Adaptive patches for mesh denoising

Jan Hurtado, Marcelo Gattass, Alberto Raposo and Jéferson Coelho
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rio de Janeiro, Brazil
{hurtado,mgattass,abraposo,jcoelho}@tecgraf.puc-rio.br

***Abstract*—The generation of triangular meshes typically introduces undesired noise which comes from different sources. Mesh denoising is a geometry processing task to remove this kind of distortion. To preserve the geometric fidelity of the desired mesh, a mesh denoising algorithm must maintain the object details while removing artificial high-frequencies from the surface. In this work, we propose a two-step algorithm which uses adaptive patches and bilateral filtering to denoise the normal vector field, and then update vertex positions fitting the faces to the denoised normals. The computation of the adaptive patches is our main contribution. We formulate this computation as local quadratic optimization problems that can be controlled by a set of parameters to obtain the desired behavior. We compared our proposal with several algorithms proposed in the literature using synthetic and real data. Our algorithm yields better results in general and is based on a formal mathematical formulation.**

I. Introduction

Nowadays 3D surface models are used in several fields and industries such as medicine, engineering, entertainment, geo-exploration, architecture and cultural heritage. Processes, such as multi-view stereo reconstruction, 3D scanning, 3D imaging, and CAD modeling usually yield triangular meshes. The data generated by these techniques should be processed to be available for production, visualization, simulation, or animation. This processing step is called digital geometry processing which is a field of computer science that uses mathematical models and algorithms [1].

3D surface models obtained from real-world data usually present undesired noise, that can result in problematic effects on later applications. For example, depth-sensing cameras reconstruct noisy surfaces due to the physical limitations of the sensors. As another example, surfaces reconstructed from medical data can present different noise introduced in different steps of the reconstruction process [2].

These models are treated using denoising techniques that seeks to remove high-frequency noise while preserving high-frequency features, such as edges and other details. The denoising step is essential in a typical geometry processing pipeline. Denoising is still a challenging problem because it is difficult to distinguish features from noise.

In a discrete setting, 3D surface models are commonly represented as triangle meshes due to its simplicity and easy processing. The denoising task over meshes is called *mesh denoising*, and it is related to the modification of the geometric properties of the mesh.

In this work, we propose a new algorithm for detail-preserving mesh denoising following a conventional two-step scheme. The first step filters the normal field, and the second one updates vertex positions to adapt them to the filtered normals. The normal field filtering is performed using adaptive patches that represent the neighborhood of a given sample point. The final result of the filtering algorithm strongly depends on the computation of the patches. The main contribution of this work is the determination of these patches. We formulate this determination as a local quadratic optimization problem controlled by parameters that yield flexibility for the proposed algorithm.

We performed several experiments to compare it with other denoising algorithms, using synthetic and real data. The results that we obtained show that our proposal successfully removes the noise while preserving-details, and in most test cases it works better than other methods.

This document is structured as follows. In Section II we present some previous work relevant to our problem. In Section III we explain how we compute the adaptive patches, and how to discretize and implement them. In Section IV we explain how our denoising algorithm works. In Section V we describe experiments and results. Finally, in Section VI we present the conclusions and future work.

II. Previous Work

Based on a diffusion process, numerous anisotropic filters were proposed [3]–[7] extending the idea of anisotropic diffusion of 2D grids to 3D surfaces. Hildebrant and Polthier used a prescribed mean curvature flow simplifying the diffusion process [8]. He and Schaefer proposed a method for sharp features preservation [9] using L_0 minimization.

The bilateral filter for images was an important inspiration for many anisotropic mesh filters. The adaptation of this filter was introduced by Fleishman et al. [10] and Jones et al. [11], and then generalized by Solomon et al. [12]. Two step methods, consisting in normal field filtering followed by vertex updating, were proposed adopting an anisotropic behavior [13]–[15]. Using a bilateral filter for normal field filtering, Zheng et al. proposed an iterative and global scheme for mesh denoising [16]. Wei et al. introduced a bilateral normal filtering using face normals and vertex normals to reach more robustness [17]. Using a guidance signal generated by computing an average normal from consistent patches, Zhang et al. proposed an extension of the joint bilateral filter [18]. Later, Li *et al.* tried to improve the consistent patch definition proposing a new metric [19].

Recently, using binary optimizations, Yadav *et al.* proposed a normal voting tensor to denoise the normal field and then update vertices [20]. Then, the same authors proposed an edge-weighted Laplace operator to avoid face normal flip and to be more robust to high-intensity noise [21]. They use a bilateral normal filtering with a Tukey's bi-weight function as bilateral weighting. Wei *et al.* proposed the usage of consistent neighborhoods, generated from a tensor voting analysis, to compute new vertex positions [22].

Our adaptive patch computation follows the idea of computing consistent patches as shown in [18], [19], and [22], performing a new optimization procedure proposed here. Our denoising algorithm uses these patches to filter the normal field in an iterative manner, including an optional step that performs a bilateral filtering [16] over the new normals to obtain even smoother results.

III. Adaptive Patches Computation as an Optimization Problem

This section presents the mathematical formulation of our proposed algorithm. Here the patch computation is written as a quadratic minimization problem. We first describe the problem as a continuous surface and then we deal with triangular meshes, the discrete form of the surface.

A. Continuous setting

Let X be a 2-manifold embedded in $\mathbb{R}^3$, and a patch X' a subset of X that represents the neighborhood of a reference point $x' \in X$. The patch we seek should adapt to the desired shape to preserve sharp features while denoising flat regions. Using the normal vector field as shape descriptor, we expect that flat regions have low normal variation, and sharp feature or curved regions have higher normal variation. In order to reach adaptation, the patch should be piecewise constant regarding the shape descriptor, so the normal difference between any points within it should be minimum.

Finding the solution X' can be formulated as a quadratic optimization problem, penalizing the error between two points $x_i \in X'$ and $x_j \in X'$ as $q_{ij} = ||n_i - n_j||$, where n_i and n_j are the normals of x_i and x_j respectively. Unfortunately, in a discrete setting, finding a crisp subset X' results in an NP-Hard combinatorial problem. Based on fuzzy set theory [23], we can relax the problem defining a fuzzy membership function $u : X \to [0, 1]$ over the entire domain X. This function defines which is the degree of inclusion of a point $x_i \in X$ to the patch X'. Instead of finding the subset X', now we have to find the function u. The solution $u = 0$ minimizes the problem for all cases using the latter formulation, resulting in a patch with area zero. To avoid this, we add a constraint for u, such that the sum of the area of X weighted by the membership function u should be equal to a fixed value a_0.

To obtain a desired solution for our mesh denoising algorithm we propose the following functional to find a membership function u that:

$$\begin{aligned}\min_u \alpha \int_{x_i \in X} \int_{x_j \in X} q_{ij} u_i u_j da da + \beta \int_{x_i \in X} ||x' - x_i|| u_i da \\ + \gamma \int_X ||\nabla u||^2 da + \delta \int_{x_i \in X} ||n' - n_i|| u_i da \\ s.t. \quad u \in [0, 1] \quad \wedge \quad \int_{x_i \in X} u da = a_0,\end{aligned} \tag{1}$$

where $u_k = u(x_k)$, and n' is the normal of the reference point x'.

The first term in eq. 1 penalizes the difference between the normals of any two points contained in the domain X. The second seeks to enforce compact solutions by penalizing the distance between any point of the patch and the reference point x'. The third term penalizes the squared gradient norm of u to obtain smoother solutions. The fourth term is a little more complicated. Depending on the influence of each of the previous terms we can obtain a solution that does not include the reference point. This solution may have no coherence with the desired reference point normal. For example, if we have a noisy cube and we are computing a patch with a reference point close to an edge, the solution can lie in the wrong face if there the region is flatter. This fourth term allows us to mitigate this problem. The parameters (α, β, γ, δ) in this equation control the importance of each term relative to the others.

B. Discretization

As our denoising algorithm works over triangular meshes, we have two candidates to use as sampled points to represent the manifold: vertex positions and face centroids. We opted to use face centroids because our algorithm uses the normal field generated by face normals. A triangular mesh M can be represented as a set of m vertices $V = \{v_1, \ldots, v_m\}$ and a set of n faces $F = \{f_1, \ldots, f_n\}$. Each face (triangle) is described by the three indices of its vertices. The position of the mesh vertices can be represented as $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ where $\mathbf{x}_i = \mathbf{x}(v_i) = (x(v_i), y(v_i), z(v_i))^T$. Face centroids can be represented as $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$ where $\mathbf{c}_i = \mathbf{c}(f_i) = (x(f_i), y(f_i), z(f_i))^T$. In a similar way, we can represent face normals as $N = \{\mathbf{n}_1, \ldots, \mathbf{n}_n\}$ where $\mathbf{n}_i = \mathbf{n}(f_i) = (n_x(f_i), n_y(f_i), n_z(f_i))^T$. Given a face f defined by the vertices v_1, v_2 and v_3, its corresponding normal can be obtained by $\mathbf{n} = (\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)$. The direction of the normal depends on the face orientation. In order to obtain a coherent normal field, faces must share the same orientation.

If we define the patch as a crisp subset, i.e., u is either 0 or 1, we can represent it as a subset of faces $F' \subseteq F$. In our formulation we define a membership function over this domain (F), so we can represent it as a vector $\mathbf{u} = \{u_1, \ldots, u_n\}^T$ where u_i represents the membership value of face f_i. Given that our sampled points are the face centroids and we want to integrate over the entire domain we need the area correspondent to each point. We assume that the area for each centroid $\mathbf{c}_i$ is the area of the corresponding face, f_i. These areas can be represented

as a vector $\mathbf{a} = \{a_1, \ldots, a_n\}^T$ or as a diagonal matrix $\mathbf{A}$ such that $(a_{ii}) = a_i$.

The first term of the optimization problem has a quadratic form and in a discrete setting can be rewritten as follows: $\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}u_i a_i u_j a_j$. Using a matrix form we have: $\mathbf{u}^T\mathbf{A}^T\mathbf{Q}\mathbf{A}\mathbf{u}$, where $\mathbf{A}$ is the diagonal matrix containing face areas and $\mathbf{Q}$ is an error matrix containing all normal differences between two faces (or centroids). Each entry of $\mathbf{Q}$ is defined by $q_{ij} = \|\mathbf{n}_i - \mathbf{n}_j\|$.

For the term that penalizes the distance between any point to the reference point, we use the distance between centroids and the corresponding face areas. We can consider a $n \times n$ matrix $\mathbf{D}$ containing all distances between two pair of points: $d_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\|$. The distance term is linear, so if the reference point index is k we can use the kth column of $\mathbf{D}$ in the optimization problem. For convenience we will call this vector as $\mathbf{d}$. Also, we have to integrate this penalization over the involved area. Let us denote the area of the reference point as a' (area of the reference face) and the ith entry of $\mathbf{d}$ as d_i. This term can be described by $a'\sum_i^n d_i a_i u_i$, and in a matrix form by $\mathbf{d}^T a'\mathbf{A}\mathbf{u}$.

For the third term we have to define a gradient norm operator in a matrix form. To simplify the operator, we approximate it with the following formulation. We assume that u is constant over all the face (triangle), so the gradient norm is zero within it. For this reason, we only need to integrate the gradient norm over the face edges. Adopting this scheme, for each point over an edge sharing faces f_i and f_j, the gradient should be orthogonal to the edge, and has only two possible directions depending on u values. We can think about it as a 1D gradient, such that the norm of the gradient over an edge point is equal to $|u_i - u_j|$. Following this idea, integrating the gradient norm over all points of the edge, results in the following expression: $\|e_{ij}\||u_i - u_j|$, where $\|e_{ij}\|$ is the corresponding edge length.

With this formulation we can define the gradient norm operator as the following matrix:

$$\mathbf{G} = (g_{ij}) = \begin{cases} \sum_{f_k \in N_f(f_i)} \|e_{ik}\| & i = j \\ -\|e_{ij}\| & f_j \in N_f(f_i) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $N_f(f_i)$ is a set containing edge based neighboring faces of f_i, and $\|e_{ij}\|$ is the lenght of the edge that shares faces f_i and f_j. So, the squared gradient norm of $\mathbf{u}$ can be calculated as follows: $\|\nabla u\|^2 \approx (\mathbf{G}\mathbf{u})^2 = \mathbf{u}^T\mathbf{G}^T\mathbf{G}\mathbf{u}$.

When giving more importance to this term, we obtain more regular solutions, i.e., with a lower variation of u. It is essential to balance the importance of this term to obtain regular solutions and be careful about too smooth solutions which are not helpful for our denoising algorithm, once these solutions tend to be not adapted to the object shape.

The coherence term which penalizes the difference between the normal of a point and the normal of the reference point is linear. So we can discretize it in the same manner as in the distance to the reference point discretization. Let us denote the area of the reference point as a' (area of the reference face) and the normal of the reference point as $\mathbf{n}'$. We can write this term as $a'\sum_i^n \|\mathbf{n}_i - \mathbf{n}'\|a_i u_i$, and in a matrix form as $\mathbf{f}^T a'\mathbf{A}\mathbf{u}$, where $\mathbf{f}$ is a n dimensional vector containing in ith position the normal difference between the reference face and the face f_i.

The lowerbound and upperbound constraints can be represented by the vectors $\mathbf{0}$ and $\mathbf{1}$, which are n-dimensional vectors containing in all their entries zeros and ones respectively. The area constraint results in a single linear constraint $\sum_i^n a_i u_i = a_0$, whose matrix representation is: $\mathbf{a}^T\mathbf{u} = a_0$. In practice, we restrict the domain of the optimization problem to a regular neighborhood limited by a given radius and by a maximum number of faces or variables for the optimization problem (the nearest ones to the reference point). So the mentioned mesh M, represents a subset of the entire mesh we want to denoise.

Considering the parameters that controls the optimization behavior we have the following quadratic optimization problem for each face centroid of the mesh:

$$\min_{\mathbf{u}} \alpha\mathbf{u}^T\mathbf{A}^T\mathbf{Q}\mathbf{A}\mathbf{u} + \beta\mathbf{d}^T a'\mathbf{A}\mathbf{u} + \gamma\mathbf{u}^T\mathbf{G}^T\mathbf{G}\mathbf{u} + \delta\mathbf{f}^T a'\mathbf{A}\mathbf{u} \quad s.t. \quad \mathbf{0} \leq \mathbf{u} \leq \mathbf{1} \wedge \mathbf{a}^T\mathbf{u} = a_0. \quad (3)$$

Factorizing, it leads to a typical quadratic optimization problem with bound constraints and a single linear equality constraint. Unfortunately, this formulation can result in non-convex optimization problems. Non-convex optimization algorithms have a high computational cost. For this reason, we use a local minima solution. The flexibility provided by the parameters allows the formulation to obtain the desired solution.

IV. Denoising Algorithm

Our algorithm deals simultaneously with unit normals and vertex positions. The parameters that weight these contributions are dependent on the size of the object. To simplify matters and establish a feeling for the optimal value of these parameters we apply a uniform scale to the object to transform the average length of the edges to the value one. After the denoising process, we retrieve the original scale. These scale steps are the pre and post-processor, respectively, of our algorithm.

The proposed algorithm consists of four main steps. The first is the computation of all information used to compute the adaptive patches. At first, we compute areas, centroids and normals for all faces. Then, we define the regular neighborhoods used to reduce the domain of optimization problems. These neighborhoods depend on two additional parameters: the maximum Euclidean radius, ρ and a maximum number of variables for the optimization problems, n_{var}. Once we have the regular neighborhoods, we can compute the matrices $\mathbf{d}$, $\mathbf{a}$, $\mathbf{f}$, $\mathbf{A}$, $\mathbf{Q}$, and $\mathbf{G}$ for all faces, as explained in the discretization.

The second step is the computation of the adaptive patches corresponding to each face. We set up the optimization problem terms using the matrices computed in the previous step. To simplify matters, the parameters α, β, γ, and δ are the same to all patches, and the area constraint a_0 is fixed to 20% of the total regular neighborhood area. Then, we use a quadratic

programming solver to find a solution. The result of this step are the membership functions, $\mathbf{u}$.

The third step consists of two normal field filtering substeps, with the second being optional. The first filtering substep uses the membership values of $\mathbf{u}$ as weights to filter normals, so the new normal of a face f_i is computed as follows:

$$\mathbf{n}'_i = \sum_{f_j \in F'} \mathbf{n}_j u_j a_j, \tag{4}$$

where F' is the subset of faces representing the regular neighborhood, u_j is the membership value of face f_j, a_j is the area of face f_j, and $\mathbf{n}'_i$ the new normal. After this computation $\mathbf{n}'_i$ is normalized. We update the normals performing a number of iterations (n_p).

The second optional substep filters the new normals $\mathbf{n}'_i$, following a bilateral scheme introduced by [16]. The main difference here is that this substep uses neighborhoods based on a Euclidean radius instead of topology based neighborhoods. For a face f_i the new normal can be computed as follows:

$$\mathbf{n}'_i = \frac{1}{W(f_i)} \sum_{f_j \in F} a_j K_c(d_j) K_s(||\mathbf{n}_j - \mathbf{n}_i||)\mathbf{n}_j, \tag{5}$$

where F' is the subset of faces representing the regular neighborhood, $W(f_i)$ is the normalization factor, and K_c and K_s are Gaussian kernel functions. We can execute this step in an iterative manner like in the previous case (n_b).

In the last step we fit faces to the filtered normals by adapting vertex positions. Taubin proposed in [13] to use orthogonality between the new normal and the edges of the corresponding face. There are different ways to approximate a solution to this formulation. In this work we adopt the approach of [15], which defines the new vertex position as:

$$\mathbf{x}'_i = \mathbf{x}_i + \frac{1}{|F_v(v_i)|} \sum_{f_k \in F_v(v_i)} \mathbf{n}'_k(\mathbf{n}'_k \cdot (\mathbf{c}_k - \mathbf{x}_i)) \tag{6}$$

where $F_v(v_i)$ represents the set of faces shared by the vertex v_i, $\mathbf{n}'_k$ the new normal of face f_k and $\mathbf{c}_k$ the centroid of f_k. We perform this update iteratively for a given number of iterations (n_v).

We summarize all of these steps in Algorithm 1 which receives as input a noisy mesh and the parameters for denoising, and returns the new vertex positions of the denoised mesh. We can temporarily store matrices in order to avoid excessive space consumption. All normal and vertex updating loops are simultaneously executed. This algorithm can be executed iteratively to obtain smoother results (n_e).

Algorithm 1 Denoising algorithm

procedure DENOISE(X,F,ρ,n_{var},α,β,γ,δ,n_p,n_b,n_v)
  $(A, C, N) \leftarrow$computeAreasCentroidsNormals(X,F)
  computeNeighborhoods(F,C,ρ,n_{var})
  for each $f_i \in F$ **do**
    $(F', \mathbf{u} = \emptyset, \mathbf{d} = \emptyset, \mathbf{a} = \emptyset) \leftarrow$ neighborhood(f_i)
    $(\mathbf{d}, \mathbf{a}, \mathbf{f}, \mathbf{A}, \mathbf{Q}, \mathbf{G}) \leftarrow$ matrixComp(F', A, C, N)
    $\mathbf{H} \leftarrow \alpha\mathbf{A}^T\mathbf{Q}\mathbf{A} + \gamma\mathbf{G}^T\mathbf{G}$
    $\mathbf{b} \leftarrow \beta\mathbf{d}^T a'\mathbf{A} + \delta\mathbf{f}^T a'\mathbf{A}$
    $a_0 \leftarrow 0.2 \cdot$sum($\mathbf{a}$)
    $\mathbf{u} \leftarrow \arg\min_{\mathbf{u}} \mathbf{u}^T\mathbf{H}\mathbf{u} + \mathbf{b}\mathbf{u}$
      $s.t. \quad \mathbf{0} \leq \mathbf{u} \leq \mathbf{1} \wedge \mathbf{a}^T\mathbf{u} = a_0.$
    neighborhood(f_i) $\leftarrow (F', \mathbf{u}, \mathbf{d}, \mathbf{a})$
  for $it \leftarrow 1$ to n_p **do**
    for each $f_i \in F$ **do**
      $(F', \mathbf{u}, \mathbf{d}, \mathbf{a}) \leftarrow$ neighborhood(f_i)
      $\mathbf{n}_i \leftarrow$ normalize$\left(\sum_{f_j \in F'} \mathbf{n}_j u_j a_j\right)$
  for $it \leftarrow 1$ to n_b **do** ▷ Optional Step
    for each $f_i \in F$ **do**
      $(F', \mathbf{u}, \mathbf{d}, \mathbf{a}) \leftarrow$ neighborhood(f_i)
      $\mathbf{n}_i \leftarrow \frac{1}{W(f_i)} \sum_{f_j \in F'} a_j K_c(d_j) K_s(||\mathbf{n}_j - \mathbf{n}_i||)\mathbf{n}_j$
  for $it \leftarrow 1$ to n_v **do**
    for each $v_i \in V$ **do**
      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \frac{1}{|F_v(v_i)|} \sum_{f_k \in F_v(v_i)} \mathbf{n}'_k(\mathbf{n}'_k \cdot (\mathbf{c}_k - \mathbf{x}_i))$
  return $\mathbf{X}$

V. Results

Denoising algorithms are usually evaluated by measuring the similarity between the resulting output and the desired output (ground truth). To test the algorithms we used irregular meshes with sharp features, large flat areas and large rounded areas.

To measure the distance between meshes we use four different quantities: distance, normal, curvature and volume. Distance-based: L^2 vertex-based mesh-to-mesh error metric (L^2VBE) [24], [25]. Normal based: L^2 normal-based mesh-to-mesh error metric (L^2NBE) [24], [25]. Curvature based: Discrete mean curvature error metric (DCE) [26]. Volume based: volume error ratio (VE). We can also visually evaluate the results of denoising algorithms rendering the resulting meshes using flat shading.

We implemented the proposed algorithm in C++ programming language, using the half-edge data structure contained in OpenMesh library [27]. To solve the quadratic optimization problems we used the CPLEX library[1]. All our experiments were performed on an Intel (R) Core (TM) i7-4770 CPU @ 3.40GHz processor with 16,0 GB RAM and Windows 8.1 64-bit operating system.

The meshes used in the experiments are from the AIM@SHAPE Shape Repository [28], the SHREC15: Range Scans based 3D Shape Retrieval [29] and The Stanford 3D Scanning Repository[2]. The parameters of other algorithms used in our test cases were provided in the corresponding work. When the test case was not presented in the referenced paper, we manually adjust the parameters following the authors' recommendations. The parameters used for our work are shown in Table I. We fixed $\rho = 2.0$ and $n_{var} = 100$.

A. Evaluation of the proposed denoising algorithm

The first example is the sharpSphere mesh corrupted with artificial Gaussian noise. We try to eliminate the noise using

[1]https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer

[2]http://graphics.stanford.edu/data/3Dscanrep/

TABLE I
PARAMETERS USED FOR OUR PROPOSAL.

mesh	$(\alpha, \beta, \gamma, \delta, n_e, n_p, n_b, n_v)$
SharpSphere	(1.0, 2.0, 0.8, 8, 3, 3, 5, 10)
Dragon	(1.0, 1.0, 0.2, 20, 1, 5, 3, 10)
Block	(1.0, 1.0, 0.1, 30, 3, 5, 2, 10)
Fandisk	(1.0, 1.0, 0.2, 10, 3, 5, 2, 10)
Joint	(1.0, 2.0, 0.3, 30, 3, 6, 2, 10)
Balljoint	(1.0, 1.0, 0.1, 5, 2, 3, 2, 10)
Gargoyle	(1.0, 1.0, 0.2, 10, 2, 5, 1, 10)
Keyboard	(1.0, 1.0, 0.2, 20, 2, 3, 1, 10)

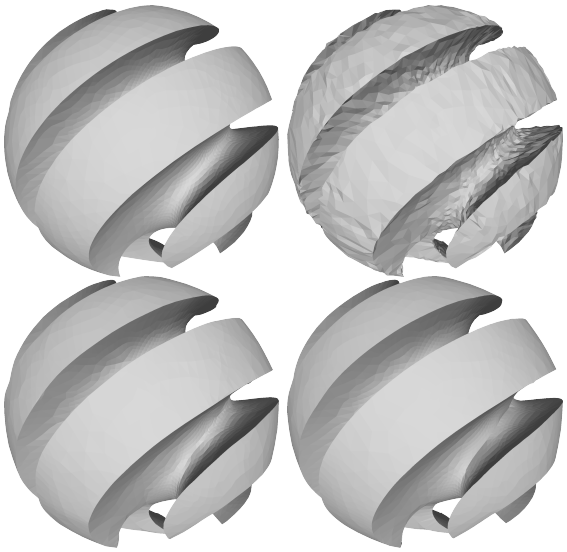

Fig. 1. From left to right and top to bottom: original, noisy mesh, result of our proposal without using bilateral normal filtering, and result of our proposal using it.

our algorithm with and without the optional bilateral normal filtering step. Figure 1 shows the corresponding results.

Table II shows the error between the original mesh, not corrupted by noise, and the resulting from our algorithm with and without the optional step. As we can see, our algorithm successfully removes the noise while preserving details. The optional bilateral normal filtering step generates a smoother mesh but yields higher errors. These errors, however, are not significant and, depending on the application, we may choose to use the optional step.

TABLE II
ALGORITHM RESULTS FOR SHARPSPHERE MESH

	L^2VBE	L^2NBE	DCE	VE
With Bilateral	0.003932	0.036556	0.085570	0.001526
Without Bilateral	0.001673	0.012719	0.076498	0.000364

In [18] the authors proposed the computation of consistent patches to obtain a guidance normal field that in principle is similar to our weighted average normals. To compare both approaches, we use our algorithm and the implementation provided by [18] over the same corrupted mesh. Both normal fields are compared them using the L^2 Normal Based metric

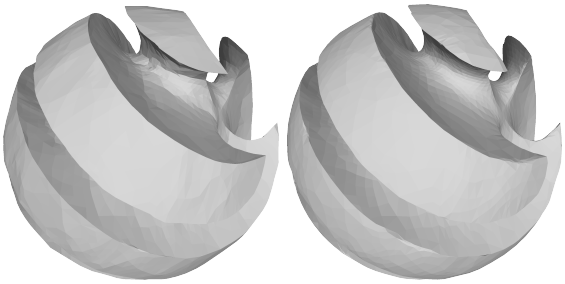

Fig. 2. Results after 20 iterations of vertex updating using estimated normals. Left: Guided normals using [18]. Right: Average normals weighted by patch membership function.

where the guidance normal field error results in 0.01603 and our weighted average normal filed in 0.00816. To show these results visually, we performed 20 iterations of vertex updating step using both normal fields. Figure 2 shows the resulting meshes. Our estimated normal field have a lower error than the guidance normal field of [18], i.e., our estimated normal field is closer to the ground truth. Moreover, our approach generates less flat regions when the triangulation is very irregular.

Table III presents the efficiency of each step of the proposed algorithm based on the Dragon mesh of The Stanford 3D Scanning Repository. The low-resolution meshes are obtained by decimation and for comparison we use the same parameters for all the meshes. We fixed $n_{var} = 20$ for this case.

TABLE III
EXECUTION TIME

Mesh	Step 1	Step 2	Step 3	Step 4	Total
10K	0.724s	6.884s	0.070s	0.012s	7.690s
25K	1.744s	16.862s	0.161s	0.019s	18.786s
50K	3.298s	33.685s	0.283s	0.039s	37.305s
75K	4.703s	51.401s	0.434s	0.060s	56.598s
100K	9.287s	67.837s	0.573s	0.087s	77.784s

The computation of the adaptive patches is the step that consumes more time in our algorithm. Also, the function that describes the execution time of this step with respect to the number of triangles, has a linear growth.

B. Comparison with other algorithms

The meshes used here are "Block", "Fandisk" and "Joint" whose vertex positions are corrupted with artificial Gaussian noise. In order to simulate an arbitrary noise source, the displacement direction is defined by the vertex normal or by a random direction. Block: noise intensity using $\sigma = 0.1l$ and following vertex normal directions. Fandisk: noise intensity using $\sigma = 0.3l$ and following random directions. Joint: noise intensity using $\sigma = 0.35l$ and following vertex normal directions. In all cases l is equal to the average edge length.

Our algorithm is focused on denoising of meshes with a reasonable amount of noise. If, however, the noise is too high, the normal field which is the base of our computation can vary substantially and the optimization procedure may yield bad results. Therefore, for data with a small signal to

noise ratio, we preprocess the mesh using a few iterations of bilateral normal filtering [16]. In the case of artificial noise, we recommend using this pre-filtering step when the noise intensity is based on $\sigma > 0.2l$. We used three iterations of the bilateral normal filtering to initialize the Fandisk and Joint meshes.

Figure 3 and Table IV show the results for Block. Figure 4 and Table V show the results for Fandisk. Finally, Figure 5 and Table VI show the results for Joint.

The algorithms proposed by [15] and [16] generate smooth meshes while preserving sharp features. They do not preserve small flat regions when the data has round areas. The method presented by [9] tries to preserve these small flat regions, but it introduces larger ones, as undesired artifacts. The paper [18] introduces wrong normals in the presence of mesh irregularities in the regions of sharp features. The algorithms proposed by [20] and [21] blur the edges with a low dihedral angle. Our method preserves the details while removing noise, resulting in low errors.

TABLE IV
RESULTS FOR BLOCK MESH

	L^2VBE	L^2NBE	DCE	VE
[15]	0.011088	0.004683	0.029590	0.000300
[16]	0.010192	0.004298	0.029038	0.000210
[9]	0.020213	0.024389	0.109556	0.024038
[18]	0.004180	0.002676	0.028262	0.000849
Ours	0.001280	0.002001	0.029958	0.000433

TABLE V
RESULTS FOR FANDISK MESH

	L^2VBE	L^2NBE	DCE	VE
[15]	0.000364	0.014546	0.138360	0.000733
[16]	0.000213	0.008664	0.131381	0.001007
[9]	0.000630	0.028042	0.197488	0.017092
[18]	0.000117	0.007719	0.113402	0.000049
[20]	0.000191	0.009118	0.122689	0.001656
[21]	0.000185	0.006311	0.097299	0.000840
Ours	0.000156	0.005791	0.116578	0.000813

TABLE VI
RESULTS FOR JOINT MESH

	L^2VBE	L^2NBE	DCE	VE
[15]	2.43e-6	0.002508	0.227227	0.000115
[16]	1.11e-6	0.001493	0.201129	0.000152
[9]	1.34e-5	0.005773	0.218453	0.002809
[18]	1.48e-6	0.001645	0.154981	0.000215
[20]	1.29e-6	0.000756	0.113735	0.000223
[21]	1.06e-6	0.000882	0.160960	0.000583
Ours	1.04e-6	0.000601	0.180188	0.000066

C. Denoising of real objets

This section studies the denoising of the "Gargoyle", "BallJoint" and "Keyboard" meshes. Figures 6, 7 and 8 show the corresponding results. In general, [9] removes the noise but also removes small details and generates undesired flat regions. [15], [16] and [18] smooth too much the shape removing some sharp features, yielding blurred results. [20] does not preserve the continuity of sharp features and introduces round edges. Our results better preserve these details and generate thinner regions with high curvature yielding in a well-defined mesh.

D. Denoising meshes generated from ultrasound exams

Here we present the result of our denoising algorithm over two meshes generated from medical ultrasound data. The meshes were obtained from fetal ultrasound exams using a two-step methodology. The first step uses an active contour-based segmentation to capture the shape of the fetus. The second reconstructs the mesh using the Marching Cubes algorithm [30]. Both meshes present the staircase artifact which is a common mesh distortion when reconstructing isosurfaces of slice-based volumes.

We perform the denoising of these meshes in two steps. First we remove the staircase artifact using our algorithm with the following parameters: ($\alpha = 1.0$, $\beta = 1.0$, $\gamma = 0.2$, $\delta = 0$, $n_e = 1$, $n_p = 2$, $n_b = 0$, $n_v = 10$). Then we remove the noise using our algorithm with the following parameters: ($\alpha = 1.0$, $\beta = 1.0$, $\gamma = 0.2$, $\delta = 15$, $n_e = 2$, $n_p = 3$, $n_b = 5$, $n_v = 10$). Figure 9 shows our results. Staircase artifacts and noise are removed while most of the details that can be distinguished in the noisy meshes are preserved.

VI. CONCLUSION AND FUTURE WORK

The numerical results presented here indicate that the proposed algorithm removes the noise while preserving the details better than the algorithms we analyze in this work. When the ground truth is available, the error of our algorithm is the lowest one in most cases. Our results with real data, also present a better definition of the details while removing the noise.

The result of our method is dependent on the choice of the optimization parameters, this presents some complexity but also allows for great flexibility. The denoising algorithm can have multiple behaviors when tuning them. For example, if we want to denoise a smooth object like a human shape, we can increase the value of the parameter γ to obtain smoother patches that result in a near anisotropic Gaussian filter, which is a good choice for this kind of objects. In the case an object has large flat regions, we can decrease the value of γ and increase the values of α and β to obtain more compact patches with low normal variation.

The proportion between a parameter of a quadratic term and a parameter of a linear term strongly depends on the scale of the mesh. The normalization step, however, solves this problem.

The computational time of our algorithm has a near-linear cost but with high constant time for the number of triangles in the local neighborhood. The limitation of the number of variables allows us to define our algorithm in this way. The quadratic optimization problems are the heaviest operations in the pipeline, but we were able to solve common problems in a reasonable time.

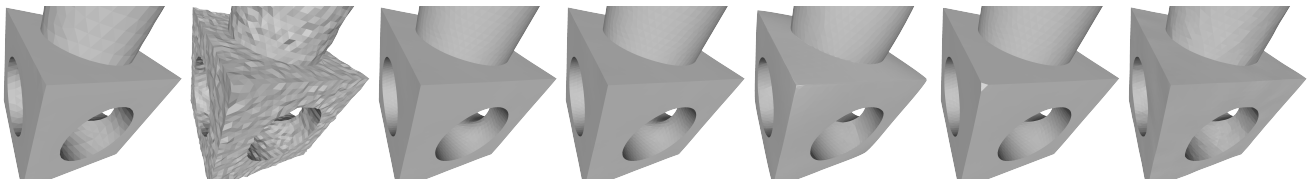

Fig. 3. Results obtained for Block mesh. From left to right: original, noisy, [15], [16], [9], [18] and our method.

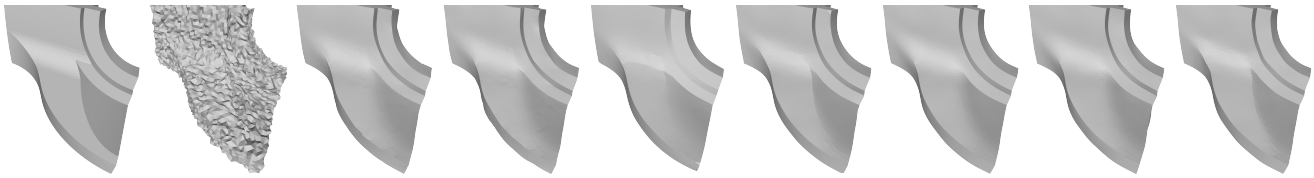

Fig. 4. Results obtained for Fandisk mesh. From left to right: original, noisy, [15], [16], [9], [18], [20], [21] and our method.

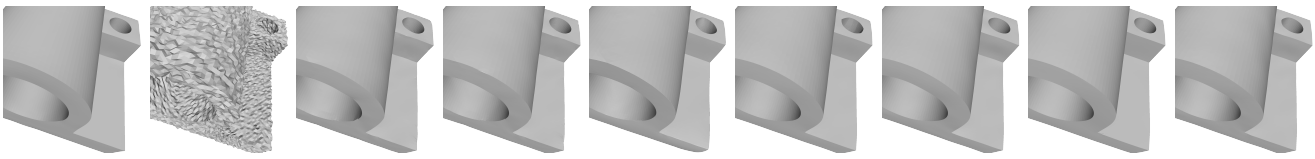

Fig. 5. Results obtained for Joint mesh. From left to right: original, noisy, [15], [16], [9], [18], [20], [21] and our method.

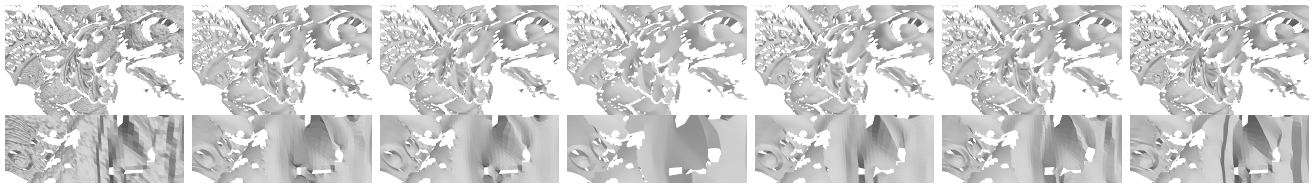

Fig. 6. Results obtained for Gargoyle model. From left to right: noisy, [15], [16], [9], [18], [20] and our method.

Fig. 7. Results obtained for Ball Joint model. From left to right: noisy, [15], [16], [9], [18], [20] and our method.

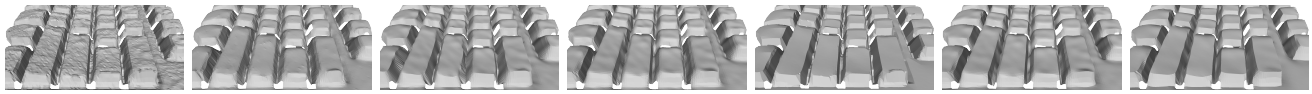

Fig. 8. Results obtained for Keyboard model. From left to right: noisy, [10], [15], [16], [9], [18], and our method.

Our approach was focused on denoising of triangular meshes, using face centroids as sampled points of the represented 2-manifold. As future work, we can use the vertex position as sampled points and perform the optimization problems on this domain instead of the face based domain. The vertex based domain was better studied in the literature and has more accurate approximations (e.g., gradient norm operator). If we work on this domain, we can extend our proposal to 3D point clouds.

We found that our adaptive patches based on normal fields can be used in other applications like mesh segmentation, remeshing or feature detection. Also, instead of using normal fields to describe the data, we can use other descriptors like curvature, saliency or heat kernels. We hope to address other applications in later work.

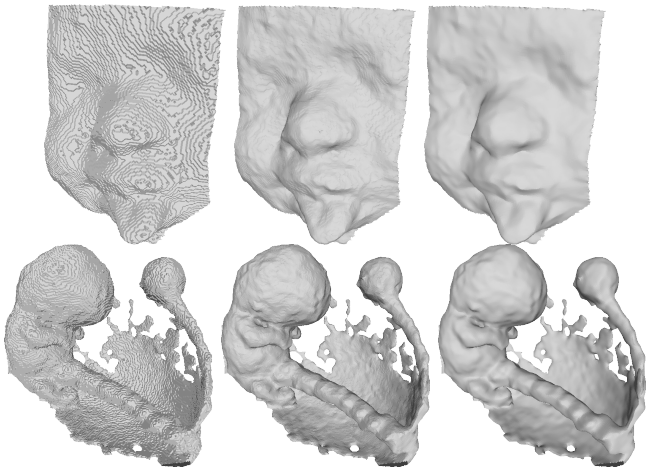

Fig. 9. Denoising of fetus meshes with staircase artifact using our method. First column: noisy mesh. Second column: resulting mesh after staircase removal. Third column: denoised mesh.

ACKNOWLEDGMENT

We thank the support provided by National Council for Scientific and Technological Development - CNPq and Coordination for the Improvement of Higher Education Personnel - CAPES.

REFERENCES

[1] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*, ser. Ak Peters Series. Taylor & Francis, 2010. [Online]. Available: https://books.google.com.br/books?id=8zX-2VRqBAkC

[2] R. Bade, J. Haase, and B. Preim, "Comparison of fundamental mesh smoothing algorithms for medical surface models," in *In Simulation und Visualisierung (2006*, 2006, pp. 289–304.

[3] Y. Ohtake, A. G. Belyaev, and I. A. Bogaevski, "Polyhedral surface smoothing with simultaneous mesh regularization," in *Geometric Modeling and Processing 2000. Theory and Applications. Proceedings*. IEEE, 2000, pp. 229–237.

[4] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Anisotropic feature-preserving denoising of height fields and bivariate data." in *Graphics interface*, vol. 11, no. 10. Citeseer, 2000, pp. 145–152.

[5] U. Clarenz, U. Diewald, and M. Rumpf, "Anisotropic geometric diffusion in surface processing," in *Proceedings of the conference on Visualization'00*. IEEE Computer Society Press, 2000, pp. 397–405.

[6] C. L. Bajaj and G. Xu, "Anisotropic diffusion of surfaces and functions on surfaces," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 1, pp. 4–32, 2003.

[7] A. F. El Ouafdi and D. Ziou, "A global physical method for manifold smoothing," in *2008 IEEE International Conference on Shape Modeling and Applications*, 2008.

[8] K. Hildebrandt and K. Polthier, "Anisotropic filtering of non-linear surface features," in *Computer Graphics Forum*, vol. 23, no. 3. Wiley Online Library, 2004, pp. 391–400.

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

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

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

[12] J. Solomon, K. Crane, A. Butscher, and C. Wojtan, "A general framework for bilateral and mean shift filtering," *CoRR*, vol. abs/1405.4734, 2014. [Online]. Available: http://arxiv.org/abs/1405.4734

[13] G. Taubin, "Linear anisotropic mesh filtering," *Res. Rep. RC2213 IBM*, vol. 1, no. 4, 2001.

[14] Y. Shen and K. E. Barner, "Fuzzy vector median-based surface smoothing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 3, pp. 252–265, May 2004. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2004.1272725

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

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

[17] M. Wei, J. Yu, W.-M. Pang, J. Wang, J. Qin, L. Liu, and P.-A. Heng, "Bi-normal filtering for mesh denoising," *IEEE transactions on visualization and computer graphics*, vol. 21, no. 1, pp. 43–55, 2015.

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

[19] T. Li, J. Wang, H. Liu, and L.-g. Liu, "Efficient mesh denoising via robust normal filtering and alternate vertex updating," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 11, pp. 1828–1842, 2017.

[20] S. K. Yadav, U. Reitebuch, and K. Polthier, "Mesh denoising based on normal voting tensor and binary optimization," *IEEE Transactions on Visualization & Computer Graphics*, no. 8, pp. 2366–2379, 2018.

[21] ——, "Robust and high fidelity mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, 2018.

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

[23] L. A. Zadeh, "Fuzzy sets," in *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*. World Scientific, 1996, pp. 394–432.

[24] 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*, 2002, pp. 124–131.

[25] A. Belyaev and Y. Ohtake, "A comparison of mesh smoothing methods," in *Israel-Korea Bi-national conference on geometric modeling and computer graphics*, vol. 2. Citeseer, 2003.

[26] S.-J. Kim, S.-K. Kim, and C.-H. Kim, "Discrete differential error metric for surface simplification," in *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.*, 2002, pp. 276–283.

[27] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, "Openmesh - a generic and efficient polygon mesh data structure," 2002.

[28] "Aim@shape shape repository," http://visionair.ge.imati.cnr.it, accessed: 2017-05-01.

[29] A. Godil, H. Dutagaci, B. Bustos, S. Choi, S. Dong, T. Furuya, H. Li, N. Link, A. Moriyama, R. Meruane *et al.*, "Shrec'15: Range scans based 3d shape retrieval," 2015.

[30] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987. [Online]. Available: http://doi.acm.org/10.1145/37402.37422