3D Mesh Generation from an RGB Image with Differential Equation Solver

Juhong Min
POSTECH
juhongm999@postech.ac.kr

Abstract

Recent approaches to the task of single-image mesh generation typically involve a mesh deformation network which progressively deforms an initial sphere mesh to get a final geometry corresponding to the input image. One limitation of this network is that it performs only a small number of deformations although deepening the network with skip connections is current de facto standard for learning robust representation. Another limiting factor is that the number of deformations is always fixed regardless of the geometric and semantic complexity of the mesh to predict. In this work we introduce a novel mesh deformation network in which the number of deformations may grow or shrink with constant memory cost by parameterizing the derivative of a mesh using differential equation solver. With such approach, we demonstrate the amount of deformations may adaptively change with advanced numerical methods.

1. Introduction

Understanding 3D world behind 2D images is a fundamental yet challenging problem in 3D vision. The advent of deep neural networks (DNN) brought a great success to recent methods [11, 15, 16] for generating a 3D representation, *e.g.*, a mesh. A typical way of constructing a mesh that corresponds to the input image is to iteratively deform an initial mesh, a sphere, into a desired geometry, using convolutional features extracted from 2D CNN. However, the existing deformation networks primarily focus on learning transformations of input vertices rather than learning residuals [8], *i.e.*, the amount of transformations required at a particular layer, a mesh gradient in our case.

Deep networks with skip connections are proven to be significantly effective in learning robust representation [2, 8, 12, 14]. Recent work [4] defines the ResNet as an ordinary differential equations (ODE), exposing various advantages of such design. In this work, we propose a mesh deformation network that learns to predict mesh gradients by framing the gradient prediction network as an ODE as illustrated in Fig. 1. We also discuss several benefits of defining mesh deformation network using ODE solver.

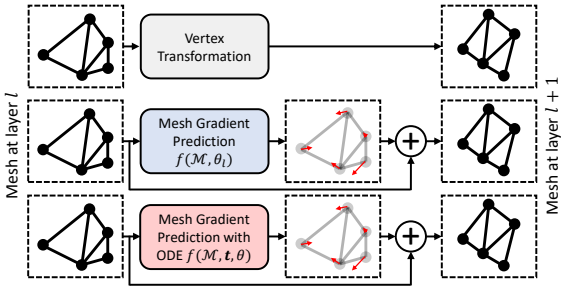

Figure 1. Existing mesh deformation networks typically involve vertex transformation (top). We show this network can seamlessly be converted into a mesh gradient predictor (middle). In this work we remodel the gradient predictor into an ordinary differential equation solver with a continuous time parameter (bottom).

2. Related Work

Single-image mesh generation. Inferring 3D representation from an image is a widely researched area in computer vision. Having a small memory cost while preserving shape details, mesh is a natural choice to efficiently represent a 3D shape. Wang *et al*. [15] propose a graph-based network to predict 3D mesh given an image by progressively deforming a sphere. The use of sphere deformation networks became a central paradigm in recent approaches [7, 11, 16]. These networks, however, mainly perform iterative vertex transformation at each layer without learning any residuals [8] although skip connections enable robust representation learning by preserving identity mapping and safe gradient flow, incorporating deeper network design [2, 12, 14]. In this aspect, conventional mesh deformation networks in the task of single-image mesh generation have a fundamental limitation in learning to predict reliable 3D representations.

Neural ordinary differential equation. Chen *et al*. [4] recently introduced a new family of deep neural networks, neural ordinary differential equations. Viewing each residual feature as a tiny gradient of the final prediction, they frame the entire network as a function of time with a gradient prediction network, ODE solver. In this paper we show the idea of neural ODE can effectively adopted to the task of single-image mesh generation.

3. Proposed Method

The proposed framework for single-image mesh generation consists of three main modules: (1) perceptual feature pooling, (2) mesh gradient prediction network, and (3) training objective. In this section, we demonstrate each module in detail and how we transform the mesh gradient prediction network into an ODE solver.

3.1. Perceptual feature pooling

Given an input image $I \in \mathbb{R}^{H\times W\times C}$, a feature extraction network extracts a series of intermediate feature blocks. We concatenate the feature blocks along channels with upsampling and denote the resultant feature vectors at each spatial position p by $\{\mathbf{f}_p\}_{p=1}^{H\times W}$. Given a mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V} = \{\mathbf{v}_i\}_{i=1}^{V}$ and edges $\mathcal{E} = \{\mathbf{e}_i\}_{i=1}^{E}$ connecting two vertices, we project 3D coordinates of the vertices $\mathcal{V}$ on input image plane using camera intrinsics and pool feature vectors nearest to each projected vertex as in [15]:

$$\mathcal{V}_\mathrm{f} = \{[\mathbf{v}_i; \mathbf{f}_i]\}_{i=1}^{V} \tag{1}$$

where $[\cdot;\cdot]$ is channel-wise concatenation and $\mathbf{f}_i$ is a feature vector spatially nearest to the coordinate of projected vertex $\mathbf{v}_i$. We now pass the obtained vertex features with the edges, $\mathcal{M}_\mathrm{f} = (\mathcal{V}_\mathrm{f}, \mathcal{E})$, as an input to our mesh gradient prediction network f.

3.2. Mesh gradient prediction network

To predict mesh gradients at layer l, a mesh $\mathcal{M}_\mathrm{f}^l$ is passed to 2-layer graph convolutional network (GCN) [1] f which outputs the amount of deformation required at l-th layer:

$$\frac{d\mathcal{M}^l}{dl} = f(\mathcal{M}_\mathrm{f}^l, \theta^l). \tag{2}$$

The resultant gradient is simply added to the input mesh:

$$\mathcal{M}^{l+1} = \mathcal{M}^l + f(\mathcal{M}^l, \theta^l) \tag{3}$$

where θ^l is learnable parameters of l-th gradient prediction network. In Sec. 3.4, we demonstrate that we can reparameterize the derivative of mesh by introducing a continuous time parameter t, viewing Eq. 3 as a first-order linear ODE.

3.3. Training objective

Following the work of [15], we use two objectives to train our model: chamfer distance loss and edge length loss.

Chamfer distance loss. Given two sets of vertices, $\mathcal{V}$ and $\mathcal{V}'$, Chamfer distance measures a sum of Euclidean distances between a vertex $\mathbf{v} \in \mathcal{V}$ and the nearest vertex in the other set $\mathbf{v}' \in \mathcal{V}'$:

$$\mathcal{L}_\mathrm{c} = \sum_{\mathbf{v}\in\mathcal{V}} \min_{\mathbf{v}'} ||\mathbf{v} - \mathbf{v}'||^2 + \sum_{\mathbf{v}'\in\mathcal{V}'} \min_{\mathbf{v}} ||\mathbf{v} - \mathbf{v}'||^2 \tag{4}$$

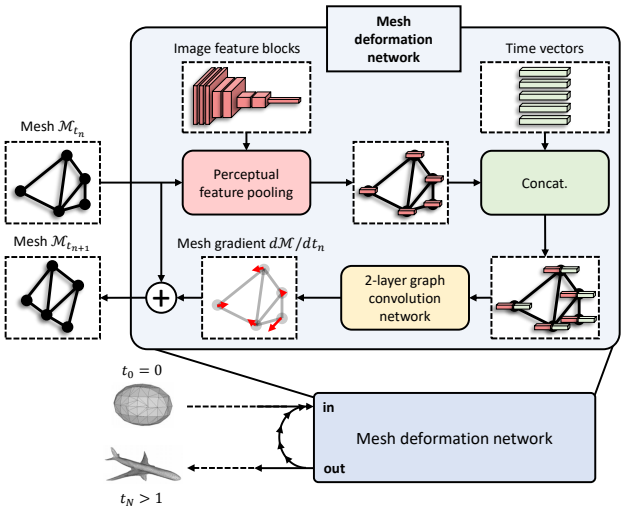

Figure 2. Overall framework of our mesh deformation network.

Edge length loss. We minimize the Euclidean distance between a vertex $\mathbf{v} \in \mathcal{V}$ and its neighboring vertices in $\bar{\mathbf{v}} \in \mathcal{V}$ in order to penalize outliers, thus preventing lengthy edges:

$$\mathcal{L}_\mathrm{e} = \sum_{\mathbf{v}} \sum_{\bar{\mathbf{v}}\in\mathcal{N}(\mathbf{v})} ||\mathbf{v} - \bar{\mathbf{v}}||^2 \tag{5}$$

The overall training objective of our model is defined as: $\mathcal{L} = \lambda_\mathrm{c}\mathcal{L}_\mathrm{c} + \lambda_\mathrm{e}\mathcal{L}_\mathrm{e}$ where λ_c and λ_e weights for the losses.

3.4. Gradient prediction with ODE solver

As most differential equations which appears in the real world are difficult to solve analytically, it is common to find approximation to the solution *numerically*. The simplest and oldest method for ODE is Euler's method; given an initial value $y(t_0) = y_0$ at time t_0 and an ODE $y'(t) = f(t, y(t))$, we compute the numerical estimates at subsequent time steps recursively with step size h as follows:

$$y(t_{n+1}) = y(t_n) + h \cdot f(t_n, y(t_n)) \tag{6}$$

which shares the same mathematical structure with our progressive mesh deformation model (Eq. 3):

$$\mathcal{M}^{l+1} = \mathcal{M}^l + h \cdot f(\mathcal{M}^l, \theta^l) \tag{7}$$

by introducing the time step h. We can now transform Eq. 3 into a function of time, placing solution trajectory in a continuous, high-dimensional vector field [4]

$$\mathcal{M}^{t_{n+1}} = \mathcal{M}^{t_n} + h \cdot f(t_n, \mathcal{M}^{t_n}, \theta) \tag{8}$$

where the mesh gradient prediction network f preserves the similar network architecture, 2-layer GCN, but now with an additional continuous time parameter t_n. In practice, we turn time constant into a vector by replication such that $\mathbf{t}_n \in \mathbb{R}^T$ and concatenate to every vertex feature vector in $\mathcal{V}_\mathrm{f}$ as follows: $\mathcal{V}_\mathrm{t} = \{[\mathbf{v}_i; \mathbf{f}_i; \mathbf{t}_n]\}_{i=1}^{V}$. The overall framework of our approach with ODE solver is illustrated in Fig. 2.

Category	Chamfer distance (CD)					
	3D-R2N2 [5]	PSG [6]	N3MR [9]	P2M [15]	P2M* [15]	$\text{ODE}_{\text{d}}^{\text{euler}}$ (ours)
plane	0.895	0.430	0.450	0.477	**0.383**	0.408
# param.	$O(L)$	$O(L)$	$O(L)$	$O(L)$	$O(L)$	**O(1)**

Table 1. Results on ShapeNet [3]. Some results are from [15]. Following the work of https://github.com/nywang16/Pixel2Mesh/blob/master/eval_testset.py (Line. 132-146), the CD is in units of 10^{-3}. P2M*: P2M trained using samples of plane category only.

4. Experiments

In this section, we compare the method with the state of the art and discuss the results.

Implementation detail. We implemented the proposed framework on top of the existing code in repository online which is available at https://github.com/Tong-ZHAO/Pixel2Mesh-Pytorch. We used Adam optimizer [10] with learning rate decay and VGG-16 [13] style feature extraction network following the work of [15]. V and T are respectively set to 2466 and 196. We set weights for the two losses as $\lambda_c = 100$ and $\lambda_e = 0.1$ and used Euler's method for our experiments if not specified otherwise.

Training the ODE solver. At each time step t_n, the step size is sampled from a uniform distribution $U(h_{\text{lo}}, h_{\text{hi}})$ in order to prevent a rapid mode collapse. To see the effects of the number of deformations, we conduct two types of experiments: *shallow* and *deep* in which step size is sampled from $U(0.2, 0.3)$ and $U(0.05, 0.2)$ respectively. Starting with the initial time step sampled from $U(0.0, 0.25)$, our model escapes the mesh deformation loop when the time step exceeds 1 ($t_N > 1$) as shown in Fig. 2. Note, the number of forward evaluations N, *i.e.*, mesh deformations, is entirely dependant on the distribution where we sample h from.

Datasets and evaluation metrics. We train and evaluate our model on ShapeNet [3] benchmark dataset which contains 50k models belonging to 13 object categories. Due to its large amount of training samples, we use samples from only plane category[1] out of 13. Chamfer distance is used to evaluate predictions of our model (lower the better).

4.1. Results and comparisons

Table 1 summarizes our results and those of recent methods [5, 6, 9, 15] and Fig. 3 illustrates qualitative results. Both quantitative and qualitative results prove that it apparently learns to find a reliable solution in a continuous time space with constant memory cost, exposing potential performance gains with better network design for ODE solver. However, we believe that further evaluations are required to verify the superiority of our model, *e.g.*, evaluations on samples of the other categories followed by additional standard evaluation metrics such as EMD and F-scores.

[1]Even in multi-GPU setting, it takes a couple of weeks for our model to train all the samples so we train plane category only for 10 iterations.

Model	$\text{ODE}_{\text{d}}^{\text{euler}}$	RES_{d}	RES_{s}	$\text{ODE}_{\text{s}}^{\text{euler}}$	$\text{ODE}_{\text{s}}^{\text{heun}}$	$\text{ODE}_{\text{s}}^{\text{midpt}}$
CD	0.408	0.354	2.566	16.940	11.711	12.558
NFE	7.55	8	4	4.01	4.01	4.02
# param.	$O(1)$	$O(L)$	$O(L)$	$O(1)$	$O(1)$	$O(1)$

Table 2. Different model comparisons. CD: chamfer distance, NFE: the average number of forward evaluations. The subscripts indicate the type of experiments (d: deep and s: shallow).

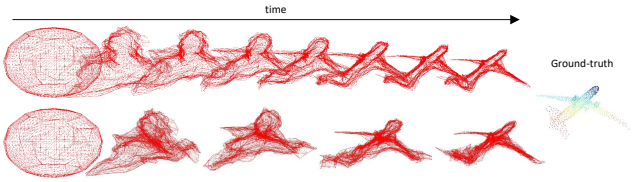

Figure 3. Mesh deformation of a sample in continuous time space with step sizes sampled from $U(0.05, 0.2)$ (top) and $U(0.2, 0.3)$ (bottom). Its continuous deformation with tiny step sizes sampled from $U(0.001, 0.005)$ is visualized at https://youtu.be/St0wEBP8S2c. Best viewed in electronics.

Mesh deformation with naïve residuals. To see the benefits of using ODE solver for mesh generation, we compare with *discrete* version of our model in which multiple (8 and 4) mesh gradient prediction networks are stacked with skip connections without the time parameter t_n like an ordinary ResNet architecture [8]. The results are shown in third and fourth columns of Tab. 2. The comparison between results of RES_{d}, RES_{s}, and $\text{ODE}_{\text{d}}^{\text{euler}}$ reveals that larger number of forward evaluations clearly provides better prediction.

Training with advanced numerical method. Fifth to last columns of Tab. 2 show results of our model trained with Euler's, Huen's, and midpoint method respectively (shallow) where Heun's and midpoint method provide better approximation to solutions than Euler's. The experimental results reveal that use of modern numerical methods for ODE may promote extra performance gains.

4.2. Benefits of using neural ODE

The use of advanced numerical methods, *e.g.*, 4-th order Runge-Kutta and Dormand Prince, help ODE solvers find accurate solutions by adaptively deciding step sizes using explicit error estimation. A constant parameter size of a model regardless of the number of forward evaluations is another advantage. Chen *et al*. [4] also show that the adjoint sensitivity method during backward pass facilitates efficient training in terms of both training time and memory.

5. Conclusion

We showed that the idea of neural ordinary differential equation can seamlessly be adopted in the task of single-image mesh generation. We believe that more careful network design choice for ODE solver along with advanced numerical methods will further facilitate the quality of the predicted geometry. We leave this investigation to future work.

References

[1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, July 2017.

[2] Remi Cadene, Hedi Ben-Younes, Nicolas Thome, and Matthieu Cord. Murel: Multimodal Relational Reasoning for Visual Question Answering. In *CVPR*, 2019.

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[4] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *NeurIPS*, 2018.

[5] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[6] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[7] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*, 2018.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[9] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations (ICLR)*, 2014.

[11] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep mesh reconstruction from single rgb images via topology modification networks. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, pages 5998–6008. Curran Associates, Inc., 2017.

[15] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018.

[16] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *ICCV*, 2019.