Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Neural Inpainting of Folded Fabrics with Interactive Editing

Guillaume Gisbert, Raphaëlle Chaine, David Coeurjolly

INSA Lyon, Universite Claude Bernard Lyon 1, CNRS, LIRIS, UMR5205, F-69621 Villeurbanne, France

ARTICLE INFO

Article history:
Received July 1, 2024

Keywords: Inpainting, Deep Learning, Fabric surfaces, Developability

ABSTRACT

We propose a deep learning approach for inpainting holes in digital models of fabric surfaces. Leveraging the developable nature of fabric surfaces, we flatten the area surrounding the holes with minor distortion and regularly sample it to obtain a discrete 2D map of the 3D embedding, with an indicator mask outlining holes locations. This enables the use of a standard 2D convolutional neural network to inpaint holes given the 3D positioning of the surface. The provided neural architecture includes an attention mechanism to capture long-range relationships on the surface. Finally, we provide *ScarfFolds*, a database of folded fabrics patches with varying complexity, which is used to train our convolutional network in a supervised manner. We successfully tested our approach on various examples and illustrated that previous 3D deep learning approaches suffer from several issues when applied to fabrics. Also, our method allows the users to interact with the construction of the inpainted surface. The editing is interactive and supports many tools like vertex grabbing, drape twisting or pinching.

1. Introduction

3D scanning of shapes has become very popular in various industries including film, video game production and cultural heritage preservation. However, the output of these scanners can suffer from several flaws due to the quality of the material, the convexity of the scanned object or its material requiring to fill missing or corrupted areas. Not to mention that some objects, especially in the field of archeology and cultural heritage, are already damaged with missing parts. In this work, we present a learning-based method for filling holes in digital fabric surfaces. Fabrics are particularly prone to self-occlusion due to their numerous folds and wrinkles, but they have an interesting geometric property of local developability. By essence, they can be isometrically deformed into the plane. While this is not entirely accurate for most textile materials due to their inherent stretchiness and elasticity, it is still a useful approximation that we leverage in our work.

Convolutional Neural Networks (CNNs) have transformed machine learning and are still widely used in the most advanced models. Their success is due to their ability to analyze the local structure of the data and to reduce the number of parameters compared to multi-layered perceptrons (MLPs). This kind of network is particularly well-suited for analyzing isotropic grid-structured data, such as images. However, most 3D data are unstructured, anisotropic, and lack an obvious orientation for the convolution kernels. Yet, many methods tackle this challenge by devising a clever use of MLP or by adapting convolution [1, 2, 3]. Another way to overcome these difficulties is by using voxels and 3D CNNs. However, they have their own issues, mainly spatial discretization and limitation to low resolutions. Besides, brute force 3D discretization can lead to incorrect topological representation in fabric surfaces with intricate folds.

The idea of our method is to take advantage of the efficiency of 2D CNNs while working on 3D mesh data, with the help of a

e-mail: guillaume.gisbert@liris.cnrs.fr (Guillaume Gisbert), raphaelle.chaine@liris.cnrs.fr (Raphaëlle Chaine), david.coeurjolly@liris.cnrs.fr (David Coeurjolly)

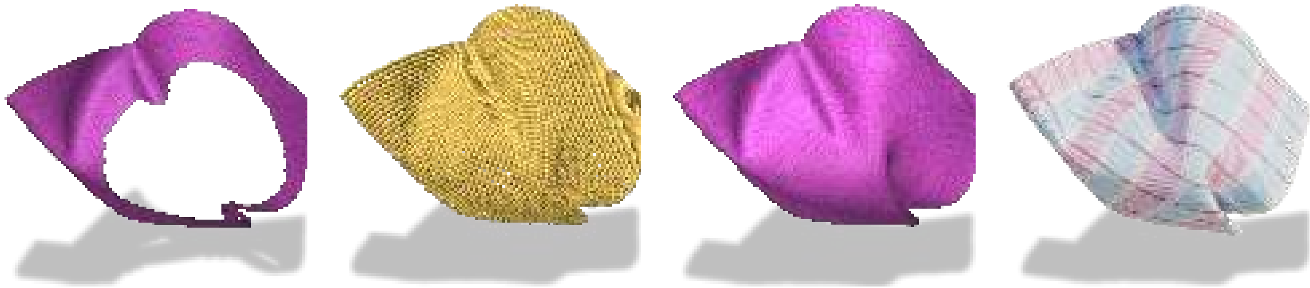

Fig. 1. From a holed mesh, the network produces a point cloud filling the hole. Since the point cloud is parametrized on a grid structure, it can be trivially meshed and textured.

3D to 2D parametrization that is respectful of the surface intrinsic geometry. Such parametrization is almost straightforward in the highly-developable case of fabrics. Indeed, fabric materials and clothing items are typically made from flat pieces, meaning most textiles can locally be flattened with minimal distortion (except at seams). Next, regularly sampling the parametrized mesh will also regularly sample the 3D mesh and provide a grid data input for a 2D convolutional network. Thus, the input data can be considered as an image where each channel corresponds to a coordinate in 3D.

Our main contributions are :

- We propose a fast approach to inpaint a hole within a flattanable connected piece of fabric, by using a 2D convolutional neural architecture.
- Our inpainting technique is accompanied by a set of editing operations enabling users to guide the positioning of the inpainting patch.
- We offer a base of examples that reflects the variety of folds that can arise on a square of fabric. This database, called *ScarfFolds*, was used to train our network in a supervised manner and it will be made available to the community for future comparisons.

2. Related work

Numerous methods have been developed to address the issue of inpainting 3D data. The following is a brief overview of relevant works for each range of methods.

Geometric methods. General methods use local geometric information to fill the hole by using a minimal area surface, subject to various constraints including regularization. The most well-known method was proposed by Liepa [4] which consists in finding a minimal area triangulation of the hole. The triangulation is refined and smoothed to match the size of neighboring triangles. Variational approaches have been developed to improve the regularization of the minimal surface. For instance, Baumgärtner et al. [5] proposed to estimate the normal field of a mesh using total variation as a regularizer for denoising and inpainting purposes. Similarly, Bonneel et al. [6] solve a Mumford-Shah problem on a mesh to estimate the normal field. The mesh is then deformed so that the faces normals corresponds to the target normals. Other works presented ad-hoc procedures such as Feng et al. [7] and Attene [8]. The former changes its inpainting strategy depending on the size of the hole while the latter produces watertight meshes by minimizing normal variation and by removing defects using swap/collapse algorithms. An alternative approach to first meshing the hole is to unfold the mesh in the plane, triangulate it and embed the mesh back into $\mathbb{R}^3$ as a minimum energy surface favoring minimal area and fairing [9]. These generic methods are generally able to propagate features in the missing area, without enforcing global prior such as developability. In fact, those approaches are unaware of the area of surface needed to fill the hole, hence the choice of minimizing it.

We can also mention the work from Harary et al. [10] which is a context-based method. Holes are filled with given patches chosen from the input mesh based on a dissimilarity metric to minimize coherence error. As far as the inpainting of fabric surfaces is concerned, Gisbert et al [11] favored developability by using an isometric map to unfold the boundaries of the hole into the plane, in order to obtain a patch of zero Gaussian curvature. In this approach, the embedding in 3D then tends to preserve the intrinsic geometry of the patch bounded by the hole boundary, while locally balancing the mean curvature (variational approach minimizing an energy inspired by those found in cloth simulation). Our approach is inspired by this approach, in the sense that it requires an isometric parametrization of the surface at the boundary of the hole.

Implicit Surfaces. These methods estimate an implicit function in $\mathbb{R}^3$. Davis et al. [12] constructs an ambient signed distance function from the input mesh and extends it to cover the missing area. The zero set of that function represents the surface. The diffusion process can benefit from knowing the point of view of the scanner, and produces watertight meshes. Likewise, Kazhdan et al introduced Poisson Surface Reconstruction [13, 14], a method able to reconstruct surfaces from a point cloud with its corresponding normals by solving a Poisson equation. This approach was found to be efficient for filling holes and merging shapes. Overall, these methods are rather robust and can process complex holes and unstructured data. However, those approaches were not yet adapted to enforce global prior such as developability. They tend to smooth the output and may struggle with rendering complex features.

Learning based. Deep learning methods can efficiently identify the structure of a shape after supervised training on a category of similar objects. This can be useful for shape

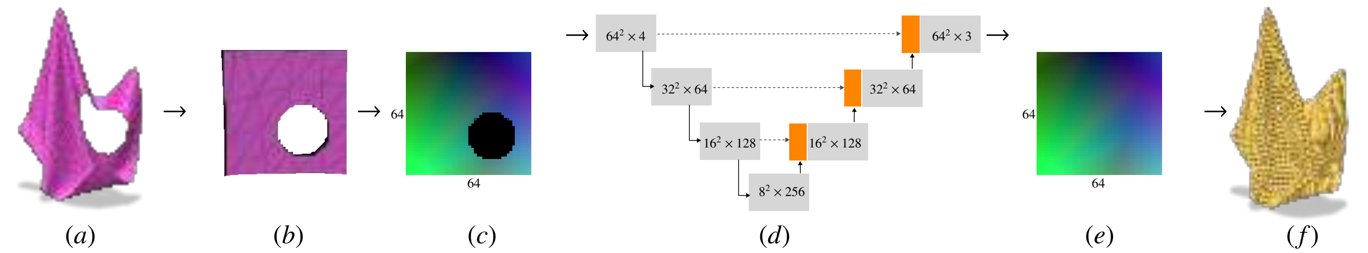

Fig. 2. Pipeline overview: starting from an input mesh with holes (a), we first parametrize the input geometry into a 2D domain (b) that is discretized into a 64×64 $4D$ image encoding the input mesh geometry (c) ($xyz-$positions are encoded as RGB values) and a hole identificator mask. An attention U-net is used to inpaint the missing part (d, e) (convolution and downsampling blocks in grey, attention blocks in orange). A final point cloud can then be reconstructed (f).

analysis, shape matching and shape classification, but also for shape reconstruction and surface inpainting. Some methods such as AtlasNet [15, 16, 17] learn to predict the surface of 3D shape from a point cloud or from a trimmed 2D image. The neural architecture they use fit several planar patches to a point cloud and use them to obtain a parametrization. Likewise, FoldingNet [18] learns to fold a single planar grid into any shape. The primary focus of the method is shape classification, but it also enables shape interpolation. In a slightly different register, Point2Mesh [19] attempts to predict watertight mesh from a point cloud. The network weights are optimized to iteratively deform the convex hull to shrink-wrap a given point cloud. Because the architecture is a convolution network, the shared weights favor local geometric self-similarity, which can be helpful for inpainting tasks.

We finally focus on machine learning works specifically designed for inpainting tasks. Deep learning has been successfully used for image inpainting. Existing approaches are usually based on the use of 2D convolution networks [20, 21], and more recent works rely on the use of attention or diffusion techniques [22, 23, 24, 25, 26]. Diffusion networks have recently demonstrated great performances in image generation and image inpainting; however, they typically suffer from longer inference times that would be detrimental to our application (see Section 3.5) We also use U-Net architecture but train it in a standard supervised manner. By doing so, we sacrifice the generation diversity, with a network that produces only one output for each input. This trade-off ensures that our system remains efficient while maintaining reliable and consistent results, making it well-suited for applications where rapid processing is crucial. Inpainting object surfaces is generally more intricate. DeepMend [27] proposed a deep learning architecture to repair fractured shapes by learning occupancy functions. One advantage is that no ground truth knowledge of the fractured region is required. However, the output is not guaranteed to be simply connected. Many methods work with voxels [28, 29, 30, 31] which enables the use of powerful 3D convolutional neural networks at the expense of an often coarse discretization of the geometry. Wen et al. [32] proposed an architecture for predicting missing parts in a point cloud. They introduced a skip-attention mechanism, similar to Attention U-Nets [33], based on PointNet++ [34]. Deep Mesh Prior [35] introduced a method for repairing meshes (denoising or inpainting) by mapping a noisy mesh to the input. Similar to Point2Mesh [19], the shared weights of the Graph Convolutional Network help with self-similarity and reconstruction. Sarmad et al. [36] proposed RL-GAN-Net, a method for inpainting partial point clouds of shapes using a GAN. The method incorporates reinforcement learning (RL) agents to determine the optimal input for the GAN, resulting in improved reconstructions. Overall, these methods are not tailored to work with garments and fabrics, as they are designed for objects with similar 3D structures. The diversity and complexity of folds can be difficult to learn with these approaches.

Deep learning methods have also been applied to garment representation. Some works focus on virtual try-on [37, 38, 39], which essentially is draping a body. Gundogdu et al. [37] introduced a method to drape a 3D body with a garment in near real-time. They use a two-stream architecture inspired by PointNet [40] which processes both the garment and the body as point clouds. Bertiche et al. [41] proposed a framework for unsupervised garment animation. They use a recurrent encoder-decoder architecture that takes the pose as input and generates the garment as output. The losses are inspired by physically-based simulation. These methods studied garment generation in the context of machine learning, but they were not designed for filling in an incomplete fabric mesh. They are usually guided by a static or animated virtual human body as input. Our approach differs from previous ones in that it does not rely on end-to-end learning. The boundary of the missing patch is determined in a planar configuration, after isometric parametrization of the hole boundary, without learning machinery. Then we use the power of neural networks to embed the missing surface patch in 3D, using a data set of folds examples.

3. Method

Given a mesh representing a fabric with a hole, the method outputs a regular sampling of an highly-developable reconstructed geometry and allows some user editing tools to control the result.

Our approach proposes to fill in the holes of the mesh by exploiting a dataset of multiple examples of fabrics in a wide variety of positions and folds. To achieve this, we use the property that pieces of fabric without seams can be considered as developable surfaces to some extent, allowing them to be flattened with minor metric distortions. Flattening the fabric around the holes reveals the 2D shape of the patch needed to fill them, which is essential to our approach. Furthermore, revealing the Euclidean 2D structure of the surface makes it possible to benefit from the most efficient architectures for inpainting a 2D signal.

Our approach proceeds in several steps. First we use an isometric parametrization of the fabric around the holes. Then we discretize the parametric space into a 2D grid and use a mask to distinguish between empty cells (null occupancy corresponding to unknown 3D embedding) and cells with 3D embedding. Then, the region around the hole can be seen as a 4-channel image, which can be used as input to an attention U-Net inpainting network and where each channel represents a coordinate in 3D, plus an occupancy information. This image-like surface is fed to a convolutionnal network that outputs an inpainted result, that can be interpreted as a point cloud and easily meshed because of the grid structure of the output. The pipeline of our approach is illustrated in Fig 2.

3.1. Parametrization

The idea of parametrizing a mesh to process it in a neural network is not new [42] but it is particularly well-suited for our developable case. In fact Gisbert et al [11] also used an isometric parametrization of the area surrounding a hole in a fabric surface in order to flatten it. Unlike this approach, which can still work with minimal surface information around the hole (a single triangle strip is enough in case of high developability), we need more information from the surrounding surface, to feed the network with better contextual information.

Let S^* denote the surface mesh to be inpainted. We focus on a region S of the mesh around the hole to be flattened. Since we aim at parametrizing the mesh isometrically, we opt for an As-Rigid-As-Possible (ARAP) [43, 44] parametrization method. ARAP tries to preserve the lengths hence the shape of each triangle. Formally, the ARAP energy can be written as follows:

$$\frac{1}{2}\sum_{f\in F}\sum_{e_{ij}\in E(f)} cot(\theta_f^{ij})\left\|e_{ij}^u - R_f\, e_{ij}\right\|^2 ,$$

where F denotes the set of triangular faces of S, $E(f)$ the set of edges of the face f, θ_f^{ij} the angle opposite to the 3D edge e_{ij} for vertices i and j in f (thus in $\mathbb{R}^3$), e_{ij}^u the same edge in the parametric space (in $\mathbb{R}^2$). Finally R_f is the rotation matrix to be optimized on a per-face basis.

3.2. 2D inpainting network

The main ingredient of our hole-completion model is an attention U-Net [33], which is a U-Net that includes a spatial attention mechanism. Attention U-Net is a standard convolutional neural network that requires grid-like structured data as input. This kind of network was designed for segmentation tasks, but has also been successful for inpainting tasks. A 64*64 grid structure is added in the parametric plane, with (x, y, z)-information at each point and a fourth channel information to indicate whether the point is in the hole. In the case of a hole point, the (x, y, z)-information is not meaningful. This 4D information is directly fed into a U-net network, that encloses 5 levels of scales. A module of attention is present at the skip-connection of the first 4 levels with the output of the next level. This increases the power of the network to take large scale interactions into account. In particular, the position of points outside the hole can influence the position of points that are inside. The network outputs a new grid, with a meaningful (x, y, z)-information for each point, including in-hole points that are filled.

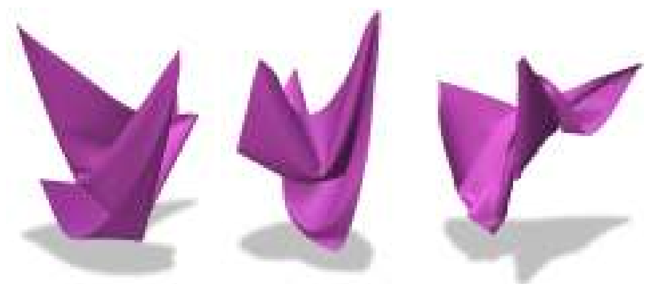

Fig. 3. Three examples of our dataset. All examples are subject to gravity.

3.3. The ScarfFolds dataset

The network has been trained on a synthetic dataset made with the Blender cloth simulator [45]. The *ScarfFolds* dataset consists of 5040 3D triangulated meshes of 2601 vertices representing square pieces of fabric deformed in space. To produce the dataset, the simulation starts with a predefined flat square mesh and randomly selects four positions in space. These points serve as target positions for the four corners of the mesh. During the simulation, the corners are linearly transported to their target positions and the fabric is also subject to gravity. This approach does not guarantee the absence of self-intersections, so the intersecting examples are filtered out (50 % of rejections). See Fig 3 for some examples of our dataset. The dataset will be made publicly available upon acceptance.

3.4. Supervised training

The *ScarfFolds* dataset contains a variety of squares of fabric in various positions and folds. These regular meshes are complete in the sense that they do not contain any holes, and can be used as ground truth by our network during training. Holes must also be taken into account to train the network to handle a wide variety of situations. These holes are not specified in the database and are generated randomly, in the 2D parametric space, to enable an even wider variety, and greater generalization power for the network. Three kinds of holes were generated: circles, rings and percolation holes (See figure 4 for an illustration). Circles and rings have a random radius and center. In a ring, we only keep a portion of the surface in a second given radius around a circle hole. Percolation holes are simply made by choosing a random point to delete and then deleting one of its neighbours, p times. During training, all the examples are trained with a random hole at each epoch.

We present a three terms loss function to minimize the weights of the network at training. The formulation comprises a data term aimed at minimizing the distance between the ground truth vertices and the predicted vertices but also the distance between the normal vectors. The remaining regularization term

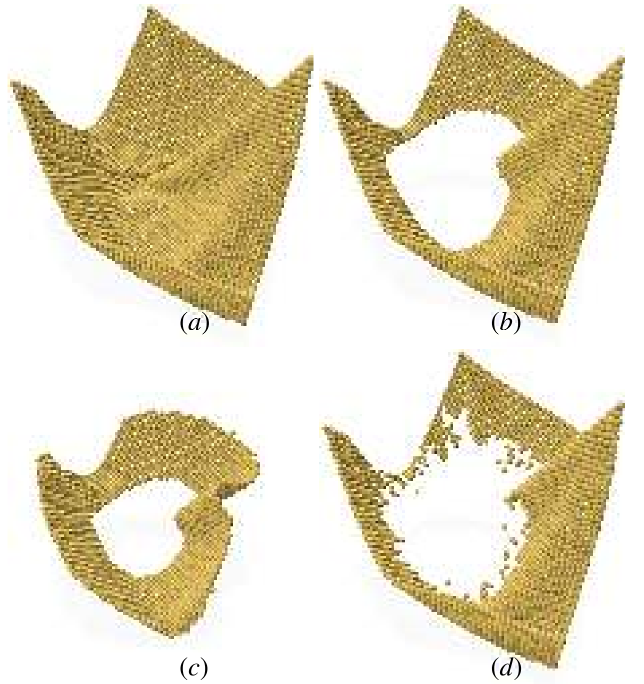

Fig. 4. Various types of holes generated for the training process: (a) the ground-truth, (b) a circular hole, (c) a ring hole, (d) a percolation hole

helps with the intersection-free completion of the missing area. Hence, the loss is defined as:

$$L = L_{data} + \alpha L_{grad} + \beta L_{selfpen}\,,$$

where α and β are user-defined parameters to balance the importance of each component.

Data Term. This is a simple MSE between the ground truth vertices and the predicted ones:

$$L_{data} = \frac{1}{N}\sum_i \left\|v_i - v_i'\right\|^2\,,$$

where v_i represent the ground-truth vertices positions, v_i' the predicted ones and N the number of vertices.

Gradient Term. This term is defined as the gradient of the vertices positions with respect to the uv directions in the parametrization space. Since the input of the network has a grid structure, the gradient can be easily computed with a kernel:

$$L_{grad} = \frac{1}{2N}\left(\sum_i \left\|\nabla_u v_i - \nabla_u v_i'\right\|^2 + \sum_i \left\|\nabla_v v_i - \nabla_v v_i'\right\|^2\right).$$

This first-order differential quantity enforces the surface normal to be consistent with the ground-truth ones. However, it carries more information because the norm is also informative.

Self-Penetration Term. To prevent the unpainted surface from self-intersecting, we add a third term to prevent vertices from being too close to each other.

$$L_{selfpen} = \sum_{i,j\ i\neq j} ReLU^2(\,r - \left\|v_i - v_j\right\|).$$

This energy is essentially a repulsive force that prevents vertices from entering a sphere of radius r around other vertices. r is set to be half of the mean distance between two adjacent vertices in the grid.

Interestingly enough, the training is efficient without requiring any additional term in the loss. The loss function to be optimized mainly includes ground truth fidelity on the *ScarfFolds* examples, and the only regularization term is to avoid self-intersections. Thus, the quality of the embedding is highly driven by the data and it is not necessary to add any regularization term to help the emergence of an isometric embedding, and to favor a high developability of the result.

3.5. Editing

At inference, our network computes a filling result for an input surface with holes that is consistent with the fabric examples of the *ScarfFolds* dataset. One important benefit is that the filling surface is generated very quickly, in a feed-forward manner. However, this solution may differ from what the user had in mind. Therefore, we propose to exploit the network's ability to take additional geometric constraints into account to guide the hole filling. These constraints can be of two types: either they can be defined jointly in parametric space and 3D space, or they can be defined solely in 3D, with an automatic optimization in 2D parametric space.

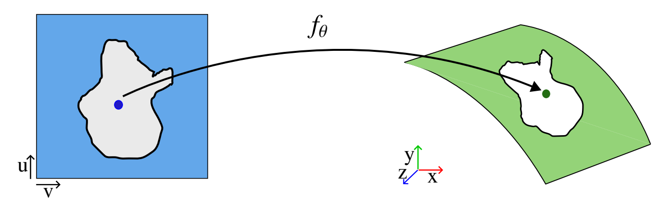

Fig. 5. Our trained network computes the neural 3D embedding of an entire patch of fabric to fill a partial 3D surface input (in green), after previously flattening it in 2D (in blue) using an isometric parametrization. For edition purposes, we can add additional constraints within the missing part.

First, we can select a 2D anchor point of the parametric space in the missing area and assign its position in 3D (see Fig. 5). The shape completion process considers the added vertex and reconstructs the shape according to the user's guidance. Thanks to this fast completion, the user can interactively adjust the target 3D position to his preferred choice. Then, various tools can be implemented to provide maximum editing freedom. The simplest one is point positioning, but fabric twisting or pinching may also be used (see Fig. 6). All these editing tools can be implemented by specifying the position of a few anchor points in the parametric plane and in 3D, jointly. Note that these are soft constraints so the target locations are not necessarily reached. The network learned how a piece of fabric should look like, and in particular, constraints that would overstretch the mesh are not met. Nonetheless, these surface editing tools are quite intuitive (see accompanying video in the Supplemental Materials).

An even more interesting editing option is to guide the generated surface towards a 3D point P by optimizing the uv-coordinates of the 2D anchor point in the parametric space. This

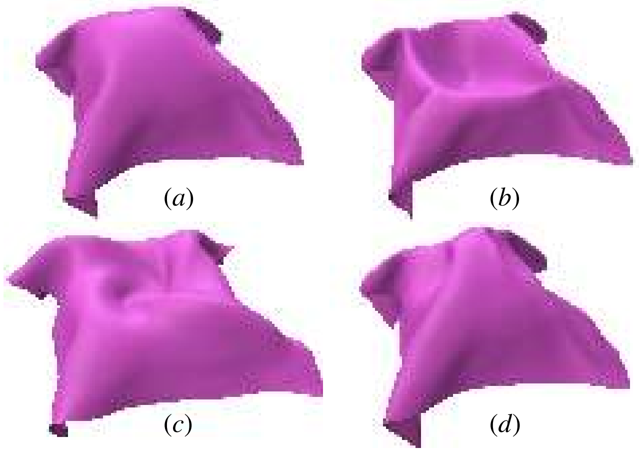

Fig. 6. User constraints for the shape completion. Various tools are shown: (*a*) No editing, (*b*) Point grabbing, (*c*) Drape twisted, (*d*) Drape pinching

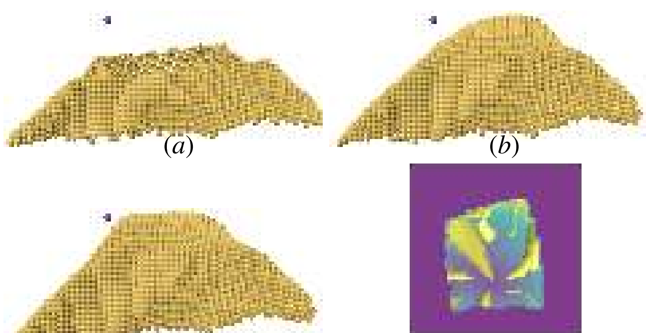

Fig. 7. The user sets a target point that for attracting the surface. The method automatically optimizes the best corresponding anchor point in the 2D parametric space and reconstructs the surface. From left-to-right, up-to-bottom: (*a*) Input; (*b*) First prediction; (*c*) Optimized surface, guided by the purple target; (*d*) Value and gradient of the anchor point adequacy in the parametric space.

is achieved by using auto-differentiation to minimize the distance between the neural 3D embedding of the anchor point and the target point P. This distance is used as a measure $L_Q(u, v)$ of the anchor point adequacy.

$$L_Q(u, v) = \left\| f_\theta^Y(u, v) - P \right\|^2 ,$$

where f_θ is the trained network, θ its parameters, Y is the 2D input grid of the network (including the hole mask and the 3D embedding of the surface around the hole), in which the hole mask has been modified at position (u, v) to account for the new 3D constraint. To minimize L_Q, the gradient with respect to the uv−coordinates is needed. Using the chain rule, this is formalized as follows:

$$\frac{\partial L_Q}{\partial(u, v)} = \frac{\partial L_Q}{\partial f_\theta} \frac{\partial f_\theta}{\partial Y} \frac{\partial Y}{\partial(u, v)} ,$$

For optimization purposes, we consider that the L_Q function takes continuous value of (u, v) as input and that the locally modified mask function places a very thin Gaussian in (u, v) (with slightly larger width than the grid resolution). This "Soft-Equal" function gives a differentiable function $L_Q(u, v)$. The energy is minimized using gradient descent and PyTorch auto-differentiation. Note that in our case, the L_Q function can only be evaluated at integer values of u and v. We slightly adapted the descent algorithm to ensure that the parameters are restricted to integer values and set the gradient to have a minimum norm of 1 to ensure that we move at least from one cell before convergence. Considering that the target position P is a soft constraint, the reconstructed mesh is an optimized balance between the constraints and the learned fabric properties. The framework is shown in Fig. 7.

4. Experiments

4.1. Experimental setup and results discussion

First, all the experiments were made using PyTorch [46] on an NVIDIA RTX A5000 GPU. Additionally, the mesh parametrization was done using C++ and LibIGL [47]. The source code can be found here[1].

Figure 8 shows some results on four examples. The first two rows are test examples from our *ScarfFolds* dataset (not used for training), the third row input is a developable synthetic example generated differently. The last row input was cropped from a statue scan (Caryatid).

For each example, we give statistics on the distance between horizontal and vertical neighbors of the grid after 3D embedding by our network. This distance is 1/64 in the planar patch. The high invariance of this distance shows that our network achieves a highly isometric 3D embedding of the planar patch and that the surface generated is highly developable.

The first three columns display the input and output of Gisbert et al. [11] which is a method that works all along with meshes. The last four columns show the input and results of learning methods using point clouds. Those fully neural approaches construct a parametrization of the shape from an input point set. For comparison purposes, we tried to specialize them on fabric surfaces by entirely retraining them on *ScarfFolds*. FoldingNet [18] fails to reconstruct the test examples of our database, a reason could be that it is an encoder-decoder architecture where the encoder aims at classifying the kind of the input based on its 3D embedding underlying structure. While this network clearly works with the ShapeNet [48] examples used with their experiments, the information contained in the latent space cannot capture the variety and complexity of folds that can be found in our dataset. Closer to our method, SANet [32] contains skip attention connections which help with the reconstruction. However, the parametrization output by the network is not as structured and isometric as ours. Gisbert et al. [11] produces convincing folds and results. Because their method is purely geometric, it is more sensitive to a lack of developability in the input compared to our data-driven approach. In Fig 13, we compare this method with ours on increasingly noisy examples. We also changed their parametrization method (LCSM [49]) to ours (ARAP [43]) to be fairer because LSCM quickly failed to produce a reasonable parametrization with noise.

[1] https://github.com/g-gisbert/Neural-Inpainting-Of-Folded-Fabric-Meshes/

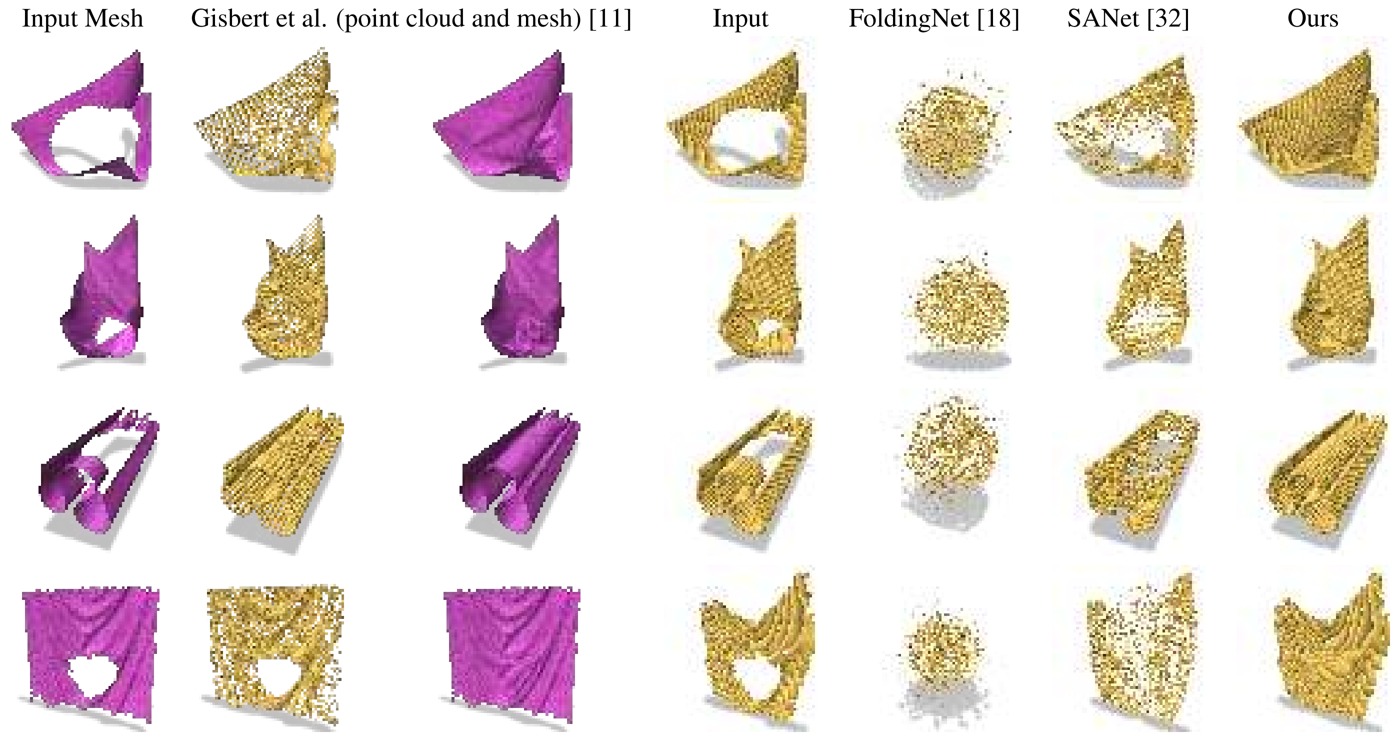

Fig. 8. Comparison between different hole filling methods. The two first rows are test examples from the dataset, the third one is a curtain obtained with a simulation, and the fourth row is taken from a statue scan (caryatid). The statistics regarding the 3D distance between neighbor grid points by using our neural approach is the following : mean: 0.01493, variance: 1.434e-06 (first row) mean: 0.01526, variance: 2.235e-06 (second row) mean: 0.01608, variance: 7.872e-07 (third row) mean: 0.01476, variance: 4.028e-06 (fourth row)

Table 1. Statistics of inference times (in seconds) between our method and Gisbert et al. [11]

(Test set of *ScarfFolds*.)

Method	Median	Variance
Gisbert et al.	4.33×10^{-1}	5.22×10^{-4}
Ours	6.32×10^{-3}	6.90×10^{-7}

Regarding timings, the geometric method of Gisbert et al. [11] computes the solution in about half a second while the order of magnitude for the learning methods is in the tens of milliseconds. We compared the exact timings of both methods on the test set of *ScarfFolds* and reported them in the table 1.

To assess the quality of fabrics holes inpainting, one approach is to compute the geometric distance between the ground truth surface and the predicted one. However, for a given hole boundary, there is no unique solution that satisfies fabric properties. Instead, we evaluate the quality of the reconstruction by measuring the Gaussian curvature, which reflects the developability of the predicted surface (see Fig. 9). We compare our results with those obtained by Gisbert et al., as well as with the ground truth. Interestingly, the network was able to reproduce the developability of the dataset without being explicitly constrained to minimize it. This result illustrates the ability of the network to inherently understand and preserve the geometric properties of the data.

Finally, we demonstrate the capabilities of our network on real-world statues (see Fig. 10). To fill the missing parts of these meshes, we parameterize a region around the hole and process it with our network. The network's output provides 3D positions on a grid, allowing us to place any triangulation in the parameterization space and obtain a corresponding 3D mesh. Due to the soft nature of the position constraints (l_2 data term), the predicted mesh may not perfectly align with the original input mesh. To address this, we project the inpainted mesh onto the input mesh to achieve the final results. This process ensures that the reconstructed regions seamlessly integrate with the original geometry, preserving the overall structure and appearance of the statues.

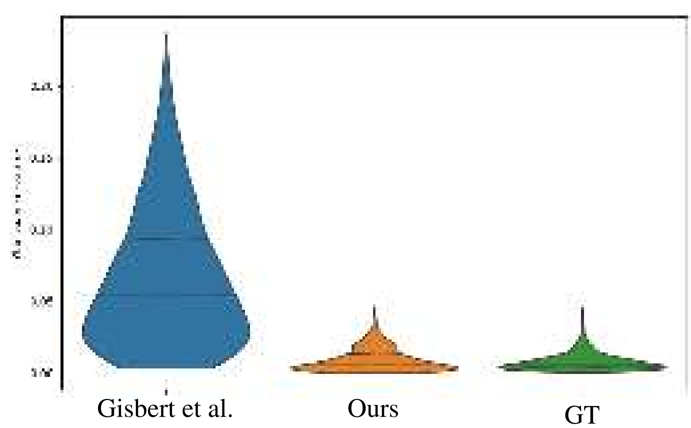

Fig. 9. Comparison of Gaussian curvatures respectively between the results produced by Gisbert et al. [11], our method and the ground truth. The measures were conducted on the test set of our *ScarfFolds* dataset.

Fig. 10. Result of our method on two statues (resp. Lucy and Caryatid). We show the input mesh with a hole, the input with the inpainted patch, and the zoomed versions.

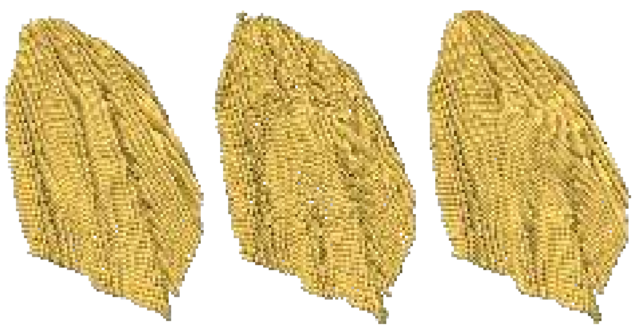

Fig. 11. Ablation study, from left to right: Our reconstruction, the reconstruction without the gradient term L_{grad} in the loss, the reconstruction with the full loss but a UNET instead of an Attention UNET.

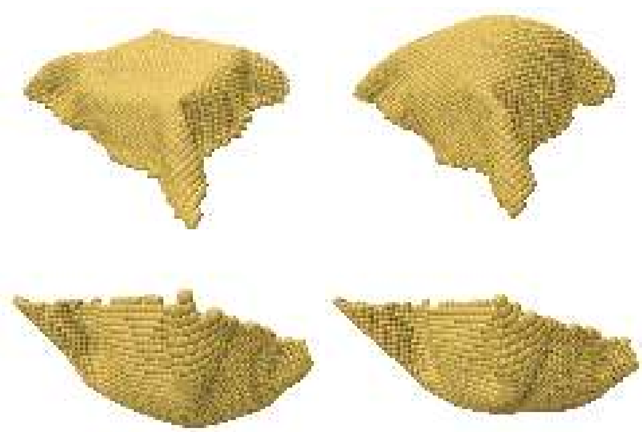

Fig. 12. The first column shows the predictions of the network trained without random rotation during training while the second column shows the results with those rotations.

4.2. Ablation study

This subsection highlights the role of the choices made in our approach. Firstly, Fig 11 demonstrates the significance of the gradient term in the loss, particularly for the surface regularity. The figure also illustrates the importance of the attention mechanism by showing the result with a simple UNET. The attention connection enables the connection between different local features across the surface. Intuitively, a fold is more likely to fade in the inpainted region without attention because it does not have information that the fold continues across the missing area.

Besides, the dataset was trained on a simulator that includes gravity. This means that the network is biased by the orientation of the input mesh. We then trained the network on the original dataset and another version which includes random rotation. Fig 12 shows the influence of both ways of training the network.

5. Conclusion

In this paper, we propose a method to fill holes in fabric meshes using a learning approach. We leverage the developable nature of fabrics by feeding a meaningful parametrization of the surface and its holes to a 2D CNN to get around the difficulties of 3D networks. The network is trained on a new *ScarfFolds* dataset that we generated to capture a variety of folds on square meshes. The method comes along with some editing operations such as (grabbing, pinching, twisting) in an interactive way. Since the approach is data-driven, it has a strong prior on the surface type and is more robust to noise in the input data compared to pure geometric methods. As future works, it would be interesting to explore a meshless approach to the problem while still being able to use 2D CNNs. That way, it would be possible to handle raw scanned data, after solving the problem of parametrizing and flattening the holes. It would also be interesting to introduce regularizing functions that could be optimized by auto-differentiation, in the same way as we were able to optimize the position of an anchor point. Also, the mesh is only required for the parametrization, so if a scan is conducted with markers, the parametrization may not be necessary.

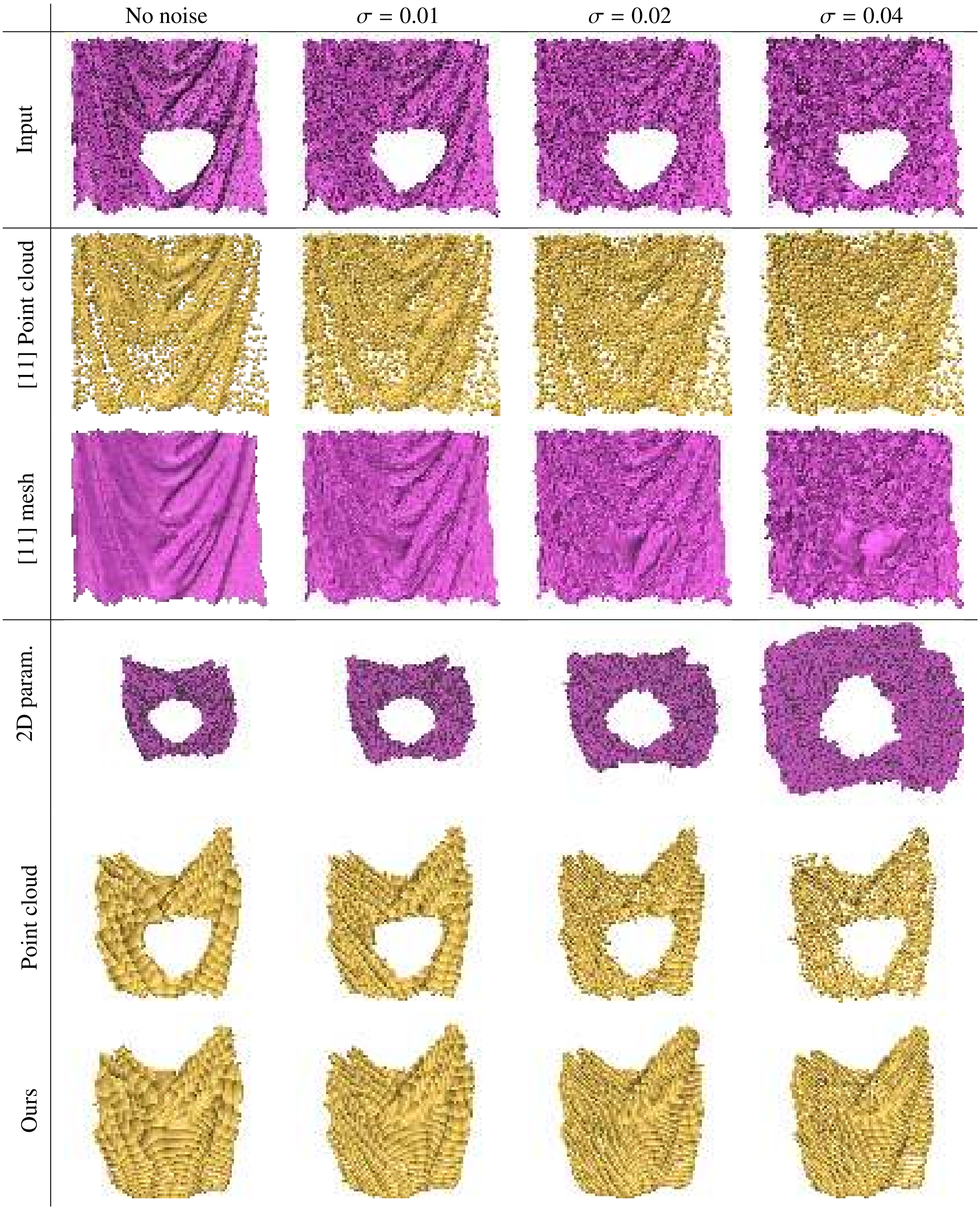

Fig. 13. Illustration of the stability of our approach with respect to noise: The first row represents the input mesh, the second row the parametrization of the region around the hole, the third row the input of the network represented as a point cloud, the fourth row the result of Gisbert et al. [11]. Finally, the last row corresponds to our reconstruction.

6. Acknowledgements

This work is part of the e-Roma project from the French Agence Nationale de la Recherche (ANR-16-CE38-0009). It was funded, in whole or in part, by l'Agence Nationale de la Recherche (ANR), project StableProxies ANR-22-CE46-0006. For the purpose of open access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version.

References

[1] Thomas, H, Qi, CR, Deschaud, JE, Marcotegui, B, Goulette, F, Guibas, L. Kpconv: Flexible and deformable convolution for point clouds. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). 2019, p. 6410–6419. doi:10.1109/ICCV.2019.00651.

[2] Sharp, N, Attaiki, S, Crane, K, Ovsjanikov, M. Diffusionnet: Discretization agnostic learning on surfaces. ACM Trans Graph 2022;01(1).

[3] Hanocka, R, Hertz, A, Fish, N, Giryes, R, Fleishman, S, Cohen-Or, D. Meshcnn: a network with an edge. ACM Trans Graph 2019;38(4). doi:10.1145/3306346.3322959.

[4] Liepa, P. Filling Holes in Meshes. In: Kobbelt, L, Schroeder, P, Hoppe, H, editors. Eurographics Symposium on Geometry Processing. The Eurographics Association. ISBN 3-905673-06-1; 2003,doi:10.2312/SGP/SGP03/200-206.

[5] Baumgärtner, L, Bergmann, R, Herrmann, M, Herzog, R, Schmidt, S, Vidal, J. Mesh denoising and inpainting using the total variation of the normal. 2020.

[6] Bonneel, N, Coeurjolly, D, Gueth, P, Lachaud, JO. Mumford-shah mesh processing using the ambrosio-tortorelli functional. Computer Graphics Forum (Proceedings of Pacific Graphics) 2018;37(7). doi:10.1111/cgf.13549.

[7] Feng, C, Liang, J, Ren, M, Qiao, G, Lu, W, Liu, S. A fast hole-filling method for triangular mesh in additive repair. Applied Sciences 2020;10(3). doi:10.3390/app10030969.

[8] Attene, M. A lightweight approach to repairing digitized polygon meshes. The Visual Computer 2010;26:1393–1406. doi:10.1007/s00371-010-0416-3.

[9] Brunton, A, Wuhrer, S, Shu, C, Bose, P, Demaine, E. Filling holes in triangular meshes by curve unfolding. 2009, p. 66 – 72. doi:10.1109/SMI.2009.5170165.

[10] Harary, G, Tal, A, Grinspun, E. Context-based coherent surface completion. ACM Trans Graph 2014;33(1). doi:10.1145/2532548.

[11] Gisbert, G, Chaine, R, Coeurjolly, D. Inpainting holes in folded fabric meshes. Computers & Graphics 2023;114:201–209. doi:10.1016/j.cag.2023.05.025.

[12] Davis, J, Marschner, S, Garr, M, Levoy, M. Filling holes in complex surfaces using volumetric diffusion. In: Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission. 2002, p. 428–441. doi:10.1109/TDPVT.2002.1024098.

[13] Kazhdan, M, Bolitho, M, Hoppe, H. Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing; vol. 7. 2006, p. 0.

[14] Kazhdan, M, Hoppe, H. Screened poisson surface reconstruction. ACM Trans Graph 2013;32(3). doi:10.1145/2487228.2487237.

[15] Groueix, T, Fisher, M, Kim, VG, Russell, B, Aubry, M. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). 2018,.

[16] Deprelle, T, Groueix, T, Fisher, M, Kim, VG, Russell, BC, Aubry, M. Learning elementary structures for 3d shape generation and matching. arXiv preprint arXiv:190804725 2019;.

[17] Bednarik, J, Parashar, S, Gundogdu, E, Salzmann, M, Fua, P. Shape reconstruction by learning differentiable surface representations. 2020, p. 4715–4724. doi:10.1109/CVPR42600.2020.00477.

[18] Yang, Y, Feng, C, Shen, Y, Tian, D. Foldingnet: Point cloud auto-encoder via deep grid deformation. 2017. URL: http://arxiv.org/abs/1712.07262; cite arxiv:1712.07262Comment: Accepted as a spotlight paper in CVPR'18.

[19] Hanocka, R, Metzer, G, Giryes, R, Cohen-Or, D. Point2mesh: a self-prior for deformable meshes. ACM Trans Graph 2020;39(4). doi:10.1145/3386569.3392415.

[20] Liu, G, Reda, FA, Shih, KJ, Wang, TC, Tao, A, Catanzaro, B. Image inpainting for irregular holes using partial convolutions. 2018. arXiv:1804.07723.

[21] Suvorov, R, Logacheva, E, Mashikhin, A, Remizova, A, Ashukha, A, Silvestrov, A, et al. Resolution-robust large mask inpainting with fourier convolutions. 2021. arXiv:2109.07161.

[22] Deng, Y, Hui, S, Zhou, S, Meng, D, Wang, J. T-former: An efficient transformer for image inpainting. In: Proceedings of the 30th ACM International Conference on Multimedia. MM '22; ACM; 2022,doi:10.1145/3503161.3548446.

[23] Dong, Q, Cao, C, Fu, Y. Incremental transformer structure enhanced image inpainting with masking positional encoding. 2022. arXiv:2203.00867.

[24] Wang, Y, Yu, J, Zhang, J. Zero-shot image restoration using denoising diffusion null-space model. 2022. arXiv:2212.00490.

[25] Lugmayr, A, Danelljan, M, Romero, A, Yu, F, Timofte, R, Gool, LV. Repaint: Inpainting using denoising diffusion probabilistic models. 2022. arXiv:2201.09865.

[26] Xiang, H, Zou, Q, Nawaz, MA, Huang, X, Zhang, F, Yu, H. Deep learning for image inpainting: A survey. Pattern Recognition 2023;134:109046. URL: https://www.sciencedirect.com/science/article/pii/S003132032200526X. doi:https://doi.org/10.1016/j.patcog.2022.109046.

[27] Lamb, N, Banerjee, S, Banerjee, NK. Deepmend: Learning occupancy functions to represent shape for repair. 2022. arXiv:2210.05728.

[28] Sharma, A, Grau, O, Fritz, M. Vconv-dae: Deep volumetric shape learning without object labels. In: Hua, G, Jégou, H, editors. Computer Vision – ECCV 2016 Workshops. Cham: Springer International Publishing. ISBN 978-3-319-49409-8; 2016, p. 236–250.

[29] Smith, EJ, Meger, D. Improved adversarial systems for 3d object generation and reconstruction. In: Conference on Robot Learning. 2017, p. 87–96.

[30] Wu, J, Zhang, C, Xue, T, Freeman, B, Tenenbaum, J. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Lee, D, Sugiyama, M, Luxburg, U, Guyon, I, Garnett, R, editors. Advances in Neural Information Processing Systems; vol. 29. Curran Associates, Inc.; 2016,.

[31] Han, X, Li, Z, Haibin, H, Kalogerakis, E, Yu, Y. High-resolution shape completion using deep neural networks for global structure and local geometry inference. 2017,doi:10.1109/ICCV.2017.19.

[32] Wen, X, Li, T, Han, Z, Liu, YS. Point cloud completion by skip-attention network with hierarchical folding. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2020;:1936–1945.

[33] Oktay, O, Schlemper, J, Folgoc, LL, Lee, M, Heinrich, M, Misawa, K, et al. Attention u-net: Learning where to look for the pancreas. 2018. arXiv:1804.03999.

[34] Qi, CR, Yi, L, Su, H, Guibas, LJ. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17; Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964; 2017, p. 5105–5114.

[35] Hattori, S, Yatagawa, T, Ohtake, Y, Suzuki, H. Deep mesh prior: Unsupervised mesh restoration using graph convolutional networks. 2021. arXiv:2107.02909.

[36] Sarmad, M, Lee, HJ, Kim, Y. Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2019;:5891–5900.

[37] Gundogdu, E, Constantin, V, Seifoddini, A, Dang, M, Salzmann, M, Fua, P. Garnet: A two-stream network for fast and accurate 3d cloth draping. In: IEEE International Conference on Computer Vision (ICCV). IEEE; 2019,.

[38] Gundogdu, E, Constantin, V, Parashar, S, Seifoddini, A, Dang, M, Salzmann, M, et al. Garnet++: Improving fast and accurate static3d cloth draping by curvature loss. CoRR 2020;abs/2007.10867. arXiv:2007.10867.

[39] De Luigi, L, Li, R, Guillard, B, Salzmann, M, Fua, P. DrapeNet: Garment Generation and Self-Supervised Draping. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023,.

[40] Charles, RQ, Su, H, Kaichun, M, Guibas, LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017, p. 77–85. doi:10.1109/CVPR.2017.16.

[41] Bertiche, H, Madadi, M, Escalera, S. Neural cloth simulation. ACM Transactions on Graphics 2022;41(6):1–14. doi:10.1145/3550454.3555491.

[42] Hernández-Bautista, M, Melero, FJ. Deep learning of curvature features for shape completion. Computers & Graphics 2023;115:204–215. doi:10.1016/j.cag.2023.07.007.

[43] Sorkine, O, Alexa, M. As-rigid-as-possible surface modeling. In: Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing. 2007, p. 109–116.

[44] Liu, L, Zhang, L, Xu, Y, Gotsman, C, Gortler, SJ. A local/global approach to mesh parameterization. In: Proceedings of the Symposium on Geometry Processing. SGP '08; Goslar, DEU: Eurographics Association; 2008, p. 1495–1504.

[45] Community, BO. Blender - a 3D modelling and rendering package. Blender Foundation; Stichting Blender Foundation, Amsterdam; 2018. URL: http://www.blender.org.

[46] Paszke, A, Gross, S, Massa, F, Lerer, A, Bradbury, J, Chanan, G, et al. Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32. Curran

Associates, Inc.; 2019, p. 8024–8035.

[47] Jacobson, A, Panozzo, D, et al. libigl: A simple C++ geometry processing library. 2018. Https://libigl.github.io/.

[48] Chang, AX, Funkhouser, TA, Guibas, LJ, Hanrahan, P, Huang, Q, Li, Z, et al. Shapenet: An information-rich 3d model repository. CoRR 2015;abs/1512.03012. URL: http://arxiv.org/abs/1512.03012. arXiv:1512.03012.

[49] Lévy, B, Petitjean, S, Ray, N, Maillot, J. Least Squares Conformal Maps for Automatic Texture Atlas Generation. ACM Transactions on Graphics 2002;21(3):10 p. URL: https://inria.hal.science/inria-00100754. doi:10.1145/566654.566590; special issue : Proceedings of ACM SIGGRAPH 2002.