An Adapted Laplacian Operator For Hybrid Quad/Triangle Meshes

Alexander Pinzón Fernández

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Sistemas e Industrial

Grupo de Investigación CIM@LAB

Bogotá, Colombia

2014

An Adapted Laplacian Operator For Hybrid Quad/Triangle Meshes

Alexander Pinzón Fernández

A thesis submitted in partial fulfillment of the requirements for the degree of:

Master in Systems Engineering and Computer Science

Advisor:
Eduardo Romero Castro , Ph.D.

Research Area:
Computer Graphics

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Sistemas e Industrial
Grupo de Investigación CIM@LAB
Bogotá, Colombia
2014

Dedicación

A Beatriz y Campo Elias mis padres que siempre me dieron la libertad de escoger, con su comprensión y apoyo me permitieron dedicar mi tiempo a la ciencia.

A mis padres

Beatriz Fernandez Vargas
Campo Elias Pinzón Rojas

Acknowledgment

I would like to thank my advisor professor Eduardo Romero and the CIM&LAB research group for their support in this thesis.

I would like to thank all the workers in Colombia who fund public education with their work.

This work was supported in part by the Blender Foundation and the Google Summer of Code program 2012 and 2013.

Abstract

In the last two decades three-dimensional modeling methods used by artists have been evolving and developing rapidly thanks to the use of vector operators of differential geometry such as the Laplacian operator. This operator allows modeling the behavior of complex applications such as noise reduction, enhancement, remeshing, UV mapping, posing and skeletonization, among others, in a simple way. The Laplacian operator is theoretically defined in a continuous and smooth domain, named manifold. In practice manifolds are often approximated by discrete polygon meshes composed by triangles and quadrangles which represent the real world three-dimensional objects with which the artists work. In these meshes, spectral structure is calculated using a discrete Laplacian operator, i.e. the discrete version of the Laplacian operator given by Pinkall in 1993. This approach only worked with triangle meshes. In 2011 Xiong extended the operator to work exclusively with quad meshes. This thesis proposes an original extension of the Laplacian operator that allows working with hybrid meshes composed by triangles and quadrangles.

Along with the operator, this work presents new sculpting and modeling applications based on enhancement. Additionally, applications on subdivision surfaces which use smoothing, mesh posing which use differential coordinates and skeletonization which use iterative contractions are developed. This series of applications demonstrates the quality, predictability and flexibility of the proposed operator.

The proposed operator was successfully used in new software tools in real production environment within 3D computer graphics software Blender. Currently these tools are available as open source software.

Resumen

En las dos últimas décadas los métodos de modelado tridimensional utilizadas por los artistas han ido evolucionando y desarrollándose rápidamente, en parte gracias al uso de operadores vectoriales de geometría diferencial, como el operador de Laplace. Este operador permite modelar de una manera sencilla el comportamiento de aplicaciones complejas tales como la reducción de ruido, realce, remallado, mapeado UV, posado y esqueletonización, entre otros. Este operador Laplaciano es teóricamente definido en un dominio continuo y suave llamado variedad, las variedades son a menudo aproximadas por mallas discretas de polígonos compuestas por triángulos y cuadrángulos que a su vez representan objetos tridimensionales del mundo real que los artistas trabajan. En estas mallas se calcula la estructura espectral con el uso de algún operador Laplaciano discreto, la versión discreta del operador Laplaciano propuesta por Pinkall en el 1993 trabaja únicamente con mallas compuestas por triángulos, y la de Xiong en el 2011 trabaja exclusivamente con cuadrángulos. Esta tesis propone una extensión original del Operador Laplaciano que permite trabajar con mallas híbridas compuestas por triángulos y cuadrángulos.

Junto con el operador, este trabajo presenta nuevas aplicaciones en esculpido y modelamiento con base en el realce, aplicaciones en subdivisión de superficies con el uso de suavizado, posado de mallas con el uso de coordenadas diferenciales y esqueletonización usando contracción iterativa. Esta serie de aplicaciones demuestra la calidad, predictibilidad y flexibilidad del operador propuesto.

El operador propuesto fue usado con exitoso en las nuevas herramientas del software para gráficos 3D por computadora Blender. Actualmente estas herramientas están disponibles como programas de código abierto.

Keywords: laplacian operator; smooth; enhance; sculpting; subdivision surface

Contents

List of Figures

List of Tables

1 Introduction

The discrete versions of the Laplace Beltrami Operator have been used in recent years for the development of new geometric modeling tools. Pinkall [29] introduced the cotangent version of the Laplace operator, which allowed finding the minimal surface when computing a discrete harmonic map with the Laplacian operator. This version has been widely studied and applied in various problems of computer geometric modeling. This type of operator was defined over manifolds, i.e. continuous domains homeomorphic to $\mathbb{R}^n$ that are represented in pratice by polygon meshes. These polygons are generally composed of triangles and quadrangles. Working with the Laplacian operator in this hybrid composition is not a mathematical challenge, most research only uses meshes composed by triangles [29, 15, 25, 37, 3, 4, 20]. In recent studies [23, 43], the Laplacian operator may be used in meshes composed exclusively by quadrangles. However, from an artistic point of view, the topology and the way the edges, triangles and quadrangles are distributed, directly affects the processes of animation, interpolation, texturing, etc., as discussed by [26], who uses a manual connection of a pair of vertices to perform animation processes and interpolation. It is then of paramount importance to develop operators that easily interact with such meshes, eliminating the need of preprocessing the mesh to convert it to triangles and change the original topology.

Presently, modeling techniques that are able to generate a variety of realistic shapes are available [7] . Editing techniques have evolved from affine transformations to advanced tools such as sculpting [11, 17, 41], editing, creation from sketches [21, 19], and complex interpolation techniques [37, 45]. Catmull-Clark based methods however require interaction with a minimum number of control points for any operation to be efficient, or in other words, a unicity condition is introduced by demanding a smooth surface after any of these shape operations. Hence, traditional modeling methods for subdividing surfaces from coarse geometry have become widely popular [9, 40]. These works have generalized a uniform B-cubic spline knot insertion to meshes. Some of these add some type of control; for instance with the use of creases to produce sharp edges [13] or the modification of some vertex weights to locally control the zone of influence [5]. Nevertheless, these methods are difficult to use as they require a large number of parameters and a very tedious customization.

On the other hand, the proposed applications require a single parameter that controls the global curvature, which is used to maintain realistic shapes, creating a family of different versions of the same object and therefore preserving the detail of the original model and a realistic appearance.

The shape inflation and shape exaggeration can thus be used as a type of brush in the sculpting process. When inflating a shape with other brushes the former ends up losing detail when moving vertices [41]. In contrast, the presented enhance method inflates a mesh by moving the vertices towards the reverse curvature direction, conserving the shape and sharp features of the model.

Contributions This work presents an extension of the Laplace Beltrami Operator for hybrid quad/triangle meshes that have a larger mesh functionality spectrum than common triangular or quadrangular meshes. The method eliminates the need of preprocessing and allows preservation of the original topology. Along with this operator, we propose a method to generate a family of parameterized shapes, in a robust and predictable way. This method enables customization of the smoothness and curvature obtained during the subdivision surfaces process. Finally, a new brush for inflating the silhouette mesh features in modeling and sculpting is proposed.

The work is organized as follows: chapter 2 presents works related to the Laplacian operator, applications in digital sculpting, deformation, and offsetting methods for polygonal meshes. It also describes the theoretical framework of the Laplacian operator for polygon meshes; in chapter 3, we show the extension of the Laplacian Operator for hybrid meshes and the applications of shape inflation ,subdivision of surfaces and sculpting; in chapter 4, we present an application for mesh smoothing of the Laplacian operator extension here proposed, and implemented into a well known software for computer modelling. Finally, in chapter 5, we present a successful application for mesh deformation and model re-posing based on differential coordinates and the adaptation of this method to work with our Laplacian operator extension.

2 Mathematical Foundation and Background

This chapter studies basic mathematical foundations on differential geometry to understand the differential operators and the Laplace Beltrami operator.

The differential geometry studies curvatures and geodesics [20]. These differential operators show a deep relationship between the geometry (curvatures, geodesics) and topology of the manifold. They have been commonly used in computer geometric modeling applications over recent years [35, 1].

2.1 Related work

Many tools, based on the Laplacian mesh processing, have been developed for modeling. These tools preserve the surface geometric details when using Laplacian operators for different processes such as smoothing, enhancing, free-form deformation, fusion, morphing and other applications [34].

The most used discretization of Laplace Beltrami operator Δ_Ω over a triangulated mesh Ω was proposed by Pinkall [29].

$$\Delta_\Omega(u) = \frac{1}{2}\sum_{j\in N_1(i)}(\cot\alpha_j + \cot\beta_j)(x_i - x_j)$$

Where $N_1$is the 1-ring neighborhood, α and β are the opposite angles to edge between vertex i and vertex j. In this work the discrete Laplacian operator is used to find the minimal surface based on energy minimization strategy using the Dirichlet's energy of the function u over a manifold represented by triangulated mesh Ω.

$$E_D(u) = \frac{1}{2}\int_\Omega \left|\nabla u\right|^2$$

Taubin [42] was the first to treat the problem of noise reduction in digital polygonal meshes from a signal processing point of view. He extended Fourier analysis to signals defined on polygonal meshes, and observed that Fourier transformation is a decomposition of the signal into eigenvectors of the Laplacian operator and reconstructs the signal with a linear

combination of these eigenvectors. Desbrun et al. [15] considered the same approach as Taubin, but they used a curvature normal ($\bar{\kappa}\mathbf{n}$) based on a cotangent Laplacian operator version for noise reduction over a diffusion process. This is the most famous and popular Laplace Beltrami operator discretization [23]; many works for mesh smoothing and fairing have been developed based on this Laplace Beltrami Operator (LBO) discretization [14, 25, 36, 27]:

$$\frac{\partial x_i}{\partial t} = -\bar{\kappa_i}\mathbf{n_i}$$

$$-\bar{\kappa_i}\mathbf{n_i} = \frac{1}{A}\sum_{j\in N_1(i)}(\cot\alpha_j + \cot\beta_j)$$

where A is the area surrounding vertex i. This Laplacian operator L was used to reduce the noise in a mesh X over a diffusion process.

The convergence of the Laplace Beltrami operator has been very important in fields such as numerical analysis, given its implications in the simulation process and geometric partial differentials equations. Xu et al. [44] established the convergence of several discrete Laplace Beltrami operators over triangulated meshes with numerical results that support the theoretical analysis. Over quadrilateral meshes, Liu et al. [23] presented a discrete Laplace Beltrami Operator based on a bilinear interpolation and its convergence over meshes composed only by quads.

In the work of Sorkine et al. [38] the Laplacian operator was used to re-pose a mesh while preserving geometry details of the surface. The details were stored in differential coordinates δ_i for every vertex v_i.

$$\delta_i = \sum_{j\in N_1(i)}\frac{1}{2}(\cot\alpha_j + \cot\beta_j)(v_i - v_j)$$

The differential coordinates represent the difference between the absolute coordinate of $v_i$and the center of mass of its immediate neighbors.

Offset methods for polygon meshing, based on the curvature defined by the Laplace Beltrami operator, have been developed. These methods adjust the shape offset by a constant distance, with high precision. Nevertheless, these methods fail to conserve sufficient detail because of the smoothing, a crucial issue which depends on the offset size [46]. In volumetric approaches, when using point-based representations, the offset boundary computation is based on the distance field and therefore when calculating such offset, the topology of the model may be different to the original [10].

Gal et al. [16] proposed automatic feature detection and shape edition with feature interrelationship preservation. They defined salient surface features like ridges and valleys, characterized by their first and second order curvature derivatives (see [28]) and angle-based

thresholds. Likewise, curves have also been classified as planar or non-planar, approximated by lines, circles, ellipses and other complex shapes. In each case, the user defines an initial change over several features which is propagated towards other features, based on the classified shapes and the inter-relationships between them. This method works well with objects that have sharp edges, composed of basic geometric shapes such as lines, circles or ellipses. However, the method is very limited when models are smooth since it cannot find the proper features to edit.

Traditionally, digital sculpting has been approached under a polygonal representation or a voxel grid-based method. Brushes for inflation operations only depend on the vertex normal [41]. In grid-based sculpting, other operations allow the addition or removal of voxels, since production of polygonal meshes requires a processing of isosurfaces from volume [17]. The drawback comes from the difficulty of maintaining the surface details during larger scale deformations.

In literature, several studies have described the skeleton extraction systems and different metrics that identify appropriate methods given a specific application [2]. One of the best methods reported in literature for the extraction of the skeleton is the Laplacian smoothing method given its advantages of homotopy representation and hierarchical connections between parts. The skeleton extraction method permits the simplification of the dimension of the object while preserving the topological structure [12]. Au et. al. [3] present a skeleton extraction method based on iterative smoothing-contraction. In this method several constraints are used to guarantee that the process converges to a skeleton formed by branches and joints. The constraints are based on the Laplacian operator; the low frequencies of the mesh are preserved with the use of an attractor to the original mesh, while the iterative smoothing process removes high frequencies.

2.2 Manifolds

A manifold is a topological space M with the following properties:

If $x \in M$, then there is some neighborhood $N(x)$ and some integer $n \geq 0$ such that $N(x)$ is homeomorphic to $\mathbb{R}^n$ [39].

Our work is related to manifolds that represent a surface in an three-dimensional Euclidean Space. These manifolds are homeomorphic to $\mathbb{R}^2$.

The manifolds are represented by polygonal meshes with points connected by triangles and quads.

2.3 Laplace Operator

In computer graphics a manifold is often approximated by a discrete mesh [34], it is therefore necessary to define a discrete Laplacian operator that acts on functions defined by such meshes.

Consider a smooth compact manifold M of dimension m isometrically embedded in a Euclidean space $\mathbb{R}^d$.

Given a twice continuously differentiable function $f \in C^2(M)$, let $\nabla_{\mathrm{M}} f$ denote the gradient vector field of f on M.

The Laplace-Beltrami operator Δ_{M} of f is defined as the divergence of the gradient; that is [39],

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = 0$$

$$\nabla^2_{\mathrm{M}} f = \Delta_{\mathrm{M}} f = 0$$

$$\Delta_{\mathrm{M}} f = div\left(\nabla_{\mathrm{M}} f\right) \tag{2-1}$$

2.3.1 Discrete Laplace Operator Setting

Discrete Laplacian operators are linear operators that act on functions defined by meshes. These functions are defined by their values at the vertices.

Thus, if a mesh M has n vertices, then functions on M will be represented by vectors with n components and a mesh Laplacian will be described by an $n \times n$ matrix [34].

Locally, the Laplacian operator takes the difference between the value of a function at a vertex and a weighted average of its values at the first-order or one-ring neighbor vertices, therefore a Laplacian matrix L has a local form that is given by

$$L(f)_i = b_i^{-1} \sum_{j \in N(i)} w_{ij} (f_i - f_j)$$

where $w_{ij}$are the weights between the vertex i and the vertex j. b_i^{-1} are the factors depending on the boundary region over vertex i. $N(i)$ denotes the neighbors that share an edge with vertex i.

3 Shape Inflation With an Adapted Laplacian Operator For Hybrid Quad/Triangle Meshes

Alexander Pinzón, Eduardo Romero
Cimalab Research Group
Universidad Nacional de Colombia
Bogota-Colombia
Email: apinzonf@unal.edu.co, edromero@unal.edu.co

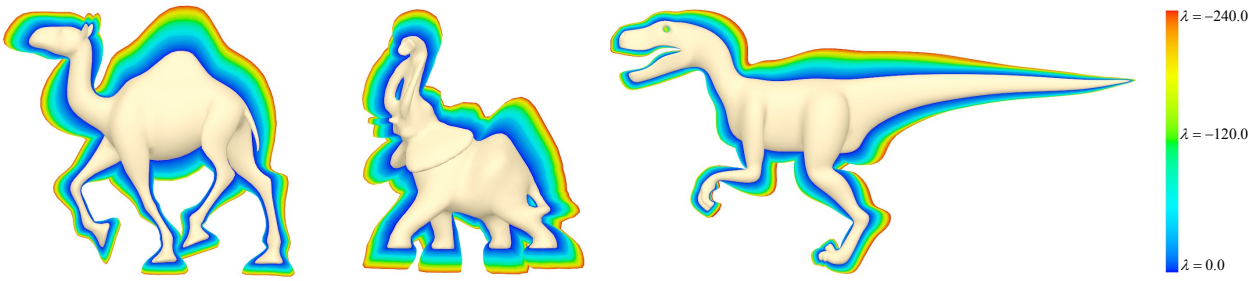


Figure **3-1**: A set of 48 successive shapes enhanced, from $\lambda = 0.0$ in blue to $\lambda = -240.0$ in red, with steps of -5.0.

This paper was accepted and presented in the Brazilian Symposium on Computer Graphics and Image Processing SIBGRAPI 2013. The conference was held on August 5-8 of 2013, in the city of Arequipa, Perú [33].

Abstract

This paper proposes a novel modeling method for a hybrid quad/triangle mesh that allows to set a family of possible shapes by controlling a single parameter, the global curvature. The method uses an original extension of the Laplace Beltrami operator that efficiently

estimates a curvature parameter, which is used to define an inflated shape after a particular operation performed in certain mesh points. Along with the method, this work presents new applications in sculpting and modeling, with the subdivision of surfaces and weight vertex groups. A series of graphics demonstrates the quality, predictability and flexibility of the method in a real production environment with software Blender.

keywords- laplacian smooth; curvature; sculpting; subdivision surface

3.1 Introduction

Over the last several years, modeling techniques that are able to generate a variety of realistic shapes, have been developed [7]. Editing techniques have evolved from affine transformations to advanced tools like sculpting [11, 17, 41], editing, creation from sketches [21, 19], and complex interpolation techniques [37, 45]. Catmull-Clark based methods however require interaction with a small number of control points for any operation to be efficient, or in other words, a unity condition is introduced by demanding a smooth surface after any of these shape operations. Hence, traditional modeling methods for subdividing surfaces from coarse geometry have become widely popular [9, 40]. These works have generalized a uniform B-cubic spline knot insertion to meshes, some of them adding some type of control, for instance with the use of creases to produce sharp edges [13], or the modification of some vertex weights to locally control the zone of influence [5]. Nevertheless, these methods are difficult to deal with since they require a large number of parameters and a very tedious customization. Instead, the presented method requires a single parameter that controls the global curvature, which is used to maintain realistic shapes, creating a family of different versions of the same object and therefore preserving the detail of the original model and a realistic appearance.

Interest in meshes composed of triangles and quads has lately increased because of the flexibility of modeling tools such as Blender 3D [6]. Nowadays, many artists use a manual connection of a couple of vertices to perform animation processes and interpolation [26]. It is then of paramount importance to develop operators that easily interact with such meshes, eliminating the need of preprocessing the mesh to convert it to triangles. The shape inflation and shape exaggeration can thus be used as a brush in the sculpting process, when inflating a shape, since current brushes end up losing detail when moving vertices [41]. In contrast, the presented method inflates a mesh by moving the vertices towards the reverse curvature direction, conserving the shape and sharp features of the model.

Contributions

This work presents an extension of the Laplace Beltrami operator for hybrid quad/triangle meshes, representing a larger mesh spectrum from what has been presented so far. The method eliminates the need of preprocessing and allows preservation of the original topology. Likewise, along with this operator, a method has been proposed to generate a family of parametrized shapes, in a robust and predictable way. This method enables customization of the smoothness and curvature obtained during the subdivision surfaces process. Finally, a new brush has been proposed for inflating the silhouette mesh features in modeling and sculpting.

This work is organized as follows: Section 3.2 presents works related to the Laplacian mesh processing, digital sculpting, and offsetting methods for polygonal meshes; in section 3.3, the theoretical framework of the Laplacian operator for polygon meshes is described; in section 3.4, the method for shape inflation and applications of subdivision of surfaces and sculpting is presented; finally some Laplacian operator results using hybrid quad/triangle meshes are shown graphically, as well as results of the shape inflation applications in sculpting, subdivision and modeling.

3.2 Related work

Many tools have been developed for modeling, based on the Laplacian mesh processing. Thanks to the advantages of the Laplacian operator, these different tools preserve the surface geometric details when being used for different processes such as free-form deformation, fusion, morphing and other applications [37].

Offset methods for polygon meshing, based on the curvature defined by the Laplace Beltrami operator, have been developed. With enough precision, these methods adjust the shape offset by a constant distance. Nevertheless, these methods fail to conserve sufficient detail because of the smoothing, a crucial issue which depends on the offset size [46]. In volumetric approaches, in the case of point-based representations, the offset boundary computation is based on the distance field, and therefore when calculating such offset the topology of the model may be different to the original [10].

[16] proposes automatic feature detection and shape edition with feature inter-relationship preservation. They define salient surface features like ridges and valleys, characterized by their first and second order curvature derivatives, see [28], and angle-based threshold. Likewise, curves have been also classified as planar or non-planar, approximated by lines, circles, ellipses and other complex shapes. In such cases, the user defines an initial change over several features which are propagated towards other features, based on the classified shapes and the inter-relationships between them. This method works well with objects that have

sharp edges, composed of basic geometric shapes such as lines, circles or ellipses. However, the method is very limited when models are smooth since it cannot find the proper features to edit.

Traditionally, digital sculpting has been approached under a polygonal representation or a voxel grid-based method. Brushes for inflation operations only depend on the vertex normal [41]. In grid-based sculpting, some other operations have allowed the addition or removal of voxels since production of polygonal meshes requires a processing of isosurfaces from volume [17]. The drawback comes from the difficulty of maintaining the surface details during larger scale deformations.

3.3 Laplacian Smooth

Computer objects, reconstructed from the real world, are usually noisy. Laplacian Smooth techniques allow a proper noise reduction on the mesh surface with minimal shape changes, while still preserving a desirable geometry as well as the original shape.

Many smoothing Laplacian functionals regularize the surface energy by controlling the total surface curvature S.

$$E(S) = \int_S \kappa_1^2 + \kappa_2^2 dS$$

where κ_1 and κ_2 are the two principal curvatures of the surface S.

3.3.1 Gradient of Voronoi Area

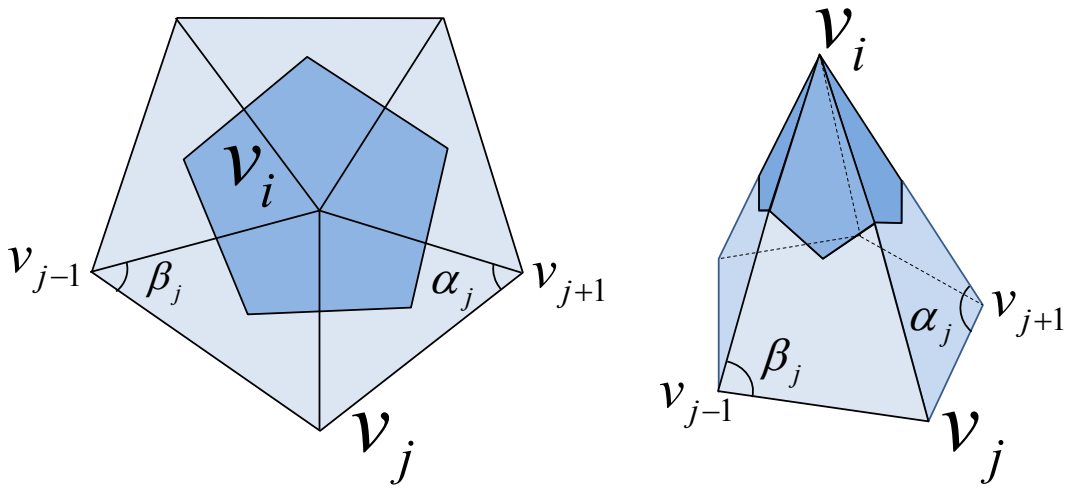


Figure **3-2**: Area of the Voronoi region around v_i in dark blue.v_j belong to the first neighborhood around v_i. α_j and β_j are opposite angles to edge $\overrightarrow{v_j - v_i}$.

Consider a surface S composed of a set of triangles around vertex v_i. Let us define the *Voronoi region* of v_i as shown in figure **3-2**. The area change produced by the movement of v_i is called the gradient of *Voronoi region* [29, 15, 25].

$$\nabla A = \frac{1}{2}\sum_{j} (\cot \alpha_j + \cot \beta_j) \tag{3-1}$$

If the gradient in equation (3-1) is normalized by the total area of the 1-ring neighborhood around v_i, the *discrete mean curvature normal* of a surface S is obtained, as shown in equation (3-2).

$$2\kappa \mathbf{n} = \frac{\nabla A}{A} \tag{3-2}$$

3.3.2 Laplace Beltrami Operator

The *Laplace Beltrami operator* LBO noted as $\triangle$ is used for measuring the mean curvature normal to the Surface S [29].

$$\triangle S = 2\kappa \mathbf{n} \tag{3-3}$$

The LBO has desirable properties: the LBO points to the reverse direction of the minimal surface area.

3.4 Proposed Method

This method exaggerates a shape using a Laplacian smoothing operator in the reverse direction, i.e., the new shape is a modified version in which those areas with larger curvature are magnified. The operator amounts to a generator of a set of models, which conserves the basic silhouette of the original shape. In addition, the presented approach can be easily mixed with traditional or uniform subdivision of surfaces. This method is based on an original extension of the Laplace Beltrami operator for hybrid quad/triangle meshes, mixing arbitrary types of meshes, exploiting the basic geometrical relationships and ensuring good results with few algorithm iterations.

3.4.1 Laplace Beltrami Operator for Hybrid Quad/Triangle Meshes TQLBO

Given a mesh $M = (V, Q, T)$, with vertices V, quads Q, triangles T. The area of 1-ring neighborhood $A(v_i)$ corresponds to a sum of the quad faces $A(Q_{v_i})$ and the areas of the triangular faces $A(T_{v_i})$ adjacent to vertex v_i.

$$A(v_i) = A(Q_{v_i}) + A(T_{v_i})$$

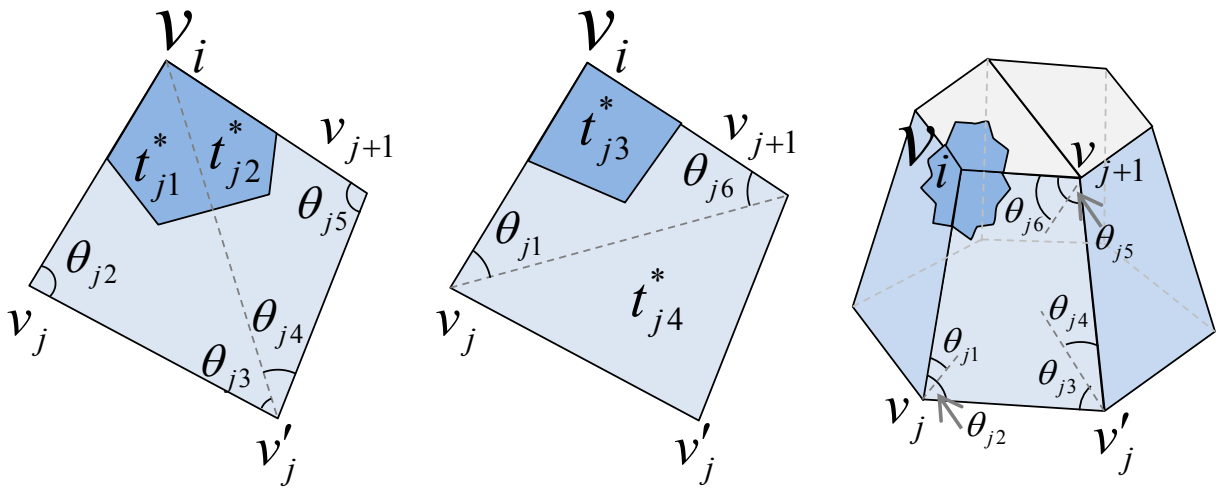


Figure **3-3**: $t^*_{j1} \equiv\triangle\ v_i v_j v'_j$, $t^*_{j2} \equiv\triangle\ v_i v'_j v_{j+1}$, $t^*_{j3} \equiv\triangle\ v_i v_j v_{j+1}$ Triangulations of the quad with common vertex v_i proposed by [Xiong 2011] to define Mean LBO.

Applying the mean average area, according to [43], from all possible triangulations, as show in figure **3-3**, the area for quads $A(Q_{v_i})$ and triangles $A(T_{v_i})$ is

$$A(v_i) = \tfrac{1}{2^m} \sum_{j=1}^{m} 2^{m-1} A(q_j) + \sum_{k=1}^{r} A(t_k)$$

where $q_1, q_2, ..., q_j, ..., q_m \in Q_{v_i}$ and $t_1, t_2, ..., t_k, ..., t_r \in T_{v_i}$

$$A(v_i) = \frac{1}{2} \sum_{j=1}^{m} \left[A\left(t^*_{j1}\right) + A\left(t^*_{j2}\right) + A\left(t^*_{j3}\right) \right] + \sum_{k=1}^{r} A(t_k) \tag{3-4}$$

Applying the gradient operator to (3-4)

$$\nabla A(v_i) = \tfrac{1}{2} \sum_{j=1}^{m} \left[\nabla A\left(t^*_{j1}\right) + \nabla A\left(t^*_{j2}\right) + \nabla A\left(t^*_{j3}\right) \right] + \sum_{k=1}^{r} \nabla A(t_k)$$

According to (3-1), we have

$$\nabla A\left(t^*_{j1}\right) = \frac{\cot\theta_{j3}(v_j - v_i) + \cot\theta_{j2}\left(v'_j - v_i\right)}{2}$$

$$\nabla A\left(t^*_{j2}\right) = \frac{\cot\theta_{j5}\left(v'_j - v_i\right) + \cot\theta_{j4}(v_{j+1} - v_i)}{2}$$

$$\nabla A\left(t^*_{j3}\right) = \frac{\cot\theta_{j6}(v_j - v_i) + \cot\theta_{j1}(v_{j+1} - v_i)}{2}$$

$$\nabla A\left(t_k\right) = \frac{\cot\alpha_k(v_k - v_i) + \cot\beta_{k+1}(v_{k+1} - v_i)}{2}$$

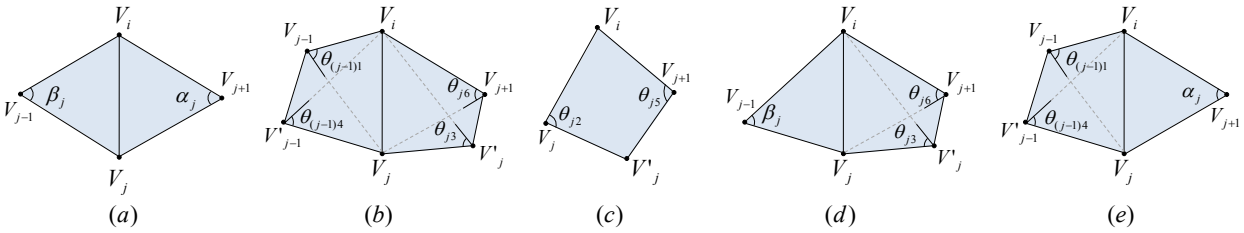


Figure **3-4**: The 5 basic triangle-quad cases with common vertex V_i and the relationship with V_j and V'_j. (a) Two triangles [Desbrun 1999]. (b) (c) Two quads and one quad [Xiong 2011]. (d) (e) Triangles and quads (TQLBO) our contribution.

Triangle and quad configurations of the 1-ring neighborhood faces, adjacent to v_i, can be simplified to five cases, as shown in figure **3-4**.

According to equation (3-2), (3-3), and five simple cases defined in figure **3-4** the TQLBO (Triangle-Quad LBO) of v_i is

$$\triangle_S(v_i) = 2\kappa\mathbf{n} = \frac{\nabla A}{A} = \frac{1}{2A}\sum_{v_j \in N_1(v_i)} w_{ij}\left(v_j - v_i\right)$$

$$w_{ij} = \begin{cases} \left(\cot\alpha_j + \cot\beta_j\right) & \text{case } a. \\ \frac{1}{2}\left(\cot\theta_{(j-1)1} + \cot\theta_{(j-1)4} + \cot\theta_{j3} + \cot\theta_{j6}\right) & \text{case } b. \\ \left(\cot\theta_{j2} + \cot\theta_{j5}\right) & \text{case } c. \\ \frac{1}{2}\left(\cot\theta_{j3} + \cot\theta_{j6}\right) + \cot\beta_j & \text{case } d. \\ \frac{1}{2}\left(\cot\theta_{(j-1)1} + \cot\theta_{(j-1)4}\right) + \cot\alpha_j & \text{case } e. \end{cases} \tag{3-5}$$

We define a TQLBO as a matrix equation

$$L(i,j) = \begin{cases} -\frac{1}{2A_i} w_{ij} & \text{if } j \in N(v_i) \\ \frac{1}{2A_i} \sum_{j \in N(v_i)} w_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3-6}$$

where L is a $n \times n$ matrix, n is the number of vertices of a given mesh M, w_{ij} is the TQLBO defined in equation (3-5), $N(v_i)$ is the 1-ring neighborhood with a shared face to v_i and A_i is the ring area around v_i.

Normalized equation of the TQLBO

$$L(i,j) = \begin{cases} -\frac{w_{ij}}{\sum_{j \in N(v_i)} w_{ij}} & \text{if } j \in N(v_i) \\ \delta_{ij} & \text{otherwise} \end{cases} \tag{3-7}$$

where δ_{ij} is the Kronecker delta function.

3.4.2 The Shape Inflation

The shape is inflated using the reverse direction of the curvature flow, moving the vertices towards those mesh portions with larger curvature. A standard diffusion process is applied:

$$\frac{\partial V}{\partial t} = \lambda L(V)$$

To solve this equation, implicit integration is used as well as a normalized version of TQLBO matrix

$$(I - |\lambda dt| W_p L) V' = V^t \tag{3-8}$$

$$V^{t+1} = V^t + \text{sign}(\lambda)(V' - V^t)$$

The vertices V^{t+1} are inflated, along their reverse curvature direction, by solving the linear system: $Ax = b$, where $A = I - |\lambda dt| W_p L$, L is the Normalized TQLBO defined in the equation (3-7), $x = V'$ are the smoothing vertices, $b = V^t$ are the actual vertices positions, W_p is a diagonal matrix with vertex weights, and λdt is the inflate factor that supports negative and positive values: negative for inflation and positive for smoothing.

The method was devised to use with weighted vertex groups, which specify the final shape inflation of the solution, meaning 0 for no changes and 1 when a maximal change is applied. The weights modify the influence zones, where the Laplacian is applied, as shown in equation

3-8 . Interestingly, the generated family of shapes may change substantially with the weights of specific control points.

The curvature cannot be calculated at the boundary of the meshes that are not closed, for that reason we use the scale-dependent operator proposed by Desbrun et al. [15], the inflation factor for boundary is represented by λ_e.

The model volume increases proportionally as the lambda becomes larger and negative, this can be counteracted with a simple volume preservation. However, the mesh may suffer large displacements when $\lambda < -1.0$ or after multiple iterations. A simple volume conservation algorithm is: if v_i^{t+1} is a mesh vertex of V^{t+1} in the $t+1$ iteration, we define $\overline{v}$ as:

$$\overline{v} = \frac{1}{n} \sum_{v_i \in V} v_i$$

$\overline{v}$is the mesh center, vol_{ini} is an initial volume, and vol_{t+1} is the volume at the iteration $t+1$, n is the number of vertices, then the scale factor

$$\beta = \left(\frac{vol_{ini}}{vol_{t+1}} \right)^{\frac{1}{3}}$$

allows to scale the vertices to:

$$v_{i\,new}^{t+1} = \beta \left(v_i^{t+1} - \overline{v} \right) + \overline{v}$$

The shape inflation uses a negative curvature flow that is an unstable process when performing many iterations, however, our method uses less than 3 iterations to get good results, and with 3 iterations or less the method behaves in a stable way.

3.5 Sculpting

A new sculpting brush is herein proposed and aims to inflate the shape, magnifying the shape curvatures of a polygon mesh in real time. This brush works best with the stroke method *Drag Dot*, allowing the user to pre-visualize the model changes before the mouse is released. Also, it allows movement of the mouse along the model to match the shape zone which is supposed to be inflated.

Brushes that perform a similar inflation can introduce mesh distortions or produce mesh self-intersections, provided that these brushes only move the vertices along the normal without any global information. In contrast, the present method searches for an improved inflation while preserving the global curvature, retaining the original shape and main model features. In addition, this method simplifies the work required for the inflation since it does not

need different brushes for inflating, softening or styling. The inflated brush can do all these operations in a single step. Real-time brushes require the Laplacian matrix to be constructed with the vertices that are within the sphere radius defined by the user, reducing the matrix to be processed. The center of this sphere depends on the place where the user clicks on the canvas and also where the click is projected on the three-dimensional mesh. Special handling is required for the boundary vertices with neighbors that are not within the brush radius: these vertices mark the boundary though the curvature is not calculated there, but they must be included in the matrix so that every vertex has their corresponding neighbors within the selection. The sculpting Laplacian matrix reads as.

$$L(i,j) = \begin{cases} -\frac{w_{ij}}{\sum\limits_{j \in N(v_i)} w_{ij}} & \text{if } \|v_i - u\| < r \wedge \|v_j - u\| < r \\ 0 & \text{if } \|v_i - u\| < r \wedge \|v_j - u\| \geq r \\ \delta_{ij} & \text{otherwise} \end{cases}$$

where $v_j \in N(v_i)$, u is the sphere center of radius r. The matrices should remove rows and columns of vertices that are not within the radius.

3.6 Subdivision surfaces

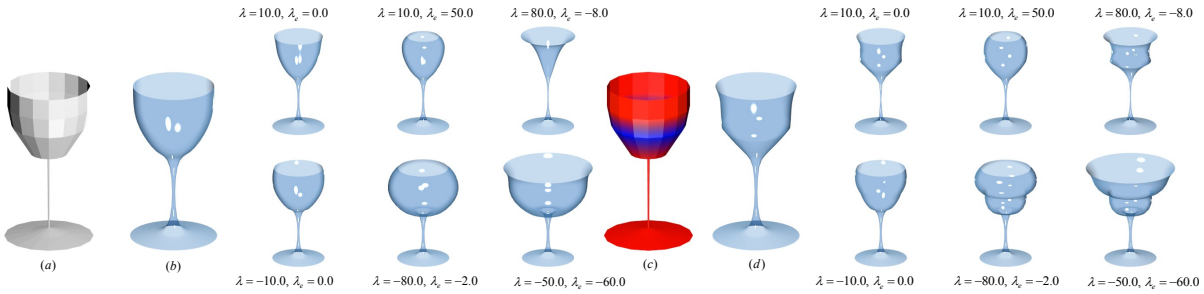


Figure **3-5**: Family of cups generated with our method, from a coarse model (a), (c): the shape, obtained from the Catmull-Clark Subdivision (b), (d), is inflated. Soft constraints, over the coarse model, is drawn in red and blue (c).

The Catmull-Clark subdivision transformation is used to smoothen a surface, as the limit of a sequence of subdivision steps [40]. This process is governed by a B-spline curve [24], performing a recursive subdivision transformation that refines the model into a linear interpolation that approximates a smooth surface. The model smoothness is automatically guaranteed [13].

Catmull-Clark subdivision surface methods generate smooth and continuous models from a coarse model and produce quick results because of the simplicity of implementation. Nevertheless, changes to the global curvature are hardly implantable. The Catmull-Clark Subdivision Surfaces, together with shape inflation, can easily generate families of shapes by changing a single parameter, allowing a model with very few vertices to be handled. In practice, this would allow an artist to choose a model from a similar set of options that would meet their needs without having to change each of the control vertices. Likewise, the presented method allows the use of vertex weight paint over the control points. The weights can be applied to a coarse model, followed by a Catmull-Clark subdivision where weights are interpolated, producing weights with smooth changes in the influence zones, as shown in figure **3-5**.c.

In equation 3-8, W_p is a diagonal matrix with weights corresponding to each vertex. Weights at each vertex produce a different solution so that the matrix must be placed in the diffusion equation, since families that are generated may change substantially with the weight of specific control points.

3.7 Results

The results of the shape inflation method with the extension of the Laplace Beltrami operator for hybrid quad/triangle meshes with several example models shown in figures: **3-1**, **3-5**, **3-6**, **3-7**, **3-8**, **3-10**, **3-11**, **3-12**, **3-9**. The shape inflation was assessed with TQLBO method on a PC with AMD Quad-Core Processor @ 2.40 GHz and 8 GB RAM.

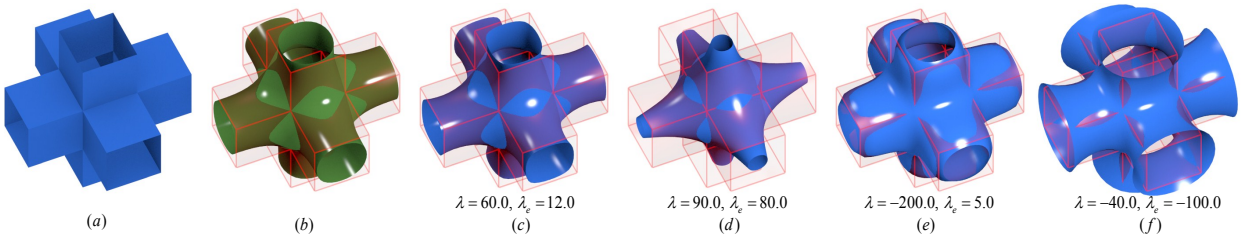


Figure **3-6**: (a) Original Model, (b) Model with Catmull-Clark Subdivision. Models with Laplacian smoothing: (c) and (d). Models with a first Laplacian filtering $\lambda = 60.0$, $\lambda_e = 12.0$ and before applying shape inflation: (e) and (f).

Figure **3-7** shows the results when applying the Laplace Beltrami Operator TQLBO of equation (3-6) in a model with a simple subdivision. In column (c) the Laplacian smoothing was applied to a model consisting of only quads. In column (d) the model was converted to triangles and then the Laplacian smoothing was applied. In column (e) the model was

randomly converted from some quads into triangles and then the Laplacian smoothing was applied, showing similar results to those meshes composed only of triangles or quads.

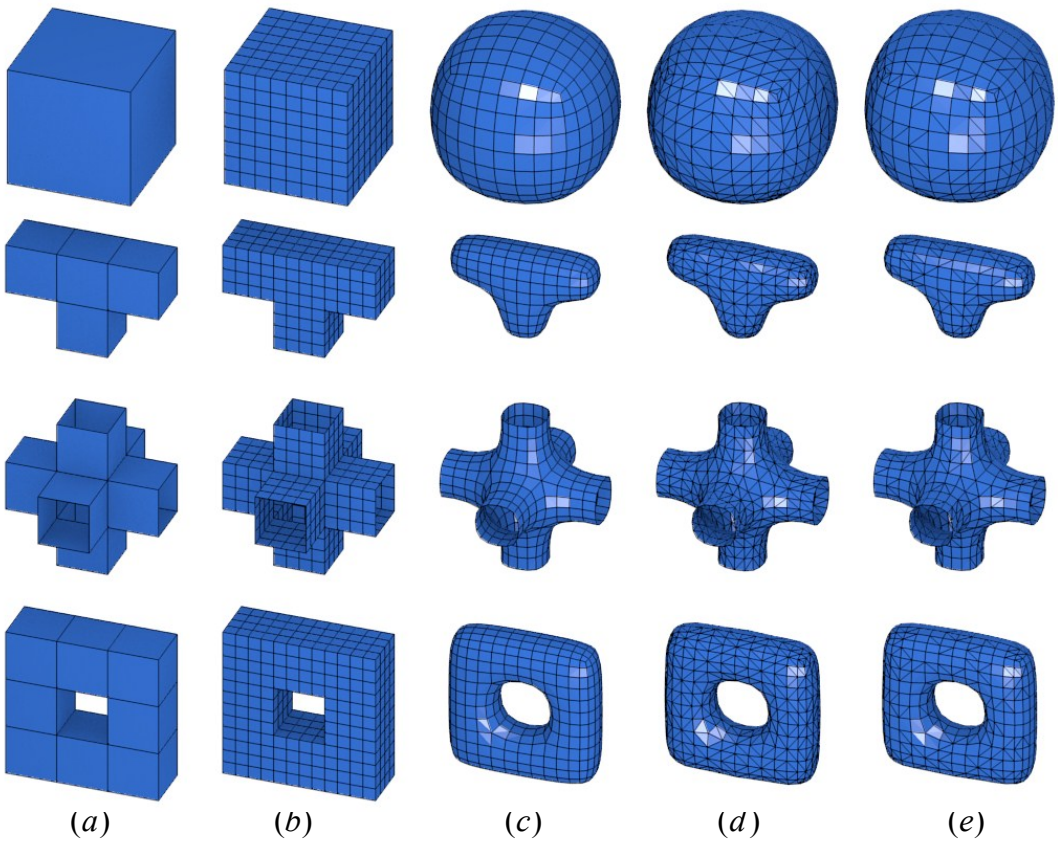


Figure **3-7**: (a) Original Model. (b) Simple subdivision. (c), (d) (e) Laplacian smoothing with $\lambda = 7$ and 2 iterations: (c) for triangles, (d) for quads, (e) for triangles and quads chosen randomly.

Methods using the Catmull-Clark Subdivision Surface and inflation allow the modification of the curvature, as shown in figure **3-5**. This test used a coarse cup model, in which the subdivision was performed, followed by Laplacian smoothing and inflation. Figure **3-5**.c, **3-5**.d also shows the use of weight vertex groups over coarse models, with subdivision surfaces that generate weights for the new interpolated vertices. These new weights were used for the inflation obtained on the 6 cups that are at the right of the figure **3-5**.d.

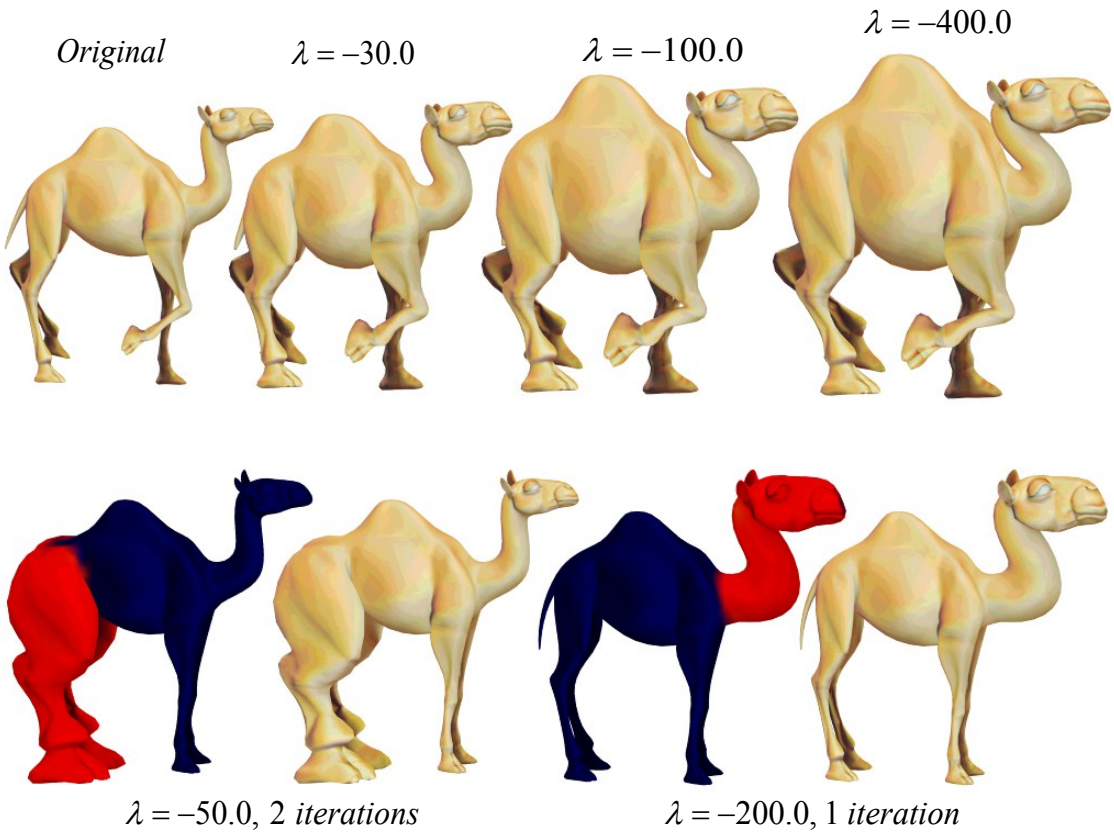


Figure **3-8**: Top row: Original camel model in left. Shape inflation with $\lambda = -30.0$, $\lambda = -100.0$, $\lambda = -400.0$. Bottom row: Shape inflation with weight vertex group, $\lambda = -50.0$ and 2 iterations for the legs, $\lambda = -200.0$ and 1 iteration for the head and neck.

Laplacian smoothing applied with simple subdivision (see figure **3-6**.c.) may produce similar results to those obtained with Catmull-Clark (see figure **3-6**.b.), whose models have average equal triangles. The one obtained with the Laplacian smoothing is shown in panel (c), (d) and the curvature modified versions are in (e) and (f). As can be observed, different versions of the original sketch can be obtained by parameterizing a single model value, a great advantage of the presented method. Figure **3-8** shows the generation of different versions of a camel according to the λ parameter. In the top row the shape inflation results are shown. As λ becomes larger and negative, the resultant shape was inflated on the more convex parts, as shown in figure **3-1**. The larger the λ parameter, the larger the model feature inflations. The bottom row of figure **3-8** shows the use of weighted vertex groups, specifying which areas will be inflated. On the left, the inflation of the camel legs produces an organic aspect, notice that the border is not distorted and smooth.

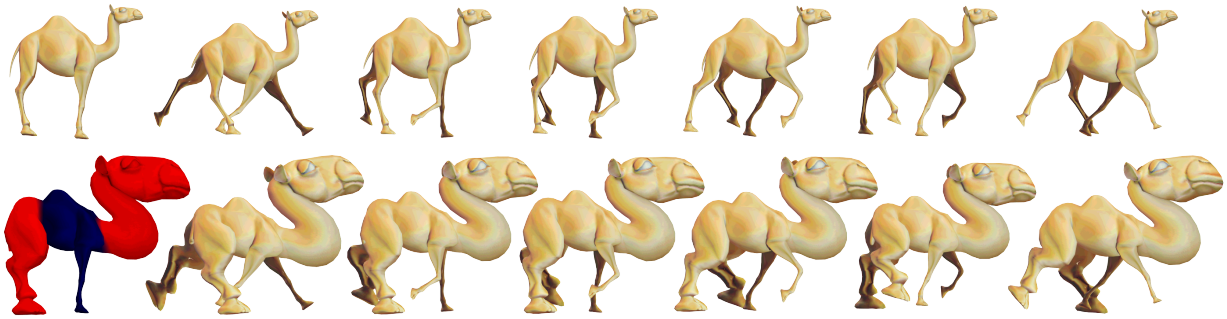


Figure **3-9**: The method is pose insensitive. The inflation for the different poses are similar in terms of shape. Top row: Original walk cycle camel model. Bottom row: Shape inflation with weight vertex group, $\lambda = -400$ and 2 iterations.

The inflation of the silhouette's features is predictable and invariant under isometric transformations, like those classically used in some animations (see Figure **3-9**). In this figure, the animation shows some camel poses during a walk. The inflation is performed at the neck and legs, as shown in the bottom left camel in figure **3-9**. Local modifications produced by the pose interpolation or animation rigging practically do not affect the result. There is a clear difference despite the pose of the camel's legs. The inflation method allows a flesh-like shape in the original pattern produced by the artist, this is due to the mesh restricted diffusion process so that small local changes are treated without affecting the global solution. The method therefore is rotation invariant since it depends exclusively on the normal mesh field.

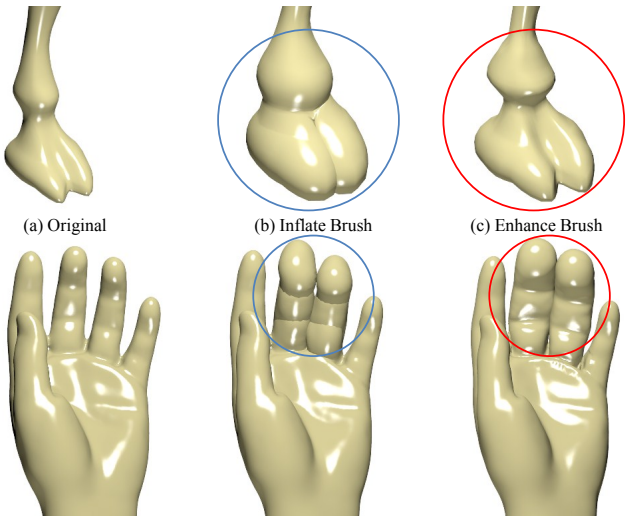


Figure **3-10**: Top row: (a) Original camel leg, (b) Inflate Brush used on leg within blue circle, (c) Enhance Brush used on leg within red circle. Bottom row: (a) Original hand, (b) Inflate Brush used on fingers within blue circle, (c) Enhance Brush used on fingers within red circle.

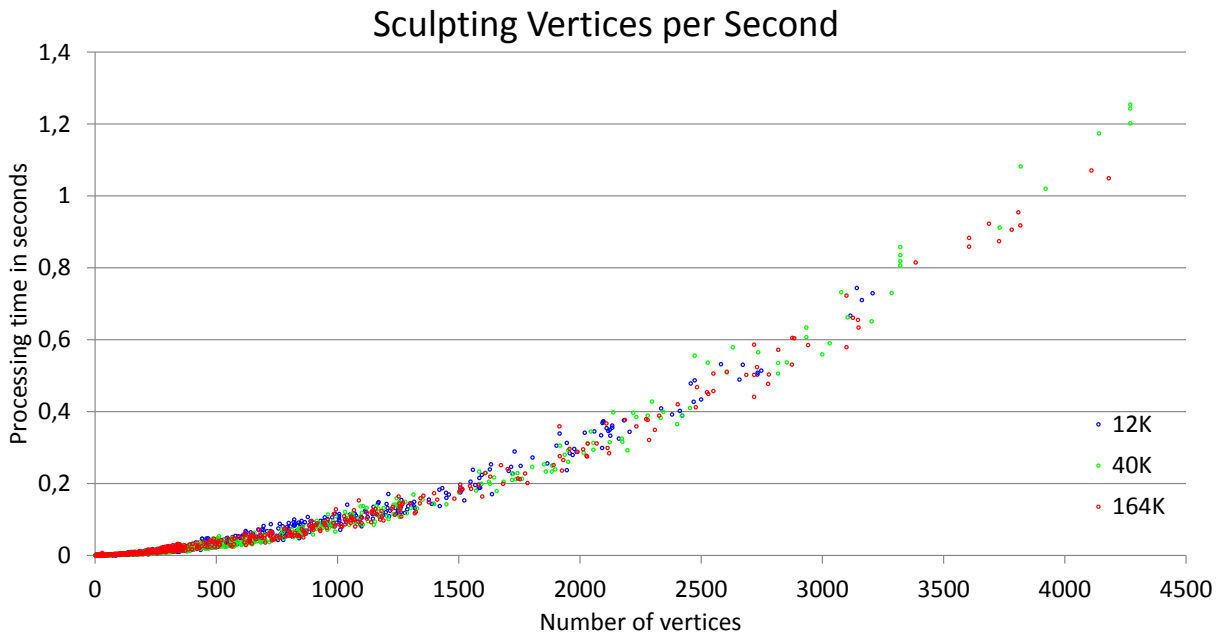


Figure **3-11**: Performance of our dynamic Enhance Brush in terms of the sculpted vertices per second. Three models with 12K, 40K, 164K vertices used for sculpting in real time.

Figure **3-10** shows the use of the Enhance Brush for sculpting in real time. One pass was used with the brush, shown by the blue and red radius. In figure **3-10**.b the camel hoof shows the inflation intersection, which looks like two bubbles, a similar pattern that is observed on the fingers on the bottom row of the same figure. The silhouette inflation is observed in figure **3-10**.c the main shape is retained together with either its finger or hoof details. Similar results can be obtained by using different brushes, however it would take several steps, while the Enhance Brush for shape inflation takes only a single step. For this reason, this new method can easily inflate organic features like muscles during the sculpting process. In figure **3-11** the Enhance Brush performance is illustrated. In this experiment three models with 12K, 40K and 164K vertices, were used. These models were sculpted with the Enhance Brush, each time the user selected a variable number of vertices for processing. The processing time for 800 vertices in the camel hoof (40k model) only took 0.1 seconds, for 2600 vertices in the leg and neck (model 40k) it took 0.5 seconds. These times are suitable in real applications since an artist sculpts a model by parts and each part is represented by an average of about 1800 vertices.

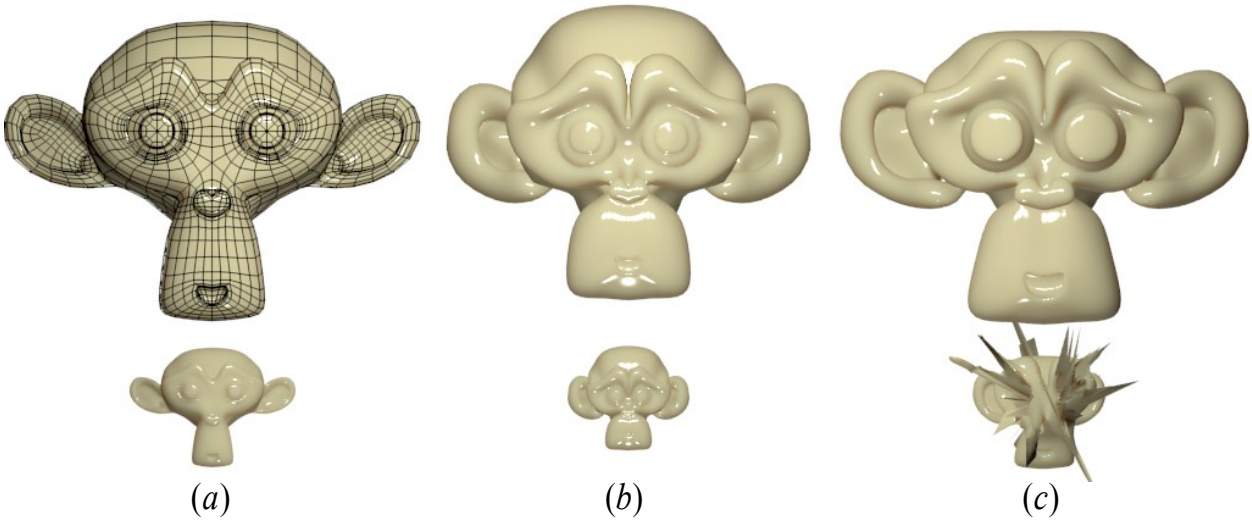


Figure **3-12**: (a) Bottom row: Original Model. Top row: Original model scaled by 4. (b) Top and bottom row: inflated with Normalized-TQLBO $\lambda = -50$ from (a) respectively (c) Top and bottom row: inflated with TQLBO $\lambda = -50$ from (a) respectively.

Tests with the Laplacian operator (equation 3-6) and its normalized version (equation 3-7), produce similar results if the triangles or quads that compose the mesh are about the same size. The normalized version is more stable and predictable because it is not divided by the area of the ring, which may be very small and cause numerical problems, as shown in figure **3-12**.c bottom row. The shape inflation of the model with the normalized Laplacian operator results in a more regular pattern. The model can be deformed with a normalized version of TQLBO with large λ ($\lambda > 400$) that can intersect itself but without any peaks. Figure **3-12**.c shows different results due to the quads' areas in the model. Quads with larger areas have smaller inflations (figure **3-12**.c skull), and smaller quads have larger inflations (figure **3-12**.c chin).

3.8 Implementation

The method was implemented as a modifier for modeling and brush for sculpting, on Blender software [6] in C and C++ language programming. Working with Blender allowed the method to be tested interactively against other methods, such as Catmull-Clark, Weight Vertex Groups and Sculpting System.

To improve the performance, it was worked with the Blender mesh structure, visiting each triangle or quad and storing its corresponding index and the sum of the Laplacian weights of the ring in a list so that only two visits were required for the list of mesh faces and two

times for the edge list, if the mesh was not closed. This drastically reduced calculations, enabling real-time processing. In the construction of the Laplacian matrix, several indices were locked at vertices, which had face areas or edge lengths with zero value that could cause spikes and bad results.

Under these conditions, the matrix of the equation 3-6 is sparse since the number of neighbors per vertex, corresponding to the number of data per row, is smaller compared to the total number of vertices in the mesh. To solve the linear system equation 3-8 OpenNL [8] was used, which is a a library for solving sparse linear systems.

3.9 Conclusion and future work

This work presented an extension of the Laplace Beltrami operator for hybrid quad/triangle meshes that can be used in production environments and provides results similar to those obtained by working only with triangles or quads. This paper has introduced a new way to change silhouettes in a mesh for modeling or sculpting in a few steps by means of the curvature model modification while preserving its overall shape. In addition, a new modeling method has also been presented and some possible applications have been illustrated. The method works properly with isometric transformations, opening the possibility of introducing it to the process of animation.

We show that this tool may work in early modeling stages, when coarse models are used, allowing the shape generated by the Catmull-Clark subdivision surfaces to be modified and thereby avoiding edition of the vertices with a change of a single parameter.

Future work includes the analysis of theoretical relationships between the Catmull-Clark subdivision surfaces and the Laplacian smoothing since they can produce very similar results.

Acknowledgment

We would like to thank anonymous friends for their support of our research.

This work was supported in part by the Blender Foundation, Google Summer of Code program in 2012.

Livingstone elephant model is provided courtesy of INRIA and ISTI by the AIM@SHAPE Shape Repository. The Hand model is courtesy of the FarField Technology Ltd. The Camel model by Valera Ivanov is licensed under a Creative Commons Attribution 3.0 Unported License. Dinosaur and Monkey models are under public domain, courtesy of Blender Foundation.

4 Mesh smoothing based on curvature flow operator in a diffusion equation

This work was accepted as part of the Blender [6] software, an open source 3D application for modeling, rendering, composing, video editing and game creation. The work was supported by an awarded internship of the Google Summer of Code 2012 program, administered by Google Inc.

4.1 Synopsis

Objects reconstructed from the real world contain undesirable noise in many computer graphics applications. A Mesh smoothing may remove that noise while still preserving the geometry and shape of the original model. This project aims to improve the mesh smoothing tools uses by Blender software, using curvature flow operator in a diffusion equation, allowing hybrid meshes composed of triangles and quads to be worked with, and using the Laplacian operator proposed by Pinzón and Romero [33].

4.2 Benefits to Blender

This project proposes a new and robust mesh smoothing tool that improves the appearance of the surfaces of models. Usually, methods to scan computer graphics objects using the Kinect ZCam need to remove the noise present at the time of capture. This mesh smoothing method produces higher quality results without shrinkage, while the smoothing tool currently used collapses after several iterations.

This mesh smoothing method allows hard and soft constraints on the positions of the mesh points in order to maintain control over the shape, which facilitates the removal of noise generated during the sculpting, but without eliminating the desired details of the model.

4.3 Deliverables

- A new and robust mesh smoothing tool for Blender.
- Some documentation pages to be included in the manual.
- A technical document for developers to improve the method in the future.
- A tutorial explaining the use of the tool.

4.4 Project Details

The mesh smoothing algorithm was implemented as a diffusion equation for specific geometric structures. The project was divided in four parts:

1. Initialization of data and necessary structures.
2. Computation of the Laplacian Matrix.
3. Definition of the sparse linear system.
4. Solution of the sparse linear system, using a preconditioned bi-conjugated gradient numerical library.

Integration of the numerical library present in Blender to solve the sparse linear system

Generation of documentation and tutorials.

4.5 Project Schedule

- 3 weeks: Understanding the Blender source code and identifying the key points for the project.
- 1 week: Definition of the data structures necessary to work with the Blender architecture.
- 1 week: Implementation of the methods for the initial configuration of the smoothing algorithm. Implementation of the Laplacian matrix calculation.
- 2 weeks: Integration of the numerical library.
- 2 weeks: Formulation of the sparse linear system and implementation of the numerical method to solve it.
- 3 weeks: Formulation and implementation of the graphical user interface.
- 2 weeks: Testing the tool.
- 3 weeks: Generation of the documentation and tutorials.

4.6 Mesh Smoothing

A common way of attenuating noise in a polygonal mesh is by a diffusion process [42, 15]. Laplacian smooth techniques allow a proper noise reduction of the mesh surface with minimal shape changes. The simple idea consists in moving the vertices in the Laplacian direction. When the cotangent version is used, the vertices are moved in the direction of the curvature flow. The complexity of the Laplacian smoothing can be linear in time and space with a fast convergence. The diffusion process can attenuate noise with only one iteration due the sparseness of the laplacian operator.

$$\frac{\partial V}{\partial t} = \lambda L\left(V\right) \tag{4-1}$$

Where L is the Laplacian matrix defined in equation 4-3 for meshes composed of triangles or quads with different sizes or irregular sampling and λ is a scalar that controls the diffusion process, and smoothing factor. The equation 4-1 can be linearly approximated using implicit integration with a Laplacian Operator version of TQLBO, the use of implicit integration makes the system more stable.

Computation of the Laplacian Matrix:

1. Definition of the sparse linear system.
2. Solution of the sparse linear system, using a preconditioned bi-conjugated gradient numerical library.

$$\left(I - \lambda dtL\right) V^{n+1} = V^{n} \tag{4-2}$$

The user may define the region of interest where the Laplacian smooth needs to be applied. For doing so, we add a diagonal matrix W_p to equation 4-2, where every element in the diagonal corresponds to the weight for every vertex.

$$\left(I - \lambda dtW_pL\right) V^{n+1} = V^{n}$$

For non-closed meshes or meshes with holes, the curvature flow cannot be computed. For this reason, the system smoothes out the edges only in the direction of the diffusion process. The boundaries are treated as a one-dimensional curve, where the Laplacian is defined as the weighted difference between the vertex and the two immediate neighbors, ensuring the curve maintains its original form as much as possible. We define a Laplacian for mesh smoothing as a matrix equation.

$$L(i,j) = \begin{cases} -\frac{1}{2A_i} w_{ij} & \text{if } j \in N(v_i) \wedge v_i \notin \text{Boundary} \\ \frac{1}{2A_i} \sum_{j \in N(v_i)} w_{ij} & \text{if } i = j \wedge v_i \notin \text{Boundary} \\ -e_{ij} & \text{if } j \in N(v_i) \wedge \{v_i, v_j\} \in \text{Boundary} \\ \frac{2}{E_i} \sum_{j \in N(v_i)} e_{ij} & \text{if } i = j \wedge \{v_i, v_j\} \in \text{Boundary} \\ 0 & \text{otherwise} \end{cases} \tag{4-3}$$

where L is a $n \times n$ matrix, n is the number of vertices of a given mesh M, w_{ij} is the TQLBO defined in equation (3-5), $N(v_i)$ is the 1-ring neighborhood which has a shared face with v_i, $e_{ij} = \frac{1}{\|v_i - v_j\|}$ is the inverse length of the edge between vertices $\{v_i, v_j\}$, $E_i = \sum_{j \in N(v_i)} e_{ij}$. A_i is the ring area around v_i.

4.7 Results and Conclusions

The developed user interface can be seen in figure **4-1**. This tool allows the λ parameters for inner points and boundaries to be set, as well as to configure soft constraints using weights defined by vertices in "Vertex Group" and also to set strong constraints by independently applying the algorithm in the axis X, Y or Z.

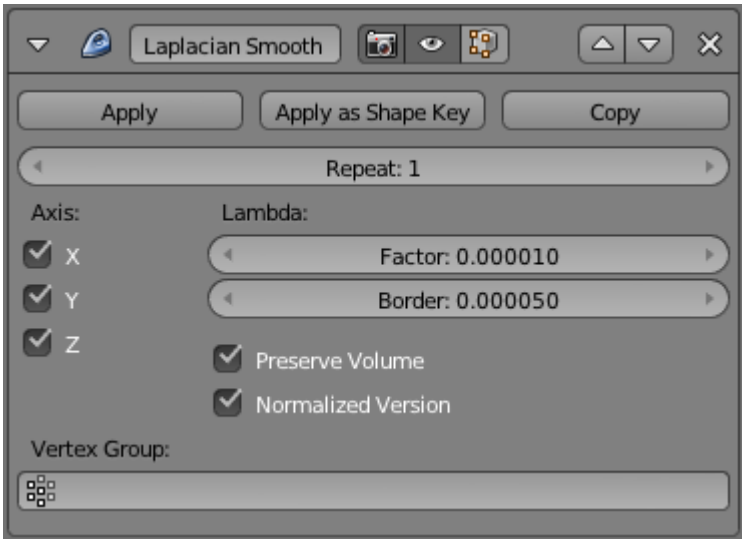


Figure **4-1**: Panel inside blender user interface of the Laplacian Smooth modifier tool.

The tool developed can set the λdt parameter of equation 4-2. Using a small Lambda factor ($\lambda < 1.0$), noise can be removed without significantly affecting the geometry (see figure **4-**

2.b), while using a large Lambda factor ($\lambda > 1.0$) smoothed versions can be obtained at the cost of losing fine geometry details (see figure **4-2**.c and **4-2**.d).

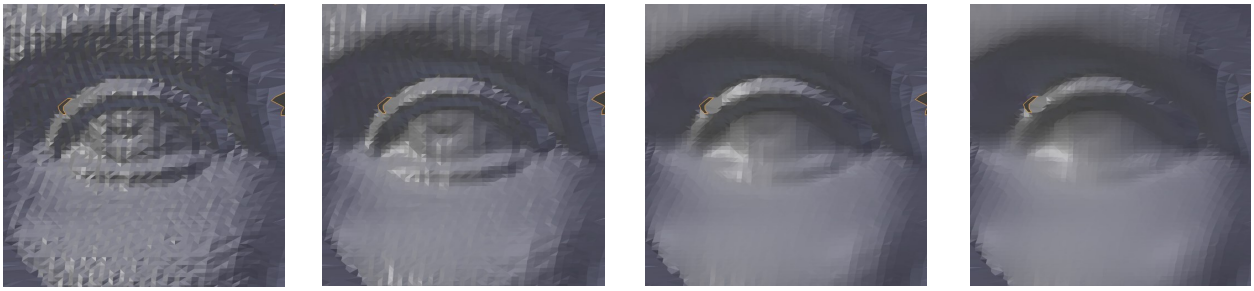


Figure **4-2**: Noise attenuation in face model with Laplacian smoothing tool using only one iteration and changing λ. (a) Original Model. (b) Smoothing $\lambda = 0.5$. (c) Smoothing $\lambda = 2.5$ (d) Smoothing with $\lambda = 5.0$.

The user can smooth the boundaries by configuring the parameter "*Border*", seen in figure **4-1**. Boundaries are treated differently since there is no way to calculate the boundary curvature flow. For this reason the Lambda factor "*Border*" just smooths them. The change of this parameter and the results seen in figure **4-3**, illustrate how the boundary inside the red circle is smoothed.

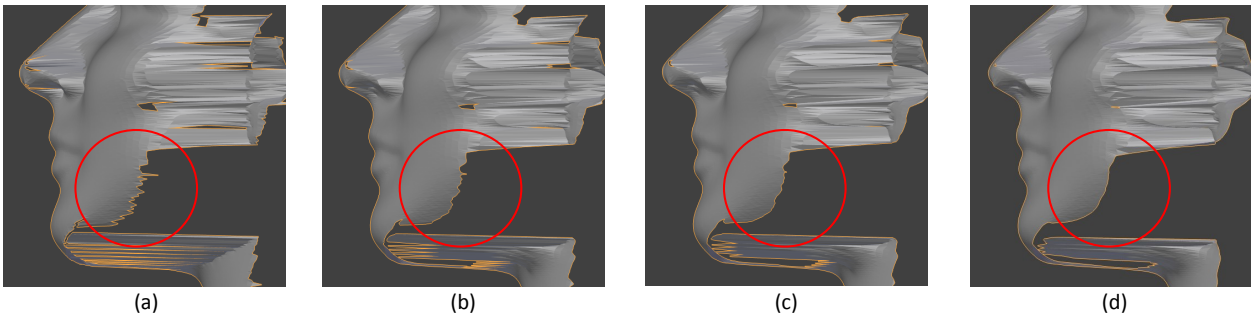


Figure **4-3**: Smoothing boundary changing λ_{Border} factor. (a) Original Model. (b) Smoothing $\lambda_{Border} = 1.0$. (c) Smoothing $\lambda_{Border} = 2.5$ (d) Smoothing with $\lambda_{Border} = 10.0$.

The tool allows the user to add soft constraints using weights for each vertex, this allows regions of interest where you want to apply the algorithm to be defined. In figure **4-4**.c, the red region corresponds to the desired region and figure **4-4**.d stands for those smoothened vertices in the red region.

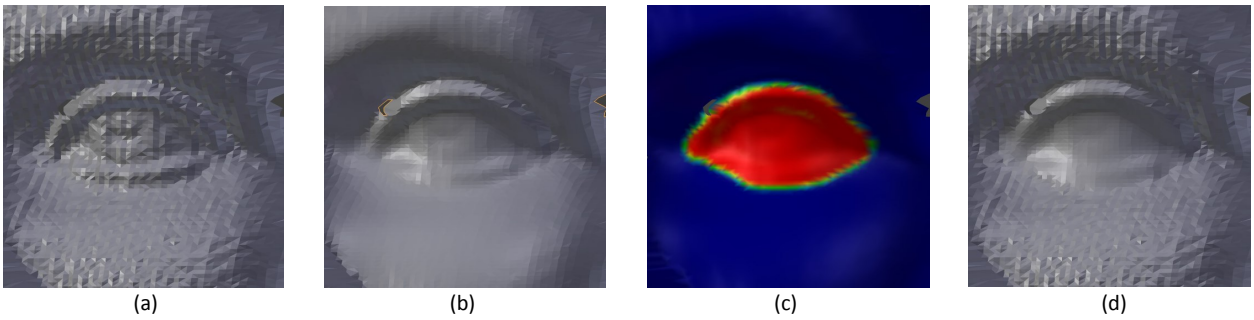


Figure **4-4**: Use of weights per vertex to constrain the effect of mesh smoothing. (a) Original Model. (b) Smoothing with $\lambda = 1.5$ (c) red vertices $weight = 1.0$, blue vertices $weight = 0.0$. (d) Smoothing with $\lambda = 2.5$. The red vertices were the only vertices smoothed.

This module was developed as a tool for Blender software to remove noise in the most efficient way. The state-of-the-art methods remove noise only if the mesh is composed by triangles, in contrast, with the developed tool, artists can now remove the noise even if the mesh is composed of triangles and quadrangles.

5 Mesh Editing with Laplacian Deform

This work was accepted and completed for Blender [6] software, which is an open source 3D application for modeling, animation, rendering, composing, video editing and game creation. In the Google Summer of Code 2013 program which was administered by Google Inc.

5.1 Synopsis

The mesh editing is generally done with affine transformations. Blender3D offers some tools that can transform vertices, such as "proportional editing object mode" with which the transformation of some vertices is interpolated with the other vertices that are connected with the use of simple distance functions.

This project proposes to implement a method for mesh editing based on sketching lines defined by the user and preserving the geometric details of the surface.

This method captures the geometric details using differential coordinates representations. The differential coordinates captures the local geometric information (curvature and direction) of the vertex based on its neighbors. This method allows you to retrieve the best possible original model after changing the positions of some vertices by using the differential coordinates of the original model.

5.2 Benefits to Blender

This project proposes a new tool for Blender users that requires the preservation of geometric details of the surface during modeling, transformation and definition of the shape keys of the mesh vertices.

The method will allow novice users to edit any polygon mesh preserving the surface details.

This method allows the user to define new shape keys in a faster and more intuitive way.

5.3 Deliverables

- A new mesh editing tool for Blender.
- Some pages of documentation to be included in the manual
- A technical document for developers to improve the method in the future.
- A tutorial explaining the use of the tool.

5.4 Project Details

The project is divided into eight parts:

1. Calculate the differential coordinates.
2. Store the fixed vertices (Hard constraints).
3. Store positions of the edited vertices.
4. Store the most representative vertex to retrieve rotation of every differential coordinate.
5. Solve the initial solution – in least-squares sense.
6. Rotate the differential coordinates based on initial solution and the most representative vertex.
7. Reconstruct the surface – in least-squares sense.
8. Generation of the documentation and tutorials.

5.5 Project Schedule

- 2 Weeks: Calculate the differential coordinates.
- 2 Weeks: Store the fixed vertices (Hard constraints).
- 2 Weeks: Store positions of the edited vertices.
- 2 Weeks: Compute initial solution.
- 2 Weeks: Rotate differential coordinates.
- 2 Weeks: Reconstruct the surface – in least-squares sense.
- 1 Week: Testing the tool and Define and implement graphical user integration.
- 2 Weeks: Generation of the documentation and tutorials.

5.6 Laplacian Deform

The Laplacian deformation facilitates mesh visualization while preserving the geometric surface details. In this method the user defines a set of anchor vertices which serve to repose the mesh by translating some of them. The system keeps the anchor vertices in fixed positions and calculates the best possible locations of the other vertices so that the shape preserves the original geometric details. This work adapts the method proposed by Sorkine et al. [37] for mesh deformations by deleting the use of static vertices. The method has also been applied to hybrid meshes composed of triangles and quads, using the proposed TQLBO. In particular, the geometric details are captured using differential coordinates representations that contain the local geometric information (curvature and direction) of the vertex and its neighbors, as shown in figure **5-1**.

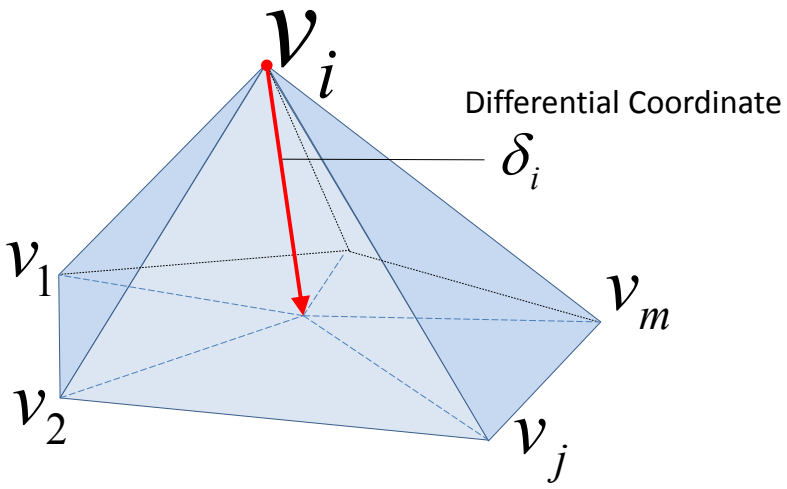


Figure **5-1**: Difference between v_i and the center of mass of its neighbors $v_1, ..., v$.

$$\delta_i = \sum_{j=1}^{m} w_{ij} \left(v_i - v_j\right) \tag{5-1}$$

where δ_i is the differential coordinate for vertex v_i. The v_j are the immediate neighbors of v_i, and w_{ij} is the weight between vertex v_i and v_j defined in equation 3-5 that is TQLBO.

Then the linear system for finding the new pose of a mesh is.

$$\begin{bmatrix} w_l L \\ W_c \end{bmatrix} X = \begin{bmatrix} \delta \\ W_c C \end{bmatrix} \tag{5-2}$$

Where w_l is the Laplacian Matrix weight L, and the Laplacian matrix L was defined in equation 3-6 . Wc is a matrix that has only ones in the indices of anchor vertices. C is a

vector with coordinates of anchor vertices after several transformations. δ are the differential coordinates defined in equation 5-1.

5.7 Testing Solvers

For this project we chose a numerical solver to be included in the Blender software after an evaluation of the initial factorization of the Laplacian deformation system.

Linear equation system to solve

$$\begin{bmatrix} w_l L \\ W_c \end{bmatrix} X = \begin{bmatrix} \delta \\ W_c C \end{bmatrix}$$

Solving the sparse linear system

$$Ax = b$$

Where:

$$A = \begin{bmatrix} w_l L \\ W_c \end{bmatrix}$$

$$x = V$$

$$b = \begin{bmatrix} \delta \\ W_c C \end{bmatrix}$$

5.7.1 Hardware Specification

- Processor: AMD Quad-Core 2.40 GHz
- RAM: 8.0 GB
- OS: Windows 7 Professional
- Graphics controller: NVIDIA Quadro FX 570

5.7.2 Software Specification

CGAL Computational Geometry Algorithms Library

Graphite Research platform for computer graphics

5.7.3 Numeric Solvers Used

CG: Conjugate gradient method.

BICGSTAB: Biconjugate gradient stabilized method.

GMRES: Generalized minimal residual method.

SUPERLU: Sparse Direct Solver, LU decomposition with partial pivoting.

TAUCS_LDLT: A library of sparse linear solvers with LDLT factorization.

CHOLMOD: Supernodal sparse Cholesky factorization.

LU factorization is a numerical method that works with large, sparse, non-symmetric systems of linear equations [22]. We chose the implementation of LU factorization in an OpenNL-SuperLU library because this method showed better performance in the computation of a solution for a Laplacian Deform linear system of equations presented in equation 5-2 which can be seen in the table and plots **5-1** and **5-2**. OpenNL SuperLU can function with the Graphics Unit Processor GPU in order to exploit the capacity of GPU to be able to work with parallel structures, faster than a traditional CPU.

Model	Vertices	CG	BICGSTAB	GMRES	SUPERLU	TAUCS	CHOLMOD
Cross	24	0.05	0.05	0.04	0.04	0.05	0.05
King	538	0.83	0.63	0.71	0.61	0.68	0.79
YModel	4770	19.60	16.44	16.93	16.06	16.88	17.95
Man	10002	33.43	27.76	29.91	28.54	29.53	30.80
Neptune	28052	133.97	136.46	136.39	133.21	142.87	142.76
Armadillo	34594	194.48	174.88	175.80	169.92	181.70	183.49

Table **5-1**: Vertices Vs Seconds, Laplacian Deform initial factorization performance.

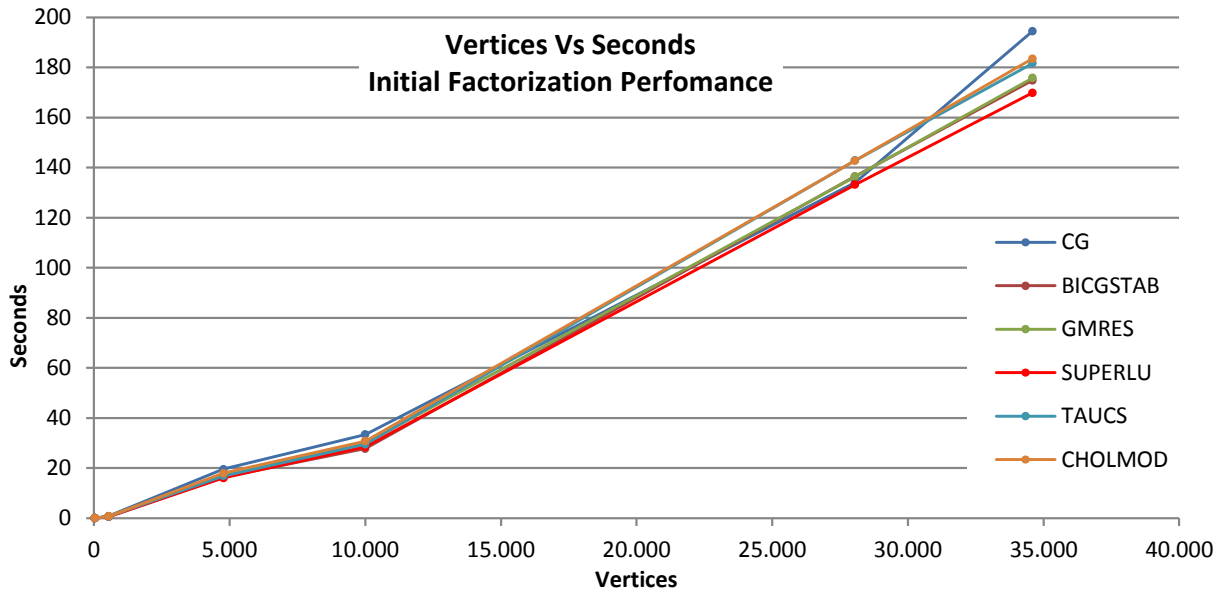


Figure **5-2**: Plot of Vertices Vs Seconds, Initial factorization performance.

5.8 Results

The user interface for Blender software can be seen in figure (**5-3**). In this tool, the user defines the anchor vertices and, with the use of a vertex, groups the features offered by Blender. The user configures this name in the field *Anchors Vertex Group*. The *Bind* option initiates the system and captures the geometry details in the form of differential coordinates and computes the factorization of the linear system, after that the system is ready to pose meshes in a real-time interaction session.

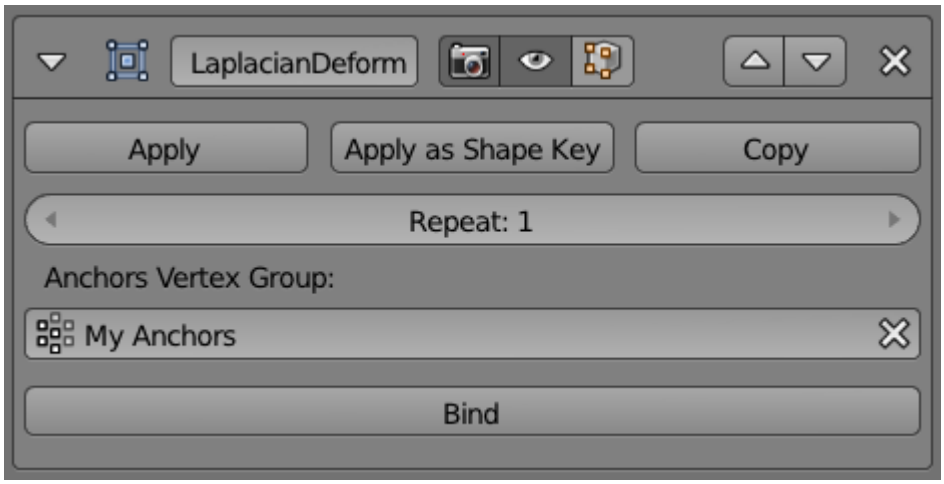


Figure **5-3**: Panel inside Blender user interface of the Laplacian Deform modifier tool.

Figure (**5-4**) shows the Laplacian Deformation applied to a model with 173K vertices, only anchor vertices were used and are represented in blue. When the user applies several transformations (location, rotation, scale) to these anchor-vertices, the system finds a solution and estimates the position of the vertices (in yellow). This method works in real time and for doing so the matrix $\begin{bmatrix} w_l L \\ W_c \end{bmatrix}$ in equation (5-2) is LU decomposed only once, with LU factorization, when the system initiates. Once the matrix is factorized, the system can solve the unknown variables in the order of milliseconds. Results are improved by solving the system of equations several times, without LU factorization at every iteration, just the differential coordinates are adjusted since the differential coordinates can be rotated. Only four iterations were necessary for obtaining figure (**5-4**) but the system finds proper solutions with only one iteration, when the angle of rotations are less thanπ.

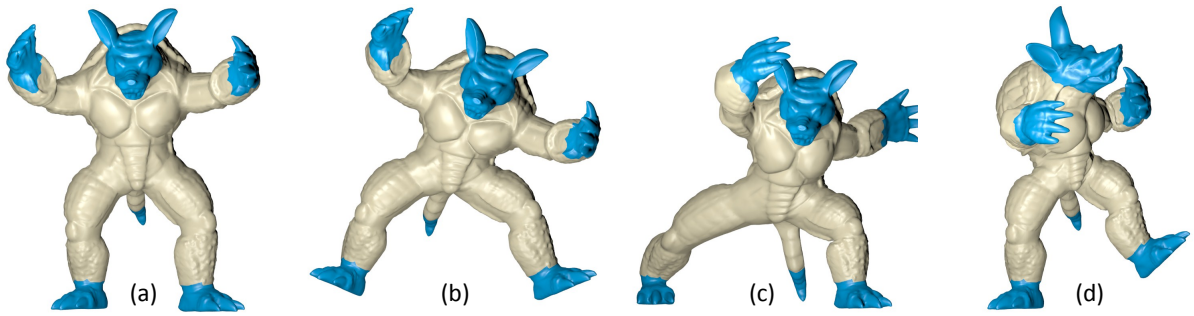


Figure **5-4**: Anchor vertices in blue. (a) Original Model, (b,c,d) new poses only change the anchor-vertices, the system finds positions for vertices in yellow.

Figure **5-5** shows a comparison after making a single transformation that rotates the parts in blue 70^{o} to the right. Results using a bicubic interpolation are shown in **5-5**.b observe how the main trunk loses its shape when rotated. The propagation of changes made by the Laplacian deformation (figure **5-5**.c) for the same transformation, shows better results.

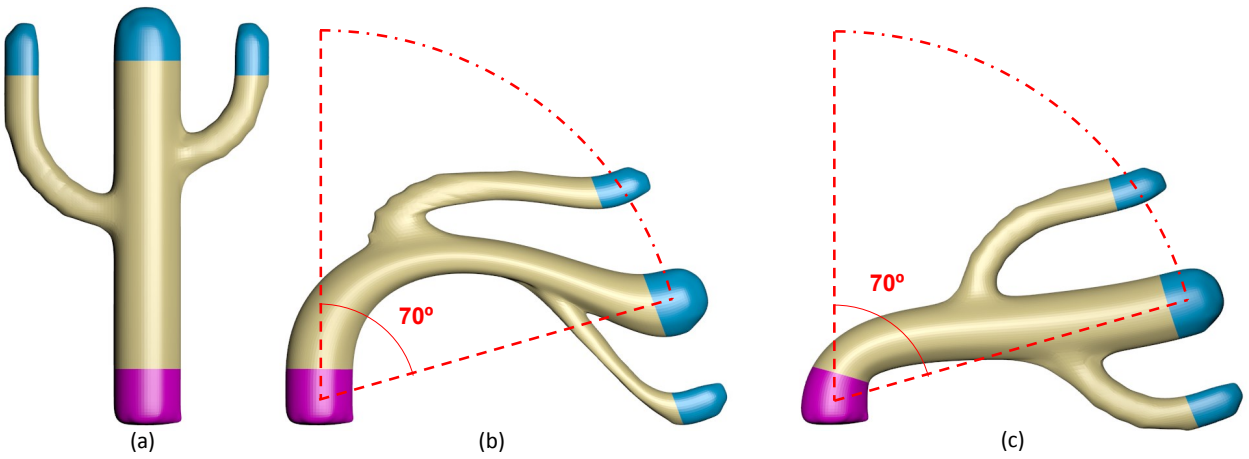


Figure **5-5**: (a) Original cactus model. (b) Blue segments are rotated 70^{o} to the right and afterwards a basic interpolation is applied to the parts in yellow (c) Blue segments are rotated 70^{o} to the right and afterwards a Laplacian deform tool is applied to the parts in yellow.

The Laplacian Deformation tool allows the user to change the model's pose while preserving the geometry details. In figure **5-6**.b and **5-6**.c observe the horse's new pose after five transformations and one head rotation. In figure **5-6**.b the shape and details are lost at every change using basic interpolation. In constrast, figure **5-6**..c illustrates how the new pose of the horse looks more natural, above all for the body and neck. This comparison indicates that the Laplacian Deform method allows any transformation to be applied without any loss of detail.

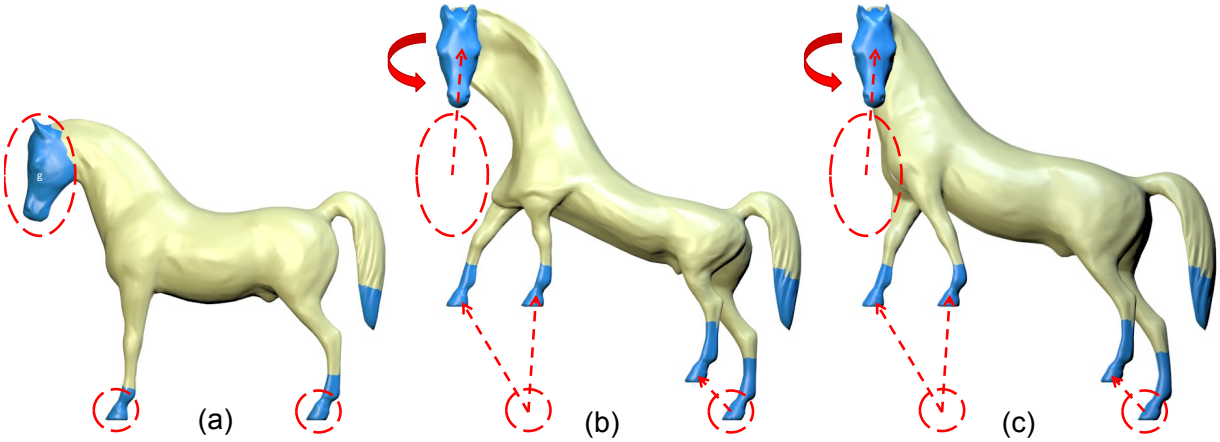


Figure **5-6**: (a) Original Horse model. (b) The blue segments are translated and rotated and then basic interpolation is applied to the yellow parts (c) The blue segments are translated and rotated and then the Laplacian Deform tool is applied to the yellow parts.

6 Skeleton Extraction

Part of this work was accepted and presented as a poster titled *Software para la Extracción del Esqueleto por Contracción y Suavizado* [Software for Skeleton Extraction by Contraction and Smoothing] at 7th International Seminar on Medical Image Processing and Analysis SIPAIM 2011. The conference was held on December 6-8 of 2011, in Bucaramanga, Colombia [31], the poster thumbnail is shown in figure **6-1**.

Part of this work was accepted and presented as a poster titled *Análisis Experimental de la Extracción del Esqueleto por Contracción con Suavizado Laplaciano* [Experimental Analysis of Skeleton Extraction by Contraction and Laplacian Smoothing] at the 6th International Seminar on Medical Image Processing and Analysis SIPAIM 2010. The conference was held on December 1-4 of 2010, in the city of Bogotá, Colombia [30], the poster thumbnail is shown in figure **6-2**.

6.1 Background

Skeleton extraction not only reduces the dimensionality but also represents a three-dimensional object as a uni-dimensional structure [12].

Skeleton extraction uses the natural shrink produced by Laplacian smoothing [25], to iteratively contract the mesh until the volume inside the surface is about zero (see figure **6-3**). This method stretches the mesh to guarantee that the object preserves, as much as possible, the original topology constraint [3].

Software para la Extracción del Esqueleto por Contracción y Suavizado

Alexander Pinzón, Eduardo Romero

Abstract

Este articulo presenta un software para el procesamiento, visualización, y extracción del esqueleto desde mallas de polígonos. El software se diseño con base en un sistema de plugins y filtros, se implemento un plugin que contenía un filtro para la extracción del esqueleto por contracción en dirección gradiente con suavizado Laplaciano. El software producido proporciona una plataforma flexible para el diseño e implementación de plugins.

Métodos de Suavizado de Mallas

Los métodos para suavizar mallas reducen el ruido, o permiten iterativamente eliminar frecuencias altas presentes en el muestreo tridimensional de los modelos.

Métodos Laplacianos

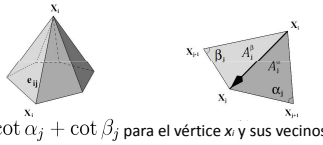
La idea básica consiste en mover un vértice en la misma dirección del Laplaciano . $Eq(1)\quad \frac{\partial X}{\partial t} = \lambda L(X)$

La ecuación 1 se implementa como la ecuación de diferencias hacia adelante así: $Eq(2)\quad X_{t+1} = (I + \lambda L)X_t$
Donde ***X*** es el conjunto de vértices, *L* es el Laplaciano, y λ ∈ $\Re$ es la velocidad de difusión.

Y la aproximación discreta de la ecuación 2 es:

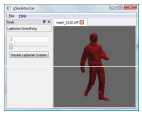
$$Eq(3)\quad L(x_i) = \sum w_{ij}(x_j - x_i),\quad x_j \in Vecinos(x_i)$$

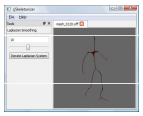
Aproximación del Laplaciano mediante la Curvatura normal



Con $w_{ij} = \cot\alpha_j + \cot\beta_j$ para el vértice x_i y sus vecinos x_j

Software Skeletonizer





sjSkeletonizer es el software desarrollado en el grupo Bioingenium para el procesamiento, visualización y extracción del esqueleto desde mallas de polígonos.

- Usa **CGAL** (Computational Geometry Algorithms Library)
- Usa **Graphite** (Software de Geometría Numérica y Computación Grafica)
- Se integraron las siguientes librerías de procesamiento numérico: ACE, AMD, ARPACK, ARPACK_UTIL, CBLAS, CCOLAMD, CHOLMOD, CLAPACK, COLAMD, F2CLIBS, METIS, MISC, NL, SUPERLU, TAUCS

Implementación para la extracción del esqueleto

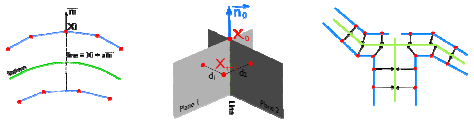
La Esqueletonización reduce la dimensionalidad y representa un cuerpo como un estructura uní-dimensional.

El esqueleto puede ser obtenido suavizando la malla pero bajo dos restricciones, W_L que da peso al Laplaciano y W_H que mantiene los vértices en su localización original.

Extracción del esqueleto. $\begin{bmatrix} W_L L \\ W_H \end{bmatrix} X_{t+1} = \begin{bmatrix} 0 \\ W_H X_t \end{bmatrix}$

Donde *L(X)* = Suavizado Laplaciano con w_{ij} basado en la curvatura de flujo $w_{ij} = \cot\alpha_j + \cot\beta_j$

Y la nueva restricción propuesta en este trabajo



Tratar de suavizar los vértices a lo largo de la línea
La distancia del punto a la linea

$$line = \overrightarrow{P_1} + t\overrightarrow{P_2},\ point\ P_0 \Longrightarrow distance(line, P_0) = \frac{|(P_2 - P_1)\times(P_1 - P_0)|}{|P_2 - P_1|}$$

Cada punto en un plano satisface esta ecuación

$$P_0 : a\mathbf{x} + b_0\mathbf{y} + c_0\mathbf{z} + d_0 = 0.$$

Resultados

- Los vértices se pueden mover a lo largo de la línea.
- El esqueleto tiene muchas ramas.
- Muchas mas ecuaciones que incógnitas.
- La solución debe ser restringida a una región particular de la línea.

Referencias y Agradecimientos

• Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. **Skeleton extraction by mesh contraction**. *ACM Transactions on Graphics*, 27(3):10, 2008. Skeleton Extraction.
•Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. **Articulated mesh animation from multi-view silhouettes**. *ACM Trans. Graph.*,27(3):1–9, 2008. 3D Reconstruction.
•Nicu D. Cornea and Patrick Min. **Curve-skeleton properties, applications, and algorithms**. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007. Skeleton Extraction Survey Member-Silver, Deborah.

Agradecimientos: The captured performance data were provided courtesy of the Computer Graphics Group of the MIT CSAIL Vision Research (Cambirdge, USA).

Contacto

Alexander Pinzón Fernández apinzonf@gmail.com
Grupo de Investigación Bioingenium www.bioingenium.unal.edu.co
Universidad Nacional de Colombia www.unal.edu.co
Facultad de Medicina, Edificio 471 Primer Piso

BioIngenium Research Group

UNIVERSIDAD NACIONAL DE COLOMBIA

Figure **6-1**: Poster *Software para la Extracción del Esqueleto por Contracción y Suavizado* [Software for Skeleton Extraction by Contraction and Smoothing] presented at the 7th International Seminar on Medical Image Processing and Analysis SIPAIM 2011.

Análisis Experimental de la Extracción del Esqueleto por Contracción con Suavizado Laplaciano

Alexander Pinzón, Fabio Martínez, Eduardo Romero

Abstract

Este articulo presenta un análisis sistemático del método de extracción del esqueleto por medio de la contracción de un volumen con suavizado Laplaciano. El trabajo realiza una aproximación experimental al problema de la extracción del esqueleto, para encontrar el rendimiento del método evaluado frente a cambios isométricos, y durante la fase de simplificación.

Esta evaluación utilizo el modelo tridimensional animado de una persona que realizaba una caminata, a este modelo se le extrajo el esqueleto y se compararon las diferencias en distintos instantes de la animación, y distintas configuraciones del proceso de simplificación. Los resultados muestran un óptimo rendimiento del método frente a las transformaciones isométricas, y múltiples problemas en la fase de simplificación de mallas.

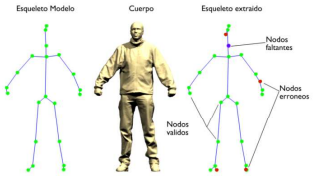


Fig. 2. Proceso de extracción del esqueleto

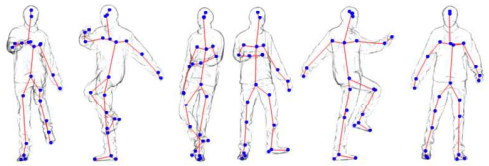


Fig. 3. Variación del numero de nodos

Contracción con suavizado Laplaciano

Este método contrae iterativamente una malla de polígonos por medio del suavizado laplaciano hasta tener un volumen de cero ver figura 1.
La contracción es tomada como un problema de minimización de energía, con los siguientes términos.

$$\|W_L L V'\|^2 + \sum_i W_{H,i}^2 \|V'_i - V_i\|^2$$

- L: Operador Lapaciano para remover las frecuencias altas, es decir suavizar los detalles de la geometría
- W_H: Fuerza de atracción que usa los vértices, para mantener información clave de la geometría durante la contracción.
- W_L: Fuerza de contracción que hace que la forma tridimensional pierda volumen.

Fig. 1. Proceso de extracción del esqueleto

Experimentación

Se uso la implementación hecha en Oscar Kin-Chung Au et al. Por los autores, para realizar la extracción del esqueleto de un modelo tridimensional que fue obtenido mediante el método de recuperación de forma desde las siluetas realizado por Vlasic et al. De este modelo se registro una caminata durante 240 cuadros.
Para evaluar el proceso de simplificación se variaron el numero de nodos usados para describir uniones en el esqueleto (ver figura 3) y se clasificaron las uniones así ver figura 2.

En el segundo experimento se seleccionaron diferentes poses para observar la correspondencia topológica entre los esqueletos extraídos y analizar el comportamiento del método frente a transformaciones isométricas de la geometría de un cuerpo.

Resultados

El método recupera de forma óptima el esqueleto de un cuerpo bajo transformaciones isométricas, ver la tabla 1, se recuperaron en promedio 19.71 nodos de los 21 necesarios para reconstruir el esqueleto en diferentes poses.
El método no pudo recuperar 1.86 nodos de los necesarios para reconstruir totalmente el esqueleto. La línea verde en la figura 3 describe el numero de nodos que hacen falta para recuperar el modelo.

	Validos	Erróneos	Faltantes
Media	19.71	4.83	1.86
SD	1.47	1.47	1.47
Min	17	3	0
Max	21	7	4

Tabla 1. Resultados obtenidos al realizar transformaciones isométricas

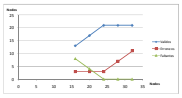


Fig. 3. Gráfico de la variación de nodos validos, erróneos y faltantes durante la fase de simplificación

Durante la fase de simplificación (ver figura 3) el método no sufre perdida ni es sensible al uso de mas de 24 nodos (línea azul). El método tiene un numero mínimo de 3 nodos recuperados erróneamente como se observa el la línea roja de la figura 3.

Referencias y Agradecimientos

• Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. **Skeleton extraction by mesh contraction**. *ACM Transactions on Graphics*, 27(3):10, 2008. Skeleton Extraction.
•Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. **Articulated mesh animation from multi-view silhouettes**. *ACM Trans. Graph.*,27(3):1–9, 2008. 3D Reconstruction.
•Nicu D. Cornea and Patrick Min. **Curve-skeleton properties, applications, and algorithms**. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007. Skeleton Extraction Survey Member-Silver, Deborah.

Agradecimientos: The captured performance data were provided courtesy of the Computer Graphics Group of the MIT CSAIL Vision Research (Cambirdge, USA).

Conclusiones y Trabajo Futuro

El método de extracción mostró ser robusto y tener baja sensibilidad frente a cambios isométricos de la geometría, el método puede trabajar de forma automática a o largo de todos los cuadros. El método recupera de forma eficiente el esqueleto sin hacer uso de un modelo, eliminando la necesidad de estimar la pose. El método permite realizar de forma sencilla y automática el seguimiento del esqueleto a lo largo del vídeo.

Como trabajo futuro es posible mejorar la recuperación de información haciendo uso de la coherencia espacio temporal no presente en la técnica de extracción, para superar la perdida de información entre cuadros de vídeo. Es posible automatizar el proceso de simplificación para encontrar el numero óptimo de nodos con lo cual puede ser representado el esqueleto, haciendo uso de algoritmos de partición de mallas.

Contacto
Alexander Pinzón Fernández apinzonf@gmail.com
Grupo de Investigación BioIngenium www.bioingenium.unal.edu.co
Universidad Nacional de Colombia www.unal.edu.co
Facultad de Medicina, Edificio 471 Primer Piso

Figure **6-2**: Poster *Análisis Experimental de la Extracción del Esqueleto por Contracción con Suavizado Laplaciano* [Experimental Analysis of Skeleton Extraction by Contraction and Laplacian Smoothing] presented at the 6th International Seminar on Medical Image Processing and Analysis SIPAIM 2010.

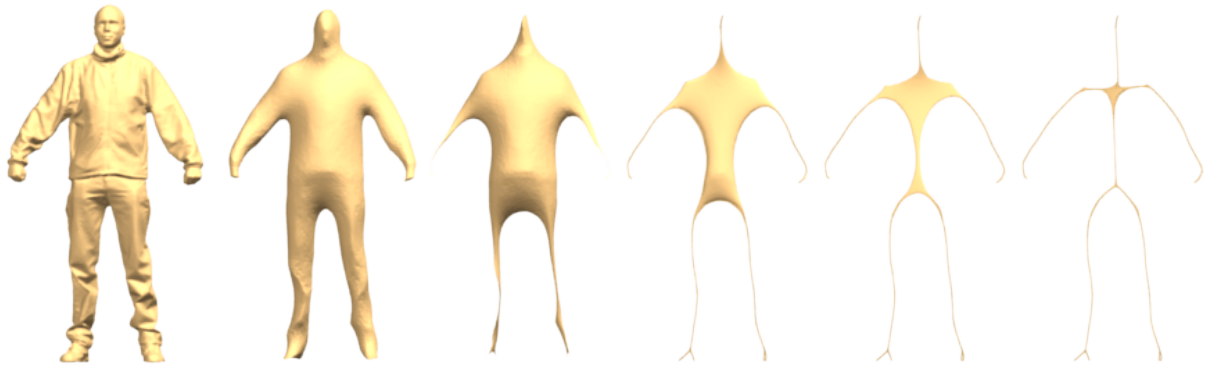


Figure **6-3**: From left to right iterative mesh contraction.

The Laplacian smooth process moves the vertices in the direction of the Laplacian. If the cotangent Laplacian operator is used, the vertices move in the direction of minimal surface [29] following the curvature flow of the mesh surface.

Au et al. [3] propose the next system of equations to iteratively contract the mesh until the volume is zero and a skeleton appears.

$$\begin{bmatrix} W_L L \\ W_H \end{bmatrix} X_{t+1} = \begin{bmatrix} 0 \\ W_H X_t \end{bmatrix} \tag{6-1}$$

where L is the Laplacian matrix described in equation 3-6, W_L is the smoothing factor and W_H is the attraction constraint factor.

W_L and W_H change at every iteration. The contraction and smoothing constraint $W_L^{t+1} = S_L W_L^t$ with $S_L = 2.0$. The attraction constraint $W_{H,i}^{t+1} = W_{H,i}^0 \sqrt{\frac{A_i^0}{A_i^t}}$.

A_i^t y A_i^0 are the current area and initial area of the ring surrounding x_i.

6.2 Contribution

This work proposes an additional constraint: it seeks to move the vertices along a normal line. This constraint eliminates the need to adjust the final skeleton, taking each skeleton node and moving it to the center of the local mesh region.

The basic idea is to move the vertices along a normal line, estimated at every vertex, based on the average of the normals of the faces.

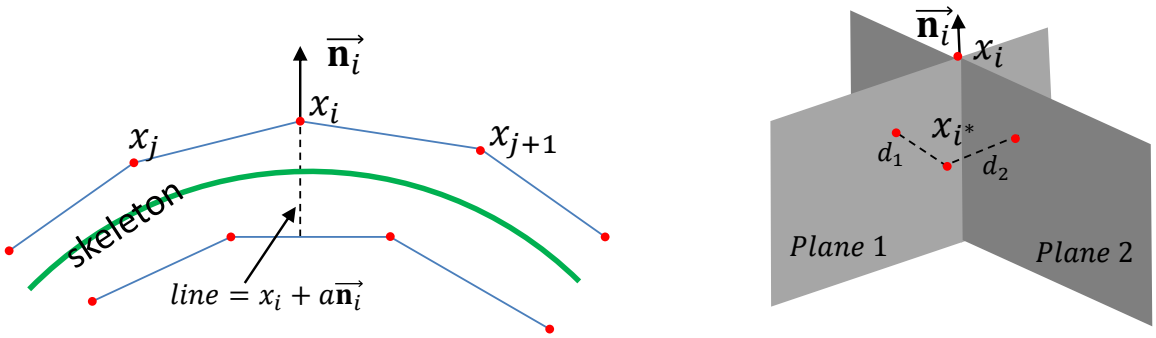


Figure **6-4**: Left: The vertex x_i moves along the line constraint. Right: the distance of vertex x_i to plane 1 and plane 2 when the position in every iteration changes.

This line can be seen as an intersection of two planes (see figure **6-4**) created with the information of the vertex x_i, the normal $\overrightarrow{\mathbf{n}_i}$ and neighbors x_j of x_i.

The plane equation.

$$a\mathbf{x} + b_0\mathbf{y} + c_0\mathbf{z} + d_0 = 0$$

There are various methods that build a plane equation using three non-collinear points. These three points were herein chosen using the vertex x_i, the normal $\overrightarrow{\mathbf{n}_i}$ and neighbors x_j of x_i.

$P_1 = x_i$, $P_2 = x_j$ when $x_j \in Neighbors\,(x_i)$ and $P_3 = P_1 - \overrightarrow{\mathbf{n}_i}$.

An equation plane only requires three non-collinear points$\{P_1, P_2, P_3\}$ and we can solve the next system of equations to find the a, b, c variables.

$$\begin{aligned} ax_1 + by_1 + cz_1 + d &= 0 \\ ax_2 + by_2 + cz_2 + d &= 0 \\ ax_3 + by_3 + cz_3 + d &= 0 \end{aligned}$$

The distance of point $P_0 = \{x_0, y_0, z_0\}$ to a plane $\Pi = a\mathbf{x} + b_0\mathbf{y} + c_0\mathbf{z} + d_0$.

$$|\Pi - P_0| = \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}} \tag{6-2}$$

In the equation above, every point $P_0 = \{x_0, y_0, z_0\}$ that belongs to the plane exists when equation (6-2) is null. If the point P_0 is not in the plane, the value of the plane equation (6-2) changes, and the absolute value is increased if P_0 is far from the plane. Let us use this simple relationship to build a valid constraint that can be put in the form of the linear equation $Ax = B$. Where A is the matrix with values $\begin{vmatrix} a & b & c \end{vmatrix}$ the x value corresponds to

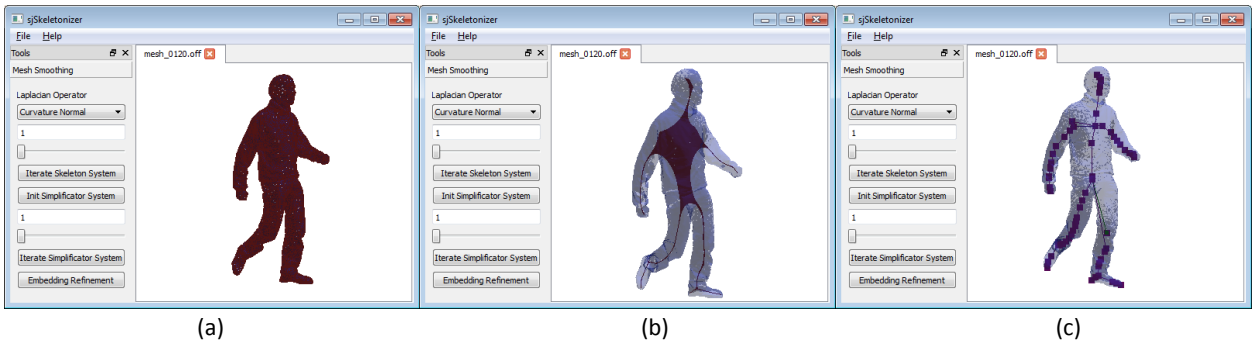


Figure **6-5**: sjSkeletonizer interface. (a) Open and display mesh. (b) Mesh contracted after 4 iterations. (c) Skeleton obtained after surface simplification.

every vertex in mesh $\{x_0, y_0, z_0\}$ and B stands for the d value of the plane equation. Provided that the linear system presented in equation 6-1 has many solutions, the system solves it in the least-squares sense. With the new constraint, the system then tries to minimize the distance between the new vertex positions and the two planes.

The new system of equations proposed

$$\begin{bmatrix} W_L L \\ W_H \\ W_D \Pi_1 \\ W_D \Pi_2 \end{bmatrix} X_{t+1} = \begin{bmatrix} 0 \\ W_H X_t \\ -D_1 \\ -D_2 \end{bmatrix} \tag{6-3}$$

where W_D is the weight for Π_1 and Π_2 constraints, which forces the vertices to move along the normal line of every vertex. Π_1 and Π_2 are matrix that contain a, b, c values of the equation of the plane for every vertex. D_1 and D_2 are the vectors with d values of the equations of the planes for every vertex.

6.3 Results and Conclusions

As a result of this work, the software *sjSkeletonizer* [32] permits the processing, visualization and extraction of the skeleton from polygonal hybrid meshes composed of triangles and quads. In figure **6-5**, the software interface can be observed. The sjSkeletonizer software allows meshes in an interactive viewer to be opened and displayed. This software contracts a mesh with the use of two different Laplacian operators: the cotangent operator and umbrella operator, where the user can customize the number of iterations. The software allows a user to visualize the original mesh and contract mesh after a particular number of iterations (figure **6-5**.b). In the final step of the process, the software simplifies the mesh (figure **6-5**.c)

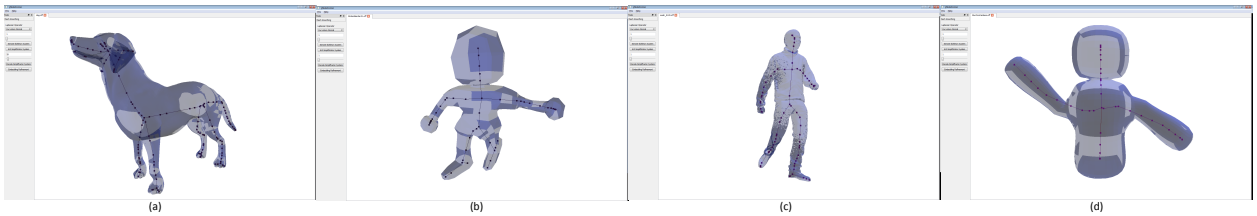


Figure **6-7**: Skeleton extracted from different models. (a) Dog model (b) Character model. (c) Person model. (d) Clay model.

using the same method proposed by Au et al. [3] that consists of a modified version of quadric error metrics by Garland et al. [18] .

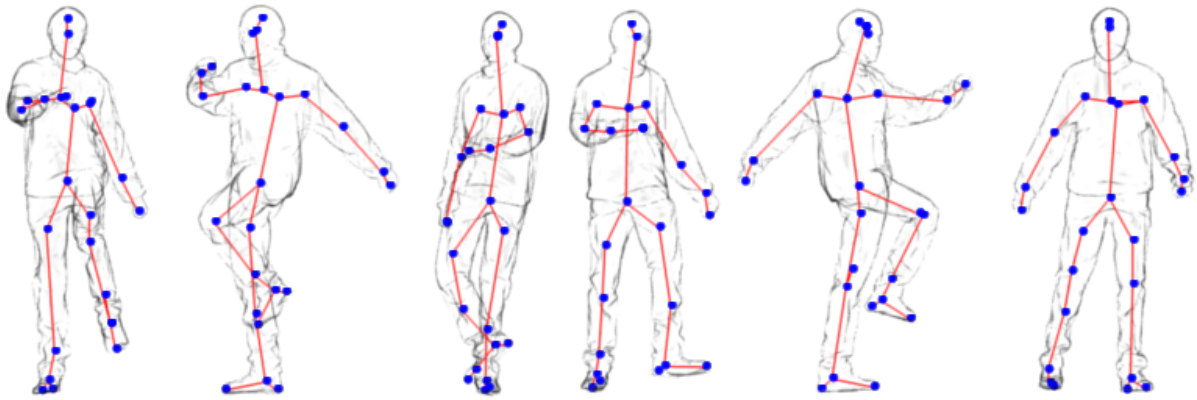


Figure **6-6**: Model with different poses and skeleton obtained with our skeleton extraction software.

Figure **6-7** shows the skeleton extracted from different models using our method. Every skeleton accurately represents the topology and geometry of the original meshes. The extracted skeletons are homotopic to the original shapes. Our method also allows the skeleton to be simplified to the user's requirement in order to guarantee that every node in the skeleton closely represents every core part of the original model.

The method proposed is robust under isometric transformations (figure **6-6**) so that if some global transformations, such as rotations of central parts of the mesh, are made, the extracted skeletons will be similar because the Laplacian operator properties are invariant under rotations. The figure shows similar skeletons for different poses of the same model, these skeletons show identical branching structure.

7 Conclusion

This work presented a novel extension of the Laplace Beltrami operator for hybrid quad/triangle meshes, and the successful application of such principles in different types of problems in computer geometric modelling like smoothing, enhancing, sculpting, deformation, reposing and skeleton extraction.

We have largely demonstrated that our method has good performance, stability and robustness of the extension proposed. This novel extension of the Laplace Beltrami operator was introduced in the computer modeling industry inside the Blender 3D computer graphics software, in order to facilitate artists who work with different tools developed using TQLBO, for any kinds of meshes, i.e., triangular meshes, quadrilateral meshes or hybrid meshes.

Bibliography

[1] Marc Alexa and Max Wardetzky. Discrete laplacians on general polygonal meshes. *ACM Trans. Graph.*, 30(4):102:1–102:10, July 2011.

[2] D. Attali, J.-D. Boissonnat, and H. Edelsbrunner. Stability and computation of the medial axis — a state-of-the-art report. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, 1:–, 2007.

[3] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Transactions on Graphics*, 27(3):10, 2008.

[4] Mikhail Belkin, Jian Sun, and Yusu Wang. Discrete laplace operator on meshed surfaces. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, SCG '08, pages 278–287, New York, NY, USA, 2008. ACM.

[5] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 113–120, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[6] Blender-Foundation. Blender open source 3d application for modeling, animation, rendering, compositing, video editing and game creation. http://www.blender.org/, 2012.

[7] Mario Botsch, Mark Pauly, Christian Rossl, Stephan Bischoff, and Leif Kobbelt. Geometric modeling based on triangle meshes. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[8] Luc Buatois, Guillaume Caumon, and Bruno Lévy. Concurrent number cruncher: an efficient sparse linear solver on the gpu. In *Proceedings of the Third international conference on High Performance Computing and Communications*, HPCC'07, pages 358–371, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, November 1978.

[10] Yong Chen and Charlie C. L. Wang. Uniform offsetting of polygonal model based on layered depth-normal images. *Comput. Aided Des.*, 43(1):31–46, January 2011.

[11] Sabine Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. *SIGGRAPH Comput. Graph.*, 24(4):187–196, September 1990.

[12] Nicu D. Cornea and Patrick Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007.

[13] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 85–94, New York, NY, USA, 1998. ACM.

[14] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Anisotropic feature-preserving denoising of height fields and bivariate data. In *In Graphics Interface*, pages 145–152, 2000.

[15] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[16] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iwires: An analyze-and-edit approach to shape manipulation. *ACM Transactions on Graphics (Siggraph)*, 28(3):#33, 1–10, 2009.

[17] Tinsley A. Galyean and John F. Hughes. Sculpting: an interactive volumetric modeling technique. *SIGGRAPH Comput. Graph.*, 25(4):267–274, July 1991.

[18] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[19] Ozgur Gonen and Ergun Akleman. Smi 2012: Short paper: Sketch based 3d modeling with curvature classification. *Comput. Graph.*, 36(5):521–525, August 2012.

[20] Uwe Hahne. *Weighting in Laplacian Mesh Editing*. PhD thesis, Bauhaus-Universitat Weimar, 2006.

[21] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[22] Bruno Levy. Numerical methods for digital geometry processing. In *2005 Israel-Korea Bi-national Conference on New Technologies and Visualization Methods for Product Development on Design and Reverse Engineering*, Haifa/Israel, Nov 2005.

[23] Liu, L., Bajaj, C., Deasy, J. O., Low, D. A., Ju, and T. Surface reconstruction from non-parallel curve networks. *Computer Graphics Forum*, 27(2):155–163, April 2008.

[24] C. Loop. Smooth subdivision surfaces based on triangles. Department of mathematics, University of Utah, Utah, USA, August 1987.

[25] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.

[26] Tony Mullen. *Introducing character animation with Blender*. Indianapolis, Ind. Wiley Pub. cop., 2007.

[27] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, GRAPHITE '06, pages 381–389, New York, NY, USA, 2006. ACM.

[28] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph.*, 23(3):609–612, August 2004.

[29] Ulrich Pinkall, Strasse Des Juni, and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[30] Alexander Pinzon, Fabio Martinez, and Eduardo Romero. Análisis experimental de la extracción del esqueleto por contracción con suavizado laplaciano, December 2010. Poster presented at 6th International Seminar on Medical Image Processing and Analysis (SIPAIM 2010), December 1–4, Universidad Nacional de Colombia, Bogota, Colombia.

[31] Alexander Pinzon and Eduardo Romero. Software para la extracción del esqueleto por contracción y suavizado, December 2011. Poster presented at 7th International Seminar on Medical Image Processing and Analysis (SIPAIM 2011), December 6–8, Universidad Industrial de Santander, Bucaramanga, Colombia.

[32] Alexander Pinzon and Eduardo Romero. sjskeletonizer: Skeleton extraction software., March 2012. http://code.google.com/p/jeronimo/.

[33] Alexander Pinzon and Eduardo Romero. Shape inflation with an adapted laplacian operator for hybrid quad/triangle meshes. *2013 26th SIBGRAPI Conference on Graphics, Patterns and Images*, 0:179–186, 2013.

[34] Pawas Ranjan. *Discrete Laplace Operator: Theory and Applications*. PhD thesis, The Ohio State University, Columbus, OH, USA, 2012.

[35] Steven Rosenberg. *The Laplacian on a Riemannian manifold: an introduction to analysis on manifolds*. Number 31. Cambridge University Press, 1997.

[36] Robert Schneider, Leif Kobbelt, and Hans-Peter Seidel. Improved bi-laplacian mesh fairing. In Tom Lyche and Larry L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces: Oslo 2000*, Innovations in Applied Mathematics, pages 445–454, Oslo, Norway, 2001. Vanderbilt University.

[37] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, pages 175–184, New York, NY, USA, 2004. ACM.

[38] Olga Sorkine. Differential representations for mesh processing. *Comput. Graph. Forum*, 1:789–807, 2006.

[39] Michael Spivak. *A Comprehensive Introduction to Differential Geometry*, volume 1. Publish or Perish, Inc, 3rd edition, January 1999.

[40] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 395–404, New York, NY, USA, 1998. ACM.

[41] Lucian Stanculescu, Raphalle Chaine, and Marie-Paule Cani. Freestyle: Sculpting meshes with self-adaptive topology. *Computers & Graphics*, 35(3):614 – 622, 2011. Shape Modeling International (SMI) Conference 2011.

[42] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 351–358, New York, NY, USA, 1995. ACM.

[43] Yunhui Xiong, Guiqing Li, and Guoqiang Han. Mean laplace-beltrami operator for quadrilateral meshes. In Zhigeng Pan, Adrian Cheok, Wolfgang Muller, and Xubo Yang, editors, *Transactions on Edutainment V*, volume 6530 of *Lecture Notes in Computer Science*, pages 189–201. Springer Berlin / Heidelberg, 2011.

[44] Guoliang Xu. Discrete laplace-beltrami operators and their convergence. *Comput. Aided Geom. Des.*, 21(8):767–784, 2004.

[45] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph.*, 24(3):496–503, July 2005.

[46] Wei Zhuo and Jarek Rossignac. Curvature-based offset distance: Implementations and applications. *Computers & Graphics*, 36(5):445 – 454, 2012.