Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2013) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2013

Iterative method for edge length equalization

J. P. Peçanha[a], J. L. Souza Filho[a], M. B. Vieira[a,*], M. Lobosco[a], S. O. Dantas[a]

[a] *Universidade Federal de Juiz de Fora, Departamento de Ciência da Computação, 36036-900, Juiz de Fora-MG, Brazil*

Abstract

This paper presents a method for triangular surface remeshing to obtain new faces whose edge lengths are as close as possible to a target value m. The process uses as input a 2-manifold mesh with arbitrary geometry and topology. The proposed algorithm runs iteratively, automatically adjusting the necessary amount of vertices, and applies a global relaxation process using a variation of Laplace-Beltrami discrete operator. We introduce geometry constraints in order to preserve salient features of the original model. The method results on a grid with edge lengths near to m with low standard deviation, i.e. the vertices are uniformly distributed over the original surface. The dual space of the final triangular surface results in a trivalent, mostly hexagonal mesh, suitable for several applications.

Keywords: Mesh Optimization; Computational Geometry; Hierarchy and Geometric Transformations

1. Introduction

Improvement of digital representations of real objects is a major research area in computer graphics. Samples acquired with 3D scanners, for example, represent the surface of objects as point clouds. Depending on the application, these clouds should be manipulated in several ways. The distribution of the acquired points can be an issue for several applications. In physics, chemistry and numerical analysis, for example, some models must meet certain requirements in terms of geometric constraints, valid intervals for valencies and distances, overall vertex distribution, polygon area restrictions, etc., in order to perform stable simulations. Meshes with some degree of regularity are often the goal of most works about remeshing. Even with *computer-aided design* (CAD), one can generate poorly sampled, oversampled, redundant or distorted triangles. Modeling and triangulation methods generally aim the fidelity of the final surfaces rather than regularity or isotropy of the resulting meshes.

In this work, we propose a method to iteratively transform an existing arbitrary mesh into a version having regular edges with controlled length. It means that the user can state the desired average size for final edges. As a consequence, it tends to obtain equilateral triangles in regions with low curvature. Depending on the final edge length desired, the mesh is simplified or refined using stellar operations. A modified version of the Laplace-Beltrami discrete operator is then applied to relax the mesh in each iteration. Since some points are pushed outside the original surface by this operation, a projection step is performed in order to keep as much as possible the original shape. Thus, our method equalizes the distance between neighbour vertices trying to preserve the original topology and geometry.

*Corresponding author. Tel.: +55-32-2102-3311.
E-mail address: marcelo.bernardes@ice.ufjf.br.

As our results show, the equalized meshes fairly represent the original surface, even if the target edge length is far from the original average. The final result can be useful for some engineering and physics applications, such as nano structure simulation, for example, which has originally motivated this research. With a primal mesh with equalized edge lengths we can easily obtain a trivalent mesh by computing its dual. This kind of mesh has received some attention recently, generally by direct hexagonal faces generation through global parametrization [1]. From an application point of view, we argue that the problem of obtaining a trivalent mesh, reducible to the edge equalization problem, is more general. It allows the final mesh to have pentagons or heptagons wherever the original geometry requires. After the edge equalization, hexagons will naturally appear in smooth surface regions.

The main contribution of our work is a straightforward method for edge equalization, reasonably fast for modeling purposes, that admits target edge lengths in a wide interval around the original average. The final result tends to have low standard-deviation of edge lengths. We propose a new formulation for mesh relaxation based on the Laplacian transformation, which includes constraints to preserve original geometry. We also propose stellar transformations based on a priority list to avoid local geometric inconsistencies.

1.1. Related Work

Surface remeshing is commonly used to prepare a mesh for a given application, adapting it to fill specific requirements. The number of elements a surface has can be changed, i.e. to reduce the number of faces and vertices [2]. The change could also be done to the shape of the polygons [3, 4], number of regular vertices [5, 6], alignment of the polygons with areas of higher curvature [7, 8, 1] and the amount of elements in these areas [5, 3].

In a uniform distribution, the vertices are equally sampled through the whole model. The means to achieve this distribution are divided into two groups: explicit remeshing and global parametrization. With explicit remeshing, the model is globally and/or locally modified until a predefined objective is reached. Surazhsky & Gotsman [5] developed a simple way to generate uniform meshes. The mesh has the area of its triangles optimized. Local operations for vertex regularization are applied until a global minimum is found. There were no shown studies that focused on the length of the edges. The edges average of the result is the same of the model. Their work also introduced the concept of local parametrization that was further formalized by Ray *et. al* [9].

In [10], a method for explicit uniform remeshing, based on local modifications and followed by global relaxations is presented. Edges with length out of a chosen threshold are treated using stellar operations, as in the present work. Although efficient, there is no freedom on choosing this threshold. The authors say that the method guarantees its functionality only when the desired edge length is very close to the average of the initial model. This fact can be refuted considering the proposed method of the presented paper. There were no informations about the number of irregular vertices, standard deviation or final averages of the tested meshes.

Uniform remeshing by global parametrization is computationally slow, so it is common to restrict the object topology to models homeomorphic to the disc and sphere [11]. For arbitrary surfaces, a graph cut is calculated and then the model is opened in a plane [7]. From a global parametrization it is possible to uniformly redistribute the vertices according to a distance [12, 8]. It is also possible to resample the vertices so the new surface is formed by non-triangular polygons such as squares [7] or hexagons [1]. However, these approachs are very sensitive to the original meshes quality since the parametrization involves global optimization based upon geometric constraints, such as feature lines and extremalities, that must be very well detected. Tuning final edge lengths might also be difficult. In a modeling perspective, the possibility of non-convergence due to bad detection of some mesh's attribute might be a serious limitation.

2. Proposed Method

The input for the algorithm that uniforms the edges is a tuple $(\mathcal{M}, m, k, n)$, where $\mathcal{M}$ is the triangular mesh, m is the distance between the vertices (called here as desired or target edge length), k the number of rings used at the Laplacian optimization step and n the number of iterations.

The method works as shown in Algorithm 1. Each step is separately explained ahead.

$\mathcal{M}' = \texttt{Copy}(\mathcal{M})$
for $i = 1$ **to** n **do**
  $m_i = \texttt{MIN}(2 \cdot \texttt{CalculateEdgesAverage}(\mathcal{M}'), m)$
  $L_p = \texttt{CreatePriorityList}(\mathcal{M}', m_i)$
  $\texttt{StellarOperations}(\mathcal{M}', L_p)$
  $\texttt{CorrectValency}(\mathcal{M}')$
  $\texttt{GlobalRelaxation}(\mathcal{M}', k)$
  $\texttt{Projection}(\mathcal{M}, \mathcal{M}')$
end
return $\mathcal{M}'$

Algorithm 1: UniformRemeshing($\mathcal{M}, m, k, n$)

2.1. Stellar Transformations with Priority List

The target edge length m is arbitrary and the number of elements present in the mesh must change according to it. The higher m is, the less faces and vertices are needed to represent the model. For the opposite, a lower m results in more faces and vertices. So the current edge's length is the criterion to decide if vertices are added or removed. An arbitrary mesh, however, can have regions which should be refined and others that should be simplified.

This step is based on the modification of *long* or *short* edges a_j which are classified as:

$$\begin{array}{rl} \text{long}, & \text{if } \left|a_j\right| > m_i + \sigma \\ \text{short}, & \text{if } \left|a_j\right| < m_i - \sigma \end{array} ,$$

where m_i is an intermediate target value for the i-th iteration, defined in such a way that the transition between the current average edge length of the model on steps i and $i + 1$ is smooth.

Edges with length within the closed interval $[m_i - \sigma,\ m_i + \sigma]$ remain unchanged during this step. Long and short edges are then candidates to be collapsed or splited, respectively. Despite the σ value can be a parameter of the method, our experiments show that $\sigma = 0.2 \cdot m_i$ generally gives good results. For completude reasons, however, some results with different σ values are presented. For an edge $a_l \in \mathcal{A} \subset \mathcal{M}'$ considered long, a new vertex is inserted at its middle position dividing it into two edges of length $|a_l|/2$ by the stellar operation of *edge split*. Edges considered too small in respect to m_i are removed from the model using *edge collapse*.

priorityList
foreach $a_i \in \mathcal{A} \subset \mathcal{M}'$ **do**
  if $|a_i| > m_i + \sigma$ **then**
    priorityList.$\texttt{Add}(a_i)$
  else if $|a_i| < m_i - \sigma$ **then**
    priorityList.$\texttt{Add}(a_i)$
  end
end
$\texttt{SortDescent}$(priorityList)
return priorityList

Algorithm 2: CreatePriorityList($\mathcal{M}', m_i, \sigma$)

When m is much greater than the current edges average, a strong mesh simplification is required. In a extreme example with high m, all edges will be candidates to be collapsed. Thus, the iteration's target m_i is set to be the minimum between the global target average m and two times the current edges average of $\mathcal{M}'_i$ (Algorithm 1). In that way, the difference between the averages of $\mathcal{M}'_i$ and $\mathcal{M}'_{i+1}$ changes smoothly and is unlikely to generate invalid triangles or change the topology of the mesh, due to the collapse of near long edges. The upper bound of two times the current average per iteration for m_i comes from the fact that the edges progressively tend to be equalized, and doubling the edges length of a star does not promote a severe local modification.

The order of application of stellar operations is important. We propose to process the longest and smallest edges first. A priority list L_p is created where the edges a_j, with higher deviation $d_j = \left|a_j - m_i\right|$, are the most important. The list L_p is the set of edges $\{a_1, ..., a_t\} \subset \mathcal{A} \subset \mathcal{M}$, with $d_1 \geq d_2, ..., d_{t-1} \geq d_t$, where $a_j \in \{a_1, \cdots, a_t\}$ if $\left|a_j\right| \notin [m_i - \sigma, m_i + \sigma] \in \mathbb{R}$ (Algorithm 2).

The stellar operations are performed after the set up of the list. The algorithm traverses the list verifying the type of each edge and applies the appropriate operation (*edge split* or *edge collapse*). All the vertices adjacent to an edge that was changed are marked as visited and removed from L_p. This avoids that a single vertex has more than one modified edge, which gives a better distribution of modified edges per iteration (Algorithm 3).

```
foreach a_i ∈ L_p do
    if VertexVisited(a_i) then
        continue
    else if |a_i| > m_i + σ then
        EdgeSplit(a_i)
        VisitEdgeVertices(a_i)
    end
    else if |a_i| < m_i − σ then
        EdgeCollapse(a_i)
        VisitEdgeVertices(a_i)
    end
end
```

Algorithm 3: StellarOperations($\mathcal{M}', L_p$)

2.2. Post Processing

After the edge adjustments by stellar operations, the valency of the vertices naturally tends to 6, which is mandatory to obtain uniform distribution in smooth regions. However, some vertices might have arbitrary valencies. A simple scheme to correct the vertex valencies based on *edge flip* operations is then applied [13, 5, 10].

Consider the two triangles of Figure 1. For each edge of the mesh, the algorithm performs an *edge flip* operation and, if the valencies of the involved vertices become near 6, the rotated edge is accepted, otherwise the operation is undone. An overview of the process is shown on Algorithm 4. The evolution of the vertices valencies over the iterations are presented in Section 3.

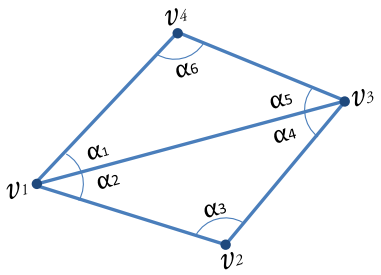

Fig. 1. Vertices v_1, v_2, v_3 e v_4 forming two triangles $f_1 = (v_1, v_2, v_3)$ e $f_2 = (v_1, v_3, v_4)$. The adjacent vertices to the edge $a_i = \overline{v_1 v_3}$ are v_1 and v_3, and the opposite vertices to a_i are v_2 and v_4. The set of vertices of the adjacent triangles to a_i is $\{v_1, v_2, v_3, v_4\}$.

2.3. Global Optimization with k-neighbourhood

After the elimination of the short and long edges and the connectivity correction of the vertices, they must be globally redistributed. The relocation of the elements should approximate as much as possible the original geometry.

foreach $a_i \in \mathcal{A} \subset M'$ **do**
  v_n = `SearchAdjacentVertices(`a_i`)`
  preDeviation = $|valency(v_1) - 6| + |valency(v_2) - 6| + |valency(v_3) - 6| + |valency(v_4) - 6|$
  `EdgeFlip(`a_i`)`
  postDeviation = $|valency(v_1) - 6| + |valency(v_2) - 6| + |valency(v_3) - 6| + |valency(v_4) - 6|$
  if preDeviation $\leq$ postDeviation **then**
    `EdgeFlip(`a_i`)`
  end
end

Algorithm 4: CorrectValency($\mathcal{M}'$)

The uniform Laplacian operator, which uses only the 1-star of the vertex, requires more iterations to obtain equalized meshes. We propose to use more than one vertex ring to improve the convergence rate. Instead of using only the nearest neighbours, the contribution of all vertices until the k-th neighbour are considered. Therefore, the vertex v_i tends more quickly to the central position of the set of the surrounding vertices. It also leads to a smoother result. The closer a neighbour is, higher is its influence to displace v_i. Thus, the weight of the influence of a vertex v_j over v_i is given by $w_{ij} = \frac{1}{S_j}$, where S_j is the star number of v_j related to v_i or, in other words, the number of hops from v_i to reach v_j.

The energy formulation for each vertex is then:

$$E_i = \sum_{j \in star_k(v_i)} w_{ij} \left\| v_i + \mathbf{t}_i - v_j - \mathbf{t}_j \right\|^2, \tag{1}$$

where $\mathbf{t}_i$ and $\mathbf{t}_j$ are, respectively, the unknown displacements for v_i and v_j, and w_{ij}. This is quadratic in each vertex, so its partial derivatives are linear expressions. The minimum is unique and can be found when the derivative with respect to the vertices are zero, resulting in a sparse linear system. The vertices' positions are known and the unknown quantities are the tangent vectors to each point, leading to the system

$$\mathbf{AX} = \mathbf{b} \quad \mapsto \quad \left(\frac{\partial E}{\partial v}\right)\mathbf{X} = 0. \tag{2}$$

The energy E_i (Eq. 1) can be written as

$$\begin{aligned} E_i &= \sum_{j \in star_k(v_i)} w_{ij} \left\| v_i + \mathbf{t}_i - v_j - \mathbf{t}_j \right\|^2 \\ &= \frac{1}{S_j} < v_i + \mathbf{t}_i - v_j - \mathbf{t}_j >< v_i + \mathbf{t}_i - v_j - \mathbf{t}_j > \\ &= \frac{1}{S_j}(< v_i, v_i > + < v_j, v_j > + < \mathbf{t}_i, \mathbf{t}_i > + < \mathbf{t}_j, \mathbf{t}_j > + \\ &2\Big(< v_i, \mathbf{t}_i > + < v_j, \mathbf{t}_j > - < v_i, v_j > - < v_i, \mathbf{t}_j > - < v_j, \mathbf{t}_i >\Big)), \end{aligned}$$

and the derivative with respect to v_i:

$$\frac{\partial E_i}{\partial v_i} = \mathcal{L}(v_i) = 2 \sum_{j \in star_k(v_i)} \frac{1}{S_j}(v_i + \mathbf{t}_i - v_j - \mathbf{t}_j) \Rightarrow \sum_{j \in star_k(v_i)} \frac{1}{S_j}(v_i + \mathbf{t}_i - v_j - \mathbf{t}_j) = 0. \tag{3}$$

We impose restrictions $\mathcal{T}_i$ on the system in order to force $\mathbf{t}_i$ to preferably preserve the original geometry of the mesh. Given a vertex v_i, we include the restriction $< \mathbf{t}_i, \mathbf{n}_{v_i} >= 0$, where $\mathbf{n}_{v_i}$ is the normal of the tangent planes (in respect the original surface) estimated over v_i, respectively. Feature lines and corners could be treated differently to preserve even more the geometry.

Let $\mathcal{V} \subset \mathcal{M}', |\mathcal{V}| = r$. The linear system with the geometric restrictions above can be written as a matrix:

$$\mathbf{AX} = \mathbf{b} \quad \mapsto \quad \begin{pmatrix} \mathcal{L} \\ \mathcal{T} \end{pmatrix} \mathbf{X} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{4}$$

where $\mathcal{L}$ is the modified Laplacian matrix of dimension $r \times r$, using the k-neighbourhood, and $\mathcal{T}$ is a $p \times r$ matrix composed by the restrictions. The dimension $p \geq 3$ depends on the number of corners and feature lines of the model. Note that the restrictions can work in an unique solution or in a least-squares sense. Thus, the resulting displacements $\mathbf{t}_i$ can be hardly constrained or not by the geometric restrictions.

2.4. Projection onto the original mesh

To ensure that every position found is over the input mesh, a step that projects the vertices of $\mathcal{M}'_i$ of each iteration over the original model $\mathcal{M}$ was included. The subproblem of finding the nearest points [13] between two sets is a high complexity computational problem. After the global relaxation, there is a set of vertices $\{v_1, ..., v_n\} \subset \mathcal{M}'_i$ whose original counterparts $\left\{v'_1, ..., v'_n\right\} \subset \mathcal{M}$ must be found. The representation $\mathcal{M}$ describes a continuous surface and, as a result, there are infinite solutions to be tested in order to obtain $min(\left\|v_i - v'_i\right\|)$ for all i. We use a simple approach that finds a v'_i candidate, not necessarily optimal, for each v_i.

The algorithm projects the vertices over the components (vertices, edges and faces) of $\mathcal{M}$ closest to v_i, generating a set of points over the surface. The element that shows the shortest distance to v_i will be the final vertex v'_i. Note that the projection does not represent, necessarily, the exact solution of the problem. However, the results of this procedure have shown satisfactory results, regardless of its simplicity.

Of course, there are several situations that may result in an invalid mesh. For example, the combination of a high edge target m can give wrong results in regions with high negative curvature. However, by setting a small number of iterations and m increasing softly, challenging meshes can be successfully equalized.

3. Results

In this section the results and the applications of the proposed method are shown for different scenarios. It is also discussed interpretations for the presented data. The algorithm was implemented using the programming language C++ on Linux (Ubuntu v11.04). The tests were run on an Intel Core i5 (2410M) computer, with 6GB of RAM and a GTM540M video card.

The first study was the state of the processed models in distinct time steps. In Figure 2, the mesh progression over each iteration is shown. One may easily notice the improvement in the distribution and regularity of vertices. After thirty iterations, the final processed mesh presents a more uniform vertex distribution, as depicted numerically in Table 1, where $\bar{x}$ is the average and s the standard deviation.

Iteration	$\bar{x}$	s	Num. Vert.	Reg. Vert.(%)
$n_i = 0$	0.6019	0.2468	9002	36.6%
$n_i = 5$	0.5008	0.0546	10404	77.8%
$n_i = 10$	0.5076	0.0454	9924	82.4%
$n_i = 20$	0.5108	0.0397	9578	86.1%
$n_i = 30$	0.5124	0.0380	9394	86.8%

Table 1. State of the grid in different iterations with parameters $m = 0.5$ and $k = 2$.

The bunny approximation reveals that the vertices distribution is gradually corrected and connectivity regularized as the number of iterations increases (Figure 2). The histograms at the lower left corners represent the amount of vertices with valency 4, 5, 6, 7 and 8 for i-th iteration. It is possible to notice that this number tends to 6 as the algorithms advances, as mentioned in Section 2.

The parameter k plays an interesting role in the algorithm. The use of $k > 1$ was effective in most tests, generating better results than the most recurrent formulation found in the literature ($k = 1$, uniform Laplacian). It was found that the use of $k = 2$ is sufficient to obtain satisfactory results in most cases. However, in some scenarios, a higher k is necessary to improve the edge length equalization, as shown in Table 2.

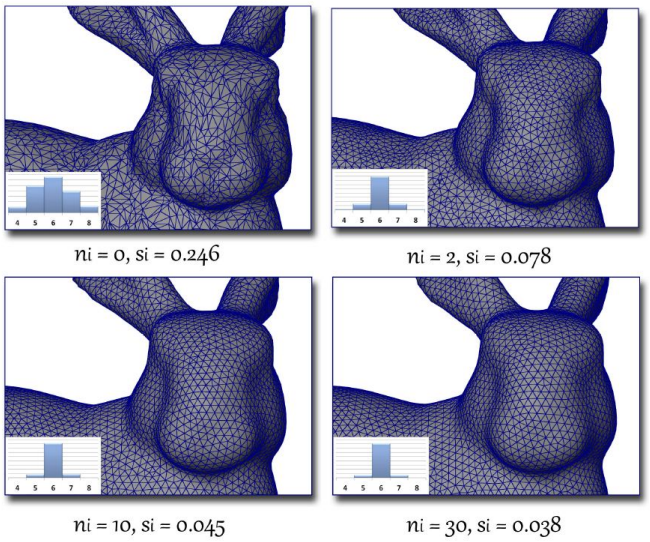

Fig. 2. Bunny head highlight. The graphics on the lower left corner show the regularity distribution for each n_i.

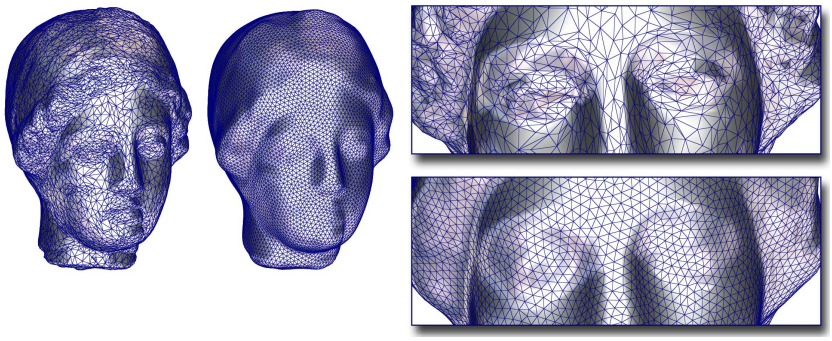

Fig. 3. *Egea* model after remeshing (bottom chart): parameters $m = 0.8, n = 20$ and $k = 4$.

Table 2 presents the final data from experiments with *Egea* model (Figure 3). It is possible to verify that, when $k > 1$, the standard deviation reaches lower values. After several experiments, we realized that σ with values between 20% and 25% of the target average m give the best results. We also observed that as σ increases, it is important that k also increase to obtain better results.

Figure 4 presents the surface *rockerarm* processed with different edge targets. The details of the model (regions of high curvature) are lost as m increases. This is expected when a strong reduction of the number of vertices is desired. Uniform meshes are not ideal for representation of salient areas, since the edge length constraint causes significant loss of information. The iterative nature of the proposed algorithm contributes negatively to this behavior because the relaxation process, which is a low pass filter, is consecutively applied.

In a oversampling process, the number of elements of a mesh may be substantially increased, depending on m. If the input m is lower then the original model edge length average, the presented method will naturally increment the amount of faces, vertices and edges (Fig. 5). One may notice in this figure a fairly conservation of the original surface geometry, in adition to the vertices uniform distribution.

3.1. Applications

One of the main applications for uniform meshes are computational simulations. The generation of hexagonal uniform meshes used as input for physics simulations of carbon nano-structures is one of the motivations of this

(m, σ, n, k)	$\bar{x}$	s	Num. Vert.	Reg. Vert. (%)
Model	0.599984	0.341534	10044	37.9%
(0.8, 0.1, 20, 1)	0.775103	0.135009	10648	**68.2%**
(0.8, 0.1, 20, 2)	0.785205	**0.126693**	10531	67.7%
(0.8, 0.1, 20, 3)	0.793493	0.129107	10450	66.6%
(0.8, 0.1, 20, 4)	**0.799652**	0.130791	10380	66.2%
(0.8, 0.2, 20, 1)	0.792564	0.076465	10447	80.1%
(0.8, 0.2, 20, 2)	**0.801297**	0.073887	10282	81.7%
(0.8, 0.2, 20, 3)	0.814370	**0.071990**	9921	84.0%
(0.8, 0.2, 20, 4)	0.825473	0.073180	9617	**84.8%**
(0.8, 0.35, 20, 1)	0.772455	0.128458	10906	**76.6%**
(0.8, 0.35, 20, 2)	0.781785	0.127766	10836	73.1%
(0.8, 0.35, 20, 3)	0.789769	0.127981	10789	71.6%
(0.8, 0.35, 20, 4)	**0.796638**	**0.127747**	10690	72.5%
(0.8, 0.41, 20, 1)	0.769633	**0.152831**	10910	**74.2%**
(0.8, 0.41, 20, 2)	0.781499	0.156026	10827	70.0%
(0.8, 0.41, 20, 3)	0.791372	0.156487	10799	69.0%
(0.8, 0.41, 20, 4)	**0.794345**	0.156241	10850	69.1%

Table 2. Remeshing values for *Egea* with different k and σ configurations. (Fig. 3).

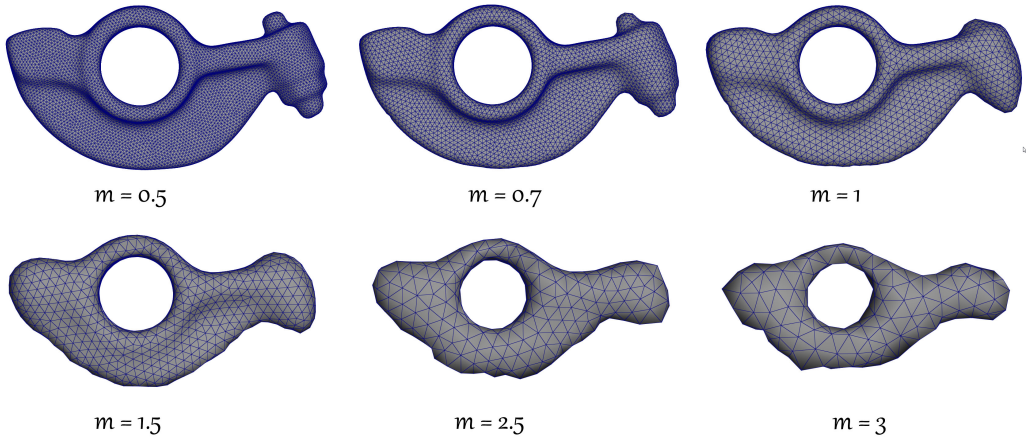

Fig. 4. *Rockerarm* model processed for different edge lenghts. Each picture corresponds to an mesh generated with specific m (shown in figure). Parameters: $\sigma = 0.2 \cdot m, n = 30$ e $k = 2$.

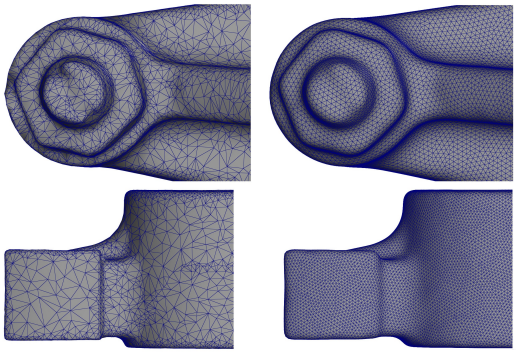

Fig. 5. Oversampling of the *Rockerarm* surface.

work. Those structures can be generated from arbitrary triangular meshes, as in Nieser *et al.* [1] or from parametric spaces. The dual space approach of the input triangular mesh is used in this paper to create hexagonal objects.

Note that each vertex in the dual version $\mathcal{D}$ of the final mesh $\mathcal{M}'$ will have exactly three incident edges, which characterizes the generated mesh as regular and trivalent. The centroid of the polygons at the opposite space is nearly each primal vertex. One may also observe that, if the primal mesh is not regular, $\mathcal{D}$ will be composed by several non hexagonal polygons.

The mesh edge length in primal and dual spaces of 2-manifolds are strongly related in terms of distribution. Triangles over a smooth surface tend to have equal areas if their edges are uniform. The distance between the centroids of neighbor triangles also tend to be uniform. Thus, the edge length for the trivalent mesh also tend to be uniform. Therefore, the problem of generating uniform trivalent meshes can be reduced to the equalization of triangular meshes. Given an uniform triangular mesh, the polygons and vertices in its dual domain tend to be uniformly distributed (Fig. 6 and 7).

For modeling carbon nano structures, for example, there is a tendency for the atoms to keep a mutual constant distance. A primal triangular object must have uniform edge lengths to have its dual directly used as input in this kind of simulation problem.

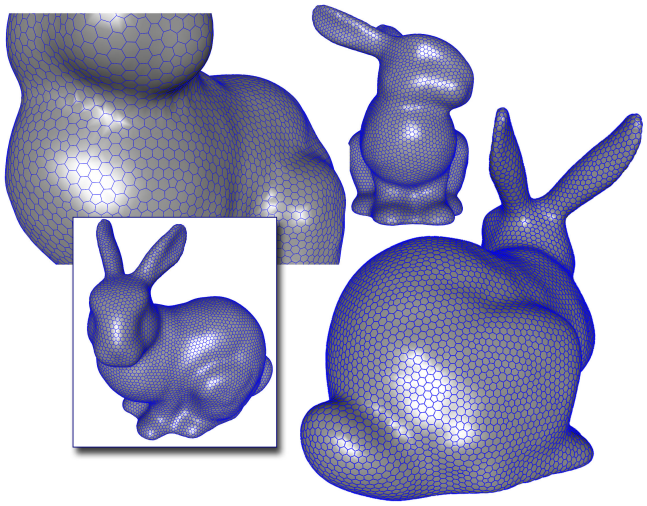

Fig. 6. Dual mesh of the *bunny* model after the uniform remeshing with $m = 0.5, n = 20$ e $k = 2$.

The dual triangular mesh of the *bunny* model is shown on Figure 6. The equalized version was generated with parameters $(m, k, n) = (0.5, 2, 20)$ followed by the calculation of the correspondent trivalent mesh. A fairly uniform distribution of the vertices can be observed.

To calculate the interaction between atoms, there are simple formulations as the Lennard-Jones potential [14] and more complex methods as used in the REBO potential [15, 16]. The energy distribution over the trivalent mesh calculated from the dual of the *bunny* model before and after the uniform remeshing can be seen in Figure 7. The colours vary from blue to red as the energy goes from low to high, respectively. It is possible to infer that the generated surfaces are more stable to use in molecular dynamics simulation than their original models.

4. Conclusion

This work presents an iterative remeshing method for obtaining models having edges with a target average m and low standard-deviation. The main contribution is an effective method for edge equalization. It is reasonably fast for modeling purposes and admits target edge lengths which are far from the original average. We propose a new formulation for mesh relaxation based on the Laplacian transformation with k-neighbourhood, which includes constraints to preserve the original geometry. We also propose stellar transformations based on a priority list to avoid local geometric inconsistencies.

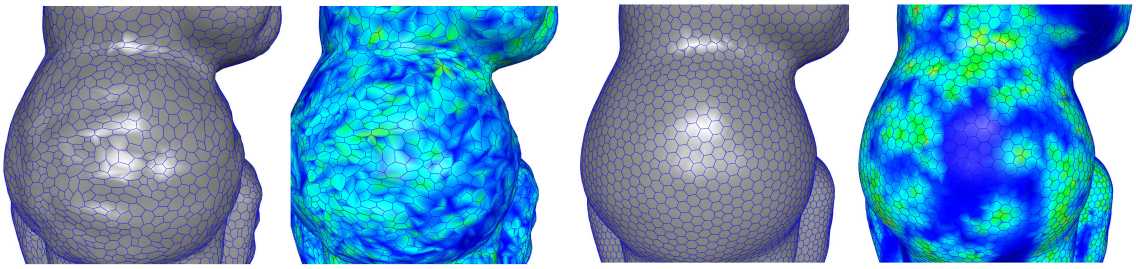

Fig. 7. Energy distribution of the atoms in a molecular system with the dual mesh of the *bunny* model before (right) and after (left) remeshing.

The number of iterations n, target length m and the number of neighbours k form the critical parameters that are intuitive and easy to define, even for equalizing challenging input models. Higher n is, more likely the resulting edges will have average m by the end of the process. But if n is too high, the geometry is more likely to be softned by the relaxation step. Higher the k is, the sequence will converge faster to the average m, but with a higher smoothing effect. Our method also admits the adjustment of the deviation σ, mainly used in the stellar operations step. Experimental results show that σ around 20% and 25% of the target m works well for all models used.

As our results show, the equalized meshes fairly represent the original surface, even if the target edge length is far from the initial average. The final result can be useful for some engineering and physics applications, such as nano structure simulation.

The mesh projection step is the least robust step in the overall processing. However, by setting properly the number of iterations and varying softly m, even challenging models with high curvature regions and arbitrary topology can be succesfully equalized. As a future work, a new approach to avoid that the iterated meshes are distorted with respect to the original model could be proposed.

Acknowledgements

Authors thank to Fundação de Amparo à Pesquisa do Estado de Minas Gerais and CAPES for financial support.

References

[1] M. Nieser, J. Palacios, K. Polthier, E. Zhang, Hexagonal global parameterization of arbitrary surfaces, in: ACM SIGGRAPH ASIA 2010 Sketches, SA '10, ACM, New York, NY, USA, 2010, pp. 5:1–5:2.
[2] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Mesh optimization, in: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, SIGGRAPH '93, ACM, New York, NY, USA, 1993, pp. 19–26.
[3] P. Alliez, E. C. d. Verdière, O. Devillers, M. Isenburg, Isotropic surface remeshing, in: Proceedings of the Shape Modeling International 2003, SMI '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 49–.
[4] D. ming Yan, B. Lévy, Y. Liu, F. Sun, W. ping Wang, Isotropic remeshing with fast and exact computation of restricted vor onoi diagram, in: ACM/EG Symposium on Geometry Processing / Computer Graphics Forum, 2009.
[5] V. Surazhsky, C. Gotsman, Explicit surface remeshing, in: Proceedings of Eurographics Symposium on Geometry Processing, Aachen, Germany, 2003, pp. 17–28.
[6] Y. Li, E. Zhang, Y. Kobayashi, P. Wonka, Editing operations for irregular vertices in triangle meshes, in: ACM SIGGRAPH Asia 2010 papers, SIGGRAPH ASIA '10, ACM, New York, NY, USA, 2010, pp. 153:1–153:12.
[7] D. Bommes, H. Zimmer, L. Kobbelt, Mixed-integer quadrangulation, ACM Trans. Graph. 28 (3) (2009) 77:1–77:10.
[8] P. Alliez, M. Meyer, M. Desbrun, Interactive geometry remeshing, ACM Trans. Graph. 21 (3) (2002) 347–354.
[9] N. Ray, W. C. Li, B. Lévy, A. Sheffer, P. Alliez, Periodic global parameterization, ACM Trans. Graph. 25 (4) (2006) 1460–1485.
[10] M. Botsch, L. Kobbelt, A remeshing approach to multiresolution modeling, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, SGP '04, ACM, New York, NY, USA, 2004, pp. 185–192.
[11] B. Lévy, Constrained Texture Mapping, in: SIGGRAPH 2001, ACM, Addison Wesley, Los Angeles, USA, 2001, p. 8 p., colloque avec actes et comité de lecture. internationale. A01-R-028 —— levy01a A01-R-028 —— levy01a.
[12] N. Pietroni, M. Tarini, P. Cignoni, Almost isometric mesh parameterization through abstract domains, IEEE Transaction on Visualization and Computer Graphics 16 (4) (2010) 621–635.
[13] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. uno Levy, Polygon Mesh Processing, AK Peters, 2010.
[14] J. E. Lennard-Jones, Cohesion, Proceedings of the Physical Society 43 (5) (1931) 461–482.
[15] D. W. Brenner, O. A. Shenderova, J. A. Harrison, S. J. Stuart, B. Ni, S. B. Sinnott, A second-generation reactive empirical bond order (REBO) potential energy expression for hydrocarbons, Journal of Physics: Condensed Matter 14 (4) (2002) 783.
[16] Z. Wang, Reactive empirical bond-order (REBO) potential (01 2006).