# Triangular Mesh Parameterization with Trimmed Surfaces

Oscar E. Ruiz, Daniel Mejia, Carlos A. Cadavid

Laboratorio de CAD/CAM/CAE
Universidad EAFIT, Medellin, Colombia
oruiz@eafit.edu.co, dmejiap@eafit.edu.co, ccadavid@eafit.edu.co

**Abstract**

Given a 2-manifold triangular mesh $M \subset \mathbb{R}^3$, with border, a parameterization of $M$ is a FACE or trimmed surface $F = \{S, L_0, ..., L_m\}$. $F$ is a connected subset or region of a parametric surface $S$, bounded by a set of LOOPs $L_0, ..., L_m$ such that each $L_i \subset S$ is a closed 1-manifold having no intersection with the other $L_j$ LOOPs. The parametric surface $S$ is a statistical fit of the mesh $M$. $L_0$ is the outermost LOOP bounding $F$ and $L_i$ is the LOOP of the i-th hole in $F$ (if any). The problem of parameterizing triangular meshes is relevant for reverse engineering, tool path planning, feature detection, re-design, etc. State-of-art mesh procedures parameterize a rectangular mesh $M$. To improve such procedures, we report here the implementation of an algorithm which parameterizes meshes $M$ presenting holes and concavities. We synthesize a parametric surface $S \subset \mathbb{R}^3$ which approximates a superset of the mesh $M$. Then, we compute a set of LOOPs trimming $S$, and therefore completing the FACE $F = \{S, L_0, ..., L_m\}$. Our algorithm gives satisfactory results for $M$ having low Gaussian curvature (i.e. $M$ being quasi-developable or developable). This assumption is a reasonable one, since $M$ is the product of manifold segmentation pre-processing. Our algorithm computes: (1) a manifold learning mapping $\phi : M \to U \subset \mathbb{R}^2$, (2) an inverse mapping $S : W \subset \mathbb{R}^2 \to \mathbb{R}^3$, with $W$ being a rectangular grid containing and surpassing $U$. To compute $\phi$ we test IsoMap, Laplacian Eigenmaps and Hessian Local Linear Embedding (best results with HLLE). For the back mapping (NURBS) $S$ the crucial step is to find a control polyhedron $P$, which is an extrapolation of $M$. We calculate $P$ by extrapolating Radial Basis Functions that interpolate points inside $\phi(M)$. We successfully test our implementation with several datasets presenting concavities, holes, and are extremely non-developable.

Ongoing work is being devoted to manifold segmentation which facilitates mesh parameterization.

**Keywords:** triangular mesh parameterization, trimmed surface, manifold learning, NURBS, RBFs.

# Glossary

**LOOP:** Closed (Piecewise Linear or Smooth) curve lying on a surface, and bounding a connected region on the surface. In this manuscript, LOOPs are denoted with $\Gamma$ or $\gamma$.

**B-REP:** Boundary Representation.

**HLLE:** Hessian Locally Linear Embedding.

**NURBS:** Non-Uniform Rational B-Spline.

**RBF:** Radial Basis Function.

$M$**:** Triangular mesh (with boundary), composed by the set of triangles $T = \{t_1, t_2, \cdots, t_q\}$ with vertex set $X = \{x_1, x_2, \cdots, x_n\}$ ($X \subset \mathbb{R}^3$).

$\partial M$**:** Boundary of $M$, whose connected components are LOOPs ($\partial M = \{\Gamma_0, \Gamma_1, ..., \Gamma_k\}$).

$\phi$**:** An homeomorphic map $\phi : M \to \mathbb{R}^2$, implemented here for dimensional reduction or manifold learning.

$U$**:** $U = \{u_1, u_2, \cdots, u_n\}$ is the parametric image of vertices of $M$ ($U = \phi(X)$, $U \subset \mathbb{R}^2$).

$\partial(\phi(M))$**:** Boundary of the parametric image of $M$. For the sake of simplicity, we assume that $\partial(\phi(M)) = \phi(\partial(M)))$.

$\gamma_i$**:** i-th LOOPs of $\partial(\phi(M))$.

$\lambda_i$**:** Re-sampling of a LOOP $\gamma_i$.

$W$**:** Rectangular grid in $R^2$ such that $U$ lies in the convex hull of $W$.

$H(W)$**:** Rectangular point set in $R^2$ being the convex hull of $W$.

$P$**:** Rectangular grid in $\mathbb{R}^3$ being the control polyhedron for the parametric surface $f$.

$f$**:** Function $f : W \to \mathbb{R}^3$ produces $P$ the control polyhedron of $S$ ($P = f(W)$) by calculating an extrapolation of $M$ in $\mathbb{R}^3$.

$S$**:** $S : \mathbb{R}^2 \to \mathbb{R}^3$ is a parametric surface which approximates and extends $M$ in $\mathbb{R}^3$. To simplify notation, $S$ refers here to both: (a) the parametric mapping (i.e. $S()$) and (b) the set of points product of the mapping $S()$ (i.e. $S(H(W)) = \{S(w_1, w_2) | (w_1, w_2) \in H(W)\}$).

$L_i$**:** Trimming curve in $M \subset \mathbb{R}^3$ defined as $L_i = S(\lambda_i)$.

$F$**:** Trimmed surface (FACE) such that $F = (S, \{L_0, L_1, \cdots)$.

$\partial F$**:** Boundary of $F$ approximated by the union of all $L_i$.

# 1 Introduction

In this manuscript, $M$ denotes a triangle-based mesh, which is a 2-manifold with border. Without loss of generality, we consider $M$ as the result of segmenting a larger triangular mesh $M$ presents a low curvature (i.e. $M$ is near-developable). Therefore, $M$ can also be referred to as *sub-mesh*.

Being a 2-manifold, $M$ admits a 2-variable parameterization, which is a homeomorphism between $M$ and a connected subset of $R^2$. However, to be usable in Reverse Engineering, CAD, CAM, or visualization, the parameterization must be accompanied by a *trimmed FACE*, which approximates the triangular mesh $M$. A *trimmed FACE* consists of a parametric surface $S : \mathbb{R}^2 \to \mathbb{R}^3$ and a set of LOOPs or closed curves $\Gamma_i \subset M$, which bound a connected sub-region of $M$.

Trimmed Surfaces are indispensable in Boundary Representations and therefore in Computer Aided Geometric Design. In a Trimmed Surface, $S()$ is usually based on a rectangular control polyhedron $P$. Triangular topology for $P$ is very unusual because it produces undefinition in tangent and normal vectors, rendering $S()$ and FACE $F$ unusable.

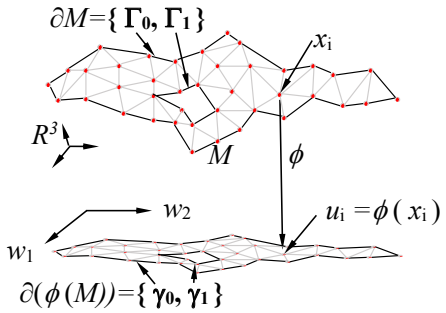
$M$ is a connected triangular mesh $M = (X, T)$, with a border described by the set of LOOPs $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_m\}$. The $\Gamma_i$ LOOPs are piecewise linear closed 1-manifolds, which do not intersect each other. Given $M$, a trimmed parametric surface $F = (S, \{L_0, L_1, \cdots, L_m\})$ is pursued, with $S$ being a smooth surface that approximates a (conveniently defined) superset of $M$. The set of curves $\{L_1, L_2, \cdots, L_m\}$ of the trimmed parametric surface $F$ approximates the boundary $\partial M$ on the surface $S$.

In this article we propose a procedure for computing the trimmed surface $F$, as follows: (i) Compute a mapping $\phi : M \subset \mathbb{R}^3 \to U \subset \mathbb{R}^2$ which describes a 2D parameter space for $M$. (ii) Compute a back-mapping (NURBS or RBFs) $S : W \to \mathbb{R}^3$, with $W$ being a rectangular region in $\mathbb{R}^2$, s.t. $W$ is superset of $U$. (iii) Compute via $S()$ a FACE boundary approximating $\partial M$.

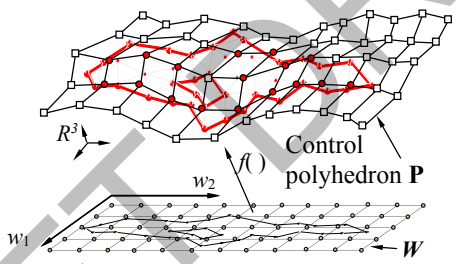
The remainder of the article is organized as follows: Section 2 reviews the relevant literature. Section 3 describes the implemented methodology. Section 4 presents and discusses some results. Section 5 concludes the paper and introduces what remains for future work.

# 2 Literature Review

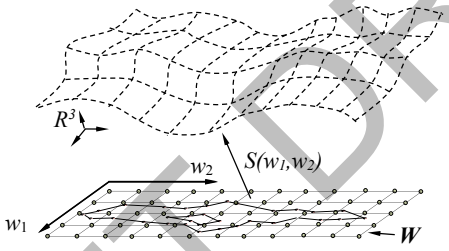
In most engineering applications, the sole parameterization of the triangular mesh $M$ does not suffice, and a Trimmed Parametric Surface $(S, \{L_0, L_1, ...\})$ approximating $M$ is required. The first process builds a function $\phi : \mathbb{R}^3 \to \mathbb{R}^2$. The second process builds a back mapping



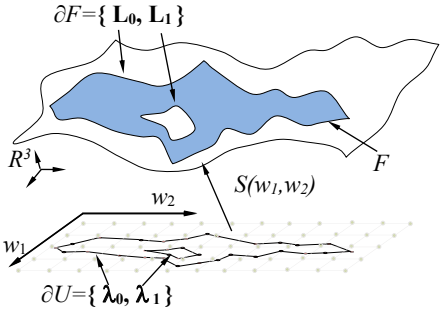

(a) Determination of Homeomorphism $\phi : M \to \mathbb{R}^2$ and mapping of Boundaries of $M$ ($\partial M$).




(b) Synthesis of Control Polyhedron $P = f(W)$.




(c) Calculation of Carrier Surface $S$.




(d) Calculation of Boundaries of FACE $F$, $\partial F$.

Figure 1: Parameterization of Mesh $M$ with a Trimmed Surface (FACE $F$).

$S : \mathbb{R}^2 \rightarrow \mathbb{R}^3$.

When parameterizing triangular meshes, the parameterization of the individual triangles may be sufficient whenever no parametric space common to *all* triangles in the mesh is required. For example, smooth parametric surfaces fit to separate triangular control polyhedra give as result $C^0$-, $C^1$-, or $C^2$-smooth triangular patches ([1, 2, 3, 4, 5, 6, 7]), which do not share a common parametric space. In contrast, ref. [8] presents a geodesic-based Piecewise Linear parameterization common to the complete triangular mesh $M$. Although very intuitive, this approach presents intersection of PL geodesics in high Gauss - curvature surfaces.

## 2.1 Dimensional Reduction $\phi : \mathbb{R}^3 \supset M \longrightarrow \mathbb{R}^2$

This section discusses the existing algorithms for dimensional reduction $\phi : M \rightarrow \mathbb{R}^2$ (Fig. 1(a)), which finds a 2D underlying parameter space in the 2-manifold triangular mesh $M \subset \mathbb{R}^3$. This parameter space $U$ is the image of the mesh points $X$ under the map $\phi$ (i.e. $U = \phi(X)$). The algorithms for dimensional reduction may be classified according to the properties that they preserve, as they synthesize $(\phi, U)$:

1. **Angle-preserving algorithms:** Conformal maps seek to preserve the angles formed by intersecting curves. The function $\phi$ is devised to minimize angle deformations [9, 10, 11, 12, 13]. An angle-preserving $\phi$ would not, in general, preserve areas or lengths, causing a strong warping in the back-mapping $S$.

2. **Area-preserving algorithms:** An authalic map $\phi :\rightarrow \mathbb{R}^2$ would satisfy $A(t_i) = A(\phi(t_i))$ for triangle $t_i \in T$. Area distortions are minimized over fixed 2-Dimensional primitives (e.g. disk or rectangle [14, 15]). Authalic maps may result in large angle distortions which in turn would produce important distortions in the back-mapping $S$.

3. **Distance-preserving algorithms:** An isometric map $\phi :\rightarrow \mathbb{R}^2$ would preserve distance among points. This means that $d_M(p, q) = d(\phi(p), \phi(q))$, for $p, q \in M$, with $d_M()$ being a distance measured on $M$ (refs. [16, 17, 18, 19, 20]). Isometric maps do not present distortions in the back-mapping $S : \mathbb{R}^2 \rightarrow \mathbb{R}^3$. However, unlike angle- and area-preserving ones, they require $M$ to be highly *developable* (i.e. with low Gaussian curvature).

   Surfaces which are strongly *non-developable* may be partitioned into smaller, more *developable* ones. The issue of Manifold Segmentation, however, is outside the scope of the present manuscript, and is the subject of our future efforts.

## 2.2 Parametric Surface $S : \mathbb{R}^2 \rightarrow \mathbb{R}^3$

Since this article aims to parameterize a mesh $M$ with a Trimmed Surface or FACE (in CAD sense), the dimensional reduction $\phi : M \rightarrow \mathbb{R}^2$ must be followed by the computation of a parametric surface $S : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ and a connected subset within it, which resembles $M$ (with concavities, holes, etc.). This goal is illustrated in Fig. 1(d).

Ref. [21] describes a method in which an isometric rectangular surface is computed, but requires the input mesh to be already of this rectangular nature (i.e. a triangular mesh of a rectangular patch) which strongly constraints the algorithm.

Ref. [22] presents an initial parameterization of $\phi : M \rightarrow U$. The Coons back mapping $S$ has domain in a subset of the parametric space $U$, which cannot present holes or concavities. As a result, $S$ can only fit a rectangular subset of $M$, leaving out the possibility of $M$ having holes or concavities.

## 2.3 Conclusions of Literature Review

Our Literature Review has found several approaches for mesh parameterization. However, they do not address meshes with concavities and holes. To enable them, it is necessary to complement the sole mesh parameterization with the synthesis of a Trimmed Surface, which smooths $M$ and expresses the holes and concavities, besides producing a parameterization. Consequently, we present here our approach, in which a triangular mesh (with boundary and holes) $M \subset \mathbb{R}^3$ is approximated by a trimmed surface $F = (S, \{L_1, L_2, \cdots, L_m\})$. First, a parameterization $U$ of $M$ is computed using dimensional reduction (Fig. 1(a)). Then, a rectangular superset of $U$, $H(W)$, is mapped back to $\mathbb{R}^3$, via a parametric surface $S(Fig.1(b)) : H(W) \longrightarrow \mathbb{R}^3$ which fits a superset of $M$ (Fig. 1(c)). FACE boundaries are drawn on $S$, to trim a FACE, with holes, which resembles $M$ (Fig. 1(d)). The calculation of a superset of $M$ in $\mathbb{R}^3$ is, of course, a very sensitive operation, for which we apply a combination of NURBs and Radial Basis Functions.

# 3 Methodology

Consider $M = (X, T)$, a connected 2-manifold in $\mathbb{R}^3$ with holes and border. The set of all LOOPs bounding $M$ is noted as $\partial M = \{\Gamma_0, \Gamma_1, \cdots, \Gamma_m\}$. Our goal is to find a FACE $F$ or Trimmed Surface $F = (S, \{L_0, L_1, \cdots, L_m\})$, composed by a parametric surface $S : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ and a set of boundaries on $S$ ( $\partial F = \{L_0, L_1, \cdots, L_m\}$ such that $\partial M \approx \partial F$.

In order to achieve this goal, we follow the procedure in Fig. 1, which appears as Data Flow in Fig. 2.

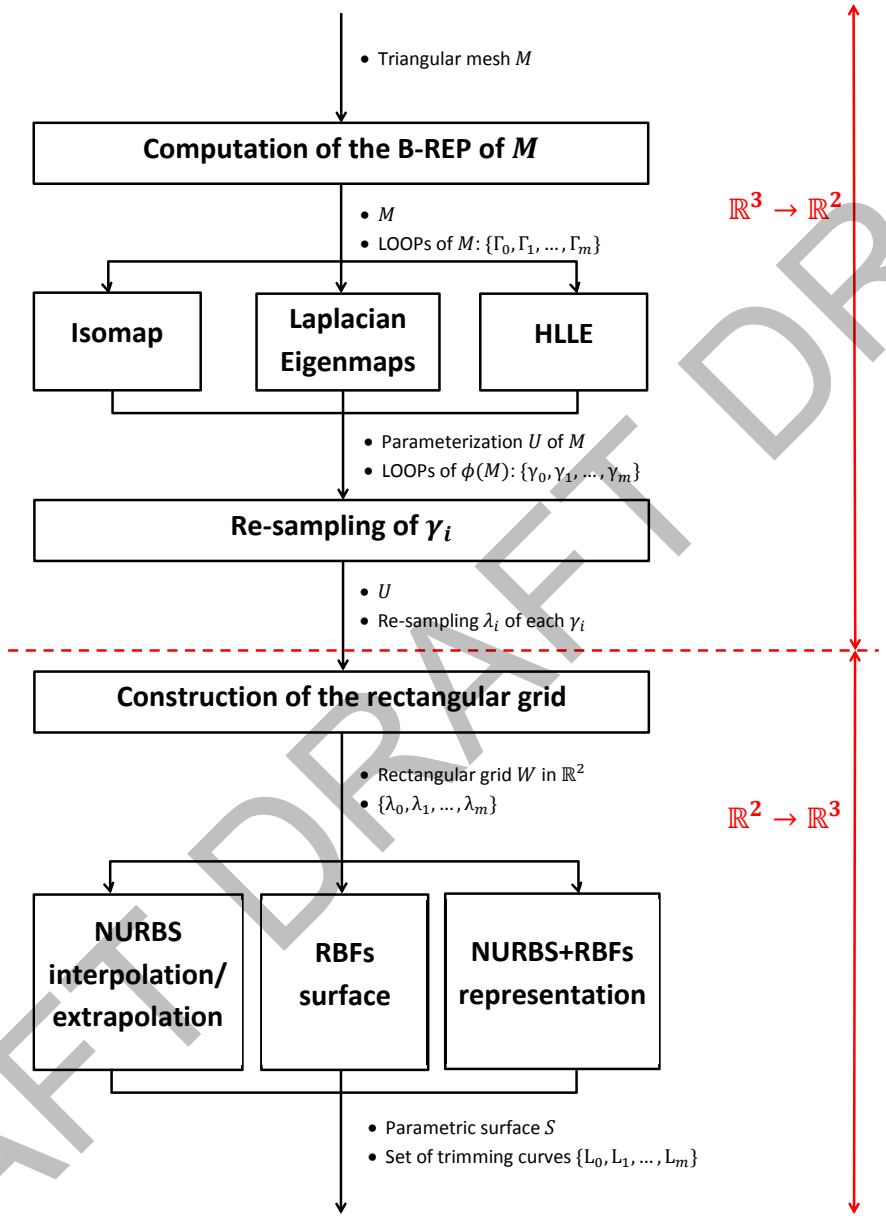



Figure 2: Construction of a trimmed parameterization of the triangular mesh $M$. $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ mapping (up) and $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ back-mapping (down).

1. **Computation of the B-REP of $M$:** In this pre-processing, a boundary representation of the triangular mesh $M$ is computed in order to extract each of LOOPs $\Gamma_i$ that bound $M$.

2. **Computation of the mapping $\phi : M \rightarrow \mathbb{R}^2$** (Fig. 1(a)): A manifold learning algorithm is applied in this step in order to extract the parameterization $U$ that characterizes the manifold and each LOOP $\gamma_i$ in the parameter space. The procedure was tested using either: i) Isomap, ii) Laplacian Eigenmaps or iii) HLLE.

3. **Construction of the rectangular grid $W \subset \mathbb{R}^2$**: (Fig. 1(b)) A rectangular grid $W$ is built in $\mathbb{R}^2$ such that $U$ lies inside the convex hull of $W$ ($U \subset H(W)$).

4. **Synthesis of a Control Polyhedron $P$ for $S$** (Fig. 1(b)): A control polyhedron $P = f(W)$ is required for the parametric surface $S$ of the Trimmed Face $F$. In Fig. 1(b), $P$ is represented as a grid of round-icon and square-icon vertices. Round-icon vertices $f(w_1, w_2)$ fall inside $M$ since $(w_1, w_2) \in U$. Square-icon vertices fall outside $M$ since $(w_1, w_2) \notin U$, and therefore $f(w_1, w_2)$ must be estimated. A Radial Basis Function $f : W \longrightarrow \mathbb{R}^3$ is created by using the condition $f(U) = X$, since the pairs $(x_i, u_i) = (x_i, \phi(x_i))$ are known. $f(H(W))$is then used to extrapolate $M$ as needed.

5. **Representation of $M$ by a trimmed surface $F = (S, \{L_0, ...L_m\})$**: Once the control Polyhedron $P$ for $S$ is estimated, the actual calculation of the parametric surface $S$ proceeds using a NURBs formulation. The boundary of $F$, $\partial F = \{L_0, ...L_m\}$, is achieved as $L_i = f(\lambda_i)$, where $\lambda_i$ is the re-sampling of the straight-edge LOOP $\gamma_i$.

The algorithms implemented for the above procedure are briefly discussed below.

## 3.1 Manifold Learning

In the aim of finding $S$, the parameterization $(\phi, U)$ that underlies the original surface must be found. Such parameterization should also be an homeomorphism to $\mathbb{R}^2$ or equivalently: i) the points keep the same connectivity through the mapping (continous map) and ii) triangles do not overlap in the parameter space (bijective map). For finding $U$ several manifold learning techniques have been developed in the literature. However, we focus on three specific ones that were implemented for the procedure described in fig. 2: i) Isomap, ii) Laplacian Eigenmaps and iii) HLLE.

### 3.1.1 Isomap-based parameterization

Isomap was first proposed by Tenembaum et al. [20]. The idea behind it is to assume that $U$ comes from an isometric map. With this in mind, a geodesic estimator $G$ is constructed recursively on $M$:

$$G_{ij} = \begin{cases} \| x_i - x_j \| & \text{if } x_j \in N_i \\ \min \{G_{ik} + G_{kj} | x_k \in N_i\} & \text{otherwise} \end{cases} \tag{1}$$

where $\| \cdot \|$ corresponds to the euclidean norm and $N_i$ is the neighborhood of $x_i$ as indicated by $T$.

The authors then propose to apply Classical Multidimensional Scaling (CMDS) on $G$ for computing $U$ i.e. solve the following equation:

$$G = U^T U \tag{2}$$

Such equation can be easily solved solved by a singular decomposition of $G$.

### 3.1.2 Laplacian Eigenmaps-based parameterization

There exists a problem with Isomap: building $G$ can be time consuming. Laplacian Eigenmaps [12] is a much faster algorithm however, only conformality is guaranteed for the mapping. Laplacian Eigenmaps poses the following optimization problem:

$$\min \sum_{i,j} a_{ij} \| u_i - u_j \|^2 \tag{3}$$

where each $a_{ij}$ is the adjacency weight (locality measure) between the points $x_i$ and $x_j$ in the original mesh.

If $D$ is a diagonal matrix defined as $d_{ii} = \sum_j a_{ij}$ and $L$ is a symmetric matrix defined as $L = D - A$, then eq. (3) can be solved (under adequate constraints) by computing the following eigenproblem:

$$LU = \Lambda DU \tag{4}$$

where $\Lambda$ is a diagonal matrix containing the eigenvalues of $L$. The matrix $L$ is known as the Laplacian of the mesh graph because it is closely related to the Laplace-Beltrami operator on manifolds.

### 3.1.3 HLLE-based Parameterization

Another alternative would be to construct a Hessian estimator as described in [23]. Since the Hessian can be seen as a measure of curvature, minimizing it would be like *flattening* the manifold. However, such minimization cannot guarantee any isometry.

By local alignment of each neighborhood, a pair of orthogonal linear functions in $\mathbb{R}^2$ must be found. Since these functions live in the plane, their *curvature* in the direction normal to that plane must be zero. The Hessian functional $\mathcal{H}$ is therefore constructed in order to measure such *curvature*:

$$\mathcal{H} = U^T K U \tag{5}$$

The objective of the algorithm is to minimize this functional. Under adequate constraints eq. (5) becomes an eigenvalue problem. The matrix $K$ is known as the HLLE kernel and is computed by locally estimating a tangent Hessian at each $x_i$.

## 3.2 Surface Representation

After the parameterization $U$ has been found, we propose to compute a rectangular grid $W$. The convex hull of this grid has to contain every point of $U$ as illustrated in fig. 3(b) in order to represent the whole surface.

To approximate the surface such grid must be mapped back to $\mathbb{R}^3$. Inner points (i.e. points of the grid that overlap with the 2-Dimensional triangulation) can be easily interpolated using barycentric coordinates. However, outer points must be extrapolated. We call this back-mapping $f : \mathbb{R}^2 \to \mathbb{R}^3$ and the image of $W$ under $f$ is the control polyhedron $P$.

We propose three different alternatives for computing the back-mapping $f$ and the parameterization $S$ of the whole surface:

### 3.2.1 NURBS Interpolation/Extrapolation

A tensor product surface is a parametric surface defined as:

$$S(w) = \sum_{ij} g_i(v_x) h_j(w_2) p_{ij} \tag{6}$$

where $w_1$ and $w_2$ are the coordinates of $w$ in the parameter space, the basis functions $g_i$ and $h_j$ are known as weights and $p_{ij}$ are known as control points. In this case $P$ works as the control surface containing such control points and small patches of $P$ are used to fit surfaces of the form defined in eq. (6). Usually these patches consist of $3 \times 3$ or $4 \times 4$ subgrids. Common basis functions used for interpolation are the well-known NURBS such as Bézier

and B-Splines.

Here we propose to build $P$ by iteratively extrapolating a smaller tensor product surface as described below:

1. Locate a subgrid $W^{(loc)}$ inside $W$ such that every $w_{ij}^{(loc)}$ lies inside the triangulation in the parameter space.
2. Compute $P^{(loc)}$ using the barycentric coordinates of $W^{(loc)}$.
3. Extrapolate the grid $P^{(loc)}$ by extending the domain of a local surface described by eq. (6) in every direction ($w_1$ and $w_2$). This extension must preserve the rectangular structure (i.e. must be a bigger grid).
4. If any of the extrapolated values of $W^{(loc)}$ lies inside the original triangulation, replace its mapped value as described in step 2.
5. Update both $W^{(loc)}$ and $P^{(loc)}$ given the extended grid.
6. Repeat steps 3-5 until $W^{(loc)} = W$.
7. Let $P = P^{(loc)}$.

After finding the control surface $P$, compute the parametric surface $S$ using the scheme defined in eq. (6).

### 3.2.2 Radial Basis Function Surface

An RBFs surface is a surface of the form:

$$f(w) = \sum_{i=1}^{n} \alpha_i \Phi \left( \parallel w - u_i \parallel \right) + \mathcal{P}(w) \tag{7}$$

where $u_i$ are allocation points which must satisfy $f(u_i) = x_i$, $\Phi(r_i(w))$ are RBFs, $\alpha_i$ are weights that can be estimated by least squares and $\mathcal{P}(w)$ is a stabilizing polynomial. Notice how this approach allows to build a global scheme from an unsorted (non-degenerated) set of points which will define the complete surface even out of the bounds defined by the boundary $\partial(\phi(M))$ in the parameter space.

We propose to approximate $S$ directly using the scheme defined in eq. (7). In this case the grid $W$ must be more refined since the representing surface $S$ is the direct result of such grid i.e. $S(w) = f(w)$.

### 3.2.3 NURBS+RBFs Representation

Here we propose to build the parametric surface by using the approximation qualities of RBFs while keeping the structure of NURBS. This two-step procedure should be followed:

1. Approximate the control polyhedron $P$ using the RBF scheme in eq. (7).
2. Compute $S$ by fitting a NURBS (eq. (6)) over the control polyhedron $P$.

The proposed representation presents several advantages over the other two approaches:

1. RBFs present better results when extrapolating the surface since NURBS takes only into account one of the parametric directions ($w_1$ or $w_2$) at the time of the extrapolation.
2. NURBS extrapolation is a local iterative method which makes it sensitive to initial guesses while the RBFs extrapolation is a global one.
3. NURBS surface requires storing only the control points while the RBFs surface require storing each point of the original mesh and their corresponding weight.
4. Computing a point in a NURBS is usually faster than in the RBFs since distances must be computed in the latter one.
5. NURBS are a standard in CAD CAM CAE tools for representing surfaces.

## 3.3 Surface Trimming

In order to trim the surface $S$, a set of curves $\{L_0, L_1, \cdots, L_m\}$ must be put on the parametric surface such that the boundaries of the original mesh are represented by such curves. The procedure for computing each $L_i$ is described below:

1. A B-REP of $M$ allows us to find each LOOP $\Gamma_i \subset \partial M$.
2. Compute each LOOP $\gamma_i$ bounding the parameterization $\phi(M)$. Since the connectivity of the mesh must be preserved under the homeomorphism $\phi$, $\gamma_i$ can be calculated by making $\gamma_i = \phi(\Gamma_i)$
3. A re-sampling $\lambda_i$ of $\gamma_i$ is carried in $\mathbb{R}^2$ as illustrated in fig. 3(b).
4. Map back each $\lambda_i$. The back-mapping is computed by evaluating the points of each LOOP under the map $S$ which was found in section 3.2. Each resulting curve $L_i$ is a trim curve of $F$.

# 4 Results

We now discuss some results of the described procedure for several datasets. For all the cases, NURBS interpolation and extrapolation was computed over bicubic $(4 \times 4)$ Bézier patches. Additionally, thin plate splines were chosen as basis functions for computing the RBFs:

$$\Phi(r) = r^2 \ln(r) \tag{8}$$

Such basis functions have good properties for surface representation given the fact that they minimize the *bending energy* functional $E$ in $\mathbb{R}^2$ [24]:

$$E(f) = \iint_{\mathbb{R}^2} \left(\frac{\partial^2 f}{\partial w_1^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial w_1 \partial w_2}\right) + \left(\frac{\partial^2 f}{\partial w_2^2}\right)^2 dA \tag{9}$$

Eq. (9) is closely related to the Hessian of $f$ which basically results in a minimization of curvature on the reconstructed surface (and in consequence a smooth surface). Additionally, unlike other RBFs, thin plate splines are parameter-free splines making the algorithm less parameterizable which is highly desirable. A linear polynomial $\mathcal{P}$ is used for stabilization of the algorithm (eq. (7)) since it lies in the kernel of $E$ (i.e. $E(\mathcal{P}) = 0$).

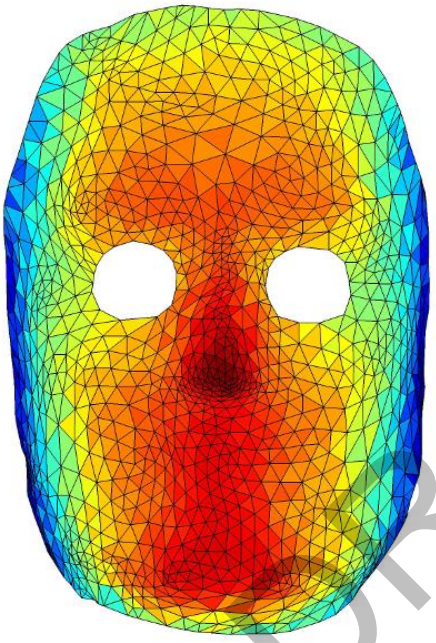
We also implemented the Floyd's shortest-path algorithm in order to compute the geodesic distances between points as described in eq. (3) for the Isomap algorithm.

## 4.1 *Face* Dataset Results

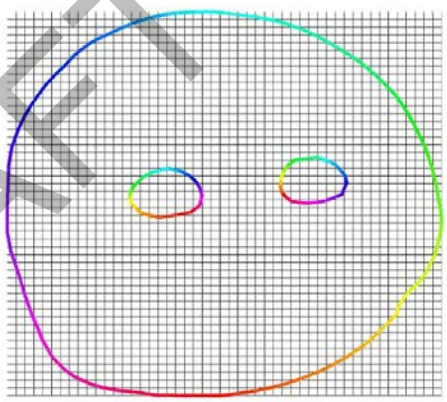
Here we present results of the proposed procedure on the *Face* model (fig. 3(a)). The parameterization $\phi$ is first computed as described in section 3.1. Results of the parameterization can be seen in fig. 4. In order to show the impact of the implemented manifold learning algorithm in the final representation, the NURBS+RBFs approach was used for the three cases (figs. 4(d), 4(e) and 4(f)).

Fig. 4(a) shows the parameterization $U$ estimated with the Isomap algorithm. In this case, such parameterization is highly isometric and resembles the original mesh. This isometry is highly desirable since NURBS and RBFs will not see high distortions at the time of interpolation and extrapolation. The resulting trimmed surface $F = \{S, L_0, L_1, L_2\}$ can be seen in fig 4(d) where a smooth representation of the original surface is achieved with no visual distortion.

Fig. 4(b) shows the parameterization estimated by the Laplacian Eigenmaps algorithm. The resulting parameterization presents high geometric distortions degenerating near the boundary. Interpolation and extrapolation are highly affected by such degeneration arising

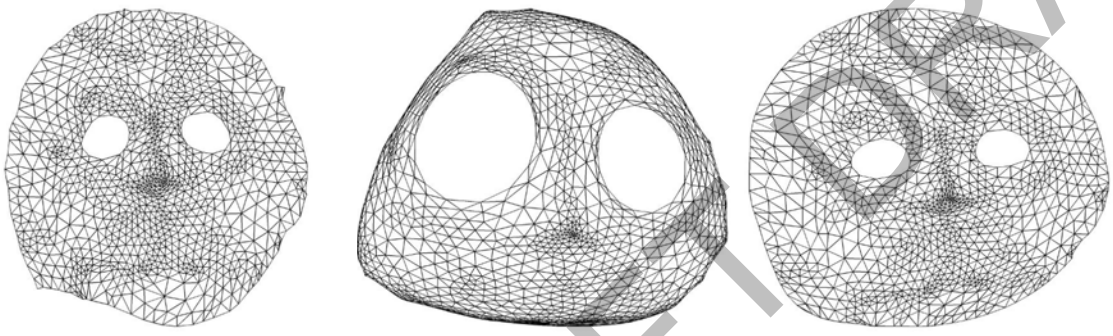


(a) *Mask* dataset $M \subset \mathbb{R}^3$.

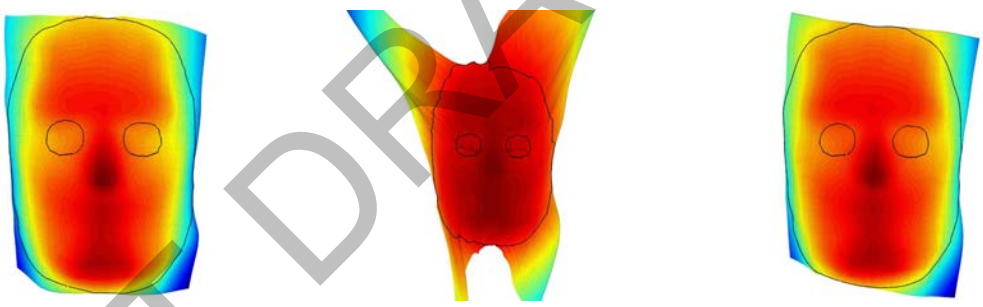


(b) Border of $\phi(M) \subset R^2$ with $\phi=$ HLLE. Border $\partial(\phi(M)) = \{\gamma_0, \gamma_1, \gamma_2\}$ (3 LOOPs). Rectangular Parameter Grid $W$ covering $\phi(M)$.

Figure 3: *Mask* dataset $M$ and Parametric Grid $W$ associated with Hessian Local Linear Embedding ($\phi$).

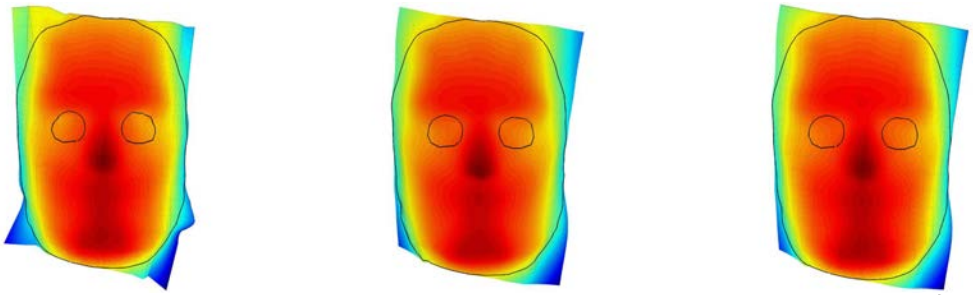


(a) Isomap parameterization. (b) Laplacian Eigenmaps parameterization. (c) HLLE parameterization.



(d) Isomap trimmed surface. (e) Laplacian Eigenmaps trimmed surface. (f) HLLE trimmed surface.

Figure 4: Parameterization $\phi$ of the *Mask* dataset for the implemented manifold learning techniques (up) and their respective trimmed surface $F$ (down).



(a) NURBS interpolation/extrapolation. (b) RBFs surface. (c) NURBS+RBFs representation.

Figure 5: Trimmed Surfaces $F = \{S, L_0, L_1, L_2\}$ of the *Mask* dataset for the three back-mapping approaches: i) NURBS, ii) RBFs and iii) NURBS+RBFs.

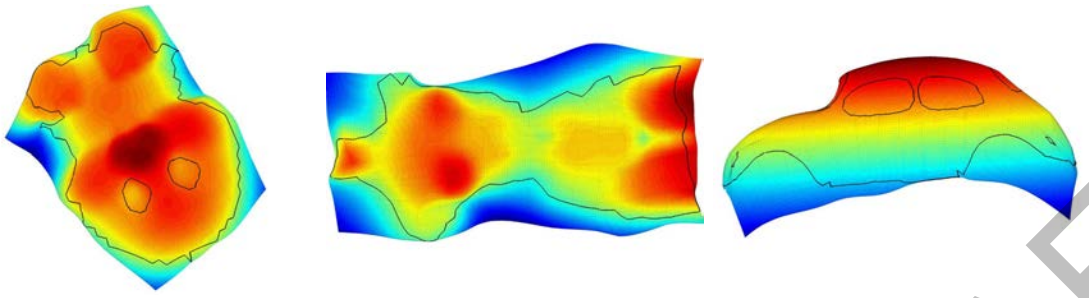
singularities in those areas as can be seen in fig. 4(e) where the trimming curves present irregular behaviour and the resulting representation presents several undesired folds as well as less smoothness than the Isomap trimmed surface (fig. 4(d)).

Fig. 4(c) shows the parameterization estimated by the HLLE algorithm. Less geometric distortions than the Laplacian Eigenmaps are introduced in this parameterization. We believe that the quality of the mesh (i.e. homogeneous size and shape of the triangles) is important for having a low distorted parameterization when parameterizing with HLLE. Such argument is based in the fact that the local alignment made by the HLLE algorithm does not consider relative distances between points. A smooth representation (comparable to the Isomap representation) is achieved as can be seen in fig. 4(f).

In fig. 5 the results of the application of the three surface representation methodologies (NURBS, RBFS and NURBS+RBFs) on the *Mask* dataset are presented. The HLLE algorithm for parameterization was used in all the cases in order to make the results comparable. Fig. 5(a) shows the trimmed surface $F$ resulting by back-mapping the HLLE parameterization $U$ using the NURBS interpolation/extrapolation. The resulting surface accurately resembles the original mesh. However, the extrapolated areas present undesired folds due to the fact that extrapolation is made through only one of the parametric directions (as discussed in section 3.2.3).

Fig. 5(b) shows the trimmed surface obtained by back-mapping $U$ with RBFs. The resulting surface also approximates accurately the surface however, extrapolation is smoother in every direction compared to the NURBS extrapolation and less folds in extrapolated areas arise.

Fig. 5(c) shows the trimmed surface obtained by the NURBS+RBFs back-mapping of $U$. The resulting surface is very similar to the RBFs one. Such similarity is because the fact that the control surface for the NURBS extrapolation is taken from the RBFs approx-



(a) Trimmed surface $F$ of the *Teddy* dataset.

(b) Trimmed surface $F$ of the *Partial-Venus* dataset.

(c) Trimmed surface $F$ of the *Beetle* dataset.

Figure 6: Trimmed surface $F$ of several datasets. Isomap was used for the $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ mapping while NURBS+RBFs were used for the back-mapping.

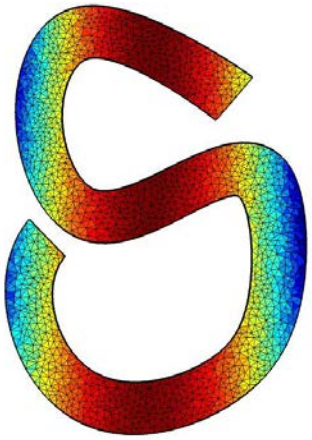
imation. Therefore the final surface obtained by RBFs representation and the obtained by NURBS+RBFs coincide at the same points of the control surface. However, inside the NURBS patches the behaviour is different from the RBFs which can be seen in the resulting surfaces where between patches there is $\mathcal{C}^0$ continuity in the NURBS+RBFs case (fig. 5(c)) contrary to the RBFs surface which presents $\mathcal{C}^2$ (more smoothness) through the whole surface (fig. 5(b)).

## 4.2 Results for Other Datasets

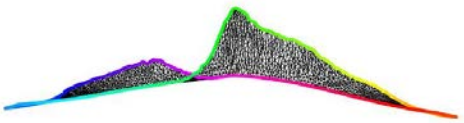
The described procedure was also tested in several datasets in order to illustrate the usefulness of it as well as its flaws. Fig. 6 shows the resulting trimming surface $F$ for highly *developable* datasets such as the *Teddy* dataset (fig. 6(a)), the *Partial-Venus* dataset (fig. 6(b)) and the *Beetle* dataset. Isomap was used for computing the parameterization $U$ while NURBS+RBFs were used for computing the back-mapping in all the three cases. Results show accurate resemblance of such surfaces with the original ones while extrapolated areas present smooth behaviour and small folds.

We observed however that Isomap may present very wrong results in specific cases even if the surface is highly *developable*. This fact is illustrated in the $S$ dataset as shown in fig. 7. Such dataset is highly *developable* but since the underlying parameterization presents a lot of non-convexities then the geodesic estimator fails to recover adequately the geometry in the parameter space. In fig. 7(b) the boundary overlaps and the whole mesh overlaps in $\mathbb{R}^2$ degenerating the resulting trimmed surface as can be observed in fig. 7(d)).

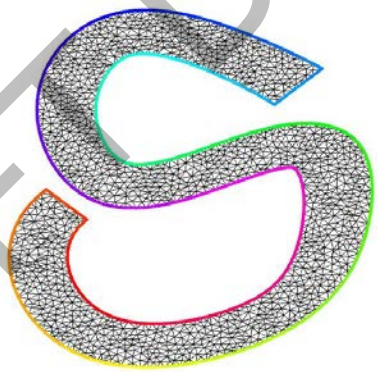
For partially overcoming such problem, HLLE presents a better parameterization of the $S$ dataset which highly resembles the geometry of the mesh on the parameter space as can be seen in fig. 7(c). The resulting trimmed surface presents the $S$ letter inscribed inside a



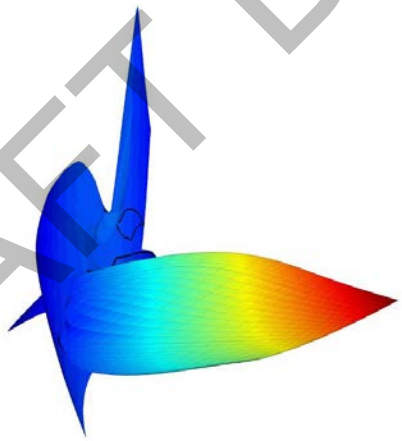
(a) $S$ dataset.



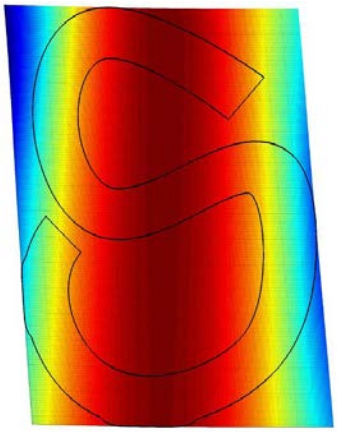
(b) Failed Isomap parameterization $\phi$ of the $S$ dataset.



(c) Successful HLLE parameterization $\phi$ of the $S$ dataset.

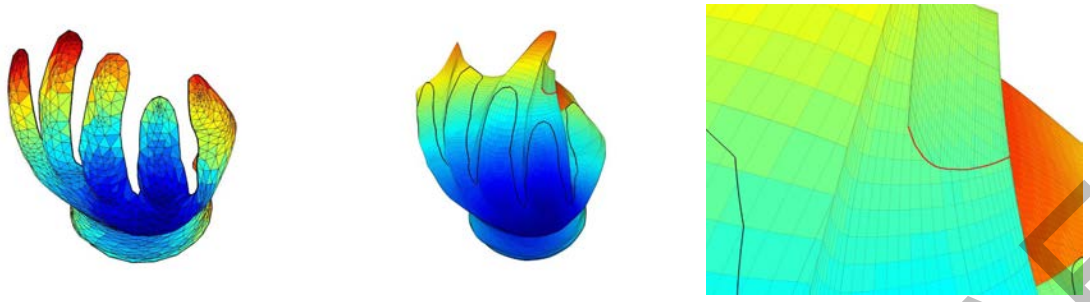


(d) Failed back-mapping $\mathbb{R}^2 \to \mathbb{R}^3$ of the Isomap parameterization.



(e) Successful back-mapping $\mathbb{R}^2 \to \mathbb{R}^3$ of the HLLE parameterization.

Figure 7: Results of the algorithm for the $S$ dataset under the RBF+NURBS approach using Isomap (left) and HLLE (right).



(a) *Partial-Glove* dataset. (b) Trimmed surface $F$ of the *Partial-Glove*. (c) Self-intersection of the trimmed surface.

Figure 8: Results for the *Partial-Glove* dataset. HLLE was used for the $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ parameterization while RBFs+NURBS were used for the back-mapping.

big rectangular patch as expected.

Fig. 8 shows the results for a *less-developable* dataset. The *Partial-Glove* mesh (fig. 8(a)) presents higher curvatures as well as several saddle-points between the fingers. HLLE was used for computing the parameterization $U$ while NURBS+RBFs were used for computing the back-mapping. The resulting trimmed surface $FS, L_0$ approximates adequately the original mesh $M$ (fig. 8(b)) however, the surface self-intersects outside the original domain. A close view of the self-intersection can be appreciated in fig. 8(c).

# 5 Conclusions

We presented a procedure for parameterizing a connected triangular mesh $M$ with borders using trimmed surfaces. This procedure maps first the triangular mesh to $\mathbb{R}^2$ using manifold learning techniques and then maps back a rectangular grid on the parameter space to $\mathbb{R}^3$ by interpolating and extrapolating the original surface.

For the implementation, three manifold learning algorithms were tested: i) Isomap, ii) Laplacian Eigenmaps and iii) HLLE. Additionally, the back-mapping $S : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ was computed by following i) a NURBS interpolation/extrapolation approach, ii) an RBFs surface approach or a NURBS+RBFs representation approach.

The surface trimming $\{L_0, L_1, \cdots, L_m\}$ was achieved by re-sampling the LOOPs bounding the mesh in the parameter space and then mapping back such re-sampling to the parametric surface. In order to get the LOOPs of the original surface, a B-REP of $M$ is first computed.

Several datasets were used for testing the algorithm. Isomap and HLLE presented in general good results for *developable* meshes since low geometric distortions are induced through the mapping to $\mathbb{R}^2$. On the other hand, Laplacian Eigenmaps presented visually bad results due to high distortions of such mapping. However, we showed how Isomap could fail in

some cases of high *developability* if the underlying parameterization presents a lot of non-convexities.

Finally, we illustrated how *less-developable* datasets could present self-intersections of the final trimmed surface which it is not desirable in CAD CAM CAE applications since manifoldness of the representation is lost.

Two problems still remain in the context of mesh parameterization: i) our procedure does not guarantee that the resulting trimming surface is a 2-manifold and ii) since most triangular surfaces are not *developable*, algorithms for automatic segmentation of the surface may be considered.

## Acknowledgements

We want to thank the CAD CAM CAE research group at *Universidad EAFIT* for providing different meshes and pre-processing them as well as providing B-REP and boundary identification algorithms for computing the trimming curves.

## Datasets

The following datasets were downloaded from the public sites: `http://www.cse.buffalo.edu/~jryde/cse673/04.html`, `http://gpeyre.github.io/numerical-tours/matlab/fastmarching_4bis_geodesic_mesh/` and `http://liris.cnrs.fr/meshbenchmark/`. The rest of the datasets are generated in the *Laboratorio de CAD CAM CAE* at *Universidad EAFIT*.

The authors declare that there is no conflict of interests regarding the publication of this article.

## References

[1] G. Li, C. Ren, J. Zhang, and W. Ma, "Approximation of Loop Subdivision Surfaces for Fast Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 4, pp. 500–514, 2011.

[2] L. Yang, D. He, and Z. Zhang, "Construct G1 Smooth Surface by Using Triangular Gregory Patches," in *Image and Graphics, 2009. ICIG '09. Fifth International Conference on*, pp. 577–580, Sept. 2009.

[3] C. Dyken, M. Reimers, and J. Seland, "Real-Time GPU Silhouette Refinement using Adaptively Blended Bézier Patches," *Computer Graphics Forum*, vol. 27, no. 1, pp. 1–12, 2008.

[4] Z. Zhang, Z. Wang, and D. He, "A New Bi-cubic Triangular Gregory Patch," in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 2, pp. 1003–1007, 2008.

[5] T. Boubekeur and S. Christophe, "Scalar Tagged PN Triangles," in *Eurographics 2005 Short Presentations*, pp. 17–20, 2005.

[6] Z. Mao, L. Ma, and W. Tan, "A Modified Nielson's Side-Vertex Triangular Mesh Interpolation Scheme," in *Computational Science and Its Applications ICCSA 2005* (O. Gervasi, M. Gavrilova, V. Kumar, A. Laganà, H. Lee, Y. Mun, D. Taniar, and C. Tan, eds.), vol. 3480 of *Lecture Notes in Computer Science*, pp. 776–785, Springer Berlin Heidelberg, 2005.

[7] A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell, "Curved PN triangles," *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pp. 159–166, 2001.

[8] D. Acosta, O. Ruiz, S. Arroyave, R. Ebratt, C. Cadavid, and J. Londono, "Geodesic-based manifold learning for parameterization of triangular meshes," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, pp. 1–14, 2014.

[9] J. Tierny, J. Daniels II, L. G. Nonato, V. Pascucci, and C. T. Silva, "Interactive Quadrangulation with Reeb Atlases and Connectivity Textures," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, pp. 1650–1663, Oct. 2012.

[10] H. Yu, T.-Y. Lee, I.-C. Yeh, X. Yang, W. Li, and J. J. Zhang, "An RBF-Based Reparameterization Method for Constrained Texture Mapping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 7, pp. 1115–1124, 2012.

[11] Y. Guo, J. Wang, H. Sun, X. Cui, and Q. Peng, "A novel constrained texture mapping method based on harmonic map," *Computers & Graphics*, vol. 29, no. 6, pp. 972–979, 2005.

[12] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," in *NIPS*, pp. 585–591, 2001.

[13] A. Sheffer and E. de Sturler, "Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening," *Engineering with Computers*, vol. 17, no. 3, pp. 326–337, 2001.

[14] X. Zhao, Z. Su, X. D. Gu, A. Kaufman, J. Sun, J. Gao, and F. Luo, "Area-Preservation Mapping using Optimal Mass Transport," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2838–2847, 2013.

[15] G. Zou, J. Hu, X. Gu, and J. Hua, "Authalic Parameterization of General Surfaces Using Lie Advection," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2005–2014, 2011.

[16] N. Pietroni, M. Tarini, and P. Cignoni, "Almost Isometric Mesh Parameterization through Abstract Domains," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 4, pp. 621–635, 2010.

[17] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler, "A Local/Global Approach to Mesh Parameterization," *Computer Graphics Forum*, vol. 27, no. 5, pp. 1495–1504, 2008.

[18] X. Sun and E. R. Hancock, "Quasi-isometric parameterization for texture mapping," *Pattern Recognition*, vol. 41, no. 5, pp. 1732–1743, 2008.

[19] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parameterizations of Surface Meshes," *Computer Graphics Forum*, vol. 21, no. 3, pp. 209–218, 2002.

[20] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science (New York, N.Y.)*, vol. 290, pp. 2319–2323, 2000.

[21] N. Pietroni, C. Massimiliano, P. Cignoni, and R. Scopigno, "An Interactive Local Flattening Operator to Support Digital Investigations on Artwork Surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1989–1996, 2011.

[22] X.-F. Zhu, P. Hu, Z.-D. Ma, X. Zhang, W. Li, J. Bao, and M. Liu, "A new surface parameterization method based on one-step inverse forming for isogeometric analysis-suited geometry," *The International Journal of Advanced Manufacturing Technology*, vol. 65, no. 9-12, pp. 1215–1227, 2013.

[23] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[24] F. L. Bookstein, "Principal warps: thin-plate splines and the decomposition of deformations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, pp. 567–585, Jun 1989.