Fast mesh denoising with data driven normal filtering using deep autoencoders

Stavros Nousias , Gerasimos Arvanitis, Aris S. Lalos [1], and Konstantinos Moustakas
[1] *Industrial Systems Institute, Athena Research Center*
{lalos}@isi.gr
Department of Electrical & Computer Engineering University of Patras, Greece
{nousias, arvanitis, moustakas}@ece.upatras.gr

***Abstract*—Through the years, several works have demonstrated high-quality 3D mesh denoising. Despite the high reconstruction quality, there are still challenges that need to be addressed ranging from variations in configuration parameters to high computational complexity. These drawbacks are crucial especially if the reconstructed models have to be used for quality check, inspection or repair in manufacturing environments where we have to deal with large objects resulting in very dense 3D meshes. Recently, deep learning techniques have shown that are able to automatically learn and find more accurate and reliable results, without the need for setting manually parameters. In this work, motivated by the aforementioned requirements, we propose a fast and reliable denoising method that can be effectively applied for reconstructing very dense noisy 3D models. The proposed method applies conditional variational autoencoders on face normals. Extensive evaluation studies carried out using a variety of 3D models verify that the proposed approach achieves plausible reconstruction outputs, very relative or even better of those proposed by the literature, in considerably faster execution times.**

***Index Terms*—3D mesh denoising, data driven normal filtering, variational autoencoders.**

I. Introduction

Recent advances in 3D scanning devices have allowed breakthroughs in the manufacturing industry. Quality assurance, reverse engineering, autonomous manufacturing, machine vision guided tasks benefit from the dynamic reconstruction of dense geometric representations of physical objects [1]. Although 3D scanning technology constantly improves, these devices generate a huge amount of noisy data that should be filtered using fast and efficient approaches. These challenging issues highlight the need for parallelizable computationally inexpensive, and accurate approaches for mesh denoising.

Motivated by the aforementioned requirements, we focus on providing a fast approach for mesh denoising based on data-driven normal filtering using deep conventional and variational autoencoders able to handle efficiently very dense models in execution times considerably lower than those achieved by state of the art denoising approaches. More specifically, the contributions of the proposed approach can be summarized in the following points: 1) It can be utilized in industrial applications for performing denoising of dense objects with features such as corners and edges. 2) It allows fast and efficient denoising of large meshes with relatively small training set. 3) It has lower complexity with regards to other state-of-the-art approaches. Evaluation studies carried out using scanned and CAD 3D industrial models, verify the effectiveness of the proposed method as compared to other recent and relevant approaches in terms of both reconstruction quality and computational efficiency.
The rest of this paper is organized as follows: Section II presents state-of-the-art methods and related works. Section III describes in detail the workflow of the proposed approach. Section IV is dedicated to the experimental setup and simulation results while conclusions are drawn in Section V.

II. Related Work

Nowadays, manufacturing has evolved to meet a great expansion in collected sensor data required to be processed and stored. Gathered from multiple endpoints, such data streams are essential to support smart manufacturing and optimized automation in production lines. Wang et al. [2] highlighted this emerging connection between deep learning and Industry 4.0 [3]. Inline 3D scanning for production line inspection and rapid prototyping facilitates constructed parts to be examined in many stages of the manufacturing process and filtered in real time with online decision support systems [4]. Cases of quality assurance protocols engaging scanning technology in automotive industry [5], maintenance in maritime industry [6] and automated reverse engineering [7] prove that error-free representations constitute a requirement for industrial sensors outcomes.

In a classic scenario, scanners yield noisy point clouds that are consequently converted to noisy 3D meshes. Mesh denoising aims to remove the noise while preserving features and multi-scale geometric details. Several methods are available in the literature with significant denoising results. Yet the need for robust and fast algorithms able to handle dense models rapidly becomes more essential in industrial applications, where these methods are expected to significantly reduce the operational cost of many manufacturing tasks. Isotropic mesh filtering [8], graph spectral processing [9]–[11], bilateral normal filtering based [12], [13] and iterative schemes [14]–[17] are among the known approaches. Although this category of approaches preserves most of the sharp features, they require heavy parameterization and fail to generalize. Furthermore, regularization based [18], L_0 minimization based methods [19] and cascaded approaches [20] offer good results but at great cost. Recent learning based approaches offer very good results that are only limited by the extraction of

978-1-7281-2927-3/19/$31.00 ©2019 IEEE

required features and descriptors, pre-processing tasks [21]–[23] and size of the dataset [24]. Our approach aims to be applied directly on the mesh nodes avoiding preprocessing, thus contributing to the field of geometric deep learning where the sampling of the latent space is non-uniform.

III. Deep autoencoders for 3D mesh denoising

A. Deep autoencoders

Deep autoencoders encompass a multi-layer neural network architecture where the hidden layers encode the input to a latent space and decode the latter to a reconstructed input. A deep autoencoder is composed of two, symmetrical deep-belief networks [25] that typically have three to five shallow layers for the encoding and the decoding part. The layers are restricted Boltzmann machines [26] while the building blocks of deep-belief networks. Variational autoencoders (VAE) [27] assume that the input vectors are generated by some random process of an unobserved continuous random variable $\mathbf{z}$. The parameters of the VAE are estimated efficiently by the stochastic gradient variational Bayes framework [27]. An improvement of the VAE is the conditional variational autoencoder (CVAE) [28] where the encoder and the decoder are conditioned under $\mathbf{x}$ and the label of $\mathbf{x}$ denoted as c.

B. Data preprocessing

In this study, we focus on triangular meshes $\mathcal{M}$ with n vertices $\mathbf{v}$ and n_f faces f. Each vertex $\mathbf{v}_i$ is denoted by $\mathbf{v}_i = [x_i,\ y_i,\ z_i]^T$, $\forall\ i = 1, \cdots, n$. Each f_j face is a triangle that can be described by its centroid $\mathbf{c}_j$ and its outward unit normal $\mathbf{n}_{c_i}$. To create the input and the output for the autoencoder we form sets of n topologically closest neighbours $\mathcal{P}_i$. Matrix $\mathbf{N}_i^{3\times(n+1)}, i \in \mathcal{P}_i$ contains the face normals for all the faces comprising the patch. The normals of each patch are rotated by angle δ_{n_i} around rotation axis $\mathbf{a}_{n_1}$ so that $\frac{1}{N}\sum_{i\in\mathcal{N}_i} A_i \cdot \mathbf{n}_{c_i} = \mathbf{c}$ where A_i is the area of i_{th} face and $\mathbf{c}$ is a constant vector $\mathbf{c} = [1\ 0\ 0]$. The motivation behind rotating each patch towards the same direction is that it allows efficient training with smaller training sets. Otherwise, we would have to add to the training set patches with every possible average normal direction resulting in very large datasets. Thus, the training set contains pairs of noisy and noise-free patches rotated by δ_{n_i} around rotation axis $\mathbf{a}_{n_1}$ defined by the normals of the noisy patch. The normalized normal vectors ranging in $[-1, 1]$ are transformed to $[0, 1]$ by the following expression $\mathbf{n}'_{c_i} = 2\cdot\mathbf{n}_{c_i} - [1\ 1\ 1]$ Finally, input for the autoencoder is the matrix $\mathbf{N}_i^{3\times(n+1)}$ reshaped to $\mathbf{Z}_i^{3(n+1)\times 1}$.

C. Autoencoder architecture and training

For the mesh denoising, a CVAE following the architecture of [28] was employed. Two dense layers for a conditional Gaussian encoder and for a conditional Bernoulli decoder were used with $N = [500, 500]$. Each dense layer is succeeded by a leaky ReLU operation and a dropout function. In order to generate labels for the training set, we perform K-means clustering [29]. The motivation behind applying K-means clustering is that it splits the dataset into groups of patches with high and low curvature, flat areas, and features i.e corners, allowing the creation of different models for each label.

At first, both noisy and ground truth dataset are appropriately prepared, namely, we generate and rotate patches. Then, the data are provided as input in a CVAE architecture. Although time-consuming, this process takes place only once. After the autoencoder has generated the filtered output $\mathbf{Z}_i^{3(n+1)\times 1}$, they are reshaped back to initial dimensions $\mathbf{Z}_i^{3\times(n+1)}$. The exported filtered normal vector for patch $\mathcal{P}_i$ is the first column of $\mathbf{Z}$, and more specifically $\hat{\mathbf{n}}_{c_i}^{3\times 1} = \mathbf{Z}_i[:, 0]$. Then each patch is rotated by the opposite angle $-\delta_{n_i}$ and the same axis $\mathbf{a}_{n_i}$ they were rotated with in the first place and transformed back to range $[-1, 1]$ using $\mathbf{n}_{c_i} = \left(\mathbf{n}'_{c_i} + [1\ 1\ 1]\right)/2$. As a final post processing step for the mesh denoising process, we use the bilateral filtering approach [17] performing only a single iteration while for the vertex updating step we perform 20 iterations. The motivation for performing a single iteration of bilateral filtering is that it allows fine tuning by removing very small artifacts. More iterations increase the computational cost without any additional benefit.

IV. Experimental analysis and simulation results

A. Experimental setup and training

Eight meshes were selected for the training of the autoencoder architecture, comprising in total of 1,977,740 patches. Noisy meshes were synthesized by adding Gaussian noise $N\ \sim\ (0,\ \ 0.1 \cdot \bar{L}_p)$ co-directional to each vertex normal direction. 1,977,740 training pairs of noisy and noise-free rotated patches were utilized for training of the autoencoders. Two configurations were tested for patch size, $n = 8$ and $n = 20$ neighbours. For the K-means clustering of the CVAE, pipeline $K = 200$ was selected. For the optimization method Adam Optimizer [30] was used with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e-8$. The learning rate started from 0.00002 and decreased with a decay rate of 0.998 per epoch. The training took place for 12 epochs. For the training of the autoencoder models an NVIDIA GeForce GTX 1080 graphics card was used with 8GB VRAM and compute capability 6.1.

B. Mesh denoising

To test the denoising capability of our method we generated synthetic noise to five models and compared our results to guided mesh normal filtering [14], bilateral normal filtering [16], L_0 minimization mesh denoising [19] and fast and effective mesh denoising [15]. As an additional comparison, the CVAE part of our pipeline was replaced with a traditional 5-layer deep autoencoder with $N = [256, 128, 64, 128, 256]$ the number of neurons for each layer. An element-wise sigmoid operation succeeds each layer and a mean square error loss function is employed.

To evaluate the reconstructed results we employ the following metrics: 1) The Hausdorff distance (HD) which represents the average one-side distance between the reconstructed and the original 3D mesh. 2) The average angle difference α between the normals of the ground truth and the reconstructed model. 3) A visualization presenting with different colors the

TABLE I: Hausdorff distance and metric α (in degree) using disparate approaches of initialization and deep architectures.

	AE				CVAE			
	8 nn	8 nn + pp	20 nn	20 nn + pp	8 nn	8 nn + pp	20 nn	20 nn + pp
Chinese-lion	**0.27752**, 8.93^o	0.28478, **8.79°**	0.34635, 9.10^o	0.40453 , 9.88^o	0.35064, 9.24^o	0.36592, 9.08^o	0.37312, 9.47^o	0.38996,10.07^o
Rockerarm	**0.00490**, 8.85^o	0.00530, 7.19^o	0.00691, 8.44^o	0.00699, 8.49^o	0.00507, 8.47^o	0.00503, **7.18°**	0.00759, 7.83^o	0.00689, 7.68^o
Sculpt	0.01408, 9.85^o	0.01273, 7.74^o	0.01706, 12.60^o	0.01699, 11.99^o	0.00823, 5.64^o	**0.00595**, **3.83°**	0.01178, 5.50^o	0.00942, 4.17^o
Gear	**0.13519**, 8.10^o	0.13801, 6.89^o	0.16017, **5.98°**	0.16651, 6.06^o	0.14779, 8.53^o	0.14298, 7.29^o	0.15948, 6.62^o	0.16620, 6.53^o
Trim-star	0.26388, 10.12^o	0.29083, 8.26^o	0.30939, 11.24^o	0.31648, 10.86^o	0.28893, 8.86^o	**0.25300**, **7.01°**	0.30534, 9.37^o	0.32031, 8.27^o

TABLE II: Execution time for presented approaches (measured in seconds).

	Bilateral normal filt.	Guided mesh normal filt.	Fast and effective	L_0 minimization	CVAE 20 pp	CVAE 8 pp
Sculpt(3669V,7342F)	0.1082	0.6465	0.0591	3.5884	0.0772	0.0754
Trimmed star(5192V, 10384F)	0.1529	0.9843	0.0869	4.2748	0.0995	0.0995
Rocker Arm(9413V,18826F)	0.3242	2.0561	0.1804	11.1609	0.1642	0.1617
Chinese Lion(50K V, 100K F)	2.0508	21.6360	1.5792	110.6100	0.9872	0.9624
Gear(250K V,500K F)	8.5630	221.1910	5.7120	2512.2500	3.8858	3.7505

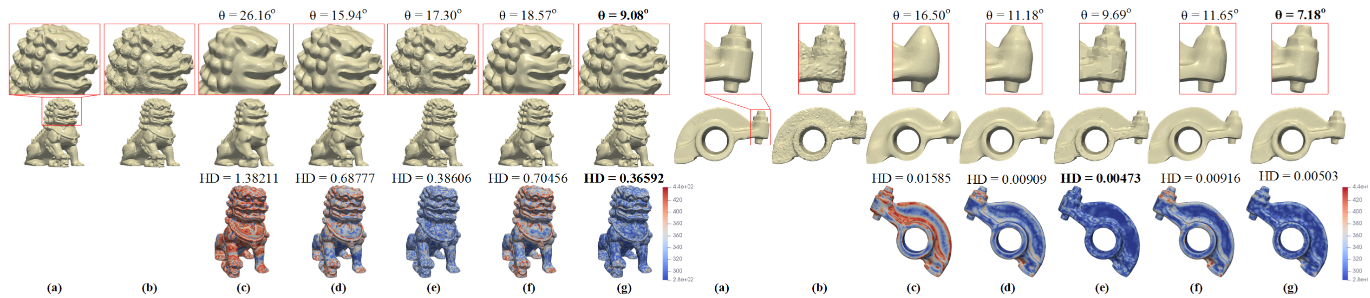


Fig. 1: Denoising results, enlarged details and visualization of the absolute distance per vertex between reconstructed and original 3D mesh for different 3D models (i.e., Chinese lion, Rocker Arm). (a) Original mesh, (b) noisy mesh, (c) fast and effective [15], (d) bilateral normal filtering [16], (e) L_0 minimization [19], (f) guided normal filtering [14], and (g) our approach.

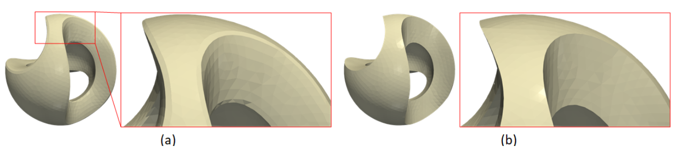


Fig. 2: Denoising results using: (a) AE, (b) CVAE.

absolute difference between reconstructed and original mesh. 4) Execution times to evaluate computational complexity using the open source implementation of [14] available in [31]. The autoencoder part of our pipeline is implemented in Python TensorFlow, while the bilateral normal filtering and vertex update in C++. The evaluation studies took place in a Intel(R) Core(TM) i7-4790 CPU @ 3.60Hz with 32GB of RAM.

Table I presents the Hausdorff distance and the mean α of the normals angular difference between original and denoised 3D models. We compare two different deep architectures (i.e., AE and CVAE), in two different patch sizes (i.e., with 8 and 20 nearest neighbours (nn)), and with or without post-processing step (pp). The value of the approach with the best performance is highlighted using bold. As we can observe, the best performance depends on the model and none of these approaches is universally the best. Nevertheless, in most of the cases, the CVAE using 8 nn for the creation of each patch and applying a post-processing step seems to have the most stable behaviour. Comparing the reconstructed meshes provided by AE and CVAE, we notice that simple AE gives a smoothed result to the object's surface but negatively affects the preserving of features, as shown in Figure 2. On the other hand, CVAE achieves to preserve the geometry of the features

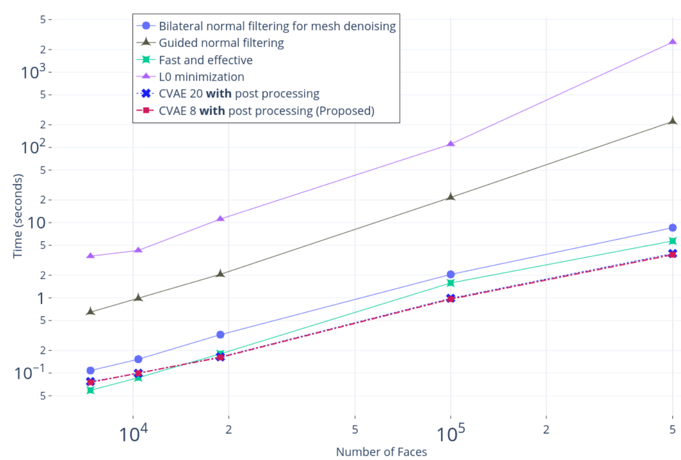
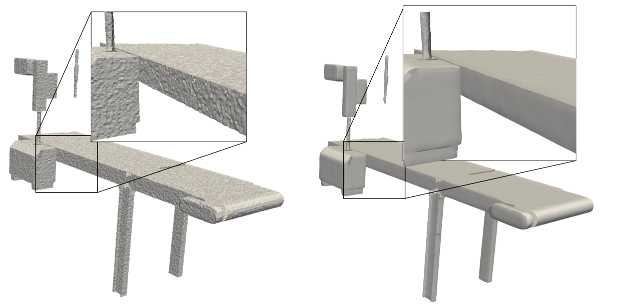


Fig. 3: Performance evaluation

but the surface of flat areas is not perfectly smoothed and some artifacts are apparent. However, this is not a problem because the post-processing step is able to remove these artifacts.

C. Performance evaluation

As Figure 3 and Table II show, our method is much faster than L_0 minimization [19] and Guided Normal Filtering [14]. Compared to fast and effective mesh denoising and traditional bilateral normal filtering our method is slower in small models but becomes faster as the number of faces increases. In the case of Rocker Arm 1 counting 18826 faces, our method outperforms all the other approaches. This can be explained by the fact that for the denoising of a single patch the autoencoder



(a) (b)

Fig. 4: (a) Noisy and (b) Denoised conveyor belt 3D model

part exhibits $O(1)$ complexity removing a large portion of the computational cost. Execution time measurements presented in Table II were computed as the mean value of 10 repetitions.

V. Discussion

It is a common trend for novel modelling and digital twins methodologies to scan industrial workplaces and office spaces [32] facilitating ergonomics optimization and real-world digitization. Figure 4 demonstrates a use case related to factory environments presenting the denoising result of a synthetic noisy conveyor belt model. In this work, we presented a very fast data-driven denoising approach using conditional variational autoencoders to patches of neighbouring normals. The main advantages of the proposed method include (i) being parameter-free, since every used parameter is predefined (i.e.,patch size), (ii) the capability to be used for any type of noise with different patterns, (iii) allowing a relatively small dataset size for the training process in comparison with other data-driven methods while at the same time, (iv) being very fast for large and dense models compared to other approaches.

Acknowledgment

This research has been funded by the Research and Innovation Action project AGEING@WORK, implemented under European Unions Horizon 2020 H2020-SC1-DTH-03-2018 grant agreement no 826299 and the DEEP-EVIoT - Deep embedded vision using sparse convolutional neural networks project (MIS 5038640) implemented under the Action for the Strategic Development on the Research and Technological Sector, co-financed by national funds through the Operational programme of Western Greece 2014-2020 and European Union funds (European Regional Development Fund).

References

[1] F. Tao, F. Sui, A. Liu, Q. Qi, M. Zhang, B. Song, Z. Guo, S. C.-Y. Lu, and A. Nee, "Digital twin-driven product design framework," *International Journal of Production Research*, pp. 1–19, 2018.

[2] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, 2018.

[3] B. Bajic, I. Cosic, M. Lazarevic, N. Sremcev, and A. Rikalovic, "Machine learning techniques for smart manufacturing: Applications and challenges in industry 4.0," *Department of Industrial Engineering and Management Novi Sad, Serbia*, p. 29.

[4] J. Dekhtiar, A. Durupt, M. Bricogne, B. Eynard, H. Rowson, and D. Kiritsis, "Deep learning for big data applications in cad and plm–research review, opportunities and case study," *Computers in Industry*, vol. 100, pp. 227–243, 2018.

[5] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, "Deep learning in the automotive industry: Applications and tools," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 3759–3768.

[6] D. Deahl, "The royal netherlands navy is 3d scanning all their ships," Oct 2017. [Online]. Available: https://www.theverge.com/2017/10/1/16387528/royal-netherlands-navy-dutch-3d-scan-ships-artec

[7] M. Nazarczuk, M. Cader, M. Kowalik, and M. Jankowski, "Proposition of the methodology of the robotised part replication implemented in industry 4.0 paradigm," in *Conference on Automation*. Springer, 2019, pp. 457–472.

[8] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. Citeseer, 1999, pp. 317–324.

[9] O. Sorkine, "Laplacian mesh processing," in *Eurographics (STARs)*, 2005, pp. 53–70.

[10] H. Zhang, O. Van Kaick, and R. Dyer, "Spectral mesh processing," in *Computer graphics forum*, vol. 29, no. 6. Wiley Online Library, 2010, pp. 1865–1894.

[11] P.-S. Wang, X.-M. Fu, Y. Liu, X. Tong, S.-L. Liu, and B. Guo, "Rolling guidance normal filter for geometric processing," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 173, 2015.

[12] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," in *ACM transactions on graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 950–953.

[13] T. R. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature-preserving mesh smoothing," in *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 943–949.

[14] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, and L. Liu, "Guided mesh normal filtering," in *Computer Graphics Forum*, vol. 34, no. 7. Wiley Online Library, 2015, pp. 23–34.

[15] X. Sun, P. Rosin, R. Martin, and F. Langbein, "Fast and effective feature-preserving mesh denoising," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 5, pp. 925–938, 2007.

[16] Y. Zheng, H. Fu, O. K.-C. Au, and C.-L. Tai, "Bilateral normal filtering for mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 10, pp. 1521–1530, 2011.

[17] G. Arvanitis, A. S. Lalos, K. Moustakas, and N. Fakotakis, "Feature preserving mesh denoising based on graph spectral processing," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 3, pp. 1513–1527, 2019.

[18] H. Zhang, C. Wu, J. Zhang, and J. Deng, "Variational mesh denoising using total variation and piecewise constant function space," *IEEE transactions on visualization and computer graphics*, vol. 21, no. 7, pp. 873–886, 2015.

[19] L. He and S. Schaefer, "Mesh denoising via l 0 minimization," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 64, 2013.

[20] M. Wei, L. Liang, W.-M. Pang, J. Wang, W. Li, and H. Wu, "Tensor voting guided mesh denoising," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 931–945, 2017.

[21] K. Sarkar, K. Varanasi, and D. Stricker, "3d shape processing by convolutional denoising autoencoders on local patches," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1925–1934.

[22] W. Zhao, X. Liu, Y. Zhao, X. Fan, and D. Zhao, "Normalnet: Learning based guided normal filtering for mesh denoising," *arXiv preprint arXiv:1903.04015*, 2019.

[23] J. Wang, J. Huang, F. L. Wang, M. Wei, H. Xie, and J. Qin, "Data-driven geometry-recovering mesh denoising," *Computer-Aided Design*, 2019.

[24] P.-S. Wang, Y. Liu, and X. Tong, "Mesh denoising via cascaded normal regression." *ACM Trans. Graph.*, vol. 35, no. 6, pp. 232–1, 2016.

[25] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[26] ——, "A practical guide to training restricted boltzmann machines," pp. 599–619, 2012.

[27] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[28] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in neural information processing systems*, 2015, pp. 3483–3491.

[29] H.-H. Bock, "Clustering methods: a history of k-means algorithms," in *Selected contributions in data analysis and classification*. Springer, 2007, pp. 161–172.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] B. D. Wangyu Zhang. (2015) Mesh denoising ui. [Online]. Available: https://github.com/bldeng/GuidedDenoising

[32] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.