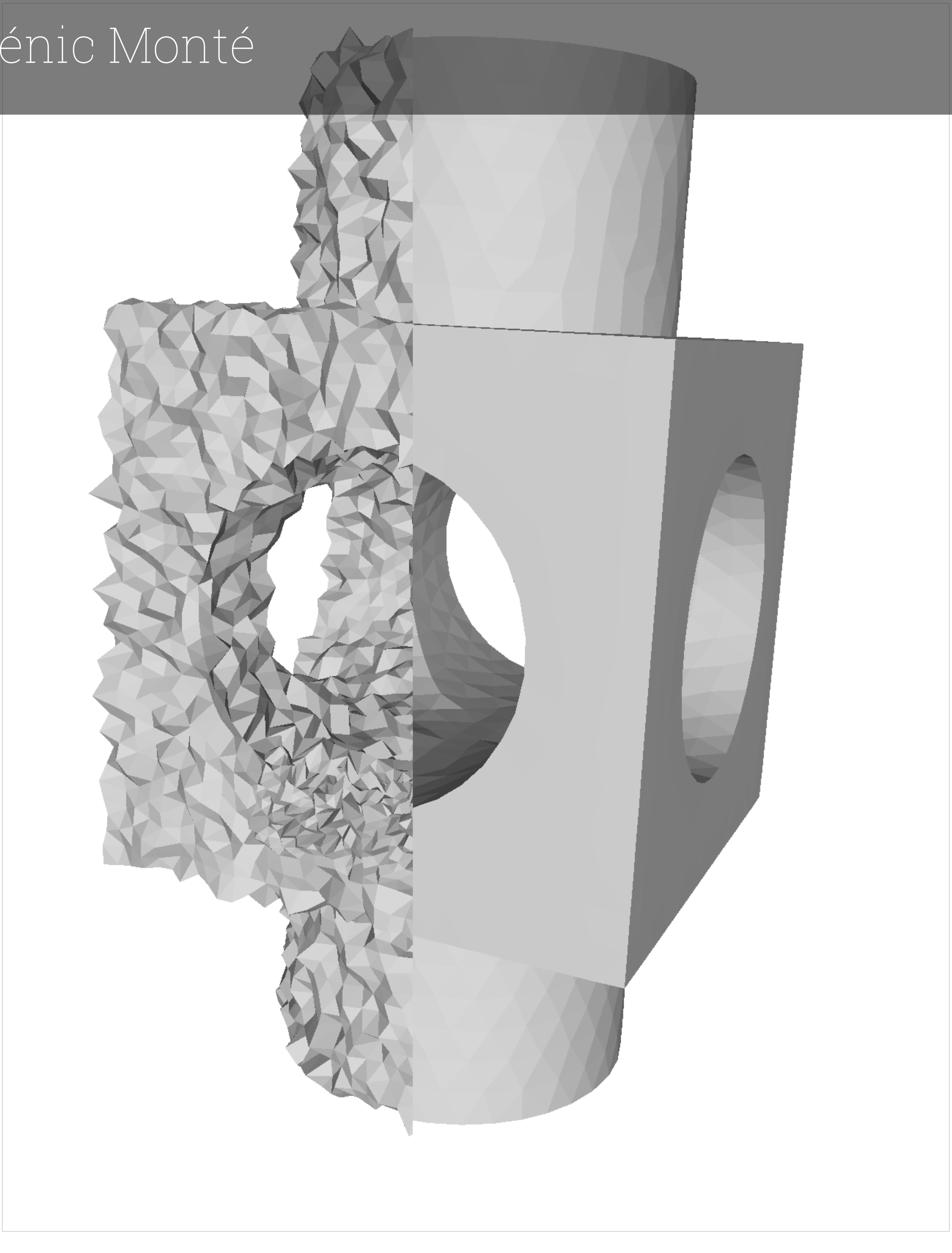
# Mesh denoising using DeltaConv

Sérénic Monté



# Mesh denoising using DeltaConv

by

Sérénic Monté

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday July 2, 2024 at 14:00.

|  |  |  |
|---|---|---|
| Student number: | 4554787 |  |
| Project duration: | November 13, 2023 – July 2, 2024 |  |
| Thesis committee: | Klaus Hildebrandt, | TU Delft, supervisor |
|  | Jackson Campolattaro, | TU Delft |
|  | Megha Khosla, | TU Delft |

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

# Abstract

Mesh data is widely used in engineering for instance for simulations, CAD engineering and visualizations. The accuracy and quality of the meshes influence the reliability and validity of these processes. Besides manual modelling, scanning is becoming increasingly more common due to the increase in devices that have scanning capabilities. Unwanted noise is often present in scanned models. The process of mesh denoising is removing the unwanted noise whilst keeping the features of the mesh. These features are often anisotropic, e.g. sharp edges and corners.

The DeltaConv convolution is an anisotropic convolution, Wiersma et al. [24] show the advantage of using the anisotropic DeltaConv convolution for anisotropic tasks over other isotropic convolutions. In this thesis it is investigated if state-of-the-art mesh denoising can benefit from using the DeltaConv convolution. This is done by integrating the DeltaConv convolution in the Dual-DMP [6] algorithm, and tuning this network.

In this thesis we found that state-of-the-art mesh denoising can benefit from using the DeltaConv convolution. Due to the expressiveness of the DeltaConv convolution, objects with sharp features are denoised better than the state-of-the-art algorithms and on smooth meshes, DeltaConv works comparable to state-of-the-art algorithms.

# Contents

# Nomenclature

## Abbreviations

| Abbreviation | Definition |
|---|---|
| AAD | Average Angular Difference, metric which evaluates the angle between normals of two meshes |
| AHD | Average Hausdorff Distance, metric which evaluates the distance between two meshes |
| CNN | convolutional neural networks |
| DDMP | Dual Deep Mesh Prior, a state-of-art mesh denoising algorithm [6] |
| Dual-DMP | Dual Deep Mesh Prior |
| DIP | Deep Image Prior [10] |
| GCN | graph convolutional neural networks or the graph convolution as described by [9] |
| GNN | graph neural networks |
| MLP | multi layer perceptron |

# 1 Introduction

Meshing is important for engineering and it involves discretizing a spacial domain into a set of cells or discrete elements. It is widely used for CAD engineering, simulations like fluid dynamics, visualization of structures and systems, etc. The accuracy and quality of the mesh has a profound impact on the validity and reliability of the results obtained from these simulations [3].

Another source of obtaining meshes is scanning the objects. 3D scanning capabilities like LiDAR are added to appliances for everyday use, for instance mobile phones, cars, etc. Scanned objects contain unwanted noise or distortions introduced by the scanner. This noise negatively influences the results and denoising is therefore an important step. Denoising is the process of removing the noise and thus enhancing the quality of the mesh while preserving the features and details of the object. To denoise with traditional methods, assumptions need to be made about the underlying model or noise distribution for most methods to work properly [19]. For instance that the noise follows a Gaussian distribution or that the points are sampled independently and regularly from the surface. However, these assumptions do not hold for most meshes and therefore the traditional methods are less usefull [3]

Mesh denoising algorithms are influenced by image filtering techniques, since the vertex position and normals of a mesh can be seen as signals [19]. Mesh denoising algorithms can be subdivided into three categories: filter-based-, optimization-based- and data driven methods. Filter based algorithms are often adapted from the image domain. Examples of adapted filters are the Bilateral filter [5] and Guided normal filter [28].

Optimization-based methods smooth meshes by optimizing a cost function. The cost function is constructed with constraints defined by the geometry, prior distribution of the noise and the input data [8]. For instance low-rank optimization, sparsity based optimization and spectral optimization [33].

The last category is the data driven methods. These methods use deep learning to learn the denoising operation on a ground truth models and noisy input. Examples of deep learning algorithms are: GPDNet [17], DDMP [6], DMRDenoise [15], multiscale facet GCN [1], and GCN-Denoiser [19].

Most meshes contain sharp edges and corners which are called anisotropic features. Preserving these features when denoising is important. Most convolutions used for denoising are isotropic like GCN [9], PointNet++ [18] and EdgeConv [23]. DeltaConv [24] is an anisotropic convolution. In the DeltaConv paper, the authors have tested to see if DeltaConv would outperform other convolutions when mimicking a Perona-Malik filter [16] for images. The test shows the advantages of the use of an anisotropic convolution for an anisotropic task. We reproduced the results of the task in figure 1.1. As can be seen in figure 1.1, DeltaConv is able to simulate the anisotropic filter, while the other convolutions tend to blur the image too much or generate artefacts [24].

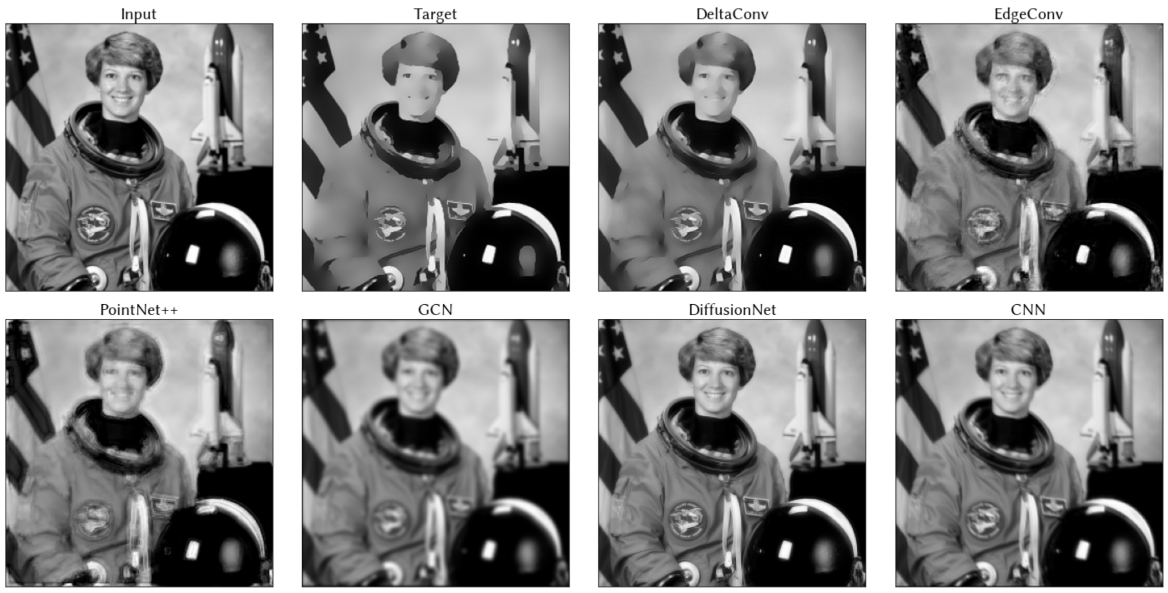


**Figure 1.1:** Reproduced results from the anisotropy test from the DeltaConv paper

If an anisotropic convolution works better for images it might also be applicable for denoising meshes. Which often have anisotropic features. In this thesis we will investigate if **state-of-the-art mesh denoising algorithms can benefit from DeltaConv convolutions**. This will be answered using the following sub-questions:

1. Can DeltaConv be integrated in a state-of-the-art mesh denoising algorithm?
2. Does the network, with DeltaConv integrated, compare to the original network in terms of quality of results?
3. How important are the different component of the DeltaConv convolution?
4. How does the network compare to other state-of-the-art convolutions and algorithms?

Integrating DeltaConv convolutions in a state-of-the-art algorithm improves the denoising performance. The main challenge when integrating the convolution is to adapt the hyperparameters of the network to work well with the more expressive DeltaConv convolution. The DeltaConv convolution consists of two parts: a vector stream and a scalar stream. For the context of mesh denoising the scalar stream is the most important component for the convolution. In the comparison with other algorithms and convolutions we found that DeltaConv is competitive with state-of-the-art for denoising meshes.

# 2 Related work

In this section we discuss the different mesh denoising algorithms. Denoising a mesh has two distinct approaches, deterministic and learning-based. Deterministic approaches can be subdivided into filter-based and optimization-based denoising algorithms. Learning-based algorithms use neural networks such as convolutional neural networks (CNNs) and Graph convolutional networks (GCNs) [22, 31].

*Filter-based mesh denoising methods* are among the first developed denoising techniques like Laplacian based smoothing. These methods were able to remove noise reasonably well, but had the disadvantage that the sharp edges also fade with the removal of the noise. To solve this problem, anisotropic diffusion filters were developed like bilateral filters [21, 5]. Improving on bilateral filter are guided normals for reliable geometry reconversion. They are suited to handle coarse and fine noise [26, 20]. A major disadvantage of this technique is that a lot of parameter tuning is required to achieve good denoising results. This makes this technique difficult to scale. [21, 5, 26, 20, 22]

*Optimization-based mesh denoising methods* tackle the problem of denoising by optimizing a cost function [3]. For instance the $L_0$ norm method [7] or total variation method [27] were adapted from signal processing on images to be used in mesh denoising. Other optimization based methods use low-rank optimization [3]. These methods assume that a mesh consists of repetitive patterns. When the similar patches are converted to vectors they are linearly correlated and this correlation can be used in the cost function [14, 12].

*learning-based denoising methods* have also found their origins in the image domain like the deterministic methods. However, the 2D learning based denoising algorithms could not be generalized to 3D without making concessions. This problem was solved with the introduction of graph convolution networks (GCN). These graph convolution networks were better suited for 3D tasks such as mesh denoising [25, 2, 4]. There are two different approaches to learning-based algorithms. The first one is an algorithm that works on the entire mesh. The second approach is where the entire mesh has been divided into patches. These patches are individually optimized under the assumption that many patches are similar. This assumption should reduce the complexity of the training objective [19].

Wang, Liu, and Tong [22] created a denoising algorithm that used handcrafted descriptors. They created *filtered facet normal descriptors* (FND) that describe the features around a facet with a series of either bilateral filters [32] or guided bilateral filters [28]. Since these filters preserve features at specific scales they are robust to scaling. The facet normal and FND features are used as input for a network with a single hidden layer. This is not enough to remove all the noise in one go, therefore the authors used a cascaded regression scheme to remove the noise in repetitive steps.

*NormalNet*, proposed by Zhao et al. [30], denoises meshes by converting a local 3D structure into a regular volumetric form and uses 3D image kernels to denoise the mesh. For every facet a patch, that consists of the 2-ring neighbourhood, is constructed and the position and direction of the facet is normalized. The normals of each facet in the patch is recomputed after normalization. Afterwards the patch is converted into a dense voxel grid and the normals of the facets are used as labels for the grid. A convolutional neural network, that works with 3D kernels is used to denoise the facet normal. These denoised facet normals are then used to update the vertex positions of the noisy mesh. This process is performed iteratively to produce a cleaned mesh.

Li et al. [13] proposed *NormalF-Net* which emulates low-rank optimization with two cascaded subnetworks to denoise meshes. The first step in the algorithm is constructing a non-local patch-group normal matrix. This is a matrix which consists of patch-vectors of patches with similar geometry. The matrix is denoised with the first subnetwork. The second subnetwork is used to prevent oversmoothing, which is common with low-rank matrices. The algorithm then outputs the denoised facet normals. The facet normals are used to update the vertex positions to be consistent with the new normals.

*DNF-Net* [11] uses a deep neural network to clean the facets normals directly. The first step for this algorithm is to generate a set of patches from the mesh. The patches are created by selecting a fixed number of the closest connected facets. The facets normals as well as the neighbour indices are directly fed into the DNF-Net architecture to produce denoised normals. The network extracts a feature map from the network input with a multiscale feature embedding unit. The feature maps are in turn denoised with two cascaded residual learning units.

Armando, Franco, and Boyer [1] propose the first mesh denoising algorithm that uses *facet graph convolutions* to denoise meshes. The network architecture builds on the U-Net architecture. In figure 2.1 an example of a U-Net is shown. Its name is derived from the shape of the network. The red arrows indicate pooling operations which coarsens the mesh. The yellow arrows indicate upsampling operations which refine the mesh. The multiscale representation enables the network to capture contexts at different scales. The algorithm works like most learning based algorithms on facets normals but it denoises a complete mesh instead of individual patches.

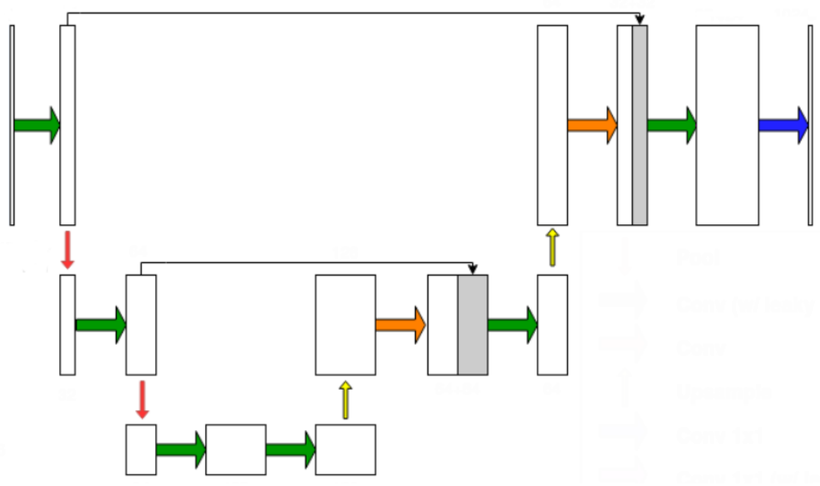


**Figure 2.1:** Example of U-Net architecture

*GCN-denoiser* [19] uses graph convolutions to denoise patches extracted from the input mesh. The patches are aligned into a common embedding to eliminate spatial transformations. The patches are converted into graphs where every node represents a facet and edges created between adjacent facets. Every graph is fed into series of GCNs to gradually remove the noise from the facet normals. The denoised mesh is recovered by updating the vertex positions to be consistent with the denoised normals.

Zhang et al. [29] use two graph networks to denoise both the normal and spatial domains in their dual graph *GeoBi-GNN* architecture. From the mesh a facet graph and a vertex graph are constructed. The vertex positions are pre-denoised by the first network. The facet graph is first enriched with the pre-denoised vertices before the facet normals are denoised by the second network. The pre-denoised vertices are then updated to align with the denoised facet normals.

*Dual-DMP* [6] moves away from the pretraining of neural networks and uses the principles from Deep Image Prior (DIP) [10]. DIP states that the structure of the network is more important than the pretraining, because usually the objects consist of mostly structural patterns. The main reason for this is that a network is able to learn structural patterns faster than random patterns. The denoising parameterization for an individual mesh can thus be learned quickly.

Dual DMP implements this idea and therefore does not train on a complete dataset but on the individual meshes it wants to denoise. The DDMP network consists of two parts, a graph convolution network (GCN) for the vertices and a GCN for the facet normals. The two networks filter noise in the vertex position and face normals independently of each other. This may introduce some inconsistencies in the output. To correct these inconsistencies iterative vertex updating could be used but is expensive in time and computing power. The authors of DDMP have therefore created a loss function based on the formulation of iterative vertex updating. This consistency error term ensures that the new vertex positions remain consistent with the facet normals. The Dual-DMP algorithm consists of 12 convolution layers and then 2 fully connected layers.

# 3 Background

## 3.1. Mesh denoising

Scans of objects are becoming increasingly easier to make and this means it can be applied in more fields, such as medical treatment, industrial modelling, reverse engineering, etc. The scans that create these meshes contain noise from various sources. The goal of mesh denoising is to remove the high frequency noise without removing the underlying features [19, 3]. Typical challenges that arise when denoising meshes is that features tend to degrade, such as blurring and overshapening. Another typical challenge is shrinkage of the volume. To evaluate if the denoising of the meshes is circumventing these problems we use two standard metrics. The first metric is the average angular distance. This metric calculates the mean angle between the facet normals of the denoised mesh and ground truth mesh. This metric is to ensure the features do not degrade. The second metric is the average one-sided Hausdorff distance. This metric penalizes shrinkage of the mesh [3].

The first metric is the average normal angular difference (AAD) (3.1)

$$\theta = \frac{\sum_{i=0}^{n} x_i}{n} \tag{3.1}$$

$$x = \arccos(n_1 \cdot n_2)$$

The second metric we use is the average one-sided Hausdorff distance (AHD) (3.2). With $h(a,b)$ the distance between a and b. $gt_{diag}$ the diagonal length of the bounding box of the ground truth mesh. All the values of the AHD in this paper are in the unit of $10^{-3}$.

$$d = 0.5 * \left( \frac{h(m1, m2)}{|gt_{diag}|} + \frac{h(m2, m1)}{|gt_{diag}|} \right) \tag{3.2}$$

## 3.2. DDMP

The graph convolutions in DDMP consist of a GCN convolution whose output gets normalized before passing through a Leaky ReLu activation layer. The output of this chain is the input for the next convolution layer. For the Positional filtering network the required inputs are the input features, the position and the set of neighbouring vertex indices per vertex. The second to last layer is a linear layer with a Leaky ReLu activation layer. The last layer is a linear layer which outputs the displacement of the vertex position. The displacement of the vertex position is added to the vertex position at the end of the forward pass. For the normal filtering network the DDMP network has the same structure, except the last layer consists of a linear layer followed by a tanh layer. The network outputs a new normal which is multiplied with the reciprocal of the normalized version of itself. The input is slightly changed since not the set neighbouring vertex indices is given but the set of neighbouring face indices is used.

The loss function defined by the authors of [6] is:

$$E = k_1 E_{pos} + k_2 E_{Lap} + k_3 E_{nrm} + k_4 E_{bnf} + k_5 E_{con} \tag{3.3}$$

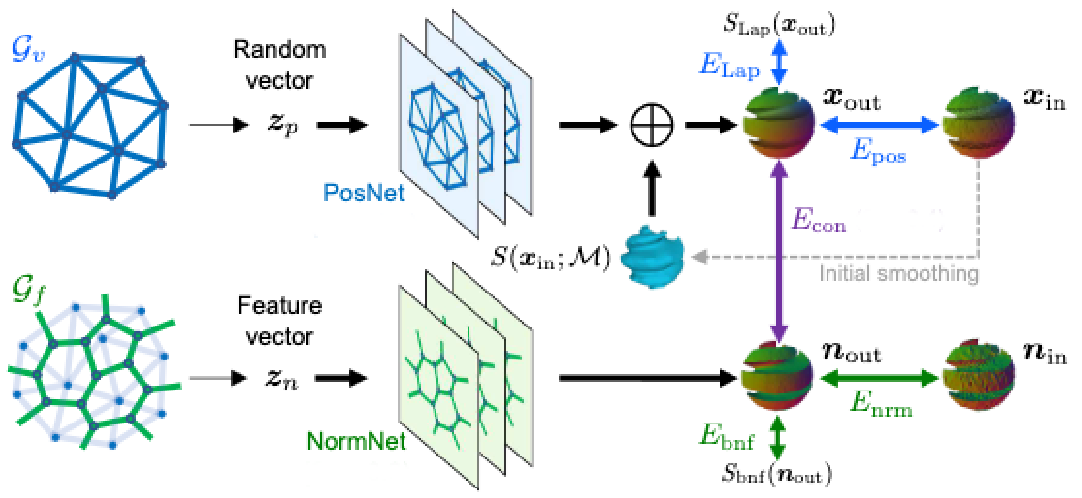


**Figure 3.1:** Dual-DMP algorithm, adapted from [6]

with $K = (k_1, k_2, k_3, k_4, k_5)$ a set of hyperparameters to give more or less weight to each individual term of the loss function.

The first error term $E_{pos}$ is the root mean squared error (RMSE) between the in- and output vertex positions, (3.4). $N_p$ is the number of vertices, $\mathbf{x}_{out}$ are the output vertex positions and $\mathbf{x}_{in}$ are the input vertex positions.

$$E_{pos}(\mathbf{x}_{out}, \mathbf{x}_{in}) = \sqrt{\frac{1}{N_p}\sum_{i=1}^{N_p}\|\mathbf{x}_{out}^i - \mathbf{x}_{in}^i\|_2^2} \tag{3.4}$$

The second error term is the Laplacian error for vertex positions, (3.5). With $V_v(i)$ the set of neighbouring vertex indices for vertex $i$

$$E_{Lap}(x_{out}) = \sqrt{\frac{1}{N_p}\sum_{i=1}^{N_p}\|\mathbf{x}_{out}^i - S_{Lap}(\mathbf{x}_{out}^i)\|_2^2} \tag{3.5}$$

$$S_{Lap}(\mathbf{x}_{out}^i) = \frac{1}{|V_v(i)|}\sum_{j \in V_v(i)} \mathbf{x}_{out}^j$$

The first two terms are only applied to the Position filtering network with $E_{pos}$ evaluates the reproducibility and $E_{Lap}$ evaluates the smoothness.

The third error term is the mean absolute error (MEA) which is used on the facet normals, (3.6). With $N_f$ the number of facets, $\mathbf{n}_{out}$ are the output facet normals and $\mathbf{n}_{in}$ are the input facet normals. They choose MEA over RMSE to let the normal filtering network be more sensitive to sharp features in the mesh.

$$E_{nrm}(\mathbf{n}_{out}, \mathbf{n}_{in}) = \frac{1}{N_f}\sum_{i=1}^{N_f}\|\mathbf{n}_{out}^i - \mathbf{n}_{in}^i\|_1 \tag{3.6}$$

The fourth error term is the MAE between the output facet normals and the normals smoothed with a bilateral filter, (3.7). With $S_{bnf}^{(t)}(\mathbf{n}_{out}^i)$ the bilateral filter applied t times to the normals.

$$E_{bnf}(\mathbf{n}_{out}) = \frac{1}{N_f}\sum_{i=1}^{N_f}\|\mathbf{n}_{out}^i - S_{bnf}^{(t)}(\mathbf{n}_{out}^i)\|_1 \tag{3.7}$$

The third and fourth error term are only applied to the normal filtering network with $E_{nrm}$ used to evaluate the reproducibility and $E_{bnf}$ for the smoothness.

The final error term is a consistency check for the vertex positions and face normals, (3.8). It is based on the formulation of iterative vertex updating. This consistency check is used since the network does not use iterative vertex updating to ensure the consistency between the updated normals and facet normals.

$$E_{con}\left(\mathbf{x}_{out}, \mathbf{n}_{out}\right) = \frac{1}{N_p} \sum_{i=1}^{N_p} \sum_{j \in F_v(i)} \mathbf{n}_{out}^{j} \cdot \left(\mathbf{c}_{out}^{j} - \mathbf{x}_{out}^{i}\right) \tag{3.8}$$

## 3.3. DeltaConv convolutions

The DeltaConv as described by Wiersma et al. [24] consists of two parallel streams, see the schematic in figure 3.2. The scalar stream which contains the additional features for every vertex, and a vector stream which contains tangent vectors which are an orthonormal pair of vectors orthogonal to the normal of the vertex. To exchange information between the streams and change the information in the streams the authors describe four different type of operations. The intra stream operations: scalar-to-scalar and vector-to-vector, and the inter stream operations: scalar-to-vector and vector-to-scalar.

The scalar-to-scalar operations are a combination of an MLP per point followed by a maximum aggregation over a k-NN region. The vector-to-vector operations are a combination of the identity and Hodge Laplacian.

The scalar-to-vector operations consist of gradient and co-gradient. The gradient represents the largest rate of change and its directions, whereas the co-gradient is the 90-degree rotation of this function. These two thus form a basis on the tangential plane. The operations to move from the vector-to-scalar stream are divergence, curl and norm. These operations analyse the vector field and indicate features where the field is flowing to, from and around. These inter stream operators form a de Rham complex. Namely, if you would consecutively apply gradient and curl the output would be zero. The same holds for consecutively applying co-gradient and divergence. Whereas gradient followed by divergence would result in the input which also holds for co-gradient followed by curl.

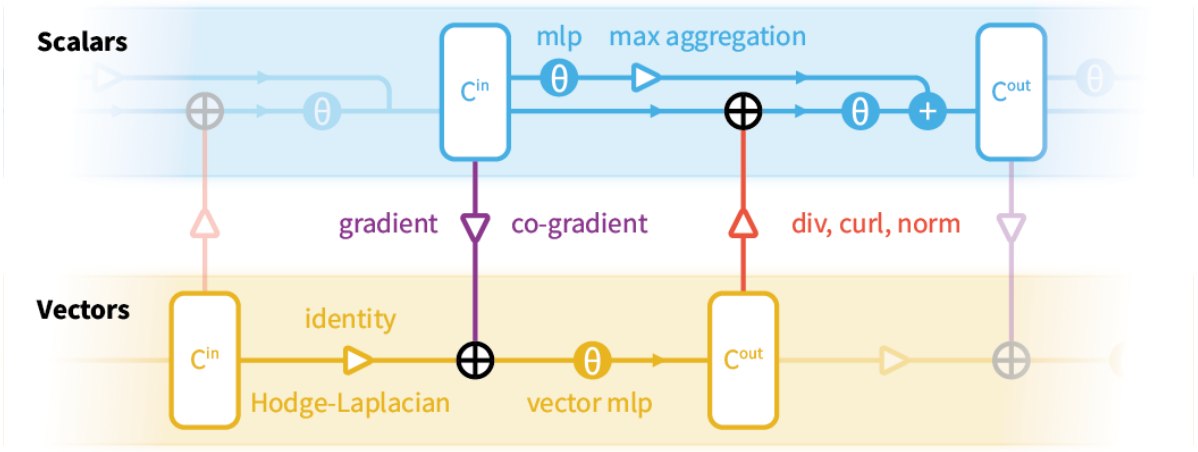


**Figure 3.2:** DeltaConv convolutions schematic, courtesy to [24]

# 4 Method

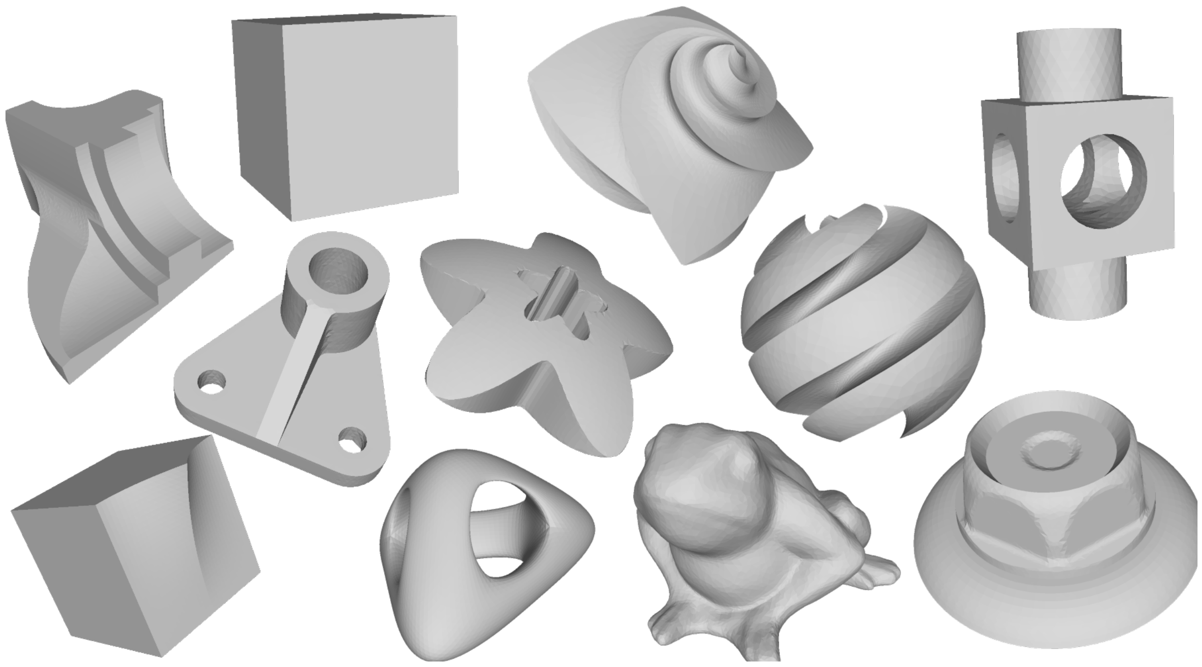


**Figure 4.1:** Ground truth meshes used in this research left to right top to bottom: fandisk, cube, octa-flower, block, part-lp, trim-star, sharp sphere, smooth feature, genus3, frog and nut

In this section the method will be discussed that is used to introduce DeltaConv convolution into the DDMP network. The GCN convolutions that are used in DDMP are isotropic. Because mesh denoising is an anisotropic problem, we want to replace the GCN convolutions with DeltaConv convolutions. Which are anisotropic.

DeltaConv and GCN are both graph convolutions, so it should be possible to replace one graph convolution with another. GCN expects the features and edge index as input parameters. DeltaConv has a different input format and expects the features, vertex positions and optionally the vertex normals. Both will output the same data format.

DeltaConv expects a fixed number of neighbours for every element in the graph. GCN on the other hand does not have any constraints on the number of neighbours selected. Therefore, the neighbour selection has to be changed from adjacent facets for the NormNet and connected vertices in the PosNet to k nearest neighbours. For NormNet the distance between facets is calculated based on the distance between the centre of the facets, and for PosNet the distance between vertices is used.

The network is a mesh denoising algorithm, scans contain noise but because scans are generally represented as point clouds we cannot use them, therefore the meshes from the synthetic dataset of Wang, Liu, and Tong [22] will be used with added artificial noise. This dataset consists mainly of CAD-models. Most CAD models are piecewise smooth and often contain sharp edges. Examples of the synthetic models used can be found in figure 4.1. The dataset is used as the ground truth meshes. The noisy meshes are created by adding white Gaussian noise with a magnitude between 0.1 and 0.4 times the average edge length. Figure 4.2 shows an example of a mesh with added noise.

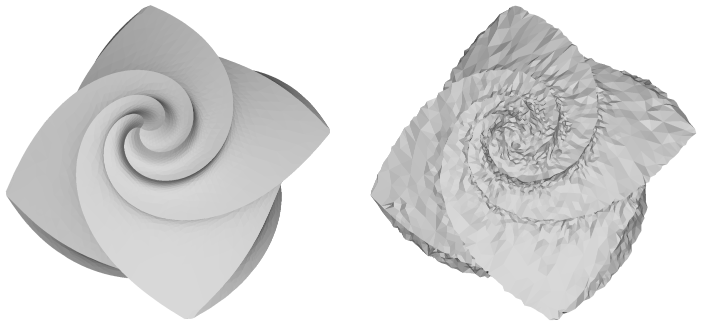


**Figure 4.2:** Left ground truth octa-flower mesh. Right the noisy octa-flower mesh with a magnitude of 0.2

The DDMP algorithm relies on an initial smoothing of the noisy mesh to calculate the displacement of the vertices. 30 iterations of Laplacian smoothing is used for the initial smoothing, figure 4.3 shows the ground truth, noisy and smoothed mesh for the sharp sphere model. The network estimates the displacement which, when added to the smoothed mesh, results in the clean mesh. The network is underfitting when the output mesh, after the training epochs, is closer to the initial smoothed mesh then the ground truth mesh. On the other hand it is overfitting if the output mesh is closer to the noisy input then the ground truth mesh.

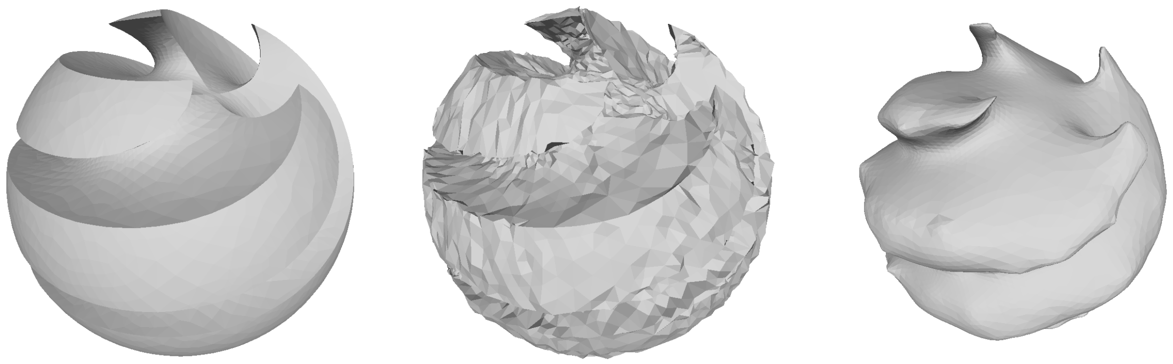


**Figure 4.3:** Left ground truth sharp sphere mesh, middle the noisy mesh and right the smoothed mesh

Apart from visual observation to check if the network is under- or overfitting. The loss function and evaluation metrics are monitored to determine if the network is under- or overfitting. To monitor the behaviour of the network every 10 epochs the output of the network is scored using the AAD and AHD metric. The evaluation scores and loss are saved and plotted. In figure 4.4 an example is shown. The Blue line is what the metrics are expected to follow when the network is fitted correctly. When the network is underfitting (orange line) the metric will not reach the minimum. The expected behaviour of an overfitting network, the network will reach a minimum before the training is over and diverge with further training.

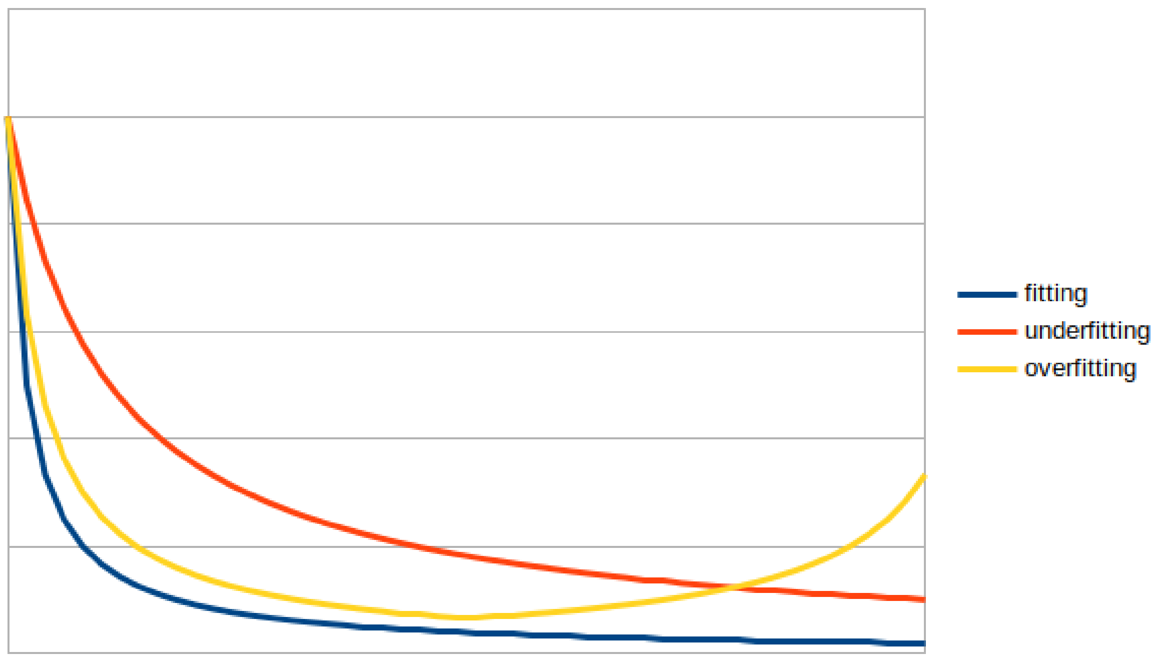


**Figure 4.4:** Expected graph of metric scores for good fitted (blue), underfitted (orange) and overfitted (yellow)

# 5 Experiments

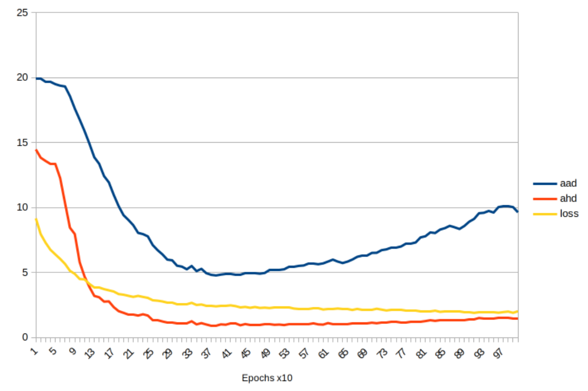
The method presented in this paper was implemented on a computer with 128 GB of RAM, an AMD Ryzen Threadripper 3970X with 32 cores and 2 NVIDIA GeForce RTX 2080 Ti with 11 GB of memory each. The POSnet and the NORMet of both Dual-DMP and the DeltaConv implementations are trained for 1000 steps with the default learning rate of $\gamma = 0.01$ and decay parameters ($\beta$1, $\beta$2) = (0.9, 0.999). To investigate the 3D mesh denoise quality of DeltaConv, we tested different forms of the algorithm. First we have the standard Dual-DMP algorithm in which only the convolution layers have been adapted to DeltaConv. Secondly, we tested a reduced network with a lower number of convolution layers. We also investigated what the best KNN value is and what the best hyperparameters for the loss function for the DeltaConv network are. In addition, we conducted an ablation study to test the effect of the different streams in DeltaConv. The last experiment we did was comparing the network with DeltaConv, GCN and EdgeConv convolutions. The quantitive results are shown in tabular format. The cells contain the AAD and AHD score of the network for a specific model. The green cells indicate the network that performs best for a model. Yellow is used if two networks perform (almost) similar. For the qualitative results the figure consists of three areas on the left side the noisy input is shown, on the right side the ground truth is shown. The middle area shows the outcome of the networks. Each row is the output of a single network type. The first image is the shaded output. The second image shows the model shaded per vertex by the angular distance between the model and the ground truth. The third image is the models shaded by the distance between the vertices of the model and the ground truth. Both the coloured variant use a gradient from blue to red via green. With blue a low value and red a high value.

## 5.1. Dual-DMP and DeltaConv

First, the quality of the denoised meshed produced by the network with the swapped convolution is determined. Table 5.1 shows that with the same settings as the DDMP network the DeltaConv network is performing worse. In figure 5.1 (left) the progression of the two metrics is plotted against the loss function. The graph shows that network is overfitting since the metrics reach their minimum before the training is over and diverge. The breakdown of the different parts of the loss function can be seen in figure 5.1 (right). The $E_{bnf}$ component is set to 0 for the first 100 epochs. It is also noticeable that the $E_{Lap}$ component increases. The overfitting could be due to a number of things, such as the hyperparameters of the loss function and the size of the network. It may also be due to the selection of the k nearest neighbours that are used to retrieve the local neighbourhood. As can be seen in figure 5.2, Dual-DMP is a lot better than the DeltaConv network without optimization. The noise is almost not reduced by the DeltaConv convolution. The zoom out shows one of the more densely triangulated areas and both algorithms have trouble with this area.

| AAD/AHD | noise | Dual-DMP | DeltaConv |
|---|---|---|---|
| Block | 33.719/4.323 | 4.568/1.038 | 15.878/2.287 |
| Cube | 20.656/3.117 | 0.785/0.344 | 0.626/0.325 |
| Fandisk | 28.421/3.231 | 2.474/0.604 | 15.735/2.088 |
| Frog | 23.268/1.753 | 4.642/0.624 | 7.798/0.833 |
| Genus3 | 22.392/2.248 | 2.597/0.59 | 5.157/0.812 |
| Nut | 22.231/1.613 | 3.265/0.466 | 4.878/0.579 |
| Octa-flower | 22.709/1.526 | 5.471/0.578 | 11.146/0.911 |
| Part-lp | 18.865/2.43 | 1.923/0.504 | 5.9/1.094 |
| Sharp-sphere | 24.455/1.983 | 5.469/0.734 | 13.336/1.25 |
| Smooth-feature | 20.581/2.913 | 1.006/0.408 | 1.532/0.508 |
| Trim-star | 29.362/3.423 | 5.25/0.94 | 16.987/2.238 |

**Table 5.1:** Quantitive comparison of AAD and AHD values between the original convolutions and DeltaConv convolution in the DDMP network.



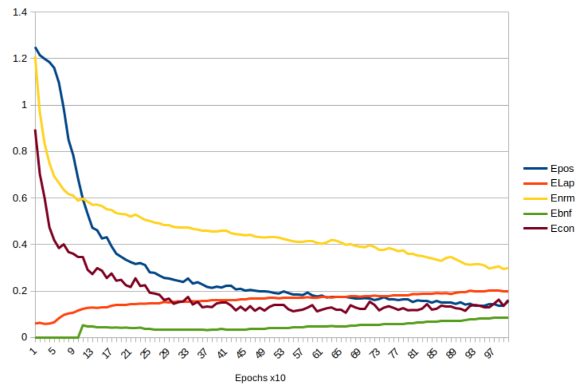


**Figure 5.1:** Left The progression of the composite loss function for DeltaConv on the original network and the metrics. Right The individual components of the loss function

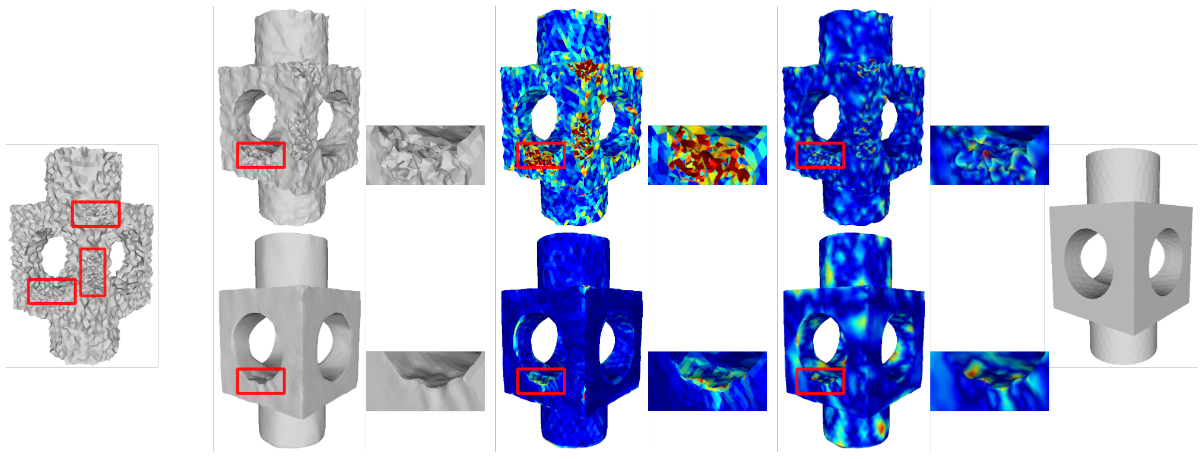


**Figure 5.2:** This is the Block model. Left the input with the more densely triangulated areas marked. Right the ground truth model. For the middle three models with zoom out the left is the normal shaded model, the middle is the model coloured by AAD value and right is the model coloured by AHD value. The top row is DeltaConv and the bottom row is DDMP.

## 5.2. K-NN

The neighbours around a vertex are selected using KNN, because this is done in an extrinsic way, it does not guarantee that all the neighbouring vertices are on the surface surrounding the vertex. To increase the chance that only points are selected that are on the surface surrounding the point, the performance was tested with a reduced number of neighbours. The network was tested with 20 (original), 10 and 8 neighbours. What is striking about the progression is that the network with 8 neighbours reaches the lowest minimum with $AAD = 3.63$ and $AHD = 0.705$ and suffers the least from overfitting, see right side of figure 5.3. With 10 neighbours (middle of figure 5.3 right) a lower minimum is achieved than with 20 neighbours (left side of figure 5.3 left) but the network suffers more from overfitting. The same can be seen in figure 5.4. The network performs also better with 8 neighbours than with 10 and 20 neighbours. The zoom out shows the transition from the flat bottom part to the cylindrical extrusion. The 8 neighbour network is able to recover the flat area better than the other two networks. The recovery on the cylindrical extrusion is similar for the 8 neighbour and 20 neighbour network.

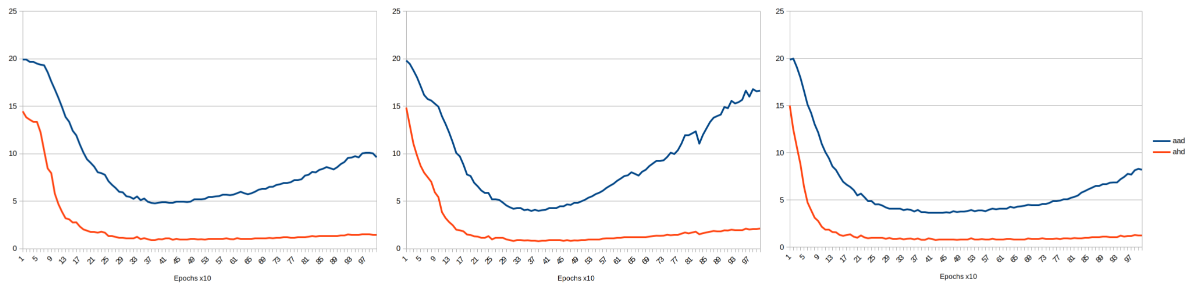


**Figure 5.3:** Left evaluation metrics for DeltaConv on the original network with 20 neighbours. Middle with 10 neighbours. Right with 8 neighbours.

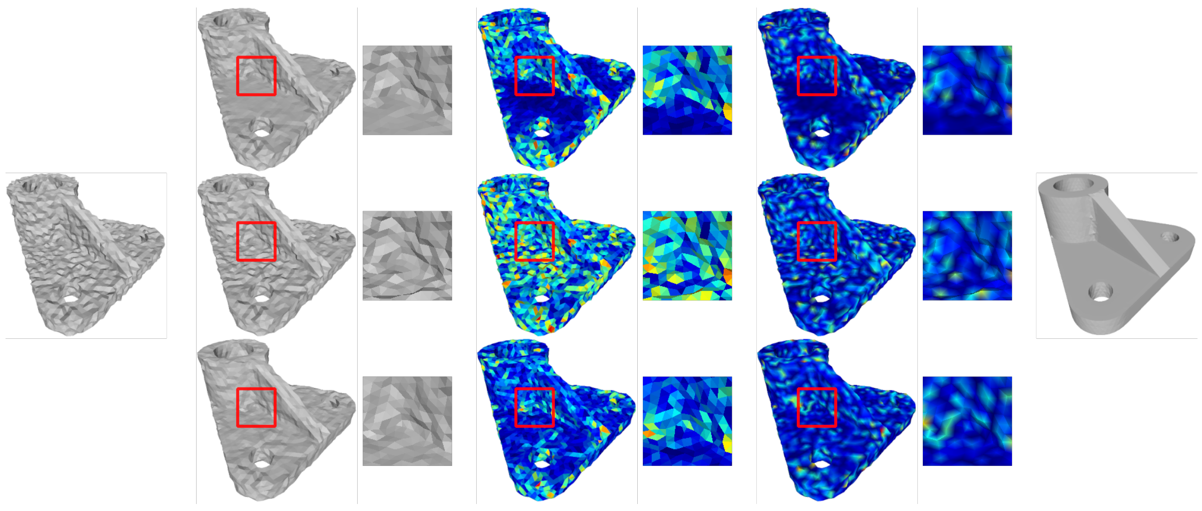


**Figure 5.4:** This is the part-lp model. Left the input with the more densely triangulated areas marked. Right the ground truth model. For the middle three models with zoom out the left is the normal shaded model, the middle is the model coloured by AAD value and right is the model coloured by AHD value. The top row is the network with 8 neighbours, middle row with 10 and bottom row with 20.

## 5.3. Smaller DeltaConv

Because DeltaConv seems to overfit, as shown in Figure 5.1 (left), the number of convolutional layers was reduced. The network was tested with 12 (original), 7 and 5 layers. For testing with the different number of convolutions, the network reaches a minimum with 7 layers $AAD = 1.731$ and $AHD = 0.48$ see figure 5.5 (middle). The network with 5 layers is too small with a minimum of $AAD = 2.634$ and $AHD = 0.598$, see figure 5.5 (right). The original 12-layer network has a minimum at $AAD = 2.032$ and $AHD = 0.567$, see figure 5.5 (left). Therefore, the network with 7 DeltaConvconvolution layers was chosen. As can be seen in figure 5.6, the output of the network with 7 layers appears to be best, however it has some difficulty with the sharp edge between the cylindrical extrusion and the bridging part. The network with 12 layers has trouble with the same area. The 5 layer network does not have a region that is worse than the other areas but has more noise across the whole model.

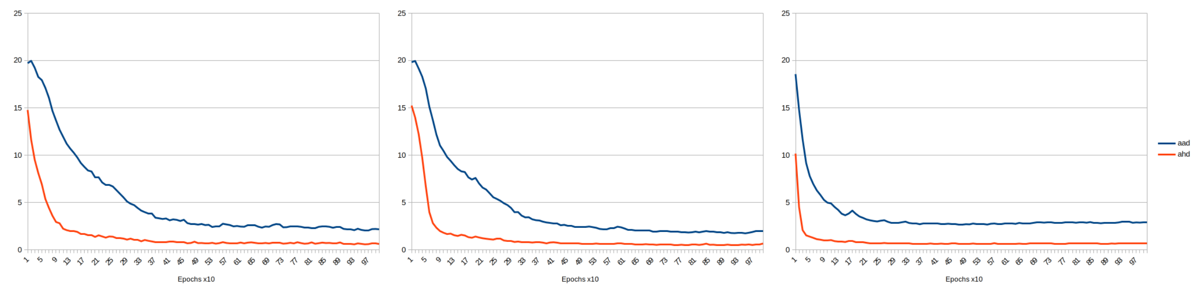


**Figure 5.5:** Left evaluation metrics with 12 layers. Middle evaluation metrics with 7 layers. Right Evaluation metrics with 5 layers.

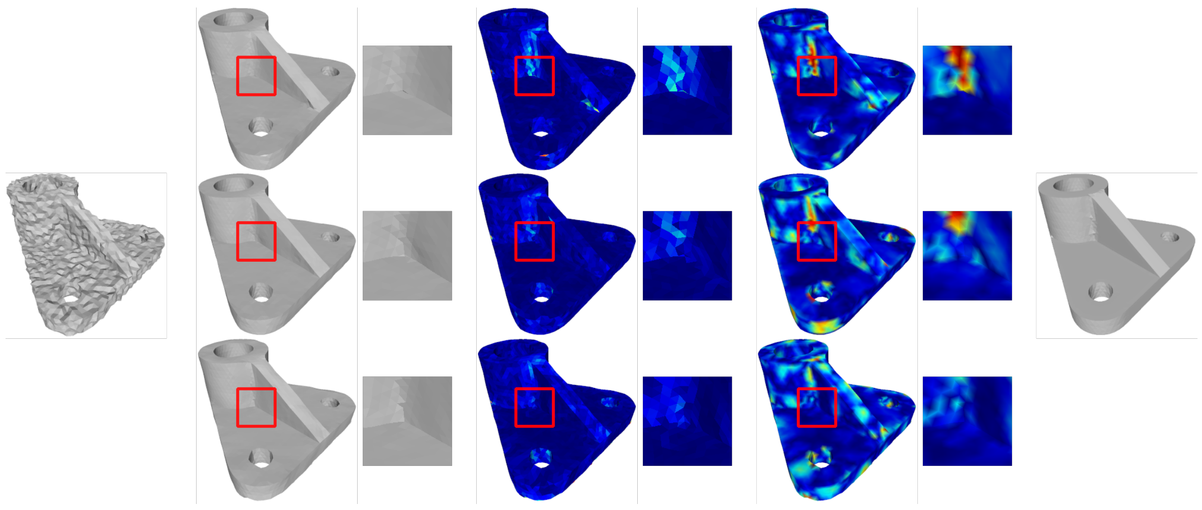


**Figure 5.6:** This is the part-lp model. Left the input. Right the ground truth model. For the middle three models with zoom out the left is the normal shaded model, the middle is the model coloured by AAD value and right is the model coloured by AHD value. The top row is the network with 12 layers, middle row with 7 and bottom row with 5.

## 5.4. Loss function

The hyperparameters of the loss function were also tuned. The RayTune framework was used to optimize the hyperparameters. The loss function has 5 hyperparameters that can be changed for every parameter integer values between 0 and 5, both inclusive, were tested. Tthe Optuna search algorithms was used to guide the search space. The original hyperparameters chosen by Dual-DMP were K = (3, 0, 3, 4, 2), the optimized parameters for models with many sharp features are K = (5, 0, 1, 5, 5) and for models with more curves the parameters are K = (1, 5, 1, 1, 0).

The major effect of the optimization can be clearly seen in figure 5.7. The top row shows the results of the network without optimization. The bottom row shows the result for of the optimized network. The zoom out shows the area the optimized model has the most trouble with.

| AAD/AHD | noise | DeltaConv | DeltaConv optimized |
|---|---|---|---|
| Block | 33.719/4.323 | 15.878/2.287 | 3.594/0.918 |
| Cube | 20.656/3.117 | 0.626/0.325 | 0.467/0.256 |
| Fandisk | 28.421/3.231 | 15.735/2.088 | 2.16/0.575 |
| Frog | 23.268/1.753 | 7.798/0.833 | 4.036/0.675 |
| Genus3 | 22.392/2.248 | 5.157/0.812 | 2.043/0.501 |
| Nut | 22.231/1.613 | 4.878/0.579 | 3.268/0.459 |
| Octa-flower | 22.709/1.526 | 11.146/0.911 | 5.026/0.558 |
| Part-lp | 18.865/2.43 | 5.9/1.094 | 1.898/0.508 |
| Sharp-sphere | 24.455/1.983 | 13.336/1.25 | 6.254/0.765 |
| Smooth-feature | 20.581/2.913 | 1.532/0.508 | 0.868/0.362 |
| Trim-star | 29.362/3.423 | 16.987/2.238 | 6.253/0.962 |

**Table 5.2:** Comparison of AAD and AHD values between DeltaConv in the original network and DeltaConv in the optimized network.

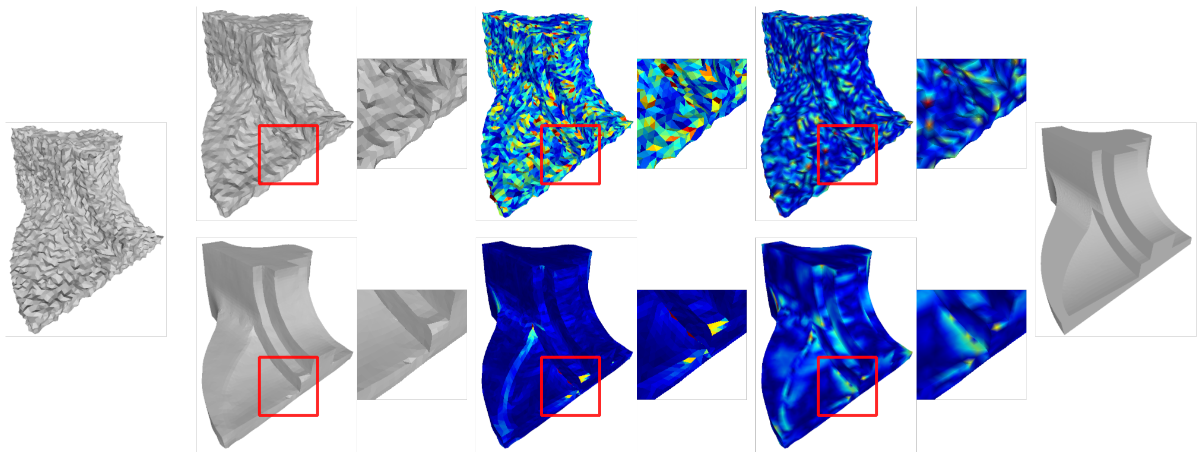


**Figure 5.7:** This is the fandisk model. Left the input. Right the ground truth model. For the middle three models with zoom out the left is the normal shaded model, the middle is the model coloured by AAD value and right is the model coloured by AHD value. The top row is the unoptimized network. Bottom row is the optimized network

## 5.5. DDMP and optimized DeltaConv

As can be seen in the table 5.3, in most cases the DeltaConv implementation is better than Dual-DMP or DeltaConv is almost as good as Dual-DMP. In the case of the Nut model DeltaConv and Dual-DMP are equivalent, this is indicated with yellow. We note that DeltaConv is better at denoising sharp features than at denoising smooth features. This is clearly visible in, for example, cube and frog. Cube is an object with only sharp corners and DeltaConv does this better than Dual-DMP. Whilst Frog is an object with only curves and here Dual-DMP is better than DeltaConv. Another work with sharp features is Octa-flower, see figure 5.8. It can be seen that the DeltaConv implementation better preserves the sharp features of Octa-flower. The centre of the flower is better recovered by the DeltaConv network than the DDMP network as can be seen in the model coloured by AAD value. The zoom out shows the outer edge of the flower in the shaded image it can be seen that the DeltaConv network recovers a sharper edge and the edge is more wavy for the DDMP network. Figure 5.9 shows the performance of both networks on the Fandisk model. The DDMP network scores similar to the DeltaConv network, the most difference can be seen when the model is coloured by AHD value. Both the Fandisk and Octa-flower model consist of mostly sharp edges, the nut model consists of combination of both sharp edges and smooth curves, see figure 5.11. In the zoom out it can be seen that DeltaConv models the sharp feature better than Dual-DMP and that Dual-DMP models the smooth edges and surfaces better.

| AAD/AHD | noise | Dual-DMP | DeltaConv |
|---|---|---|---|
| Block | 33.719/4.323 | 4.568/1.038 | 3.594/0.918 |
| Cube | 20.656/3.117 | 0.785/0.344 | 0.467/0.256 |
| Fandisk | 28.421/3.231 | 2.474/0.604 | 2.16/0.575 |
| Frog | 23.268/1.753 | 4.642/0.624 | 4.036/0.675 |
| Genus3 | 22.392/2.248 | 2.597/0.59 | 2.043/0.501 |
| Nut | 22.231/1.613 | 3.265/0.466 | 3.268/0.459 |
| Octa-flower | 22.709/1.526 | 5.471/0.578 | 5.026/0.558 |
| Part-lp | 18.865/2.43 | 1.923/0.504 | 1.898/0.508 |
| Sharp-sphere | 24.455/1.983 | 5.469/0.734 | 6.254/0.765 |
| Smooth-feature | 20.581/2.913 | 1.006/0.408 | 0.868/0.362 |
| Trim-star | 29.362/3.423 | 5.25/0.94 | 6.253/0.962 |
| Ccylinder | 20.031/2.994 | 2.278/0.663 | 2.216/0.655 |
| Coverrear-lp | 19.098/2.038 | 1.664/0.38 | 2.041/0.425 |
| Cylinder | 19.197/5.146 | 2.6/1.278 | 2.141/1.014 |
| Icosahedron | 18.896/1.749 | 1.732/0.289 | 1.159/0.274 |
| Rocker-arm | 24.693/1.612 | 6.676/0.66 | 6.939/0.817 |
| Sculpt | 18.873/2.986 | 2.904/1.004 | 3.055/0.952 |

**Table 5.3:** Quantitive comparison of AAD and AHD values between the DDMP network optimized DeltaConv network

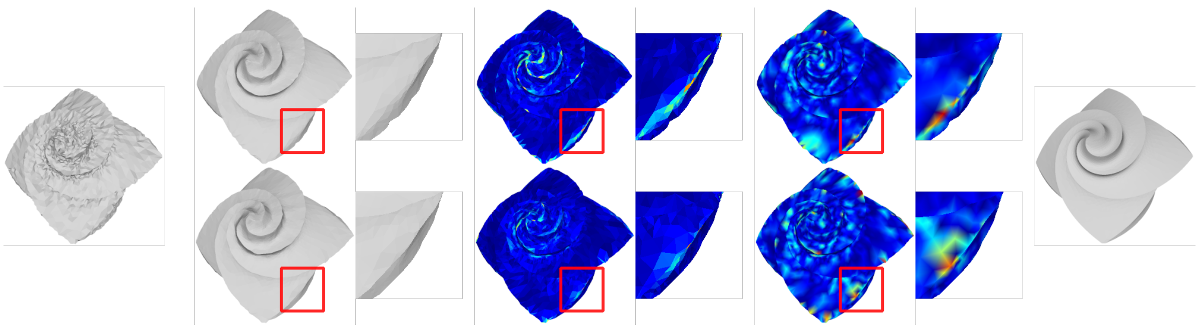


**Figure 5.8:** This is the octa-flower model. Left the input. Right the ground truth model. For the middle three models with zoom out the left is the normal shaded model, the middle is the model coloured by AAD value and right is the model coloured by AHD value. The top row is denoised by the DDMP network. Bottom row is denoised by the optimized network

The averaged metrics, figure 5.10 (AAD left and AHD right), for the DDMP network, DeltaConv network and the scalar stream only network. We notice that the three networks follow a similar curve and

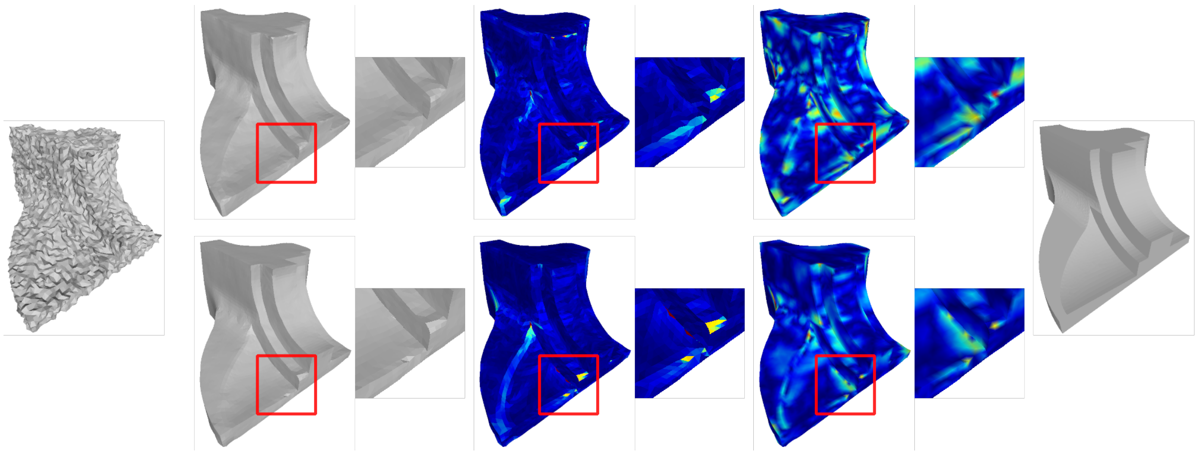


**Figure 5.9:** This is the fandisk model. Left the input. Right the ground truth model. For the middle three models with zoom out the left is the normal shaded model, the middle is the model coloured by AAD value and right is the model coloured by AHD value. The top row is denoised by the DDMP network. Bottom row is denoised by the optimized network

converge around the same epoch.

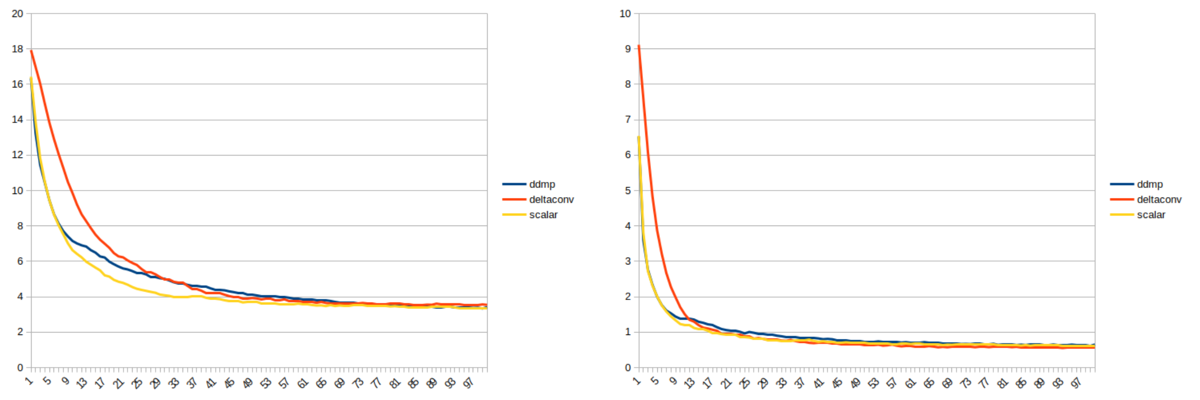


**Figure 5.10:** The average AAD (left) and AHD (right) value for the DDMP, Optimized DeltaConv and Scalar stream only network

## 5.6. Ablation study

An ablation study was performed to gain an insight into the importance of the components of the DeltaConv convolution. The first test shows the results of either the vector or scalar stream of the convolution being disabled. In table 5.4 the performance of the full network in relation to the scalar only and vector only stream are shown. The scalar stream performs better than the vector stream and is close to the performance of the full network.

| AAD/AHD | DeltaConv | scalar | vector |
|---|---|---|---|
| Block | 3.594/0.918 | 3.633/0.974 | 11.393/2.58 |
| Cube | 0.467/0.256 | 0.33/0.216 | 1.682/0.643 |
| Fandisk | 2.16/0.575 | 2.139/0.537 | 8.906/1.701 |
| Frog | 4.036/0.675 | 4.784/0.656 | 8.703/1.634 |
| Genus3 | 2.043/0.501 | 3.238/0.749 | 7.326/1.303 |
| Nut | 3.268/0.459 | 3.514/0.479 | 9.171/1.187 |
| Octa-flower | 5.026/0.558 | 5.231/0.585 | 23.34/3.118 |
| Part-lp | 1.898/0.508 | 1.896/0.534 | 4.526/1.116 |
| Sharp-sphere | 6.254/0.765 | 5.724/0.7 | 24.876/3.694 |
| Smooth-feature | 0.868/0.362 | 0.533/0.3 | 2.567/0.736 |
| Trim-star | 6.253/0.962 | 5.806/0.957 | 14.179/2.92 |

**Table 5.4:** Ablation study to inspect the two streams of DeltaConv on the optimized network

The scalar stream uses fewer features than the vector stream. To check, if the number of features influence the scalar stream, a comparison is made with the scalar stream with half the number of features and with double the number of features, see table 5.5 for the quantitive results. There is no relation between the number of features and the performance of the network.

| AAD/AHD | scalar | fewer features | more features |
|---|---|---|---|
| Block | 3.633/0.974 | 3.408/0.887 | 4.197/1.12 |
| Cube | 0.33/0.216 | 0.577/0.317 | 0.401/0.239 |
| Fandisk | 2.139/0.537 | 2.902/0.623 | 2.165/0.554 |
| Frog | 4.784/0.656 | 4.642/0.627 | 4.818/0.616 |
| Genus3 | 3.238/0.749 | 3.276/0.611 | 4.009/0.841 |
| Nut | 3.514/0.479 | 3.766/0.52 | 3.48/0.472 |
| Octa-flower | 5.231/0.585 | 5.132/0.554 | 5.132/0.554 |
| Part-lp | 1.896/0.534 | 2.109/0.583 | 1.94/0.556 |
| Sharp-sphere | 5.724/0.7 | 6.772/0.87 | 6.228/0.736 |
| Smooth-feature | 0.533/0.3 | 1.203/0.482 | 0.599/0.294 |
| Trim-star | 5.806/0.957 | 6.137/0.994 | 5.65/0.913 |

**Table 5.5:** Comparison of the AAD and AHD values for the scalar stream, scalar stream with half the number of features and scalar stream with double the number of features

The runtime that is measured, is the time it takes the algorithm to complete the training phase of a 1000 epochs. We compare DDMP, the optimized DeltaConv network with only the scalar stream, the complete network (dynamic) and the network if we precompute the neighbours, gradient, divergence and no longer update them during the fitting of the network (static). As can be seen in table 5.6, there is a marginal difference between dynamic and static, the performance is also comparable. The scalar stream is on average 6.7% slower than the DDMP network. The dynamic and static networks are on average 5.3 times slower than DDMP.

| time in seconds | DDMP | scalar | DeltaConv dynamic | DeltaConv static |
|---|---|---|---|---|
| Block | 136 | 136 | 750 | 747 |
| Cube | 99 | 102 | 504 | 492 |
| Fandisk | 98 | 107 | 583 | 545 |
| Frog | 142 | 162 | 853 | 796 |
| Genus3 | 106 | 114 | 566 | 550 |
| Nut | 108 | 129 | 620 | 591 |
| Octa-flower | 115 | 128 | 678 | 637 |
| Part-lp | 73 | 72 | 329 | 310 |
| Sharp-sphere | 145 | 165 | 850 | 808 |
| Smooth-feature | 100 | 107 | 504 | 509 |
| Trim-star | 102 | 92 | 443 | 400 |
| Average performance | 100% | 106.7% | 539.0% | 515.1% |

**Table 5.6:** Runtime in seconds comparison of DDMP, scalar stream, DeltaConv with recalculation of the operators and DeltaConv without recalculation of the operators.

## 5.7. EdgeConv comparison

We performed another experiment to verify if the same results could be reached with another graph convolution then DeltaConv. For this experiment we used EdgeConv. As can be seen in table 5.7, the performance of EdgeConv without and with optimization are lower than those of DeltaConv. The DeltaConv convolution is also outperforming the GCN convolution that is used in the DDMP algorithm. Figure 5.11 shows the performance of the four networks on the Nut model. The optimized EdgeConv model is able to recover the model a lot better than the unoptimized EdgeConv model but is still less than the other models.

| AAD/AHD | noise | Dual-DMP | DeltaConv$^2$ | EdgeConv$*$ | EdgeConv$^2$ |
|---|---|---|---|---|---|
| Block | 33.719/4.323 | 4.568/1.038 | 3.594/0.918 | 32.413/4.162 | 4.301/1.006 |
| Cube | 20.656/3.117 | 0.785/0.344 | 0.467/0.256 | 4.566/0.898 | 1.438/0.449 |
| Fandisk | 28.421/3.231 | 2.474/0.604 | 2.16/0.575 | 26.885/3.073 | 2.449/0.546 |
| Frog | 23.268/1.753 | 4.642/0.624 | 4.036/0.675 | 18.028/1.43 | 5.431/0.659 |
| Genus3 | 22.392/2.248 | 2.597/0.59 | 2.043/0.501 | 13.666/1.468 | 4.07/0.652 |
| Nut | 22.231/1.613 | 3.265/0.466 | 3.268/0.459 | 21.548/1.574 | 4.122/0.505 |
| Octa-flower | 22.709/1.526 | 5.471/0.578 | 5.026/0.558 | 20.826/1.416 | 5.283/0.571 |
| Part-lp | 18.865/2.43 | 1.923/0.504 | 1.898/0.508 | 8.713/1.363 | 2.496/0.582 |
| Sharp-sphere | 24.455/1.983 | 5.469/0.734 | 6.254/0.765 | 22.318/1.831 | 7.139/0.801 |
| Smooth-feature | 20.581/2.913 | 1.006/0.408 | 0.868/0.362 | 4.844/0.869 | 1.589/0.393 |
| Trim-star | 29.362/3.423 | 5.25/0.94 | 6.253/0.962 | 27.858/3.279 | 8.151/1.166 |

**Table 5.7:** Comparison of AAD and AHD values between DDMP, DeltaConv and EdgeConv in the original network and DeltaConv and EdgeConv in the optimized network. The * stands for the original implementation without tuning and the 2 stands for the network optimized with tuning

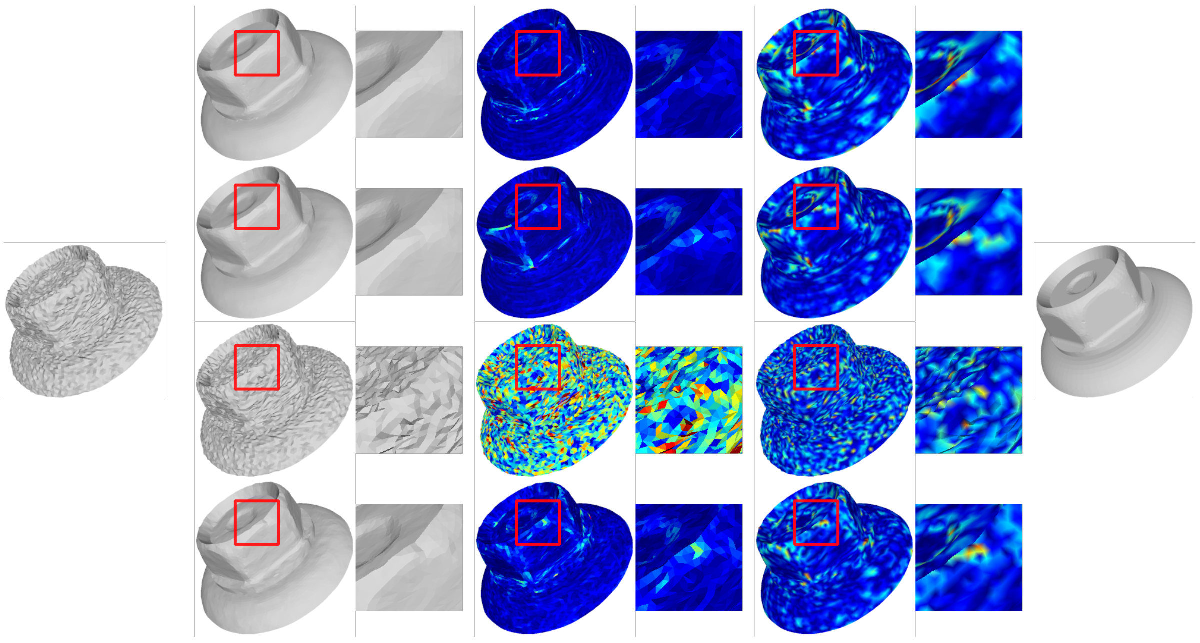


**Figure 5.11:** This is the nut model. Left the input. Right the ground truth model. For the middle three models with zoom out the left is the normal shaded model, the middle is the model coloured by AAD value and right is the model coloured by AHD value. The top row is DDMP, second row is DeltaConv, third row is EdgeConv in the DDMP network and bottom row is EdgeConv in the optimized network.

## 5.8. Algorithm comparison

The results from table 5.8 are adapted from the DDMP paper [6]. For the results of the DDMP algorithm the values were used that were found in this thesis. Both the DDMP and our network perform better than the other algorithms.

| Model | noise | BNF | GNF | DNF-Net | GCN-D | DDMP | DeltaConv |
|---|---|---|---|---|---|---|---|
| Block | 33.72/4.32 | 7.06/1.26 | 4.11/0.99 | 4.13/0.99 | 4.19/0.98 | 4.57/1.04 | 3.59/0.92 |
| Fandisk | 28.42/3.23 | 3.49/0.76 | 2.93/0.68 | 3.33/0.69 | 3.28/0.70 | 2.47/0.60 | 2.16/0.58 |
| Nut | 22.23/1.61 | 5.70/0.78 | 4.05/0.55 | 4.21/0.52 | 3.64/0.50 | 3.27/0.47 | 3.27/0.46 |
| Part-lp | 18.87/2.43 | 2.07/0.66 | 2.59/0.69 | 2.84/0.67 | 2.40/0.62 | 1.92/0.50 | 1.90/0.51 |
| Trim-star | 29.36/3.42 | 6.57/1.16 | 6.95/1.11 | 5.55/0.93 | 4.86/0.89 | 5.25/0.94 | 6.25/0.96 |

**Table 5.8:** comparison with, Bilateral Normal filtering (BNF) [32], Guided normal filtering (GNF) [28], DNF-Net [11], GCN-denoiser (GCN-D) [19], DDMP [6] and our network (DeltaConv

## 5.9. Real scan

A final experiment to verify if the network will perform as well with scanned objects as it did with synthetic models with artificial nose. The input and denoised result are shown in figure 5.12. In the figure is visible that the surface noise is removed and the shape of the scanned object does not degrade. The surface of the scan is not as smooth as for CAD models and the algorithm keeps these imperfections well.

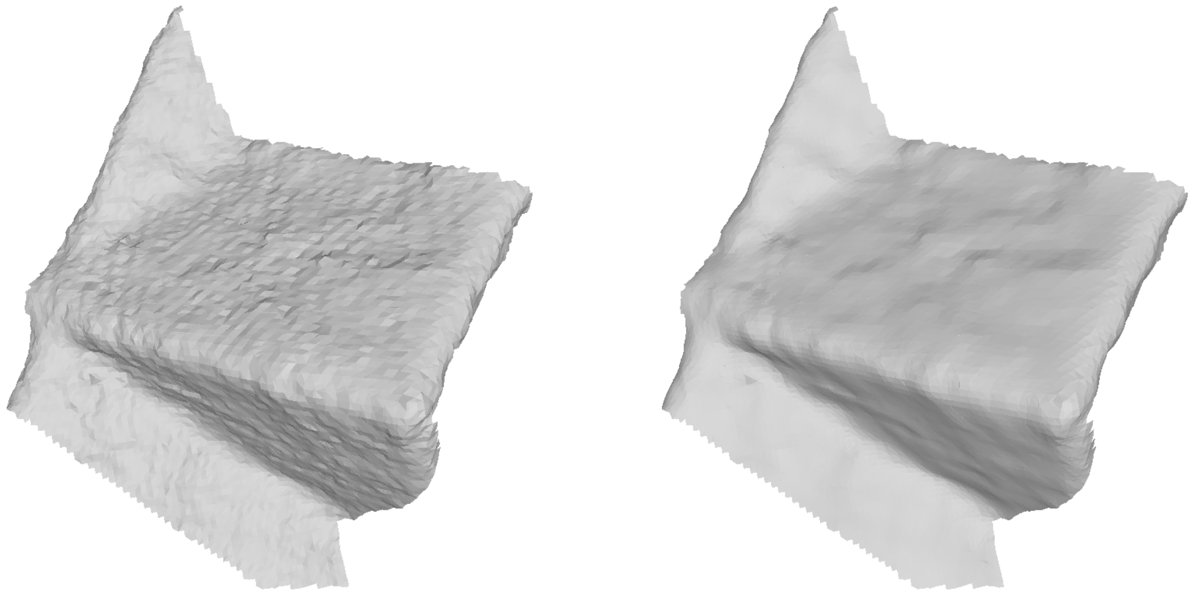


**Figure 5.12:** The input scan (left) and the denoised result (right) for the pyramid scan

# 6 Discussion

In this section we discuss the results we obtained from running our experiments, at the same time we examine the results in the light of our research question: can state-of-the-art mesh denoising benefit from using DeltaConv. The method should preserve features while removing noise.

The first subquestion: is it possible to replace the graph convolution in a state-of-the-art graph convolutional mesh denoising algorithm with DeltaConv. After replacing the convolutions of DDMP with DeltaConv the results were considerably less than the results of DDMP as can be seen in figure 5.2. If we look at the left plot of figure 5.1 we can see that the network is overfitting, as we would expect from a more expressive convolution. The loss function is decreasing as expected, but the AAD metric has a minimum before the network is fully trained. This is typical behaviour when the network is overfitting.

To solve the overfitting it is needed to change the shape of the architecture as well as the hyperparameters this was done for smooth meshes and meshes with a lot of sharp edges. For the tuning of the shape of the architecture, the number of layers tested were 12, 7 and 5, see figure 5.6. For the tuning of the hyperparameters the number of neighbours DeltaConv uses were with 20, 10 and 8, see figure 5.4. The loss function was optimized with RayTune. The overfitting was solved with for the following settings: 7 convolutional layers, 8 neighbours and for the loss function for smooth meshes K = (1, 5, 1, 1, 0) and for meshes with sharp edges K = (5, 0, 1, 5, 5). Table 5.2 shows the results before and after optimization.

The second subquestion: can the DDMP network with DeltaConv integrated compare to the original DDMP network (with GCN integrated in it). The expectation was that DeltaConv would produce better results faster, since it uses more information and is anisotropic compared to the GCN convolution that is originally used in the DDMP network. After optimization the DeltaConv networks achieved results that are on par or better than DDMP, but the convergence was at the same rate for both. The computing time per epoch for DeltaConv took about 5 times as much time. In Figure 5.10 the average AAD (left) and AHD (right) values are plotted for the DDMP network, the optimized DeltaConv network and the DeltaConv network with only the scalar stream. The plots show that the three networks converge at a similar pace. The speed of the algorithm was also tested if it made a difference to calculate the neighbours, divergence and gradient on initialization instead of in every epoch. This had no significant effect on neither the timing nor the performance, see table 5.6.

The next subquestion: how important are the different components of the DeltaConv convolutions. The first thing that was tested were the individual components of the convolutions, i.e. the scalar and vector stream of the DeltaConv algorithm. The results can be seen in table 5.4. This table shows that the vector stream performs much worse than the full DeltaConv algorithm, while the scalar stream is similar to the full DeltaConv algorithm. The scalar stream uses less features than the vector stream. To test if the number of features influence the results, the scalar stream was tested with double and half the amount of features. Table 5.5 shows that scalar stream is robust to the number of features since the performance of the three variants was similar.

The last subquestion: how DeltaConv compares to other state-of-the-art denoising algorithms. The DeltaConv is compared to GCN, but also to EdgeConv. We compared to the EdgeConv convolution both in the original DDMP network as in the optimized network. We also optimized the hyperparameters of the loss function for EdgeConv, but no additional changes were necessary. In the table 5.7 can be seen that EdgeConv does not perform as well as the other two selected convolutions. The overall performance between GCN and DeltaConv is similar. Were DeltaConv performs better on meshes with many sharp features and were on smoother meshes GCN has the edge over DeltaConv. The results of the DeltaConv network are also compared to the results found by Hattori et al. [6]. The DeltaConv network performs similar to DDMP and both outperform the other algorithms however the test is only on a select number of meshes (5) all of which are models with many sharp features.

# 7 Conclusion

The research question posed at the beginning of this thesis is if"*state-of-the-art mesh denoising algorithms can benefit from DeltaConv convolutions*". We have shown that state-of-the-art graph convolutional mesh denoising algorithms can benefit from using the DeltaConv convolution. The research question was divided into four subquestions.

The first subquestion is: can DeltaConv be integrated in a state-of-the-art mesh denoising algorithm? It is possible in DDMP to change GCN to DeltaConv. Because the expressiveness of DeltaConv is greater than that of GCN the hyperparameters as well as the shape of the network had to be adjusted to get meaningful results.

The next subquestion is: does DeltaConv compare to DDMP in terms of quality of results? After changing the shape of the network and optimizing the hyperparameters the DeltaConv network performs as well as or better than DDMP. The convergence rate is identical, but the computing time per epoch is about 5 times that of DDMP. The DeltaConv network is better at preserving the sharp features of the mesh, whereas DDMP performs better on smooth curves. This can best be seen with a visual inspection.

The third subquestion is: How important are the different component of the DeltaConv convolution? In the ablation study we tested the performance of the vector stream and scalar stream individually. We noticed that the scalar stream performs almost identical to the full DeltaConv convolution. In the visual inspection can be seen that the scalar stream generates sharper features then vector stream. Especially in the areas where there is a transition from a smooth curve to a sharp edge the scalar stream tends to oversharpens these features.

The final subquestion is: How does the network compare to other state-of-the-art convolutions and algorithms? DeltaConv results are compared to the results of the 5 overlapping meshes published in the paper by Hattori et al. [6]. The quantitive results show that the DeltaConv network outperforms the other algorithms and is similar to DDMP. The DeltaConv convolution was also tested against the EdgeConv and GCN convolutions. DeltaConv and GCN perform similar, with the GCN better on smooth meshes and DeltaConv on meshes with sharp features. EdgeConv does not perform as well as the two other convolutions.

In this thesis we investigate if state-of the-art mesh denoising can benefit from DeltaConv convolutions. We have shown that integrating DeltaConv in an existing state-of-the-art architecture is possible. To get optimal results the network size needed to be reduced and the hyperparameters need to be optimized. The optimized DeltaConv network performs as well as or better than other state-of-the-art mesh denoising algorithms. Due to the expressiveness of DeltaConv the performance on denoising meshes with sharper features works best. For smooth meshes DeltaConv performs comparable to the other algorithms. Thus, state-of-the-art denoising algorithms can benefit from the more expressive DeltaConv convolution.

## 7.1. Future work

The neighbours for every vertex is currently determined by selecting the K nearest neighbours in an extrinsic way. It would be interesting to investigate if other strategies for selecting the K nearest neighbours would increase the performance. For instance by selecting the neighbours in an intrinsic way or selecting connected vertices. We are also interested in what the effect will be on thin structures since K nearest neighbours is more susceptible to select vertices on the other side of the mesh in those cases.

In the ablation study we noticed that the scalar stream by itself performs almost as well as the complete convolution. It would be interesting to investigate if optimizations of the vector stream could boost the overall performance of the network.

# References

[1] Matthieu Armando, Jean-Sébastien Franco, and Edmond Boyer. “Mesh Denoising With Facet Graph Convolutions”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.8 (Aug. 2022). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 2999–3012. ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3045490. URL: https://ieeexplore.ieee.org/document/9296808.

[2] Harold C. Burger, Christian J. Schuler, and Stefan Harmeling. “Image denoising: Can plain neural networks compete with BM3D?” In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012 IEEE Conference on Computer Vision and Pattern Recognition. ISSN: 1063-6919. June 2012, pp. 2392–2399. DOI: 10.1109/CVPR.2012.6247952. URL: https://ieeexplore.ieee.org/document/6247952.

[3] Honghua Chen, Mingqiang Wei, and Jun Wang. *Geometric and Learning-based Mesh Denoising: A Comprehensive Survey*. Sept. 2, 2022. DOI: 10.48550/arXiv.2209.00841. arXiv: 2209.00841[cs]. URL: http://arxiv.org/abs/2209.00841.

[4] J. R. Diebel, S. Thrun, and M. Brunig. “A Bayesian method for probable surface reconstruction and decimation”. In: *Acm Transactions On Graphics* 25.1 (Jan. 1, 2006). Publisher: ASSOC COMPUTING MACHINERY, pp. 39–59. ISSN: 0730-0301. DOI: 10.1145/1122501.1122504. URL: https://espace.library.uq.edu.au/view/UQ:385739.

[5] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. “Bilateral Mesh Denoising”. In: *ACM Transactions on Graphics* 22 (May 29, 2003). ISSN: 1581137095. DOI: 10.1145/1201775.882368.

[6] Shota Hattori et al. “Learning Self-prior for Mesh Denoising Using Dual Graph Convolutional Networks”. In: *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*. Berlin, Heidelberg: Springer-Verlag, Oct. 23, 2022, pp. 363–379. ISBN: 978-3-031-20061-8. DOI: 10.1007/978-3-031-20062-5_21. URL: https://doi.org/10.1007/978-3-031-20062-5_21.

[7] Lei He and Scott Schaefer. “Mesh denoising via L0 minimization”. In: *ACM Transactions on Graphics* 32.4 (July 21, 2013), 64:1–64:8. ISSN: 0730-0301. DOI: 10.1145/2461912.2461965. URL: https://doi.org/10.1145/2461912.2461965.

[8] Klaus Hildebrandt and Konrad Polthier. *Constraint-based Fairing of Surface Meshes*. ISSN: 1727-8384. The Eurographics Association, 2007. ISBN: 978-3-905673-46-3. URL: http://dx.doi.org/10.2312/SGP/SGP07/203-212.

[9] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: International Conference on Learning Representations. July 21, 2022. URL: https://openreview.net/forum?id=SJU4ayYgl.

[10] Victor Lempitsky, Andrea Vedaldi, and Dmitry Ulyanov. “Deep Image Prior”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. ISSN: 2575-7075. June 2018, pp. 9446–9454. DOI: 10.1109/CVPR.2018.00984. URL: https://ieeexplore.ieee.org/document/8579082.

[11] Xianzhi Li et al. “DNF-Net: A Deep Normal Filtering Network for Mesh Denoising”. In: *IEEE Transactions on Visualization and Computer Graphics* 27.10 (Oct. 2021). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 4060–4072. ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3001681. URL: https://ieeexplore.ieee.org/document/9115285.

[12] Xianzhi Li et al. “Non-Local Low-Rank Normal Filtering for Mesh Denoising”. In: *Computer Graphics Forum* 37.7 (2018). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13556, pp. 155–166. ISSN: 1467-8659. DOI: 10.1111/cgf.13556. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13556.

[13] Zhiqi Li et al. “NormalF-Net: Normal Filtering Neural Network for Feature-preserving Mesh Denoising”. In: *Computer-Aided Design* 127 (Oct. 1, 2020), p. 102861. ISSN: 0010-4485. DOI: 10.1016/j.cad.2020.102861. URL: https://www.sciencedirect.com/science/article/pii/S0010448520300543.

[14] Xuequan Lu et al. “Low Rank Matrix Approximation for 3D Geometry Filtering”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.4 (Apr. 2022). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 1835–1847. ISSN: 1941-0506. DOI: 10.1109/TVCG.2020.3026785. URL: https://ieeexplore.ieee.org/document/9210753.

[15] Shitong Luo and Wei Hu. “Differentiable Manifold Reconstruction for Point Cloud Denoising”. In: *Proceedings of the 28th ACM International Conference on Multimedia*. Oct. 12, 2020, pp. 1330–1338. DOI: 10.1145/3394171.3413727. arXiv: 2007.13551[cs]. URL: http://arxiv.org/abs/2007.13551.

[16] P. Perona and J. Malik. “Scale-space and edge detection using anisotropic diffusion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.7 (July 1990). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 629–639. ISSN: 1939-3539. DOI: 10.1109/34.56205. URL: https://ieeexplore.ieee.org/document/56205.

[17] Francesca Pistilli et al. *Learning Graph-Convolutional Representations for Point Cloud Denoising*. European Conference on Computer Vision. July 6, 2020. DOI: 10.48550/arXiv.2007.02578. URL: http://arxiv.org/abs/2007.02578.

[18] Charles R. Qi et al. “PointNet++: deep hierarchical feature learning on point sets in a metric space”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., Dec. 4, 2017, pp. 5105–5114. ISBN: 978-1-5108-6096-4.

[19] Yuefan Shen et al. “GCN-Denoiser: Mesh Denoising with Graph Convolutional Networks”. In: *ACM Transactions on Graphics* 41.1 (Feb. 28, 2022), pp. 1–14. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3480168. arXiv: 2108.05128[cs]. URL: http://arxiv.org/abs/2108.05128.

[20] Yuzhong Shen and K.E. Barner. “Fuzzy vector median-based surface smoothing”. In: *IEEE Transactions on Visualization and Computer Graphics* 10.3 (May 2004). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 252–265. ISSN: 1941-0506. DOI: 10.1109/TVCG.2004.1272725. URL: https://ieeexplore.ieee.org/document/1272725.

[21] Gabriel Taubin. “A signal processing approach to fair surface design”. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’95. New York, NY, USA: Association for Computing Machinery, Sept. 15, 1995, pp. 351–358. ISBN: 978-0-89791-701-8. DOI: 10.1145/218380.218473. URL: https://dl.acm.org/doi/10.1145/218380.218473.

[22] Peng-Shuai Wang, Yang Liu, and Xin Tong. “Mesh denoising via cascaded normal regression”. In: *ACM Transactions on Graphics* 35.6 (Dec. 5, 2016), 232:1–232:12. ISSN: 0730-0301. DOI: 10.1145/2980179.2980232. URL: https://doi.org/10.1145/2980179.2980232.

[23] Yue Wang et al. “Dynamic Graph CNN for Learning on Point Clouds”. In: *ACM Transactions on Graphics* 38.5 (Oct. 10, 2019), 146:1–146:12. ISSN: 0730-0301. DOI: 10.1145/3326362. URL: https://dl.acm.org/doi/10.1145/3326362.

[24] Ruben Wiersma et al. “DeltaConv: anisotropic operators for geometric deep learning on point clouds”. In: *ACM Transactions on Graphics* 41.4 (July 22, 2022), 105:1–105:10. ISSN: 0730-0301. DOI: 10.1145/3528223.3530166. URL: https://dl.acm.org/doi/10.1145/3528223.3530166.

[25] Li Xu et al. “Deep Edge-Aware Filters”. In: *Proceedings of the 32nd International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228. PMLR, June 1, 2015, pp. 1669–1678. URL: https://proceedings.mlr.press/v37/xub15.html.

[26] H. Yagou, Y. Ohtake, and A. Belyaev. “Mesh smoothing via mean and median filtering applied to face normals”. In: *Geometric Modeling and Processing. Theory and Applications. GMP 2002. Proceedings*. Geometric Modeling and Processing. Theory and Applications. GMP 2002. Proceedings. July 2002, pp. 124–131. DOI: 10.1109/GMAP.2002.1027503. URL: https://ieeexplore.ieee.org/document/1027503.

[27] Huayan Zhang et al. “Variational Mesh Denoising Using Total Variation and Piecewise Constant Function Space”. In: *IEEE Transactions on Visualization and Computer Graphics* 21.7 (July 2015). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 873–886. ISSN: 1941-0506. DOI: 10.1109/TVCG.2015.2398432. URL: https://ieeexplore.ieee.org/document/7029103.

[28] Wangyu Zhang et al. “Guided Mesh Normal Filtering”. In: *Computer Graphics Forum* 34.7 (2015), pp. 23–34. ISSN: 0167-7055. DOI: 10.1111/cgf.12742. URL: https://doi.org/10.1111/cgf.12742.

[29] Yingkui Zhang et al. “GeoBi-GNN: Geometry-aware Bi-domain Mesh Denoising via Graph Neural Networks”. In: *Computer-Aided Design* 144 (Mar. 1, 2022), p. 103154. ISSN: 0010-4485. DOI: 10.1016/j.cad.2021.103154. URL: https://www.sciencedirect.com/science/article/pii/S0010448521001639.

[30] Wenbo Zhao et al. “NormalNet: Learning-Based Mesh Normal Denoising via Local Partition Normalization”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.12 (Dec. 2021). Conference Name: IEEE Transactions on Circuits and Systems for Video Technology, pp. 4697–4710. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2021.3099939. URL: https://ieeexplore.ieee.org/document/9495794.

[31] Zhibo Zhao et al. “A Multi-Stream Network for Mesh Denoising Via Graph Neural Networks with Gaussian Curvature”. In: *2023 IEEE International Conference on Image Processing (ICIP)*. 2023 IEEE International Conference on Image Processing (ICIP). Oct. 2023, pp. 1355–1359. DOI: 10.1109/ICIP49359.2023.10222463. URL: https://ieeexplore.ieee.org/abstract/document/10222463/authors#authors.

[32] Youyi Zheng et al. “Bilateral Normal Filtering for Mesh Denoising”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.10 (Oct. 2011). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 1521–1530. ISSN: 1941-0506. DOI: 10.1109/TVCG.2010.264. URL: https://ieeexplore.ieee.org/document/5674028.

[33] Lang Zhou et al. “Point cloud denoising review: from classical to deep learning-based approaches”. In: *Graphical Models* 121 (May 1, 2022), p. 101140. ISSN: 1524-0703. DOI: 10.1016/j.gmod.2022.101140. URL: https://www.sciencedirect.com/science/article/pii/S1524070322000170.