Kernel-Based Laplacian Smoothing Method for 3D Mesh Denoising

Hicham Badri[1], Mohammed El Hassouni[1,2], and Driss Aboutajdine[1]

[1] LRIT, Faculty of Science,
[2] DESTEC-FLSHR,
University Mohammed V -Agdal- Rabat, Morocco
{Hichammbadri,Mohamed.Elhassouni}@gmail.com, aboutaj@fsr.ac.ma

Abstract. In this paper, we present an improved Laplacian smoothing technique for 3D mesh denoising. This method filters directly the vertices by updating their positions. Laplacian smoothing process is simple to implement and fast, but it tends to produce shrinking and oversmoothing effects. To remedy this problem, firstly, we introduce a kernel function in the Laplacian expression. Then, we propose to use a linear combination of denoised instances. This combination aims to reduce the number of iterations of the desired method by coupling it with a technique that leads to oversmoothing. Experiments are conducted on synthetic triangular meshes corrupted by Gaussian noise. Results show that we outperform some existing methods in terms of objective and visual quality.

1 Introduction

Denoising is one of the greatest challenges in image processing and computer graphics. Fast and efficient algorithms are needed to recover noised data (images and 3D models) while preserving their geometrical structure. Measurements are perturbed by noise in all real applications. Notably, 3D models acquisition using scanners. These scanners provide the real scanned object as a 3D digital mesh, usually represented as a triangular mesh, that can be manipulated by any 3D processing tool, for many purposes in various fields such as medical imaging, video games, etc ...

In recent years, various partial differential equations (PDE)-based techniques have been used for 2D image denoising such as the anisotropic diffusion proposed by P. Perona et al. [7]. 2D image denoising techniques were adapted for 3D mesh denoising. Taubin [8] proposed a Laplacian-based technique called Laplacian flow that repeatedly adjusts the location of each vertex to the geometric center of its vertex neighborhood. This technique is quite simple and fast but produces an oversmoothing result. Many anisotropic diffusion methods were proposed such as the diffusion and curvature flow technique presented in [9], this method gives better results than the Laplacian flow technique but takes more processing time.

Y. Zhang et al. proposed an efficient diffusion technique [1] based on solving a nonlinear discrete partial differential equation. M. El Hassouni et al. improved

A. Elmoataz et al. (Eds.): ICISP 2012, LNCS 7340, pp. 77–84, 2012.

this technique [2] by using other diffusion functions such as Laplacian, Reduced Centered Gaussian and Rayleigh function instead of the Cauchy function.

In this article, we propose a vertex-based method for 3D mesh denoising. The main idea is to reduce the smoothing effect by introducing a kernel function in the Laplacian flow expression. Then we present a linear combination of denoised instances in order to reduce the number iterations of the desired technique by combining it with a fast (oversmoothing) technique like the local averaging method. Experimental results show that the proposed work gives competitive results in comparison with existing methods while reducing the processing time.

This paper is organized as follows : Section 2 presents the problem formulation. Section 3 describes the Laplacian smoothing method. In section 4, we present the proposed work. Section 5 deals with experimental results. Finally, we give some concluding remarks in Section 6.

2 Problem Formulation

A 3D object is usually presented as polygonal or triangular mesh. A triangle mesh $\mathbb{M}$ denotes a triple $\mathbb{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$ where $\mathcal{V} = \{v_1, ..., v_k\}$ represents the set of vertices, $\mathcal{E} = \{e_{ij}\}$ denotes the set of edges and $\mathcal{T} = \{t_1, ..., t_n\}$ denotes the set of triangles. An edge e_{ij} consists of two vertices $\{v_i, v_j\}$ and we say that two vertices $v_i, v_j \in \mathcal{V}$ are adjacent if the are connected by an edge $e_{ij} \in \mathcal{E}$ (and we write $v_i \sim v_j$). We define the neighborhood of a vertix v_i, the set of adjacent vertices $v_i^* = \{v_j \in \mathcal{V} : v_i \sim v_j\}$. The degree of a vertex v_i is the number of the neighboring vertices $d(i) = |\, v_i^*|$. We denote by t_i^* the set of all triangles sharing a vertex or an edge with a triangle $t_i \in \mathcal{T}$ and by $\mathcal{T}(v_i^*)$ the set of triangles of the neighborhood v_i^*.

We denote by $n(t_j)$, with $t_j \in t_i^*$ a triangle, the unit normal of t_j and by n_i the normal at a vertex v_i given by the following formula :

$$n_i = \frac{1}{d_i} \sum_{t_j \in \mathcal{T}(v_i^*)} n(t_j) \tag{1}$$

We denote by $A(t_j)$ the area of the triangle t_j. The mean edge length $\overline{l}$ of the mesh is given by :

$$\overline{l} = \frac{1}{|\mathcal{E}|} \sum_{e_{ij} \in \mathcal{E}} \| \, e_{ij} \, \| \tag{2}$$

Where

$$\begin{cases} \| \, e_{ij} \, \| = \| \, v_i - v_j \, \| & if \; v_i \sim v_j \\ \| \, e_{ij} \, \| = 0 & otherwise \end{cases}$$

Measurements are perturbed by noise (supposed additive in our case) in all real applications. This can be formulated by :

$$\hat{v} = v + \eta \tag{3}$$

Where $\hat{v}$ is the observed vertex, v is the original one and η is a random noise process assumed to be Gaussian.

3 Laplacian Smoothing

Laplacian Smoothing is a very simple PDE-based smoothing approach formulated as follows [8]:

$$v_i \leftarrow v_i + \sum_{v_j \in v_i^*} \left(\frac{v_j - v_i}{d_i} \right) \tag{4}$$

This process can be done repeatedly to correct the location of each vertex to the geometric center of its neighboring vertices. This approach is simple and fast, however, it produces an oversmoothing result after few iterations.

Note that the *Laplacian Smoothing* is just a special case of the *Weighted Laplacian Filter* [11], also called *Local Averaging*:

$$v_i \leftarrow v_i + \frac{1}{\sum_{v_j \in v_i^*} w_{ij}} \sum_{v_j \in v_i^*} w_{ij}(v_j - v_i) \tag{5}$$

Where w_{ij} are the weights defined as :

$$w_{ij} = \begin{cases} > 0 & if\ v_j \in v_i^* \\ 0 & otherwise \end{cases}$$

For $w_{ij} = 1$ if $v_j \in v_i^*$ we retrieve the *Laplacian Smoothing* update rule, $\sum_{v_j \in v_i^*} w_{ij} = d_i$.

4 Proposed Method

In this section, we present at first an improved version of the Laplacian technique by introducing a kernel function. Then, we propose a linear combination of two denoised instances in order to reduce the number of iterations.

4.1 Kernel Based Laplacian Smoothing

The proposed approach consists in introducing a kernel function g to attenuate the added term, hence, overcome the oversmoothing problem.

The update rule is given by :

$$v_i \leftarrow v_i + \sum_{v_j \in v_i^*} \left(\frac{v_j - v_i}{d_i} \right) \left(\frac{g(|\nabla v_i|)}{\rho} \right) \tag{6}$$

Where

$$|\nabla v_i| = \left(\sum_{v_j \in v_i^*} \| \frac{v_i}{\sqrt{d_i}} - \frac{v_j}{\sqrt{d_j}} \|^2 \right)^{1/2} \tag{7}$$

g is a kernel function and ρ is a parameter to estimate, it can be either a scalar or a 3D vector (x,y,z). In this paper, we are going to use ρ as a scalar.
Another update rule can also be used, inspired by [1]:

$$v_i \leftarrow v_i + \sum_{v_j \in v_i^*} \left(\frac{v_j - v_i}{d_i} \right) \left(\frac{g(|\nabla v_i|) + g(|\nabla v_j|)}{\rho} \right) \tag{8}$$

The same functions in [2] can be used as a kernel function. Note that for the following ρ value applied to the equation (8), we retrieve the formula presented in [1] :

$$\begin{cases} \rho = \frac{\sqrt{d_i d_j}}{d_i + \sqrt{d_i d_j} + \beta} \\ \beta = \frac{-v_j \sqrt{d_i d_j} + v_i d_i}{v_j - v_i} \end{cases}$$

The update rule (6) is the one that will be evaluated in this paper.

4.2 Linear Combination

The following method consists in combining 2 techniques using the following linear formula :

$$\widetilde{\mathbb{M}} = \alpha \widetilde{\mathbb{M}_1} + (1 - \alpha) \widetilde{\mathbb{M}_2}, \qquad \alpha \in [0, 1] \tag{9}$$

Where $\widetilde{\mathbb{M}_1}$ is the denoised mesh with the method 1 and $\widetilde{\mathbb{M}_2}$ and denoised mesh with the method 2.

This technique can be used to reduce the number of iterations of the desired technique by coupling it with a fast method that leads quickly to oversmoothing (like the Local Averaging).

The only combining approach that will be evaluated in this paper is the Local Averaging.The version used is the same one in [10]. Other combinations may give better results.

5 Experimental Results

This section presents simulation results where the proposed method is applied to 3D models contaminated by an additive zero-mean Gaussian noise.
For ease to use, we developed a simple Graphical User Interface using Matlab/Java and the toolbox graph [10]. The 3D objects used in this section are presented in Figure 1. To quantify the performance of the proposed method in comparison with other 3D mesh denoising techniques, we compute the face-normal error metric [4] given by :

$$E_{fne} = \frac{1}{A(\hat{\mathbb{M}})} \sum_{\hat{t_i} \in \hat{T}} A(\hat{t_i}) \parallel n(t_i) - n(\hat{t_i}) \parallel$$

Where $n(t_j)$ and $n(\hat{t}_i)$ are the unit normals, $A(\hat{t}_i)$ is the area of the triangle $\hat{t}_i$ and $A(\hat{\mathbb{M}})$ is the total area of the mesh defined by the following formula :

$$A(\hat{\mathbb{M}}) = \sum_{\hat{t}_i \in \hat{\mathcal{T}}} A(\hat{t}_i)$$

We also use the visual error metric given by :

$$\begin{cases} E_{ve} = \frac{1}{2m} \left(\sum_{i=1}^{m} \parallel v_i - \hat{v}_i \parallel^2 + \sum_{i=1}^{m} \parallel I(v_i) - I(\hat{v}_i) \parallel^2 \right) \\ I(v_i) = v_i - \frac{1}{d_i} \sum_{v_j \in v_i^*} v_j \end{cases}$$

This metric is computed on mesh vertices.

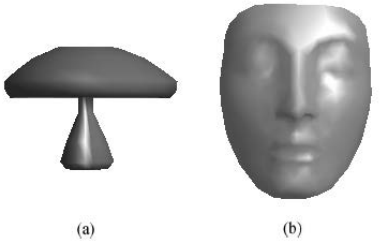

Fig. 1. 3D meshes used for experimentation : (a) mushroom (226 vertices), (b) nefertiti (299 vertices)

For all methods, parameters have been tuned experimentally in order to get the best trade-off between the visual error and the face-normal error for each technique. Also, We choose to report results only for one Kernel function and one noise level for each processed object. Here, use Laplacian as a Kernel function for all experiments with ($c = 2$).

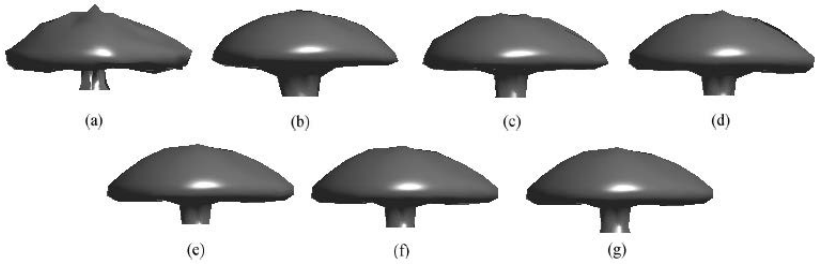

Fig. 2. Mesh denoising results for the *mushroom* instance. (a) Noisy 3D model $\sigma^2 = 0.0015$, (b) Laplacian smoothing (1 iteration), (c) Local Averaging (1 iteration), (d) Improved vertex-based diffusion (5 iterations), (e) Kernel-based Laplacian (1 iteration, $\rho = 0.65$) , (f) Linear combination : Local Averaging (1 iteration) + improved vertex-based diffusion, (3 iterations, $\alpha = 0.53$), (g) Linear-combination : Local averaging (1 iteration) + kernel-based Laplacian (3 iterations, $\alpha = 0.58$, $\rho = 2$)

Figures 3 and 5 show the visual errors and Face-normal error for compared denoising methods. We remark that proposed method (kernel based Laplacian (e)) outperforms the standard Laplacian smoothing (b) and the improved vertex-based diffusion (d). Note that only one iteration is sufficient in this case, while 5 iterations were needed for the improved vertex-based diffusion. According to these graphs, linear combination gives better results for the improved vertex-based diffusion (f) while reducing the number of iterations. Combining proposed method and local averaging improves slightly the denoising performance.

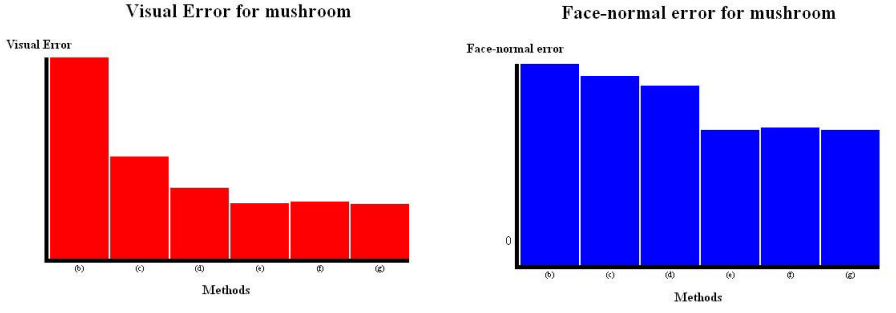

Fig. 3. Visual Error and Face-normal error for the *mushroom* object

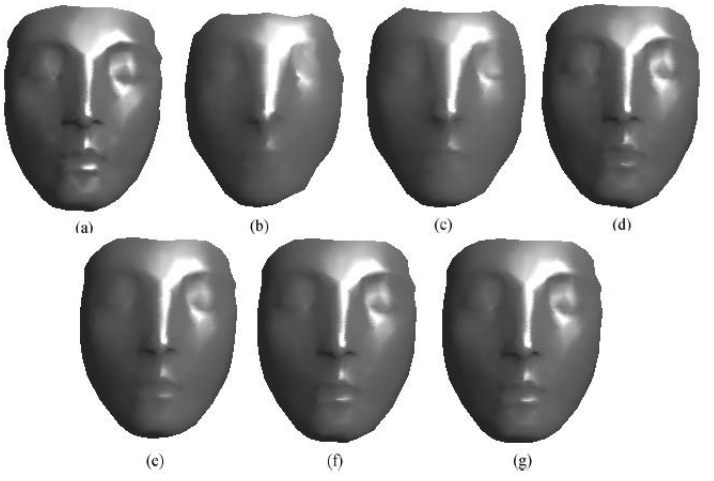

Fig. 4. Mesh denoising results for the *nefertiti* instance. (a) Noisy 3D model $\sigma^2 = 0.001$, (b) Laplacian smoothing (1 iteration), (c) Local Averaging (1 iteration), (d) Improved vertex-based diffusion (3 iterations), (e) Kernel-based Laplacian (1 iteration, $\rho = 0.72$) , (f) Linear-combination : Local Averaging (1 iteration) + improved vertex-based diffusion (1 iteration, $\alpha = 0.3$), (g) Linear-combination : Local averaging (1 iteration) + kernel based Laplacian (1 iteration, $\alpha = 0.28$, $\rho = 3$)

Figure 2 and 4 are given to assess the visual impact of the denoising. We can see that for both case the Kernel Based Laplacian processed objects exhibit a more appealing visual appearance.

Note that Local Averaging using one iteration was the only technique used in the linear combination scheme. Other combinations may give better results.

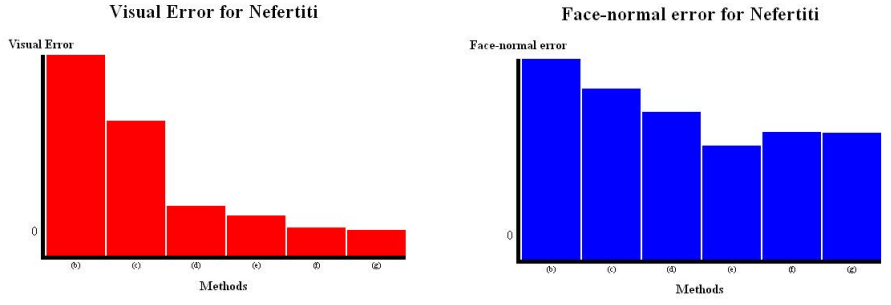

Fig. 5. Visual Error error and Face-normal error for the *nefertiti* object

6 Conclusion

We presented in this paper an kernel-based Laplacian smoothing method for 3D mesh denoising. The main idea is to reduce the smoothing effect by introducing a kernel function. Then we proposed a linear combination of denoised instances by different techniques. This method can be used to reduce the number of iterations in iterative denoising techniques of the desired method by combining it with a very fast (oversmoothing) technique like the local averaging method. Experimental results showed that the kernel-based Laplacian method proposed outperforms the standard Laplacian technique and the improved vertex-based diffusion [2] while using only 1 or 2 iterations. The linear-combination technique gave also good results while reducing the number of iterations. Using other techniques instead of the Local Averaging may give better results.

References

[1] Zhang, Y., Ben Hamza, A.: Vertex-Based Diffusion for 3-D Mesh Denoising. IEEE Transactions on Image Processing 16(4) (April 2007)
[2] El Hassouni, M., Aboutajdine, D.: 3D-Mesh denoising using an improved vertex based anisotropic diffusion. International Journal of Computer Science and Information Security (IJCSIS) 8(2) (2010)
[3] Field, D.: Laplacian smoothing and Delauney triangulations. Comm. App. Num. Meth. 4, 709–712 (1988)
[4] Taubin, G.: Linear Anisotropic Mesh Filtering. Res. Rep. RC2213 IBM (2001)
[5] Karni, Z., Gotsman, C.: Spectral compression of mesh geometry. In: Proc. SIGGRAPH, pp. 279–286 (2000)

[6] Mashiko, T., Yagou, H., Wei, D., Ding, Y., Wu, G.: 3D Triangle Mesh Smoothing via Adaptive MMSE Filtering. In: Proceedings of the Fourth International Conference on Computer and Information Technology (CIT 2004), pp. 734–740 (2004)
[7] Perona, P., Malik, J.: Scale space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(7), 629–639 (1990)
[8] Taubin, G.: A signal processing approach to fair surface design. In: Proc. SIGGRAPH, pp. 351–358 (1995)
[9] Desbrun, M., Meyer, M., Schroder, P., Barr, A.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proc. SIGGRAPH, pp. 317–324 (1999)
[10] http://www.ceremade.dauphine.fr/~peyre/numerical-tour/tours/meshproc_3_denoising/
[11] Desbrun, M., Meyer, M., Schröder, P., Barr, A.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proc. SIGGRAPH, pp. 317–324 (1999)