Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images

Nanyang Wang[1⋆], Yinda Zhang[2⋆], Zhuwen Li[3⋆],
Yanwei Fu[4], Wei Liu[5], Yu-Gang Jiang[1†]

[1]Shanghai Key Lab of Intelligent Information Processing,
School of Computer Science, Fudan University
[2]Princeton University [3]Intel Labs [4]School of Data Science, Fudan University [5]Tencent AI Lab
nywang16@fudan.edu.cn yindaz@cs.princeton.edu lzhuwen@gmail.com
yanweifu@fudan.edu.cn wl2223@columbia.edu ygj@fudan.edu.cn

Abstract. We propose an end-to-end deep learning architecture that produces a 3D shape in triangular mesh from a single color image. Limited by the nature of deep neural network, previous methods usually represent a 3D shape in volume or point cloud, and it is non-trivial to convert them to the more ready-to-use mesh model. Unlike the existing methods, our network represents 3D mesh in a graph-based convolutional neural network and produces correct geometry by progressively deforming an ellipsoid, leveraging perceptual features extracted from the input image. We adopt a coarse-to-fine strategy to make the whole deformation procedure stable, and define various of mesh related losses to capture properties of different levels to guarantee visually appealing and physically accurate 3D geometry. Extensive experiments show that our method not only qualitatively produces mesh model with better details, but also achieves higher 3D shape estimation accuracy compared to the state-of-the-art.

Keywords: 3D shape generation · Graph convolutional neural network · Mesh reconstruction · Coarse-to-fine · End-to-end framework

1 Introduction

Inferring 3D shape from a single perspective is a fundamental human vision functionality but is extremely challenging for computer vision. Recently, great success has been achieved for 3d shape generation from a single color image using deep learning techniques [6, 9]. Taking advantage of convolutional layers on regular grids or multi-layer perception, the estimated 3D shape, as the output of the neural network, is represented as either a volume [6] or point cloud [9]. However, both representations lose important surface details, and is non-trivial to reconstruct a surface model (Fig. 1), i.e. a mesh, which is more desirable for many real applications since it is lightweight, capable of modelling shape details, easy to deform for animation, to name a few.

In this paper, we push along the direction of single image reconstruction, and propose an algorithm to extract a 3D triangular mesh from a single color image. Rather

⋆ indicates equal contributions.
† indicates corresponding author.

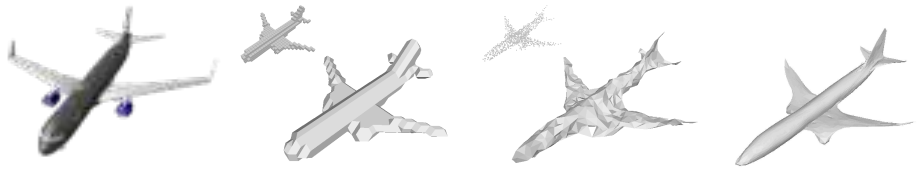

Fig. 1. Given a single color image and an initial mesh, our method can produce a high-quality mesh that contains details from the example.

than directly synthesizing, our model learns to deform a mesh from a mean 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 in detail. It also enables the control of the trade-off between the complexity of the deep learning model and the quality of the result. Lastly, it provides the chance to encode any prior knowledge to the initial mesh, e.g. topology. As a pioneer study, in this work, we specifically work on objects that can be approximated using 3D mesh with genus 0 by deforming an ellipsoid with a fixed size. In practice, we found most of the commonly seen categories can be handled well under this setting, e.g. car, plane, table, etc. To achieve this goal, there are several inherent challenges.

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 a given color image represented in a 2D regular grid. It requires the integration of the knowledge learned from two data modalities. On the 3D geometry side, we directly build a graph based fully convolutional network (GCN) [3, 8, 18] on the mesh model, where the vertices and edges in the mesh are directly represented as nodes and connections in a graph. Network feature encoding information for 3D shape is saved on each vertex. Through forward propagation, the convolutional layers enable feature exchanging across neighboring nodes, and eventually regress the 3D location for each vertex. On the 2D image side, we use a VGG-16 like architecture to extract features as it has been demonstrated to be successful for many tasks [10, 20]. To bridge these two, we design a perceptual feature pooling layer which allows each node in the GCN to pool image features from its 2D projection on the image, which can be readily obtained by assuming known camera intrinsic matrix. The perceptual feature pooling is enabled once after several convolutions (i.e. a deformation block described in Sec. 3.4) 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 ground truth. In practice, we observe that network trained to directly predict mesh with a large number of vertices is likely to make 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 add local details as the number of vertices increases later. Besides the graph unpooling layer, we use a deep GCN enhanced by shortcut connections [13] as the backbone of our architecture, which enables large receptive fields for global context and more steps of movements.

Representing the shape in 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 favor smooth surface; an edge loss to encourage uniform distribution of mesh vertices for high recall; and a laplacian loss to prevent mesh faces from intersecting each other. All of these losses are essential to generate quality appealing mesh model, and none of them can be trivially defined without the graph representation.

The contributions of this paper are mainly in three aspects. First, we propose a novel end-to-end neural network architecture that generates a 3D mesh model from a single RGB image. Second, we design a projection layer which incorporates perceptual image features into the 3D geometry represented by GCN. Third, our network predict 3D geometry in a coarse to fine fashion, which is more reliable and easy to learn.

2 Related Work

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) [27] for large-scale high-quality reconstruction and simultaneous localization and mapping (SLAM) [4] for navigation. Though they are very successful in these scenarios, they are restricted by 1) the coverage that the multiple views can give and 2) the appearance of the object that wants to reconstruct. The former restriction means MVG cannot reconstruct unseen parts of the object, and thus it usually takes a long time to get enough views for a good reconstruction; the latter restriction means MVG cannot reconstruct non-lambertian (e.g. reflective or transparent) or textureless objects. These restrictions lead to the trend of resorting to learning based approaches.

Learning based approaches usually consider single or few images, as it largely relies on the shape priors that it can learn from data. Early works can be traced back to Hoiem *et al.*[14] and Saxena *et al.*[25]. Most recently, with the success of deep learning architectures and the release of large-scale 3D shape datasets such as ShapeNet [5], learning based approaches have achieved great progress. Huang *et al.*[15] and Su *et al.*[29] retrieve shape components from a large dataset, assemble them and deform the assembled shape to fit the observed image. However, shape retrieval from images itself is an ill-posed problem. To avoid this problem, Kar *et al.*[16] learns a 3D deformable model for each object category and capture the shape variations in different images. However, the reconstruction is limited to the popular categories and its reconstruction result is usually lack of details. Another line of research is to directly learn 3D shapes from single images. Restricted by the prevalent grid-based deep learning architectures, most works [6, 11] outputs 3D voxels, which are usually with low resolutions due to the memory

constraint on a modern GPU. Most recently, Tatarchenko *et al.*[30] have proposed an octree representation, which allows to reconstructing higher resolution outputs with a limited memory budget. However, a 3D voxel is still not a popular shape representation in game and movie industries. To avoid drawbacks of the voxel representation, Fan *et al.*[9] propose to generate point clouds from single images. The point cloud representation has no local connections between points, and thus the point positions have a very large degree of freedom. Consequently, the generated point cloud is usually not close to a surface and cannot be used to recover a 3D mesh directly. Besides these typical 3D representations, there is an interesting work [28] which uses a so-called "geometry image" to represent a 3D shape. Thus, their network is a 2D convolutional neural network which conducts an image to image mapping. Our works are mostly related to the two recent works [17] and [24]. 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.

Our base network is a graph neural network [26]; this architecture has been adopted for shape analysis [31]. In the meanwhile, there are charting-based methods which directly apply convolutions on surface manifolds [2, 22, 23] 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 [3].

3 Method

3.1 Preliminary: Graph-based Convolution

We first provide some background about graph based convolution; more detailed introduction can be found in [3]. 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 learned from the input color image (if they exist). Running convolutions updates the features, which is equivalent as applying a deformation.

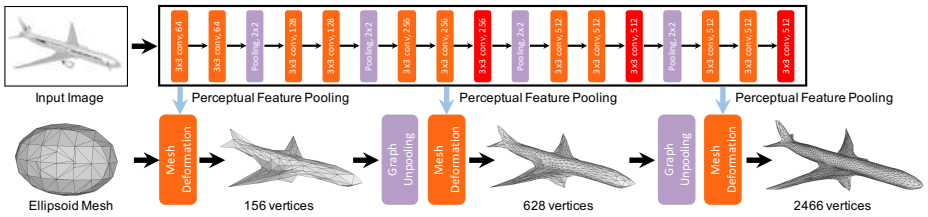

Fig. 2. The cascaded mesh deformation network. Our full model contains three mesh deformation blocks in a row. Each block increases mesh resolution and estimates vertex locations, which are then used to extract perceptual image features from the 2D CNN for the next block.

3.2 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 extract perceptual feature 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 convolution 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 maintain 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.*[9] with three other mesh specific loss – Surface normal loss, Laplacian regularization loss, and Edge length loss. The remaining part of this section describes details of these components.

3.3 Initial ellipsoid

Our model does not require any prior knowledge of the 3D shape, and always deform 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 radius of three axis. The mesh model is generated by implicit surface algorithm in Meshlab [7] 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.

3.4 Mesh deformation block

The architecture of mesh deformation block is shown in Fig. 3 (a). In order to generate 3D mesh model that is consistent with the object shown in the input image, the deformation block need to pool feature ($\mathbf{P}$) from the input image. This is done in conjunction

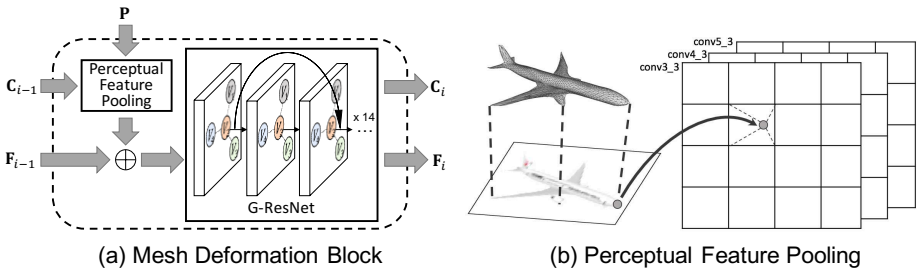

Fig. 3. (a) The vertex locations $\mathbf{C}_i$ are used to extract image features, which are then combined with vertex features $\mathbf{F}_i$ and fed into G-ResNet. $\bigoplus$ means a concatenation of the features. (b) The 3D vertices are projected to the image plane using camera intrinsics, and perceptual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

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 ResNet (G-ResNet). 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.

Perceptual feature pooling layer We use a VGG-16 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 input image plane using camera intrinsics, and then pool the feature from four nearby pixels using bilinear interpolation. In particular, we concatenate feature 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 feature 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 concatenated with the 3-dim feature (coordinate) since there is no learnt shape feature at the beginning.

G-ResNet After obtaining 1408-dim feature 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 efficient exchange of the information between vertices. 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 as the small receptive field issue on 2D CNN.

To solve this issue, we make a very deep network with shortcut connections [13] 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. The serial of G-ResNet block produces a new 128-dim 3D feature. In addition

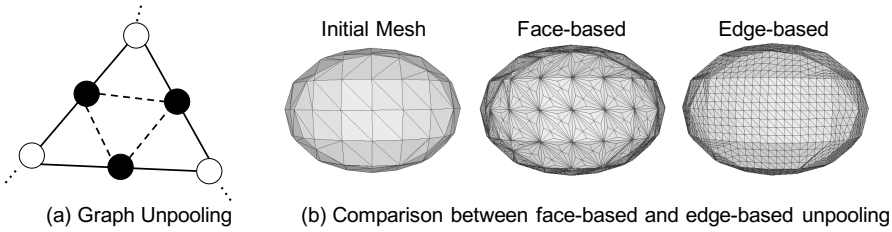

Fig. 4. (a) Black vertices and dashed edges are added in the unpooling layer. (b) The face based unpooling leads to imbalanced vertex degrees, while the edge-based unpooling remains regular.

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.

3.5 Graph unpooling layer

The goal of unpooling layer is to increase the number of vertex in the GCNN. 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 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-point of this edge (Fig. 4 (a)). The 3D feature for 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 vertex 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.

3.6 Losses

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 [9] to constrain the location of mesh vertices, a normal loss to enforce the consistency of surface normal, a laplacian regularization to maintain relative location between neighboring vertices during deformation, and an edge length regularization to prevent outliers. These losses are applied with equal weight on both the intermediate and final mesh.

Unless otherwise stated, 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 pixel of p, till the end of this section.

Chamfer loss The Chamfer distance measures the distance of each point to the other set: $l_c = \sum_p \min_q \|p - q\|_2^2 + \sum_q \min_p \|p - q\|_2^2$. It is reasonably good to regress the vertices close to its correct position, however is not sufficient to produce nice 3D mesh (see the result of Fan et al. [9] in Fig. 1).

Normal loss We further define loss on surface normal to characterize high order properties: $l_n = \sum_p \sum_{q=\arg\min_q(\|p-q\|_2^2)} \|\langle p-k, \mathbf{n}_q\rangle\|_2^2,\ \text{s.t. } k \in \mathcal{N}(p)$ where q is the closest vertex for p that is found when calculating the chamfer loss, k is the neighboring pixel of p, $\langle\cdot,\cdot\rangle$ is the inner product of two vectors, and $\mathbf{n}_q$ is the observed surface normal from ground truth.

Essentially, this loss requires the edge between a vertex with its neighbors to 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 as 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.

Regularization Even with 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 ground truth, and causes flying vertices (Fig. 5).

Laplacian regularization To handle these problem, 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, that 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 $\delta_p = p - \sum_{k\in\mathcal{N}(p)} \frac{1}{\|\mathcal{N}(p)\|} k$, and the laplacian regularization is defined as: $l_{lap} = \sum_p \|\delta'_p - \delta_p\|_2^2$, where δ'_p and δ_p are the laplacian coordinate of a vertex after and before a deformation block.

Edge length regularization. To penalize flying vertices, which ususally cause long edge, we add an edge length regularization loss: $l_{loc} = \sum_p \sum_{k\in\mathcal{N}(p)} \|p-k\|_2^2$.

The overall loss is a weighted sum of all four losses, $l_{all} = l_c + \lambda_1 l_n + \lambda_2 l_{lap} + \lambda_3 l_{loc}$, where $\lambda_1 = 1.6e-4$, $\lambda_2 = 0.3$ and $\lambda_3 = 0.1$ are the hyperparameters which balance the losses and fixed for all the experiments.

4 Experiment

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 analyse 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.

4.1 Experimental setup

Data. We use the dataset provided by Choy et al. [6]. The dataset contains rendering images of 50k models belonging to 13 object categories from ShapeNet [5], 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. For fair comparison, we use the same training/testing split as in Choy et. al. [6].

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 other within certain threshold τ. A F-score [19] as the harmonic mean of precision and recall is then calculated. Following Fan et. al. [9], 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, high-order details, for which a standard evaluation metric is barely missing in literature. Thus, we recommend to pay attention on qualitative results for better understanding of these aspects.

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. [6] (3D-R2N2) producing 3D volume, and Fan et. al. [9] (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 [21] method.

We also compare to Neural 3D Mesh Renderer (N3MR) [17] which is so far the only deep learning based mesh generation model with code public available. For fair comparison, the models are trained with the same data using the same amount of time.

Training and Runtime. Our network receives input images of size 224×224, and 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 epoch 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 to generate a mesh with 2466 vertices.

4.2 Comparison to state of the art

Tab. 1 shows the F-score 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 smaller threshold τ,

Threshold	τ				2τ			
Category	3D-R2N2	PSG	N3MR	Ours	3D-R2N2	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 $\tau = 10^{-4}$. Larger is better. Best results under each threshold are bolded.

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 signal in images and lacks of explicit handling of the 3D mesh.

We also show the CD and EMD for all categories in Tab. 2. Our approach outperforms the other methods in most categories and achieves the best mean score. The major competitor is PSG, which produces a point cloud and has the most freedom; this freedom leads to smaller CD and EMD, however does not necessarily leads to a better mesh model without proper regularization. To demonstrate this, we show the qualitative results to analyze why our approach outperforms the others. Fig. 8 shows the visual results. To compare the quality of mesh model, we convert volumetric and point cloud to mesh using standard approaches [21, 1]. 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. 8. We tried octree based solution [30] to increase the volume resolution, but found it still hard to recover surface level details as much as our model. PSG produces sparse 3D point clouds, and it is non-trivial to recover meshes from them. This is due to the applied Chamfer loss acting like a regression loss which gives too much degree of freedom to the point cloud. N3MR produces very rough shape, which might be sufficient for some rendering tasks, however cannot recover complicated objects such as chairs and tables. In contrast, our model does not suffer from these issues by leveraging a mesh representation, integration of perceptual feature, and carefully defined losses during the training. Our result is not restricted by the resolution due to the limited memory budget and contains both smooth continuous surface and local details.

Category	CD				EMD			
	3D-R2N2	PSG	N3MR	Ours	3D-R2N2	PSG	N3MR	Ours
plane	0.895	**0.430**	0.450	0.477	0.606	**0.396**	7.498	0.579
bench	1.891	0.629	2.268	**0.624**	1.136	1.113	11.766	**0.965**
cabinet	0.735	0.439	2.555	**0.381**	**2.520**	2.986	17.062	2.563
car	0.845	0.333	2.298	**0.268**	1.670	1.747	11.641	**1.297**
chair	1.432	0.645	2.084	**0.610**	1.466	1.946	11.809	**1.399**
monitor	1.707	**0.722**	3.111	0.755	1.667	1.891	14.097	**1.536**
lamp	4.009	**1.193**	3.013	1.295	1.424	1.222	14.741	**1.314**
speaker	1.507	0.756	3.343	**0.739**	**2.732**	3.490	16.720	2.951
firearm	0.993	**0.423**	2.641	0.453	0.688	**0.397**	11.889	0.667
couch	1.135	0.549	3.512	**0.490**	2.114	2.207	14.876	**1.642**
table	1.116	0.517	2.383	**0.498**	1.641	2.121	12.842	**1.480**
cellphone	1.137	0.438	4.366	**0.421**	0.912	1.019	17.649	**0.724**
watercraft	1.215	**0.633**	2.154	0.670	0.935	0.945	11.425	**0.814**
mean	1.445	0.593	2.629	**0.591**	1.501	1.653	13.386	**1.380**

Table 2. CD and EMD on the ShapeNet test set. Smaller is better. Best results under each threshold are bolded.

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 3. Ablation study that evaluates the contribution 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, small is better.

4.3 Ablation Study

Now we conduct controlled experiments to analyse the importance of each component in our model. Tab. 3 reports the performance of each model by removing one component from the full model. Again, we argue that these commonly used evaluation metrics does 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, in fact produces the worst mesh (Fig. 5, the last 2nd column). As such, we use qualitative result Fig. 5 to show the contribution of each component in our system.

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

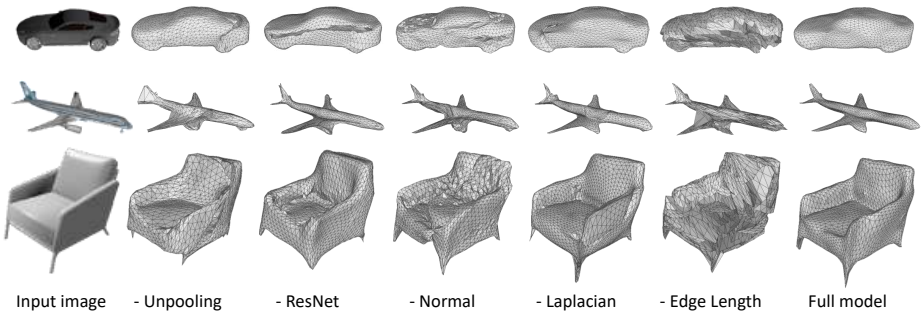

Fig. 5. Qualitative results for ablation study. This figure truly reflects the contribution of each components especially for the regularization ones.

that the deformation makes mistake easier at beginning, which cannot be fixed later on. Consequently, there are some obvious artifacts in some parts of the objects.

G-ResNet We then remove the shortcut connections in G-ResNet, and make it regular GCN. As can be seen from Tab. 3, there is a huge performance gap in all four measurement metrics, which means the failure of optimizing Chamfer distance. The main reason is the degradation problem observed in the very deep 2D convolutional neural network. Such problem leads to a higher training error (and thus higher testing error) when adding more layers to a suitably deep model [13]. Essetially, our network has 42 graph convolutional layers. Thus, this phenomenon has also been observed in our very deep graph neural network experiment.

Loss terms We evaluate the function of each additional terms besides the Chamfer loss. As can be seen in Fig. 5, removing normal loss severely impairs the surface smoothness and local details, e.g. seat back; removing Laplacian term causes intersecting geometry because the local topology changes, e.g. the hand held of the chair; removing 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.

Number of Deformation Blocks We now analyze the effects of the number of blocks. Figure Fig. 6 (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 experiment, we found that 4 blocks results in too many vertices and edges, which slow down our approach dramatically even though it provides better accuracy on evaluation metrics. Therefore, we use 3 blocks in all our experiment for the best balance of performance and efficiency. Fig. 6 (right) shows the output of our model after each deformation block. Notice how mesh is densified with more vertices and new details are added.

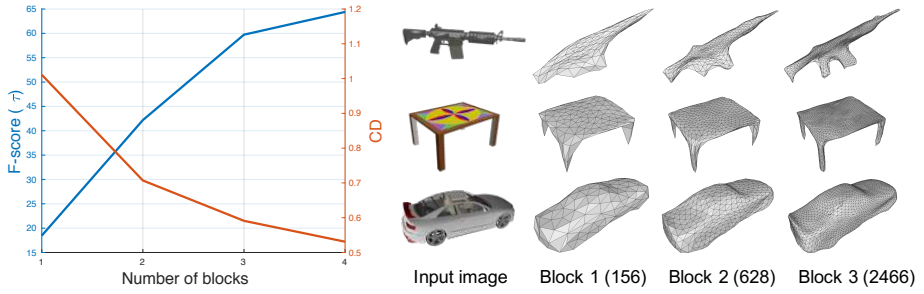

Fig. 6. Left: Effect of number of blocks. Each curve shows the mean F-score (τ) and CD for different number of blocks. Right: Sample examples showing the output after each block.

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

4.4 Reconstructing Real-World images

Following Choy et. al. [6], we test our network on the Online Products dataset and Internet images for qualitative evaluation on real images. We use the model trained from ShapeNet dataset and directly run on real images without finetuning, and show results in Fig. 7. As can be seen, our model trained on synthetic data generalizes well to the real-world images across various categories.

5 Conclusion

We have presented an approach to extract 3D triangular meshes from singe images. We exploit the key advantages the mesh presentation can bring to us, and the key issues required to solve 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 carefully design our network structure and propose 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

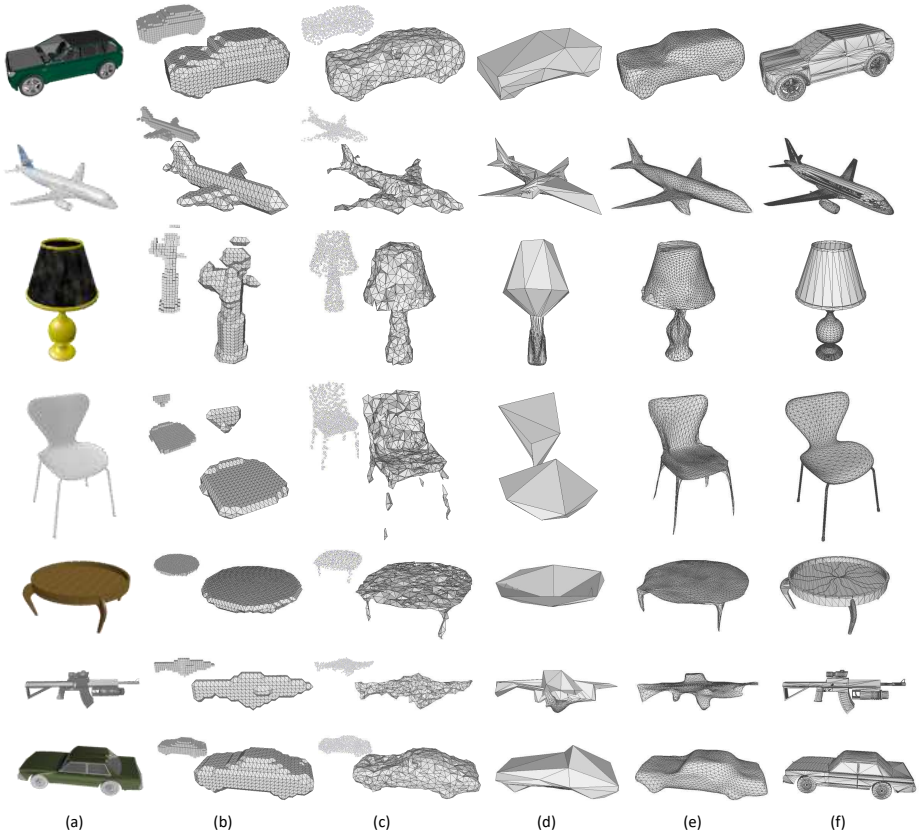

Fig. 8. Qualitative results. (a) Input image; (b) Volume from 3D-R2N2 [6], converted using Marching Cube [21]; (c) Point cloud from PSG [9], converted using ball pivoting [1]; (d) N3MR[17]; (e) Ours; (f) Ground truth.

loss and normal loss. Our results are significantly better than the previous state-of-the-art using other shape representations such as 3D volume or 3D point cloud. Thus, we believe mesh representation is the next big thing in this direction, and we hope that the key components discovered in our work can support follow-up works that will further advance direct 3D mesh reconstruction from single images.

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.

Acknowledgements This work was supported by two projects from NSFC (#61622204 and #61572134), two projects from STCSM (#16JC1420401 and #16QA1400500), Eastern Scholar (TP2017006), and The Thousand Talents Plan of China (for young professionals, D1410009).

References

1. Bernardini, F., Mittleman, J., Rushmeier, H.E., Silva, C.T., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE Trans. Vis. Comput. Graph. **5**(4), 349–359 (1999)
2. Boscaini, D., Masci, J., Rodolà, E., Bronstein, M.M.: Learning shape correspondence with anisotropic convolutional neural networks. In: NIPS (2016)
3. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: Going beyond euclidean data. IEEE Signal Process. Mag. **34**(4), 18–42 (2017)
4. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.: Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. IEEE Transactions on Robotics **32**(6), 1309–1332 (2016)
5. 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.: ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR] (2015)
6. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: ECCV (2016)
7. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: Meshlab: an open-source mesh processing tool. In: Eurographics Italian Chapter Conference (2008)
8. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS. pp. 3837–3845 (2016)
9. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: CVPR (2017)
10. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR (2016)
11. Girdhar, R., Fouhey, D.F., Rodriguez, M., Gupta, A.: Learning a predictable and generative vector representation for objects. In: ECCV (2016)
12. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2004)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
14. Hoiem, D., Efros, A.A., Hebert, M.: Recovering surface layout from an image. International Journal of Computer Vision **75**(1), 151–172 (2007)
15. Huang, Q., Wang, H., Koltun, V.: Single-view reconstruction via joint analysis of image and shape collections. ACM Trans. Graph. **34**(4), 87:1–87:10 (2015)
16. Kar, A., Tulsiani, S., Carreira, J., Malik, J.: Category-specific object reconstruction from a single image. In: CVPR (2015)
17. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: CVPR (2018)
18. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2016)
19. Knapitsch, A., Park, J., Zhou, Q., Koltun, V.: Tanks and temples: benchmarking large-scale scene reconstruction. ACM Trans. Graph. **36**(4), 78:1–78:13 (2017)
20. Li, Z., Chen, Q., Koltun, V.: Interactive image segmentation with latent diversity. In: CVPR (2018)
21. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: SIGGRAPH (1987)
22. Masci, J., Boscaini, D., Bronstein, M.M., Vandergheynst, P.: Geodesic convolutional neural networks on riemannian manifolds. In: ICCV Workshop (2015)
23. Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: CVPR (2017)

24. Pontes, J.K., Kong, C., Sridharan, S., Lucey, S., Eriksson, A., Fookes, C.: Image2mesh: A learning framework for single image 3d reconstruction. Tech. Rep. arXiv:1711.10669 [cs.CV] (2017)
25. Saxena, A., Sun, M., Ng, A.Y.: Make3d: Learning 3d scene structure from a single still image. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 824–840 (2009)
26. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Networks **20**(1), 61–80 (2009)
27. Schönberger, J.L., Frahm, J.: Structure-from-motion revisited. In: CVPR (2016)
28. Sinha, A., Unmesh, A., Huang, Q., Ramani, K.: Surfnet: Generating 3d shape surfaces using deep residual networks. In: CVPR (2017)
29. Su, H., Huang, Q., Mitra, N.J., Li, Y., Guibas, L.J.: Estimating image depth using shape collections. ACM Trans. Graph. **33**(4), 37:1–37:11 (2014)
30. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In: ICCV (2017)
31. Yi, L., Su, H., Guo, X., Guibas, L.J.: Syncspeccnn: Synchronized spectral CNN for 3d shape segmentation. In: CVPR (2017)