Iterative remeshing for edge length interval constraining

João Vitor de Sá Hauck[1], Ramon Nogueira da Silva[1], Marcelo Bernardes Vieira[1], and Rodrigo Luis de Souza da Silva[1]

Universidade Federal de Juiz de Fora, Departamento de Ciência da Computação, 36036-900, Juiz de Fora-MG, Brazil
jhauck@ice.ufjf.br,ramon.nogueira@ice.ufjf.br,
marcelo.bernardes@ice.ufjf.br, rodrigoluis@ice.ufjf.br

Abstract. This paper presents an iterative method to remesh an arbitrary surface into a mesh with all edge lengths within an interval. The process starts with a triangular 2-manifold mesh. It uses stellar operations to achieve the necessary amount of vertices and triangles. Subsequently, it applies a constrained version of the Laplacian filter in order achieve a more uniform distribution of the vertices over the surface. In order to prevent the natural shrink caused by the Laplacian filter, we perform a projection over the original surface. We also apply a post processing step to correct the lengths of troubling edges. Our method results in a regular mesh, with vertices uniformly distributed. The dual mesh obtained can be useful for several applications. The main contribution of this work is a new approach for edge length equalization, with explicit constraints definition, lower global geometry losses and lower memory cost if compared to previous works.

Keywords: iterative remeshing, edge length equalization, interval constraining

1 Introduction

Computational models of real objects are currently used for several applications. The growing need of geometric models conducted to the development of many technologies for mesh generation, such as computer vision algorithms with 3D scanners [1, 2] or direct modeling softwares [3, 4]. However, these technologies not always lead to an optimal mesh representation for a specific application. Therefore, the improvement of the quality of these representations became a primal research area in computer graphics.

The precise quality criteria for an arbitrary mesh depends on its usage. For real time applications, for instance, it is usually required a simplification of the model, in order to achieve high performance. In physics and chemistry simulations [5, 6], some constraints may be necessary to guarantee the fidelity of the results, e.g. constraints on edge lengths, valid vertices valency, proper distributions of vertices, etc. Bommes *et al* [7] enumerate the quality aspects most commonly required.

This work is interested in the regularization of the edge lengths of a triangular 2-manifold mesh. Specifically, our goal is to impose a constraining interval for the lengths. So, we iteratively remesh the model until all the edge lengths satisfy the defined constraints. Since the average of the edge lengths in a region impact on the amount of the edges and faces found there, the method applies stellar operations to adjust the amount of edges locally. It also applies an approximation of the Laplacian filter to relax the mesh on each iteration. Then, it projects the vertices over the original surface, in order to preserve the geometry. Finally, after the execution of the iterations, it performs a post processing step that eliminates the remaining problems. Although the process is designed to maintain the original geometry of the model, some local geometry losses may occur, specially in regions with high curvature.

As our results indicate, the method generates models which satisfy the input constraining for most cases. Moreover, in the final mesh the standard-deviation of edge lengths tends to be low. The resulting mesh can also be used to generate a very regular trivalent mesh, by computing its dual. This kind of mesh can be greatly useful for engineering and physics applications, such as nano carbon simulations, which firstly motivated this work.

2 Related works

There are many remeshing processes focused on obtaining more regular meshes for applications. N-Symmetric fields [8] can be used for generating highly regular polygonal meshes. The work of [7] applies the method for remeshing an arbitrary triangular mesh into a high quality quadrangular mesh. The method is computationally expensive, since the formalism proposed by [8] results in a mixed-integer system, which is a well known NP-Hard problem.

Using a similar approach, Huang J. *et al* [9] aims to obtain a mesh where the angle between two arbitrary edges of a triangle is 60^o. Computing the N-Symmetric field for this technique requires a set of feature lines of the model, which can be either automatically estimated or defined by the user. That estimation has a high computational cost.

The work presented in [10, 11] proposes a method to obtain a regular trivalent mesh. At first it estimates a quadrangular mesh, as proposed by [7]. The following step computes the rhomboid mapping of that quadrangular mesh, resulting in the desired trivalent mesh. Although is not the main goal of that work, a very regular triangular mesh could be obtained by computing the dual from that resulting trivalent mesh.

Pietroni *et al* [12] use a global parametrization to obtain an almost isometric mesh. This method leads to a very regular triangular mesh. Their results are really competitive as showed in the results, however the global parametrization is complex and computationally expensive, since it requires a set of feature lines of the model. As well, they are not explicitly concerned with edges lengths.

In the approach of [13], a regular triangulation is not the final objective, but a necessary step for achieving a self-supporting surface, i.e., a surface that stands

in static equilibrium without external support. The method used for obtaining the regular triangulation is the power diagram [14].

The method proposed by [15] processes a mesh to obtain a more regular version. The first step is to apply the edge split operation in edges that are longer than $\frac{4}{3}l$ and edge collapse in edges that are shorter than $\frac{4}{5}l$. Next, they perform edges flips to correct the valency. Finally, they equalize the triangle area with an area based tangential vertex smoothing. This work is not focused in edge length so it did not constrain the edge length into an interval.

Surazhsky *et al* [16] proposed a remeshing method based on area equalization and angle smoothing. Their method aims a mesh with triangle areas almost uniform and maximizes the minimum angle of the triangle. This interesting approach build a high regular surface. They also propose a new method to smooth the surface based on angles. However, this method does not remove elements and is not suitable for simplifications.

For the problem of edge length equalization [17], the goal is to obtain an average edge length that is close to a user defined value, with low standard deviation. Our work uses a similar approach. That method solves a large linear system to apply the Laplacian filter. In our work, that process is no longer necessary due to a new explicit approximation. This allows the method to process larger models with the same memory cost.

3 Proposed method

Our method is an extension of the work of [17]. However, while [17] aims to obtain a mesh where the average edge length is as close as possible to a target value, our goal is to generate a mesh without any *long* or *short* edges a_j, classified as:

$$\begin{matrix} \text{long, if } |a_j| > e_{max} \\ \text{short, if } |a_j| < e_{min} \end{matrix},$$

where e_{min} is the shortest edge length allowed and e_{max} is the longest edge length allowed.

The input for this algorithm is a tuple $(\mathcal{M}, e_{min}, e_{max}, n, k, p)$, where $\mathcal{M}$ is the triangular mesh, n the number of iterations, k the number of rings used at the Laplacian optimization step and p is the number of iterations before the original mesh is replaced by the current mesh in order to relax next projections. At the end of each iteration, we save the resulting mesh if it has a lower amount of *long* and *short* edges than the current saved mesh. The Algorithm 1 is an overview of the proposed method.

Detailed information about the *CorrectValency* and *Projection* procedures can be found in [17]. The other steps will be explained ahead.

3.1 Stellar operations

The amount of edges necessary to achieve the constraining interval is directly related to the average length m of the interval. Thus, in this step we modify

$\mathcal{M}' = \texttt{Copy}(\mathcal{M})$
$m = \frac{e_{min}+e_{max}}{2}$
for $i = 1$ **to** n **do**
  if $p > 0$ *and (i mod p) = 0* **then**
    $\mathcal{M} = \texttt{Copy}(\mathcal{M}')$
  end
  `StellarOperations`($\mathcal{M}'$)
  `CorrectValency`($\mathcal{M}'$)
  `LowPassFiltering`($\mathcal{M}'$, k)
  `Projection`($\mathcal{M}$, $\mathcal{M}'$)
end
`PostProcess`($\mathcal{M}'$)
return $\mathcal{M}'$

Algorithm 1: UniformRemeshing($\mathcal{M}, e_{min}, e_{max}, n, k, p$)

the number of edges in the model to reach a feasible amount. However, if m is much greater than the current average edge length, a strong simplification of the mesh is required. In order to prevent the degeneration of the mesh, we calculate intermediate values for e_{min} and e_{max} that only allow smooth transformations. These values are defined as:

$$\begin{aligned} ei_{min} &= MIN(2 \cdot m_i, m) - \tfrac{e_{max}-e_{min}}{2} \\ ei_{max} &= MIN(2 \cdot m_i, m) + \tfrac{e_{max}-e_{min}}{2} \end{aligned},$$

where m_i is the current average edge length of the model.

The order of appliance of the stellar operations is important. Therefore, we create a priority list of edges, as presented in [17]. Once the list is set, the algorithm traverses it processing each edge.

If the edge length is shorter than ei_{min}, it is collapsed. Otherwise, if the edge length is longer than ei_{max}, then it is split. This modifies the amount of edges locally, since in an arbitrary mesh some regions need to be refined while others need to be simplified. If the edge is neither shorter or longer nothing is done. This may occur when one of the vertices were modified by another stellar operation. After an edge is processed, all vertices affected are marked as processed. When an edge has its two vertices marked, it is removed from the list.

Originally, both the remaining vertex of the edge collapse operation and the new vertex of the edge split operation can be placed in an arbitrary position over the operated edge. Hence, in order to optimize the convergence of our algorithm we compute the position that minimizes the equation:

$$\sum_{V_j} (|V_i - V_j| - m)^2,$$

where V_i is the vertex we want to position and V_j the vertices connected to V_i.

3.2 Low-pass filter

To achieve equalized edge lengths, we globally distribute the vertices uniformly over the surface. To do so, after the valency correction step, as described in [17], we proceed to the low-pass filtering. In this work, we use a modified version of the Laplacian filter.

The classic Laplacian filter is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x_1} + ... + \frac{\partial^2 f}{\partial^2 x_n}.$$

It is a measurement of the dispersion in $\mathbb{R}^n$ of a function f. Taubin [18] propose a discrete approach to the Laplacian operator. The approach is:

$$L(V_i) = \sum_{V_j} w_{ij}(V_i - V_j),$$

with V_j in the neighborhood of V_i. In the literature, many weights were proposed for w_{ij}. There are schemes based on cotangent [19] and neighborhood [17].

The discrete Laplacian is largely used due to its simplicity. Basically, its appliance moves each vertex to the average of its neighbors. This procedure tends to equalize edge lengths, minimizing the standard deviation. The Laplacian must be zero to achieve these properties and the system to be solved is given by:

$$\sum_{v_j} w_{ij}(v_i - v_j) = 0.$$

The technique employed in this work does not solve the system. Instead, we run an iterative approximation that gives us almost the same results, significantly reducing the memory cost.

In the classical Laplacian filter, we add some additional constraints to reduce the geometry loss:

$$N_i \cdot D_i = 0,\ \forall D_i \in \mathcal{M}',$$
$$|D_i| = 0\ \forall D_i \in \mathcal{B},$$

where N_i is the normal of the current mesh in the vertex V_i and D_i is the unknown displacement of the vertex V_i. These constraints were imposed in the approximation in a extremely restrict way. It means that all restrictions were exactly matched.

The iterative Algorithm 2 approximates the constrained Laplacian filtering described above. It calculates the new vertex position based on the k-neighborhood as proposed in [17]. The first step is to compute for each vertex the new position without the application of the new constraints. This position is defined by the center of mass of all neighbors vertices weighted by their ring number in such a way that distant vertices contribute less than near vertices.

The second step is to impose the constraint $N_i \cdot D_i = 0$ by removing the vector projection of D_i in N_i. When all displacements are computed, the vertices V_i are updated, except those on the borders.

foreach $V_i \in \mathcal{M}'$ **do**
  $kStar$=`getKStar` (V_i,k)
  fat=0
  foreach $V_j \in kStar$ **do**
    V_i'=V_i'+$\frac{V_j}{star}$
    fat=fat+$\frac{1}{star}$
  end
  V_i'=$\frac{V_i'}{fat}$
  D_i=V_i'-V_i
  D_i=D_i `-projection` (D_i,N_i)
end
foreach $V_i \in \mathcal{M}'$ **do**
  if $V_i \notin \mathcal{B}$ **then**
    V_i=V_i+D_i
  end
end

Algorithm 2: LowPassFiltering($\mathcal{M}'$, k)

3.3 Post processing

After n iterations, we proceed to the last step of the algorithm. This step is run over the saved mesh with best results from the previous iterations. It distributes the vertices locally, only in the problematic regions.

First, we propose an error measurement for a region around an edge:

$$\sum_{V_i} \sum_{V_j} (|V_i - V_j|^2 - m^2)^2, \tag{1}$$

where V_i are the vertices in the forth star of the edge and V_j the vertices in the first star of V_i.

If we want to approximate the edge lengths to m, it is enough to minimize the Equation 1. Nonetheless, it may greatly modify the local geometry, once there are no constraints for the vertices displacement. Thence, we restrict those displacements to the tangent plane. For each vertex V_i, we obtain an orthonormal base with its normal vector.

This local base is $< N_i, T_{i1}, T_{i2} >$, where N_i is the normal direction, and T_{i1}, T_{i2} are the directions over the tangent plane. Using this local base we impose a restriction in the Equation 1, and the final error measurement to minimize is:

$$\sum_{V_i} \sum_{V_j} (|V_i + \alpha_i \cdot T_{i1} + \beta_i \cdot T_{i2} - (V_j + \alpha_j \cdot T_{j1} + \beta_j \cdot T_{j2})|^2 - m^2)^2, \tag{2}$$

where α_i and β_i are the variables in the function and represent the displacement over the tangent plane. If the index i does not exist both α_i and β_i are set zero.

In this work we use a conjugate gradient method [20] for minimizing Equation 2. Due to performance and numerical problems, we do not minimize it for all the vertices in the model. Instead, we process the model per edge.

As we expect to minimize the geometry losses, we seek to apply the transformations firstly in the regions with the greatest amount of *long* or *short* edges. To do so, we first create a new priority list of edges. For this list, the priority assigned to each edge is proportional to the number of *long* and *short* edges in its neighborhood. Next, we perform the minimization of Equation 2 for each edge on the list, in order.

In the cases in which even this technique does not solve the problem for all the edges, we perform the minimization without constraints, allowing three degrees of freedom to each vertex. This is enough for most of remaining cases, at the cost of geometry losses.

4 Experimental results

In this section we discuss the generated results of the proposed method. The algorithm was implemented using C++ programming language and compiled using GCC 4.6.3. All tests were performed in a Intel Xeon(R) CPU E31220 @ 3.10GHz x 4 computer with 8 GBs of RAM. The graphics card was an AMD Radeon HD 5700 series.

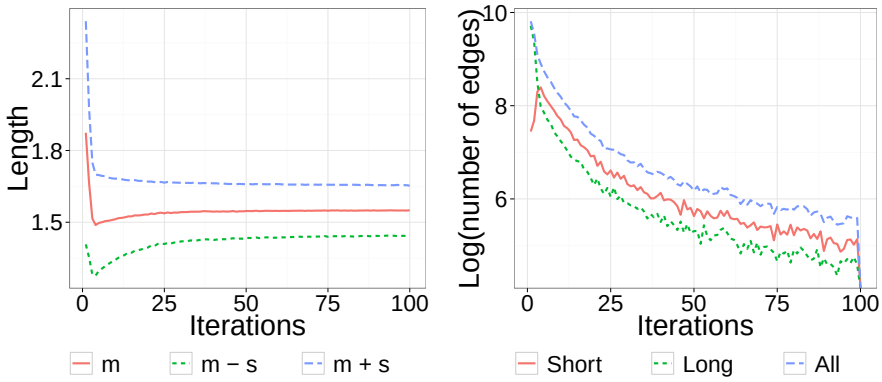

Fig. 1. Progression of Egea model through time

First we analyze the progression of the method over the time. Figure 1 exhibits the graphics for the progression of the *Egea* model, and Figure 2 shows the graphics for *Fertility* model. The graphics on the left illustrates the average edge length of the model through the iterations, and the deviation around it, where m is the average length and s is the standard-deviation. The graphic on the right shows the decrease of the amount of *long* and *short* edges over the

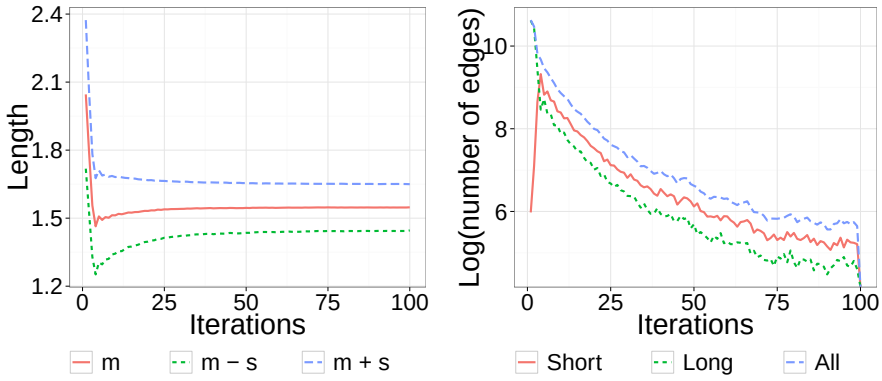

Fig. 2. Progression of Fertility model through time

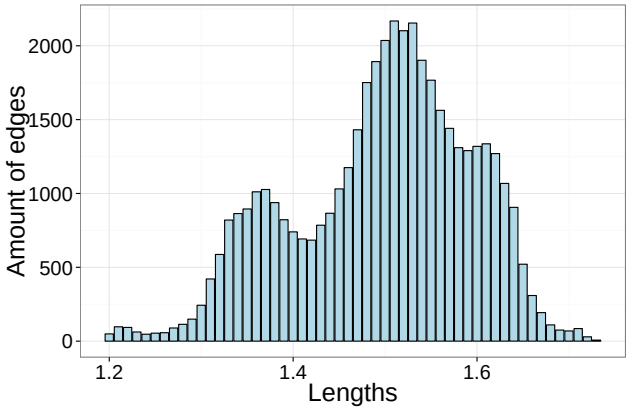

Fig. 3. Edge lengths for the processed *Egea* model, with $e_{min} = 1.2$ and $e_{max} = 1.8$.

time. For visualization purposes, the y-axis is presented in logarithmic scale. In both graphics the hundredth iteration corresponds to the post processing step of the algorithm.

One may notice that the average edge length quickly converges for a value very close to the mean of the endpoints of the interval constraining. Furthermore, the amount of *long* and *short* edges for both models is clearly reduced as the number of iterations increase. The final results for *Egea* mesh is depicted on Figure 4, and the Figure 3 illustrates the distribution of edge lengths for the final result.

We also illustrates the final results for the *Bunny* model in the Figure 5, pointing out the method behavior in regions with high curvature. As we can notice, some local geometry distortions may occur in those cases.

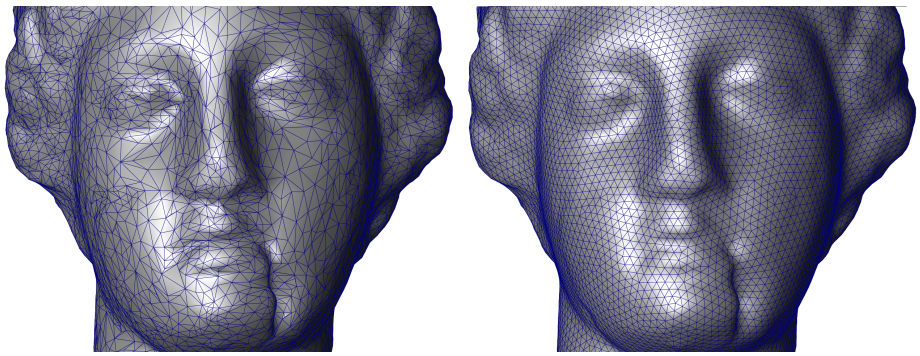

Fig. 4. *Egea* comparison between original mesh and processed mesh with parameters (1.2,1.8,100,2,0)

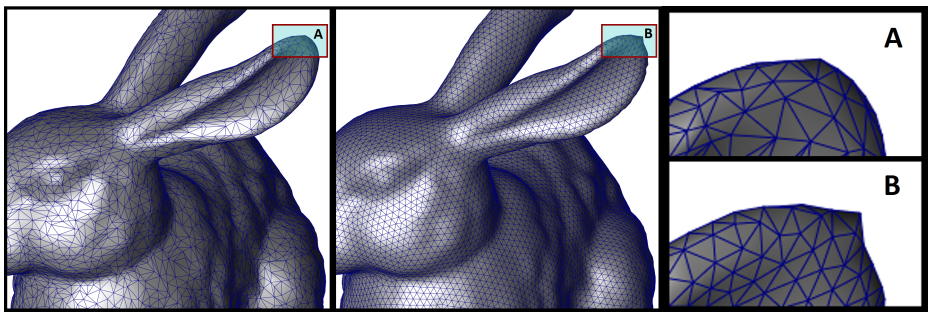

Fig. 5. *Bunny* comparison between original mesh and processed mesh with parameters (1.2,1.8,100,2,0). The images on the right evidence the local geometric distortions in a region with high curvature.

The experimental data from Egea model can be found in Table 1, where $\overline{x}$ is the average edge length, S the standard-deviation; O_{iter}, O_{cpp}, O_{final} are the total number of *short* and *long* edges after the iterations, after the constrained post processing step and after the free post processing, respectively, and *Reg. Vert.* is the percentage of vertices with valency 6.

As the data reveals, the best results are usually achieved when $k = 2$. It also reveals that parameter p can accelerate the convergence of the algorithm, once the number of final *long* and *short* edges is lower as the frequency in which the mesh is replaced is higher. However, it is important to note that it can also slightly modify the geometry, as the projection is performed in a mesh gradually more distinct from the original, and the geometry distortions caused by the Laplacian filter become permanent.

Figure 6 presents the surface *rockerarm* with variant values for e_{min} and e_{max}. The refined version greatly represents the original surface. When the process aims to larger edges, however, details of the model are lost. This is a natural consequence when a strong simplification is performed.

(e_{min},e_{max},n,k,p)	$\overline{x}$	S	O_{iter}	O_{cpp}	O_{final}	Reg. Vert.	Time(s)
Model	2.25	0.702222	-	-	33673	76.87	-
(1.2,1.8,100,1,0)	1.528363	0.150698	5568	790	3	79.482452	469.759777
(1.2,1.8,100,2,0)	1.549898	0.104768	321	13	0	89.262466	**229.957724**
(1.2,1.8,100,3,0)	1.558817	0.111454	803	47	0	88.976165	239.299956
(1.2,1.8,100,4,0)	1.56931	0.121848	1229	110	2	88.402027	270.382254
(1.2,1.8,100,1,10)	1.525323	0.158969	6119	887	4	77.620942	592.869659
(1.2,1.8,100,2,10)	1.550390	0.101163	**66**	**0**	0	90.076699	302.429551
(1.2,1.8,100,3,10)	1.558510	0.10615	199	**0**	0	89.833185	299.922871
(1.2,1.8,100,4,10)	1.569758	0.105946	243	10	1	90.880503	312.314491
(1.2,1.8,100,1,25)	**1.524605**	0.155988	5856	753	2	78.148691	561.070489
(1.2,1.8,100,2,25)	1.550454	**0.100802**	91	5	0	90.104096	282.562190
(1.2,1.8,100,3,25)	1.556311	0.107920	372	18	0	89.789667	288.441856
(1.2,1.8,100,4,25)	1.573154	0.108984	595	70	0	**90.528588**	301.354567
(1.2,1.8,100,1,50)	1.526876	0.155968	5966	684	0	78.074253	517.338868
(1.2,1.8,100,2,50)	1.549427	0.102719	177	2	0	89.747730	266.794273
(1.2,1.8,100,3,50)	1.559335	0.109731	508	46	0	89.532036	273.067145
(1.2,1.8,100,4,50)	1.569257	0.116656	907	102	0	89.014995	298.019677

Table 1. *Egea* experiments with different k and p values

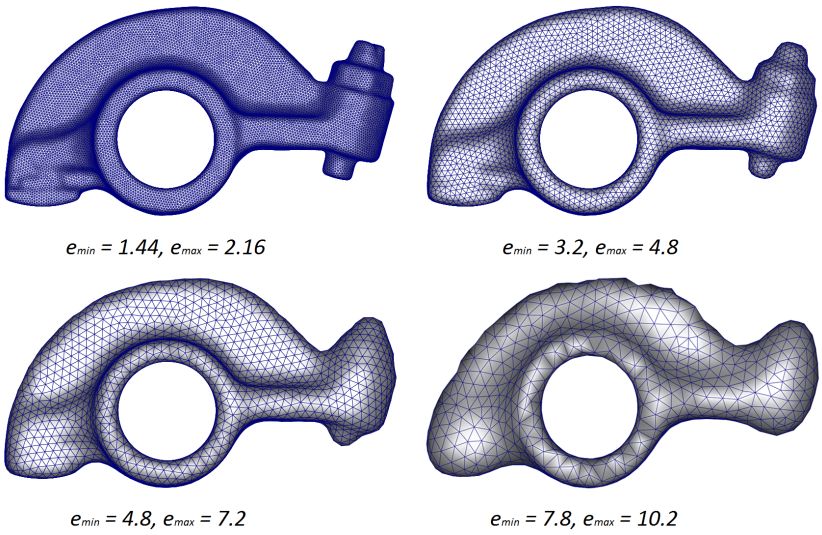

Fig. 6. Final result for *rockerarm* model with variant intervals for edge lengths. Each picture illustrates a specific length constraining.

4.1 Applications

A direct application of regular meshes is computational simulation. Uniform hexagonal meshes can be used for the simulation of nano carbon-structures, in

which each vertex represents a carbon. This kind of mesh can be generated by regular triangular meshes. In this work, we compute the dual mesh of our final triangular mesh to obtain an highly uniform hexagonal mesh. One may observe that if the primal mesh is not regular, there will be non hexagonal polygons on the dual.

To analyze the quality of the resulting dual mesh we calculate the Lennard-Jones potential with $eps = 10.1$ and $\sigma = 0.9$. As depicted in Figure 7 the processed model is much more uniform, avoiding great energy variations. This will lead to a much more stable structure.

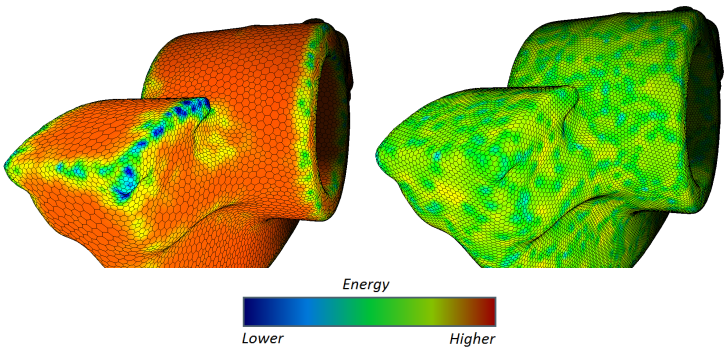

Fig. 7. The first picture is the original *rockerarm* model with potential from -33.4 to 1.21 and the second one is the processed model with potential from -16.94 to -2.99.

Another application is process a model to be more stable for other numerical methods as finite elements. Due the great regularity of the output mesh, several numerical problems are avoid.

5 Conclusion

This paper presents a method to remesh an arbitrary triangular 2-manifold mesh with all the edge lengths within an user defined interval. The main contribution of this work is an explicit definition of constraints for the edge lengths. In addition, we achieve a lower memory cost than previous approaches. We apply stellar operations to adjust the number of edges of the model, and we make use of a new approximation of the Laplacian filter for mesh relaxation, including constraints that prevent geometry losses. We also introduce a post processing step for the problems unsolved at the standard iterations.

Our results indicate that the method achieves the goal for a wide range of lengths. Moreover, the resulting mesh fairly represent the original surface in most of cases, and can be useful for several applications.

The parameters have important roles on the algorithm. For higher number of iterations n, the algorithm is more likely to reduce the number of *long* and

short edges before the post processing step. Nonetheless, if the parameter p is low, but not zero, the geometry is more likely to be softened, and the effect is aggravated as the number of iterations increase. Lower values of p are useful for accelerating the convergence, but it should be balanced with a low n value if one wants to preserve the geometry of the surface. We can also observe that the number of neighbors k for the Laplacian transformation can also accelerate the convergence, accelerating the smoothing effect as well. For the minimum and maximum values allowed for edge lengths, as the values e_{min} and e_{max} are greater and the difference e_{min} - e_{max} is smaller, the final geometry losses are greater and the convergence of the method is slower.

The major problem faced by the method is to maintain the local geometry in regions where the curvature is high. As a future work, a new approach that avoids local geometry distortions could be proposed.

5.1 Acknowledgments

Authors thank to FAPEMIG and CAPES for financial support.

References

1. Rocchini, C., Cignoni, P., Montani, C., Pingi, P., Scopigno, R.: A low cost 3d scanner based on structured light. Computer Graphics Forum **20**(3) (2001) 299–308
2. Vieira, M.B., Velho, L., Sa, A., Carvalho, P.C.: A camera-projector system for real-time 3d video. In: Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on, IEEE (2005) 96–96
3. Dembogurski, R., Dembogurski, B., Souza da Silva, R., Vieira, M.: Interactive mesh generation with local deformations in multiresolution. In Murgante, B., Misra, S., Carlini, M., Torre, C., Nguyen, H.Q., Taniar, D., Apduhan, B., Gervasi, O., eds.: Computational Science and Its Applications – ICCSA 2013. Volume 7971 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 646–661
4. Schöberl, J.: Netgen an advancing front 2d/3d-mesh generator based on abstract rules. Computing and Visualization in Science **1**(1) (1997) 41–52
5. Iijima, S., et al.: Helical microtubules of graphitic carbon. nature **354**(6348) (1991) 56–58
6. Rapaport, D.C.: The Art of Molecular Dynamics Simulation. Cambridge University Press, New York, NY, USA (1996)
7. Bommes, D., Zimmer, H., Kobbelt, L.: Mixed-integer quadrangulation. ACM Trans. Graph. **28**(3) (July 2009) 77:1–77:10
8. Ray, N., Vallet, B., Li, W.C., Lévy, B.: N-symmetry direction field design. In: ACM Transactions on Graphics. (2008) Presented at SIGGRAPH.
9. Huang, J., Zhang, M., Pei, W., Hua, W., Bao, H.: Controllable highly regular triangulation. Science China Information Sciences **54**(6) (2011) 1172–1183
10. Pampanelli, P.C.P.: Mesh generation through the mapping of triangular models into rhomboid space. M.sc. dissertation, Universidade Federal de Juiz de Fora, Advisor: Marcelo Bernardes Vieira, Co-advisors: Marcelo Lobosco and Sócrates de Oliveira Dantas (2011)

11. Pampanelli, P.P., Peanha, J., Campos, A.M., Vieira, M.B., Lobosco, M., de Oliveira Dantas, S.: Rectangular hexagonal mesh generation for parametric modeling. In: Computer Graphics and Image Processing (SIBGRAPI), 2009 XXII Brazilian Symposium on, IEEE (2009) 120–125
12. Pietroni, N., Tarini, M., Cignoni, P.: Almost isometric mesh parameterization through abstract domains. Visualization and Computer Graphics, IEEE Transactions on **16**(4) (2010) 621–635
13. Liu, Y., Pan, H., Snyder, J., Wang, W., Guo, B.: Computing self-supporting surfaces by regular triangulation. ACM Trans. Graph. **32**(4) (July 2013) 92:1–92:10
14. Aurenhammer, F.: Power diagrams: properties, algorithms and applications. SIAM J. Comput. **16**(1) (February 1987) 78–96
15. Botsch, M., Kobbelt, L.: A remeshing approach to multiresolution modeling. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM (2004) 185–192
16. Surazhsky, V., Gotsman, C.: High quality compatible triangulations. Engineering with Computers **20**(2) (2004) 147–156
17. de Oliveira, J.P.P.N.: Iterative method for edge length equalization. In: International Conference on Computational Science, Barcelona,Spain (2013) 481–490
18. Taubin, G.: A signal processing approach to fair surface design. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. SIGGRAPH '95, New York, NY, USA, ACM (1995) 351–358
19. Alliez, P., Meyer, M., Desbrun, M.: Interactive geometry remeshing. ACM Trans. Graph. **21**(3) (July 2002) 347–354
20. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing. Cambridge University Press, New York, NY, USA (1992)