd columns).

6 Implementation

We describe the details of the data generation and the network architecture in this section.

Data generation. We construct three datasets using a tablecloth model and a skirt model with character motions. The two models are regular and irregular garment shapes, respectively. The meshes in each dataset are simulated from a fixed template model. For the tablecloths, we generate two datasets, called DRAPING and HITTING (see Fig. 7). The DRAPING dataset is created by randomly handling one of the topmost vertices of the tablecloth and letting the fabric fall freely. It contains 13 simulation sequences, each with 400 frames. 10 sequences are randomly selected for training and remaining 3 sequences are for testing. In addition to simulating a piece of tablecloth in a free environment, we also construct a HITTING dataset where a sphere interacts with the tablecloth. Specifically, we select spheres of different sizes to hit the tablecloth back and forth at different locations, and obtain a total of 35 simulation sequences, with 1,000 frames for each sequence. We randomly select 27 sequences for training and 8 sequences for testing. The

SKIRT dataset is created by the long skirt garments worn by an animated character (shown in Fig. 7). A mannequins has rigid parts as [19] and is driven by publicly available motion capture data from CMU [51]. We select dancing motions including 7 sequences (in total 30,000 frames), in which 5 sequences are randomly selected for training and 2 sequences are for testing. To simulate cloth stably, we interpolate 8 times between two adjacent motions from the original data.

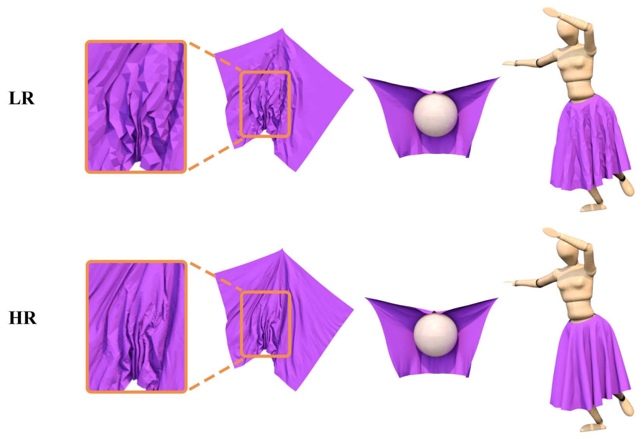

Fig. 7. We test our algorithm on three datasets including DRAPING (left), HITTING (middle) and SKIRT (right). The top and bottom rows show the LR and HR cloth meshes, respectively.

We apply the ARCSim engine [19] to produce all simulations with remeshing disabled. A material called the Gray Interlock is adopted for its anisotropic behaviors, from a library of measured cloth materials [52]. To meet a collision-free initial state for skirts, we first manually put the skirt on a template mannequin (T pose), then interpolate 80 frames between the T pose and the initial poses of motion sequences. In addition, for synchronized simulation, we set the spring stiffness constant $c = 10$ in the equation (1).

Network architecture. For different datasets, we train each model separately. Our proposed MFSR consists of shared and task-specific layers. The shared network has 16 identical RDB [17], where six of them are densely connected layers for each RDB, and the growth rate is set to 32. The basic network settings, such as the convolutional kernel and activation function, are set according to [17]. For the upscaling operation, *i.e.*, from the coarse resolution features to fine ones, we consider several different mechanisms, *e.g.* , pixel shuffle module [38], deconvolution, nearest and bilinear, and finally choose the bilinear upscaling layer because it can prevent checkerboard artifacts in the generated meshes. In our upsampling network, the upscale factor is set to 4. The upscale factor (in one dimension) for corresponding meshes is set to be as close to 4 as possible. For example, the LR and the HR tablecloth meshes have 749 and 11,393 vertices, respectively, the latter being roughly 16 times as many as the former. Converting meshes to images, we set the size of LR images in tablecloth to be 192×128, and HR ones 768×512. The image aspect ratio is the same to the uv proportion in material space to achieve uniform sampling.

We implement our network using PyTorch 1.0.0. In each training batch, we randomly extract 16 LR/HR pairs with the size of 72×72 and 288×288 as input. Adam optimizer [53] is used to train our network, and its β_1 and β_2 are both set to 0.9. The base learning rate is initialized to 1e-4, and is divided by 10 every 20 epochs. To avoid learning rate becoming too small, we fix it after 60 epochs. The training procedure stops after 120 epochs and takes about a day and a half. In all our experiments, we set the length of the input frames $n = 3$ for the kinematics-based loss in the equation (6). Besides, we set the weights $w_d = 0.9, w_n = 0.03, w_v = 0.03$ and $w_{kine} = 0.03$ in the equation (6).

7 Results and evaluations

In this section, we evaluate the results obtained with our method both quantitatively and qualitatively. The runtime performance and visual fidelity are demon-

Table 1. Statistics and timing (sec/frm) of the tablecloth and skirt testing examples.

Benchmark	#verts LR	#verts HR	tracked sim.	ours	speedup	our components			
						coarse sim.	mesh/image conversion	synthesizing (GPU)	refinement
DRAPING	749	11,393	4.27	0.345	**12**	0.129	0.089	0.0553	0.0718
HITTING	749	11,393	4.38	0.341	**13**	0.135	0.109	0.0531	0.0434
SKIRT	1,303	19,798	10.23	0.709	**14**	0.227	0.18	0.0281	0.274

strated with various scenes: draping and hitting tablecloths, and long skirts worn by animated character, separately. We compare our results against simulation methods and demonstrate the benefits of our method for cloth wrinkle synthesis. The effectiveness of our network components is also analyzed, for various loss functions and network architectures.

7.1 Runtime performance

We implement our method on a 2.50GHz Core 4 Intel CPU for coarse simulation and mesh-image conversion, and a NVIDIA GeForce® GTX 1080Ti GPU for image synthesizing. Table 1 shows average per-frame execution time of our method for the different garment resolutions. The execution time contains four parts: coarse simulation, mesh/image conversion, image synthesizing, and refinement. For reference, we also statistic the simulation timings of a CPU-based implementation of tracked high-resolution simulation using ARCSim [19]. Our algorithm is averagely 13 times faster than the tracked simulation. The low computational cost of our method makes it suitable for the interactive applications.

7.2 Wrinkle synthesis results and comparisons

Generalization to new hanging. We use the training data in the DRAPING dataset to learn a synthesizer, then evaluate the generalization to new hanging vertices. Fig. 8 shows the deformations of tablecloths of three test sequences in the DRAPING dataset. We compare our results with the HR meshes of tracked physics-based simulation. Our approach successfully produces the realistic and abundant wrinkles in different deformation sequences, in details, tablecloths appear many middle and small wrinkles when falling from different directions.

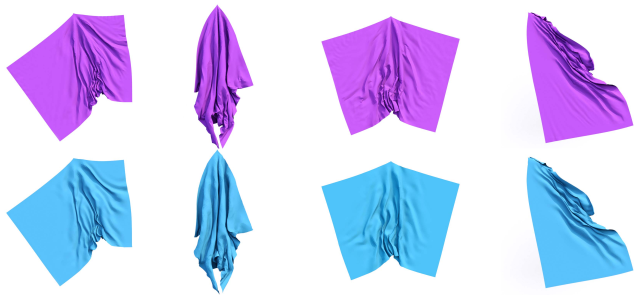

Fig.8. The super-resolved results of our method (bottom) is able to produce as many wrinkles as the ground-truth tracked HR simulation (top).

Generalization to new balls. Shown in Fig. 9, we visually evaluate the quality of our algorithm in the HITTING dataset, which illustrates the performance when generalizing to new crashing balls of various sizes and initial positions. We show four test examples comparing the ground-truth HR of the tracked simulation with our method. For testing, the initial positions of balls are set to four different places which are unseen in training data. Additionally, in the third and fourth columns of Fig. 9, the diameter of the ball is set to 0.5 which is also a new size not used for training. When various sizes of balls crash into the cloth in different positions, our method can successfully predict the plausible wrinkles, with 12 times faster running speed than physics-based simulation.

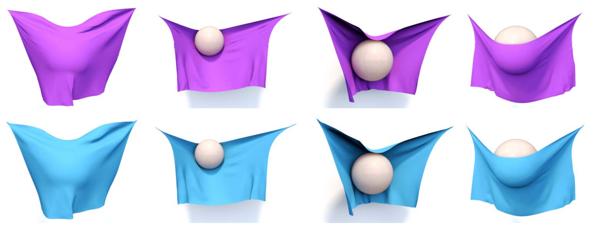

Fig.9. Comparison between the ground-truth tracked simulation (top) and our super-resolved meshes (bottom), on testing animation sequences in HITTING dataset. Our method succeeds to predict the small and mid-scale wrinkles of the garments with 12 times faster running speed than physic-based ones.

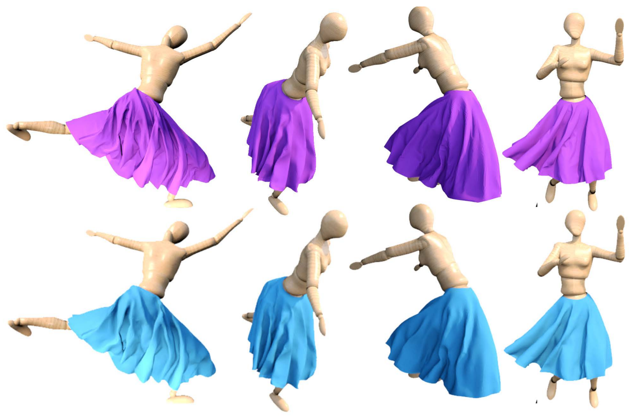

Fig. 10. Comparison between ground-truth tracked simulation (top) and our super-resolved meshes (bottom), on the test frames in SKIRT dataset. Our method succeeds to predict the dynamic wrinkles of the long skirts as realistic as physic-based ones.

Generalization to new motions. In Fig. 10, we show the deformed long skirt produced by our approach on the mannequins while changing various poses over time. The human poses are from two testing motion sequences 05_04 in the subject of modern dance and 55_02 in the subject of lambada dance [51]. We visually compare the results of our algorithm with the ground-truth simulation. The mid-scale wrinkles are successfully predicted by our approach when generalizing to various dancing motions not in the training set. For instance, in the first column of Fig. 10, the skirt slides forward and forms plausible wrinkles due to an extended and straight leg caused by the character pose of sideways arabesque. As for dancing sequences, please see the accompanying video for more animated results and further comparisons.

Comparison with other methods. Given detailed meshes simulated by the physics-based technique as ground truth, we compare our results with our implementation of a CNN-based method [13] and a conventional machine learning-based method [8]. The performance is evaluated on the Tablecloth dataset combining DRAPING and HITTING by a single network. The partition of the dataset for training and testing and the training parameters of our method are the same with the setting illustrated in § 6.

We train the network of Chen *et al.*[13] with the same setting reported in their paper [13]. The peak signal-to-noise ratio (PSNR) and vertex-wise mean square error (VMSE) are used to evaluate the quality of reconstructions, quantitatively. As shown in Table 2, our MFSR gains better performance than [13] with a higher PSNR and a lower VMSE. As shown in Fig. 11, with the LR meshes as inputs, our MFSR successfully produces rich and consistent wrinkles thanks to multiple features, while the results of Chen *et al.*approximate inaccurate wrinkles depending on the position. The velocity and kinematics-based loss functions also contribute to more stable results than Chen *et al.*(please refer to the accompanying video). The differences between the result and the ground truth are highlighted in Fig. 11 using color coding. In the results of Chen *et al.*, it clearly highlights the bottom left, bottom right corners and wrinkle lines, where our results look closer to the ground truth.

In addition, we compare our MFSR with state-of-the-art data-driven method [8] (not deep learning based) to generate cloth wrinkles. Our results have lower vertex-wise error than [8] as shown in Table 2. As mentioned in [8], their method handles quasistatic

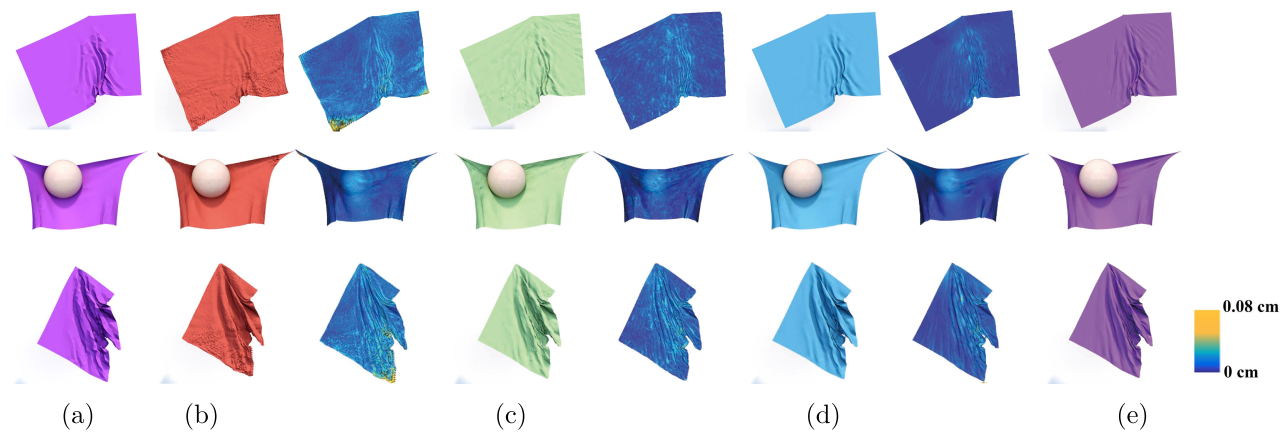

Fig.11. Comparison of the reconstruction results for unseen data in the Tablecloth dataset combining DRAPING and HITTING. (a) the input coarse meshes, (b) the results of Chen *et al.*[13], (c) the results of Zurdo *et al.*[8], (d) our results, (e) the ground truth. The reconstruction accuracy is qualitatively showed as a difference map. Reconstruction errors are color-coded and warmer colors indicate larger errors. Our method leads to significantly lower reconstruction errors.

wrinkle formation without dynamic features. They use the edge ratio between current and the rest state as mesh descriptors, contrarily, our algorithm enhances the LR deformation using multiple descriptors with spatial and dynamic information. Fig. 11 and the accompanying video show that the results of [8] have small artifacts and are lack of temporal coherence, while our technique can realize stable and realistic cloth. Besides, collisions are not handled in their work. Our approach solves cloth penetrations with controllable cost (see in Table 1).

Table 2. Comparison of pixel-wise and vertex-wise error values (PSNR/VMSE) of Zurdo *et al.*[8], Chen *et al.*[13] and our MFSR. Our results gain better performance than them with a higher PSNR and a lower VMSE.

Benchmark	Methods	Metrics	
		PSNR ↑	VMSE ↓
DRAPING	Chen *et al.*	59.07	4.19e-4
	Zurdo *et al.*	-	1.30e-4
	Ours	**68.91**	**7.09e-5**
HITTING	Chen *et al.*	59.15	1.17e-4
	Zurdo *et al.*	-	5.78e-5
	Ours	**72.25**	**4.69e-5**

7.3 Ablation study

Next, we study the effect of different components of our proposed network, including loss function and network architecture.

Loss function. To demonstrate the effectiveness of our proposed loss functions, we conduct the experiments with different loss combinations on three datasets, *i.e.* , DRAPING, HITTING, and SKIRT, respectively. The training and testing datasets are selected as mentioned in § 6. We use the displacement loss as the baseline and progressively add the remaining loss terms of our mesh MFSR, to obtain the comparative results.

Table 3 reports the quantitative evaluation of PSNR between generated displacement images and ground truth in various settings of loss functions. Red text indicates the best performance and the blue text indicates the second-best. The result shows that our algorithm has either a best or second-best performance through combining all loss terms in a multi-task learning framework. Notice that without the constraints of velocity and kinematics-based loss, the results with $\mathcal{L}_n$ gain lower PSNR although it encourages wrinkle gen-

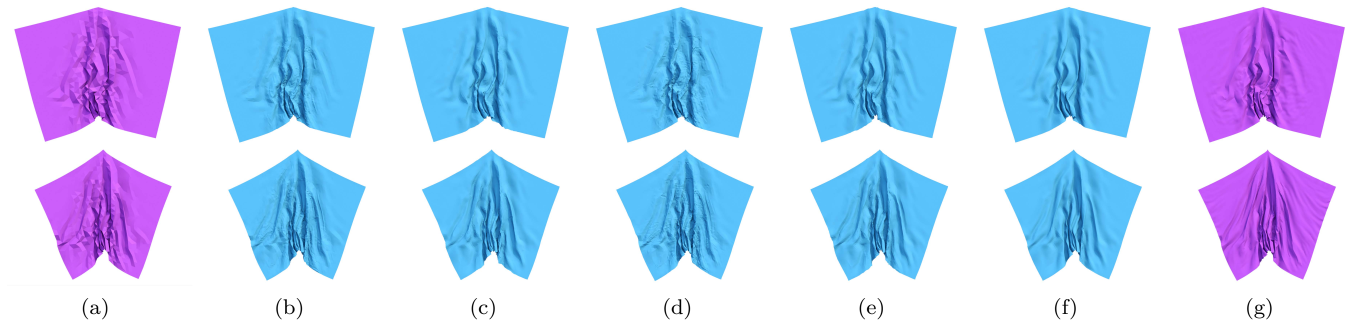

Fig.12. The evaluation of each loss term for our mesh MFSR model. (a) input LR mesh, (b) results with $\mathcal{L}_d$, (c) results with $\mathcal{L}_{d+n}$, (d) results with $\mathcal{L}_{d+v}$ (e) results with $\mathcal{L}_{d+n+v}$, (f) our results, (g) ground truth.

eration in SR results.

Table 3. Comparison of PSNR of displacement images using different training loss terms.

Benchmark	$\mathcal{L}_d$	$\mathcal{L}_{d+n}$	$\mathcal{L}_{d+v}$	$\mathcal{L}_{d+n+v}$	$\mathcal{L}_{all}$
DRAPING	67.90	62.83	**67.92**	63.47	**68.68**
HITTING	67.11	68.23	**71.41**	69.75	**71.26**
SKIRT	**62.48**	60.76	**62.48**	61.31	**62.88**

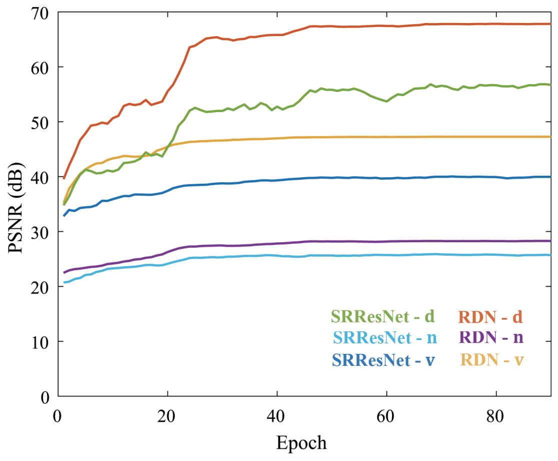

Fig. 13. Convergence analysis of the displacement **d**, the normal **n** and the velocity **v** with RDN [17] and SRResNet [16]. PSNR between the inference of super-resolution networks and the ground truth is used for this quantitative evaluation, where RDN achieves better performance.

In Fig. 12, we show visual results from the DRAPING dataset, to evaluate the performance of different loss combinations. The baseline model (see Fig.12(a)) only including displacement loss generates cloth meshes with too many unrealistic small wrinkles and inconsistent thin lines. Directly combining velocity loss with the baseline (see Fig.12(c)) is not able to solve this problem, contrarily introducing some unexpected horizontal short lines. The results including normal loss also suffer from uneven lines and sharp triangles on the folders and boundaries (*i.e.*, several buckling triangles near the handler and center wrinkle lines on the top of Fig.12(b)(d)). Finally, the results with all our proposed loss functions (see Fig.12(e)) are visually very close to the ground truth (see Fig.12(f)) with the realistic and consistent wrinkles and folders, which verifies the effectiveness of our proposed four loss functions.

Network architecture. To further investigate the performance of different networks (SRResNet and RDN), we conduct experiments on the DRAPING dataset. In particular, we validate our cloth animation results on randomly selected 800 pairs of LR/HR meshes from the DRAPING dataset, which are excluded from the training set, and cover different complex hanging motions in pendulum movement. In Fig. 13, we depict the convergence curves of three different features in the above validation dataset. The convergence curves show that RDN achieves better performance than SRResNet, and further stabilizes the training process in all three features. The improved performance and stabilization are benefited from contiguous memory, local residual learning and global feature fusion in RDN. In SRResNet, local convolutional layers do not have direct access to the subsequent layers so it neglects to fully use information of each convolutional layer. As a result, RDN achieves better performance than SRResNet.

8 Conclusions and future work

In this paper, we propose a multi-feature framework to synthesize details for cloth animations. We embed spatial and temporal features of 3D meshes into multiple images and learn a super-resolution network jointly. We also design a kinematics-based loss to maintain the temporal consistency of the predicted sequences. Quantitative and qualitative results reveal that our method can synthesize realistic wrinkles in various scenes, such as draping cloth, garments interacting with moving balls and human bodies. We also give details on synchronized simulation, as it contributes to construct paired 3D meshes. These aligned coarse and fine meshes can also be used for other applications such as 3D shape matching of incompatible shape structures.

Limitations and Future work. Nevertheless, several limitations remain open for the future work. In our work, the training data is the paired LR/HR meshes generated by a synchronized simulation. While tracking the LR cloth, the HR cloth cannot show dynamic properties of a full simulation. We would like to address this limitation by imposing unsupervised learning or cycle/dual generative adversarial networks to learn a mapping between the high resolution meshes and low resolution meshes in the future. In addition, the dataset should be further expanded including more scenes, motion sequences, and garment shapes to create more diverse results. Our work is not independent of physics based simulation, but is an acceleration one. Thus the estimated wrinkles from networks are related to the materials setting in physics-based simulation phase. It could be an interesting future direction to generalize our method to diverse materials thus generate different types of wrinkles.

References

[1] J. Liang and M. C. Lin, "Machine learning for digital try-on: Challenges and progress," *Computational Visual Media*, pp. 1–9, 2020.

[2] M. Wang, X. Q. Lyu, Y. J. Li, and F. L. Zhang, "VR content creation and exploration with deep learning: A survey," *Computational Visual Media*, pp. 1–26, 2020.

[3] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," in *SIGGRAPH*, 1987, pp. 205–214.

[4] X. Provot, "Collision and self-collision handling in cloth model dedicated to design garments," in *EG Workshop on Computer Animation and Simulation*, 1997, pp. 177–189.

[5] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *SIGGRAPH*, 1998, pp. 43–54.

[6] R. Bridson, S. Marino, and R. Fedkiw, "Simulation of clothing with folds and wrinkles," in *Proc. Symp. Computer Animation*, 2003, pp. 28–36.

[7] H. Wang, F. Hecht, R. Ramamoorthi, and J. O'Brien, "Example-based wrinkle synthesis for clothing animation," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 107:1–107:8, 2010.

[8] J. S. Zurdo, J. P. Brito, and M. A. Otaduy, "Animating wrinkles by example on non-skinned cloth," *IEEE Trans. Visual. Comput. Graph.*, vol. 19, no. 1, pp. 149–158, 2013.

[9] I. Santesteban, M. A. Otaduy, and D. Casas, "Learning-based animation of clothing for virtual try-on," in *Computer Graphics Forum*, vol. 38, no. 2. Wiley Online Library, 2019, pp. 355–366.

[10] W.-W. Feng, Y. Yu, and B.-U. Kim, "A deformation transformer for real-time cloth animation," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 108:1–108:10, 2010.

[11] E. de Aguiar, L. Sigal, A. Treuille, and J. K. Hodgins, "Stable spaces for real-time clothing," *ACM Trans. Graph.*, vol. 29, no. 3, pp. 106:1–106:9, 2010.

[12] L. Kavan, D. Gerszewski, A. W. Bargteil, and P.-P. Sloan, "Physics-inspired upsampling for cloth simulation in games," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 93:1–93:10, 2011.

[13] L. Chen, J. Ye, L. Jiang, C. Ma, Z. Cheng, and X. Zhang, "Synthesizing cloth wrinkles by CNN-based geometry image superresolution," *Computer Animation and Virtual Worlds*, vol. 29, no. 3-4, p. e1810, 2018.

[14] Y. J. Oh, T. M. Lee, and I.-K. Lee, "Hierarchical cloth simulation using deep neural networks," in *Proceedings of Computer Graphics International 2018*, 2018, pp. 139–146.

[15] Z. Lähner, D. Cremers, and T. Tung, "Deepwrinkles: Accurate and realistic clothing modeling," in *European Conference on Computer Vision.* Springer, 2018, pp. 698–715.

[16] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, and Z. Wang, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016.

[17] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1–10.

[18] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," *ACM Trans. on Graph.*, vol. 21, no. 3, pp. 355–361, 2002.

[19] R. Narain, A. Samii, and J. F. O'Brien, "Adaptive anisotropic remeshing for cloth simulation," *ACM Trans. on Graph.*, vol. 31, no. 6, pp. 147:1–10, 2012.

[20] T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan, "Fast simulation of mass-spring systems," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1 – 7, 2013.

[21] P. Guan, L. Reiss, D. A. Hirshberg, A. Weiss, and M. J. Black, "DRAPE: Dressing any person," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 35:1–35:9, 2012.

[22] 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.*, vol. 32, no. 4, pp. 87:1–87:8, 2013.

[23] E. Gundogdu, V. Constantin, A. Seifoddini, M. Dang, M. Salzmann, and P. Fua, "Garnet: A two-stream network for fast and accurate 3d cloth draping," *arXiv preprint arXiv:1811.10983*, 2018.

[24] T. Y. Wang, D. Ceylan, J. Popovic, and N. J. Mitra, "Learning a shared shape space for multimodal garment design," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1:1–1:14, 2018.

[25] T. Y. Wang, T. Shao, K. Fu, and N. J. Mitra, "Learning an intrinsic garment space for interactive authoring of garment animation," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, p. 220, 2019.

[26] 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.*, vol. 33, no. 4, pp. 105:1–105:9, 2014.

[27] Y. P. Xiao, Y. K. Lai, F. L. Zhang, C. p. Li, and L. Gao, "A survey on deep geometry learning: From a representation perspective," *Computational Visual Media*, vol. 6, no. 2, pp. 113–133, 2020.

[28] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.

[29] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *IEEE ICCV*, 2015.

[30] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3D shape surfaces using geometry images," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 223–240.

[31] Q. Tan, L. Gao, Y. Lai, J. Yang, and S. Xia, "Mesh-based autoencoders for localized deformation component analysis," *CoRR*, vol. abs/1709.04304, 2017.

[32] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia, "Variational autoencoders for deforming 3d mesh models," *arXiv preprint arXiv:1709.04307*, 2017.

[33] L. Gao, Y.-K. Lai, D. Liang, S.-Y. Chen, and S. Xia, "Efficient and flexible deformation representation for data-driven surface modeling," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 5, pp. 1–17, 2016.

[34] L. Gao, Y.-K. Lai, J. Yang, Z. Ling-Xiao, S. Xia, and L. Kobbelt, "Sparse data driven mesh deformation," *IEEE transactions on visualization and computer graphics*, 2019.

[35] M. Zhang, T. Wang, D. Ceylan, and N. J. Mitra, "Deep detail enhancement for any garment," *arXiv e-prints*, pp. arXiv–2008, 2020.

[36] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[37] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European Conference on Computer Vision*, 2016, pp. 391–407.

[38] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *IEEE CVPR*, 2016, pp. 1874–1883.

[39] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks for super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1664–1673.

[40] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, "Video super-resolution with convolutional neural networks," *IEEE Transactions on Computational Imaging*, vol. 2, no. 2, pp. 109–122, 2016.

[41] M. Chu, Y. Xie, L. Leal-Taixé, and N. Thuerey, "Temporally coherent gans for video super-resolution (tecogan)," *arXiv preprint arXiv:1811.09393*, vol. 1, no. 2, p. 3, 2018.

[42] P. Bhattacharjee and S. Das, "Temporal coherency based criteria for predicting video frames using deep multi-stage generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 4268–4277.

[43] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–15, 2018.

[44] W. Kabsch, "A discussion of the solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 34, no. 5, pp. 827–828, 1978.

[45] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[46] M. Bergou, S. Mathur, M. Wardetzky, and E. Grinspun, "TRACKS: Toward directable thin shells," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 50:1–10, 2007.

[47] M. Müller and M. Gross, "Interactive virtual materials," in *Proceedings of Graphics Interface 2004*. Canadian Human-Computer Communications Society, 2004, pp. 239–246.

[48] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[49] R. Burden and J. Faires, "The bisection algorithm. numerical analysis," 1985.

[50] J. Ye, G. Ma, L. Jiang, L. Chen, J. Li, G. Xiong, X. Zhang, and M. Tang, "A unified cloth untangling framework through discrete collision detection," *Computer Graphics Forum*, vol. 36, no. 7, pp. 217–228, 2017.

[51] J. Hodgins, "Cmu graphics lab motion capture database," 2015.

[52] H. Wang, J. F. O'Brien, and R. Ramamoorthi, "Data-driven elastic models for cloth: modeling and measurement," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 71:1–71:12, 2011.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.