Computer Aided Geometric Design 72 (2019) 84–97

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Preventing self-intersection with cycle regularization in neural networks for mesh reconstruction from a single RGB image ☆

Siyu Hu, Xuejin Chen *

CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System, University of Science and Technology of China, China

ARTICLE INFO

Article history:
Available online 13 June 2019

Keywords:
Self-intersection
Cycle regularization
Mesh reconstruction
Neural network

ABSTRACT

Self-intersection in surfaces is a typical defect that makes a 3D model unsuitable for many applications. Existing neural networks for 3D surface mesh reconstruction are faced with the challenge of integrating self-intersection prevention. In this paper, we propose a trainable cycle regularization in mesh reconstruction networks to prevent self-intersection. It is a general technique that can be easily implemented with existing surface mesh generation networks. Our experiments on two latest mesh reconstruction networks demonstrate that with the proposed cycle regularization, self-intersections in the generated meshes are significantly reduced, while the shape similarity is comparable with the original networks under the Chamfer distance metric.

1. Introduction

Inferring 3D shape from a single view image is a traditional problem for computer vision. In computer graphics, 3D modeling from a given image has also been extensively studied. In recent years, deep neural networks Choy et al. (2016); Fan et al. (2017); Kato et al. (2017); Kar et al. (2015); Wu et al. (2015); Dou et al. (2017); Tatarchenko et al. (2017); Sinha et al. (2017); Wu et al. (2018) have achieved great success in this field. Unlike classic shape from X (e.g. Horn (1975); Rhodin and Breuß (2013); Criminisi and Zisserman (2000)) approaches, these neural networks are able to recover not only the visible frontal shape but also the invisible part for an object from a single-view color image by learning and representing complicated prior knowledge from a large dataset.

Existing networks all rely on variants of 2D convolution neural networks to extract information and encode 2D images, but use quite different techniques to represent and decode 3D shapes. Started from 3D ShapeNets Wu et al. (2015) and greatly improved by introducing octree structure Tatarchenko et al. (2017), volumetric representation and 3D convolutional networks are most commonly used in this problem. A point set generation network Fan et al. (2017) uses unordered point set representation and directly regresses a point set using both convolutional and fully connected branches. Dou et al. (2017) employ a bilinear model to represent 3D faces and regress the interpolation coefficients to generate a 3D face model from one single image. Sinha et al. (2017) explicitly use spherical parameterization as a post-processing stage to represent a 3D shape as a geometry image in the parameter domain.

☆ Editor: Jiri Kosinka.

* Corresponding author.
E-mail addresses: sy891228@mail.ustc.edu.cn (S. Hu), xjchen99@ustc.edu.cn (X. Chen).

https://doi.org/10.1016/j.cagd.2019.06.001
0167-8396/

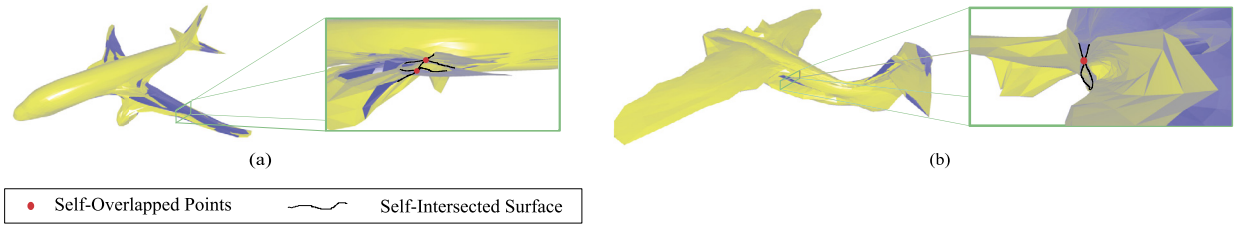

Fig. 1. Self-intersections in 3D surface mesh reconstruction networks. (a) A surface mesh of a plane generated by AtlasNet Groueix et al. (2018) (sphere as its parameter domain). (b) A surface mesh of a plane generated by Pixel2Mesh Wang et al. (2018). Outside faces are rendered as golden and inside faces are rendered as bluish. Some inside triangles are exposed due to self-intersections. We highlight some self-intersected faces and their corresponding overlapped points in the close-up view. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

In recent study, AtlasNet (Groueix et al. (2018)) and Pixel2Mesh (Wang et al. (2018)) to be specific, a new idea has been applied on this problem. Neural networks are designed to learn the mapping from a predefined surface (square or sphere for AtlasNet, and ellipsoid for Pixel2Mesh) to a target surface instead of directly regressing the absolute positions of surface points as in Fan et al. (2017). These methods have shown great potential in generating meshes for generic objects. It is convenient to integrate mesh-related operations and energy functions in these networks. For example, Pixel2Mesh integrates graph-based unpooling and Laplacian regularization in the network.

Despite all of these progresses, there are still a number of issues such as how to encode and generate more shape details, how to express complex topology for mesh in neural networks. These issues prevent network-generated meshes from being practicable in many real-world applications. However, we believe deep learning is the most promising technique to integrate more intelligent methods in 3D modeling.

In this paper, we address a specific issue that appears in both AtlasNet Groueix et al. (2018) and Pixel2Mesh Wang et al. (2018). As shown in Fig. 1, the AtlasNet and Pixel2Mesh frequently generate meshes with self-intersected surfaces. This issue appears partially because AtlasNet and Pixel2Mesh employ the Chamfer distance loss, which is used firstly to train the point set generation network (PSGN, Fan et al. (2017)). The Chamfer distance loss was designed to measure the discrepancy between two unordered point sets and it does not take surface into consideration. AtlasNet uses Poisson surface reconstruction as post-processing or double-sided lighting in rendering to cover up this issue. Pixel2Mesh adopts a coarse-to-fine framework and uses additional losses (i.e. edge length loss and Laplacian loss) in order to alleviate this issue. They all fail to address the essential reason behind this issue, while most efforts in these mesh generation networks have been put on increasing shape details for the generated meshes.

In this paper, we tackle the issue of self-intersection from the essential reason behind it, which is non-injectivity of the predicted mapping. Without injectivity, two points on the predefined source surface could be mapped to the same point by the neural network, which leads self-intersection or self-overlapping in the generated surface.

To enforce injectivity, one possible strategy is to start from a feasible solution and keep every deformation or optimization step inside feasible regions. In works of deformation (e.g. Sederberg and Parry (1986); Gain and Dodgson (2001)), such strategy is usually executed as follows. A clean mesh that is free from self-intersection is chosen as the initial mesh and local self-intersection is prevented by constraining the Jacobian of the mapping function in the subsequent steps in deformation. In works of parameterization optimization for surface with disk topology, Tutte's embedding Tutte (1963) or its variants are typically employed to get an initial bijective mapping and triangle fold (local self-intersection) is prevented with different techniques in follow-up optimization steps. More specifically, triangle fold can be prevented by adding barrier energy from distortion metrics (e.g. Poranne and Lipman (2014); Aigerman et al. (2014)), bounding the triangle distortion (e.g. Smith and Schaefer (2015); Lipman (2012)) or using a progressive strategy Liu et al. (2018).

It is non-trivial to adopt their strategies for training neural networks, since existing networks learn to predict the mapping for many different shapes simultaneously and only a batch of these shapes are sampled from the dataset in each training iteration. One challenge is to initialize the network parameters to ensure that initial outputs are free of self-intersection for all possible inputs. Another challenge is to alter batch-based optimizer to constrain the deformation of outputs within an injective region for all possible inputs.

In order to learn injective mapping for meshes, we propose cycle regularization which is deduced from the basic decision theorem of injectivity. It reduces not only local self-intersections but also global self-interferences of the surface. It is easy to implement by reusing the existing differentiable layers within existing surface mesh reconstruction networks, such as AtlasNet Groueix et al. (2018) and Pixel2Mesh Wang et al. (2018).

Our strategy is to use an inverse 3D decoder to learn an inverse mapping from the target surface back to the source surface along with the forward mapping in the original network. Therefore, a point from the source surface can be mapped to the target surface and then mapped back. We use the difference after such cycle mapping to form our regularization term and we call it cycle regularization. While the network learning a mapping to approximate the target surface, our regularization term tries to ensure that an inverse mapping exists (i.e. making the forward mapping injective, as we explain in Sec. 3.2). Note that the inverse 3D decoder is only needed in the training phase. Therefore, it is a regularization technique which does not increase the complexity of the original neural network for mesh generation.

In summary, our contributions in this paper are three-fold.

- We propose a novel cycle regularization technique to prevent self-intersection for surface mesh reconstruction networks.
- We apply our cycle regularization technique on two latest mesh generation networks, AtlasNet Groueix et al. (2018) and Pixel2Mesh Wang et al. (2018), showing that our technique keeps the network end-to-end trainable by using existing differentiable layers.
- We validate with experiments that when trained with the proposed cycle regularization, these networks are able to produce surface meshes with significantly less self-intersections, and still lead to comparable Chamfer distance between the generated mesh and the ground-truth mesh compared with the original networks.

2. Related work

3D reconstruction and modeling from a single image have been extensively studied as the problem of *shape from monocular cues*, including shadings Zhang et al. (1999), focuses Favaro et al. (2008); Favaro and Soatto (2005), and textures Aloimonos (1988). These methods usually recover 2.5D surfaces from 2D images. Learning-based approaches, especially deep learning methods, can acquire more complicated priors by learning from datasets and recover much more complete 3D shape from a single image.

2.1. General learning approaches

As far as we know, early learning-based approaches can be traced back to Hoiem et al. (2007) and Saxena et al. (2007). These methods learn to segment and classify regions in an image and finally produce a coarse 3D scene by folding the 2D image. More recent techniques divide the problem into two stages Su et al. (2014); Huang et al. (2015). They first retrieve shape components from a large dataset, and then assemble the components and deform the assembled shape to fit the observed image. These methods need to segment all object models into components for the database. However, shape retrieval from images itself is an challenging problem due to the loss of information during 3D-to-2D projection. Kar et al. (2015) avoid the retrieval step by learning a deformable 3D shape for each category and learn to predict deformation from an input image for these specific categories.

2.2. 3D neural networks

Most recently, researchers have developed techniques to represent 3D shapes inside deep learning frameworks. Unlike images, 3D shapes are not canonical functions on well-organized grids. This leads to exploration on various representations of 3D shapes.

Volume Occupancy. An intuitive way to apply convolutional layers in 3D is to use volume occupancy of regular 3D grids to represent 3D shapes Wu et al. (2015). It is subsequently used for 3D shape generation Choy et al. (2016); Girdhar et al. (2016). The main disadvantage of volumetric representation was its large memory consumption due to the raising of dimension when extending 2D grids to 3D. Octree representation is proposed to support higher resolution outputs with limited memory, and used for shape generation Tatarchenko et al. (2017) and shape analysis Wang et al. (2017).

Point Cloud. Compared to regular 3D grids, point cloud representation is not limited by fixed local connections. Many networks have been proposed to take unordered 3D point sets as input and extract geometric features from a 3D point set for classification or segmentation Charles et al. (2017); Qi et al. (2017); Li et al. (2018). The first attempt to generate a set of discrete points from a single image using neural networks was made by Fan et al. (2017). However, it is non-trivial to construct continuous surface models from the predicted point sets, since the local variations of point positions are not continuous in the predicted point sets.

Mesh. Meshes are widely used in game and movie industries. In addition to vertex positions, mesh representation conveys local structures of vertices. However, mesh representation is not well supported in current neural networks. To generate mesh models using neural networks, composition weights of a series of base meshes are predicted by networks Pontes et al. (2017) and Dou et al. (2017). Since it is only possible to choose or learn base meshes for a specific class of objects, these two networks only generate meshes for a specific class of objects, such as face. In comparison, the idea of learning to map from a predefined domain as in AtlasNet Groueix et al. (2018) and Pixel2Mesh Wang et al. (2018), can generate surface meshes for generic objects. Our work is a follow-up of their general idea and addresses a specific issue of surface self-intersection in their networks.

2.3. Parameterization for neural networks

The idea of utilizing surface parameterization for neural networks has been explored by Sinha et al. (2017, 2016). Typically, a non-trainable procedure is involved for the creation of geometry images. Manifold surfaces are required as training data so that the shapes can be parameterized using spherical parameterization and turned into geometry images. However, public datasets like ShapeNet Chang et al. (2015) contain meshes that are not manifold surfaces. In comparison, we are

seeking techniques that can be integrated into networks and can be trained end-to-end along with the networks. Based on the same insight (self-intersection for surface mesh is related to non-injective mapping) as the parameterization techniques (e.g. Lipman (2012); Schüller et al. (2013); Poranne and Lipman (2014); Aigerman et al. (2014); Smith and Schaefer (2015); Liu et al. (2018)), we propose a novel technique that is more suitable for training neural networks in an end-to-end manner.

2.4. Cycle neural networks

The general idea of using neural networks to map from one domain to another domain and then map back has been utilized in previous works. In CycleGAN Zhu et al. (2017), a famous example of such work, the cycle relation provides an extra constraint for translation between unpaired data from different domains. In comparison, our work uses the same general idea to enforce injectivity for 3D surface mesh generation networks and prevent self-intersections in generated meshes.

2.5. Non-learning based self-intersection removal

Non-learning based methods (e.g. Jung et al. (2004); Pekerman et al. (2008); Yamakawa and Shimada (2009); Li and Barbič (2018)) for removing self-intersections follow the pipeline that first identifies the self-intersected faces and then alters the faces with their proposed methods. However, it is difficult to integrate such a pipeline into neural networks or to formulate their alteration operations in a differentiable manner.

3. Proposed method

In this section, we first establish the relationship between self-intersection and non-injectivity. Then we introduce the proposed cycle regularization technique as shown in Fig. 3 and explain in details about how we respectively apply this general technique onto AtlasNet Groueix et al. (2018) and Pixel2Mesh Wang et al. (2018), whose network structures are quite different from each other.

3.1. Injective mapping and self-overlapped points

Definition 1. Let f be a function whose domain is a set X. The function f is said to be injective provided that

$$\forall \mathbf{a}, \mathbf{b} \in X,\ f(\mathbf{a}) = f(\mathbf{b}) \Rightarrow \mathbf{a} = \mathbf{b}. \tag{1}$$

Equivalently,

$$\forall \mathbf{a}, \mathbf{b} \in X, \mathbf{a} \neq \mathbf{b} \Rightarrow f(\mathbf{a}) \neq f(\mathbf{b}). \tag{2}$$

Start with the definition of injective mapping at Definition 1, we can intuitively induce the conclusion that given a surface with $X = \{\mathbf{x} \mid \mathbf{x}$ is a point on the surface$\}$, another surface with $Y = \{\mathbf{y} \mid \mathbf{y}$ is a point on the surface$\}$ and a function $f : X \to Y$. If $\exists\ \mathbf{a}, \mathbf{b} \in X$, $\mathbf{a} \neq \mathbf{b}$ and $f(\mathbf{a}) = f(\mathbf{b})$ (i.e. the overlapped points on the target surface exists) then by definition, f is not an injective function. Equivalently (as the converse negative proposition), we can make sure there is no self-overlapped points on the target surface by enforcing injectivity for f.

3.2. Cycle regularization

Theorem 1. *A function f with a left inverse is necessarily injective. That is, given $f : X \to Y$, if there is a function $g : Y \to X$ such that,*

$$\forall \mathbf{x} \in X,\ g(f(\mathbf{x})) = \mathbf{x}, \tag{3}$$

then f is injective.

Based on Theorem 1, we turn an injective constraint to our cycle regularization term as:

$$cycle_X(f, g) = \sum_{\mathbf{x} \in X} ||g(f(\mathbf{x})) - \mathbf{x}||^2. \tag{4}$$

In mesh reconstruction networks, this term (Eq. (4)) is illustrated in Fig. 2. As shown in Fig. 2, points from the source surface (as the red point) is mapped to image points (as the green point) on the target surface by a 3D decoder f to generate a shape of plane. These image points are then mapped back to points on the source surface (as the blue point) by the inverse decoder g. We are minimizing the difference after such cycle mapping (as the distance between red point and blue point) to ensure that f has an left inverse mapping g and enforce injectivity for f.

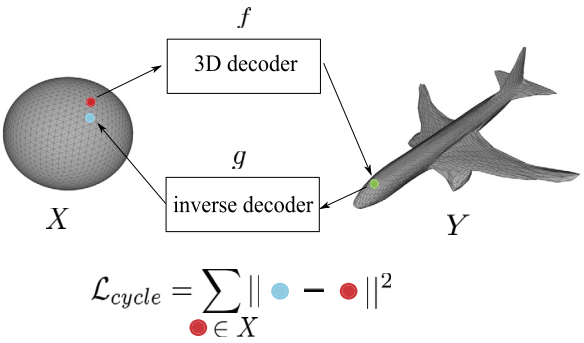

Fig. 2. Illustration of our cycle regularization.

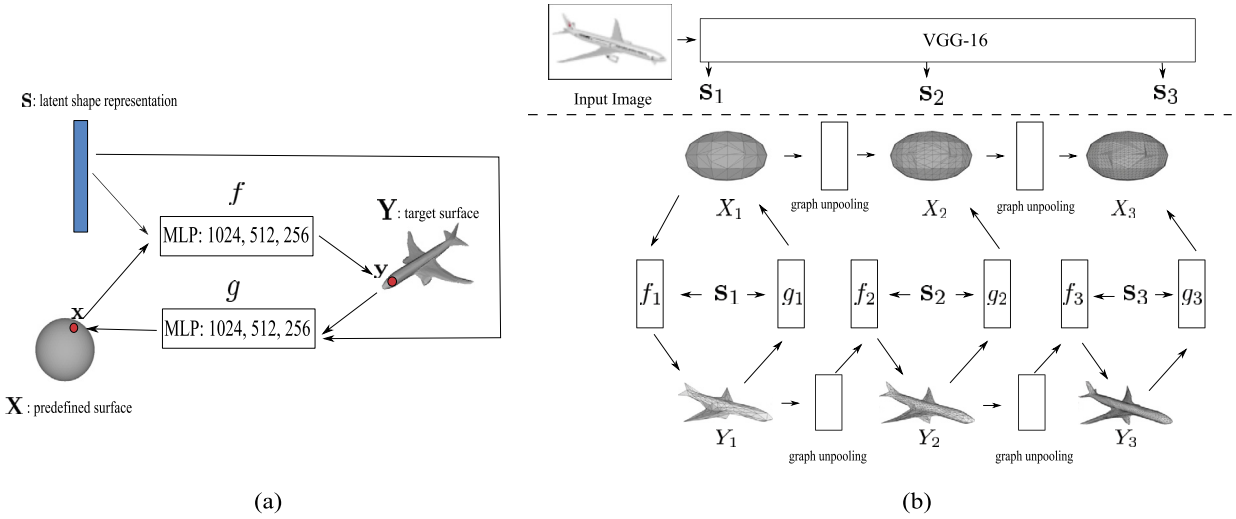

Fig. 3. The cycle regularization implemented along with two networks. (a) is the implementation with AtlasNet Groueix et al. (2018). f is the forward 3D surface decoder in the original network and g is our inverse decoder used to form the regularization term. (b) is the implementation with Pixel2Mesh Wang et al. (2018). Pixel2Mesh Wang et al. (2018) adopts the coarse-to-fine framework and uses three G-ResNet blocks (f_1, f_2, f_3) to map the mesh to target shape on three different point density. The graph-unpooling layers are used for mesh upsampling. We use three point-wise MLP (g_1, g_2, g_3) as the inverse decoders for each level of point density and form a regularization term for each level.

By minimizing this term to zero:

$$\hat{f}, \hat{g} = \arg\min_{(f,g)} cycle_X(f, g), \tag{5}$$

we can get the $\hat{f}$ that has the left inverse function $\hat{g}$, therefore $\hat{f}$ is injective.

However, it is almost impossible to actually minimize a regularization term to zero, especially during training neural networks. But when this term is minimized to sufficiently small, we can construct a g^* that is the left inverse function of $\hat{f}$ and guarantee that $\hat{f}$ is injective. An example of such case is that when the regularization term is so small that for any $\mathbf{x}$, $\hat{g}(\hat{f}(\mathbf{x}))$ is closer to $\mathbf{x}$ than any other point in X. Such example can be summarized by Proposition 1.

Proposition 1. *Given sets X and Y that are two subsets of Euclidean space $\mathcal{R}^3$, a function $f : X \to Y$ and a function $g : Y \to X$, if*

$$\exists\, g, \forall \mathbf{x} \in X, ||g(f(\mathbf{x})) - \mathbf{x}||^2 \leq \min_{\mathbf{b} \in X} ||g(f(\mathbf{x})) - \mathbf{b}||^2, \tag{6}$$

then f is injective. $\mathbf{x}$ and $\mathbf{b}$ are both used to represent elements in X.

Proposition 1 can be proved by simply compositing the nearest neighbors with the function g. We can construct a nearest neighbor function $q : \mathcal{R}^3 \to X$ as:

$$\forall \mathbf{a} \in \mathcal{R}^3, q(\mathbf{a}) = \arg\min_{\mathbf{b} \in X} ||\mathbf{a} - \mathbf{b}||^2 \tag{7}$$

then

$$\begin{aligned} &||g(f(\mathbf{x})) - \mathbf{x}||^2 \leq \min_{\mathbf{b}\in X} ||g(f(\mathbf{x})) - \mathbf{b}||^2 \\ &\Rightarrow \mathbf{x} = \arg\min_{\mathbf{b}\in X} ||g(f(\mathbf{x})) - \mathbf{b}||^2 \\ &\Rightarrow q\Big(g\big(f(\mathbf{x})\big)\Big) = \mathbf{x} \end{aligned} \tag{8}$$

then $g^*(\mathbf{y}) = q\big(g(\mathbf{y})\big)$ is the left inverse of f, therefore f is injective. Here, $\mathbf{y}$ is used to represent an arbitrary element in Y.

Remaining Gap. Even when the condition in Proposition 1 is met after optimization, there is still no guarantee for the injectivity of f over a continuous surface. The reason is that when we turn Eq. (3) into Eq. (4), a theoretical gap exits. The constraint in Eq. (3) is defined over continuous surfaces, while the term in Eq. (4) is defined on discrete point sets. In order to fill in the gap as much as we can, we randomly sample X from a predefined surface in the training phase instead of minimizing the cycle regularization for any specific set of X. Sampling different X from the predefined surface is already employed by AtlasNet. This is a commonly used technique to encode latent variation in generation networks since VAE Kingma and Welling (2013). For Pixel2Mesh, we add such a sampling in the inverse decoding. We did not add it in forward 3D decoding to avoid altering the original forward inference path for Pixel2Mesh. We want to keep our cycle regularization general in the way that it can work along with surface mesh generation networks without altering the original network. Our controlled experiment validates that it is better for our cycle regularization technique to work with such a sampling.

3.3. Implementation in networks

As a simple inference from the *Universal Approximation Theorem*, AtlasNet Groueix et al. (2018) has stated that it is possible to use a multilayer perceptron with ReLU nonlinearities and sufficient hidden units to approximate any shape within a small positive error ϵ. In practice, we employ another 3D surface decoder to approximate g. Then we explain how we implement this technique for AtlasNet and Pixel2Mesh respectively. Generally speaking, we reuse their network structures respectively and show that our cycle regularization is generally suitable for this type of networks.

Cycle-AtlasNet. Depending on a shape representing feature $\mathbf{s}$, AtlasNet uses point-wise MLP f with parameters θ_f to learn to map points in $X = \{\mathbf{x}|\mathbf{x}$ are points uniformly sampled from surface $P\}$ to points in $Y = \{\mathbf{y}|\mathbf{y}$ are points uniformly sampled from surface $S\}$. In our implementation, P is a spherical surface. S is the generated surface for an object. We use another point-wise MLP g with parameters θ_g to map points from Y back to X. Along with our cycle regularization, the total loss function for Cycle-AtlasNet is:

$$\begin{aligned} \mathcal{L}_{(X,Y)}(\theta_f, \theta_g) = &\sum_{\mathbf{x}\in X} \min_{\mathbf{y}\in Y} ||f(\mathbf{x}; \mathbf{s}, \theta_f) - \mathbf{y}||^2 \\ &+ \sum_{\mathbf{y}\in Y} \min_{\mathbf{x}\in X} ||f(\mathbf{x}; \mathbf{s}, \theta_f) - \mathbf{y}||^2 \\ &+ \lambda \sum_{\mathbf{x}\in X} ||g\big(f(\mathbf{x}; \mathbf{s}, \theta_g); \mathbf{s}, \theta_f\big) - \mathbf{x}||^2, \end{aligned} \tag{9}$$

where the shape representation feature is simply concatenated to each point so that f and g depend on a global shape feature $\mathbf{s}$. λ is the weight for cycle regularization term. In AtlasNet, $\mathbf{s}$ is generated from either PointNet He et al. (2016) for auto-encoding or ResNet-18 He et al. (2016) for single view reconstruction. g is a MLP with the same number of units in hidden layers as f, as shown in Fig. 3.

Cycle-Pixel2Mesh. Comparing to AtlasNet, Pixel2Mesh uses a more complicated network structure which generates shapes in a coarse-to-fine framework. As shown in Fig. 3, Pixel2Mesh uses three blocks of graph-based convolutional residual network (f_1, f_2, f_3) to map mesh-based features to the generated surface at three different levels of point density. To make cycle regularization compatible with such a framework, we also use three point-wise MLPs as the inverse decoder and form our regularization term as:

$$\begin{aligned} \mathcal{L}_{cycle}(\theta_f, \theta_g) = &\sum_{\hat{\mathbf{x}}\in\hat{X}_1} ||g_1(\hat{\mathbf{y}_1}; \mathbf{s}_1, \theta_{g_1}) - \hat{\mathbf{x}}||^2 \\ &+ \sum_{\hat{\mathbf{x}}\in\hat{X}_2} ||g_2(\hat{\mathbf{y}_2}; \mathbf{s}_2, \theta_{g_2}) - \hat{\mathbf{x}}||^2 \\ &+ \sum_{\hat{\mathbf{x}}\in\hat{X}_3} ||g_3(\hat{\mathbf{y}_3}; \mathbf{s}_3, \theta_{g_3}) - \hat{\mathbf{x}}||^2, \end{aligned} \tag{10}$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}_l$ are sampled as a convex combination of vertexes in each triangle as:

$$\begin{aligned}\hat{\mathbf{x}} &= \sum_{n=1}^{3} w_n \mathbf{x}_n,\\ \hat{\mathbf{y}}_l &= \sum_{n=1}^{3} w_n f_l(\mathbf{x}_n; \mathbf{s}_l, \theta_{f_l}),\\ \sum_{n=1}^{N=3} w_n &= 1.\end{aligned} \tag{11}$$

The combination coefficients w_n are randomly generated but kept the same for corresponding $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}_l$ in each level l. θ_f is the whole parameter sets for f_1, f_2, f_3. θ_g is the whole parameter sets for g_1, g_2, g_3. $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ are features extracted from different layers of VGG-16 based on the input point positions X_1, X_2, X_3 in the network. $\hat{X}_1, \hat{X}_2, \hat{X}_3$ are the sets of sampled points at each level. As mentioned in Sec. 3.2, random sampling from predefined surface is crucial for our technique. Since Pixel2Mesh does not include such sampling in their method, we add it in our inverse decoding. In other words, in our inverse decoding we do not directly map the vertices from the target surface back to the predefined surface. Instead, we map the sampled points $\hat{\mathbf{y}}_l$ from each face back to the corresponding samples $\hat{\mathbf{x}}$. The correspondence between $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}_l$ are ensured to be locally injective by using the same set of convex combination coefficients w_n for the sampling. We multiply our regularization term with weight λ and add it to the original loss function of Pixel2Mesh (see Wang et al. (2018) for the original loss function). For more details of our implementation, please refer to our supplemental material[1] for the code.

4. Experiments and discussions

Data. To fairly evaluate the effect of the proposed cycle regularization, we use the datasets released by AtlasNet and Pixel2Mesh respectively. Their model sets are both subsets of ShapeNet Chang et al. (2015) and they both used the rendered images from Choy et al. (2016) in dimensions of 224×224. However, the absolute sizes and positions of the models and the sampled points as ground truth are not processed in the same way in these two datasets, which makes it unreasonable to compare them all together. Therefore we evaluate the effect of our regularization technique with these two networks separately with their own released data.

Training. Unless specifically explained, the network models shown in this paper are trained as follows. The parameters for image encoders and forward 3D decoders θ_f are initialized with the parameters released by AtlasNet Groueix et al. (2018) and Pixel2Mesh Wang et al. (2018) respectively. The parameters for inverse decoders θ_g are randomly initialized. We use the ADAM Kingma and Ba (2014) optimizer with the same learning rate as in the original codes of AtlasNet and Pixel2Mesh respectively. We use $\lambda = 0.25$ as an empirical choice for the weight of our cycle regularization term.

Evaluation Criteria. In order to quantitatively evaluate the issue of self-intersection and self-overlap, we compute the percentage of self-intersected triangles ("SI") over the total number of triangles. For this evaluation, we provide our code in the supplemental material which calculates the "SI" for an input mesh. We also use Chamfer distance ("CD") as the original AtlasNet and Pixel2Mesh to evaluate how well the generated mesh approximates the ground truth shape. For the evaluation of Chamfer distance we used the codes that are already implemented by AtlasNet Groueix et al. (2018) and Pixel2Mesh Wang et al. (2018).

Mesh Generation and Visualization. Since we are addressing an issue regarding the quality of generated meshes, any post-processing used in AtlasNet Groueix et al. (2018) is not used in our experiments. All the triangulations are directly transferred from the source surface. We do not divide the triangles to get denser vertices either. The meshes from AtlasNet all have 2500 vertices and about 4k faces. The meshes from Pixel2Mesh all have 2466 vertices and 4928 faces.

To better expose the issue we are addressing in this paper, we render the generated mesh in MeshLab and use the "gooch.gdp" in the software as our shader. In such mode, triangles are rendered golden outside and bluish inside. The golden region and bluish region interlacing at surfaces evidently indicates the self-intersected surfaces.

4.1. Experiments on toy data

Being free of self-intersection is a rather geometric prior for surface mesh than a semantic one. Therefore, in this experiment, we do not involve any semantic networks and show the effect of our proposed technique in approximating a specific shape. Such experiments with toy data quickly provide an intuitive view to observe the effect of our cycle regularization

[1] https://data.mendeley.com/datasets/52z7nxkkz6/draft?a=3e7e6179-8290-4dea-9025-1998a594da12.

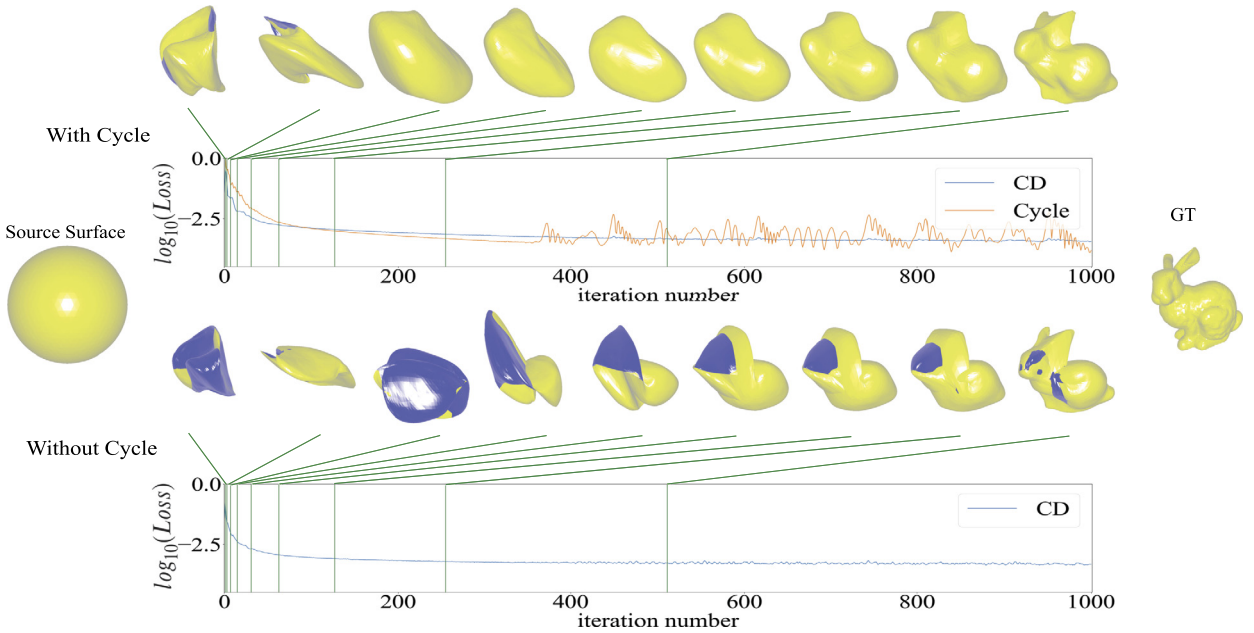

Fig. 4. Optimization of a randomly initialized MLP to approximate a specific target ground truth (GT) shape with and without cycle regularization. The cycle regularization takes effect after only a few iterations. It does not only keep the mapping injective during optimization, but also corrects the self-intersections in output that are generated by the random initialization. When optimized without the cycle regularization term, self-intersections occur during the entire optimization process.

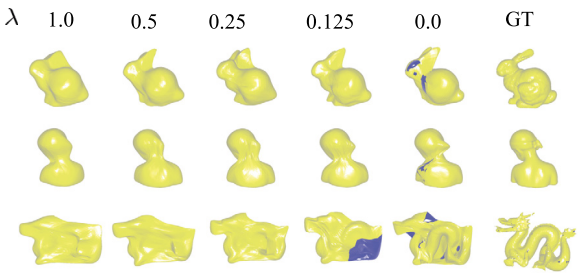

Fig. 5. Deformation results with different λ.

term at a much smaller cost than actually training networks on a large dataset. We optimize the same objective function as in Eq. (9), but do not use semantic networks (neither ResNet-18 He et al. (2016) nor PointNet Charles et al. (2017)) to generate the latent shape representation **s**. We treat **s** as a free variable. We use the same MLP for f and g as in Eq. (9), but we only optimize the output shape to approach a specific ground truth shape. We randomly initialize the parameters θ_f, θ_g and **s** with standard normal distributions and sample X from a spherical surface. We use ADAM Kingma and Ba (2014) as the optimizer with 0.001 as the learning rate. We set the maximum iteration number to 1024 for all experiments in this subsection. As shown in Fig. 4, the optimization process typically converges much earlier before 1024 iterations. Under such setting, we are optimizing a randomly initialized MLP, whose initial output usually contains self-intersections, to approximate a specific ground truth shape.

We first visualize the converging process when optimized with and without the proposed cycle regularization term. As the case shown in Fig. 4, our cycle regularization takes effect after only a few iterations. It not only keeps the mapping injective in following iterations, but also corrects self-intersections from the random initialization.

We then test on different λ to control the contribution of the cycle regularization term in the entire objective function. As shown in Fig. 5, visually speaking, when $\lambda = 0.25$, the deformed shape is able to approximate more details than using a larger λ (i.e. $\lambda = 0.5$ and 1.0), and it is also sufficient to enforce the injective mapping. Therefore, we use $\lambda = 0.25$ as an empirical choice in following experiments. However, finer tuning of λ for specific networks is possible.

We also explore cases with higher genus by manually choosing torus as the source surface. Though this is not a viable approach to enable neural networks to generate shapes with complex topology, it allows us to observe the effect of our cycle regularization in the cases of genus-one. As shown in Fig. 6, with a torus as the source surface, the cycle regularization significantly reduces self-intersection and prevents the collapse of the hole in the torus. As shown in the cases of “okay” and “love” shapes, the outputs without cycle regularization are wildly self-intersected and the hole in the tori both collapsed,

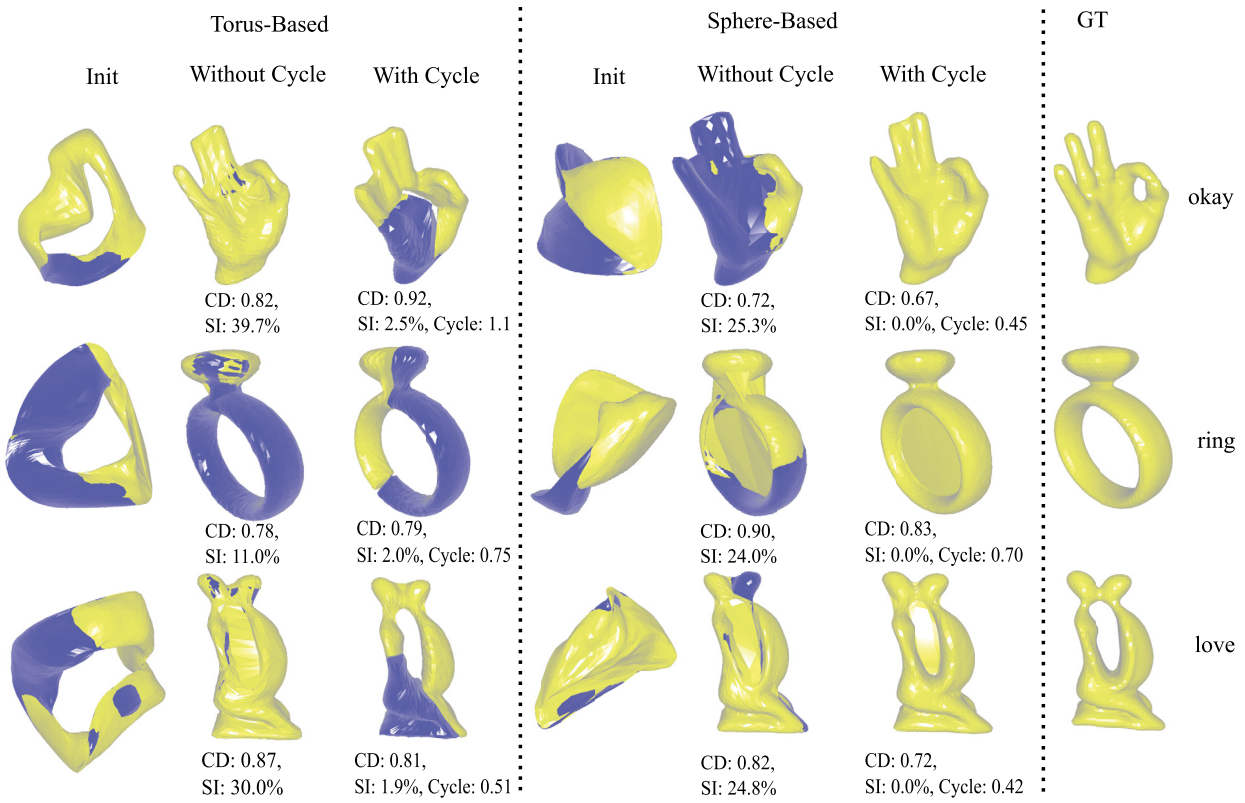

Fig. 6. Approximation of genus-one surfaces with and without cycle regularization. The columns of "Init" are the initial output shapes. The columns of "With Cycle" or "Without Cycle" are the final output shapes optimized with or without cycle regularization, respectively. Under each output shape, we report its Chamfer distance (CD) to ground truth, the percentage of self-intersection (SI) and the cycle regularization loss (Cycle, if used).

while the outputs with cycle regularization all preserve the hole. However, the torus-based outputs are more easily to get stuck at a local minimum of the cycle regularization term, where the remaining self-intersected triangles tends to twist together and form two knots. With a spherical source surface, though the outputs do not contain the hole as in the ground truth models, they are less likely to get stuck at a local minimum of the cycle regularization term with self-intersections.

4.2. Cycle regularization in AtlasNet and Pixel2Mesh

In this subsection, we conduct experiments to evaluate our cycle regularization along with two latest networks, AtlasNet Groueix et al. (2018) and Pixel2Mesh Wang et al. (2018). We report the quantitative evaluation for AtlasNet and Pixel2Mesh in Table 1 and Table 2 respectively. We show visual examples for AtlasNet and Pixel2Mesh in Fig. 7 and Fig. 8 respectively. It can be seen that after applying our cycle regularization, the percentage of self-intersected triangles are significantly reduced, while the generated shapes remain comparable to the shapes generated by the original networks in the Chamfer distance metric.

More specifically, in Table 1, we evaluate our cycle regularization with AtlasNet on two tasks, auto-encoding (AE) and single view reconstruction (SVR). In the auto-encoding task, the neural network reconstructs a complete 3D mesh from a point set (encoded by PointNet Charles et al. (2017) to generate shape representation feature $\mathbf{s}$) which contains relatively more complete information about the 3D shape. In the single view reconstruction task, the neural network takes a single view image (encoded by ResNet-18 He et al. (2016) to generate $\mathbf{s}$) as input and reconstructs a complete 3D mesh. Therefore, it is easier to reach lower Chamfer distance (0.0017) in auto-encoding. In the single view reconstruction task, there are cases, as highlighted by red rectangles in Fig. 7, that preventing self-intersections provides extra prior knowledge for the network to learn to generate more details for the generated shape. We believe this is perhaps why our performance in "CD" are slightly worse than the original AtlasNet in auto-encoding (0.0019 vs. 0.0017) but slightly better than the original AtlasNet in single view reconstruction (0.0050 vs. 0.0052). Nevertheless, both the "SI" criteria in Table 1 and the visual examples in Fig. 7 supports that the proposed cycle regularization can effectively reduce self-intersections in AtlasNet. Averagely, the percentage of self-intersected faces in our generated meshes decreases about two orders of magnitude comparing to meshes generated by the original networks.

In Table 2, we show evaluation on Pixel2Mesh. Our model is slightly worse than the original Pixel2Mesh in terms of Chamfer distance. We believe this is because we have not yet properly investigated how our cycle regularization term

Table 1
Evaluation on AtlasNet trained without and with cycle regularization (**Ours**). Chamfer distance (CD) (10^3 times at left) and the percentage of self-intersected (SI) faces (at right) are reported. "AE" stands for the shape auto-encoding task, and "SVR" stands for single view reconstruction task. Sphere means that the models are using a sphere as the predefined surface to sample points. The mean is data-wise as it is implemented in the evaluation code of AtlasNet.

	AE-sphere				SVR-sphere			
	AtlasNet		Ours		AtlasNet		Ours	
Cellphone	1.3,	0.53%	1.4,	3.4e−3%	3.8,	1.4%	3.7,	2.7e−4%
Watercraft	1.5,	2.3%	1.8,	6.8e−4%	4.3,	7.4%	4.3,	2.6e−4%
Monitor	1.8,	1.8%	2.0,	9.8e−4%	6.9,	3.4%	6.5,	9.8e−4%
Car	1.8,	0.52%	1.8,	8.0e−4%	3.9,	0.47%	3.8,	1.8e−3%
Couch	1.9,	2.5%	1.9,	8.8e−4%	5.1,	2.0%	4.9,	1.7e−3%
Cabinet	2.0	2.3%	2.2,	1.2e−2%	5.3,	3.6%	5.2,	4.3e−3%
Lamp	2.7,	14%	3.4,	5.5e−2%	13.2,	19%	13.1,	2.0e−2%
Plane	1.0,	18%	1.2,	1.9e−3%	2.6,	18%	2.6,	2.9e−3%
Speaker	2.9,	0.77%	2.9,	1.1e−3%	10.2,	1.7%	9.6,	3.1e−4%
Bench	1.3,	11%	1.6,	7.4e−3%	4.0,	12.3%	3.9,	1.6e−2%
Table	1.7,	12%	2.0,	2.1e−2%	4.9,	10.7%	4.8,	1.79e−5%
Chair	1.9,	12%	2.1,	2.7e−2%	5.3,	10.9%	5.3,	2.3e−2%
Firearm	0.7,	4.9%	0.9,	2.1e−3%	2.2,	18.2%	2.2,	1.2e−3%
Mean	**1.7**,	8.5%	1.9,	**1.3e−2%**	5.2,	9.6%	**5.0**,	**1.2e−2%**

Table 2
Evaluation on Pixel2Mesh trained with (**ours**) and without cycle regularization. For cycle regularization, the cases with fixed X (the vertices of the ellipsoids as X) and random X (sampled as in Eq. (11)) are both evaluated. Chamfer distance (CD) (10^3 times and at left) and percentage of self-intersected (SI) faces (at right) are reported. The mean is data-wise calculated.

	Pixel2Mesh		Ours			
			Fixed X		Random X	
Cellphone	0.303,	0.22%	0.304,	3.85e−3%	0.288,	3.85e−3%
Watercraft	0.433,	0.84%	0.438,	2.51e−2%	0.433,	1.25e−2%
Monitor	0.390,	0.585%	0.425,	1.15e−2%	0.397,	9.27e−3%
Car	0.233,	0.145%	0.242,	1.39e−3%	0.239,	1.24e−3%
Couch	0.361,	0.21%	0.384,	3.67e−3%	0.377,	2.26e−3%
Cabinet	0.268,	0.167%	0.283,	5.32e−3%	0.276,	5.80e−3%
Lamp	0.728,	10.3%	0.788,	0.190%	0.795,	0.182%
Plane	0.265,	1.82%	0.300,	3.75e−2%	0.289,	3.37e−2%
Speaker	0.523,	0.487%	0.524,	5.39e−3%	0.523,	5.34e−3%
Bench	0.323,	1.13%	0.349,	3.32e−2%	0.350,	1.48e−2%
Table	0.304,	1.17%	0.333,	4.98e−2%	0.330,	3.87e−2%
Chair	0.392,	1.68%	0.420,	6.82e−2%	0.414,	5.10e−2%
Firearm	0.326,	1.86%	0.352,	8.64e−2%	0.349,	7.36e−2%
Mean	0.345,	1.47%	0.369,	4.20e−2%	0.364,	3.45e−2%

interferes with other loss terms in Pixel2Mesh. However, both the overall performance in Table 2 and the visual examples in Fig. 8 are already evident enough to support the main idea in this paper that the proposed cycle regularization significantly reduces self-intersection in the generated meshes for Pixel2Mesh. In Fig. 9, we provide more examples proving that the proposed cycle regularization takes effect in all three levels in the coarse-to-fine framework of Pixel2Mesh.

To evaluate the effect of the random sampling for X on reducing the theoretical gap, which is discussed in Sec. 2, we conducted controlled experiments to train our model with a fixed point set X of an ellipsoid and randomly sampled X from the ellipsoid (as in Eq. (11)). From Table 2, we can see that our cycle regularization works better when trained with random X, especially by the measurement of self-intersection. The model trained with random X has lower average "SI" across almost all categories (except cellphone and cabinet). Thus, the results shown in Fig. 8 and Fig. 9 are all generated from the model trained with random X.

4.3. Limitations and future work

Averagely speaking, with our cycle regularization, less than two self-intersected triangles are generated in each output mesh. However, the self-intersections are not totally prevented. As shown in Fig. 10, some failure cases are shown. In the future, we would like to deduce a more elegant hard constraint on the network parameters to enforce injectivity.

Our cycle regularization is only validly deduced for the cases that the mesh reconstruction network maps from only one source surface to the target surface. This fundamental limitation prevents us to apply it for surfaces with more complicated topology. However, we believe a possible better solution for this limitation would be applying mask on the surface to handle

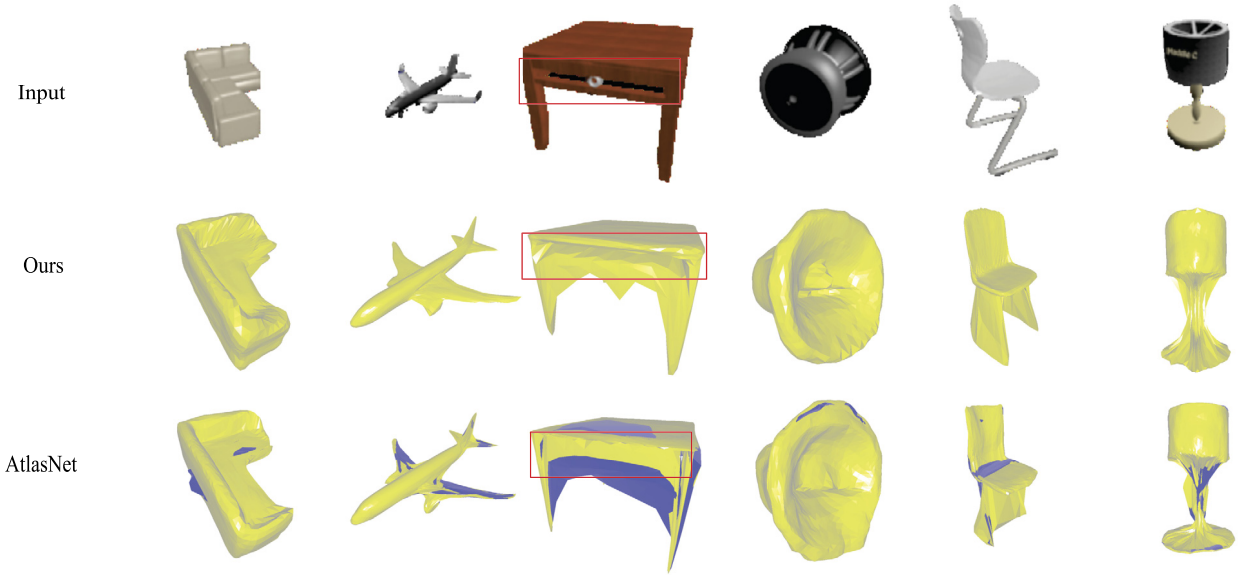

Fig. 7. Cycle regularization on AtlasNet. All the visualized cases here are selected from the test set of AtlasNet. We manually adjust the view direction for some meshes to better expose the differences. The red rectangles highlight a case where more details are preserved than the original network because injectivity is enforced with our cycle regularization.

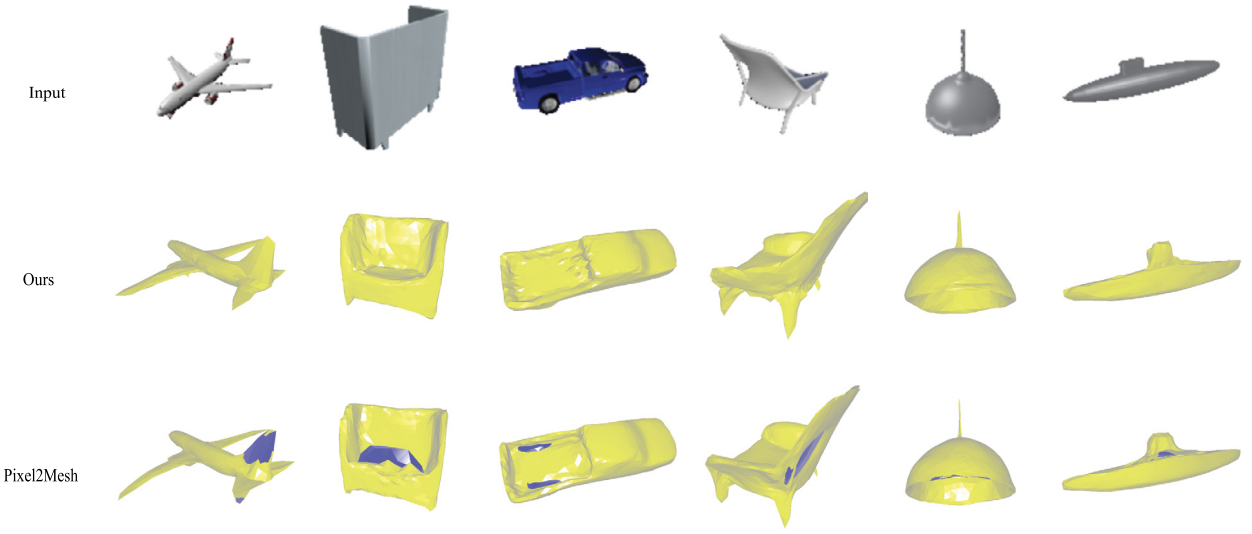

Fig. 8. Cycle regularization on Pixel2Mesh. These examples are selected from the test set of Pixel2Mesh. We adjust the view direction for some meshes to better expose the differences.

holes instead of using multiple source surfaces. We would like to use attention techniques in deep learning to predict such masks.

5. Conclusions

In this paper, we propose the cycle regularization technique for preventing surface self-intersections in generic surface mesh reconstruction networks. The cycle regularization technique reduces self-intersections by enforcing injectivity in neural networks. It stems from the basic injectivity decision theorem and enforces injectivity by simultaneously learning an inverse mapping along with original mapping. It is quite simple to implement with existing mesh generation networks. Our experiments on two latest mesh generation networks prove that our technique is effective for reducing self-intersections in the generated meshes.

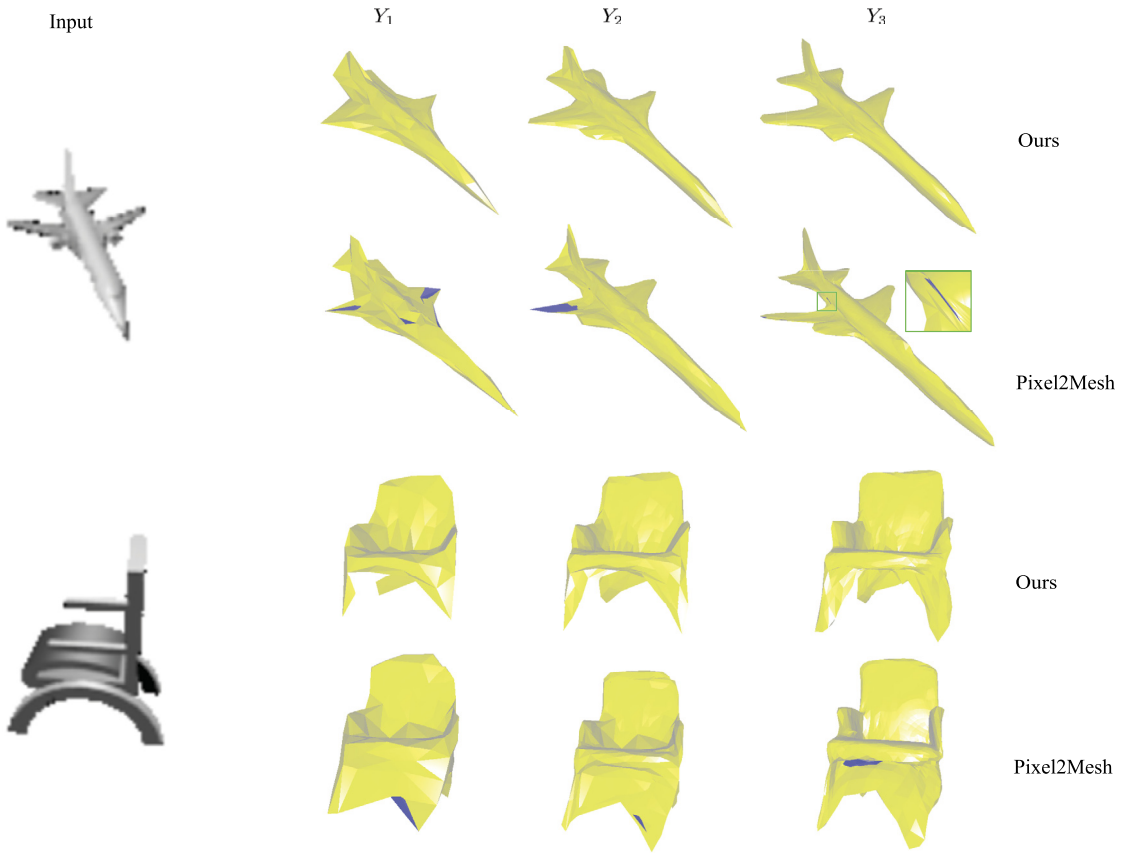

Fig. 9. Visualization of the effect of our cycle regularization in the coarse-to-fine framework of Pixel2Mesh. The green rectangle shows a close-up view of a subtle self-intersection in the airplane model.

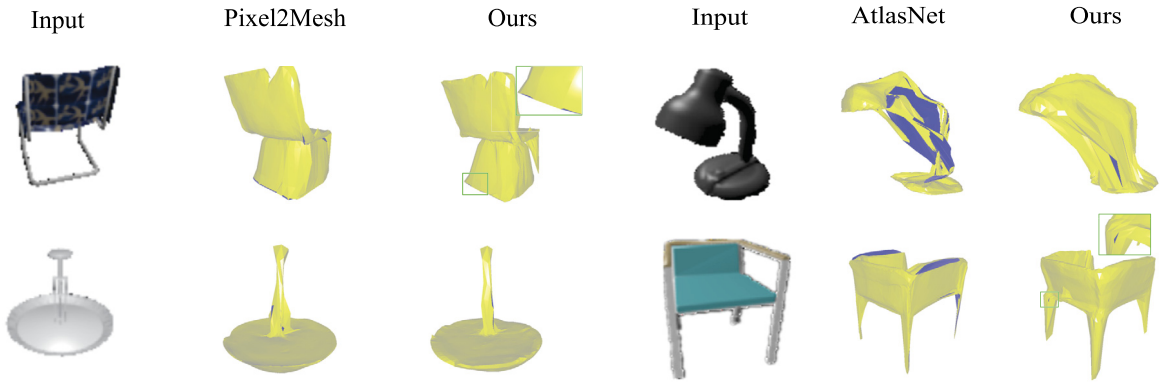

Fig. 10. Failure cases of our cycle regularization. Though self-intersections are significantly reduced for most cases, they are not entirely removed in these results.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank Dr. Xin Tong and Hao Su for their insightful comments and suggestions. This work was supported by the National Key Research and Development Plan of China under Grant No. 2016YFB1001402, the National Natural

Science Foundation under Grant No. 61632006, and the Fundamental Research Funds for the Central Universities under Grant WK3490000003.

References

Aigerman, N., Poranne, R., Lipman, Y., 2014. Lifted bijections for low distortion surface mappings. ACM Trans. Graph. 33, 69. URL: http://doi.acm.org/10.1145/2601097.2601158. https://doi.org/10.1145/2601097.2601158.

Aloimonos, J., 1988. Shape from texture. Biol. Cybern. 58, 345–360. URL: https://doi.org/10.1007/BF00363944. https://doi.org/10.1007/BF00363944.

Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F., 2015. ShapeNet: an Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

Charles, R.Q., Su, H., Kaichun, M., Guibas, L.J., 2017. Pointnet: deep learning on point sets for 3D classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 77–85.

Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S., 2016. 3D-R2n2: a unified approach for single and multi-view 3D object reconstruction. CoRR arXiv:1604.00449 [abs]. URL: http://arxiv.org/abs/1604.00449.

Criminisi, A., Zisserman, A., 2000. Shape from texture: homogeneity revisited. In: Proc. British Machine Vision Conference. BMVC, pp. 82–91. URL: https://www.microsoft.com/en-us/research/publication/shape-from-texture-homogeneity-revisited/.

Dou, P., Shah, S.K., Kakadiaris, I.A., 2017. End-to-end 3D face reconstruction with deep neural networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1503–1512. URL: https://ieeecomputersociety.org/10.1109/CVPR.2017.164.

Fan, H., Su, H., Guibas, L., 2017. A point set generation network for 3D object reconstruction from a single image. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR.

Favaro, P., Soatto, S., 2005. A geometric approach to shape from defocus. IEEE Trans. Pattern Anal. Mach. Intell. 27, 406–417. https://doi.org/10.1109/TPAMI.2005.43.

Favaro, P., Soatto, S., Burger, M., Osher, S.J., 2008. Shape from defocus via diffusion. IEEE Trans. Pattern Anal. Mach. Intell. 30, 518–531. https://doi.org/10.1109/TPAMI.2007.1175.

Gain, J.E., Dodgson, N.A., 2001. Preventing self-intersection under free-form deformation. IEEE Trans. Vis. Comput. Graph. 7, 289–298. https://doi.org/10.1109/2945.965344.

Girdhar, R., Fouhey, D.F., Rodriguez, M., Gupta, A., 2016. Learning a predictable and generative vector representation for objects. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.), Computer Vision – ECCV 2016. Springer International Publishing, Cham, pp. 484–499.

Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M., 2018. A papier-mâché approach to learning 3D surface generation. In: The IEEE Conference on Computer Vision and Pattern Recognition. CVPR.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 770–778. URL: https://ieeecomputersociety.org/10.1109/CVPR.2016.90. https://doi.org/10.1109/CVPR.2016.90.

Hoiem, D., Efros, A.A., Hebert, M., 2007. Recovering surface layout from an image. Int. J. Comput. Vis. 75, 151–172. URL: https://doi.org/10.1007/s11263-006-0031-y. https://doi.org/10.1007/s11263-006-0031-y.

Horn, B.K.P., 1975. Obtaining shape from shading information. The Psychology of Computer Vision. URL: https://ci.nii.ac.jp/naid/10017628596/en/.

Huang, Q., Wang, H., Koltun, V., 2015. Single-view reconstruction via joint analysis of image and shape collections. ACM Trans. Graph. 34, 87. URL: http://doi.acm.org/10.1145/2766890. https://doi.org/10.1145/2766890.

Jung, W., Shin, H., Choi, B.K., 2004. Self-intersection removal in triangular mesh offsetting. Comput-Aided Des. Appl. 1, 477–484. URL: https://doi.org/10.1080/16864360.2004.10738290. https://doi.org/10.1080/16864360.2004.10738290. https://doi.org/10.1080/16864360.2004.10738290.

Kar, A., Tulsiani, S., Carreira, J., Malik, J., 2015. Category-specific object reconstruction from a single image. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1966–1974.

Kato, H., Ushiku, Y., Harada, T., 2017. Neural 3D mesh renderer. CoRR arXiv:1711.07566 [abs]. URL: http://arxiv.org/abs/1711.07566. arXiv:1711.07566.

Kingma, D., Ba, J., 2014. Adam: a method for stochastic optimization. URL http://arxiv.org/abs/1412.6980. cite arXiv:1412.6980.
Comment: Published as a Conference Paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Kingma, D.P., Welling, M., 2013. Auto-encoding variational Bayes. ArXiv preprint arXiv:1312.6114.

Li, Y., Barbič, J., 2018. Immersion of self-intersecting solids and surfaces. ACM Trans. Graph. (SIGGRAPH 2018), 37.

Li, Y., Bu, R., Sun, M., Chen, B., 2018. Pointcnn. CoRR arXiv:1801.07791 [abs]. URL: http://arxiv.org/abs/1801.07791. arXiv:1801.07791.

Lipman, Y., 2012. Bounded distortion mapping spaces for triangular meshes. ACM Trans. Graph. 31, 108. URL: http://doi.acm.org/10.1145/2185520.2185604. https://doi.org/10.1145/2185520.2185604.

Liu, L., Ye, C., Ni, R., Fu, X.M., 2018. Progressive parameterizations. ACM Trans. Graph. (SIGGRAPH) 37.

Pekerman, D., Elber, G., Kim, M.S., 2008. Self-intersection detection and elimination in freeform curves and surfaces. Comput. Aided Des. 40, 150–159. URL: http://www.sciencedirect.com/science/article/pii/S0010448507002357. https://doi.org/10.1016/j.cad.2007.10.004.

Pontes, J.K., Kong, C., Sridharan, S., Lucey, S., Eriksson, A.P., Fookes, C., 2017. Image2mesh: a learning framework for single image 3D reconstruction. CoRR arXiv:1711.10669 [abs]. URL: http://arxiv.org/abs/1711.10669. arXiv:1711.10669, 2017.

Poranne, R., Lipman, Y., 2014. Provably good planar mappings. ACM Trans. Graph. 33, 76. URL: http://doi.acm.org/10.1145/2601097.2601123. https://doi.org/10.1145/2601097.2601123.

Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 5105–5114. URL: http://papers.nips.cc/paper/7095-pointnet-deep-hierarchical-feature//learning-on-point-sets-in-a-metric-space.pdf.

Rhodin, H., Breuß, M., 2013. A mathematically justified algorithm for shape from texture. In: Kuijper, A., Bredies, K., Pock, T., Bischof, H. (Eds.), Scale Space and Variational Methods in Computer Vision. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 294–305.

Saxena, A., Sun, M., Ng, A.Y., 2007. Learning 3-d scene structure from a single still image. In: 2007 IEEE 11th International Conference on Computer Vision, pp. 1–8.

Schüller, C., Kavan, L., Panozzo, D., Sorkine-Hornung, O., 2013. Locally injective mappings. Comput. Graph. Forum. https://doi.org/10.1111/cgf.12179.

Sederberg, T.W., Parry, S.R., 1986. Free-form deformation of solid geometric models. SIGGRAPH Comput. Graph. 20, 151–160. URL: http://doi.acm.org/10.1145/15886.15903. https://doi.org/10.1145/15886.15903.

Sinha, A., Bai, J., Ramani, K., 2016. Deep learning 3D shape surfaces using geometry images. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.), Computer Vision – ECCV 2016. Springer International Publishing, Cham, pp. 223–240.

Sinha, A., Unmesh, A., Huang, Q., Ramani, K., 2017. Surfnet: generating 3D shape surfaces using deep residual networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 791–800 URL: https://ieeecomputersociety.org/10.1109/CVPR.2017.91. https://doi.org/10.1109/CVPR.2017.91.

Smith, J., Schaefer, S., 2015. Bijective parameterization with free boundaries. ACM Trans. Graph. 34, 70. URL: http://doi.acm.org/10.1145/2766947. https://doi.org/10.1145/2766947.

Su, H., Huang, Q., Mitra, N.J., Li, Y., Guibas, L., 2014. Estimating image depth using shape collections. ACM Trans. Graph. 33, 37. URL: http://doi.acm.org/10.1145/2601097.2601159. https://doi.org/10.1145/2601097.2601159.

Tatarchenko, M., Dosovitskiy, A., Brox, T., 2017. Octree generating networks: efficient convolutional architectures for high-resolution 3D outputs. In: 2017 IEEE International Conference on Computer Vision. ICCV, pp. 2107–2115. URL: https://ieeecomputersociety.org/10.1109/ICCV.2017.230. https://doi.org/10.1109/ICCV.2017.230.

Tutte, W.T., 1963. How to draw a graph. Proc. Lond. Math. Soc. s3–13, 743–767. URL: https://doi.org/10.1112/plms/s3-13.1.743. https://doi.org/10.1112/plms/s3-13.1.743.

Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X., 2017. O-CNN: octree-based convolutional neural networks for 3D shape analysis. ACM Trans. Graph. (SIGGRAPH) 36.

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G., 2018. Pixel2mesh: generating 3D mesh models from single RGB images. In: ECCV.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: a deep representation for volumetric shapes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1912–1920.

Wu, J., Zhang, C., Zhang, X., Zhang, Z., Freeman, W.T., Tenenbaum, J.B., 2018. Learning shape priors for single-view 3D completion and reconstruction. In: The European Conference on Computer Vision. ECCV.

Yamakawa, S., Shimada, K., 2009. Removing self intersections of a triangular mesh by edge swapping, edge hammering, and face lifting. In: Clark, B.W. (Ed.), Proceedings of the 18th International Meshing Roundtable. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 13–29.

Zhang, R., Tsai, P.S., Cryer, J.E., Shah, M., 1999. Shape-from-shading: a survey. IEEE Trans. Pattern Anal. Mach. Intell. 21, 690–706. https://doi.org/10.1109/34.784284.

Zhu, J.Y., Park, T., Isola, P., Efros, A.A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision. ICCV.