Pixel2Mesh: 3D Mesh Model Generation via Image Guided Deformation

Nanyang Wang*, Yinda Zhang*, Zhuwen Li*, Yanwei Fu*, Hang Yu, Wei Liu, Xiangyang Xue and Yu-Gang Jiang†

Abstract—In this paper, we propose an end-to-end deep learning architecture that generates 3D triangular meshes from single color images. Restricted by the nature of prevalent deep learning techniques, the majority of previous works represent 3D shapes in volumes or point clouds. However, it is non-trivial to convert these representations to compact and ready-to-use mesh models. Unlike the existing methods, our network represents 3D shapes in meshes, which are essentially graphs and well suited for graph-based convolutional neural networks. Leveraging perceptual features extracted from an input image, our network produces the correct geometry by progressively deforming an ellipsoid. To make the whole deformation procedure stable, we adopt a coarse-to-fine strategy, and define various mesh/surface related losses to capture properties of various aspects, which benefits producing the visually appealing and physically accurate 3D geometry. In addition, our model by nature can be adapted to objects in specific domains, e.g., human faces, and be easily extended to learn per-vertex properties, e.g., color. Extensive experiments show that our method not only qualitatively produces the mesh model with better details, but also achieves the higher 3D shape estimation accuracy compared against the state-of-the-arts.

Index Terms—3D shape generation, graph convolutional neural network, mesh reconstruction, coarse-to-fine, end-to-end framework.

1 Introduction

Inferring the 3D shape from a single perspective is a fundamental human vision functionality. Its success of it can potentially reinforce many high-level vision tasks, such as shape retrieval, pose estimation, and object detection. However, it is extremely challenging for computer vision as it requires searching for 3D shapes that explain the observation as well as hallucinating occluded parts.

Recently, a great success has been achieved for 3d shape generation from a single color image using deep learning techniques [1], [2], [3], thanks to their superior learning capability to integrate the shape prior. Taking advantage of convolutional layers on regular grids or multi-layer perceptions, the estimated 3D shape, as the output of the neural network, is usually represented as either a volume [1] or a point cloud [2]. However, both shape representations have strong limitations and lose important surface details. Volumetric representation requires high spatial resolution to capture geometry details [1], which consumes a tremendous amount of memory to represent the solid interior or empty outside space. Point cloud representation lacks connectivity among vertices and surface topology [2], and it is thus non-trivial to produce a watertight shape. In contrast, a more desirable shape representation is a composition of 3D surfaces, i.e., a mesh model, since it is light-weight, capable of modelling shape details, and easy to deform for animation, to name a few. A comparison between different shape representations is shown in Fig. 1.

In this paper, we push along the direction of single image reconstruction, and propose a deep learning framework [4], [5] to extract a 3D triangular mesh from a single color image. Rather than directly synthesizing a mesh, i.e., vertex locations and connectivity, out of a neural network, our model learns to deform a mesh from an initial shape to the target geometry. This benefits us from several aspects. First, deep network is better at predicting residual, e.g., a spatial deformation, rather than structured output, e.g., a graph. Second, a series of deformations can be added up together, which allows shape to be gradually refined from coarse to fine. This practically enables efficient generation of rough shapes, and only requires more computation if fine-grained details are necessary. Last but not least, it provides the chance to encode any prior knowledge to the initial mesh, e.g., topology. As a pioneering study, in this work, we specifically work on objects that can be approximated using a 3D mesh with genus 0 by deforming an ellipsoid with a fixed size. In practice, we found that most of the commonly seen categories can be handled well under this setting, e.g., car, plane, table, human face, etc.

Since there is no 3D information available from the input, i.e., an RGB image, our model needs to infer a shape that is visually consistent with the given color image, and hallucinate the shape on the occluded backside. To achieve this goal, several inherent challenges need to be addressed. The first challenge is how to represent a mesh model, which is essentially an irregular graph, in a neural network and still be capable of extracting shape details effectively from

* *indicates equal contributions,* † *refers to the corresponding author.*

- *Nanyang Wang, Hang Yu, Xiangyang Xue, and Yu-Gang Jiang are with the School of Computer Science, Fudan University, Shanghai, China. E-mail: {nywang16, xyxue, ygj}@fudan.edu.cn.*
- *Yinda Zhang is with the Department of Computer Science, Princeton University, Princeton, New Jersey. Email: yindaz@cs.princeton.edu.*
- *Zhuwen Li is with Nuro, Inc., Mountain View, California. Email: lzhuwen@gmail.com.*
- *Yanwei Fu is with School of Data Science, and MOE Frontiers Center for Brain Science, Shanghai Key Lab of Intelligent Information Processing Fudan University. Email: yanweifu@fudan.edu.cn.*
- *Wei Liu is with the Tencent AI Lab, Shenzhen, China. Email: wl2223@columbia.edu.*

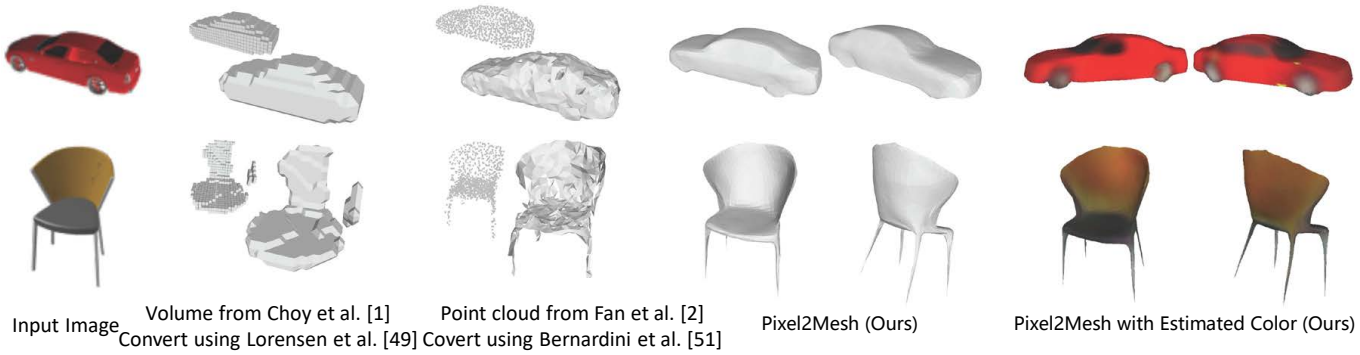


Fig. 1. Given a single color image and an initial mesh, our method can produce a high-quality mesh that contains details from the example. Comparatively, previous work based on volume and point cloud cannot recover detail and is non-trivial to be converted into water-tight mesh models. Moreover, our model can predict per-vertex properties such as a texture map.

a given color image represented in a 2D regular grid. It requires the integration of the knowledge learned from two data modalities. For the 3D geometry, we directly build a graph based convolutional network (GCN) [6], [7], [8] on the mesh model, where the vertices and edges in the mesh are represented as nodes and connections in the neural network. Features encoding information for 3D shape are saved at each vertex. Through forward propagation, the convolutional layers enable feature exchanging across neighboring nodes, and eventually regress the 3D location for each vertex. For the 2D image, we use a VGG-16 like architecture to extract features as it has been demonstrated to be successful for many tasks [4], [5], [9]. To bridge these two architectures, we design a perceptual feature pooling layer to allow each node in the GCN to pool image features from its 2D projection on the image, which can be readily obtained via the known camera intrinsic matrix. The perceptual feature pooling is enabled once after several convolutions (i.e., a deformation block described in Section 3.3) using updated 3D locations, and hence the image features from correct locations can be effectively integrated with 3D shapes.

Given the graph representation, the next challenge is how to update the vertex location effectively towards the ground truth. In practice, we observe that a network trained to directly predict the mesh with a large number of vertices is likely to make a mistake in the beginning and hard to fix later. One reason is that a vertex cannot effectively retrieve features from other vertices with a number of edges away, i.e., the limited receptive field. To solve this problem, we design a graph unpooling layer, which allows the network to initiate with a smaller number of vertices and increase during the forward propagation. With fewer vertices at the beginning stages, the network learns to distribute the vertices around to the most representative location, and then adds local details as the number of vertices increases later. Besides the graph unpooling layer, we use a deep GCN enhanced by shortcut connections [10] as the backbone of our architecture, which enables large receptive fields for global context and more steps of movements.

Representing the shape in a graph also benefits the learning procedure. The known connectivity allows us to define higher order loss functions across neighboring nodes, which are important to regularize 3D shapes. Specifically, we define a surface normal loss to constrain the surface, an edge loss to encourage a uniform distribution of mesh vertices for high recall, and a laplacian loss to prevent mesh faces from intersecting with each other. All of these losses are essential to generate a quality appealing mesh model, and none of them can be trivially defined without the graph representation.

Yet these losses not only capture information in different aspects, but also enable a spatially variant behavior of the model generation by heterogeneously weighting losses at different locations, which allow our model to adapt to a specific domain smoothly. For example, modeling 3D human faces requires comparatively more vertices near eyes and nose than other regions so as to capture fine-grained distinctive details. This can be achieved by increasing the weights of the chamfer loss (i.e., pay more attention to regressing 3D position) and edge-length regularization (i.e., shorter edges and denser vertices) near the important regions indicated by a predefined weight vector over the vertices. Moreover, on par with accurate 3D geometry, our model by nature can easily predict per-vertex properties as well. In fact, the 3D location, i.e., (x, y, z) coordinate, of a vertex can be considered as one kind of property, and the same network architecture can thus be extended to predict other properties as long as they can be represented as values or classes on each vertex, such as color for both visible and occluded vertices. Fig. 1 shows an example that our approach can predict a textured 3D mesh.

The contributions of this paper are mainly as follows.

- First, rather than volumetric representation or point cloud, we propose a novel end-to-end neural network architecture that generates 3D mesh models from single RGB images.
- Second, we propose generation by deformation in a coarse to fine fashion, in which the deformation is guided by a GCN with perceptual feature pooling layers.
- Third, we demonstrate that our method produces more accurate 3D shapes than the state-of-the-arts, easily adapts to shapes of a specific domain (e.g., human face), and can predict per-vertex properties (e.g., color).

An early and preliminary version of this work has been published in [11]. Compared with [11], modification of the

network architecture, new features, and evaluation experiments have been added. Particularly, our model is extended to predict per-vertex properties, such as color. It thus shows that our model can generate the nice-looking textured mesh for both visible and occluded regions of the 3D object. Furthermore, we adapt our model to other category domains, i.e., human face, and demonstrate that our model can be trained to behave spatially heterogeneously such that important local details can be recovered with more vertices in certain interesting areas. Essentially, our generated 3D human face mesh successfully captures distinctive facial characteristics.

2 Related Work

2.1 Multi-view geometry

3D reconstruction has been well studied based on the multi-view geometry (MVG) [12] in the literature. The major research directions include structure from motion (SfM) [13] for large-scale high-quality reconstruction and simultaneous localization and mapping (SLAM) [14] for navigation. Though they achieve great successes, traditional MVG has a few drawbacks. The data captures is usually time consuming and requires the expertise to obtain a good number of views with both high object coverage and image overlaps. The view-dependent surface appearance, e.g., caused by specular reflections, can often cause troubles for correspondence matching under challenging illumination, which are addressed by some literature [15] but still far from being practically feasible. These restrictions lead to the trend of resorting to learning based approaches. Problematic for MVS methods are specular reflections, which can also occur for uniform or homogenous materials.

2.2 Single image 3D reconstruction

Learning based approaches usually consider single or few images, as they largely rely on the shape priors that can be learnt from data. Early works can be traced back to Hoiem *et al.* [16] and Saxena *et al.* [17]. Most recently, with the success of deep learning and the release of large-scale 3D shape datasets such as ShapeNet [18] and Pix3D [19], learning based approaches have achieved a great progress. Huang *et al.* [20] and Su *et al.* [21] retrieved shape components from a large dataset, assembled them, and deformed the assembled shape to fit the observed image. Despite some successes, the shape retrieval may not be ideal and potentially increases chances for the whole system failure.

On the other hand, the line of research that directly learns 3D shapes from single images is promising due to the fashion of end-to-end learning. Restricted by the prevalent grid-based deep learning architectures, most works [1], [22] output 3D volumes, which are usually with low resolutions due to the memory constraint on a modern GPU. Then, Tatarchenko *et al.* [23] proposed an octree representation, which allows reconstructing higher resolution outputs with a limited memory budget. However, 3D volume is still not a popular shape representation in game and movie industries. To avoid drawbacks of the voxel representation, many efforts have been devoted to exploring new 3D representation. Fan *et al.* [2] proposed to generate point clouds from single images. One drawback of the point cloud representation is that there is no local connections between points, and thus the point positions have a very large degree of freedoms. Consequently, the generated point cloud is usually not close to a surface and cannot be used to recover a 3D mesh directly.

Kar *et al.* [24] proposed a deformable shape representation to learn category specific shape knowledge but required auxiliary semantic classification and segmentation during the inference. Sinha *et al.* [25] proposed "geometry image" to represent a 3D shape which allows using well-studied 2D convolutions for 3D shape. A similar idea has been adopted by Matryoshka Networks [26] but with more sophisticated compositions. Mostly recently, Tatarchenko *et al.* [27] discussed the encoder-decoder based single image reconstruction approaches and showed their limitations. After our ECCV conference paper, ONet [28] was proposed as a novel implicit representation for learning-based 3D reconstruction. This method improves the geometry details but requires comparatively long inference time due to sampling. The implicit surface representation is orthogonal, and potentially useful for improving the explicit mesh representation, which is adopted in this work.

Our work is mostly related to the two recent works [29] and [30]. However, the former adopts simple silhouette supervision, and hence does not perform well for complicated objects such as car, lamp, etc; the latter needs a large model repository to generate a combined model.

2.3 Graph neural network

CNN has shown powerful in many domains, such as images and natural language processing. However, it is non-trivial to apply CNN to graph because of the irregular data structure. Recently, research on graphs with deep learning methods has drawn huge attention, because lots of real world data can be represented as graphs, such as social networks, traffic networks, and 3D mesh models. In fact, an image can be viewed as a graph with regular connections. Our base network is a graph neural network [31]; this architecture has been adopted for shape analysis [32]. Meanwhile, there are charting-based methods which directly apply convolutions on surface manifolds [33], [34], [35] for shape analysis. As far as we know, these architectures have never been adopted for 3D reconstruction from single images, though graph and surface manifold are natural representations for meshed objects. For a comprehensive understanding of the graph neural network, the charting-based methods, and their applications, please refer to this survey [6].

2.4 3D face reconstruction

Another closely related research topic is 3D face reconstruction, in the single perspective setup. In the literature, the most common scheme is based on 3D morphable model (3DMM) [36], which represents 3D face by a principal components analysis (PCA) basis. Most of the early works [37], [38], [39], [40] fit the 3D face by minimizing the error between special feature points between input image and rendered 3D model, and then get the 3DMM parameters by solving a non-linear optimization problem. However, these methods are highly sensitive to the initialization such as the

accuracy of feature points detector. With the success of deep learning, many recent methods learn to estimate the 3DMM coefficients leveraging CNN regression. [41], [42] propose to predict the coefficient update at each iteration. [43], [44] directly predict the 3DMM coefficients by end-to-end CNN architectures. However, the 3DMM schemes are restricted by the principal components which result in a limited shape space. More recently, Jackson *et al.* [45] proposed to reconstruct the 3D structure by direct volumetric CNN regression. Feng *et al.* [46] proposed a method to predict the position map in the UV space. We demonstrate that our method can be used to reconstruct 3D face bypassing 3DMM fiting and regression. As far as we know, our proposed framework is the first model that can reconstruct 3D meshes for both general objects and human faces.

3 OVERVIEW

Graph-based Convolution. We first provide a brief introduction to graph based convolution; more details can be found in [6]. A 3D mesh is a collection of vertices, edges and faces that defines the shape of a 3D object; it can be represented by a graph $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathbf{F})$, where $\mathcal{V} = \{v_i\}_{i=1}^{N}$ is the set of N vertices in the mesh, $\mathcal{E} = \{e_i\}_{i=1}^{E}$ is the set of E edges with each connecting two vertices, and $\mathbf{F} = \{f_i\}_{i=1}^{N}$ are the feature vectors attached on vertices. A graph based convolutional layer is defined on irregular graph as:

$$f_p^{l+1} = w_0 f_p^l + \sum_{q \in \mathcal{N}(p)} w_1 f_q^l, \tag{1}$$

where $f_p^l \in \mathbb{R}^{d_l}$, $f_p^{l+1} \in \mathbb{R}^{d_{l+1}}$ are the feature vectors on vertex p before and after the convolution, and $\mathcal{N}(p)$ is the neighboring vertices of p; w_0 and w_1 are the learnable parameter matrices of $d_l \times d_{l+1}$ that are applied to all vertices. Note that w_1 is shared for all edges, and thus (1) works on nodes with different vertex degrees. In our case, the attached feature vector f_p is the concatenation of the 3D vertex coordinate, feature encoding 3D shape, and feature extracted from the input color image (if they exist). Running convolutions updates the features, which is equivalent to applying a deformation.

3.1 System overview

Our model is an end-to-end deep learning framework that takes a single color image as input and produces a 3D mesh model in camera coordinate. The overview of our framework is illustrated in Fig. 2. The whole network consists an image feature network and a cascaded mesh deformation network. The image feature network is a 2D CNN that extracts perceptual features from the input image, which is leveraged by the mesh deformation network to progressively deform an ellipsoid mesh into the desired 3D model. The cascaded mesh deformation network is a graph-based convolutional network (GCN), which contains three deformation blocks intersected by two graph unpooling layers. Each deformation block takes an input graph representing the current mesh model with the 3D shape feature attached on vertices, and produces new vertices locations and features. Whereas the graph unpooling layers increase the number of vertices to increase the capacity of handling details, while still maintaining the triangular mesh topology. Starting from a smaller number of vertices, our model learns to gradually deform and add details to the mesh model in a coarse-to-fine fashion. In order to train the network to produce stable deformation and generate an accurate mesh, we extend the Chamfer Distance loss used by Fan *et al.* [2] with three other mesh specific losses – Surface normal loss, Laplacian regularization loss, and Edge length loss. The remaining part of this section describes details of these components.

3.2 Initial ellipsoid

Our model does not require any prior knowledge of the 3D shape, and always deforms from an initial ellipsoid with average size placed at the common location in the camera coordinate. The ellipsoid is centered at 0.8m in front of the camera with 0.2m, 0.2m, 0.4m as the radii of three axes. The mesh model is generated by the implicit surface algorithm in Meshlab [47] and contains 156 vertices. We use this ellipsoid to initialize our input graph, where the initial feature contains only the 3D coordinate of each vertex.

3.3 Mesh deformation block

The architecture of mesh deformation block is shown in Fig. 3 (a). In order to generate a 3D mesh model that is consistent with the object shown in the input image, the deformation block needs to pool feature ($\mathbf{P}$) from the input image. This is done in conjunction with the image feature network and a perceptual feature pooling layer given the location of vertex ($\mathbf{C}_{i-1}$) in the current mesh model. The pooled perceptual feature is then concatenated with the 3D shape feature attached on the vertex from the input graph ($\mathbf{F}_{i-1}$) and fed into a series of Graph based ResNets (G-ResNets). The G-ResNet produces, also as the output of the mesh deformation block, the new coordinates ($\mathbf{C}_i$) and 3D shape feature ($\mathbf{F}_i$) for each vertex.

3.3.1 Perceptual feature pooling layer

We use a VGG-16 like architecture up to layer conv5_3 as the image feature network as it has been widely used. Given the 3D coordinate of a vertex, we calculate its 2D projection on the input image plane using camera intrinsics, and then pool the features from four nearby pixels using bilinear interpolation. In particular, we concatenate features extracted from layer 'conv3_3', 'conv4_3', and 'conv5_3', which results in a total dimension of 1280. This perceptual feature is then concatenated with the 128-dim 3D features from the input mesh, which results in a total dimension of 1408. This is illustrated in Fig. 3 (b). Note that in the first block, the perceptual feature is 3-dim features (3D coordinates) since there is no learnt shape feature at the beginning.

3.3.2 G-ResNet

After obtaining 1408-dim features for each vertex representing both 3D shape and 2D image information, we design a graph based convolutional neural network to predict new location and 3D shape feature for each vertex. This requires an efficient exchange of the information between vertices.

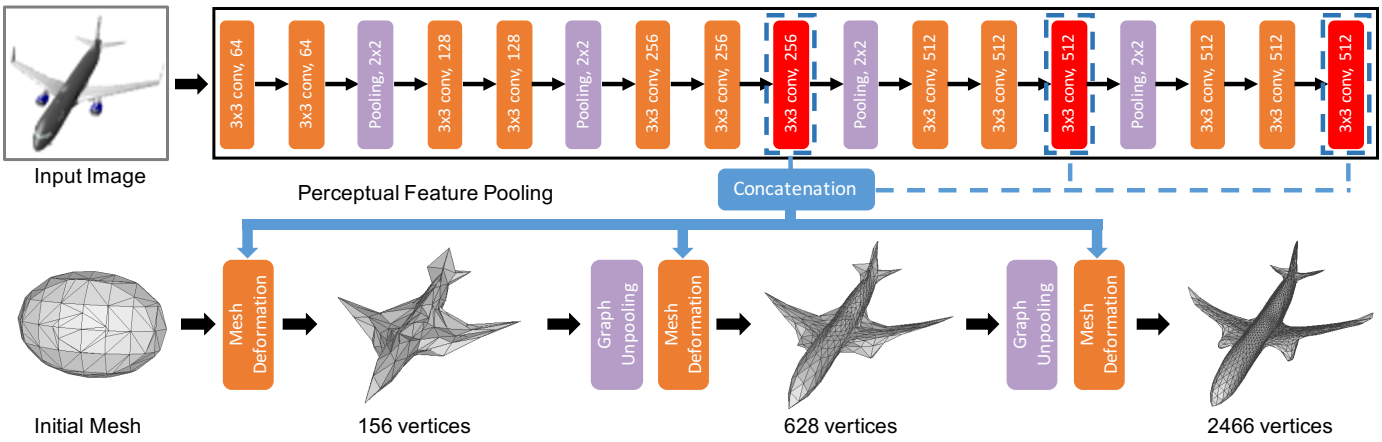
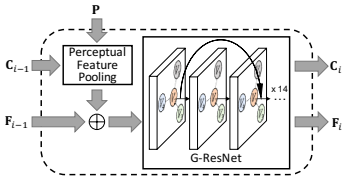
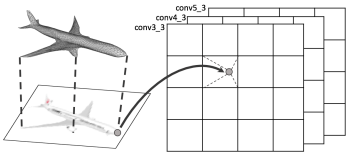


Fig. 2. The cascaded mesh deformation network. Our full model contains three mesh deformation blocks in a row. Each block increases mesh resolution and estimates vertex locations, which are then used to extract perceptual image features from the 2D CNN for the next block.

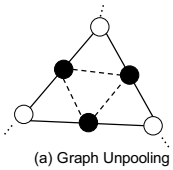


(a) Mesh Deformation Block

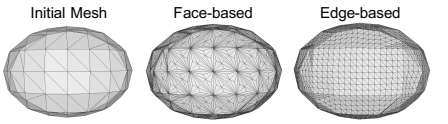


(b) Perceptual Feature Pooling

Fig. 3. (a) The vertex locations $\mathbf{C_i}$ are used to extract image features, which are then combined with vertex features $\mathbf{F_i}$ and fed into G-ResNet. $\bigoplus$ means a concatenation of the features. (b) The 3D vertices are projected to the image plane using camera intrinsics, and perceptual features are pooled from the 2D-CNN layers using bilinear interpolation.



(a) Graph Unpooling



(b) Comparison between face-based and edge-based unpooling

Fig. 4. (a) Black vertices and dashed edges are added in the unpooling layer. (b) The face based unpooling leads to imbalanced vertex degrees, while the edge-based unpooling remains regular.

However, as defined in (1), each convolution only enables the feature exchanging between neighboring pixels, which severely impairs the efficiency of information exchanging. This is equivalent to the small receptive field issue on 2D CNN.

Since increasing the depth of the network is equivalent to increasing the receptive field, we make a very deep network with shortcut connections [10] and denote it as G-ResNet (Fig. 3 (a)). In this work, the G-ResNet in all blocks has the same structure, which consists of 14 graph residual convolutional layers with 128 channels. There is no weight sharing. The serial of G-ResNet block produces a new 128-dim 3D features. In addition to the feature output, there is a branch which applies an extra graph convolutional layer to the last layer features and outputs the 3D coordinates of the vertex.

3.4 Graph unpooling layer

The goal of unpooling layer is to increase the number of vertices in the GCN. It allows us to start from a mesh with fewer vertices and add more only when necessary, which reduces memory costs and produces better results. A straightforward approach is to add one vertex in the center of each triangle and connect it with the three vertices of the triangle (Fig. 4 (b) Face-based). However, this causes imbalanced vertex degrees, i.e., number of edges on a vertex. Inspired by the vertex adding strategy of the mesh subdivision algorithm prevalent in computer graphics, we add a vertex at the center of each edge and connect it with the two end-points of this edge (Fig. 4 (a)). The 3D feature for a newly added vertex is set as the average of its two neighbors. We also connect three vertices if they are added on the same triangle (dashed line.) Consequently, we create 4 new triangles for each triangle in the original mesh, and the number of vertices is increased by the number of edges in the original mesh. This edge-based unpooling uniformly upsamples the vertices as shown in Fig. 4 (b) Edge-based.

3.5 Color Pixel2mesh

As show in Fig. 5, our model can have an additional parallel branch in the last block to produce not only the 3D location of each vertex but also other per-pixel properties. In fact, the 3D location, i.e., (x; y; z) coordinate, of a vertex can be viewed as one kind of properties, and the same network architecture can thus be extended to predict other properties as long as they can be represented as values or classes on each vertex, such as vertex color. In Fig. 5, the additional task is to estimate the RGB value for each vertex on the generated mesh.

4 Loss Functions

We define four kinds of losses to constrain the property of the output shape and the deformation procedure to guarantee appealing results. We adopt the Chamfer loss [2] to constrain the location of mesh vertices, a normal loss to enforce the consistency of surface normal, a Laplacian regularization

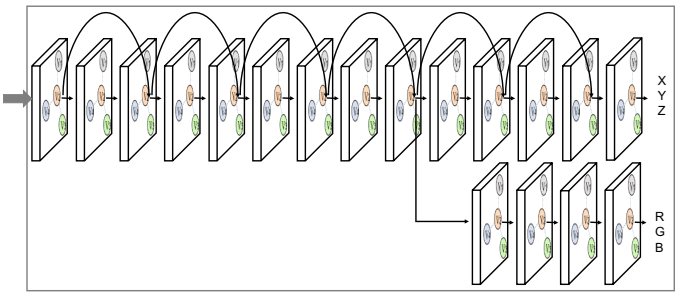


Fig. 5. The additional branch in the last block to predict the vertex property.

to maintain the relative location between neighboring vertices during deformation, and an edge length regularization to prevent outliers. These losses are applied with equal weights on both the intermediate and final meshes.

It is worth mentioning that the behavior of the model can be controlled by tuning vertex weights. For example, the chamfer loss focuses on accurate location of the 3D surface. The surface normal loss encourages high order details, like smoothness. The edge length regularization controls the vertex density. This is beneficial when adapting our model to specific category domains. We show in Section 5.4.3 that our model can successfully produce 3D models for human faces and capture distinctive facial details in important regions. We use a weight vector to construct our weighted loss function. The weight vector records the weight of each point on the ground truth points.

Unless otherwise mentioned, we use p for a vertex in the predicted mesh, q for a vertex in the ground truth mesh, $\mathcal{N}(p)$ for the neighboring vertex of p, and $\mathbf{w}(q)$ for the weight of vertex q in the ground truth, till the end of this section.

4.1 Chamfer loss

The first challenge is how to design a loss function for comparing the generated mesh vertices and the groundtruth. Inspired by [2], we use the Chamfer distance to measure the distance of each point to the other set:

$$l_c = \sum_p \min_q \|p-q\|_2^2 + \sum_q \min_p \|p-q\|_2^2. \tag{2}$$

It is reasonably good to regress the vertices close to its correct position, but is not sufficient to produce a nice 3D mesh (see the result of Fan *et al.* [2] in Fig. 1).

Considering human face reconstruction, different from objects in general categories, face exhibits abundant distinctive characteristics near central regions. Hence, we further propose a weighted chamfer loss to capture this spatial variant, which focuses on regressing the 3D position on the central region of the face, and put less attention to the head to reduce the probability of being confused by the disturbing clothes or hair.

Weighted chamfer loss can be defined as:

$$\begin{aligned} l_c = &\sum_p \min_q \|p-q\|_2^2 \cdot \underset{q=\arg\min_q \|p-q\|_2^2}{\mathbf{w}(q)} \\ &+ \sum_q \min_p \|p-q\|_2^2 \cdot \mathbf{w}(q), \end{aligned} \tag{3}$$

where the weight vector $\mathbf{w}$ records the weight of each vertex q in the ground truth; so q has the exact predefined weight value $\mathbf{w}(q)$. Since the same vertex p may have different semantic meanings in different predictions, it is non-trivial to define a weight vector in the predicted mesh. Thus, the weight of vertex p can be represented by the weight of the closest vertex for p that is found in the groundtruth, i.e., $\mathbf{w}(q)$, and $q = \arg\min_q \|p-q\|_2^2$.

4.2 Normal loss

Smoothness is an important surface property for high quality meshes, and the known local connections allow us to apply an additional constraint provided by the surface normal. To fully take advantage of the mesh representation, we further define a loss on the surface normal to characterize high order properties. Essentially, this loss requires the edge between a vertex and its neighbors to be perpendicular to the observation from the ground truth. One may find that this loss does not equal to zero unless on a planar surface. However, optimizing this loss is equivalent to forcing the normal of a locally fitted tangent plane to be consistent with the observation, which works practically well in our experiment. Moreover, this normal loss is fully differentiable and easy to optimize. Particularly, this loss has the form of,

$$l_n = \sum_p \sum_{\substack{q=\arg\min_q(\|p-q\|_2^2) \\ k\in\mathcal{N}(p)}} \|\langle p-k, \mathbf{n}_q\rangle\|_2^2, \tag{4}$$

where q is the closest vertex for p that is found when calculating the chamfer loss, k is the neighboring vertex of p, $\langle\cdot,\cdot\rangle$ is the inner product between two vectors, and $\mathbf{n}_q$ is the observed surface normal from the ground truth.

4.3 Regularization

Even though using the Chamfer loss and Normal loss, the optimization is easily stucked in some local minimum. More specifically, the network may generate some super large deformation to favor some local consistency, which is especially harmful at the beginning when the estimation is far from the ground truth, and causes flying vertices (Fig. 12).

Laplacian regularization. To handle these issues, we first propose a Laplacian term to prevent the vertices from moving too freely, which potentially avoids mesh self-intersection. The Laplaician term serves as a local detail preserving operator, which encourages neighboring vertices to have the same movement. In the first deformation block, it acts like a surface smoothness term since the input to this block is a smooth-everywhere ellipsoid; starting from the second block, it prevents the 3D mesh model from deforming too much, so that only fine-grained details are added to the mesh model. To calculate this loss, we first define a Laplacian coordinate for each vertex p as

$$\delta_p = p - \sum_{k\in\mathcal{N}(p)} \frac{1}{\|\mathcal{N}(p)\|} k, \tag{5}$$

and the Laplacian regularization is defined as

$$l_{lap} = \sum_p \|\delta_p' - \delta_p\|_2^2, \tag{6}$$

where δ'_p and δ_p are the Laplacian coordinates of a vertex after and before a deformation block.

Edge length regularization. To penalize flying vertices, which ususally cause long edges, we add an edge length regularization loss:

$$l_{loc} = \sum_{p} \sum_{k \in \mathcal{N}(p)} \|p - k\|_2^2. \tag{7}$$

When reconstructing a human face, central regions areas require more vertices to reconstruct comparing to the other areas; thus we define a weighted edge length regularization to control vertex density; it forces a shorter edge for a region with higher weight, which is equivalent to gathering more vertices locally for better details.

Weighted edge length regularization can be defined as:

$$l_{loc} = \sum_{p} \sum_{k \in \mathcal{N}(p)} \|p - k\|_2^2 \cdot \frac{\mathbf{w}(q)}{q = \arg\min_q \|p-q\|_2^2}. \tag{8}$$

For the vertex p in the predicted mesh, we use the weight of the closest vertex for p to represent its own weight. All the edges connected by p share the same weight in calculating the edge length.

4.4 Property loss

Our model can have multiple parallel branches in the last block to produce not only the 3D location of each vertex but also other per-pixel properties, e.g., vertex color. Given the ground truth of the properties on all vertices, we define the property loss as:

$$l_{prop} = \sum_{p} \|\mathbf{P}_p - \mathbf{P}_q\|_2^2, \tag{9}$$

where q is the closest vertex for p when calculating the chamfer loss, and $\mathbf{P}_p$ and $\mathbf{P}_q$ are the properties estimated for p and observed on q.

The overall loss is a weighted sum of all four losses, $l_{all} = l_c + \lambda_1 l_n + \lambda_2 l_{lap} + \lambda_3 l_{loc} + \lambda_4 l_{prop}$, where $\lambda_1 = 1.6e{-}4$, $\lambda_2 = 0.3$ and $\lambda_3 = 0.1$ are the hyperparameters which balance the losses and are fixed for all the experiments by default. λ_4 is set to zero unless a property estimation is needed.

5 Experiments

In this section, we perform an extensive evaluation on our model. In addition to comparing with previous 3D shape generation works for evaluating the reconstruction accuracy, we also analyze the importance of each component in our model. Qualitative results on both synthetic and real-world images further show that our model produces triangular meshes with smooth surfaces and still maintains details depicted in the input images.

5.1 Experimental setup

5.1.1 Dataset.

ShapeNet. We use the dataset provided by Choy *et al.* [1]. The dataset contains rendering images of 50k models belonging to 13 object categories from ShapeNet [18], which is a collection of 3D CAD models that are organized according to the WordNet hierarchy. A model is rendered from various camera viewpoints, and camera intrinsic and extrinsic matrices are recorded. More specifically, four views are rendered from each mesh via randomly sampled viewpoints. For fair comparison, we use the same training/testing split as in Choy *et al.* [1]. The mesh models are transformed in camera coordinates based on the camera parameters from the 3D-R2N2 [1]; the 3D shapes are downscaled by a factor of 0.57 to generate rendering.

Online Products. The dataset [48] is composed of 23,000 images of items sold online without ground-truth 3D CAD models. We perform qualitative evaluation on this dataset to show the efficacy of our model on the real world images. Particularly, our model is trained on the 13 shapenet categories, and is directly applied to this dataset for qualitative evaluation. Quantitative evaluation is infeasible since the ground truth 3D shape is not provided.

3D Face Dataset. Our model is also tested on the 3D face reconstruction task. Particularly, our model is trained on the large pose 300W-LP dataset [42], which contains more than 6K unconstrained face images and 3D face meshes. The ground truth meshes are generated by fitting 3DMM [36] parameters based on Basel Face Model (BFM) [49], which only contain the face regions of a head. We further complete the ground truth face regions into water-tight full head models using the method from [42] to make them suitable for our method.

5.1.2 Evaluation Metric.

We adopt the standard 3D reconstruction metric. We first uniformly sample points from our result and ground truth. We calculate precision and recall by checking the percentage of points in prediction or ground truth that can find a nearest neighbor from the others within certain threshold τ. An F-score [50] as the harmonic mean of precision and recall is then calculated. Following Fan *et al.* [2], we also report the Chamfer Distance (CD) and Earth Mover's Distance (EMD). For F-Score, larger is better. For CD and EMD, smaller is better.

On the other hand, we realize that the commonly used evaluation metrics for shape generation may not thoroughly reflect the shape quality. They often capture occupancy or point-wise distance rather than surface properties, such as continuity, smoothness, and high-order details, for which a standard evaluation metric is barely missing in the literature. Thus, we recommend to pay attention to qualitative results for better understanding of these aspects.

5.1.3 Baselines.

We compare the presented approach to the most recent single image reconstruction approaches. Specifically, we compare with two state-of-the-art methods - Choy et al. [1] (3D-R2N2) producing 3D volume, and Fan et al. [2] (PSG) producing point cloud. Since the metrics are defined on point cloud, we can evaluate PSG directly on its output, our method by uniformly sampling point on surface, and 3D-R2N2 by uniformly sampling point from mesh created using the Marching Cube [51] method.

Category	F-score(τ)↑				F-score(2τ)↑			
	3DR2N2	PSG	N3MR	Ours	3DR2N2	PSG	N3MR	Ours
plane	41.46	68.20	62.10	**71.12**	63.23	81.22	77.15	**81.38**
bench	34.09	49.29	35.84	**57.57**	48.89	69.17	49.58	**71.86**
cabinet	49.88	39.93	21.04	**60.39**	64.83	67.03	35.16	**77.19**
car	37.80	50.70	36.66	**67.86**	54.84	77.79	53.93	**84.15**
chair	40.22	41.60	30.25	**54.38**	55.20	63.70	44.59	**70.42**
monitor	34.38	40.53	28.77	**51.39**	48.23	63.64	42.76	**67.01**
lamp	32.35	41.40	27.97	**48.15**	44.37	58.84	39.41	**61.50**
speaker	45.30	32.61	19.46	**48.84**	57.86	56.79	32.20	**65.61**
firearm	28.34	69.96	52.22	**73.20**	46.87	82.65	63.28	**83.47**
couch	40.01	36.59	25.04	**51.90**	53.42	62.95	39.90	**69.83**
table	43.79	53.44	28.40	**66.30**	59.49	73.10	41.73	**79.20**
cellphone	42.31	55.95	27.96	**70.24**	60.88	79.63	41.83	**82.86**
watercraft	37.10	51.28	43.71	**55.12**	52.19	**70.63**	58.85	69.99
mean	39.01	48.58	33.80	**59.72**	54.62	69.78	47.72	**74.19**

TABLE 1
F-score (%) on the ShapeNet test set at different thresholds, where $\tau = 10^{-4}$. For F-score, larger is better. Best results under each threshold are bolded.

Category	CD↓					EMD↓				
	3DR2N2	PSG	N3MR	AtlasNet	Ours	3DR2N2	PSG	N3MR	AtlasNet	Ours
plane	0.895	**0.430**	0.450	0.468	0.477	0.606	**0.396**	7.498	0.659	0.579
bench	1.891	0.629	2.268	0.703	**0.624**	1.136	1.113	11.766	1.204	**0.965**
cabinet	0.735	0.439	2.555	0.433	**0.381**	2.520	2.986	17.062	**2.503**	2.563
car	0.845	0.333	2.298	0.340	**0.268**	1.670	1.747	11.641	1.407	**1.297**
chair	1.432	0.645	2.084	0.724	**0.610**	1.466	1.946	11.809	1.534	**1.399**
monitor	1.707	**0.722**	3.111	0.848	0.755	1.667	1.891	14.097	1.641	**1.536**
lamp	4.009	**1.193**	3.013	1.575	1.295	1.424	1.222	14.741	1.35	**1.314**
speaker	1.507	0.756	3.343	0.812	**0.739**	**2.732**	3.490	16.720	3.108	2.951
firearm	0.993	**0.423**	2.641	0.461	0.453	0.688	**0.397**	11.889	0.735	0.667
couch	1.135	0.549	3.512	0.621	**0.490**	2.114	2.207	14.876	1.91	**1.642**
table	1.116	0.517	2.383	0.577	**0.498**	1.641	2.121	12.842	1.673	**1.480**
cellphone	1.137	0.438	4.366	0.443	**0.421**	0.912	1.019	17.649	0.868	**0.724**
watercraft	1.215	**0.633**	2.154	0.839	0.670	0.935	0.945	11.425	0.907	**0.814**
mean	1.445	0.593	2.629	0.680	**0.591**	1.501	1.653	13.386	1.500	**1.380**

TABLE 2
CD and EMD on the ShapeNet test set at different thresholds. For CD and EMD, small is better. Best results under each threshold are bolded.

Method.	F-score(τ)↑	F-score(2τ)↑	CD↓	EMD↓
Tartarchenko et al	65.335	79.733	0.361	1.273
Ours	**72.128**	**87.247**	**0.236**	**1.220**

TABLE 3
Comparsion to Tartarchenko et al.(128^3) on ShapeNet-cars.

We further compare to other works which also produce mesh as 3D representation; these include Neural 3D Mesh Renderer (N3MR) [29], and AtlasNet [52], since their codes are publically available and their performances are competitive in the literature. For fair comparison, the models are trained with the same data using the same number of epochs.

5.1.4 Training Details and Runtime.

Our network receives input images of size 224×224, and an initial ellipsoid with 156 vertices and 462 edges. The network is implemented in Tensorflow and optimized using Adam with weight decay 1e-5. The batch size is 1; the total number of training epochs is 50; the learning rate is initialized as 3e-5 and drops to 1e-5 after 40 epochs. The total training time is 72 hours on a NVidia Titan X. During testing, our model takes 15.58ms on average to generate a mesh with 2466 vertices.

5.2 Comparison to state of the arts

Table 1 shows F-scores with different thresholds of different methods. Our approach outperforms the other methods in all categories except watercraft. Notably, our results are significantly better than the others in all categories under a

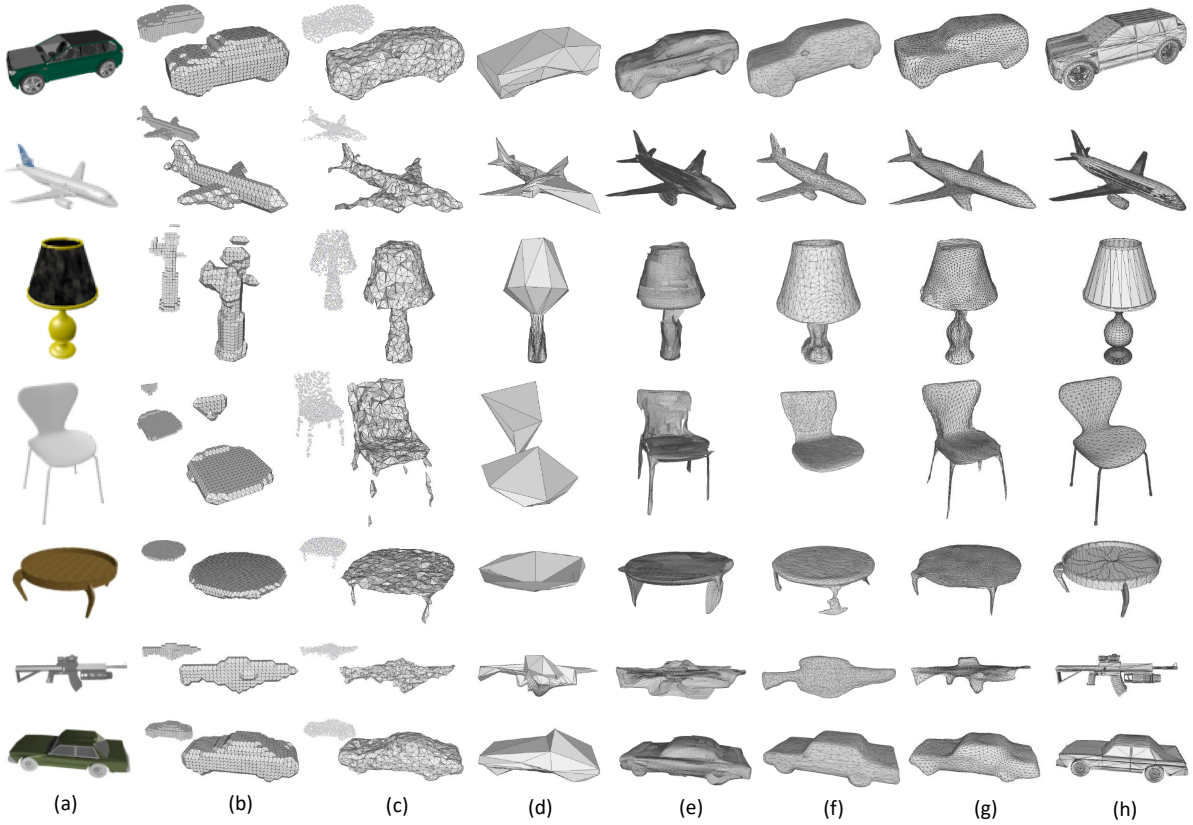


Fig. 6. Qualitative results. (a) Input image; (b) volume from 3D-R2N2 [1], converted using Marching Cube [51]; (c) point cloud from PSG [2], converted using ball pivoting [53]; (d) N3MR [29]; (e) AtlasNet [52]; (f) ONet [28]; (g) ours; (h) ground truth.

Fig. 7. Qualitative results of real-world images from the Online Products dataset and Internet.

Fig. 8. Ground truth meshes (top) and failure cases of Pixel2Mesh (bottom).

smaller threshold τ, showing at least 10% F-score improvement. N3MR does not perform well, and its result is about 50% worse than ours, probably because their model only learns from limited silhouette signals in images and lacks of explicit handling of the 3D mesh.

We also show the CD and EMD for all categories in Table 2. Our approach outperforms most of the other methods in most of the categories and achieves the best mean performance. The major competitor is PSG [2], which produces results in 3D point clouds. The point cloud representation has more flexibility and capacity in exhibiting geometry details, but does not necessarily leads to a better mesh model without proper postprocessing. To demonstrate this, we show the qualitative results in Fig. 6. To compare the quality of the mesh model, we convert volumetric and point cloud to mesh using standard approaches [51], [53]. As we can see, the 3D volume results produced by 3D-R2N2 lack of details due to the low resolution, *e. g.*, the legs are missing in the chair example as shown in the 4-th row of Fig. 6. We further try the octree based solution [23] to increase the volume resolution, but find that it is still hard to recover surface level details as much as our model. PSG produces

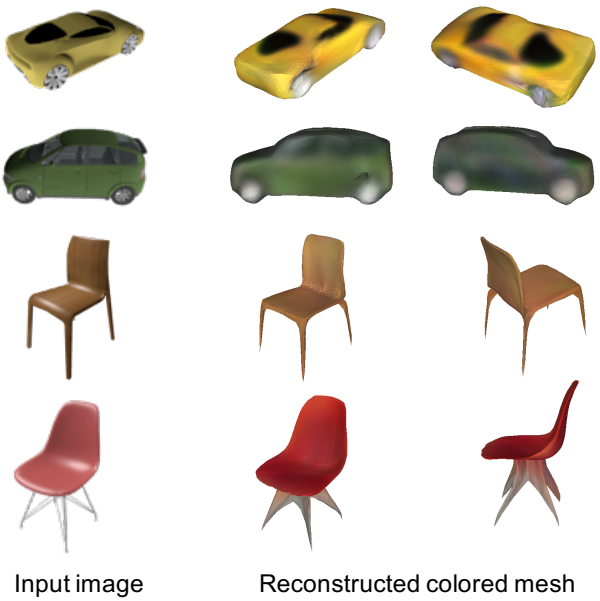


Fig. 9. Visualization of predicted colored meshes.

sparse 3D point clouds that produce relatively smaller error in Chamfer distance. However, the point clouds do not necessarily produce mesh with noise-free surface due to the lack of surface related loss function (e.g., on surface normal). For methods producing 3D mesh, N3MR produces very rough shape, which may be sufficient for some rendering tasks, but cannot recover complicated objects such as chairs and tables. We also test AtlasNet [52] with 25 patches and 2500 points, which adds relatively more details but still contains strong artifacts and holes. The ONet [28] is one of the most state-of-the-arts of this topic, published in CVPR 2019. We directly run the released trained model from the ONet authors to produce the meshes in Fig. 6. It can be seen that ONet can produce reasonable meshes with smooth surface details (e.g., the car and airplane in Fig. 6 (f))), but occasionally ignores some small parts of objects, e.g., the legs of chair. While producing a reasonably good result, Pixel2Mesh typically fails when the target geometry cannot be represented by a genus-0 mesh, such as the lamp and chair example in Fig. 8.

5.3 Comparison to Octree based Voxel Generation

A promising direction to improve the shape accuracy and detail is to increase the spatial resolution of the generated 3D shape. Here we compare to an Octree based Voxel Reconstruction method [23], which produces high resolution voxels, i.e., 128^3. Following Tartarchenko *et al.* [23], we train on Shapenet-car and compare to it as shown in Tab. 3. Our model consistently outperforms the octree based approach in all the evaluation metrics. This is presumable because that high spatial resolution only provides the chance of encoding better geometry, but does not trivially allow high order losses, like normal loss, to directly learn geometry details.

5.4 Model Analysis

In this section, we verify the capability of our model in generalizing to real-world images, category in specific domains, and predicting additional per-vertex properties.

5.4.1 Reconstructing Real-World Images

ShapeNet is a large dataset of 3D CAD models without corresponding real-world images. During training, the images are obtained by rendering using off-the-shelve physically based renderers. One reasonable concern is that models trained on synthetic images may suffer from the domain gap issue when testing on real-world images. To fill the domain gap, several 3D object datasets, which contain real images and associated 3D models, have been released; these datasets include Pascal 3D+, ObjectNet3D, Pix3D [54] and IKEA. However, they all have their limitations. For Pascal 3D+ and ObjectNet3D, the groudtruth 3D models are not reconstructed but manually selected from ShapeNet and aligned with the image object, so the 3D shape and position are not accurate, which might be sufficient for the low-resolution voxel reconstruction method, but not suitable for mesh or point cloud reconstructions. On the other hand, IKEA has well aligned image-shape pairs, but it only has 759 images and 90 3D models, relatively small to train our model from scratch. In addition, the images in IKEA dataset usually contain multiple objects that are not genus-0 with mutual occlusion on heavy background.

Thus we evaluate the generalization of our model trained on ShapeNet without fine-tuning and test our network on the Online Products dataset and Internet images for qualitative evaluation on real images. The results are shown in Fig. 7. As can be seen, our model trained on synthetic data generalizes well to the real-world images (better with white background) across various categories.

5.4.2 Generating Colored Mesh

Here we show that our model can predict additional per-vertex properties, taking vertex color as an example. The task here is to predict the RGB value for each vertex on the generated mesh. As shown in Fig. 5, we attach an additional branch after the last deformation block to predict a 3 dimensional feature on each vertex. Since ShapeNet provides textured 3D models, the color prediction task can be learned in a fully supervised fashion.

Fig. 9 shows results of our generated mesh with predicted color. As can be seen, our model not only learns to recover the color for vertices visible from the input image viewpoint, but also hallucinates reasonable color for invisible vertices, e.g., vertices occluded in the back. Notice that fulfilling this task is non-trivial as it requires knowledge of geometry symmetry (e.g., car windows) and object part semantics (e.g., chair top and leg).

5.4.3 3D Face Reconstruction

Here we show that our model can easily adapt to shapes of a specific domain by tuning the loss weights, taking 3D face reconstruction as an example. We adjust our models to favor the shape generation for human face in the following aspects. First, face is a relatively complex model and requires more vertices to capture details. Therefore, we use 4 blocks, which produce 9858 vertices. Second, faces in dramatically different poses result in large shape variance and make convergence harder. To make the whole

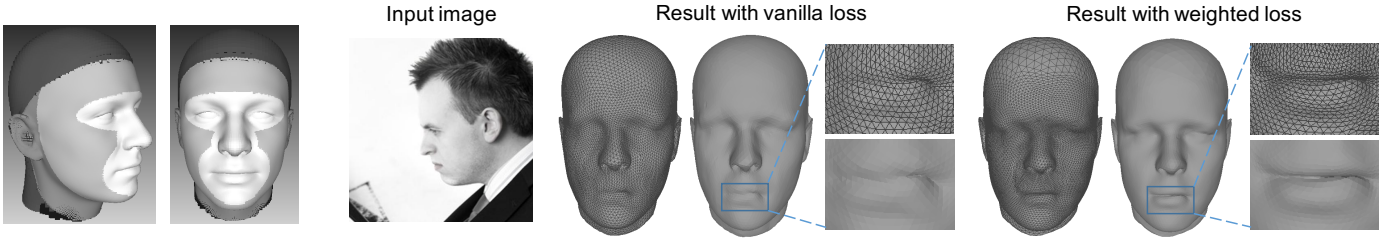


Fig. 10. (a) The ground truth face points is subdivided into 4 regions, central region has larger weight. (b)The result with weighted loss lead to more vertices in central region and fine-grained details.

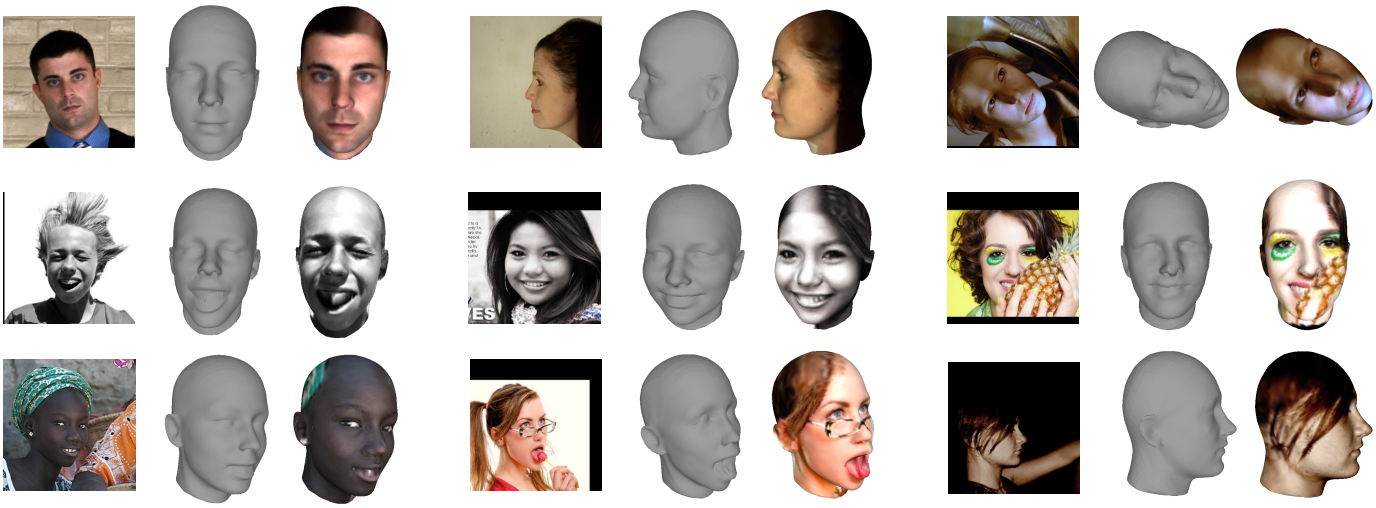


Fig. 11. Visualization of predictions on the AFLW2000-3D dataset. Even designed for generic shape generation, our method adapts to specific human face domain naturally and recover good geometry with distinctive facial details. Our model also is not sensitive to pose alignment.

deformation procedure more stable, during training, we adopt a more aggressive coarse-to-fine strategy by starting with a strong Laplacian regularization to recover global alignment and then gradually decreasing it to capture more local details. Lastly, different from objects in general categories, face exhibits abundant distinctive characteristics near central regions, and therefore these areas require more vertices to reconstruct comparing to the other areas. To fulfill this spatial variant requirement, as Fig. 10 (a) shows, we subdivide ground truth points into four regions and assign each a different weight when computing the loss. From dark to bright, we assign regions with increasing weights, i.e., 0.25, 0.5, 1.5, and 2, to reflect their raising importances. The weights are applied on each vertex in the region when calculating the chamfer loss and edge-length loss. We use a weight vector to construct our weighted loss function, and the weight vector records the weight of each point on the ground truth points. The weighted chamfer loss focuses on regressing the 3D position on the central region of the face, and puts less attention to the head to reduce the probability of being confused by the disturbing clothes or hair. The weighted edge length regularization forces the shorter edge for region with higher weight, which is equivalent to gathering more vertices locally for better details.

As shown in Fig. 10 (b), the model with vanilla loss generates uniform distributed vertices as it treats all edges equally when regularizing the edge length. In contrast, the result with the weighted loss has more vertices in the central face region than the back-head region, and produces fine-grained details in the key regions, e.g., mouth. As shown in Fig. 11, our method works well on human faces, even with large pose and expression.

Quantitatively, we compare to the state-of-the-art face reconstruction models, including VRN [45], VRN-Multitask [45], VRN-Guided [45], 3DDFA [42], and EOS [55], on the AFLW2000-3D dataset [42] with the standard data split. We employ the Normalized Mean Error (NME) as the evaluation metric as it has been widely used in literature. The results are shown in Table 5. It can be seen that the performance of our model is competitive among the state-of-the-art face reconstruction models even though our model is designed for generic objects with no specific optimization for human face.

5.5 Ablation Study

Now we conduct controlled experiments to analyze the importance of each component in our model. Table 4 reports the performance of each model by removing one component from the full model. Again, we argue that these commonly used evaluation metrics do not necessarily reflect the quality of the recovered 3D geometry. For example, the model with

no edge length regularization achieves the best performance across all; however, it in fact produces the worst mesh (Fig. 12, the last 2nd column). As such, we use the qualitative result in Fig. 12 to show the contribution of each component in our system.

5.5.1 Graph Unpooling

We first remove the graph unpooling layers, and thus each block has the same number of vertices as in the last block of our full model. It is observed that the deformation makes a mistake easier at beginning, which cannot be fixed later on. Consequently, there are some obvious artifacts in some parts of the objects.

5.5.2 G-ResNet

We then remove the shortcut connections in G-ResNet, and make it regular GCN. As can be seen from Table 4, there is a huge performance gap in all four measurement metrics, which implies the failure of optimizing Chamfer distance. The main reason is the degradation problem observed in the very deep 2D convolutional neural network. Such a problem leads to a greater training error (and thus a greater testing error) when adding more layers to a suitably deep model [10]. Essentially, our network has 42 graph convolutional layers. Thus, this phenomenon has also been observed in our very deep graph neural network experiment.

5.5.3 Loss Terms

We evaluate the function of each additional term besides the Chamfer loss. As can be seen in Fig. 12, removing the normal loss severely impairs the surface smoothness and local details, e.g., seat back; removing the Laplacian term causes intersecting geometry because the local topology changes, e.g., the hand held of the chair; removing the edge length term causes flying vertices and surfaces, which completely ruins the surface characteristics. These results demonstrate that all the components presented in this work contribute to the final performance.

5.5.4 Number of Deformation Blocks

We now analyze the effects of the number of blocks. Figure Fig. 13 (left) shows the mean F-score(τ) and CD with regard to the number of blocks. The results indicate that increasing the number of blocks helps, but the benefit is getting saturated with more blocks, e.g., from 3 to 4. In our experiments, we find that 4 blocks result in too many vertices and edges, which slows down our approach dramatically even though it provides better accuracy in evaluation metrics. Therefore, we use 3 blocks in all our experiment for the best balance of performance and efficiency.

6 Conclusion

We presented an approach to extract 3D triangular meshes from a singe RGB image. We exploited the key advantages the mesh presentation can bring to us, and the key issues required to be solved for success. The former includes surface normal constraints and information propagation along edges; the latter includes perceptual features extracted from images as a guidance. We proposed a very deep cascaded graph convolutional neural network with "shortcut" connections. Meshes are progressively refined by our network trained end-to-end with the chamfer loss and normal loss, together with important regularization losses. Our results are significantly better than the previous state-of-the-art methods using 3D volume or 3D point cloud as the representation. We would believe that mesh representation is the next big thing in this direction, and hope that the key components discovered in this work can inspire follow-up works.

Future work Our method only produces meshes with the same topology as the initial mesh. In the future, we will extend our approach to more general cases, such as scene level reconstruction, and learn from multiple images for multi-view reconstruction [56].

Acknowledgments

This work was supported in part by NSFC Projects (U1611461, 61702108), Science and Technology Commission of Shanghai Municipality Projects (19511120700, 19ZR1471800, 19511132000), Shanghai Research and Innovation Functional Program (17DZ2260900), and Shanghai Municipal Science and Technology Major Project (2018SHZDZX01). Dr. Yanwei Fu is supported by Eastern Scholar (TP2017006).

References

[1] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *ECCV*, 2016.
[2] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *CVPR*, 2017.
[3] M. Waechter, M. Beljan, S. Fuhrmann, N. Moehrle, J. Kopf, and M. Goesele, "Virtual rephotography: Novel view prediction error for 3d reconstruction," in *ACM Transactions on Graphics*, 2017.
[4] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *CVPR*, 2016.
[5] Z. Li, Q. Chen, and V. Koltun, "Interactive image segmentation with latent diversity," in *CVPR*, 2018.
[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, 2017.
[7] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016, pp. 3837–3845.
[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2016.
[9] Z.-H. Zhou, "Abductive learning: towards bridging machine learning and logical reasoning," *Science China Information Sciences*, vol. 62, no. 7, p. 76101, 2019.
[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
[11] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *ECCV*, 2018.
[12] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.
[13] J. L. Schönberger and J. Frahm, "Structure-from-motion revisited," in *CVPR*, 2016.
[14] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.
[15] H. Jin, S. Soatto, and A. J. Yezzi, "Multi-view stereo reconstruction of dense shape and complex appearance," *International Journal of Computer Vision*, vol. 63, no. 3, pp. 175–189, 2005.

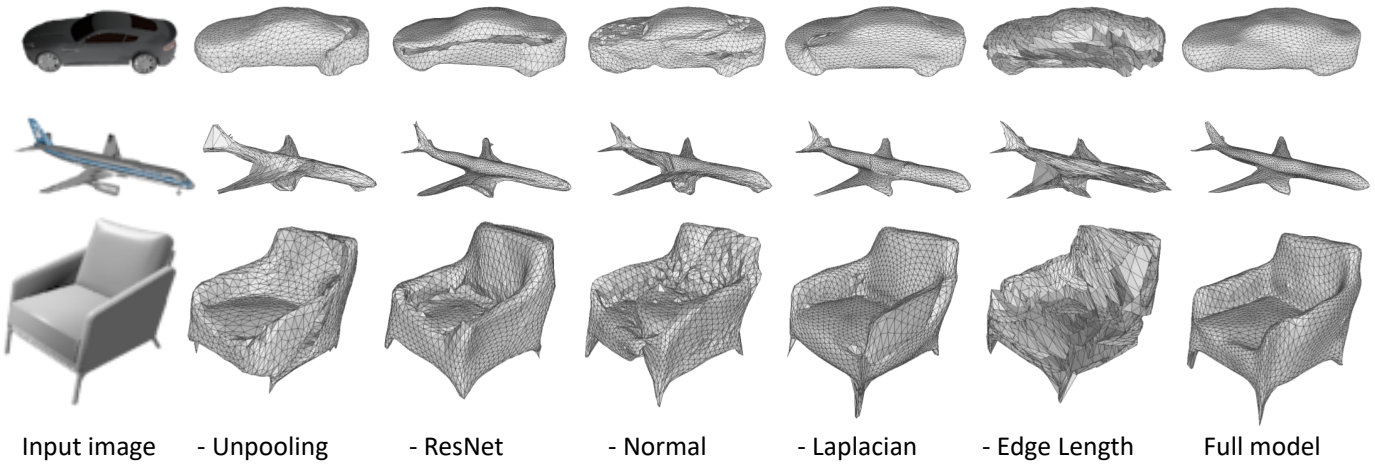


Fig. 12. Qualitative results for ablation study. Each column shows the results from the model trained with the corresponding model component disabled. The results reflect the contribution of each component especially for the regularization ones.

Category	-ResNet	-Laplacian	-Unpooling	-Normal	-Edge length	Full model
F (τ)↑	55.308	60.801	60.222	58.668	60.101	59.728
F (2τ)↑	71.567	75.202	76.231	74.276	76.053	74.191
CD↓	0.644	0.596	0.561	0.598	0.552	0.591
EMD↓	1.583	1.350	1.656	1.445	1.479	1.380

TABLE 4
Ablation study that evaluates the contributions of different ideas to the performance of the presented model. The table reports all 4 measurements. For F-score, larger is better. For CD and EMD, smaller is better.

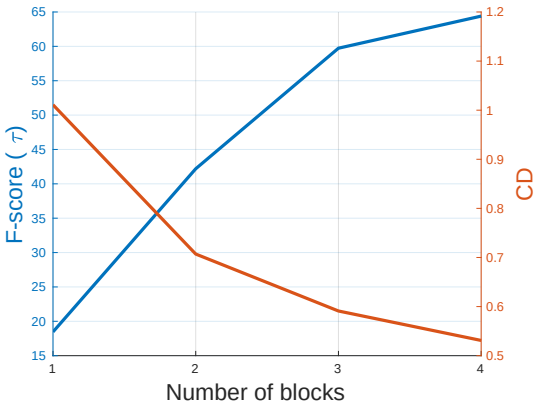


Fig. 13. Left: Effect of number of blocks. Each curve shows the mean F-score (τ) and CD for different numbers of blocks. Right: Sample examples showing the output after each block.

Method	VRN	VRN-Multitask	VRN-Guided	3DDFA	EOS	Ours
NME	0.0676	0.0698	0.0637	0.1012	0.0971	0.0656

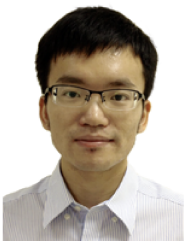
TABLE 5
Comparsion to state-of-the-art face reconstruction methods using Normalised Mean Error (NME) on the AFLW2000-3D dataset.

[16] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 151–172, 2007.
[17] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, 2009.
[18] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.
[19] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, "Pix3d: Dataset and methods for single-image 3d shape modeling," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.
[20] Q. Huang, H. Wang, and V. Koltun, "Single-view reconstruction via joint analysis of image and shape collections," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 87:1–87:10, 2015.
[21] H. Su, Q. Huang, N. J. Mitra, Y. Li, and L. J. Guibas, "Estimating image depth using shape collections," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 37:1–37:11, 2014.
[22] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *ECCV*, 2016.
[23] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs," in *ICCV*, 2017.
[24] A. Kar, S. Tulsiani, J. Carreira, and J. Malik, "Category-specific object reconstruction from a single image," in *CVPR*, 2015.
[25] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani, "Surfnet: Generating 3d shape surfaces using deep residual networks," in *CVPR*, 2017.
[26] S. R. Richter and S. Roth, "Matryoshka networks: Predicting 3d geometry via nested shape layers," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.
[27] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox, "What do single-view 3d reconstruction networks learn?" in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[28] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[29] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," in *CVPR*, 2018.

[30] J. K. Pontes, C. Kong, S. Sridharan, S. Lucey, A. Eriksson, and C. Fookes, "Image2mesh: A learning framework for single image 3d reconstruction," Tech. Rep. arXiv:1711.10669 [cs.CV], 2017.
[31] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.
[32] L. Yi, H. Su, X. Guo, and L. J. Guibas, "Syncspeccnn: Synchronized spectral CNN for 3d shape segmentation," in *CVPR*, 2017.
[33] D. Boscaini, J. Masci, E. Rodolà, and M. M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *NIPS*, 2016.
[34] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *ICCV Workshop*, 2015.
[35] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *CVPR*, 2017.
[36] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999, pp. 187–194.
[37] X. Zhu, Z. Lei, J. Yan, D. Yi, and S. Z. Li, "High-fidelity pose and expression normalization for face recognition in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 787–796.
[38] Y. J. Lee, S. J. Lee, K. R. Park, J. Jo, and J. Kim, "Single view-based 3d face reconstruction robust to self-occlusion," *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, p. 176, 2012.
[39] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2387–2395.
[40] P. Huber, G. Hu, R. Tena, P. Mortazavian, P. Koppen, W. J. Christmas, M. Ratsch, and J. Kittler, "A multiresolution 3d morphable face model and fitting framework," in *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.
[41] A. Jourabloo and X. Liu, "Large-pose face alignment via cnn-based dense 3d model fitting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4188–4196.
[42] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment in full pose range: A 3d total solution," *computer vision and pattern recognition*, vol. 41, no. 1, pp. 146–155, 2016.
[43] A. T. Tran, T. Hassner, I. Masi, and G. Medioni, "Regressing robust and discriminative 3d morphable models with a very deep neural network," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 1493–1502.
[44] P. Dou, S. K. Shah, and I. A. Kakadiaris, "End-to-end 3d face reconstruction with deep neural networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 21–26.
[45] A. S. Jackson, A. Bulat, V. Argyriou, and G. Tzimiropoulos, "Large pose 3d face reconstruction from a single image via direct volumetric cnn regression," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 1031–1039.
[46] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou, "Joint 3d face reconstruction and dense alignment with position map regression network," in *ECCV*, 2018.
[47] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: an open-source mesh processing tool," in *Eurographics Italian Chapter Conference*, 2008.
[48] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.
[49] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3d face model for pose and illumination invariant face recognition," in *Advanced video and signal based surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, 2009, pp. 296–301.
[50] A. Knapitsch, J. Park, Q. Zhou, and V. Koltun, "Tanks and temples: benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 78:1–78:13, 2017.
[51] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *SIGGRAPH*, 1987.
[52] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "Atlasnet: A papier-mâché approach to learning 3d surface generation," *arXiv preprint arXiv:1802.05384*, 2018.
[53] F. Bernardini, J. Mittleman, H. E. Rushmeier, C. T. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Trans. Vis. Comput. Graph.*, vol. 5, no. 4, pp. 349–359, 1999.
[54] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, "Pix3d: Dataset and methods for single-image 3d shape modeling," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
[55] P. Huber, G. Hu, R. Tena, P. Mortazavian, W. Koppen, W. Christmas, M. Rtsch, and J. Kittler, "A multiresolution 3d morphable face model and fitting framework," in *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.
[56] ChaoWen, Y. Zhang, Z. Li, and Y. Fu, "Pixel2mesh++: Multi-view 3d mesh generation via deformation," in *ICCV*, 2019.

Nanyang Wang received the Master degree in computer science from the School of Computer science, Fudan University, Shanghai, China. He is currently working at Alibaba DAMO Academy. His research is focused on 3D mesh reconstruction, non-rigid registration and extreme multi-label classification.

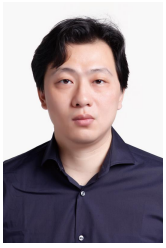
Yinda Zhang is a Researsh Scientist at Google. His research interests lie at the intersection of computer vision, computer graphics, and machine learning. Recently, he is actively working on empowering 3D vision and percetion via machine learning, including dense depth estimation, 3D shape analysis, and 3D scene understanding. He received Ph.D. in Computer Science from Princeton University. Before that, he received a Bachelor degree from Tsinghua University in Beijing China, and a Master degree from National University of Singapore.



Zhuwen Li received the BE degree in computer science from Tianjin University, in 2008, the masters degree in computer science from Zhejiang University, in 2011, and the PhD degree from the Department of Electrical and Computer Engineering, National University of Singapore, in 2014. Currently, he is a Research Scientist at Nuro, Inc. Recently, he is working on perception in autonomous driving, specifically in 3D structure recovery, motion analysis, point cloud analysis and object detection.

Yanwei Fu received the Ph.D. degree from Queen Mary University of London in 2014, and the M.Eng. degree from the Department of Computer Science and Technology, Nanjing University, China, in 2011. He held a post-doctoral position at Disney Research, Pittsburgh, PA, USA, from 2015 to 2016. His work has led to many awards, including Discovery Early Career Researcher Award (DECRA), Australia in 2016, ACM China Rising Star 2017, IEEE ICME 2019 best paper award and Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning. He published more than 50 journal/conference papers including IEEE TPAMI, TMM, ECCV, and CVPR. He had filed more than 30 patents in China and 5 patents in US. He is currently a tenure-track Professor with Fudan University. His research interests are image and video understanding, few-shot and zero-shot learning, and learning based 3D reconstruction.

Hang Yu received the Master degree in the Department of Information Science & Technology at Beijing Normal University in 2018, and he is currently working toward the PhD degree in Academy for Engineering & Technology, Fudan University. His research interest is computer vision and 3D Vision.



Wei Liu (M'14-SM'19) is currently a Distinguished Scientist of Tencent, China and a director of Computer Vision Center at Tencent AI Lab. Prior to that, he has been a research staff member of IBM T. J. Watson Research Center, Yorktown Heights, NY, USA from 2012 to 2015. Dr. Liu has long been devoted to research and development in the fields of machine learning, computer vision, pattern recognition, information retrieval, big data, etc. Dr. Liu currently serves on the editorial boards of IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Circuits and Systems for Video Technology, Pattern Recognition, Neurocomputing, etc. He is an Elected Member of the International Statistical Institute (ISI) and a Fellow of the British Computer Society (BCS).

Xiangyang Xue received the BS, MS, and PhD degrees in communication engineering from Xidian University, Xian, China, in 1989, 1992, and 1995, respectively. He is currently a professor of computer science with Fudan University, Shanghai, China. His research interests include multimedia information processing and machine learning.

Yu-Gang Jiang received the PhD degree in Computer Science from City University of Hong Kong in 2009 and worked as a Postdoctoral Research Scientist at Columbia University, New York during 2009-2011. He is currently Professor and Dean at School of Computer Science, Fudan University, Shanghai, China. His research lies in the areas of multimedia, computer vision and AI security. His work has led to many awards, including the inaugural ACM China Rising Star Award, the 2015 ACM SIGMM Rising Star Award, and the research award for excellent young scholars from NSF China.