Reading Report of ”Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow”

Yucong Zheng

Abstract

This report is a summary of implicit fairing technology introduced by [Mathieu Desbrun 1999]

Keywords: diffusion process, curvature flow, implicit method, anti-shrinking method

1 Introduction

Sometimes we need to create computer graphics objects using data from the real world. However, since the raw data are usually imperfectly-measured, the created computer graphics objects will have some undesirable noise and uneven edges. Fairing is a technique that can remove those noise and make irregular meshes smooth. Using the idea of diffusion and curvature flow, The author introduce several fairing methods and analyze their advantages and drawbacks from the view of both theory and implementation.

2 Notation and Definition

I am a little confused with the notation in the paper, so I modify them a little in my reading report.

- M and v_i: The mesh is noted as M. From a continuous view, M is a function of the surface. From a discrete view, M is a $m * 3$ matrix, where m denotes the number of vertices. The i^{th} vertex is noted as v_i, which is a $1 * 3$ vector and is also the i^{th} row of M.

- $\vec{L}$ and L: $\vec{L}$ is a $m*3$ matrix, it means the movement of every vertex. Usually, $\vec{L}$ is computed from the mesh M, so we also write $\vec{L}(M)$. L is a $m * m$ matrix, each entry L_{ij} is the relation between v_i and v_j. $\vec{L}$ and L are closely related, and actually $\vec{L}(M) = LM$. $\vec{L}$ is often used in theory derivation while L is often used in implementation.

- $N_1(i)$: The set of the 1-ring neighbors of vertex v_i.

3 Idea 1: Diffusion Process

The idea of diffusion is that each vertex is influenced by the coordinates of other vertices (”other vertices” usually means the neighbors of the exact vertex). In each iteration, every vertex update its position, and has new impacts on others. They update again and again until some properties satisfied. The mathematical expression of diffusion process is:

$$\frac{\partial M}{\partial t} = \lambda \vec{L}(M) \tag{1}$$

In discrete view, $\lambda * \partial t$ measures the rate of iteration. Then we change the initial fairing problem into two subproblems, namely, defining $\vec{L}(M)$ and solving ∂M.

4 Operator 1: Umbrella Operator

As the first operator, we define a easy $\vec{L}(M)$. The equation is:

$$\vec{L}(x_i) = \frac{1}{\#N_1(i)} \sum_{j \in N_1(i)} x_j - x_i \tag{2}$$

Using the definition of $\vec{L}(x_i)$, we can also compute L, the equation is:

$$L_{ij} = \begin{cases} \frac{1}{\#N_1(i)}, & \text{if } j \in N_1(i) \\ 0, & \text{if } j \notin N_1(i) \text{ and } j \neq i \\ -1, & \text{if } j = i \end{cases} \tag{3}$$

We find umbrella operator looks like the Laplacian convolution mask $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ in image processing, and actually they both are deduced from Laplacian operator. However, umbrella operator is under the assumption that all edges and angles are of the same size. This is not quite reasonable and we will fix it latter.

5 Method 1: Explicit Method

Then we solve the Eq(1). In discrete view, we can rewrite Eq(1) into:

$$\frac{M^{n+1} - M^n}{dt} = \lambda \vec{L}(M^n) \tag{4}$$

which means:

$$M^{n+1} = (I + \lambda dt L) M^n \tag{5}$$

Because it is a forward Euler method, we should be careful with the iterative rate. λdt should be less than 1, or the vertices sticking out of the surface will be pulled in too hard and sink into the surface, and vice versa, which will create oscillations.

6 Method 2: Implicit Method

Since we have to control the speed of iteration in explicit method, the process is time-consuming if the mesh is large. An alternative is using implicit Euler method. We can rewrite Eq(1) into:

$$\frac{M^{n+1} - M^n}{dt} = \lambda \vec{L}(M^{n+1}) \tag{6}$$

which means:

$$(I - \lambda dt L) M^{n+1} = M^n \tag{7}$$

Then we convert a matrix computational problem into an equation solving problem. It is more complicated, but since the coefficient matrix is a static sparse matrix, we can solve it efficiently.

7 Method 3: Automatic Anti-Shrinking Fairing

Nevertheless, implicit method still has some drawbacks. As the author writes in the paper, ”pure diffusion will, by nature, induce shrinkage”. Suppose we do pure implicit fairing on a sphere. For every vertex, its neighbors are all on the one side. They pull the vertex into the direction of the center, so after iterations, the volume of the sphere decreases.

Automatic anti-shrinking fairing can fix this problem. After each iteration, we enlarge the mesh to the original volume. The expression is:

$$M^n \leftarrow (V_0/V_n)^{1/3} M^n \tag{8}$$

Because we can use Gauss theory to compute the volume of a closed mesh, automatic anti-shrinking fairing is not hard to implement.

8 Operator 2: Weighted Umbrella Operator

As we have discussed, umbrella operator's assumption is not very reasonable. Consider the 1-D case, for a function $f(x)$, if we know $y_1 = f(x_1)$, $y_2 = f(x_2)$, $y_3 = f(x_3)$ where $x_1 < x_2 < x_3$, then we will use $f''(x_2) = \frac{2}{x_3-x_1}(f'(x_{32}) - f'(x_{21})) = \frac{2}{x_3-x_1}(\frac{y_3-y_2}{x_3-x_2} - \frac{y_2-y_1}{x_2-x_1})$ to approximate the double derivative in x_2. Extend the case to 3-D, we can get the equation of weighted umbrella operator:

$$L(x_i) = \frac{2}{\sum_{j\in N_1(i)} |e_{ij}|} \sum_{j\in N_1(i)} \frac{x_j - x_i}{|e_{ij}|} \tag{9}$$

where $|e_{ij}|$ is the length of edge e_{ij}. Comparing to the umbrella operator, weighted umbrella operator can deal with irregular meshes, but cost much more time. In each iteration, the coefficient matrix $(I - \lambda dtL)$ is no longer static, so the time complexity is much higher. An approach speeding up the operator is to assume the edge length always being same to the original length. Then weighted umbrella operator can be fast and of high quality.

9 Idea 2: Curvature Flow

So far, our discussions are all under the idea of diffusion process. Unfortunately, diffusion process is not perfect. Suppose a vertex is on the plane of its neighbors. Since the surface is absolutely smooth, the vertex should not move any more. However, diffusion process and Laplacian operator have both normal and tangential components on a vertex, so the vertex will still move. This is not good, and the curvature flow can help fix it.

The equation of curvature flow is:

$$\frac{\partial x_i}{\partial t} = -\overline{\kappa}_i \mathbf{n}_i \tag{10}$$

where $\overline{\kappa}_i$ is the mean curvature ($\overline{\kappa} = (\kappa_1 + \kappa_2)/2$), **n** is the norm vector in vertex v_i. Using curvature flow, the vertex tends to move toward the plane, rather than the center of its neighbors.

10 Operator 3: Curvature Operator

Using the definition of curvature normal $\overline{\kappa}\mathbf{n}$:

$$\frac{\nabla A}{2A} = \overline{\kappa}\mathbf{n} \tag{11}$$

where A is the area of a small region around the point P where the curvature is needed. Then we can deduce the curvature operator:

$$L(x_i) = \frac{1}{\sum_{j\in N_1(i)}(\cot\alpha_j^l + \cot\alpha_j^r)} \sum_{j\in N_1(i)} (\cot\alpha_j^l + \cot\alpha_j^r)(x_i - x_j) \tag{12}$$

When we implement curvature operator, we may encounter a face of zero area computing the matrix L. If this happens, we should skip (or sometimes delete) these special degenerate triangles.

11 Comparison and Result

- Umbrella operator changes the shape of the objects and the triangles tend to become uniformly distributed. So if the initial mesh has triangles with almost the same size, then umbrella operator will work well; or the mesh will be distorted, because different parts will have different shrinking rate. However, umbrella operator is very easy to implement, and since the coefficient matrix L is static, the time cost is low.
- Weighted umbrella operator almost keep the original distribution of triangles, so it can deal with meshes with irregular triangles. Although it will still introduce tangential drifts, it is not very significant. Using the equal-length approximation, the coefficient matrix L is also static, so weighted umbrella is as easy as umbrella operator and has low time cost.
- Curvature operator only has normal drifts on vertices, so it will produce meshes with most respects to the original shape and triangle distribution. However, we should update the coefficient matrix L every iteration, so the curvature operator has a high time complexity.

12 Discussion

12.1 Soft and Hard Material

Sometimes we need to put hard and soft constraints on the mesh. To implement hard constraints, we can make a set of vertices stay fixed, which equals to $L(x_i) = 0$ mathematically. To implement soft constraints, we can add weight coefficients to each vertices.

12.2 Boundary

Sometimes the surface is not closed or with some holes. There are two ways to deal with this situation. The first is to smooth the boundary, thus the boundary gets rounder and rounder. The second is to add virtual vertices to make the surface closed, then we can use usual method to fair it.

References

MATHIEU DESBRUN, MARK MEYER PETER, S. O. A. H., 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. ACM Press/Addison-Wesley Publishing.