Mesh Denoising based on Normal Voting Tensor and Binary Optimization

Sunil Kumar Yadav, Ulrich Reitebuch, and Konrad Polthier

Abstract—This paper presents a two-stage mesh denoising algorithm. Unlike other traditional averaging approaches, our approach uses an element-based normal voting tensor to compute smooth surfaces. By introducing a binary optimization on the proposed tensor together with a local binary neighborhood concept, our algorithm better retains sharp features and produces smoother umbilical regions than previous approaches. On top of that, we provide a stochastic analysis on the different kinds of noise based on the average edge length. The quantitative results demonstrate that the performance of our method is better compared to state of the art smoothing approaches.

Index Terms—Geometry Processing, Mesh Smoothing, Normal Voting Tensor, Eigenvalue Binary Optimization, Noise Analysis.

1 INTRODUCTION

Mesh denoising is a central preprocessing tool in discrete geometry processing with many applications in computer graphics such as CAD, reverse engineering, virtual reality and medical diagnosis. The acquisition of 3D surface data takes place using 3D measurement technologies such as 3D cameras and laser scanners. During the surface measurement, noise is inevitable due to various internal and external factors; this degrades surface data quality and its usability. The main goal of any mesh denoising algorithm is to remove spurious noise and compute a high quality smooth function on the triangle mesh while preserving sharp features.

In general, noise and sharp features both have high frequency components, so decoupling the sharp features from noise is still a challenging problem in mesh denoising algorithms. Traditionally, noise is removed by using a low pass filtering approach, but this operation leads to feature blurring. A variety of Laplacian-based surface smoothing algorithms are available to overcome the problem of feature blurring. Our smoothing approach uses eigenanalysis and a binary optimization of the proposed element based normal voting tensor to decouple noise and sharp features. We design an iterative denoising method that removes low and high frequency noise while preserving sharp features in smooth surfaces. Our algorithm does not produce piecewise flat areas (false features) in the denoised triangular mesh.

1.1 Contributions

We introduce a simple and effective mesh denoising algorithm which does not follow the classic Laplacian approach of surface smoothing. Our algorithm follows a two stage denoising process. In the first stage, we process noisy face normals. In the second stage, we update the vertex positions accordingly. Our main contributions are as follows:

- We propose a tensor-based smoothing technique with stable and fast convergence property to remove the undesired noise from noisy surfaces.
- We apply a binary optimization technique on the eigenvalues of the proposed element-based normal voting tensor (ENVT) that helps us to retain sharp features in the concerned geometry and improves the convergence rate of the algorithm.
- We give a stochastic analysis of the effect of noise on the triangular mesh based on the minimum edge length of the elements in the geometry. It gives an upper bound to the noise standard deviation to have minimum probability for flipped element normals.

2 RELATED WORK

In the last two decades, a wide variety of smoothing algorithms have been introduced to remove undesired noise while preserving sharp features in the geometry. The most common technique for noise reduction is mainly based on the Laplacian on surfaces. For a comprehensive review on mesh denoising, we refer to [1] and [2]. We give a short overview of major related works in this section.

Isotropic smoothing methods are the earliest smoothing algorithms. These algorithms have low complexity but suffer from severe shrinkage and further feature blurring [3]. Desbrun et al. [4] introduced an implicit smoothing algorithm that produces stable results against irregular meshes and avoids the shrinkage by volume preservation. Later, the concept of the differential coordinates was introduced by Alexa [5], as a local shape descriptor of a geometry. Su et al. exploited the differential coordinates concept for mesh denoising by computing the mean of the differential coordinates at each vertex and then computes a smooth surface according to the mean differential coordinates [6]. This method produces less shrinkage but is unable to preserve shallow features. The differential coordinates framework has been extended for a variety of mesh processing algorithms by Sorkine [7]. In general, isotropic smoothing
Figure 1: The pipeline for the proposed smoothing algorithm. The yellow blocks show the face normal processing and the blue block represents the last stage (the vertex update) of the algorithm.

methods are prone to shrink volumes and blur features, but effective in noise removal.

Anisotropic diffusion is a PDE-based de-noising algorithm introduced by Perona and Malik [8]. The same concept was extended for surface denoising using a diffusion tensor [9]. Similarly, the anisotropic diffusion of surface normals was introduced for surface smoothing by Ohtake et al. [11]. These methods compute smooth normals by a weighted sum of the neighborhood element normals. The sharp feature identification, using the surface normals, has been introduced by computing the angle between the neighbor normals [13]. Tasziden et al. [14] exploited the level set surface model along the anisotropic diffusion of surface normals to produce desired results. Later, the prescribed mean curvature-based surface evolution algorithm was introduced by Hildebrandt et al. [15]. It avoids the volume shrinkage and preserves features effectively during the denoising process. Several other algorithms related to anisotropic diffusion are based on the bilateral smoothing [16], which was initially introduced by Tomasi et al. [17] for image smoothing. Later, the Gaussian kd-tree was introduced to accelerate the bilateral and local mean filtering [18]. Researchers have proposed a general framework for bilateral and mean shift filtering in any arbitrary domain [19]. These algorithms are simple and effective against noise and feature blurring. In general, anisotropic denoising methods are more robust against volume shrinkage and are better in terms of feature preservation, but the algorithm complexity is higher compared to isotropic algorithms.

The two step denoising methods are simple and quite robust against noise. These algorithms consist of face normal smoothing and vertex position updates [20]. Face normals are treated as signals on the dual graph of the mesh with values in the unit sphere. The Laplacian smoothing of the face normals on the sphere was introduced by Taubin [20] where displacement of the concerned face normal is computed along a geodesic on the unit sphere. The face normal smoothing is done by rotating the face normal on the unit sphere according to the weighted average of the neighbor face normals. Different linear and non-linear weighting functions have been introduced by different algorithms for face normal smoothing. For example, Yogou et al. [21] computed the mean and the median filtering of face normals to remove noise. Later, a modified Gaussian weighting function was applied to face normal smoothing in an adaptive manner to reduce feature blurring [22]. In continuation, the alpha trimming method introduced a non-linear weighting factor which approximates both, the mean and the median filtering [23]. Bilateral normal is one of the most effective and simple algorithms among the two step methods [24], where the weighting function is computed based on the normal differences (similarity measurement) and spatial distances between neighboring faces. Recently, a total variational method has been also introduced for mesh normal filtering [25]. After the preprocessing of the face normals, vertex position updates are done by using the orthogonality between the corresponding edge vector and the face normal [26]. The two step denoising methods are simple in implementation and produce effective results. However, on noisy surfaces, it is difficult to compute the similarity function because of the ambiguity between noise and sharp features; this may lead to unsatisfactory results.

In recent mesh denoising methods, compressed sensing techniques are involved to preserve sharp features precisely and remove noise effectively [27]. For example, the $L^0$ mesh denoising method assumes that features are sparse on general surfaces and introduces an area based differential operator. This method utilizes the $L^0$-optimization to maximize flat regions on noisy surfaces to remove noise [28]. The $L^0$ method is effective against high noise but also produces piecewise flat area on smooth surfaces. Later, the weighted $L^1$-analysis compressed sensing optimization was utilized to recover sharp features from the residual data after global Laplacian smoothing [29]. Recently, the ROF (Rudin, Osher and Fatemi) based algorithm has been introduced in [30]. This method applies $L^1$-optimization on both data fidelity and regularization to remove noise without volume shrinkage. In general, the compressed sensing based denoising algorithms are robust against high intensity noise and recover not only the sharp but also the shallow features, but at the same time these algorithms produce false features (piecewise flat areas) on smooth geometries.

A multistage denoising framework was applied in recent methods [31] and [32] where feature identification is done by the eigenanalysis of the NVT (normal voting tensor) [33]. [34]. Then, the whole geometry is divided into different clusters based on the features and then smoothing is applied on different clusters independently. Later, the guided mesh normals were computed based on the consistent normal orientation and bilateral filtering was applied [35]. Recently, Wei et al. [36] exploited both vertex and face normal information for feature classification and surface smoothing. In continuation, researchers detected features on the noisy surface using quadratic optimization and then remove noise using $L^1$-optimization while preserving features [37]. Multistage denoising algorithms produce effective results against different levels of noise but have higher algorithm complexity because of the different stages.

In our method, the face normal smoothing is motivated by the NVT-based algorithms. Noise and features are decoupled using the eigenanalysis of the ENVT and noise is removed by the multiplication of the ENVT to the corresponding face normal.
3 Method

Figure 1 shows the whole pipeline of our algorithm. The face normal smoothing (the yellow blocks in Figure 1) consists of four steps: (1) We compute the geometric neighborhood for the concerned face using a local binary scheme. (2) We define the element-based normal voting tensor within its geometric neighborhood. (3) To remove noise effectively, we apply a binary optimization on the eigenvalues of the computed tensor. (4) We multiply the modified ENVT to the corresponding face normal to suppress the noise. In the last stage (the blue block in Figure 1), we update the vertex positions using the orthogonality between the edge vectors and the face normals. In this section, we explain each stage of the proposed algorithm briefly.

3.1 Local Binary Neighbor Selection

The first step of our denoising scheme is the preprocessing of the face normals using the neighboring face normals. To select the neighborhood area $\Omega$, there are three possibilities: Combinatorial, geodesic and geometrical neighborhood. Each of these terms are explained in Appendix A. The geometrical neighborhood is applied in the proposed algorithm because it depends only on the radius of the disk irrespective of mesh resolution unlike the topological algorithm because it depends only on the radius of the hood. Each of these terms are explained in Appendix A.

To select the neighborhood area of the face normals using the neighboring face normals. The first step of our denoising scheme is the preprocessing of the proposed algorithm briefly.

3.2 Element-based Normal Voting Tensor (ENVT)

We define an ENVT on every element of a properly oriented triangulated mesh similar to the vertex based voting tensor proposed in [33]. The ENVT $C_i$ is a covariance matrix, defined on the face $f_i$:

$$C_i = \frac{1}{\sum_{j \in \Omega} w_{ij}} \sum_{j \in \Omega} w_{ij} A_j n_j \cdot n_j^T,$$

(2)

where $A_j$ is the area of the corresponding element $f_j$ and $w_{ij}$ is the weighting function as mentioned in Equation 1. Weighting by corresponding element areas makes the ENVT more robust against irregular sampling. The eigenanalysis of the given tensor identifies features on triangulated surfaces similar to the methods [33] and [32]. In our algorithm, the ENVT is represented as a mesh denoising operator which is able to suppress the noise contents from noisy surfaces while preserving sharp features. The similarity between the ENVT and the shape operator is discussed in Appendix B. The ENVT is a symmetric and positive semi definite matrix so we can represent $C_i$ using an orthonormal basis of the eigenvectors $e_k$ and real eigenvalues $\lambda_k$:

$$C_i = \sum_{k=0}^{2} \lambda_k e_k e_k^T.$$  

(3)

Geometrical Interpretation: On a noise free triangulated mesh, a planar area has only one dominant eigenvalue in surface normal direction. Two dominant eigenvalues indicate edge features where the weakest eigenvector will be along the edge direction. At a corner all three eigenvalues are dominant. Consider a cube model where the eigenvalues of the ENVT are sorted in decreasing order ($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$) and normalized, then for orthogonal features we can write: $\{\lambda_1 = 1, \lambda_2 = \lambda_3 = 0\}$ (face), $\{\lambda_1 = \lambda_2 = \frac{\sqrt{3}}{2}, \lambda_3 = 0\}$ (edge) and $\{\lambda_1 = \lambda_2 = \lambda_3 = \frac{\sqrt{3}}{3}\}$ (corner).

3.3 Eigenvalues Binary Optimization

Let us consider a noisy mesh, corrupted by a random noise with standard deviation $\sigma_n$ bounded by minimum edge length. On a planar area (face) of the geometry: $\lambda_1 \gg \sigma_n$,
and the other two eigenvalues will be proportional to noise intensity, \( \lambda_2, \lambda_3 \propto \sigma_n \). Similarly, on an edge of the geometry: \( \lambda_1, \lambda_2 \gg \sigma_n \) and \( \lambda_3 \propto \sigma_n \). On a corner of the geometry: \( \lambda_1, \lambda_2, \lambda_3 \gg \sigma_n \).

We apply binary optimization to remove noise effectively by setting the less dominant eigenvalues to zero and the dominant eigenvalues to one. Our optimization technique removes noise not only from the planar area but also along the edge direction of sharp features during the denoising process. We implemented the binary optimization by introducing a scalar threshold value \( \tau \) which is proportional to the noise intensity \( \tau \propto \sigma_n \) and smaller than the dominant eigenvalues. The \( \tilde{\lambda} \) are modified eigenvalues of the ENVT using the following optimization technique. There are three eigenvalues for feature classification so our optimization method checks the following three cases:

- At corners of noisy surfaces (smooth or sharp), the smallest eigenvalues should be bigger than the threshold value i.e. \( \lambda_3 \geq \tau \). Hence:
  \[ \tilde{\lambda}_1 = 1, \quad i \in \{1, 2, 3\} \quad \text{if} \quad \lambda_3 \geq \tau. \]

- At edges of noisy geometry (smooth or sharp), the less dominant eigenvalue should be smaller than the threshold value i.e. \( \lambda_2 < \tau \) and \( \lambda_3 \geq \tau \). Hence:
  \[ \tilde{\lambda}_2 = \tilde{\lambda}_1 = 1, \quad \tilde{\lambda}_3 = 0 \quad \text{if} \quad \lambda_2 < \tau, \quad \lambda_3 \geq \tau. \]

- In the last case, we check for planar area of the geometry. Having \( \lambda_2 < \tau \) and \( \lambda_3 < \tau \) show that the only dominant eigenvalue is \( \lambda_1 \). Hence:
  \[ \tilde{\lambda}_1 = 1, \quad \tilde{\lambda}_2 = \tilde{\lambda}_3 = 0 \quad \text{if} \quad \lambda_1 \geq \tau, \quad \lambda_3 < \tau. \]

There are the three possible combinations during the eigenvalue binary optimization. The threshold \( \tau \) has to be set by the user according to the noise intensity.

3.4 De-Noising using ENVT

Our denoising method is inspired by the feature classification characteristic of the eigenvalues of the ENVT. The smallest eigendirections (one for edges of the geometry, two in planar areas) represent noise. Multiplication of the ENVT with the corresponding element normal will suppress noise in the weak eigendirection. This operation also strengthens the element normal in the strongest eigendirection. The visual representation of this operation is shown and explained in Appendix C.

3.4.1 Anisotropic Face Normal Denoising

We recompute the ENVT by using the same eigenvectors with modified eigenvalues:

\[
\tilde{C}_f = \sum_{k=0}^{2} \tilde{\lambda}_k e_k e_k^T.
\]

Now, \( \tilde{C}_f \) will have the quantized eigenvalues according to the different features on the surface. To remove noise, we multiply the corresponding element normal with the newly computed tensor \( \tilde{C}_f \). The multiplication will lead to noise removal while retaining sharp features.

\[
\tilde{n}_i = d n_i + \tilde{C}_f n_i = d n_i + \sum_{k=0}^{2} \tilde{\lambda}_k (e_k, n_i) e_k,
\]

where \( d \) is the damping factor to control the speed of preprocessing of the face normals. We use \( d = 3 \) for all experiments. The second row of Figure 5 shows the face normal denoising using the tensor multiplication.

![Figure 5: Stable convergence with different number of iterations and the corresponding processed face normals (XY-plane view). (a) Noisy cube model, (b) The result after 30 iterations, low frequency noise can be seen on the face of the Cube model. (c) After 60 iterations, smoother compared to Figure (b). (d) After 200 iterations. There is no significant difference between Figure (c) and (d).](image)

3.4.2 Vertex Update

In the last stage of the denoising algorithm, we synchronize the vertex position with corresponding newly computed face normals. To compute the proper vertex position, the orthogonality between edge vectors and face normals is used. An energy function is then defined as follows:

\[
\min_{v_i} \sum_{j=0}^{N_v(i)-1} \sum_{(i,j) \in \partial F_k} \| \tilde{n}_k \cdot (v_i - v_j) \|^2,
\]

where \( v_i \) is the vertex position and \( N_v \) represents the number of vertices of the vertex star of \( v_i \), \( \partial F_k \) is the boundary.
edge of the vertex star of \( v_i \) shared with face \( f_k \). \( \vec{n}_i \) is the smooth face normal at \( f_k \). Taubin [20] explained that the face normal vector can be decomposed into a normal and a tangential component and the main problem here is to find the vertex positions which minimize the tangential error. The possible solution of Equation 6 may be a mesh with degenerate triangular faces. Like Taubin [20], to avoid the degenerate solution, we are using gradient descent that will lead to the optimal vertex positions.

\[
\vec{v}_i = v_i + \frac{1}{F(v_i)} \sum_{j=0}^{N,v(i)-1} \sum_{(i,j) \in \partial f_k} \vec{n}_k \cdot (v_i - v_j),
\]

(7)

where \( F(v_i) \) is the number of faces connected to the vertex \( v_i \). We iterate the whole procedure several times and the number of iterations depends on the noise intensity. Figure 3 shows the effect of each stage of the face normal processing in the proposed algorithm.

### 3.5 Effect of Noise on the Proposed Method

Noise is inevitable during digital data acquisition of real life objects. The high intensity of noise flips edges in the geometry and that leads to the inconsistent face normals on a geometry. As we mentioned in section 3.2, the ENVT is defined on properly oriented surfaces with consistence face normals because the spectral decomposition of the ENVT is invariant to the face normal orientation. In this section, we give a stochastic approximation about the relation between noise and geometry resolution to prevent edge flips in the geometry. Let us consider a smooth triangular mesh \( M_s \) which is corrupted by noise \( \vec{N} \), \( M = M_s + \vec{N} \). The noise \( \vec{N} \) can be approximated by a random vector \( X_n \) consisting of three independent random variables. We assume, that the random vector \( X_n \) follows the Gaussian distribution; this is a realistic model for noise from 3D scanning [39]. Let \( \sigma_n \) be the standard deviation of noise in each independent direction, then:

\[
P\{ |X_n| \leq \sigma_n \} = 0.682,
\]

\[
P\{ |X_n| \leq 2\sigma_n \} = 0.954,
\]

\[
P\{ |X_n| \leq 3\sigma_n \} = 0.997.
\]

To explain the probability of normal flips, we switch to the 2D case of a polygon in \( \mathbb{R}^2 \). Let us consider an edge vector \( l \) between two vertices \( v_0 \) and \( v_1 \) in \( \mathbb{R}^2 \); \( l = (v_0 - v_1) \). We give a probabilistic estimation of the effect of noise on the edge \( l \) w.r.t. the noise intensity (standard deviation) \( \sigma_n \). Our analysis is mainly focused on the proper orientation of the edge normal. Wrong orientation of the edge normal \( \vec{n} \) leads to an edge flip in the smooth geometry. We denote by \( \Omega_1 \) and \( \Omega_2 \) the sets of correctly oriented edge normals and wrong oriented edge normals respectively. The probabilistic estimation of the orientation of the edge normals based on noise intensity and edge length is given as follows:

- Probability of an edge to have a correctly oriented edge normal:

\[
P\{ \vec{n} \in \Omega_1 \} = \begin{cases} 0.682 & \text{if } \sigma_n \leq \frac{1}{4} \\ 0.954 & \text{if } \sigma_n \leq \frac{3}{4} \end{cases}.
\]

- Similarly, the probability of an edge to have a wrong oriented edge normal:

\[
P\{ \vec{n} \in \Omega_2 \} \leq \begin{cases} 0.318 & \text{if } \sigma_n \leq \frac{1}{4} \\ 0.046 & \text{if } \sigma_n \leq \frac{3}{4} \end{cases}.
\]

Due to the presence of noise, edge flipping may occur when the vector sum of the vertex dislocations at the edge is bigger than the edge length. This is similar to the sampling theorem, where a signal can be reconstructed properly if and only if the data is sampled with a frequency bigger than twice the highest frequency of the data signal.

Using the given analysis, for a given probability density function and an upper bound to the standard deviation, we can estimate the expected number of edge flips in the geometry. If a surface is affected by noise only in normal direction, then there is no edge flip, irrespective of the probability density function of the noise.

We also experimented with uniformly distributed noise where the random variable \( X_n \) follows the uniform distribution; this is a realistic model for noise from 3D scanning [39]. Let \( \sigma_n \) be the standard deviation of noise in each independent direction, then:

\[
P\{ |X_n| \leq \sigma_n \} = 0.682,
\]

\[
P\{ |X_n| \leq 2\sigma_n \} = 0.954,
\]

\[
P\{ |X_n| \leq 3\sigma_n \} = 0.997.
\]

### Table 1: Quantitative Comparison

| Models  | Method | MSE (mm) | PSNR (dB) | Parameters |
|---------|--------|----------|-----------|------------|
| Cube    | [15]   | 5.267    | 3.142     | (0.3, 0.3, 30) |
| [F] = 3808 |       | 1.935    | 1.432     | (0.05, 0.05, 50) |
| [V] = 1906 |       | 1.159    | 1.247     | (0.3, 60) |
| [N] = 401 |       | 4.476    | 1.166     | (1.4) |
| [S] = 4167 |       | 1.446    | 1.166     | (Default) |
| Ours    | [28]   | 0.790    | 1.034     | (0.3, 0.15, 50) |
| Devil   | [15]   | 4.983    | 3.886     | (0.3, 0.4) |
| [F] = 25906 |      | 3.444    | 3.532     | (0.2, 0.05, 30) |
| [V] = 12986 |      | 2.641    | 2.965     | (0.3, 50) |
| [N] = 6699 |      | 4.012    | 2.628     | (4.0) |
| [S] = 2870 |      | 4.947    | 2.944     | (Default) |
| Ours    | [28]   | 5.396    | 3.152     | (0.3, 0.15, 50) |
| Joint   | [15]   | 4.983    | 3.886     | (0.3, 0.4) |
| [F] = 52226 |      | 3.444    | 3.532     | (0.2, 0.05, 30) |
| [V] = 26111 |      | 2.641    | 2.965     | (0.3, 50) |
| [N] = 6699 |      | 4.012    | 2.628     | (4.0) |
| [S] = 2870 |      | 4.947    | 2.944     | (Default) |
| Ours    | [28]   | 5.396    | 3.152     | (0.3, 0.15, 50) |
| Fandisk | [15]   | 9.567    | 4.422     | (0.4, 0.4) |
| [F] = 6475 |      | 5.856    | 4.910     | (0.07, 0.05, 30) |
| [V] = 2124 |      | 2.727    | 1.977     | (0.470) |
| [N] = 4788 |      | 5.410    | 5.415     | (1.4) |
| [S] = 2221 |      | 1.702    | 1.702     | (Default) |
| Ours    | [15]   | 3.1      | 3.1       | (0.3, 0.2, 50) |
| Rocke    | [15]   | 5.372    | 44.22     | (0.3, 0.3, 40) |
| [F] = 42912 |     | 5.982    | 51.35     | (0.2, 0.05, 50) |
| [V] = 24106 |     | 5.713    | 23.47     | (1.0, 100) |
| [N] = 7486 |     | 34.70    | 34.70     | (1.4) |
| [S] = 6486 |     | 30.45    | 30.45     | (Default) |
| Ours    | [15]   | 5.410    | 20.05     | (0.3, 0.1, 60) |
| Vase    | [15]   | 2.92     | 1.72      | (1.6) |
| [F] = 3827 |      | 4.26     | 0.83      | (Default) |
| [V] = 7650 |      | 3.34     | 0.42      | (0.3, 0.03, 30) |

### 4 Experiments, Results and Discussion

We evaluated the capacity of our algorithm on various kinds of CAD (Figure 9, 14). CAGD (Figure 10, 15) models corrupted with synthetic noise and real scanned data (Figure 16, 17, 20) models with different types of features. Noisy surfaces with non-uniform mesh corrupted with different kinds of noise (Gaussian, Impulsive, Uniform) in different random directions are also included in our experiments. We
compared our method to several state-of-the-art denoising methods in which we implemented [15], [24], [28] and [40] based on their published article and several results of [18], [35], [36] and [37] are provided by their authors.

Parameters: We discussed several parameters (geometric neighbor radius $r$, dihedral angle threshold $\rho$, eigenvalue threshold $\tau$, damping factor $d$ and iteration $p$). Throughout, the whole experimentation, we fixed $\rho = 0.8$, $d = 3$. Effectively, there are only 3 parameters to tune the results, in which $\tau$ is the most important as it depends on noise intensity but at the same time this parameter is not highly sensitive. We use $\tau \in \{0.3 - 0.4\}$ for synthetic data and $\tau \in \{0.05 - 0.1\}$ for real data because real data have smaller noise intensity compared to synthetic data in our experiments. The neighborhood radius $r$ depends on the number of elements within the geometric neighborhood.

Figure 6: The effect of the eigenvalue binary optimization threshold value $\tau$ on results of the proposed algorithm. The first row shows the cube model ($|V| = 24578$, $|F| = 49152$) corrupted by synthetic Gaussian noise ($\sigma_n = 0.4l_e$) where $l_e$ is the average edge length and the second row shows the scanned box model (real data with $|V| = 149992$, $|F| = 299880$) and corresponding results regarding different values of $\tau$. The third row shows the magnified area of the box model. For the box model, the algorithm produced the optimal result with smaller value of $\tau = 0.03$ because of low noise whereas the cube model needed the higher value of $\tau = 0.5$ because of high intensity noise. The bigger value of $\tau$ can lead to the feature blurring as shown in Figure (b) for the box model.

Figure 7: The results obtained by our method against different kinds of noise. (a) Original vase model. (b) Impulse noise. (c) Corresponding result with our method. (d) The vase model is corrupted by uniformly distributed random noise. (e) Corresponding result with our method.

Figure 8: Robustness against different levels of noise: The first row shows the cube model corrupted with different levels of noise. The second row shows the corresponding results obtained by the proposed method. In Figure (d), the noise level is bigger than the feature size and it is impossible to decouple features from noise. As a consequence, we are not able to recover the perfect cube.

region. We iterate several times ($p \in \{40 - 60\}$) to obtain better result. In the quantitative comparison Table 1, the parameters are given in the following format: $(\tau, r, p)$. For the [35] and [36] methods, we mention Default in the parameter column because smooth models are provided by those authors. We are following a similar pattern for other algorithms too. $(\sigma_c, \sigma_s, p)$ for [40], $(\sigma_s, p)$ for [24], $(\lambda, s, p)$ for [15] and $(\alpha)$ for [28], where $\sigma_c, \sigma_s$ are the standard deviation of the Gaussian function in the bilateral weighting, $s$ and $\lambda$ represent the step size and the smoothing threshold. The term $\alpha$ controls the amount of smoothing.

Effect of $\tau$: To see the effect of different values of $\tau$, we have experimented with two different models, the Box model (real data, with different level of features and less
noise) and the Cube model (with limited features and high noise). With smaller values of $\tau \in [0.01, 0.05]$, there is not much change on the cube model because of the higher noise whereas the box model manages to remove the noise and also the shallow features with increasing $\tau$. So $\tau$ is responsible for removing the noise and also for preserving features. If the feature size is smaller than the noise intensity, feature preservation is an ill-posed problem as shown in Figure 6.

**Neighborhood Comparison:** Figure 9 shows that the geometrical neighborhood is more effective against irregular meshes compared to the topological neighborhood. The geodesic neighborhood is quite similar to the geometrical neighborhood but it is not appropriate when a model is corrupted by high intensity of noise.

**Visual Comparison:** The Block (Figure 9), the Joint (Figure 10), the Cube (Figure 11) and the Devil (Figure 12) have non-uniform meshes corrupted with Gaussian noise in random direction. Figure 10 shows that the proposed method produces a smooth model with sharp features without creating any false features (piecewise flat areas like frequency noise (we can see smooth ripples). Method [35] could not manage to retain the circular area. It also produces good results but at the narrow cylindrical area it could not manage to retain the circular area. It also produces some false features at the non-uniform sharp corner. Method [36] could not manage to retain the sharp features. Similarly, we can see this behavior for the Cube model in Figure 11. The Rockerarm model (Figure 12) has a considerably non-uniform mesh and our method better retains sharp features (the screw part) compared to [35], [15], [40] while removing the noise better compared to methods [24], [28]. Figure 13 shows the robustness of our method against volume shrinkage. The horns of the model have the minimum shrinkage (the screw part) compared to [35], [15], [40] while removing the noise better compared to other methods. Figure 14 shows the comparison of our method to method [37] with four different models (real and synthetic data). For the Pierret and the Cube models, our method produces smoother (non-feature region of the Pierret model) results while preserving all necessary features (corners at the cube) compared to method [37]. For the Julius model, our method produces a smoother result, but at the cost of some fine features (around the eyes) compared to method [37]. For the Vase model, method [37] produces quite similar results to ours. However, the shrinkage effect is bigger in method [37] as shown in Table 1. Figure 17 shows the robustness of the proposed algorithm against irregular meshes and holes in the real data. The Gorgoyle and the Eagle model have several holes and spikes, but our smoothing algorithm manages to produce a smooth surface with proper features.

**Robustness against noise:** Our method is invariant against different kinds of noise as shown in Figure 7 where the vase model is corrupted by impulsive and uniform noise. The proposed method does not produce appropriate results above a certain level of noise as shown in Figure 8. Figure 18 shows that our method is robust against random edge flips.

**Quantitative Comparison:** In this section, we give a quantitative comparison of our method with state-of-the-art methods. We are using two different parameters: $E_v$ ($L^2$ vertex-based error) and MSAE (the mean square angular error). The positional error from the original ground truth model is represented by $E_v$ and defined as [26]:

$$E_v = \sqrt{\frac{1}{3} \sum_{k \in F} \sum_{i \in V} \sum_{j \in F_i(i)} A_j \text{dist}(\hat{v}_i, T)^2},$$

where $F$ is the triangular element set and $V$ represents the set of vertices. $A_k$ and $A_j$ are the corresponding element areas and $F_i(i)$ is the number of elements in the $i^{th}$ vertex-ring, $\text{dist}(\hat{v}_i, T)$ is the closest $L^2$-distance between the newly computed vertex $\hat{v}_i$ and the triangle $T$ of the reference model. The MSAE computes the orientation error between the original model and the smooth model and is defined as:

$$MSAE = E[\angle(\hat{n}, n)],$$

where $\hat{n}$ is the newly computed face normal and $n$ represents the face normal of the reference model. $E$ stands for the expectation value. The quantitative comparison Table 1 shows that our method performs better for most of the models e.g. Cube, Devil, Joint etc. For some model like Fandisk, our method produces quite similar numeric errors as state-of-the-art methods. **Convergence:** Our smoothing algorithm has a stable and fast convergence (as shown in...
Modification of the eigenvalues of the ENVT will not affect the orientation of the corresponding face normal when noise is removed because the difference between two eigenvalues will be zero and also the less dominant eigenvalue will be zero. There will be no more modification on noisy surfaces after some iteration when we meet the explained scenario as shown in Figure 5, where there is no significant change (visually) after 60 iterations. Figure 21 shows that the proposed method converges with minimum error compared to the methods [15] and [24]. We can see that after 40 iterations, our method is almost stable and does not produce significant changes. The eigenvalue binary optimization not only helps in preserving features, but also improves the convergence rate of the algorithm.

Running Time Complexity: Running time complexity of the proposed method is similar to most of the two-step denoising methods [11], [20], [23], [24]. The neighborhood computation is done by the growing disk method to compute the ENVT. The ENVT computation has the complexity of $O(\cdot n_f \cdot p)$, where $c$ is the number of elements within the neighborhood, $n_f$ and $p$ are the numbers of elements and iterations respectively. The tensor multiplication procedure has the running complexity of $O(n_f)$. Similarly, the vertex update procedure has the complexity of $O(c \cdot n_v \cdot p)$, where $n_v$ is the number of vertices in the geometry. In general,
Figure 13: The Devil model consists of non-uniform triangles corrupted by Gaussian noise with standard deviation $\sigma_n = 0.15l_e$. The first row shows the results produced by state-of-the-art methods and our proposed method. The second row shows the magnified view of the left eye and the right horn of the devil. The results show that the proposed method has minimum shrinkage at the horn area.

Figure 14: Fandisk model corrupted with a Gaussian noise ($\sigma_n = 0.3l_e$) in random direction. The first row shows the results produced by state-of-the-art methods and our proposed method. The second row shows the corner and the cylindrical region of the concerned model. The proposed method does not produce any false features in the umbilical region while retains sharp features and corners.

Figure 15: Left to right: (a) Noisy Dragon model corrupted with synthetic noise. (b) Denoised model using [18]. (c) The result obtained using the proposed method.

$n_f > n_v$, so the overall complexity of the algorithm will be $O(c \cdot n_f \cdot p)$. The number of elements in the geometric neighborhood $c$ plays an important role in the running time of the algorithm as shown in Table 2. For example, the Devil model has smaller number of elements and larger running time compared to the Joint model because of the different geometric neighborhood radius. The Bilateral normal method [24] uses a fix number of neighborhood elements (depending on the valence of vertex) for face normal smoothing and is a bit faster compared to the proposed method. However, the other recent two step denosing methods [35], [36] are slower compared to our method because of their additional denoising steps.

Table 2: Running Time (in seconds)

| Models | Cube | Devil | Joint | Fandisk | Rockerarm | Vase |
|--------|------|-------|-------|---------|-----------|------|
| Time (s) | 1.2 | 51.9 | 38.2 | 5.06 | 27.5 | 1.1 |

5 Conclusion and Future work

In this paper, we presented a simple and effective tensor multiplication algorithm for feature-preserving mesh denoising. The concept of the element-based normal voting tensor (ENVT) has been introduced and eigenanalysis of this tensor leads to decoupling of features from noise. We have shown that the proposed method does not need any additional Laplacian-based smoothing technique to remove the noise, like multistage state-of-the-art methods [31], [32], [35], [36]. Our method removes noise by multiplying the ENVT to the corresponding face normal, and this reduces the complexity of the algorithm. We have introduced the concept of eigenvalue binary optimization that not only enhances sharp features but also improves the convergence rate of the proposed algorithm. The local binary neighborhood selection helps to select similar elements in the neighborhood to compute the element based normal voting tensor which avoids feature blurring during the denoising.
process. We provide a stochastic analysis of the noise effect on the geometry depending on the average edge length of the triangulated mesh. On the basis of this analysis, we can provide an upper bound on the noise standard deviation depending on the minimum edge length to reconstruct the smooth surface from the noisy surface. The experimental results (visual and quantitative) show the capability of the proposed algorithm. Our method produces good results not only in terms of visual but also quantitatively with all kind of data including CAD, CAGD and real data. We have also shown the robustness of the algorithm against different kinds and levels of noise. We also discussed the wrong orientation of triangles in presence of strong noise. In future work, we would like to solve problem of edge flips and extend our algorithm to point set smoothing.

REFERENCES

[1] M. Botsch, M. Pauly, C. Ross, S. Bischoff, and L. Kobbelt, “Geometric modeling based on triangle meshes,” in ACM SIGGRAPH 2006 Courses, ser. SIGGRAPH ’06. New York, NY, USA: ACM, 2006.

[2] M. Botsch, P. Alliez, L. Kobbelt, M. Pauly, and B. Lévy, Polygon mesh processing. Natick, Mass. A K Peters, 2010.

[3] D. A. Field, “Laplacian smoothing and delaunay triangulations,” Communications in Applied Numerical Methods, vol. 4, no. 6, pp. 709–712, 1988.

[4] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, “Implicit fairing of irregular meshes using diffusion and curvature flow,” in Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, ser. SIGGRAPH ’99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 317–324.

[5] M. Alexa, “Differential coordinates for local mesh morphing and deformation,” The Visual Computer, vol. 19, no. 2-3, pp. 105–114, 2003.
Figure 19: The Julius model corrupted by Gaussian noise ($\sigma_n = 0.2 l_e$) in random direction. The results are produced by state-of-the-art methods and the proposed method. Our method does not outperform state-of-the-art methods and the output is quite similar to method [24]. The proposed method removes low frequency noise better than method [35] where we can see small ripples of noise on the Julius model.

Figure 20: Comparison with method [37]: Real data (scanned Cube, Pierret model) and Vase, Julius models are corrupted by synthetic noise. Second row shows the results obtained by Lu et al. [37] and third row presents the results obtained by the proposed method.

Figure 21: Convergence plot comparison between the methods [15], [24] and our method. The error metrics MSAE and $E_v$ are computed for the Cube model and $p$ is the number of iterations. Figure shows that the proposed method has better convergence rate compared to the methods [15] and [24].

[6] Z.-X. Su, H. Wang, and J.-J. Cao, “Mesh denoising based on differential coordinates,” in Shape Modeling and Applications, 2009. SMI 2009. IEEE International Conference on, June 2009, pp. 1–6.

[7] O. Sorkine, “Differential representations for mesh processing,” Computer Graphics Forum, vol. 25, no. 4, pp. 789–807, 2006.

[8] P. Perona and J. Malik, “Scale-space and edge detection using anisotropic diffusion,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 12, no. 7, pp. 629–639, Jul 1990.

[9] U. Clarenz, U. Diewald, and M. Rumpf, “Anisotropic geometric diffusion in surface processing,” in Proceedings of the Conference on Visualization ‘00, ser. VIS ‘00. Los Alamitos, CA, USA: IEEE Computer Society Press, 2000, pp. 397–405.

[10] C. L. Bajaj and G. Xu, “Anisotropic diffusion of surfaces and functions on surfaces,” ACM Trans. Graph., vol. 22, no. 1, pp. 4–32, Jan. 2003.

[11] Y. Ohtake, A. Belyaev, and I. Bogavski, “Mesh regularization and adaptive smoothing,” Computer-Aided Design, vol. 33, no. 11, pp. 789 – 800, 2001.

[12] Y. Ohtake and E. Belyaev, “Nonlinear diffusion of normals for stable detection of ridges and ravines on range images and polygonal models,” in In MVA2000, IAPR Workshop on Machine Vision Applications, 2000, pp. 497–500.

[13] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, “Feature sensitive surface extraction from volume data,” in Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ser. SIGGRAPH ’01. New York, NY, USA: ACM, 2001, pp. 57–66.

[14] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher, “Geometric surface smoothing via anisotropic diffusion of normals,” in Proceedings of the Conference on Visualization ’02, ser. VIS ’02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 125–132.

[15] K. Hildebrandt and K. Polthier, “Anisotropic filtering of non-linear surface features,” Computer Graphics Forum, vol. 23, pp. 391–400, 2004.

[16] D. Barash, “Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24,
APPENDIX A
NEIGHBORHOOD DEFINITIONS

Definition A.1. Combinatorial neighborhood is defined as the set of all elements connected with the vertices of the corresponding face:

$$\Omega_i = \{f_j | v_{i1} \in f_j \lor v_{i2} \in f_j \lor v_{i3} \in f_j\},$$

where the neighborhood region is presented by $\Omega$ and the vertices $v_{i1}, v_{i2}$ and $v_{i3}$ belong to the face $f_i$.

Definition A.2. Geometrical neighborhood is defined as the set of all elements belonging to the disk area of the desired radius and centered at the corresponding element:

$$\Omega_i = \{f_j | |c_j - c_i| \leq r\},$$

where $c_i$ and $c_j$ are the centroids of the central and neighbor element and $r$ is the radius of the neighbor disk for the geometrical neighborhood.

Definition A.3. Geodesic neighborhood is defined as the set of all elements within the shortest distance defined by the radius $r$:

$$\Omega_i = \{f_j | D(f_i, f_j) \leq r\},$$

where $f_i$ is the source point and $D(f_i, f_j) : \mathcal{M} \rightarrow \mathbb{R}$ is a geodesic distance function on a manifold surface $\mathcal{M}$.

APPENDIX B
RELATION BETWEEN THE SHAPE OPERATOR AND THE ENV\textit{T}

Let us consider a smooth manifold surface $\mathcal{M}$ embedded in $\mathbb{R}^3$. We assume that the surface is orientable and it has well defined normal field $\mathcal{N} : \mathcal{M} \rightarrow S^2$ on $\mathcal{M}$. Then we can define our proposed tensor $C_{\Omega}(p) : \mathbb{R}^3 \rightarrow \mathbb{R}^{3 \times 3}$ around a point $p$:

$$C_{\Omega}(p) = \int_{\Omega} (n \cdot n^T) d\Omega,$$

where $n \in \mathcal{N}$ and $\Omega$ is the local neighborhood area of point $p$.

To compute the shape operator on a surface, the surface must be $C^2$ and properly oriented. The shape operator is defined on the tangent plane $S : T_p \mathcal{M} \times T_p \mathcal{M} \rightarrow \mathbb{R}$. $T_p \mathcal{M}$ represents the tangent plane, spanned by the basis vectors $\xi_1$ and $\xi_2$. The eigenvalues of the shape operator $S$ are the principle curvatures $\kappa_1$ and $\kappa_2$ and eigenvectors will be a basis of the tangent plane ($\xi_1, \xi_2$). The eigenvectors are called the principle curvature directions.

Represented in the orthonormal basis of the normal and the principle curvature directions, the surface normal in a local neighborhood of point $p$ can be approximated by:

$$n(\xi_1, \xi_2) = [1, \kappa_1 \xi_1, \kappa_2 \xi_2].$$

Now, we can compute the covariance matrix using the above normal vector:

$$n \cdot n^T = \begin{bmatrix}
1 & \kappa_1 \xi_1 & \kappa_2 \xi_2 \\
\kappa_1 \xi_1 & \kappa_1^2 \xi_1^2 & \kappa_1 \kappa_2 \xi_1 \xi_2 \\
\kappa_2 \xi_2 & \kappa_1 \kappa_2 \xi_1 \xi_2 & \kappa_2^2 \xi_2^2
\end{bmatrix}.$$  (8)
Figure 22: The Devil model consists of non-uniform triangles corrupted by Gaussian noise with standard deviation \( \sigma_n = 0.15l_e \). The results produced by state-of-the-art methods and the proposed method are colored by the absolute value of the area weighted cotangent mean curvature.

Figure 23: The Julius model corrupted by Gaussian noise \( (\sigma_n = 0.2l_e) \) in random direction. The results which are produced by state-of-the-art methods and the proposed method show that our method is quite similar to method [24] and better compared to methods [35] and [37].

Figure 24: The Cube model consists of non-uniform triangles corrupted by Gaussian noise \( (\sigma_n = 0.3l_e) \) in normal direction. The mean curvature coloring shows that the proposed method outperforms state-of-the-art methods.

Figure 25: Shows the basic idea behind the proposed method to remove noise in \( \mathbb{R}^2 \) where \( e_i \) and \( \lambda_i \) represent the eigenvectors and eigenvalues of the proposed element based voting tensor and \( n \) shows the noisy normal. We rotate the noisy normal \( n \) towards the dominant eigendirection \( e_1 \) by a corresponding tensor multiplication.

For a small symmetric \( \Omega \), the integral for the off-diagonal components will be zero:

\[
\int_\Omega n \cdot n^T d\Omega = \begin{bmatrix}
1 & 0 & 0 \\
0 & \kappa_1^2 & \kappa_1^2 \\
0 & 0 & \kappa_2^2
\end{bmatrix}.
\]

So by this approximation, the shape operator is contained in the covariance matrix as the lower right \( 2 \times 2 \) sub-matrix. Therefore the second and third eigenvalue of the NVT approximate the squares of the principal curvatures \( \kappa_1 \) and \( \kappa_2 \).

**APPENDIX C**

**A VISUAL REPRESENTATION OF THE MESH DENOISING**

Traditionally, face normal smoothing is done by rotating the concerned face normal along a geodesics on the unit sphere whereas our method aligns the face normals by projection.
We project noisy face normals towards the smooth normals by multiplication of the ENVT to the corresponding face normal.

A demonstration of the ENVT multiplication to the corresponding noisy face normal in $\mathbb{R}^2$ is shown in Figure 25. If $\lambda_2 > \lambda_1$, then it will strengthen the face normal in the desired direction $e_1$ and suppresses noise in the $e_2$ direction. The whole procedure consists of the following steps:

- A noisy face normal $n$ is decomposed according to the eigenbasis $(e_1$ and $e_2$) of the element based normal voting tensor, Figure 25 (left).
- Then, the modified eigenvalues ($\lambda_1$ and $\lambda_2$) get multiplied in the corresponding eigendirections to suppress the weak eigenvalue in weak eigendirection, Figure 25 (middle).
- Finally, the new element normal $n^p$ is obtained by normalizing $C_i \cdot n$, in Figure 25 (right).

**APPENDIX E**

**COMPARISON WITH ROBUST IMPLICIT MOVING LEAST SQUARES (RIMLS)**

The RIMLS algorithm is a features preserving point set surface reconstruction method [41]. It follows the moving least squares (MLS) method along with robust statistics to produce a high fidelity surface in presence of noise. Figure 27 shows the comparison of the proposed method with the RIMLS method.

**APPENDIX D**

**MEAN CURVATURE COLORING**

Mean curvature coloring can be used as a tool to show the capability of a denoising algorithm. In our experiments, we use the cotangent mean curvature operator that is computed at each vertex of the geometry [15]. The coloring is done by using the absolute value of the mean curvature vector. Figure 22 shows that our method is able to remove noise quite similar to methods [24] and [35] while it produces less shrinkage and better retains the sharper features compared to the other methods. Similarly, for the Julius model (Figure 23), the proposed method produces a smoother surface compared to methods [35] and [37] and retains sharper features similar to the Bilateral normal method [24], as shown in Figure 23. The Cube model has non-uniform meshes and mean curvature coloring indicates that our method is better compared to other state-of-the-art methods in terms of noise removal and retaining sharp features. Figure 26 shows that the proposed method is also effective for real data (data obtained from 3D scanners).