Mean Curvature Flow and Applications

Maria Eduarda Duarte
Departamento de Matemática
Universidade Federal de Santa Catarina
Florianópolis, Brazil
Email: m.duarte@ufsc.br

Leonardo Sacht
Departamento de Matemática
Universidade Federal de Santa Catarina
Florianópolis, Brazil
Email: leonardo.sacht@ufsc.br

***Abstract*—Input data for geometry processing is commonly problematic: the discretization of the volume of a surface may contain self-intersections or noise. These problems are related to curvature. In this work we review some concepts in differential geometry, geometry processing and present the *Mean Curvature Flow* to solve those problems. We review some modifications in the discrete case to improve the discrete flow. Then we analyse an implementation of the *Conformalized Mean Curvature Flow*.**

I. Introduction

During the physical scanning process that we perform to create 3D digital surface models, high-frequency noise can occur [1]. We increase the quality of a scanned real surface reducing the noise in the data by smoothing the surface (2-manifold) (Figure 1). To minimize the noise we reduce changes in curvature. Intuitively, we want to change the curvature of our surface like heat changes in an object (actually we specifically use the *heat equation*); the heat flow from warmer to cooler regions, a process that we call *diffusion*. In our case, the neighborhood of high curvature points is moved to produce the new surface. The idea is to choose a suitable velocity vector for each point and obtain a geometric continuous evolution time for the surface.

Another recent application of the Mean Curvature Flow (MCF) is to remove self-intersections in surfaces for which we want to calculate the volume [2]. The MCF can be modified to converge to unit sphere [3], i.e., a surface without self-intersections. An example of this flow is in Figure 2, where all intersections in the input mesh were removed.

In this paper we show how the *Mean Curvature Flow* can solve these problems in geometry processing, as well as providing didactic materials with formal definitions. We recommend knowledge in Linear Algebra [4], Calculus of Several Variables [5] and introductory ideas in Differential Geometry (Chapter 2 and 3 of [6]).

To understand these problems we review the concept of curvature in 2-manifolds. So we can start to talk about the continuous MCF. To deal with the discrete case we first show how the surface is represented in discrete data using a *polygon mesh*. Finally we present the discrete MCF (and its variations), showing its difficulties and discussing MATLAB implementations.

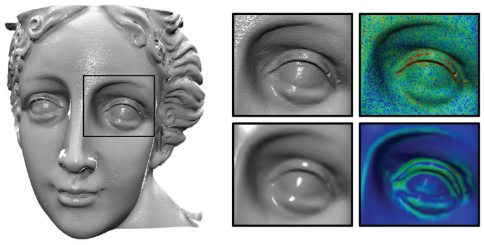

Fig. 1. On the left a 3D scan of a statue's face corrupted by typical measurement noise. The color-coded visualization of mean curvature flow on the right shows the difference before (top) and after (bottom) the denoising process around the eye region. This image was taken from [7].

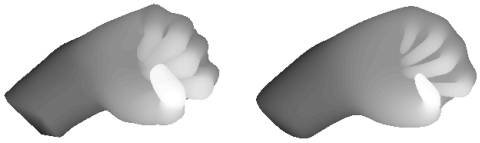

Fig. 2. The *Hand* on the left has self-intersections in the fingers. We flow the surface according to the Conformalized Mean Curvature Flow to obtain a new surface without self-intersections (right).

II. Mean Curvature Flow

A. Mean Curvature Flow - Continuous case

1) Curvature: The Curvature of a curve measures how far the curve is from being a straight line and we use second derivatives to measure it. Surfaces have an analogous approach: the *normal curvature* of a surface measures how far it is from being a flat plane and also uses *partial* second derivatives to measure it. However, a point in a surface can have more than one normal curvature.

Let S be a surface, C a curve on S passing through p. Denote $k = \|C''\|$ and n the normal vector of the curve. The normal curvature k_n in $p \in S$ relative to C is obtained projecting the second derivative vector kn over the normal N of the surface at p and getting its norm (Figure 3). But which curve should we use? We have infinite options. In differential geometry there is only two most important normal curvatures (for a more in-depth discussion see [6]).

Definition 1 (Principal Curvature)**.** *Let k_1 and k_2 be the maximum and the minimum respectively of the normal curvature in a point p on a surface S. We call k_1 and k_2*

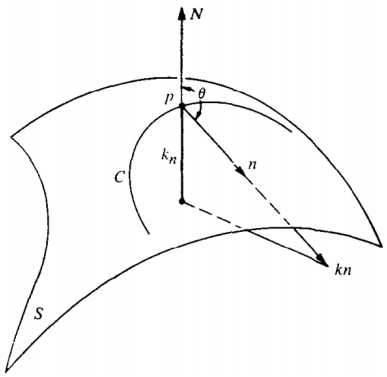

Fig. 3. We project the second derivative vector kn of a curve C in some surface S and take the norm to obtain the normal curvature k_n at $p \in S$. The vectors N and n are normals of the surface and the curve respectively. This image was taken from [6].

principal curvatures *in p; the correspondent directions are called* principal directions *in p.*

A proof to existence of maximum, minimum can be found in Chapter 3 of [6].

We define the *Mean Curvature* (at some point) as the average of the principal curvatures

$$H := \frac{k_1 + k_2}{2}.$$

The *Mean Curvature Vector* is defined as $\overrightarrow{H} = HN$.

Now we will discuss the *Laplace Beltrami Operator*, which plays an important role in the Mean Curvature Flow, and understand its connection with Mean Curvature.

2) Laplace-Beltrami Operator: It can be seen as a second derivative of a multiple variable function. Let $f : U \subset \mathbb{R}^2 \to \mathbb{R}$, U be open subset. The *Laplacian* of f, denoted by Δf, is defined as the divergence of the gradient of f:

$$\Delta f = \operatorname{div}(\nabla f) = \operatorname{div}\left(\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v}\right) = \frac{\partial^2 f}{\partial u^2} + \frac{\partial^2 f}{\partial u^2}$$

Let us get some intuition: the gradient at a point $(u, v) \in \mathbb{R}^3$ points in the direction of greatest rate of increase of f. So if $f(u, v)$ is a local maximum, the gradient at the neighborhood of (u, v) will always point to (u, v). Then the divergent of the gradient will be negative. If $f(u, v)$ is a local minimum, the opposite occurs and the divergent will be positive. So the Laplacian measures how much of a minimum point is (u, v) and it is an analogous to the second derivative in ordinary calculus. Summing up, $\Delta f(u, v)$ is, intuitively, the average difference $f(x, y) - f(u, v)$ over small circles around (u, v).

The Laplace-Beltrami Operator is a generalization of the Laplace Operator to operate in functions defined on surfaces. This depends on the metric used and can be extended to operate in other geometric objects. For our purposes we keep the same notation and use the Euclidian metric.

Making a parallel with the heat equation, let $S \subset \mathbb{R}^3$ be a surface and $g : S \to \mathbb{R}$ the temperature at a point $p \in S$. The heat equation is given by

$$\frac{\partial g}{\partial t} = a^2 \Delta g, \quad a \in \mathbb{R}.$$

If the average temperature over small spheres around a point p is hotter than at p, then in the next time the temperature at that point will rise. What happens if we use this equation to move points? In the place of heat, what properties are we altering?

3) Mean Curvature Flow: We want to evolve an initial surface M_0 throughout time t obtaining a family of surfaces $\{M_t\}_{t\in[0,T)}$ with the respective parametrization $\varphi_t : U \subset \mathbb{R}^2 \to M_t \subset \mathbb{R}^3$. To find those surfaces we solve the *geometric diffusion equation*, a linear partial differential equation (PDE)

$$\frac{\partial \varphi}{\partial t} = \Delta \varphi \tag{1}$$

where the Laplacian of φ is the vector formed by the Laplacian with respect to each coordinate of φ. The parametrization φ also depends on the time t and each φ_t is found fixing t. If $p = (p_1, p_2, p_3) \in M_t$ is local maximum point at the coordinate x, then the Laplacian at x will be negative and p_1 will reduce in the next time. Therefore the Mean Curvature Flow reduces curvature locally.

Finally we can see the connection between the Mean Curvature and the Mean Curvature Flow (until now it was just the name). A theorem by Weierstraβ (see [8]) that states that, given an orientable[1] surface in Euclidean space, one has:

$$\Delta \varphi = 2\overrightarrow{H}.$$

Then according to equation (1) the velocity of each point at the surface will be given by the mean curvature vector. The existence and uniqueness of a solution to (1), i.e., a family of immersions $\{\varphi_t\}_{t\in[0,T)}$, can be proved for compact 2-manifolds. In case of interest, the proof can be found in [9].

This Flow has some interesting properties, namely, it is the flow of steepest decrease of surface area (for results see [10]) and also reduces volume [11]; besides that, every convex, embedded, compact surface converge to a single point $q \in \mathbb{R}^3$. If we rescale to keep the area constant, the surface converges to spheres [9] . But if the surface is not convex, the MCF form singularities. In the discrete case, the flow can be modified (Equation 4) to avoid their formation (see [3]). In Figure 4 we show the formation of singularities (top) in convex regions (fingers of *armadillo man*) and the evolved surface following the proposed *Conformalized Mean Curvature Flow* without singularities. We provide more details on these flows in the next sections.

B. Mean Curvature Flow to discrete surface

When an object is represented in the computer the storage is limited, so we use discrete data to represent the digital surface. Due to its simplicity and flexibility piecisewise linear

[1]In this work we use only orientable, embedded, compact surfaces.

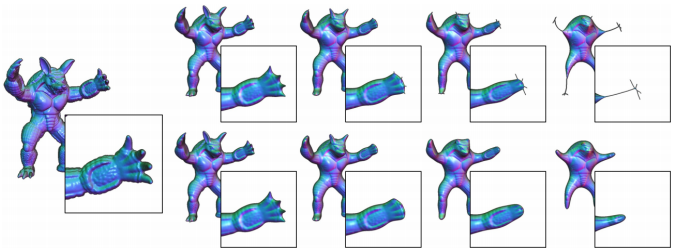

Fig. 4. The initial surface (*armadilo man*) on the left is evolved with the traditional semi-implicit MCF (top) and the cMCF (bottow). The hand of armadilo, that is a non convex region, evolves and forms neck pinch singularities on the top; the cMCF avoid their formation. Image from [3].

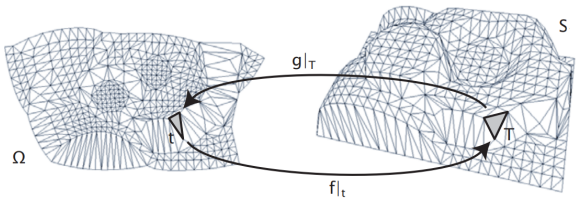

Fig. 5. The parametrization f maps vertex from the domain Ω to the triangle mesh S. Image from [13].

approximations (*polygonal meshes*) are used: a collection of vertices, edges and faces. In Figure 5 we have a triangular mesh which is extensively used in Computer Graphics [12] and also the only one used in this work. For more information about polygonal meshes and data structure see [7].

The parametrization φ of a polygonal mesh (discrete surface) is now defined by its vertices $\varphi(v_i)$ where v_i is a vertice of the mesh on the domain in $\mathbb{R}^2$ (Figure 5). So we represent the parametrization as a vector $\varphi := (\varphi(v_1), \ldots, \varphi(v_m))$ (note that the entries are vertex positions of the triangle mesh in $\mathbb{R}^3$) and the Laplacian becomes the vector $\Delta\varphi = (\Delta\varphi(v_1), \ldots, \Delta\varphi(v_n))$.

Note that the geometric diffusion equation (1) is a continuous time-dependent PDE, so we have to discretize both in space and time, i.e., discretize the Laplace-Beltrami Operator and the partial derivative using *finite element method* [14].

1) Discrete Laplace-Beltrami Operator: Consider the surface S discretized by a triangular mesh. Let $\varphi : U \subset \mathbb{R}^3 \to S$ be the discrete parametrization. To compute the Laplacian at some vertex $v_i \in U$ we use information at a local neighborhood $N_n(v_i)$ of the vertex called *n-ring*. The distance between two vertices v_i and v_j on the 2D domain mesh is given by the minimum number of edges between them. Define $N_n(v_i)$ as the the set of vertices with distance less than n from v_i (analogous to an open ball). From now on we fix $n = 1$.

We also consider an average region defined on vertex one-ring neighborhoods that we call *cell*. There are three variants (Figure 6): the *Barycentric cell* connect the triangle barycenter with the edge midpoints; in place of barycenter we can use the circumcenters to obtain the *Voronoi cell*. As Figure 6 illustrates, the circumcenter can be outside of the triangle generating a worse approximation [7]. We can replace the circumcenter of obtuse triangles by the midpoint of the edge

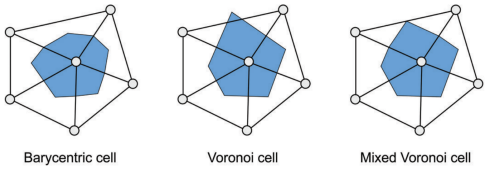

Fig. 6. The local average regions are indicated with blue color, they are used to compute discrete differential operators at the central vertex of the one-ring neighborhood. Image from [7].

opposed to the central vertex, obtaining the *Mixed Voronoi cell*.

There are different ways to discretize the Laplacian operator and none of them keep every natural property of the continuous case [15]. In this work the only discretization used is the *Cotangent Formula* (2) that is one of the most widely used for triangle meshes applied to smoothing processing [7]. We denote A_i as the area of the cell. For more discrete operators see [12].

$$\Delta\varphi(v_i) := \frac{1}{2A_i} \sum_{v_j \in N_1(v_i)} (\cot\alpha_{i,j} + \cot\beta_{i,j})(\varphi(v_j) - \varphi(v_i)) \tag{2}$$

If $w_i = 1/2A_i$ and $w_{i,j} = \cot\alpha_{i,j} + \cot\beta_{i,j}$, then we have

$$\Delta\varphi = \begin{pmatrix} \Delta\varphi(v_1) \\ \vdots \\ \Delta\varphi(v_m) \end{pmatrix} = \underbrace{\mathbf{D}\mathbf{M}}_{\mathbf{L}} \begin{pmatrix} \varphi(v_1) \\ \vdots \\ \varphi(v_m) \end{pmatrix} = \mathbf{L}\,\varphi \tag{3}$$

where $\mathbf{D} = \mathrm{diag}(w_1, \ldots, w_m)$ is a diagonal matrix and $\mathbf{M}$ is a symmetric matrix with edges weights w_{ij}:

$$\mathbf{M}_{ij} = \begin{cases} -\sum_{v_k \in N_1(v_i)} w_{ik}, & i = j, \\ w_{ij}, & v_j \in N_1(v_i), \\ 0, & \text{otherwise.} \end{cases} .$$

We can modify the system matrix $\mathbf{L}$ to obtain some desirable properties [7]. Note that $\mathbf{M}$ is a *sparse matrix*, since the Laplacian is defined locally in terms of the 1-ring. Thus $\mathbf{L}$ is also sparse. The Laplacian matrix is not symmetric in general, because of the scale matrix $\mathbf{D}$. We can turn into a *symmetric* system multiplying by the inverse $\mathbf{D}^{-1}$ getting

$$\mathbf{M}\,\varphi = \mathbf{D}^{-1}\,\Delta\varphi. \tag{4}$$

Including some constrains and changing signs of the Laplacian system, we also obtain a *positive definite system*. These 3 properties are beneficial to solve the Laplacian system [7].

2) Discrete partial derivative: To discretize the time derivative in (1) we use the *Euler Method* [16]. We split the interval of time in regular intervals of size h obtaining $\{t, t+h, t+2h, \ldots\}$ and use the finite difference approximation. Let $\varphi_t = (\varphi_t(v_1), \ldots, \varphi_t(v_m))$ for each t, $\frac{\partial\varphi}{\partial t} = \left(\frac{\partial\varphi}{\partial t}(t, v_1), \ldots, \frac{\partial\varphi}{\partial t}(t, v_m)\right)$ and $\frac{\partial\varphi}{\partial t} \approx \frac{\varphi_{t+h} - \varphi_t}{h}$. Solving for φ_{t+h} yields the *explicit Euler integration*

$$\varphi_{t+h} = \varphi_t + h\frac{\partial \varphi}{\partial t}.$$

We now use equations (1) and (3) to obtain the discretization of the MCF

$$\varphi_{t+h} = \varphi_t + h\,\mathbf{L}_t\,\varphi_t.$$

Note that the matrix $\mathbf{L}_t$ change each time. If we evaluate the Laplacian $\Delta\varphi$ at the next time step $(t+h)$ instead of the current time t we obtain the *implicit Euler integration*

$$\varphi_{t+h} = \varphi_t + h\,\mathbf{L}_{t+h}\,\varphi_{t+h},$$

The implicit MCF is more stable to be computed: we can increase the time step h. On the other hand in the implicit case we have to solve a linear system at each step, while in the explicit formulation we have a matrix-vector product. Another formulation of MCF, the one we use, is the *semi-implicit* formula [3]

$$\varphi_{t+h} = \varphi_t + h\,\mathbf{L}_t\,\varphi_{t+h}$$

We can write the above equation as

$$(\mathbf{D}_t^{-1} - h\,\mathbf{M_t})\varphi_{t+h} = \mathbf{D}_t^{-1}\varphi_t$$

The work [3] suggests the following discrete formulation (5) called *Conformalized Mean Curvature Flow* (cMCF) to avoid singularities, where the matrix $\mathbf{M}_0$ is fixed, and only the matrix $\mathbf{D_t}$ is updated at each time.

$$(\mathbf{D}_t^{-1} - h\,\mathbf{M_0})\varphi_{t+h} = \mathbf{D}_t^{-1}\varphi_t \tag{5}$$

With some evidences in [3] we can conjecture that the cMCF has a stable convergence for genus-zero meshes and converges to a conformal parametrization of the surface onto sphere.

III. Implementation of the Conformalized Mean Curvature Flow

We now analyze an implementation in MATLAB of the Conformalized Mean Curvature Flow (5). The algorithm is available in the toolbox [17] for Geometry Processing in MATLAB. In this section we explain the general structure of the algorithm and show some results. For a more complete analysis of the cMCF and comparison with others flows see [3] and [18].

A triangle mesh in MATLAB is represented by two matrices V of vertex positions and F of the triangle indices (form the faces). We provide these two matrices in the cMCF function and it returns two matrices U, with new vertex positions, and $U\mathrm{steps}$ with the vertex positions at each time step obtained solving (5). Some adjustment are available such as change of the time step h and the number of maximum iterations. The default cell is Barycentric but the Voronoi is also available.

The program stops when the surface converge to a sphere or exceeds the maximum number of iterations. We evolve the surface in Figure 2 to remove self-intersections. The algorithm has the option to run until we have a self-intersection free surface. In Figure 7 we show the evolution of a surface with genus 1 and h=0.017 that converges in 89 iterations. See that the most expressive changes occur before the time step 25.

Fig. 7. The evolution of a surface using cMCF. The respective time steps are 1,2,5,25 and 89, when it converged. Image generated using MATLAB.

IV. Conclusion

This work describes and apply the Mean Curvature Flow and show some applications in Geometry Processing. We approached the continuous and discrete cases and discussed an implementation in MATLAB. This flow has a couple of discrete variations to achieve some properties.

The most expressive variation of MCF presented is the Conformalized Mean Curvature Flow that avoid singularities in non-convex regions and is conjectured to converge to conformal parametrization onto the sphere. It is an efficient flow that can reduce noise and remove auto-intersections.

Acknowledgment

The authors thank the National Council for Scientific and Technological Development CNPq - Brazil for the support.

References

[1] T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. H. Gross, "Post-processing of scanned 3d surface data." *SPBG*, vol. 4, pp. 85–94, 2004.

[2] L. Sacht, A. Jacobson, D. Panozzo, C. Schüller, and O. Sorkine-Hornung, "Consistent volumetric discretizations inside self-intersecting surfaces," in *Computer Graphics Forum*, vol. 32, no. 5. Wiley Online Library, 2013, pp. 147–156.

[3] M. Kazhdan, J. Solomon, and M. Ben-Chen, "Can mean-curvature flow be modified to be non-singular?" in *Computer Graphics Forum*, vol. 31, no. 5. Wiley Online Library, 2012, pp. 1745–1754.

[4] G. Strang, *Linear Algebra and Its Applications*. Brooks Cole, 2006.

[5] J. E. Marsden and A. Tromba, *Vector calculus*. Macmillan, 2003.

[6] P. Manfredo, *do Carmo. Differential geometry of curves and surfaces*. Prentice Hall, 1976.

[7] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon mesh processing*. CRC press, 2010.

[8] U. Dierkes, S. Hildebrandt, A. Küster, and O. Wohlrab, "Minimal surfaces," in *Minimal Surfaces I*. Springer, 1992, pp. 53–88.

[9] F. MARTÍN and J. PÉREZ, "An introduction to the mean curvature flow," 2014.

[10] T. H. Colding and W. P. Minicozzi, *A course in minimal surfaces*. American Mathematical Soc., 2011, vol. 121.

[11] T. H. Colding, I. Minicozzi, and P. William, "Minimal surfaces and mean curvature flow," *arXiv preprint arXiv:1102.1411*, 2011.

[12] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *Visualization and mathematics III*. Springer, 2003, pp. 35–57.

[13] K. Hormann, B. Lévy, and A. Sheffer, "Mesh parameterization: Theory and practice," 2007.

[14] G. Strang and G. J. Fix, *An analysis of the finite element method*. Prentice-hall Englewood Cliffs, NJ, 1973, vol. 212.

[15] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun, "Discrete laplace operators: no free lunch," in *Symposium on Geometry processing*, 2007, pp. 33–37.

[16] K. Atkinson, W. Han, and D. E. Stewart, *Numerical solution of ordinary differential equations*. John Wiley & Sons, 2011, vol. 108.

[17] A. Jacobson *et al.*, "gptoolbox: Geometry processing toolbox," 2016, http://github.com/alecjacobson/gptoolbox.

[18] K. Crane, U. Pinkall, and P. Schröder, "Robust fairing via conformal curvature flow," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 61, 2013.