Denoising of triangulated meshes

Theo Braune
DMA - École Normale Superieure
theo.braune@ens.fr

January 27, 2021

Abstract

In the following we present different iterative mesh denoising algorithms, where we implement in a first phase different filters to smooth our mesh. We will use local averaging methods to achieve a smoothing of the the mesh. In a first step we will implement three different filters. One filter with a combinatorial average of the vertex positions, one filter with the distance between neighbouring vertices and as a third filter the Laplace operator with cotangent weights of the angles.
In a second step we use these filter operators to solve with a forward Euler method the Laplace heat equation with our noisy mesh as initial condition. In the final part we will make numerical experiments with several figures and different sizes of noise. In doing so, we compare the different approaches and try to optimise the signal noise ratio over the number of iterations or the stopping time of the time-evolution of the Laplace equation.
Keywords: Differential geometry, mesh processing

1 Introduction

A triangulated mesh is a data representation for 3D models. It contains the geometric point positions for each vertex and data about the connectivity of these points. These are often generated by 3D-scanning and in the process of scanning, saving and restoring, it is natural that the mesh gets some noise. But in applications as 3D-printing this could cause several problems. Therefore one is interested in methods of reducing the noise. We will assume that the noise of the mesh only acts on the positions of the vertices in the mesh and does not change the connectivity of the points in the mesh.
For us a triangulated mesh is a data structure consisting of two triples $M = (V, E, F)$ and $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, where V is the set of vertices and $E \subset V \times V$ the set of edges. Furthermore $F \subset E \times E \times E$ is the set of faces, that satisfies the compatibility condition that if $(i, j), (j, k), (k, i) \in E$, we also have $(i, j, k) \in F$. In our case we will consider the edges without orientation. That means, with an abuse of notation that (i, j) and (j, i) will be the same edges for us.

For all $i \in V$ we have $v_i \in \mathbb{R}^3$ and their union is the set $\mathcal{V} = \{v_i \mid i \in V\}$. This is the geometric point position of the vertex. Analogously we define

$$\mathcal{E} = \bigcup_{(i,j)\in\mathcal{E}} \{v_i, v_j\}, \quad \mathcal{F} = \bigcup_{(i,j,k)\in\mathcal{F}} \{v_i, v_j, v_k\}$$

For the vertices we assume that the noise is going to be of the form

$$\tilde{v}_i = v_i + \varepsilon(Rn_i),$$

where R is white noise and n_i are the normal vectors of the vertices.
We are aiming for a map that reduces the noise. To quantify this, we introduce the signal-to-noise ratio (SNR) analogous to the noise of images.

Definition 1. *Let $M, \tilde{M}$ be two meshes with the same number of vertices and the same edge set. Then we define signal noise ratio as*

$$\mathrm{SNR}(M, \tilde{M}) = -20 \cdot \log_{10}\left(\frac{\sum_i \|v_i - \tilde{v}_i\|}{\sum_i \|v_i\|}\right).$$

If we write our vertices as $V = (V_1, V_2, V_3)$, where $V_i \in \mathbb{R}^n$, it is going to be our goal to build an operator $W\colon \mathbb{R}^n \to \mathbb{R}^n$, such that for the vertices of the mesh, the application

$$(V_1, V_2, V_3) \mapsto (WV_1, WV_2, WV_3)$$

maximises the SNR ratio. We will then iterate this process and try to increase the value of the SNR like that.

For the sake of simplicity we will assume, that our mesh has no boundary, i.e $\partial M = \emptyset$. This means that for each edge $(i, j) \in E$, there are exactly two faces $f_l, f_m \in F$, such that $(i, j) \in f_l$ and $(i, j) \in f_m$.

For the Matrix W, we will test different approaches. Among others we will use the approach for the discretized Laplace-Beltrami Operator drawn out in [PP93]. We will use this operator afterwards to solve the heat equation in order to reduce the noise.

For the implementation we use the library Open3d `http://www.open3d.org/`, that allows to use meshes very easily in the needed data structures. Furthermore several helpful methods for the visualization or an simple use of the Python matplotlib, are already implemented in this library.

1.1 Related Work

We will use an approach that will make use of the local geometry and averaging methods. This will yield to difficulties when there are fine texture details, because these will vanish in our denoising process quite quickly. The approach drawn out in [ZLZ+19] attempts to address this problem through convolutional neural networks. They follow a similar ideas as in the image analysis with CNN, in such a way, that they use convolution filters to extract data of the mesh first and then analyse and modify them. The idea presented in [LLL+19] uses similar to our approach a Laplace-Denoising approach, but uses additionally statistical estimation of the noise in order to obtain a better detail preservation.

2 Method

2.1 Denoising by averaging

The basic idea of these filters will be to "average" around a vertex. We will specify what we mean by that. We will implement three differnt algorithms that have a similar idea, but use different approaches in detail.

For each $(i,j) \in E$ we define some specific *edge weight* $w_{ij} \in \mathbb{R}_+$. This will yield a matrix

$$W \in \mathbb{R}^{V \times V}, \text{ where } w_{i,j} = 0, \text{ if } (i,j) \notin E, \text{ and } w_{ij} \geq 0, \text{ if } (i,j) \in E.$$

The easiest way to define such a matrix is given by the choice of

$$w_{i,j} = \begin{cases} 0, & (i,j) \notin \mathcal{E} \\ 1, & (i,j) \in \mathcal{E} \end{cases},$$

which will be the combinatorial weights for us. Another opportunity is given by

$$w_{i,j} = \begin{cases} 0, & (i,j) \notin E \\ \|v_i - v_j\|, & (i,j) \in E \end{cases}.$$

The approach drawn out in [PP93] defines weights of the form

$$w_{i,j} = \begin{cases} 0, & (i,j) \notin E \\ \cot\big(\alpha_{i,j}\big) + \cot\big(\beta_{i,j}\big), & (i,j) \in E \end{cases},$$

We will come back why this is a convenient choice of weights. In all these cases we will build a local average operator, in such a way that for $i \in V$, we set $d_i = \sum_{(i,k)\in\mathcal{E}} w_{i,k}$ and set then

$$\tilde{W} = D^{-1}W, \text{ i.e } \quad \tilde{w}_{i,j} = \frac{w_{i,j}}{\Sigma_{(i,k)\in E} w_{i,k}}$$

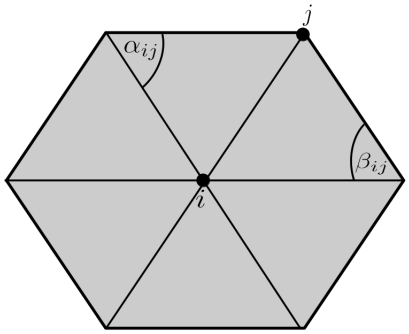


Figure 1: Illustration of the angles that will be used for the computation of the cotangent weights

We will then apply this averaging operator the the components of our mesh. Let s be the number of iterations, then we set $\tilde{V} = \tilde{W}^s V$, where we apply the operator on each of the components.
There are several more choices of weights that one could use. In [FDCo03] they draw out an other approach of choice of weights.

2.2 Denoising by heat diffusion

The main idea will be to consider the noisy mesh as initial condition of a heat equation. As one can imagine from the heat evolution in the one-dimensional case, the heat will distribute over the time. We want to use this effect in such a way, that the heat evolution will level out the noise of the mesh. We will follow the prescription of [MD99] in order to implement this algorithm.
As drawn out in [PP93] the filter with the cotangent weights could be used as an approximation for the Laplace-Beltrami operator in coordinates, which is for a coordinate frame $\{\frac{\partial}{\partial x_i}\}_{i=1}^n$ on an n-dimensional manifold given by

$$\Delta u = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial}{\partial x_j}\left(\sqrt{\det(g)}(g^{-1})_{ij}\frac{\partial}{\partial x_j}u\right), \quad g_{ij} = \left\langle \frac{\partial}{\partial x_i}, \frac{\partial}{\partial x_j} \right\rangle$$

which is unpenetrable, therefore the coordinate free derivation with the cotangent weights is superior for our applications.
For a weight matrix W and the matrix $D = \mathrm{diag}(d_i)$ defined as above, we define the laplacian as $L = D - W$, therefore we obtain for the Laplace operator normalized with the matrix D the heat equation with the time step $\tau > 0$

$$\partial_t V^{(t)} = \Delta V^{(t)} \Longrightarrow \quad \frac{1}{\tau}(V^{(n+1)} - V^{(n)}) = -D^{-1}LV^{(n)},$$

which results with a forward Euler-step in the relation

$$V^{(n+1)} = (1-\tau) \cdot V^{(n)} + \tau \tilde{W} V^{(n)}.$$

We immediately see, that this is for $\tau = 1$ just the special case of the iterative mesh denoising.

3 Results

We will illustrate our denoising algorithms with a knot first. We will apply our filters several times to the mesh and measure in each step the Signal-Noise-Ratio.

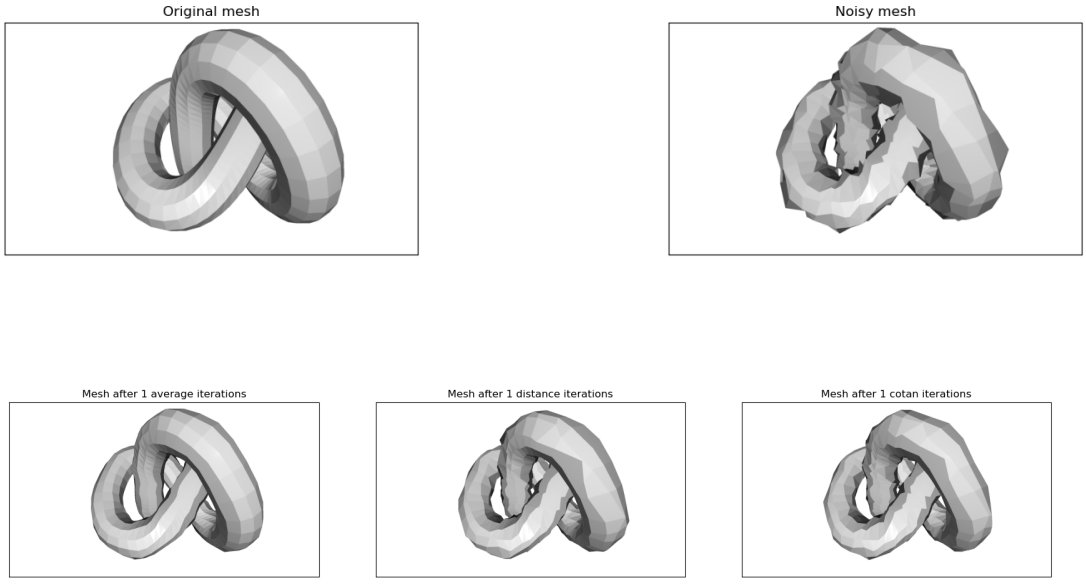


Figure 2: Example of a knot as a mesh with noise. For this particular example we have $\mathrm{SNR}(M_{\mathrm{orig}}, M_{\mathrm{noise}}) = 22.71$. In the second row we show the figures after one denoising iteration.

After one iteration we see that visually the easiest vertex averaging algorithm yields the mesh, that is closest to the original mesh. The evolution of the SNR confirms this. Over several

iterations, however, the SNR for the approach with the combinatorial weights shows the fastest decline, whereas the approach with the distance filter seems to be the most constant. If we analyse the plots of the meshes after 10 iterations, we find that the mesh for the filter with the average has the best SNR value.

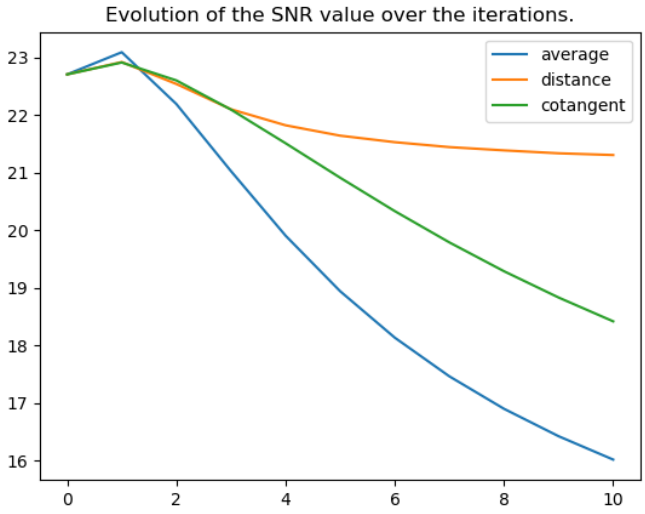


Figure 3: Evolution of the SNR over the number of iterations

However if we study the plots of the meshes after 10 iterations, we look at the meshes after 10 denoising iterations, we obtain

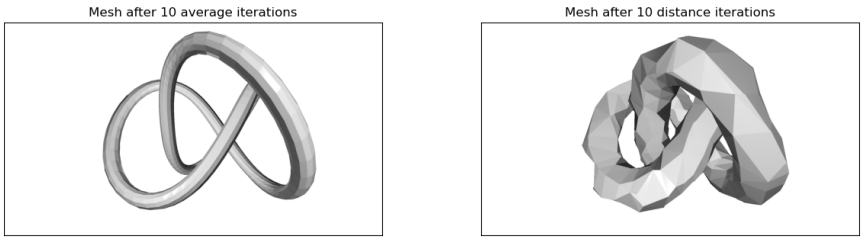


Figure 4: Plot of the mesh after 10 denoising iterations.

We observe that in the average approach we have a contraction of the surface. This makes the surface much smoother, but the differences of the vertex vectors are much higher in the distance approach, which makes the SNR value much higher. One must therefore check whether the SNR is actually a suitable value for all cases of denoising.

The iterative denoising corresponds, as described in chapter 3 to a forward Euler step, with step size 1. For the following experiment, we will reduce the step size to $\tau = 0.2$ and test with different initial noise sizes. In the table we display the optimal stopping time, the maximal SNR value and the percentage of improvement of the SNR.

SNR	average	distance	cotan
22.85	(0.8, 23.364, 1,78%)	(0.6, 23.051, 0.87%)	(1.0, 23.076, 0.98%)
28.92	(0.4, 29.22, 1,03%)	(0.2, 29.00, 0.28%)	(0.4, 29.03, 0.37%)
16.83	(1.4, 17.503, 3.84%)	(1.0, 17.18, 2.04%)	(2.2, 17.21, 2.32%)
12.93	(1.8, 13.67, 5.41%)	(1.2, 13.36, 3.22%)	(4.0, 13.40, 3.5%)

Table 1: Testing of different initial noises for the mesh. The triples in the cells include the optimal stopping time, the best SNR and the percentage of improvement of the SNR

This experiment shows, that the higher the initial noise, the longer it takes to denoise optimally. Furthermore we can observe, that the higher the initial noise, the higher is the possibility to increase the SNR with respect to the initial mesh.

It turned out, that there is still a problem with our denoising algorithm with the cotangent laplace operator. Since our programm has some deficits concerning the computational optimality, we are forced to use meshes of reduced size, because for the unreduced Stanford bunny the computation time would be way too large. In order to attack this problem, we used an in-build function from the Open3D library. It turns out that this function changes the triangle topology of our mesh, in such a way that the computation of the cotangent weights does not work anymore. Therefore we are for the cotangent laplace operator limited to unreduced meshes.
One can pose the question what this means for the other algorithms. These do not fail, but in the case that triangles are saved dublicated, then the average might not be build correctly. Nevertheless the results of our numerical tests show that they provide in practise a benefit.

If we take as a mesh the F-16 fighter airplane with missiles and repeat the same process of denoising, we can see that for similar initial noise values, we will get problems with our missiles. For the average Laplace operator we obtain

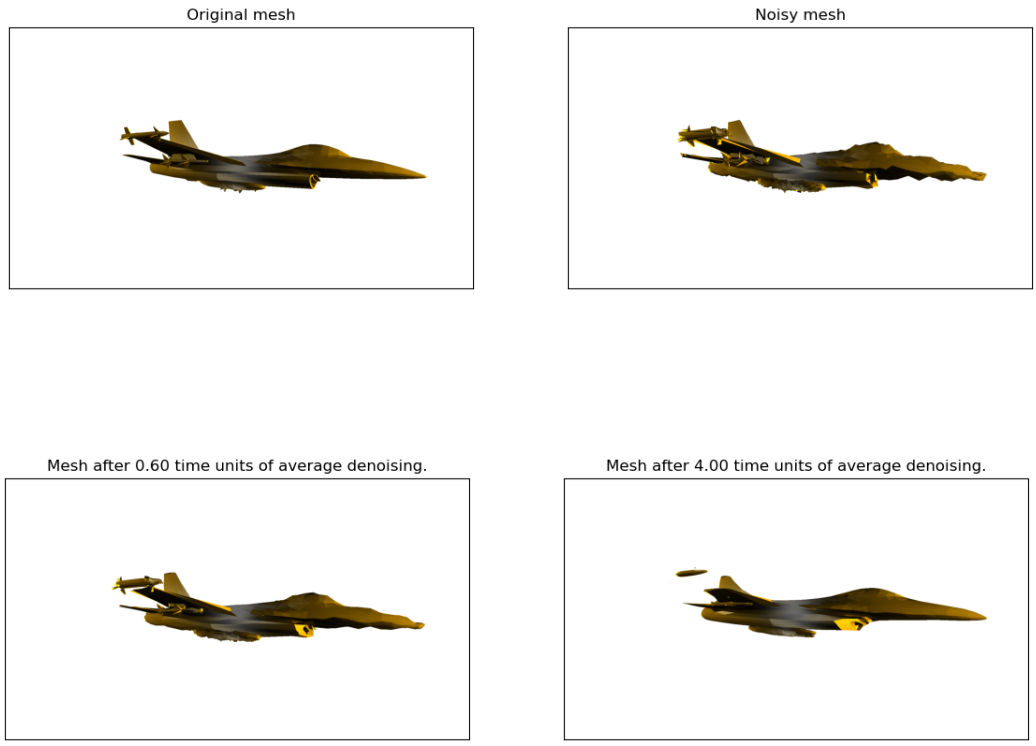


Figure 5: Plot of the time evolution of the F-16 mesh in with the combinatorial average Laplacian.

We observe that the main body of the jet becomes denoised rapidly. But the approach is problematically for fine mesh details such as the missiles. Here this goes even so far, that it seems as if that the missiles are fired of. This causes a very rapid decreasing of the SNR for this approach.

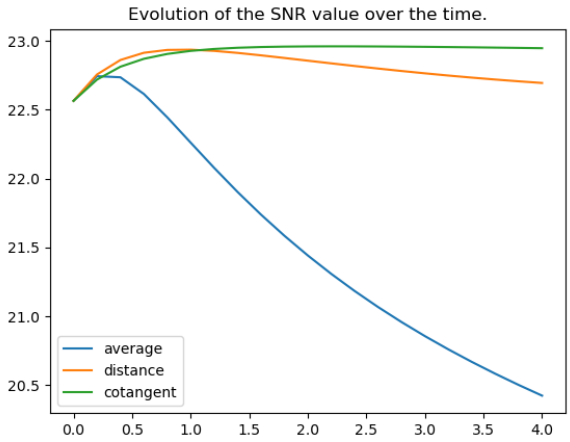


Figure 6: Evolution of the SNR for the different Laplacians.

We observe in this case, that the optimal SNR is obtained for the cotangent Laplacian after the maximal time. If we look the mesh after 4 time units of denoising, we get

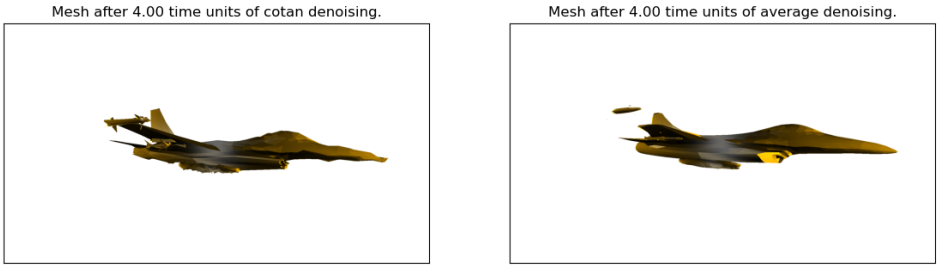
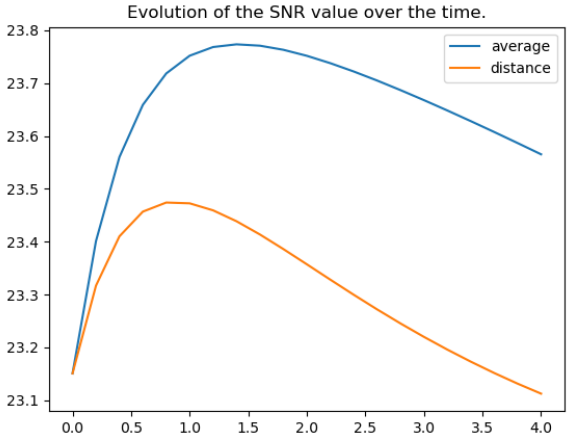


Figure 7: The F-16 mesh after 4 time units of denoising with the cotangent Laplacian (left) and the average Laplacian (right).

We can therefore conclude (among other things with further tests) that the cotangent and distance method works better on finer meshes. They need a longer evolution time, but manage

to get finer details in contrast to the average method. On the other hand, the average method is better at denoising large areas without fine details.
If we consider a jet that has a similarly fine structure but no missiles, we obtain the following SNR curve with the average and distance method



We observe that the average method now yields a better denoising result. This supports our thesis that the average method is superior for meshes without fine details.
In this particular example we are in the unfortunate position, that our cotangent Laplacian does not work, since we had to reduce the size of the jet, because otherwise the computation time would had been too large. This is for future work definitely a point, where one needs to work on.

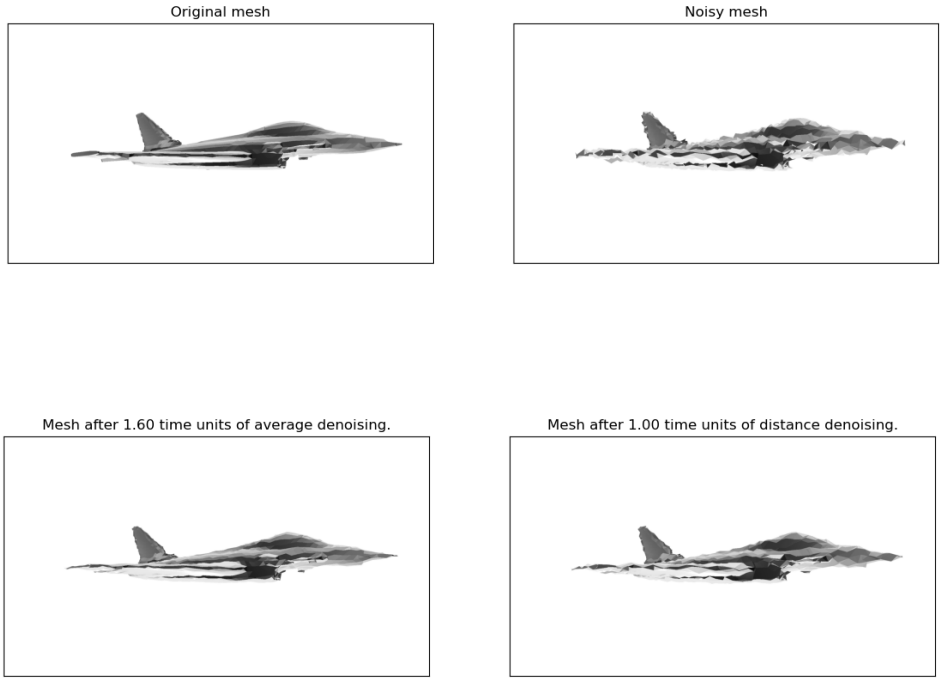


Figure 8: The F-16 mesh without missiles after their optimal stopping time.

4 Conclusion and further ideas

We have seen that the weight matrices are suitable for different meshes. If you have a priori information about the figure, you can make a suitable choice. In the example of the F-16 with missiles, however, the fuselage was shown to be well denoised. In a next step, one could therefore try to find these fine surface details and treat them differently during denoising. Such an idea is being developed in [ZLZ$^+$19], where first the surface is examined with convolutional neural networks and this information is then used for denoising. Another way to increase smoothness on surfaces during denoising is to introduce a penalty term that measures, for example, the norm of the gradient. In this case, one would not perform a discrete Euler step, but solve a minimisation problem over the set of vertices and solve it by a solver for sparse matrices. In this case, one speaks of quadratic regularisation.

References

[FDCo03] Shachar Fleishman, Iddo Drori, and Daniel Cohen-or. Bilateral mesh denoising. *ACM TRANSACTIONS ON GRAPHICS*, 22:950–953, 2003.

[LLL+19] Tao Li, Wei Liu, Hao Liu, Jun Wang, and Ligang Liu. Feature-convinced mesh denoising. *Graphical Models*, 101:17 – 26, 2019.

[MD99] Peter Schröder Alan H. Barr Mathieu Desbrun, Mark Meyer. Implicit fairing of irregular meshes using diffusion and curvature flow, 1999.

[PP93] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experiment. Math.*, 2(1):15–36, 1993.

[ZLZ+19] Wenbo Zhao, Xianming Liu, Yongsen Zhao, Xiaopeng Fan, and Debin Zhao. Normalnet: Learning based guided normal filtering for mesh denoising. *CoRR*, abs/1903.04015, 2019.