Mesh Processing using On-the-Fly Connectivity Reconstruction given by Regular Triangulations

Fernando B. Pires*, Carlos A. Dietrich†, João L. D. Comba†, Luis Gustavo Nonato*
*ICMC-USP, São Carlos, SP, Brazil
†Instituto de Informática - UFRGS

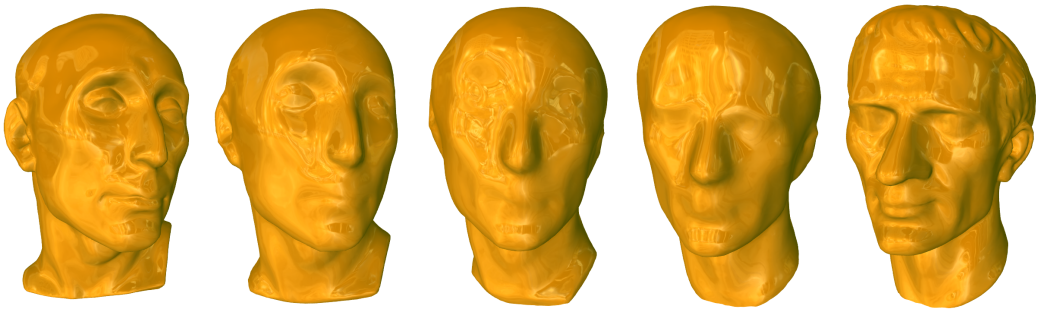

Figure 1. Morphing sequence between the Nicolo and Julius datasets. For each dataset, we create a Regular Triangulation (RT) using a linear programming procedure that assign weights to vertices, thus allowing the connectivity to be represented implicitly. In this example, the morphing between the two meshes is easily defined by computing intermediate vertices and weights, and reconstructing the connectivity by simply computing the RT.

***Abstract*—Several applications in visual and geometric computing require the ability to modify graphical models in such way that geometric queries or high quality renderings can be generated with great accuracy. Polygonal meshes are the popular choice of representation, and several mesh processing operations such as morphing, level-of-detail or deformation, among others, introduce challenges on how this task can be performed. A common problem that arises in such applications is that the result of a mesh processing operation can either require a costly mesh re-computation, thus impairing real-time usage, or it requires constant updates and additional storage to keep several information required to perform this task. In particular, the topological information is often harder to maintain updated, since it is often destroyed or modified during such operations. In this work we propose a new framework to reconstruct connectivity information in such way that the quality of the mesh can be recovered. The connectivity retrieval is accomplished by assigning weights to the vertices of the triangulation, converting it in a regular triangulation. Once weights have been computed, the connectivity can be rebuild by algorithms devoted to construct regular triangulation. The effectiveness of our new paradigm is illustrated through two mesh processing applications: mesh morphing and level-of-detail rendering.**

***Keywords*-regular triangulation; mesh morphing; mesh representation;**

I. Introduction

The manipulation of polygonal meshes raise several aspects that need to be carefully addressed in many geometric processing operations. One important point is enforcing topological consistency, which plays vital role in applications where polygonal meshes undergo transformations (e.g. morphing, level-of-detail or subdivision surfaces). The simple need of inserting and removing vertices requires local reconstruction of connectivity to keep a valid mesh representation. In many cases, intensive mesh manipulation leads to a complete loss of the original incidence and adjacency relations, being necessary a history mechanism to recover the original mesh. Depending on the order that operations are performed, different representation might be obtained if some vertices are removed from a surface mesh and later reinserted. Preserving such consistent reconstruction of connectivity information can be highly desirable, specially in cases where much effort was spent on creating a quality mesh. However, only a very restrictive class of surface meshes can be algorithmically reconstructed from the geometry of its vertices (i.e. disregarding connectivity information).

In this work we address the problem of reconstructing the connectivity of a mesh without storing the original incidence and adjacency information. Our proposal is based on a linear programming algorithm that converts a mesh into a Regular Triangulation (RT), which consists of a generalization of Delaunay Triangulation with weights assigned to vertices. We demonstrate with this construction that the connectivity information becomes implicitly encoded at vertex weights, and can be algorithmically recovered at any given time.

Mesh operations using this framework are simply performed by manipulating vertex weights. For instance, to remove a vertex we reduce its weight to an appropriate negative value, which causes its automatic elimination. In Figure 1 we illustrate the application of our framework to compute the morphing between two different meshes.

We can summarize the main contributions of our work as: 1) we provide a theoretical and computational framework that shows how a triangulation can be converted into a regular triangulation, which is achieved by transforming the geometrical problem into a well-posed linear programming problem; 2) we show how the proposed framework, combined with mesh parametrization methods, can be used to reconstruct connectivity information that is implicitly stored at the regular triangulation; 3) we validate the potential of our approach in two important mesh operations, namely, mesh morphing and level-of-detail simplification.

II. Related Work

Regular Triangulations Regular Triangulations (also known as Weighted Delaunay Triangulations) and Power Diagrams are basic geometrical tools that have been often used for problems related with surface [1] and quality mesh generation for numerical simulations [2], [3]. The widespread use of regular triangulations and power diagrams in mesh generation is related to the advances in the algorithms that handle these geometrical entities, which results in a solid foundation from both theoretical [4] and computational [5], [6] sides.

The essential property of allowing the assignment of weights to points renders regular triangulation and power diagram as extremely flexible and dynamic geometrical structures, whose usefulness goes beyond mesh generation. Several theoretical and computational frameworks have been developed towards understanding and exploiting such potentiality. One example is the work of Edelsbrunner [7], which explores the equivalence between regular triangulations (power diagrams) and the boundary of convex polyhedra in one dimension above (see also [8], [9]) to derive an algebra of circles from which a new paradigm for designing smooth surfaces is formulated. The relationship between regular triangulation and convex polyhedra has also been investigated by Massada et al. [10] to propose an output-size sensitive algorithm to enumerate all regular triangulations. From a theoretical point of view, it has been shown a close connection to well established algebraic concepts, such as the relation of regular triangulations and convex polyhedra to Gröbner bases [11].

Mesh Processing Many mesh processing operations create versions of input meshes with different connectivity information. For instance, morphing between meshes requires generating an intermediate mesh that combines the geometric and topological aspects of the input meshes. Common to all methods discussed in the literature is the definition of a correspondence between vertices and connectivity information. This is often done by first creating a mapping into an unified space in which the geometry and connectivity from both meshes are mapped, and by a synthesis algorithm that generates an intermediate mesh from this space. Several papers discuss aspects that range from how and when such mappings are defined [12], [13], [14], to how mesh information can be mapped to a parametric space, such as the inter-surface mapping described in [15]. There are several variations on how the connectivity information is generated for intermediate meshes [16], [17], [18], [19], [20]. In this paper we offer an implicit way to reconstruct connectivity information that can be used in conjunction with several of the methods proposed. We also investigate how our proposal can be applied to generating different level-of-detail representations of a given mesh, discussed in several work and summarized in the book by Lubke et al[21]. Unlike local operations such as vertex collapses that incrementally change topology information, our proposal provides a more general way to locally reconstruct connectivity information that allows more complex topological changes, which might be necessary to provide a smoother simplification process.

III. Regular Triangulation and Weight Calculation

In this section we formally introduce basic definitions and properties of the mathematical framework of RTs, and describe how they can be employed towards representing any triangulation. We restrict our presentation to the two-dimensional Euclidean space $\mathbb{R}^2$, but a more general description can be found in [9].

A. Basic Concepts

A *weighted point* $s \in \mathbb{R}^2 \times \mathbb{R}$ is defined by its location $p \in \mathbb{R}^2$ and weight $w \in \mathbb{R}$, and can be interpreted as a circle with center p and radius $w^{\frac{1}{2}}$. In this discussion we assume positive weights, although there is no theoretical inconvenient in dealing with negative weights (circles with imaginary radius). The *power distance* between two circles s_i and s_j is defined as $\mathtt{pow}(s_i, s_j) = d^2(p_i, p_j) - w_i - w_j$, where $d(\cdot, \cdot)$ is the Euclidean distance. In particular, the power distance between s_i and a point $x \in \mathbb{R}^2$ is given by $\mathtt{pow}(s_i, x) = d^2(p_i, x) - w_i$. If s_i and s_j are two non concentric circles, the set of points x such that $\mathtt{pow}(s_i, x) = \mathtt{pow}(s_j, x)$ is called the *chordale* of s_i and s_j. It is well known that the three chordales defined by the circles s_i, s_j, and s_l (with non-collinear centers) intersect in a common point p that fulfills $w = \mathtt{pow}(s_i, p) = \mathtt{pow}(s_j, p) = \mathtt{pow}(s_l, p)$. Furthermore, the circle s with center p and radius $w^{\frac{1}{2}}$ satisfies $\mathtt{pow}(s_i, s) = \mathtt{pow}(s_j, s) = \mathtt{pow}(s_l, s) = 0$ and is called the *orthocircle* of s_i, s_j and s_l.

Let $h_{s_i}(s_j)$ denote the closed half-space bounded by the chordale of s_i and s_j that contains the points with the

smallest power distance with respect to s_i. The *power cell* of s_i is given by

$$V(s_i) = \bigcap_{j,\, j \neq i} h_{s_i}(s_j) \tag{1}$$

Letting $S = \{s_1, \ldots, s_n\}$ be a set of non-concentric circles, the power cells of the circles in S comprise a partition of $\mathbb{R}^2$ in convex polygons, the so-called *Power Diagram* ($PD(S)$). The Power Diagram coincides with the usual Voronoi Diagram if all circles in S have the same radius. However, the Power Diagram produced by circles with different radius may not satisfy the containment condition, that is, a circle s_i may not be contained in its power cell. Furthermore, depending on the weights, a circle can give rise to an empty power cell. In this case, such circle is called *redundant*.

If we suppose that the center of the circles in S are not collinear, then the intersection of three (or more) power cells is either empty or a vertex of $PD(S)$. It is not difficult to realize that a vertex of $PD(S)$ is the center of the orthocircle defined by at least three circles whose power cells have non-empty intersection. In a non-degenerate case, each vertex v of $PD(S)$ is defined by the intersection of exactly three power cells. In such case, the center of the three circles whose power cells intersect in v define a triangle and the union of all triangles makes up a simplicial complex, called *Regular Triangulation* (we are using the term "Regular Triangulation" as synonymous with "weighted Delaunay triangulation, though some authors make distinction between them). In fact, such a correspondence leads to a duality relationship that associates circles to power cells, power cell edges to triangle edges and Power Diagram vertices to triangles, in this last case the Power Diagram vertex is called the orthocircle of the associated triangle. Note that redundant circles do not have dual power cells, and thus do not appear at the RT. From the duality property we can obtain RT from the Power Diagram (and vice versa) in $O(n)$. In the two-dimensional case existing algorithms can compute both structures in $O(n \log n)$ [6] (expected time).

An important fact that is deeply exploited in this work is the strong relation between Power Diagrams in $\mathbb{R}^2$ and arrangement of planes in $\mathbb{R}^3$. Such a relationship can be obtained by associating to each circle $s_i \in S$ a non vertical plane in $\mathbb{R}^3$ which is defined by the following function:

$$\pi_{s_i}(x) = 2 < x, p_i > - < p_i, p_i > + w_i \tag{2}$$

where $< \cdot, \cdot >$ is the usual dot product in $\mathbb{R}^2$. It can be shown that a point $x \in \mathbb{R}^2$ lies in $V(s_i)$ iff, at x, $\pi_{s_i} \geq \pi_{s_j}, i \neq j$. Thus, $PD(S)$ is the vertical projection of the Upper Envelope of the planes π_{s_i}, which is a convex polyhedron in $\mathbb{R}^3$ (for a detailed definition of Upper Envelope and its relation with the Power Diagram see [8]). Figure 2 illustrates the correspondence between Power Diagram, Regular Triangulation, and Upper Envelope.

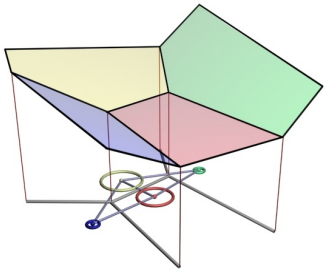

Figure 2. Power Diagram, Regular Triangulation and Upper Envelope.

B. The Dynamics of Power Diagrams

Equation (2) helps us to understand the behavior of $PD(S)$ when the radius (weight) of a circle changes. Suppose that both $PD(S)$ and $RT(S)$ have already been computed from a set of circles S and let s_i be a circle of S. When the radius of s_i increases the plane π_{s_i} moves up, enforcing the center of the orthocircles that comprise the vertices of $V(s_i)$ to be moved away from p_i (center of s_i), as illustrated in figure 3.

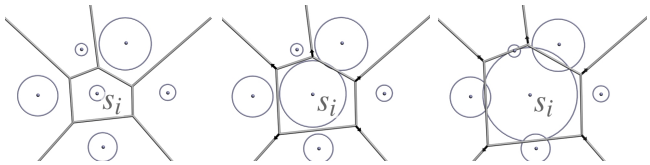

Figure 3. The center of the neighbor orthocircles move away from the center of s_i when w_i increases.

Due to the duality relationship, the RT remains unchanged until the increase of w_i gives rise to a degenerate case, that is, two diagram vertices are collapsed into a single one, causing the intersection of more than three power cells in such vertex. When a degenerate case occurs, any further increase of w_i gives rise to an edge flipping in the RT, as shown in Figure 4. A similar phenomenon occurs when the weight w_i is reduced.

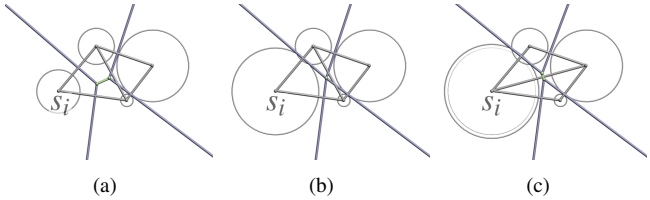

Figure 4. Edge flipping after a degenerate case.a) original configuration b) the increase in the weight of s_i gives rise to a degenerate case; c) further increase of w_i causing an edge flip.

The dynamic process described above allows us to realize how to control the weights so as to avoid edge flipping. Furthermore, the understanding of such a dynamics is the main key to convert any given triangulation into a RT, as we shall see in the next subsection.

C. Weight Calculation

Let $P = \{p_1, \ldots, p_n\}$ be a set of points in $\mathbb{R}^2$ and T be a simplicial decomposition of the convex hull of P. Our goal is to compute weights for the points in P so as to convert T into a RT.

Let $\sigma = [p_i, p_j, p_k]$ and $\tau = [p_j, p_k, p_l]$ be two triangles of T sharing a common edge e with ends p_j and p_k. Suppose that weights w_j, w_k, w_l have already been assigned to the vertices of τ. A degenerate case occurs if we tune w_i in such a way that the triangles σ and τ share the same orthocircle. Using (2) one can build the system of equations (3) whose solution supplies x (the center of the orthocircle) and the value of w_i that causes the degeneracy:

$$\begin{cases} \pi_{s_i}(x) - \pi_{s_j}(x) = 0 \\ \pi_{s_i}(x) - \pi_{s_k}(x) = 0 \\ \pi_{s_i}(x) - \pi_{s_l}(x) = 0 \end{cases} \tag{3}$$

By solving the system (3) one obtain:

$$w_i = -\frac{a_j(c_l b_k - c_k b_l) + a_k(c_j b_l - c_l b_j) + a_l(c_k b_j - c_j b_k)}{a_j(b_k - b_l) + a_k(b_l - b_j) + a_l(b_j - b_k)} \tag{4}$$

where $a_\star = 2(p_i^x - p_\star^x), b_\star = 2(p_i^y - p_\star^y)$, and $c_\star = < p_\star, p_\star > - < p_i, p_i > -w_\star$, $\star = j, k, l$ (p^x and p^y denote the Cartesian coordinates of p). It is important to point out that in equation (4) w_i depends linearly on w_j, w_k and w_l.

As formulated, the value of w_i computed from equation (4) gives rise to a degenerate case. However, based on the dynamics of RTs discussed before, if the weight of p_i is set as $w_i - \epsilon$, for any $\epsilon > 0$, the degeneracy can be removed, thus assuring the edge e in the RT. From this observation and the linear dependency of w_i regarding w_j, w_k and w_l, one can rewrite equation (4) as:

$$\alpha_i w_i + \alpha_j w_j + \alpha_k w_k + \alpha_l w_l \leq G + \epsilon \tag{5}$$

where the coefficients $\alpha_\star$, $\star = i, j, k, l$, and the term G can be obtained by an algebraic manipulation of (4).

By applying the above reasoning for each interior edge of T we obtain a linear system of inequalities from which one can obtain weights for points in P so as to convert T into a RT. In fact, such system of inequalities supplies a multitude of possible weights for the points in P. In order to reach an unique solution we can introduce an objective function that, jointly with the system of inequalities (5), gives rise to a linear programming problem, which can be solved by conventional methods such as the simplex method [22]. In our implementation we are assuming the sum of the weights as objective function, that is, we are looking for the solution that minimizes the weights summation. This particular choice is easy to implement and prone to produce solutions in which most weights are equal to zero, which can be desirable in some situations. It is worth mentioning that other objective functions could be employed. The development describe above allow us to state the problem of converting a given triangulation T into a RT as follows:

The RT Linear Programming Problem: Let $P = \{p_1, \ldots, p_n\}$ be a set of points in $\mathbb{R}^2$ and T a triangulation for the convex hull of P. T can be converted into a RT by associating to points in P the weight $\{w_1, \ldots, w_n\}$, which satisfy the following linear programming problem:

$$\begin{aligned} \textit{minimize}: \quad & Z = \sum_{s=1}^{n} w_s \\ \textit{subject to}: \quad & \alpha_i w_i + \alpha_j w_j + \alpha_k w_k + \alpha_l w_l \leq G_r + \epsilon \\ & w_s \geq 0 \\ & r = 1, \ldots, m;\ s = 1, \ldots, n \end{aligned} \tag{6}$$

where m is the number of internal edges of T and i, j, k, l are the indices of all four points p_i, p_j, p_k, p_l defining a quadrilateral whose diagonal is an edge of T.

Although there is no theoretical guarantee regarding the solution of the LP problem above (in fact there are cases where no solution exists), in all experiments we have performed we were able to nd a set of weights for the vertices that allowed to rebuild the original triangulation correctly.

As discussed in the introduction, the topological structure of the triangulation T can be discarded after assigning weights to the points in P, as the original triangles and edges can be recovered algorithmically, as for example by using the incremental ipping algorithm [6]. This result is very useful in a wide range of applications, two of which are discussed in the next section.

IV. Mesh Operations Using RTs

In this section we summarize a recipe for using regular triangulations to perform mesh operations. Here we assume that a valid mapping of the input mesh to a 2D convex parametric space is possible, and limit the application of the framework of regular triangulations to a two-dimensional case. Given an input 3D mesh M, we first create a parametrization to a 2D convex domain. Using the mesh vertices mapped to the parametric space and the connectivity of M we define a 2D triangulation, which is converted into a 2D regular triangulation as described in the previous section.

At this point the connectivity is encoded at the weights of the regular triangulation, and therefore the original connectivity can be discarded. All further mesh processing operations operate over the parametric domain. Since this domain is always defined, we guarantee that we always have a valid triangulation at the end. The generation of a new mesh with different connectivity information is obtained by simply discarding vertices from the triangulation in the parametric domain or setting a given weight to a negative value. We rebuild the regular triangulation from the remaining vertices and their corresponding weights (several algorithms are described in the literature for this, we are

currently using the one provided by the CGAL library). Figure 5 illustrates an example, showing a given mesh (a), its parametrization to a 2D domain (b), the removal of vertices in the parametric space (c) and the resulting mesh with new connectivity obtained (d).

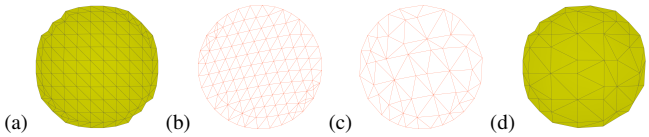

Figure 5. Example of different connectivity reconstructions using regular triangulations. Given a input mesh (a), deletion of vertices is performed in its parametric space (b), which results a new set of vertices that is used to create a new regular triangulation (c), which produces a new connectivity that is copied to corresponding vertices of the resulting mesh (d).

V. Applications

In this section we validate the framework of regular triangulations in two mesh processing operations: morphing between meshes, and mesh simplification to generate continuous level-of-detail. Our focus in these experiments is to illustrate the usefulness and potential of the framework, stressing the importance of having an algorithmic way to reconstruct connectivity information.

A. Mesh Morphing

Mesh morphing is an important mesh operation that requires generating an intermediate valid mesh between a source and a destination meshes. Most morphing algorithms require a correspondence between source and destination meshes, which allows geometric coordinates of the vertices to be naturally interpolated, but the hardest problem to solve is the reconstruction of a valid connectivity for intermediate meshes. The framework of regular triangulations allows a direct way to reconstruct the topology of intermediate meshes without the need of explicitly mesh manipulation. In Figure 1 we illustrate the steps required to perform a 2D morphing operation.

In order to further illustrate our proposal, we tackle the even more challenging problem of creating the morphing between input meshes that are represented by multiple charts. The need to handle multiple charts is challenging since the continuity across charts need to be kept consistent to avoid cracks. The algorithm for mesh morphing is described in the sections below, and involve the following steps:

- **Chart correspondence**: morphing is performed between pairs of corresponding charts in the source and destination meshes
- **Enforcing Boundary Consistency**: each chart is represented as a regular triangulation, but vertices that are shared by multiple charts (boundary vertices) need a special treatment to enforce mesh continuity.
- **Vertex correspondence**: mapping that involves both geometric coordinates, as well as weight information given by the regular triangulation and additional information to enforce boundary continuity
- **Creating Intermediary Meshes**: producing the output mesh at any intermediate point.

1) Chart correspondence: Given a source mesh M_s and a destination mesh M_d, we process them in such way that each one is broken into the same number of charts, with a 1-1 correspondence between charts in the source and destination meshes (Figure 6). We encode the topology defined by each multi-chart in graphs G_s and G_d, which are by construction homeomorphic to each other and have a direct correspondence among vertices. Enforcing the homeomorphism can be done manually or using several algorithms described in the literature, such as the inter-surface mapping described in [15]. Since our goal is to simply validate the technique, we employ a simple iterative procedure controlled by the user. It starts by the user specifying n corresponding pairs of control points in both source and destination meshes. An explicit mapping is defined automatically between each pair specified, and charts are created by a region growing algorithm using control points as seeds.

Once both G_s and G_d are created, we perform an additional relaxation procedure to find a rigid transformation that reduces geometric distortion between vertex coordinates. This is accomplished by setting a simple mass-spring relaxation procedure, but several other strategies described in the literature can also be used. The mass-spring system is formed by masses defined over the control points specified by the user over M_s e M_d, and two types of springs, connecting adjacent vertices in the same mesh, and connecting a vertex in the source mesh to the corresponding vertex in the destination mesh (and vice-versa). The first set of springs enforces geometric proximity inside a mesh, while the second minimizes the geometric distortion during morphing.

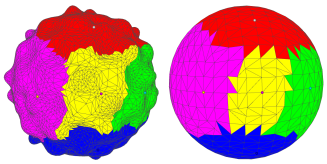

Figure 6. Creating corresponding charts. A source mesh (left) is broken in the same number of patches as the (right) destination mesh, and each chart from the source mesh is associated a chart in the destination mesh.

2) Enforcing Boundary Consistency: Each chart in the source and destination meshes is processed individually as if it was a single mesh, and converted into a regular triangulation as described in the previous section. As a result, we obtain one regular triangulation for each chart, each containing for each vertex its geometric coordinates (v_x, v_y, v_z), parametric coordinates (v_u, v_v) and weight v_w.

Vertices that appear at the boundary of charts (boundary vertices) deserve special attention, since they belong to sev-

eral charts, and therefore may have possibly been associated a different weight to each chart they belong to. In order to properly handle boundary vertices, we store in addition to the local weight a second weight v_{wmin}representing the minimum weight of all charts the boundary vertex appears. This additional information guarantees that boundary vertices have the same behavior during morphing because v_{wmin} is used to decide when the vertex is included or removed from the mesh. If this decision was solely based on the local weight of the vertex inside a patch, it would be possible that a given vertex could suffer a different morphing in each patch that contains it, thus producing inconsistent boundary transitions between charts.

3) Vertex correspondence: Only the correspondence between charts from the source to the destination mesh is established at this point. For the morphing to be performed, it is necessary to establish the correspondence between every vertex v_s in a given chart of M_s to a vertex v_d in the corresponding chart in M_d. This is accomplished in a per-chart basis, that starts by first finding a corresponding corner vertex in both the source and destination chart. Once this vertex is found, corresponding boundary loops are recovered by a traversal along boundary vertices. The steps of this algorithm can be summarized as follows:

1) Find a corner boundary vertex v_s in M_s that is incident to $k > 2$ charts.
2) Find the corresponding boundary vertex v_d in the destination mesh (this is enforced during chart creation).
3) Reconstruct boundary loops by tracking common boundaries of 2 clusters. Mark v_s and v_d as visited, and push $\binom{k}{2}$ searches on a stack having v_s and v_d as starting points, and every pair of clusters shared by them as directions.
4) While the stack is not empty, pop a given search and traverse simultaneously M_s and M_d in the given direction tracing the boundary between the two clusters until a new corner vertex is found. If this vertex was not visited, repeat step 2 using this vertex as starting point.

Once boundary loops are reconstructed, mapping internal vertices across charts is easily defined by finding their closest correspondence in parametric space of source and destination charts. The mapping of boundary vertices, however, is more involved since it is performed individually for each corresponding boundary loop in the source and destination meshes, which most likely have a different number of vertices.

Assuming for the sake of the discussion that the boundary loop of M_s has more vertices than M_d, then we need to map several vertices of M_s to one given vertex in M_d. In order to properly remove vertices at intermediate morphing steps, we need to gradually remove vertices from M_s until only one vertex is maintained. This is accomplished by defining an *invalid* weight mapping (a mapping to a negative weight, which automatically removes a vertex from the regular triangulation). This invalid mapping is defined to all but one vertices of M_s that are closer in the parametric space to a given vertex in M_d (the only valid mapping is to the closest vertex of M_s). Invalid mappings are computed by a backward search that process all vertices of M_d to find the closest vertices of M_s. Figure 7 illustrates this mapping process.

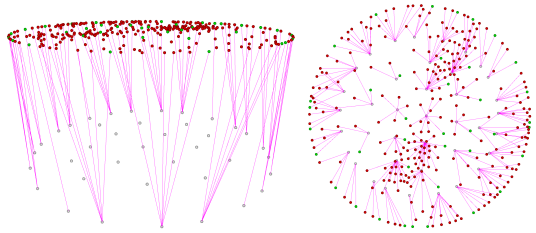

Figure 7. Mapping in parametric space. Two different views of the mapping for source and destination meshes of Figure 6: (left) side-view with only boundary mappings (right) top-view showing boundary and interior vertices. Red vertices are scheduled to be removed and green vertices to be maintained.

4) Creating Intermediary Meshes: Every corresponding pairs of charts in the source and destination meshes are traversed. The mapping defined for each vertex v_s in M_s to a vertex v_d in M_d as described above is used to generate the geometric and parametric coordinates, as well as the resulting vertex weight. This is accomplished as follows:

- Check if the mapping of v_s to v_d is invalid. If this is the case, geometric and parametric coordinates are interpolated between v_s and v_d, and the weight of v_s is interpolated to a given negative value (user-defined). This will in essence make the vertex disappear once the weight becomes negative. Otherwise (valid mapping), all attributes of v_s are interpolated to the attributes of v_d.
- The resulting weight is compared against a given threshold. If the value is greater or equal, the vertex is inserted in a regular triangulation using the interpolated attributes.
- Once all vertices in a given chart were processed, the topology defined by the regular triangulation is used to reconstruct the intermediary mesh

Figure 8 shows the results for two morphing sequences.

B. Level-of-Detail

Level-of-Detail generation illustrates another problem that might benefit from the on-the-fly topology generation provided by regular triangulations. Given an input mesh, we need to obtain a simplified valid mesh in a given level-of-detail. We first implemented a simplification algorithm that allows the user to select certain regions of interest over the input mesh that are to be preserved as much as possible in the process. Remaining regions are simplified based on a

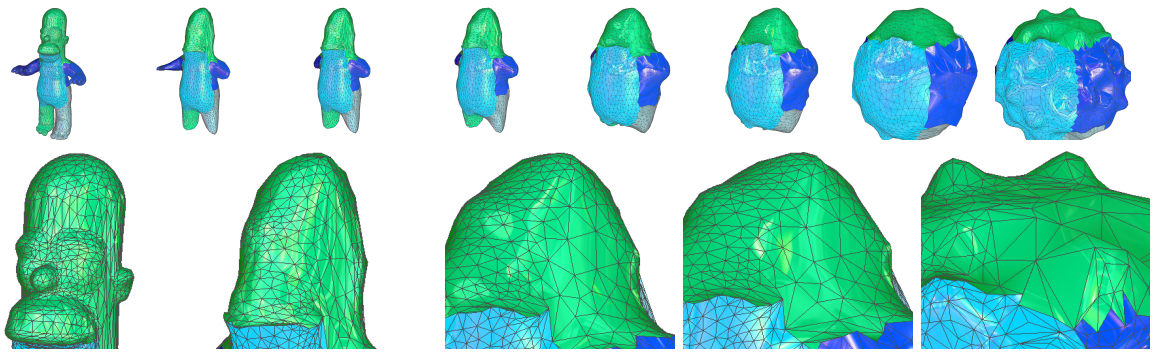

Figure 8. Multi-chart morphing sequence between the Homer and Bucky Ball datasets. Each mesh is segmented into corresponding charts and morphing is applied as described between each chart. An intermediate morphing sequence can be generated at any instant between the two mesh models. First row contains 8 intermediate results, and second and third rows display detailed results of each of these results.

simplification level informed by the user. The contribution that regular triangulations bring to this problem is that a correct topology of the simplified mesh is automatically reconstructed to any desired simplification level.

For this application we consider a mesh without partition into charts. The first simplification algorithm we implemented involves user interaction, and starts by the user indicating the control points representing regions of interest in the original mesh. All vertices in the mesh are processed and are associated a new attribute representing the closest distance to a given interest point. In addition to this value, the weights for each vertex in the regular triangulation are computed in the parametric space. The generation of the simplified mesh starts by the user informing a parameter that represents a given distance that will define which vertices will be preserved in the construction of the mesh. Every vertex that is within the distance given by this parameter to its corresponding interest point is sent to the regular triangulation. The regular triangulation is computed and displayed to the user. Top row of Figure 9 illustrates the results of this approach.

We also tried a variation of the algorithm that would not depend on user input, but instead was based on local geometric features of the mesh such as curvature. The procedure is simple: the curvature at each vertex is approximated and associated as an additional parameter (similar to the closest distance parameter used before). The remaining of the algorithm is the same, the curvature is compared against a threshold informed by the user. Bottom row of Figure 9 show the results of this approach for different levels of detail.

VI. Conclusion

In this work we introduced a methodology for representing and manipulating triangular meshes. The method is based on a theoretical and computational framework that computes appropriated weights for vertices of a given triangulation in such way that it defines a RT, thus allowing to disregard connectivity information, since connectivity is implicitly stored at the weights of mesh vertices. The flexibility and ease of implementation of this new approach has been investigated in applications such as level-of-detail and mesh morphing. Unlike similar alternatives to perform those operations, our solution does not require special care to keep the mesh connectivity consistent, since this is automatically enforced by our methodology.

We are currently working in two other problems using the theoretical results presented here. The first one concerns mesh compression, since connectivity information can be completely discarded. The second problem is multi-scale tetrahedral mesh representation. By extending the proposed theory to the three-dimensional case we would be able to to deal with tetrahedral mesh simplification and refinement in a broader sense, thus being capable of handling any kind of tetrahedral mesh algorithmically.

Acknowledgment

This work was supported by grants Fapesp-Brazil (#2008/03349-6), CNPq-NSF (#491034/2008-3) in the framework of Instituto Nacional de Ciłncia e Tecnologia em Medicina Assistida por Computao Cientfica (CNPq, Brazil).

References

[1] N. Amenta, S. Choi, and R. Kolluri, "The power crust," *Computatinal Geometry*, vol. 19, pp. 127–153, 2001.

[2] S.-W. Cheng, T. K. Dey, and E. A. Ramos, "Delaunay refinement for piecewise smooth complexes," in *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1096–1105.

[3] S.-W. Cheng, T. Dey, and T. Ray, "Weighted delaunay refinement for polyhedra with small angles," in *14th International Meshing Roundtable*. Springer-Verlag,, 2005, pp. 11–14.

[4] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng, "Sliver exudation," in *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*. New York, NY, USA: ACM Press, 1999, pp. 1–13.

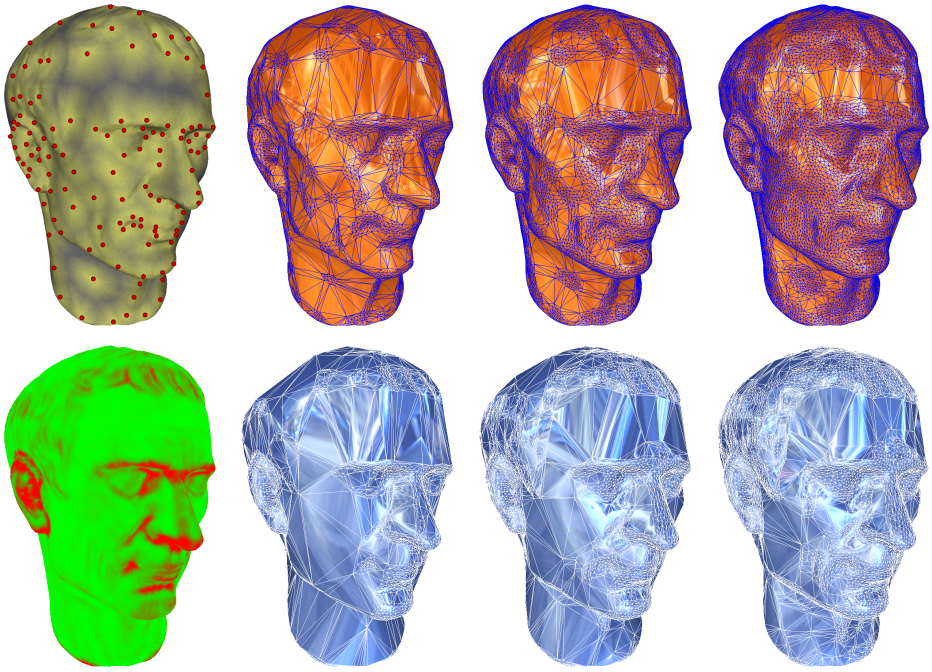

Figure 9. Mesh simplification with user specified regions of interest (first row) and curvature (second row). (a) interest points or curvature, (b)-(d) increasing levels of detail of the mesh.

[5] M. Vigo, N. Pla, and J. Cotrina, "Regular triangulations of dynamic sets of points," *Comput. Aided Geom. Des.*, vol. 19, no. 2, pp. 127–149, 2002.

[6] H. Edelsbrunner and S. Shah, "Incremental topological flipping works for regular triangulations," *Algorithmica*, vol. 15, pp. 223–241, 1996.

[7] H. Edelsbrunner, "Deformable smooth surface design." *Discrete & Comput. Geom.*, vol. 21, no. 1, pp. 87–115, 1999.

[8] H. Edelsbrunner and R. Seidel, "Voronoi diagrams and arrangements," *Discr. Comput. Geom.*, vol. 1, pp. 25–44, 1986.

[9] F. Aurenhammer, "Power diagrams: Properties, algorithms and applications," *SIAM J. Comput.*, vol. 16, pp. 78–96, 1987.

[10] T. Masada, H. Imai, and K. Imai, "Enumeration of regular triangulations," in *Symposium on Computational geometry, SCG*, 1996, pp. 224–233.

[11] J. DeLoera, J. Sturmfels, and R. Thomas, "Gröbner bases and triangulations of the second hypersimplex," *Combinatorica*, vol. 15, no. 3, pp. 409–424, 1995.

[12] M. Floater and C.Gotsman, "How to morph tilings injectively," *J. Comput. Appl. Math.*, vol. 101, pp. 117–129, 1999.

[13] V. Surazhsky and C. Gotsman, "Morphing stick figures using optimized compatible triangulations," in *Pacific Graphics*, 2001, p. 40.

[14] J. Danciger, S. L. Devadoss, and D. Sheehy, "Compatible triangulations and point partitions by series-triangular graphs," *Comput. Geom. Theory Appl.*, vol. 34, no. 3, pp. 195–202, 2006.

[15] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe, "Inter-surface mapping," in *SIGGRAPH '04*, 2004, pp. 870–877.

[16] M. Ahn and S. Lee, "Mesh metamorphosis with topology transformations," *pg*, vol. 00, p. 481, 2002.

[17] M. Ahn, S. Lee, and H.-P. Seidel, "Connectivity transformation for mesh metamorphosis," in *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2004, pp. 75–82.

[18] J. Parus and I. Kolingerová, "Morphing of meshes with attributes," in *SCCG '04: Proceedings of the 20th spring conference on Computer graphics*. New York, NY, USA: ACM Press, 2004, pp. 73–81.

[19] A. W. F. Lee, D. Dobkin, W. Sweldens, and P. Schröder, "Multiresolution mesh morphing," in *SIGGRAPH '99*, 1999, pp. 343–350.

[20] D. Zorin, P. Schröder, and W. Sweldens, "Interpolating subdivision for meshes with arbitrary topology," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1996, pp. 189–192.

[21] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney, *Level of Detail for 3D Graphics*. New York, NY, USA: Elsevier Science Inc., 2002.

[22] G. Dantzig and M. Thapa, *Linear Programming: 1: Introduction*, ser. Springer Series in Operations Research and Financial Engineering. Springer, 1997.