Pixel2Mesh++: 3D Mesh Generation and Refinement From Multi-View Images

Chao Wen, Yinda Zhang, Chenjie Cao, Zhuwen Li, Xiangyang Xue, Member, IEEE, and Yanwei Fu

Abstract—We study the problem of shape generation in 3D mesh representation from a small number of color images with or without camera poses. While many previous works learn to hallucinate the shape directly from priors, we adopt to further improve the shape quality by leveraging cross-view information with a graph convolution network. Instead of building a direct mapping function from images to 3D shape, our model learns to predict series of deformations to improve a coarse shape iteratively. Inspired by traditional multiple view geometry methods, our network samples nearby area around the initial mesh’s vertex locations and reasons an optimal deformation using perceptual feature statistics built from multiple input images. Extensive experiments show that our model produces accurate 3D shapes that are not only visually plausible from the input perspectives, but also well aligned to arbitrary viewpoints. With the help of physically driven architecture, our model also exhibits generalization capability across different semantic categories, and the number of input images. Model analysis experiments show that our model is robust to the quality of the initial mesh and the error of camera pose, and can be combined with a differentiable renderer for test-time optimization.

Index Terms—3D shape generation, multi-view, graph convolutional neural network, mesh reconstruction

1 INTRODUCTION

The topic of shape generation has become increasingly popular over the past decade. With the astonishing capability of deep learning, lots of works have been demonstrated to successfully generate the 3D shape from merely a single color image. However, due to limited visual cues from only one viewpoint, single image based approaches usually produce rough geometry in the occluded area and do not perform well when generalized to domains other than training, e.g., cross semantic categories.

Adding a few more multi-view images (e.g., 3-5) of the object is an effective way to provide the shape generation system with more information about the 3D shape. On one hand, multi-view images provide more visual appearance information, and thus grant the system with more chance to build the connection between 3D shape and image priors. On the other hand, it is well-known that traditional multi-view geometry methods [2] accurately infer 3D shape from correspondences across views, which is analytically well defined and less vulnerable to the generalization problem. Even though typical multiview methods suffer from many problems, such as large baselines and textureless regions, and are likely to degenerate with very limited input images (e.g., less than 5), the way they leverage cross-view information might be implicitly encoded and learned by a deep model. While well-motivated, there are very few works in the literature exploiting in this direction. And a naïve multi-view extension that fuses output 3D shapes or intermediate image features of single image based model does not work well as shown in Fig. 1.

In this work, we propose a deep learning model to generate the object shape from multiple color images with or without known camera poses. Especially, we focus on endowing the deep model with the capacity of improving shapes using cross-view information. The key component in our method is a new network architecture, named Multi-View Deformation Network (MDN), which works in conjunction with the Graph Convolutional Network (GCN) architecture proposed in Pixel2Mesh [1] to generate accurate 3D geometry shape in the desirable mesh representation. In Pixel2Mesh, a GCN is trained to deform an initial shape to the target using features from a single image, which often produces plausible shapes but lack of accuracy (Fig. 1 P2M). We inherit this characteristic of “generation via deformation” and further deform the mesh in MDN using features carefully pooled from multiple images. Instead of learning to hallucinate via shape priors like in Pixel2Mesh, MDN reasons shapes according to correlations across different views through a physically driven architecture inspired by classic multi-view geometry methods. In particular, MDN proposes hypothesis deformations for each vertex and move it to the optimal location that best explains features pooled from multiple views. By imitating correspondences search rather than learning priors, MDN generalizes well in various aspects, such as cross semantic category, number of input views, and the mesh initialization.
made MDN more convenient to use in real scenario where camera pose may not be immediately accessible. Last but not least, the nature of “generation via deformation” allows an iterative refinement. In particular, the model output can be taken as the input, and the quality of the 3D shape is gradually improved throughout iterations. With these desiring features, our model achieves the state-of-the-art performance on ShapeNet for shape generation from multiple images under standard evaluation metrics.

To summarize, we propose a GCN framework that produces 3D shapes in mesh representation from a small number of observations of the object in different viewpoints. The core component is a physically driven architecture that searches optimal deformation to improve a coarse mesh using perceptual feature statistics built from multiple images, which produces accurate 3D shape and generalizes well across different semantic categories, numbers of input images, and the quality of coarse meshes. Our method can produce 3D shapes even when the initial topology is varied and with or without knowing the ground truth camera position.

An early version of this work has been published in [5]. Compared with [5], we extend our method to work without known camera pose, with various initialization methods, more experiments and model analysis, as well as other technical improvements such as differentiable renderer. We show that our method can further deform the initial coarse meshes from those objects of arbitrary topology.

2 RELATED WORK

2.1 3D Shape Representations

Since 3D CNN is readily applicable to 3D volumes, the volume representation has been well-exploited for 3D shape analysis and generation [6], [7]. With the debut of PointNet [8], the point cloud representation has been adopted in many works [9], [10]. Most recently, the mesh representation [1], [11] has become competitive due to its compactness and nice surface properties. Some other representations have been proposed, such as geometry images [12], depth images [13], [14], classification boundaries [15], [16], [17], [18], [19], signed distance function [20], etc., and most of them require post-processing to get the final 3D shape. Consequently, the shape accuracy may vary and the inference take extra time.

2.2 Single View Shape Generation

Classic single view shape reasoning can be traced back to shape from shading [21], [22], texture [23], and de-focus [24], which only reason the visible parts of objects. With deep learning, many works leverage the data prior to hallucinate the invisible parts, and directly produce shape in 3D volume [6], [25], [26], [27], [28], [29], [30], point cloud [9], mesh models [11], [31], [32], [33], [34], or as an assembling of shape primitive [35], [36], [37], [38], [39], [40]. Alternatively, 3D shape can be also generated by deforming an initialization, which is more related to our work. Tulsiani et al. [41] and Kanazawa et al. [42] learn a category-specific 3D deformable model and reason the shape deformations in different images. Wang et al. [1] learn to deform an initial ellipsoid to the desired shape in a coarse to fine fashion. Combining deformation and assembly, Huang et al. [43] and Su et al.
[44] retrieve shape components from a large dataset and deform the assembled shape to fit the observed image. Kurynkov et al. [45] learns free-form deformations to refine shape. Even though with impressive success, most single view deep models adopt an encoder-decoder framework, and it is arguable if they perform shape generation or shape retrieval [46]. Groueix et al. [31] also learn to generate 3D shapes by deforming explicit surfaces. Their method dubbed Atlassnet, uses the global latent vector and MLP parameters to represent the patches deformation. In contrast, our work models the deformation by passing message in GCN and gathers local perceptual feature for each vertex of GCN. It is feasible for Atlassnet to model the shape with holes by using multiple patches, while it is non-trivial to handle patch overlap and get the continuous surface meshes. Our architecture, however, allows our network to generate and refine plausible smooth shapes when proper coarse shape initialization is used.

2.3 Multi-View Shape Generation

Recovering 3D geometry from multiple views has been well studied. Traditional multi-view stereo (MVS) [2] relies on correspondences built via photo-consistency and thus it is vulnerable to large baselines, occlusions, and texture-less regions. Most recently, deep learning based MVS models have drawn attention, and most of these approaches [47], [48], [49], [50] rely on a cost volume built from depth hypotheses or plane sweeps. Donne et al. [51] refine a set of input depth maps in the image domain. However, these approaches usually generate depth maps, and it is non-trivial to fuse a full 3D shape from them. On the other hand, direct multi-view shape generation uses fewer input views with large baselines, which is more challenging and has been less addressed. Choy et al. [6] propose a unified framework for single and multi-view object generation reading images sequentially. Kar et al. [52] learn a multi-view stereo machine via recurrent feature fusion. Gwak et al. [53] learns shapes from multi-view silhouettes by ray-tracing pooling and further constrains the ill-posed problem using GAN. Our approach belongs to this category but is fundamentally different from the existing methods. Rather than sequentially feeding in images, our method learns a GCN to deform the mesh using features pooled from all input images at once.

2.4 Differentiable Rendering

Differentiable rendering retains derivatives during the forward process of image synthesis [54]. Therefore, many differentiable rendering enhanced neural networks can directly optimize their neural fields [55] through image pixel colors, i.e., neural rendering. According to the representation, this type of work can be categorized into neural radiance fields [56], implicit differentiable renderer [57], [58] and differentiable rasterizer [59]. Neural radiance fields (NeRF) and its successors [56], [60], [61], [62], [63], [64], [65] focus on representing the scene as 5D neural radiance fields. The success of these approaches mostly requires per-scene optimization, which is more suitable for novel-view synthesis tasks than 3D shape generation. Among these, Yu et al. [62] propose pixelNeRF that shows the potential of cross-category and sparse input view rendering. However it is difficult to extract a high-quality surface. Implicit differentiable renderer approaches [57], [58] can generate 3D shapes via 2D supervision and volumetric rendering. Liu et al. [59] propose a soft version rasterizer and learn to deform template mesh using multiple silhouette and color images. Despite that these approaches could produce compelling results having dense input views, they still have performance degradation when very few views are available. Our method can not only handle this situation, but also can absorb differentiable renderer as a tool to optimize 3D shapes during inference.

2.5 Camera Pose Estimation

As a sub-problem in 3D shape recovery, many works are trying to tackle the problem of camera pose estimation. Kendall et al. [66] proposed PoseNet, which represents the pose as the camera position and camera rotation, and obtains the value of the pose by normalizing the output quaternion. Since two quaternions with opposite signs represent the same rotation under boundary condition, Wu et al. [67] use Euler angles to represent rotation. Peng et al. [68] regard camera pose estimation as a key point regression problem, and use voting strategy to densely estimate the key point offset. Recently, Zhou et al. [69] show that the discontinuity of the rotation representation is the main reason for the difficulty in pose estimation, and proposed a continuous 6D rotation representation. Xu et al. [4] continue to use the 6D rotation representation, and convert the pose numerical regression to the matching problem of the point cloud, which enhances the generalization performance. We followed the 6D rotation representation, and further represented the translation as the scalar offset of the object along the optical axis in the camera coordinate.

3 METHOD

3.1 System Overview

Our model receives multiple color images of an object captured from different viewpoints (with or without known camera poses) and produces a 3D mesh model in the world coordinate. This procedure is conceptually visualized in Fig. 2. Our approach consists of two parts: 3D shape prediction and camera pose estimation. The 3D shape prediction framework adopts the strategy of coarse-to-fine (Fig. 3), in which a plausible but rough shape is generated first, and details are added later. Realizing that existing 3D shape generators usually produce reasonable shape even from a single image, we simply use Pixel2Mesh [1] or DISN [4] trained from multiple views to produce the coarse shape, which is taken as input to our Multi-View Deformation framework.
Network (MDN) for further improvement. In MDN, each vertex first samples a set of deformation hypotheses from its surrounding area (Fig. 4a). Each hypothesis then pools cross-view perceptual feature from early layers of a perceptual network, where the feature resolution is high and contains more low-level geometry information (Fig. 4b). These features are gathered by projection with the help of the ground truth or predicted camera parameters. The cross-view perceptual features are further leveraged by the network to reason the best deformation to move the mesh vertex. It is worth noting that our MDN can be applied iteratively for multiple times to gradually improve shapes. In cases where we cannot obtain the camera parameters in advance, we estimate them through a neural network.

The multi-view deformation network, which is the core of our system to enable the network exploiting cross-view information for shape generation. We assume that we know the extrinsics of the camera. (We will come back to address this issue in the next section.) It first generates deformation hypotheses for each vertex and learns to reason an optimum using feature pooled from inputs. Our model is essentially a GCN, and can be jointly trained with other GCN based models like Pixel2Mesh. We refer reader to [70], [71] for details about GCN, and Pixel2Mesh [1] for graph residual block which will be used in our model.

### 3.2 Multi-View Deformation Network

In this section, we introduce Multi-View Deformation Network, which is the core of our system to enable the network exploiting cross-view information for shape generation. We assume that we know the extrinsics of the camera. (We will come back to address this issue in the next section.) It first generates deformation hypotheses for each vertex and learns to reason an optimum using feature pooled from inputs. Our model is essentially a GCN, and can be jointly trained with other GCN based models like Pixel2Mesh. We refer reader to [70], [71] for details about GCN, and Pixel2Mesh [1] for graph residual block which will be used in our model.

#### 3.2.1 Deformation Hypothesis Sampling

The first step is to propose deformation hypotheses for each vertex. This is equivalent as sampling a set of target locations in 3D space where the vertex can be possibly moved to. To uniformly explore the nearby area, we sample from a level-1 icosahedron centered on the vertex with a scale of 0.02, which results in 42 hypothesis positions (Fig. 4a, left). We then build a local graph with edges on the icosahedron
surface and additional edges between the hypotheses to the vertex in the center, which forms a graph with 43 nodes and 120 + 42 = 162 edges. Such the local graph is built for all mesh vertices, and then fed into a GCN to predict vertex movements (Fig. 4a, right).

### 3.2.2 Cross-View Perceptual Feature Pooling

The second step is to assign each node (in the local GCN) features from the multiple input color images. Inspired by Pixel2Mesh, we use the prevalent VGG-16 architecture to extract perceptual features. Since the camera poses are either as known inputs or can be predicted by our camera pose estimation network, each vertex and hypothesis can find their projections in all input color image planes using camera intrinsics and extrinsics and pool features from four neighboring feature grids using bilinear interpolation (Fig. 4b). Different from Pixel2Mesh where high level features from later layers of the VGG (i.e., ‘conv3_3,’ ‘conv4_3,’ and ‘conv5_3’) are pooled to better learn shape priors, MDN pools features from early layers (i.e., ‘conv1_2,’ ‘conv2_2,’ and ‘conv3_3’), which are in high spatial resolution and considered maintaining more detailed information.

To combine multiple features, concatenation has been widely used as a loss-less way; however, it ends up with total dimension changing with respect to the number of input images. Statistics feature has been proposed for multi-view shape recognition [72] to tackle this problem. Inspired by this work, we concatenate some statistics (mean, max, and std) of the features pooled from all views for each vertex, which makes our network naturally adaptive to variable input views and behave invariant to different input orders. This also encourages the network to learn from cross-view feature correlations rather than each individual feature vector. In addition to image features, we also concatenate the 3-dimensional vertex coordinate into the feature vector. In total, we compute for each vertex and hypothesis a 339 dimension feature vector.

### 3.2.3 Deformation Reasoning

The next step is to reason an optimal deformation for each vertex from the hypotheses using pooled cross-view perceptual features. Note that picking the best hypothesis of all needs an argmax operation, which requires stochastic optimization and usually is not optimal. Instead, we design a differentiable network component to produce desirable deformation through soft-argmax of the 3D deformation hypotheses, which is illustrated in Fig. 5. Specifically, we first feed the cross-view perceptual feature $P$ into a scoring network, consisting of 6 graph residual convolution layers [1] activated by ReLU, to predict a scalar weight $c_i$ for each hypothesis. All the weights are then fed into a softmax layer and normalized to scores $s_i$, with $\sum_{i=1}^{43} s_i = 1$. The vertex location is then updated as the weighted sum of all the hypotheses, i.e., $v = \sum_{i=1}^{43} s_i h_i$, where $h_i$ is the location of each deformation hypothesis including the vertex itself. This deformation reasoning unit runs on all local GCN built upon every vertex with shared weights, as we expect all the vertices leveraging multi-view feature in a similar fashion.

### 3.2.4 Differentiable Renderer

Lastly, as an option, our method can be combined with differentiable renderer to achieve instance optimization during inference. We use common mesh rasterizer renderer pipeline [59], [73] as an optional component. When the silhouettes of images can be obtained from the input object, the 3D shape of the object can be further optimized by matching the silhouette of the predicted shapes with the multi-view input silhouette images through differentiable renderer.

### 3.3 Camera Pose Estimation Network

In this section, we introduce camera pose estimation network. In case the poses of the camera cannot be obtained, we use another convolution network to estimate them.

Given an input image without camera pose, our goal is to estimate the corresponding camera extrinsics. The models in the ShapeNet Core dataset [74] are aligned for each category. Therefore, the world space is aligned with respect to this canonical coordinate system. As discussed in [75], regressing camera pose directly often fails, Zhou et al. [69] proposed a continuous rotation representation. We inherit this representation and regard the camera translation as the offset of the object along the direction of the camera optical axis, so that the camera extrinsics can be represented as 6D rotation and 1D translation. As illustrated in Fig. 6, we use a simple convolution network to regress the rotation and
We train our model fully supervised since using a template with a fixed topology limits the expression ability as initialization of the coarse shape. We combine the implicit methods MVDISN with stronger representation for multi-view shape generation from multi-view images, as compared in Table 1. Essentially, these methods produce the coarse meshes, which can be utilized as the initial coarse mesh module in our Pixel2mesh++. Specifically,

1. P2M-M: we directly run single-view Pixel2Mesh on each of the input image and fuse multiple results [78], [79].

2. MVP2M: We extend Pixel2Mesh to access multiple images in a single network forward pass by having it pools multi-view features from all the inputs. This can be achieved by replacing the perceptual feature pooling layers with our multi-view version as introduced in Section 3.2.2. In particular, perceptual features are pooled from layer ‘conv3_3,’ ‘conv4_3,’ and ‘conv5_3’ from the VGG-16 network, and feature statistics (Section 4.5.1) are calculated and concatenated, which ends up with a 1280 dimension feature vector. In practice, we also tried to pool geometry related features from early convolution layers (i.e., ‘conv1_2,’ ‘conv2_2,’ and ‘conv3_3’), but found it doesn’t work as well as the case with semantic feature pooled from later layers. Using a template with a fixed topology will limit the expression ability of the shape.

3. MVDISN: in order to show the generalization of our method in arbitrary topology, we build another implicit representation baseline based on DSN. We equip DSN with our perceptual feature pooling layers and extend it to a multi-view version (Table 1, MVDISN). The output signed distance function of MVDISN is converted to mesh using marching cubes algorithm.

Our Variants. We thus define two variants of our Pixel2mesh++ via using different initial coarse meshes, which are produced by MVP2M and MVDISN. Particularly, we have the variants as follows,

1. Ours-P: As in Fig. 2, we take MVP2M as the initial coarse shape, and combine with our MDN. This variant needs ellipsoid mesh as input for coarse shape generation. We train the model in an end-to-end manner.

2. Ours-D: Since using a template with a fixed topology will limit the expression ability of the shape. We combine the implicit methods MVDISN with stronger representation ability as initialization of the coarse shape. The variant of Ours-D is also learned end-to-end in our experiments. Note
that MVDISN can also produce the coarse meshes of objects of arbitrary topology, and our method can still efficiently improve the quality of object meshes.

4 EXPERIMENTS

In this section, we perform extensive evaluation of our model for multi-view shape generation. We compare to state-of-the-art methods, as well as conduct controlled experiments with respect to (w.r.t.) various aspects, e.g., cross category generalization, quantity of inputs, etc. All the experiments assume camera poses are known without specification.

4.1 Experimental Setup

4.1.1 Dataset

We adopt the dataset provided by Choy et al. [6] as it is widely used by many existing 3D shape generation works. The dataset is created using a subset of ShapeNet [74] containing 50 k 3D CAD models from 13 categories. Each model is rendered from 24 randomly chosen camera viewpoints, and the camera intrinsic and extrinsic parameters are given. For fair comparison, we use the same training/testing split as in Choy et al. [6] for all our experiments. The ground truth models are transformed to camera coordinate based on the camera parameters from the Choy et al. [6], and scaled by a factor of 0.57 to aligned with rendering images following Wang et al. [1].

4.1.2 Evaluation Metric

3D Shape Generation. As for 3D shape, we use standard evaluation metrics for 3D shape generation. Following Fan et al. [9], we calculate Chamfer Distance (CD) between 2048 points uniformly sampled from the ground truth and our prediction to measure the surface accuracy. We also use F-score following Wang et al. [1] to measure the completeness and precision of generated shapes, where τ=1.0×10⁻⁴. For CD, the smaller is better. For F-score, the larger is better. We also evaluate volumetric IoU. We transform the ground truth and predictions back to canonical view to calculate IoU. We obtain unbiased estimates of the volumetric IoU by randomly sampling 100 k points following Mescheder et al. [15].

Camera Pose Estimation. We use four types of standard metrics to evaluate our camera pose estimation method: 2D reprojection error $d_{2D}$, mean distance $d_{SD}$ [4], 2D reprojection accuracy $Acc_{2D}$, and average 3D distance of model points (ADD) accuracy metric $ADD_{3D}$ [68]. The reprojection error $d_{2D}$ is compared with the image size 137×137 pixels. Different from the threshold selected for general 6 DOF estimation tasks [68], we use more strict criteria. Specifically, we set the threshold to 2 for the $Acc_{2D}$ metric and the threshold to 5% for the $ADD_{3D}$ metric.

4.1.3 Implementation Details

3D Shape Generation. For initialization, we use Pixel2Mesh [1] to generate a coarse shape with 2,466 vertices, or use DISN [4] to generate a coarse shape with varying topology. To improve the quality of initial mesh, we equip these methods with our cross-view perceptual feature pooling layer, which allows it to extract features from multiple views. We denote these multi-view version shape generation methods as MVP2M and MVDISN. The network is implemented in Pytorch and optimized using Adam with weight decay as $5×10⁻⁵$ and mini-batch size as 1. The model use MVP2M as coarse shape generation method is trained for 40 epochs with learning rate $1×10⁻⁵$ over 72 hours, and the model use MVDISN is trained for 100 epochs with learning rate $3×10⁻⁴$ over 96 hours. The MDN network is trained for another 20 epoch over 24 hours with the learning rate as $1×10⁻⁶$. The overall network needs to learn the statistical information between the multi-view features, and the MDN module needs plausible initialization from the coarse shape generation stage to effectively refine the 3D shape. Therefore, the convergence speed of our method is slower than single view deep models and requires longer training time. During training, we randomly pick three images for a mesh as input. All models are trained on a NVIDIA Titan Xp GPU. During test, it takes 0.32 s to produce a mesh.

Camera Pose Estimation. The camera parameter estimation network is trained for 70 epochs, over 48 hours. The learning
rate is set to $1 \times 10^{-4}$. All 24 views of a mesh and the ground truth mesh points near surface are used as training data. During inference, we fed a single image to the network to obtain the corresponding camera pose.

4.2 Comparison to Multi-View Shape Generation

Table 1 shows quantitative comparison in F-score. As can be seen, our baselines already outperform other methods, which show the advantage of mesh representation in finding surface details. Moreover, directly equipping Pixel2Mesh with multi-view features does not improve too much (even slightly worse than the average of multiple runs of single-view Pixel2Mesh), which shows dedicate architecture is required to efficiently learn from multi-view features. In contrast, our Multi-View Deformation Network significantly further improves the results from the MVP2M baseline (i.e., our coarse shape initialization). It is worth noting that the MVDISN method has more accurate initial shapes and is a stronger baseline. Therefore, using this baseline directly is better than using MVP2M. However, by combining it with our MDN method, the performance will also be significantly improved.

Table 2 shows quantitative comparison in Chamfer Distance. Our method achieves the best performance overall. The notation $\dagger$ indicates the methods which do not require camera extrinsics.

We show the Chamfer Distance on each semantic category. Our method achieves the best performance overall. The notation $\dagger$ indicates the methods which do not require camera extrinsics.

4.3 Generalization Capability

Our MDN is inspired by multi-view geometry methods, where 3D location is reasoned via cross-view information. In this section, we investigate the generalization capability of MDN in different aspects to improve the initialization mesh. For all the experiments in this section, we fix the coarse stage and train/test MDN under different settings.

4.3.1 Semantic Category

We first verify how our network generalizes across semantic categories. We fix the initial MVP2M and train MDN with 12 out of 13 categories and test on the one left out. The improved and the absolute F-scores($t$) upon the initialization from

WEN ET AL.: PIXEL2MESH++: 3D MESH GENERATION AND REFINEMENT FROM MULTI-VIEW IMAGES

2173

TABLE 2
Comparison to Multi-View Shape Generation Methods

| Category | 3DR2N2$\dagger$ | LSM | P2M-M | MVP2M | Ours-P | MVDISN | Ours-D |
|----------|-----------------|-----|-------|-------|-------|-------|-------|
| Couch    | 0.806           | 0.730 | 0.496 | 0.440 | 0.383 | 0.396 | 0.362 |
| Cabinet  | 0.613           | 0.634 | 0.359 | 0.381 | 0.354 | 0.765 | 0.737 |
| Bench    | 1.362           | 0.572 | 0.594 | 0.462 | 0.368 | 0.343 | 0.285 |
| Chair    | 1.534           | 0.495 | 0.561 | 0.494 | 0.402 | 0.409 | 0.346 |
| Monitor  | 1.465           | 0.592 | 0.654 | 0.585 | 0.496 | 0.332 | 0.285 |
| Firearm  | 0.432           | 0.385 | 0.428 | 0.250 | 0.184 | 0.165 | 0.137 |
| Speaker  | 1.443           | 0.767 | 0.697 | 0.666 | 0.604 | 0.746 | 0.663 |
| Lamp     | 6.780           | 1.768 | 1.184 | 0.891 | 0.681 | 0.820 | 0.722 |
| Cellphone| 1.161           | 0.362 | 0.360 | 0.373 | 0.318 | 0.179 | 0.152 |
| Plane    | 0.854           | 0.496 | 0.457 | 0.251 | 0.183 | 0.271 | 0.226 |
| Table    | 1.243           | 0.994 | 0.441 | 0.463 | 0.406 | 0.587 | 0.532 |
| Car      | 0.358           | 0.326 | 0.264 | 0.260 | 0.243 | 0.362 | 0.344 |
| Watercraft| 0.869          | 0.509 | 0.627 | 0.413 | 0.337 | 0.332 | 0.279 |
| Mean     | 1.455           | 0.664 | 0.548 | 0.456 | 0.381 | 0.439 | 0.390 |

TABLE 3
Comparison to Baselines and Our Variants

| Category | MVP2M | Ours-P | MVDISN | Ours-D |
|----------|-------|--------|--------|--------|
| Couch    | 0.546 | 0.569  | 0.671  | 0.688  |
| Cabinet  | 0.536 | 0.550  | 0.398  | 0.397  |
| Bench    | 0.211 | 0.224  | 0.418  | 0.399  |
| Chair    | 0.343 | 0.362  | 0.315  | 0.509  |
| Monitor  | 0.383 | 0.408  | 0.519  | 0.502  |
| Firearm  | 0.371 | 0.426  | 0.621  | 0.539  |
| Speaker  | 0.480 | 0.496  | 0.525  | 0.528  |
| Lamp     | 0.218 | 0.240  | 0.321  | 0.310  |
| Cellphone| 0.613 | 0.644  | 0.679  | 0.628  |
| Plane    | 0.424 | 0.484  | 0.556  | 0.533  |
| Table    | 0.306 | 0.314  | 0.453  | 0.442  |
| Car      | 0.513 | 0.523  | 0.602  | 0.596  |
| Watercraft| 0.391 | 0.429  | 0.565  | 0.556  |
| Mean     | 0.411 | 0.436  | 0.527  | 0.508  |

We show the Chamfer Distance on each semantic category. We show the volumetric IoU on each semantic category.
Table 4

| View Numbers | F-score(r) | F-score(2r) | CD |
|--------------|-----------|------------|----|
| #train       |           |            |    |
| 2            | 64.48     | 78.74      | 0.515 |
| 3            | 66.44     | 80.33      | 0.484 |
| 4            | 67.66     | 81.36      | 0.468 |
| 5            | 68.29     | 81.97      | 0.459 |

‘Resp.’ means that MDN is trained with the same views as the testing. MDN performs consistently better when more view is given, even trained using only 3 views.

4.4 Model Robustness Analysis

In this section, we analyze the robustness of the model in many aspects. We analyze the model’s ability to deal with initialization noise, cameras position error, and instance level optimization during test with differentiable renderer when silhouette mask are available. Through these analyses, we explore why multiple views help for 3D reconstruction.

4.4.1 Initialization

First, we test if the model overfits to the input initialization, i.e., the MVP2M. To this end, we add translation and random noise to the rough shape from MVP2M. We also take as input the mesh converted from 3DR2N2 using marching cubes algorithm [79]. As shown in Fig. 9, MDN successfully removes the noise, aligns the input with ground truth, and adds significant geometry details. This experiments show that MDN is tolerant to input variance.

4.4.2 Camera Pose

We compare the performance of our method on camera pose estimation with Insafutdinov et al. [75] and Xu et al. [4] in Table 5. Given a point cloud in world space, we transform this point cloud using ground truth camera parameters and predicted parameters to camera view respectively. We first calculate the the mean distance in 3D space which is called $d_{3D}$. We then project the point cloud onto the input image plane using known intrinsics to compute the 2D reprojection error $d_{2D}$. We also evaluate $Acc_{2D}$ and $ADD_{2D}$ metrics. With the help of continuous rotation and compact 1D translation representation, our method outperforms Insafutdinov et al. [75] and Xu et al. [4] in terms of $d_{3D}$, $Acc_{2D}$ and $ADD_{2D}$. And our method achieves comparable results on $d_{3D}$.
Moreover, we test whether MDN is able to produce plausible 3D shape when camera pose has error. Since gathering image features require reasonable camera pose, using random camera pose is not feasible. Instead, we use camera pose estimation network to obtain reasonable camera extrinsics. As shown in Table 6 (w/ pred cam), even using predicted camera parameters, our MDN can also refine the initial mesh shape and improve performance compared to the baseline algorithms. Benefiting from more visual cues provided by multi-view images, the above experimental results show that MDN has a certain degree of robustness to camera pose errors.

4.4.3 Differentiable Renderer

We test the effect of our method combined with the differentiable renderer. The differentiable renderer can perform test-time optimization for 3D shape during inference.

In this section, we use the differentiable renderer during inference. The 3D shapes can be further optimized by matching the silhouette of the predicted shapes with the multi-view input images. We use 2D silhouette loss for the optimization, so that the generated 3D model has a better alignment at the views corresponding to the input images. Optimization at the image level helps to restore the unique geometric details of the 3D shape. Moreover, the optimization process can make the shape and the input image align better. Since the previous method based on the differentiable renderer either requires a large number of multi-view silhouettes as supervision [59], or processes a single color image [80], it is not suitable for our task. On the basis of Liu et al. [59], we only use the part of its differentiable renderer.

We use three input images with corresponding silhouettes and network output 3D mesh models for optimization. As shown in Table 6 (w/ silhouette), through the optimization leveraging the differentiable renderer, the performance of our method can be further improved. The quantitative results are shown in the Fig. 10. It can be seen that using the differentiable renderer can further optimize the predicted 3D shape, especially geometric details such as larger corners.

4.4.4 Comparison of Neural Rendering Approaches

We also compare our method with representative neural rendering methods, i.e., Differentiable Volumetric Rendering (DVR) [57] and Implicit Differentiable Rendering (IDR) [58] in Fig. 11 qualitatively. Quantitative results of them are also shown in Fig. 11. Both DVR and IDR are trained with the instance-level, i.e., a separate model needs to be trained for

| Methods | w/pc | w/gtc | w/s | F-score | CD$_t$ |
|---------|------|-------|-----|---------|------|
| Zhou et al. [69] | √ | √ | | 70.07 | 81.79 | 0.518 |
| DISN [4] | √ | | | 75.24 | 85.04 | 0.390 |
| Ours-D | √ | | | 70.26 | 81.95 | 0.511 |
| Ours-P | √ | | | 75.74 | 85.33 | 0.383 |

where pc, gtc, and s are short for predicted camera pose, ground-truth camera pose, and silhouette. CD: Chamfer Distance.
each individual instance. Such online optimization is powerful in reconstruction, but suffers from poor efficiency to test a new coming instance. Because the computation with expensive inference time optimization is required. We train these methods according to the official multi-view reconstruction setting. The initial shape of our method is MVDISN. Since our method only takes 3 images for each inference, we also compare the results of DVR and IDR with 3-view input images. Although rendering-based methods can achieve good results in Figs. 11b and 11c with 24-view inputs, it costs several hours to train an instance-level neural rendering model for each sample, which is inefficient. Besides, if we reduce the input views to three, the performances of DVR and IDR are degraded obviously as shown in Figs. 11d and 11e. And our method can get comparable results with 3-view input images more efficiently.

4.4.5 Cross-Dataset Generalization

We evaluated the cross-data generalization ability of our method. We train models on ShapeNet and test them on the ABC dataset [81] directly without any finetuning for the generalization analysis. Qualitative and quantitative results are shown in Fig. 12. Thanks to feature learning from multiple views, our coarse shape generation from MVDISN has certain cross-data generalization capabilities. And our MDN can further refine the 3D shape in both qualitative and quantitative results, especially when the coarse shape has geometry noise. Besides, the single-view based OccNet [15] may fail to achieve reliable results in ABC dataset. Nevertheless, generalization across datasets is a very challenging task. When the geometric shape of the new dataset is complex or the coarse shape generation stage failed, the performance of our approach will degrade (e.g., the third example in Fig. 12). We also provide qualitative examples from the online products dataset and Internet. Fig. 17 to evaluate the cross-data generalization ability.

4.5 Ablation Study

In this section, we verify the qualitative and quantitative improvements from statistic feature pooling, re-sampled Chamfer distance, and iterative refinement.
4.5.1 Statistical Feature

We first check the importance of using feature statistics. We train MDN with the ordinary concatenation. This maintains all the features loss-less to potentially produce better geometry, but does not support variable number of inputs any more. Surprisingly, our model with feature statistics (Table 7, “Full Model”) still outperforms the one with concatenation (Table 7, “-Feat Stat”). This is probably because that our feature statistics is invariant to the input order, such that the network learns more efficiently during the training. It also explicitly encodes cross-view feature correlations, which can be directly leveraged by the network.

We show the metrics of the MDN with statistics feature or re-sampling loss disabled.

| Metrics             | F-score(τ) ↑ | F-score(2τ) ↑ | CD ↓ |
|---------------------|--------------|---------------|------|
| -Feat Stat          | 65.26        | 79.13         | 0.511|
| -Re-sample Loss     | 66.26        | 80.04         | 0.496|
| Full Model          | 67.32        | 81.22         | 0.381|

4.5.2 Re-Sampled Chamfer Distance

We then investigate the impact of the re-sampled Chamfer loss. We train our model using the traditional Chamfer loss only on mesh vertices as defined in Pixel2Mesh, and all metrics drop consistently (Table 7, “-Re-sample Loss”). Intuitively, our re-sampling loss is especially helpful for places with sparse vertices and irregular faces, such as the elongated
lamp neck as shown in Fig. 13, 3rd column. It also prevents
big mistakes from happening on a single vertex, e.g., the spike
on bench, where our loss penalizes a lot of sampled points on
wrong faces caused by the vertex but the standard Chamfer
loss only penalizes one point.

4.5.3 Number of Iteration

Fig. 15 shows that the performance of our model keeps
improving with more iterations, and is roughly saturated at
three. Therefore we choose to run three iterations during
the inference even though marginal improvements can be
further obtained from more iterations. Moreover, we pro-
vide qualitative examples with different iterations in Fig. 16.

5 Conclusion

We propose a graph convolutional framework to produce
3D mesh model from multiple images. Our model learns to
exploit cross-view information and generates vertex deforma-
tion iteratively to improve the mesh produced from the
direct prediction methods, e.g., Pixel2Mesh and its multi-
view extension. Inspired by multi-view geometry methods,
our model searches in the nearby area around each vertex
for an optimal place to relocate it. Compared to previous
works, our model achieves the state-of-the-art performance,
produces shapes containing accurate surface details rather
than merely visually plausible from input views, and shows
good generalization capability in many aspects. For future
work, combining with efficient shape retrieval for initializa-
tion, integrating with multi-view stereo models for explicit
photometric consistency, and extending to scene scales are
some of the practical directions to explore. On a high level,
how to integrating the similar idea in emerging new represen-
tations, such as part based model with shape basis and
learned function [20] are interesting for further study.

Acknowledgments

Dr. Yanwei Fu is also with and Shanghai Key Lab of Intelli-
gent Information Processing, and Fudan ISTBI–ZJNU Algo-
rum Centre for Brain-inspired Intelligence, Zhejiang Normal
University, Jinhua, China. Chao Wen and Yinda Zhang are
equal contribution to this work.

References

[1] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang,
“Pixel2Mesh: Generating 3D mesh models from single RGB
images,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 55–71.
[2] A. Harlley and A. Zisserman, Multiple View Geometry in Computer
Vision (2.ed.). Cambridge, U.K.: Cambridge Univ. Press, 2006.
[3] A. Kendall, H. Martirosyan, S. Dasgupta, and P. Henry, “End-to-
end learning of geometry and context for deep stereo regression,”
in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 66–75.
[4] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, “DISN: Deep
implicit surface network for high-quality single-view 3D
reconstruction,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 492–502.
[5] C. Wen, Y. Zhang, Z. Li, and Y. Fu, “Pixel2Mesh: Multi-view 3D
mesh generation via deformation,” in Proc. IEEE Int. Conf. Comput.
Vis., 2019, pp. 1042–1051.
[57] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, “Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 3504–3515.

[58] L. Yariv et al., “Multiview neural surface reconstruction by disentangling geometry and appearance,” in Proc. 34th Int. Conf. Neural Inform. Process. Syst., 2020, pp. 2492–2502.

[59] S. Liu, T. Li, W. Chen, and H. Li, “A general differentiable mesh renderer for image-based 3D reasoning,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 1, pp. 50–62, Jan. 2022.

[60] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt, “Neural sparse voxel fields,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 15 651–15 663.

[61] J. L. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, “Mip-NeRF: A multiscale representation for neural radiance fields,” in Proc. IEEE Int. Conf. Comput. Vis., 2021, pp. 5855–5864.

[62] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, “PixelNeRF: Neural radiance fields from one or a few images,” in Proc. Conf. Comput. Vis. Pattern Recognit., 2021, pp. 4578–4587.

[63] A. Yu et al., “AutoInt: A learning feature interaction learning via self-attentive neural networks,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., 2019, pp. 1161–1170.

[64] A. Trevithick and B. Yang, “GRF: Learning a general radiance field for 3D scene representation and rendering,” in Proc. IEEE Int. Conf. Comput. Vis., 2021, pp. 15 182–15 192.

[65] P. Henzler et al., “Unsupervised learning of 3D object categories from data collected in the wild,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 4700–4709.

[66] A. Kendall, M. Grimes, and R. Cipolla, “PoseNet: A convolutional network for real-time 6-DoF camera relocalization,” in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 2938–2946.

[67] J. Wu, L. Ma, and X. Hu, “Delving deeper into convolutional neural networks for camera relocalization,” in Proc. Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5644–5651.

[68] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, “PVNet: Pixelwise voting network for 6DoF pose estimation,” in Proc. IEEE Comput. Vis. Pattern Recognit., 2019, pp. 4561–4570.

[69] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, “On the continuity of rotation representations in neural networks,” in Proc. IEEE Comput. Vis. Pattern Recognit., 2019, pp. 5743–5753.

[70] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning beyond euclidean data,” IEEE Signal Process. Mag., vol. 34, no. 4, pp. 18–42, Jul. 2017.

[71] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Representations, 2017.

[72] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, “Multiview convolutional neural networks for 3D shape recognition,” in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 945–953.

[73] N. Ravi et al., “Accelerating 3D deep learning with PyTorch3D,” 2020, arXiv:2007.08501.

[74] A. X. Chang et al., “ShapeNet: An information-rich 3D model repository,” 2015, arXiv:1512.03012.

[75] E. Isaafudinov and A. Dosovitskiy, “Unsupervised learning of shape and pose with differentiable point clouds,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 2807–2817.

[76] L. Ladicky, O. Saurer, S. Jeong, F. Minnichedda, and M. Pollefeys, “From point clouds to mesh using regression,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 3893–3902.

[77] E. Smith, S. Fujimoto, A. Romero, and D. Meger, “GEOMetrics: Exploiting geometric structure for graph-encoded objects,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 8566–8576.

[78] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in Proc. 23rd Annu. Conf. Comput. Graph. Interactive Techn., 1996, pp. 303–312.

[79] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” in Proc. 14th Annu. Conf. Comput. Graph. Interactive Techn., 1987, pp. 163–169.

[80] W. Chen et al., “Learning to predict 3D objects with an interpolation-based differentiable renderer,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 9609–9619.

[81] S. Koch et al., “ABC: A big cad model dataset for geometric deep learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 9601–9611.

Chao Wen received the BE degree in computer science from the University of Electronic Science and Technology of China, China, in 2018, the master’s degree in computer science from the Academy of Engineering and Technology, Fudan University, China. His research is focused on 3D shape reconstruction and human image synthesis.

Yindia Zhang received the bachelor’s degree from Dept. Automation, Tsinghua University, the master’s degree from Dept. ECE, National University of Singapore co-supervised by Prof. Ping Tan and Prof. Shuicheng Yan, and the PhD degree in computer science from Princeton University, advised by Professor Thomas Funkhouser. He is a senior research scientist with Google. His research interests lie with the intersection of computer vision, computer graphics, and machine learning. Recently, he focuses on empowering 3D vision and perception via machine learning, including dense depth estimation, 3D shape analysis, 3D scene understanding, and neural rendering.

Chenjie Cao received the MS degree in computer science from East China University of Science and Technology, in 2019. He is currently working toward the PhD degree in statistics from Fudan University. His research interests include machine learning, deep learning, 3D shape reconstruction and computer vision.

Zhuwen Li received the BE degree in computer science from Tianjin University, in 2008, the master’s 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 with Nuro, Inc. Recently, he is working on perception in autonomous driving, specifically in 3D structure recovery, motion analysis, point cloud analysis, and object detection.

Xiangyang Xue (Member, IEEE) 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.

Yanwei Fu received the MEng degree from the Department of Computer Science and Technology, Nanjing University, China, in 2011, and the PhD degree from the Queen Mary University of London, in 2014. He held a postdoctoral position with Disney Research, Pittsburgh, Pennsylvania, from 2015 to 2016. He is currently a tenure-track professor with Fudan University. He was appointed as the professor of Special Appointment (Eastern Scholar) with Shanghai Institutions of Higher Learning. His work has led to many awards, including the IEEE ICME 2019 best paper. He published more than 100 journal/conference papers including IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Multimedia, ECCV, and CVPR. His research interests are one-shot learning, and learning based 3D reconstruction.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.