Fast mesh denoising with data driven normal filtering using deep autoencoders

Stavros Nousias, Gerasimos Arvanitis, Aris S. Lalos †, and Konstantinos Moustakas

† 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
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 pre-processing,
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 ran-
don process of an unobserved continuous random variable 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 x and the label of x denoted as c.

B. Data preprocessing

In this study, we focus on triangular meshes M with n
vertices v and nf faces f. Each vertex vi is denoted by vi =
[xi, yi, zi]T, ∀ i = 1, · · · , n. Each fj face is a triangle that
can be described by its centroid cj and its outward unit normal
ncj. To create the input and the output for the autoencoder we
form sets of n topologically closest neighbours Pi. Matrix
N3×(n+1), i ∈ Pi contains the face normals for all the faces
comprising the patch. The normals of each patch are rotated by
angle δni around rotation axis an, so that 1 n

N

i ∈P


· 1

A_i · nc_i = c

where A_i is the area of ith face and c is a constant
vector 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
δni around rotation axis an, defined by the normals of the noisy
patch. The normalized normal vectors using

N3×(n+1) are transformed to [0, 1] by the following expression

n′c_i = 2 · nc_i − [1 1 1] Finally, input for the autoencoder is the matrix

N3×(n+1) reshaped to Zi(n+1)×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 appro-
priately prepared, namely, we generate and rotate patches.
Then, the data are provided as input in a CVAE archi-
tecture. Although time-consuming, this process takes place
only once. After the autoencoder has generated the filtered
output Zi(n+1)×1, they are reshaped back to initial dimensions
Zi3×(n+1). The exported filtered normal vector for patch Pi is
the first column of Z, and more specifically 3×1 = Zi[1; 0].
Then each patch is rotated by the opposite angle −δni,
and the same axis an, they were rotated with in the first
place and transformed back to range [−1, 1] using n′c_i =
(n′c_i + [1 1 1]) / 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 per-
forming 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 autoen-
coder architecture, comprising in total of 1,977,740 patches.
Noisy meshes were synthetized by adding Gaussian noise
N ∼ (0, 0.1 · Lp) 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 β1 = 0.9, β2 = 0.999
and ϵ = 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], L0 minimization mesh denoising [19] and fast and ef-
efficient 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 follow-

ing 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 $\alpha$ (in degree) using disparate approaches of initialization and deep architectures.

<table>
<thead>
<tr>
<th>Model</th>
<th>$8,nn$</th>
<th>$8,nn +,pp$</th>
<th>$20,nn$</th>
<th>$20,nn +,pp$</th>
<th>$8,nn$</th>
<th>$8,nn +,pp$</th>
<th>$20,nn$</th>
<th>$20,nn +,pp$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese-lion</td>
<td>0.27752</td>
<td>8.93$^a$</td>
<td>0.28478</td>
<td>8.79$^a$</td>
<td>0.34635</td>
<td>9.10$^a$</td>
<td>0.40453</td>
<td>9.88$^a$</td>
</tr>
<tr>
<td>Rocker arm</td>
<td>0.00490</td>
<td>8.85$^a$</td>
<td>0.00530</td>
<td>7.19$^a$</td>
<td>0.00691</td>
<td>8.44$^a$</td>
<td>0.00699</td>
<td>8.49$^a$</td>
</tr>
<tr>
<td>Sculpt</td>
<td>0.01408</td>
<td>9.85$^a$</td>
<td>0.01273</td>
<td>7.74$^a$</td>
<td>0.01706</td>
<td>12.60$^a$</td>
<td>0.01699</td>
<td>11.99$^a$</td>
</tr>
<tr>
<td>Gear</td>
<td>0.13519</td>
<td>8.10$^a$</td>
<td>0.13801</td>
<td>6.89$^a$</td>
<td>0.16017</td>
<td>5.98$^a$</td>
<td>0.16651</td>
<td>6.06$^a$</td>
</tr>
<tr>
<td>Trim-star</td>
<td>0.26388</td>
<td>10.12$^a$</td>
<td>0.29083</td>
<td>8.26$^a$</td>
<td>0.30939</td>
<td>11.24$^a$</td>
<td>0.31648</td>
<td>10.80$^a$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Bilateral normal filter</th>
<th>Guided mesh normal filter</th>
<th>Fast and effective $L_0$ minimization</th>
<th>CVAE 20 pp</th>
<th>CVAE 8 pp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sculpt(3669V,1742F)</td>
<td>0.162</td>
<td>0.6465</td>
<td>0.0591</td>
<td>3.5884</td>
<td>0.0772</td>
</tr>
<tr>
<td>Trimmed star(5193V, 1084F)</td>
<td>0.1529</td>
<td>0.9643</td>
<td>0.0869</td>
<td>4.2748</td>
<td>0.0995</td>
</tr>
<tr>
<td>Rocker Arm(9413V,18826F)</td>
<td>0.3242</td>
<td>2.0561</td>
<td>0.1804</td>
<td>11.1609</td>
<td>0.1642</td>
</tr>
<tr>
<td>Chinese Lion(50K V, 100K F)</td>
<td>2.0508</td>
<td>21.6360</td>
<td>1.5792</td>
<td>110.6100</td>
<td>0.9872</td>
</tr>
<tr>
<td>Gear(250K V,500K F)</td>
<td>8.5630</td>
<td>221.1910</td>
<td>5.7120</td>
<td>252.2500</td>
<td>3.8858</td>
</tr>
</tbody>
</table>

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 an Intel(R) Core(TM) i7-4790 CPU @ 3.60Hz with 32GB of RAM.

Table I presents the Hausdorff distance and the mean $\alpha$ 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 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
Union funds (European Regional Development Fund).
Operational programme of Western Greece 2014-2020 and European
under the Action for the Strategic Development on the Research
convolutional neural networks project (MIS 5038640) implemented
826299 and the DEEP-EVIoT - Deep embedded vision using sparse
Unions Horizon 2020 H2020-SC1-DTH-03-2018 grant agreement no
computational cost. Execution time measurements presented in
fast for large and dense models compared to other approaches.
dataset size for the training process in comparison with other
noise with different patterns, (iii) allowing a relatively small
The main advantages of the proposed method include (i)
very fast data-driven denoising approach using conditional
noisy conveyor belt model. In this work, we presented a
environments presenting the denoising result of a synthetic
model part exhibits $O(1)$ complexity removing a large portion of the
Section 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 dig-
itization. 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 network 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**

[6] D. Deahl, “The royal netherlands navy is 3d scan-