Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas - ICE
Bacharelado em Ciência da Computação

Modeling nanocarbon structures using adaptive remeshing

Ramon Nogueira da Silva

JUIZ DE FORA
DEZEMBRO, 2014

Modeling nanocarbon structures using adaptive remeshing

Ramon Nogueira da Silva

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas - ICE
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Marcelo Bernardes Vieira

JUIZ DE FORA
DEZEMBRO, 2014

Modeling nanocarbon structures using adaptive remeshing

Ramon Nogueira da Silva

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS - ICE DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Marcelo Bernardes Vieira
Doctor in Computer Science

Sócrates de Oliveira Dantas
Doctor in Physics

Marcelo Caniato Renhe
Master in Systems Engeneering

JUIZ DE FORA

11 DE DEZEMBRO, 2014

For my dad.

Resumo

Este trabalho visa avaliar a qualidade de um algoritmo de remalhamento em seu uso para a geração de nanoestruturas de carbono. O algoritmo representa moléculas de nanocarbono como superfícies de variedade 2. Partindo de modelos geométricos de forma arbitrária, o método aplica um remalhamento iterativo e adaptativo para que o modelo apresente características para as quais se espera obter simulações físicas estáveis. Colocando essa esperança à prova, esse trabalho propõe avaliar a qualidade dessas estruturas com simulações de dinâmicas moleculares. O potencial REBO2 é utilizado para cálculo das forças de atração e repulsão entre os átomos.

Palavras-chave: remalhamento, simulação, nanoestruturas.

Abstract

This work seeks to evaluate the quality of a remeshing algorithm in its use for the generation of carbon nanostructures. The algorithm represents nanocarbon molecules as a 2-manifold mesh. Starting from geometric models of arbitrary shape, the method performs an iterative and adaptive remeshing in order to achieve a model that presents the features for which its physics simulations should be stable. In order to verify its effectiveness, this work proposes to evaluate the quality of the resulting structures with molecular dynamics simulations. The REBO2 potential is used to calculate the attraction and repulsion forces between the atoms.

Keywords: remeshing, simulation, nanostructures.

Acknowledgements

Thanks to my family, for always believing in me. Specially my mom, for being just amazing.

Thanks to FAPEMIG and UFJF for financial support.

Thanks to my laboratory colleagues, for all the work and all the play.

Thanks to my true friends. To Yuri, for hearing the things I needed to say. To Lucas, for saying the things I needed to hear. To Ibraim, for sharing his hapiness. To Cândida, for reminding me to read and write. To Cedrik, for the things he make me think about.

Thanks to my training partners, Zelicássio, Flávio and Diego, for helping me to unload any stress and for giving me motivation to go on. Thanks to my instructor Daniel, for teaching me more than just Krav Maga.

Thanks to my advisor Marcelo Bernardes, for putting me in the right way the times I deviated.

And thanks to Ju, for being with me.

"...a scientist must also be absolutely like a child. If he sees a thing, he must say that he sees it, whether it was what he thought he was going to see or not. See first, think later, then test. But always see first. Otherwise you will only see what you were expecting.".

Douglas Adams (So long, and thanks for all the fish)

Contents

List of Figures **7**

Abbreviations List **8**

1 Introduction **9**
1.1 Problem definition 10
1.2 Objectives 10

2 Related works **11**

3 Fundamentals **13**
3.1 Manifold 13
3.2 Polygon meshes 13
3.3 Remeshing 14
3.4 Dual mesh 14
3.5 Stellar operations 16
3.5.1 Edge split 16
3.5.2 Edge collapse 16
3.5.3 Edge flip 17
3.6 Laplacian operator 17
3.7 REBO2 potential 18
3.8 Molecular dynamics 19

4 Adaptive Remeshing **20**
4.1 Stellar operations with priority list 21
4.2 Valence optimizer 24
4.3 Global optimization 25
4.4 Local optimization 27
4.5 Projection 29
4.6 Post processing 30
4.7 Dual computing 30

5 Experimental results **32**

6 Conclusion **38**

Referências Bibliográficas **39**

List of Figures

3.1 The Möbius strip, a classical example of one-sided surface 13
3.2 Example of polygon representation of real objects and remeshing. The real object (3.2a) was discretized into a triangular mesh (3.2b) to be stored in computer memory. The model was then remeshed (3.2c) to present low standard deviation of edge lengths, characteristic that was not present before. 15
3.3 Edge split operation 16
3.4 Edge collapse operation 17
3.5 Edge flip operation 17
3.6 Example of the effect of the Laplacian operator 18

4.1 Visual overview of the Adaptive Remeshing steps. The color of the edges illustrates their length and the color of the vertices illustrates their valence. *Long* edges are in red and *short* edges are in green. Edges in blue have lengths that satisfy the interval. For vertices, the color red indicates a valence greater than 6 and the color green indicate a valence lower than 6. 22

5.1 *Egea* model generated with parameters $p = 0$ and $s = 2$. The time step of each iteration of the simulation was 0.001 ps. For the visualization, the bonds depicted in the last picture correspond to the bonds formed in the first iteration, so we can see the displacement of the repelled atoms. 32
5.2 *Egea* model generated with parameters $p = 10$ and $s = 5$. The molecule presents geometric distortions, but maintain the main structure without losing any atoms. 33
5.3 *Egea* model at the end of 5000 iterations of molecular dynamics simulations for different parameters. 34
5.4 Resulting molecules of different models. The method was capable of achieving very stable structures. 35
5.5 Rockerarm 36

Abbreviations List

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
REBO	Reactive Empirical Bond Order
REBO2	Second generation Reactive Empirical Bond Order

1 Introduction

Nowadays, nanostructures are present in several areas of technology. There are applications involving microprocessors, super resistant clothes, cancer treatment, etc. The proper study and understanding of these structures can bring a revolution in many aspects of human well being. Consequently, they have drawn attention of a large amount of researchers worldwide.

The major barrier involving studies in nanoscale is the high cost associated to the manufacturing of nanomaterials. It makes impracticable to perform laboratory experiments. This fact led to intense searches for tools that would make feasible the studies of these materials.

The computer graphics and geometry area provides supporting tools for a wide range of science's branches. It supplies scientific applications with proper interfaces for problems solving, while allowing simulation of real events, such as some physics and chemistry phenomena, inside virtual environment. This kind of support has become mandatory to the nanostructures researches. It makes possible the study of the properties and behavior of such structures at a lower cost.

A noteworthy example of this support are the works involving nanostructures generation. As we will see in Chapter 2, there are many efforts in the solving of this problem. The reason is that the simple achieving of a stable structure for simulation can be a challenging task, since there are many features that the molecules should present in order to keep the bonds of atoms. Most works found in literature are concerned to very specific structures, in a way that their solutions apply only to certain applications. General purpose algorithms are still scarce.

Considering that, this work aims to evaluate the adaptive remeshing algorithm proposed in (Hauck et al., 2015) in its use to prepare a model of arbitrary shape to be applied on simulations of nanocarbons. The method is based on the intrinsic relationship between a molecule stability and its geometry. Through the modeling of the nanostructure as a 2-manifold mesh, the algorithm is able to apply geometric transformations on the

model until it satisfies some of its bonding requirements. The final structures obtained by the algorithm will be evaluated by the simulation of their molecular dynamics, using the REBO-2 potential (Brenner et al., 2002).

1.1 Problem definition

Let $\mathcal{M}$ be a 2-manifold input mesh of arbitrary geometry. The problem is to obtain a mesh $\mathcal{M}'$ of minimal geometric discrepancy from $\mathcal{M}$, in such a way that, being $|e'_j|$ the length of the edge e'_j in angstrons, then $1.2 \leq |e'_j| \leq 1.8$ for every $e'_j \in \mathcal{E}'$, where $\mathcal{E}' \subset \mathcal{M}'$ is the set of edges from $\mathcal{M}'$. Also, if $\mathcal{M}'$ is submitted to t iterations of the REBO-2 molecular dynamics simulations, then $\mathcal{M}'_t$ remains homeomorphic to $\mathcal{M}$ and the geometric distortions of $\mathcal{M}'_t$ are minimal, for any t in $[0, +\infty[$.

1.2 Objectives

The main objective of this work is to produce nanocarbon structures of any arbitrary shape that are stable for simulations of molecular dynamics. The secondary objectives are:

- To understand the geometric features associated with the stability of nanocarbon molecules;
- To describe the implementation of a remeshing method capable of achieving models that satisfy those features for different input sizes and shapes;
- To verify the stability of the resulting structures with molecular dynamics simulations;
- To compare the method performance and efficiency against energy minimization approaches.

2 Related works

The generation of carbon nanostructures can be done in two different ways. The first is through physics approaches, in which the goal is to minimize the total potential energy of a system in order to achieve stable molecule. These methods usually make use of a potential energy expression, e.g. the Lennard-Jones potential (Lennard-Jones, 1931) and the REBO2 potential (Brenner et al., 2002), which is minimized with a numerical method, such as the conjugate gradient method (Press et al., 1992). The second is through geometric approaches, in which the geometric features are considered rather than the physics ones, in a way that, if the geometric requirements are achieved, so are the physics. This chapter provides some examples of this kind of works.

Works focused on direct geometric modeling of nanostructures make a study of what geometric features are associated to what physics features. A natural disadvantage of this approach is that solutions are usually developed to very specific problems, e.g. the generation of carbon nanotubes. There are few evidences of methods focused on generating structures for general purposes.

The first efforts to generate nanostructures with a geometric approach date from 1992, when Hamada et al. (1992) proposes a way to describe any carbon nanotube using only two integers n and m, known as Hamada indexes. These indexes relate to the generation of straight lines on hexagonal grids (Liu, 1993), a concept later extended in the work of Pampanelli et al. (2009) for the generation of hexagonal models that can represent several kinds of nanocarbon structures.

Zsoldos et al. (2004) proposes a method to construct junctions of carbon nanotubes. The work describes some patterns used for building different types of junctions. The structures are composed mostly by hexagons, but pentagons appear in the regions where the structure must be closed. Similarly, heptagons appear in the regions where the diameter of the junction is higher. The method is limited to the specific geometry and topology of the junctions of carbon nanotubes.

It is also possible to combine energy minimization approaches and geometric

methods. Lászlo and Rassat (2003) present a study of the geometric characteristics of nanotubes that, if physically optimized, naturally result in an helicoidal structure. Again, this work is limited to these structures.

As a different approach, it is possible to remesh models to satisfy physical conditions rather than generating those structures. The advantage of remeshing methods over direct modeling methods is the possibility of working with a higher degree of freedom, since remeshing methods usually apply to arbitrary model. Although remeshing methods, in general, are used to improve geometric features, there are works that apply a remeshing approach to achieve quality in other aspects, like good cloth simulation (Narrain et al., 2012) and the generation of surfaces that are self-supporting (Liu et al., 2013).

For the generation of nanocarbon structures, it is necessary to obtain regular trivalent models. Instead of approaching this problem directly, Peçanha et al. (2013) argue that remesh a triangular mesh to be more regular is not only an equivalent problem, but also more general for use in applications. In that work, an iterative method for edge length equalization is presented, in such a way that the dual of the final model could be used for nanocarbon simulations. The method adjust the amount of edges with stellar operations and improve their distribution with a low-pass filtering over the surface.

Following that line, Hauck et al. (2014) propose an improvement to the method. It inserts a more strong constraining to the edge lengths, and adds a post processing step to achieve convergence. The work have the disadvantage of presenting serious geometric distortions for some cases. In order to correct it, a new improvement to the method was proposed in (Hauck et al., 2015). The post-processing step was integrated to the iterations cycle, achieving lower geometric distortions and a faster convergence rate. The authors demonstrate that the potential energy of a model generated with their technique is much more uniform than the original model. However, no real simulation was performed to validate the stability of the model. Because of that, the goal of the present work is to verify that stability.

3 Fundamentals

This chapter presents some basic concepts for the understanding of this work.

3.1 Manifold

Manifolds are topological spaces, immersed in spaces of higher dimension, that are similar to the Euclidian space at small regions around each point. A surface is a k-manifold if it is immersed in a Euclidian space $\mathbb{R}^n$, with $k \leq n$, and its homeomorphic to the Euclidian space $\mathbb{R}^k$ in the neighborhood of each point (Guillemin and Pollack, 1974).

This work operates over 2-manifold surfaces immersed in the tridimensional space. This kind of surface can be orientable if it has two sides. Although uncommon, some surfaces can present only one side, as depicted in Figure 3.1. This work is concerned only to orientable surfaces.

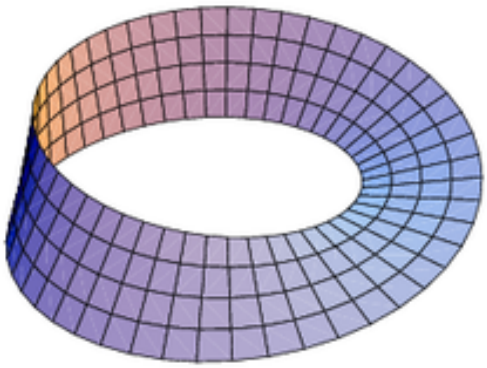


Figure 3.1: The Möbius strip, a classical example of one-sided surface

3.2 Polygon meshes

The computer is a machine that works with discrete domains and codomains. As a consequence, continuous manifolds must be discretized to be represented in computer. The most common way to do this is to represent the surface as a polygon mesh.

Polygon meshes are a set of faces, edges, and vertices. Each vertex is a position

and each edge is a connection between two vertices. A face is a closed collection of edges, consisting on a polygon, usually convex due to its rendering simplicity. Figure 3.2 illustrates how a real object is represented as a polygon mesh.

To present a formal definition, let $\mathcal{V}$ be a set of vertices. Let $\mathcal{E}$ be a set of edges, with $\overline{e}_j \in \mathcal{E}$ if $\overline{e}_j$ is the pair $\{v_1^j, v_2^j\} \subset \mathcal{V}$. Also, let $\mathcal{F}$ be a set of faces, with $\overline{f}_j \in \mathcal{F}$ if $\overline{f}_j$ is a closed collection $\{e_1, ..., e_s\} \subset \mathcal{E}$. Then, $\mathcal{M} \equiv \mathcal{V} \cup \mathcal{E} \cup \mathcal{F}$.

For most applications, including this work, the meshes are chosen to be composed of triangles only. That is because triangles are the only polygon capable of defining a plane without ambiguity. Moreover, any mesh composed of convex polygons can be easily transformed into a triangular mesh through a process known as triangulation.

3.3 Remeshing

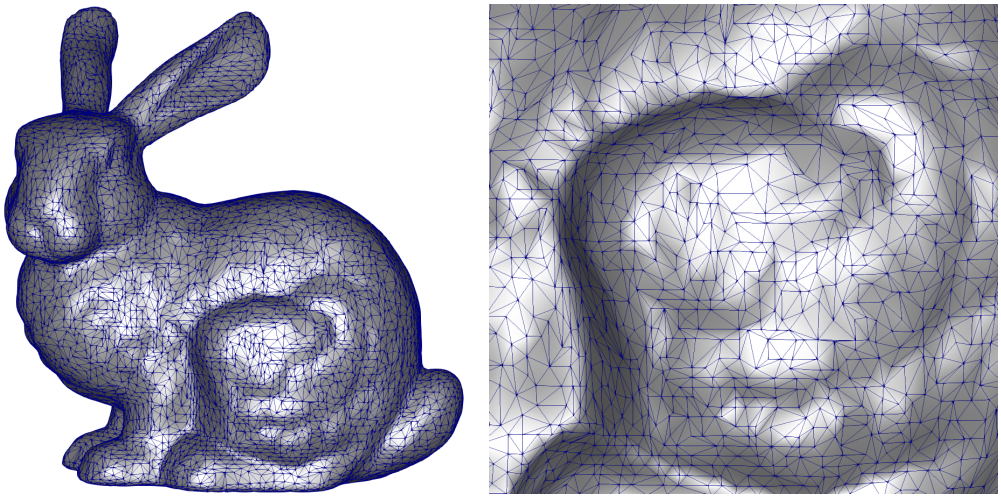
The process of remeshing a given mesh $\mathcal{M}$ is to obtain a new mesh $\mathcal{M}'$ that is close enough to $\mathcal{M}$ and satisfy certain quality that $\mathcal{M}$ does not (Botsch et al., 2010). There is no precise definition of how close the mesh must be. Each application can define its own similarity criteria, that can be topological, geometric, or the maintenance of defined constraints. Bommes et al. (2009) enumerate some quality aspects usually considered.

3.4 Dual mesh

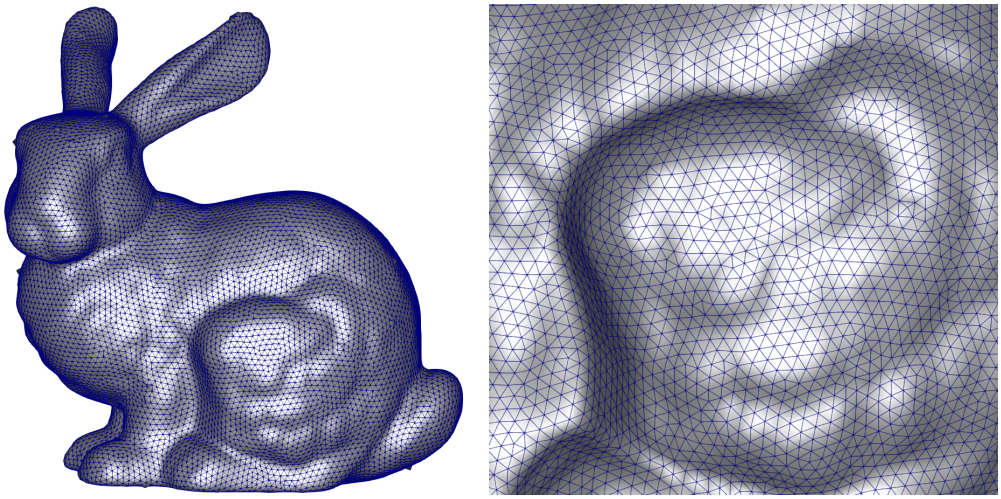
According to Taubin (2002), the dual of a 2-manifold mesh can be defined as a mesh with the same topology, in which the location of its faces correspond to the locations of the vertices in the primal, and vice versa. In addition, the vertices of the dual are positioned in the centroid of its corresponding face in the primal.

One can easily notice that the number of sides of a polygon in the primal mesh corresponds to the valence of the corresponding vertex on the dual mesh. Consequently, the dual of triangular mesh is a trivalent mesh. This property allows the generation of nanocarbon molecules from triangular meshes, since those molecules are composed of trivalent bonds.

(a) The original object for the *Stanford Bunny*



(b) The polygon mesh representing the *Stanford Bunny*



(c) The *Stanford Bunny* remeshed to present low standard deviation of edge lengths.

Figure 3.2: Example of polygon representation of real objects and remeshing. The real object (3.2a) was discretized into a triangular mesh (3.2b) to be stored in computer memory. The model was then remeshed (3.2c) to present low standard deviation of edge lengths, characteristic that was not present before.

3.5 Stellar operations

Stellar operations are local modifications on a mesh that preserve its Euler characteristic. The Euler characteristic of a mesh $\mathcal{M}$, denoted by χ, is defined as:

$$\chi = |\mathcal{V}| - |\mathcal{E}| + |\mathcal{F}|$$

where $|\mathcal{V}|$, $|\mathcal{E}|$ and $|\mathcal{F}|$ are, respectively, the number of vertices, edges and faces of $\mathcal{M}$.

For triangular meshes, there are three stellar operations applied in this work: the *edge split*, the *edge collapse* and the *edge flip*.

3.5.1 Edge split

This operation splits an edge $\overline{e}_j$ into two new, also creating two additional edges. Firstly, a new vertex v_n is created in an arbitrary position over $\overline{e}_j$. Next, we connect this new vertex to the two opposite vertices of $\overline{e}_j$. The operation is illustrated in Figure 3.3.

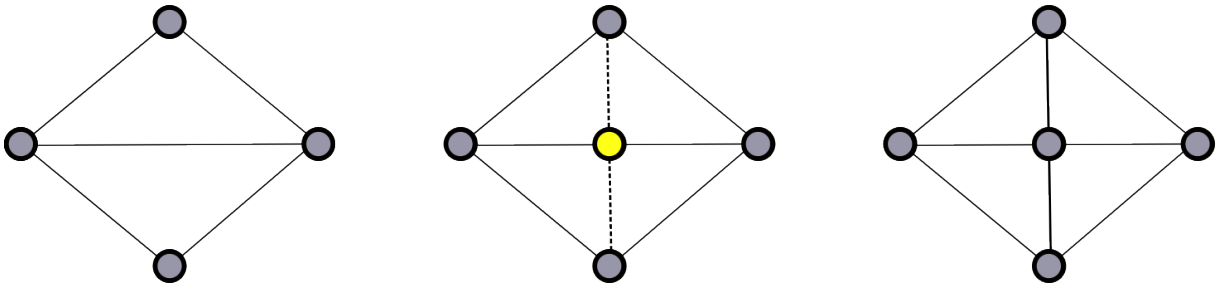


Figure 3.3: Edge split operation

3.5.2 Edge collapse

The edge collapse operation removes an edge $\overline{e}_j$ from the mesh. One of the original vertices connected by $\overline{e}_j$ is removed, and all the edges connected to it are connected to the remaining vertex. The remaining vertex is then repositioned in any arbitrary point over $\overline{e}_j$. The operation is illustrated in Figure 3.4.

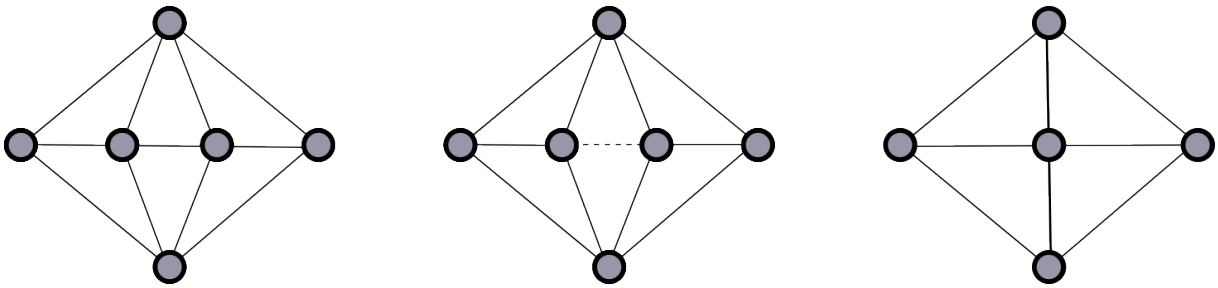


Figure 3.4: Edge collapse operation

3.5.3 Edge flip

This operation replaces an edge $\overline{e}_j$ with an edge connecting its opposing vertices. It is illustrated in Figure 3.5.

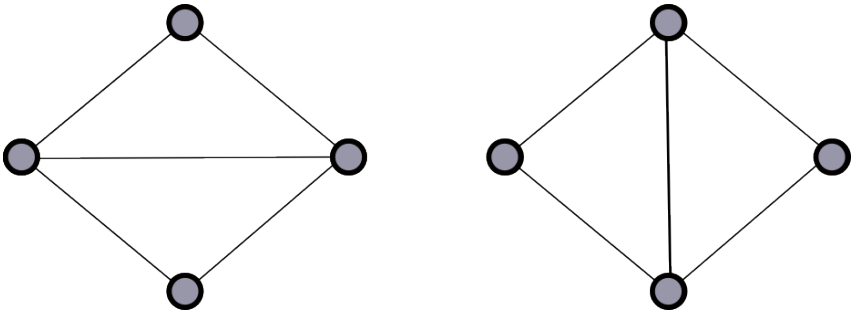


Figure 3.5: Edge flip operation

3.6 Laplacian operator

The Laplacian operator ∇^2 of a function $g : \mathbb{R}^n \mapsto \mathbb{R}$ is a measure of its dispersion in $\mathbb{R}^n$. It is given by the following formula:

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

If the function g is a signal, the value of the Laplacian increases along with the high frequencies of g. Therefore, to obtain the values of $\{x_1, ..., x_n\}$ for which the Laplacian is zero is equivalent to eliminate the high frequencies of g. Consequently, the Laplacian operator works as a low-pass filter.

Taubin (1995) demonstrates that a surface can be treated as a signal, and that the problem of smoothing the surface can be reduced to a low-pass filtering of the surface's signal. He proposes a discrete linear approach, denoted by $\mathcal{L}$, to calculate the Laplacian

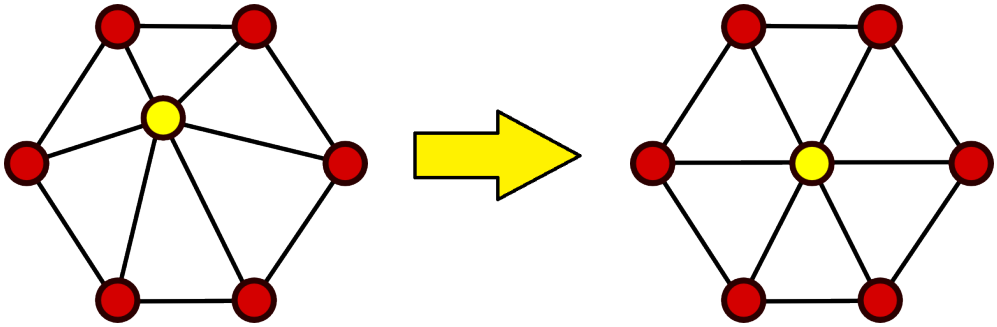


Figure 3.6: Example of the effect of the Laplacian operator

for remeshing applications, given by the equation:

$$\mathcal{L}(\overline{v}_i) = \sum_{\overline{v}_j} w_{ij}(\overline{v}_i - \overline{v}_j), \tag{3.1}$$

where $\overline{v}_j$ are all the vertices in the first star of $\overline{v}_i$ and w_{ij} is the weight of the edge $\overline{\overline{v}_i\overline{v}_j}$, with $\sum_j w_{ij} = 1$.

The literature provides many strategies for assign the w_{ij} value of an edge, e.g. schemes with edge lengths (Taubin, 1995) and cotangent (Alliez et al., 2002). This work applies the scheme of neighborhood (Peçanha et al., 2013).

The effect of the Laplacian in a mesh moves all vertices to the median of their neighbors, as depicted in Figure 3.6. This tends to reduce the standard deviation of the edge lengths, making the mesh more smooth and more equalized.

3.7 REBO2 potential

The REBO2 model (Brenner et al., 2002) was proposed as a low cost function to calculate the potential energy from covalent bonds. Its efficiency comes from the fact that only the nearest neighbors iterations are considered, in contrast to total energy approaches. This fact makes the REBO2 a widely used model for large atomic systems. In this model, the binding energy E_b is given by:

$$E_b = \sum_i \sum_{j,j>i} [V^R(r_{ij}) - b_{ij}V^A(r_{ij})], \tag{3.2}$$

where, $V^R(r)$ and $V^A(r)$ are the pair-additive functions that represent the interatomic repulsions and attractions, r_{ij} is the distance between atoms i and j and b_{ij} is the bond order function between atoms i and j. Wang (2006) describes the whole formulation with great clarity.

3.8 Molecular dynamics

Molecular dynamics are computer simulations of the behavior of molecules over time. The atoms and molecules are modeled in virtual environment, where they interact so we can observe their motion. Usually, if we have an expression for the potential energy between the particles, the trajectory of each atom can be determined by the Newton's law of motion. Time is discretized in intervals of duration t, in a way that each iteration of the algorithm computes the movement of each atom for the next instant.

In this work, all molecular dynamics performed make use of the REBO2 expression for potential energy.

4 Adaptive Remeshing

This work seek to evaluate the efficiency of the algorithm Adaptive Remeshing, from the work *Adaptive remeshing for edge length interval constraining* (Hauck et al., 2015), accepted for publication, for generating nanocarbon molecules. The algorithm operates over triangular meshes and is concerned to the edge lengths of the model. More precisely, it aims to obtain a mesh without any long or short edges, defined as:

$$\begin{matrix} \text{long}, & \text{if } |e_j| > l^{max} \\ \text{short}, & \text{if } |e_j| < l^{min} \end{matrix} ,$$

where $\bar{e}_j \in \mathcal{E}$ is an edge from $\mathcal{M}$, l^{min} is the smallest value allowed for edge length and l^{max} is the biggest value allowed for edge length.

Although it is not the main goal of the algorithm, it also improves the valence of the vertices as a way to achieve more equalized edge lengths. The method is iterative and adapts itself over time in accord to the current features of the model. At the end of the iterations, it computes the dual of the resulting mesh to obtain a the trivalent mesh used for simulations. Its input is the tuple $(\mathcal{M}, l^{min}, l^{max}, n, k, p, s)$, where $\mathcal{M}$ is a 2-manifold mesh, n is the maximum number of iterations, k is the number of rings used for the Laplacian approximation, p is the number of steps performed before each replacement of the projection mesh and s is the percentage threshold of *short* and *long* edges for which the algorithm switches the optimization strategy.

For this work, we seek to obtain models in which these characteristics are optimized for nanocarbon simulations. As the edges represent the bonds between the atoms, their length neither can be so long that the attractive force is not enough to keep the bond, or so short that the repulsive force between the atoms destroy the molecule. Hence, this work constrains the length of the edges to the interval between 1.2 and 1.8 angstrons.This value was chosen based on Figure 1 of (Brenner et al., 2002), in which the potential energies are given as a function of the interatomic distance. The figure indicates that a value close to 1.5 angstrons gives the best results. The lengths of the edges of the models were constrained to the interval that allows up to 20% of standard deviation. Moreover, the

method imposes that the valence of each vertex cannot be lower than 5 or greater than 7. As each vertex corresponds to a polygon in the dual space, its valence corresponds to the number of sides the polygon will have. The best case scenario would be a structure composed of hexagons only. However, in order to compensate the different curvatures over the surface, some pentagons and heptagons may be necessary.

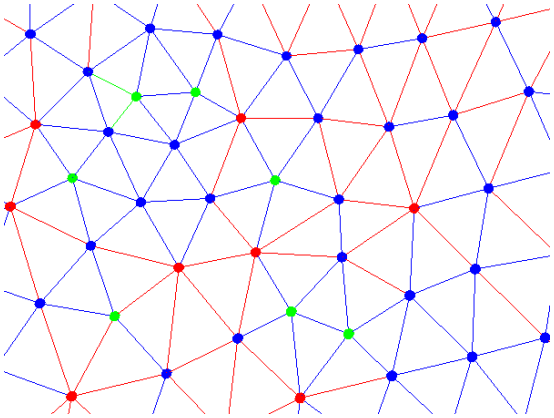
Algorithm 1 is an overview of the method, and Figure 4.1 provides a visual overview. Detailed information about each step will be provided in the following sections.

Algorithm 1: AdaptiveRemeshing($\mathcal{M}, l^{min}, l^{max}, n, k, p, s$)

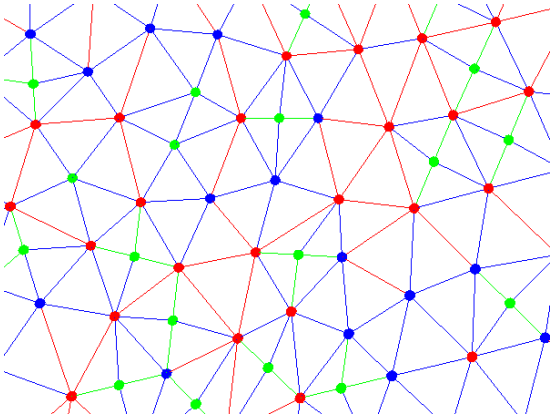
```
1  M' = Copy(M)
2  M_p = Copy(M)
3  m = (l^min + l^max) / 2
4  while (short + long) > 0 and iter < n do
5      if p > 0 and (i mod p) = 0 then
6          M_p = Copy(M')
7      end if
8      StellarOperations(M')
9      ValenceOptimizer(M')
10     if CalcEdgesPercent() ≤ s then
11         NonLinearOptimizer(M')
12     end if
13     else
14         LowPassFiltering(M', r)
15     end if
16     M' = Projection(M', M_p)
17 end while
18 NonLinearOptimizer(M')
19 PostProcess(M')
20 M' = Dual(M')
21 return M'
```

4.1 Stellar operations with priority list

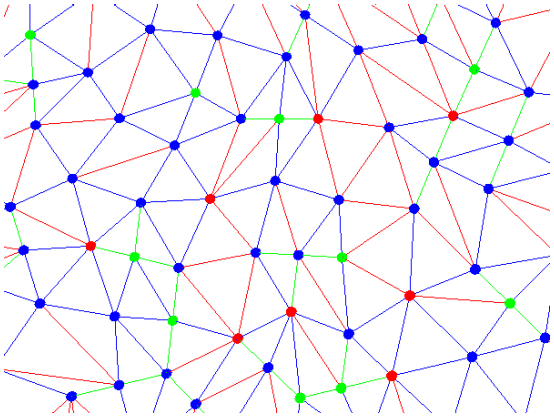
The first step of the algorithm seeks to adjust the amount of vertices through the application of stellar operations. This adjustment must be local, because arbitrary models have regions that need to be simplified and others that need to be refined (Peçanha et al., 2013).



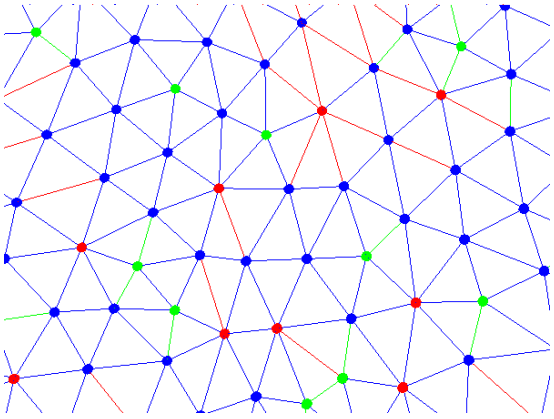
(a) The original mesh before any processing.



(b) The model after the application of stellar operations. *Long* edges are split and *short* edges are collapsed.



(c) Result of the valence optimizer step. The valence of most vertices become 6, as we can see by the amount of blue vertices.



(d) Result of the optimization step, in which the edges are equalized. The triangles become more regular, and the length of most edges are constrained to the interval (edges in blue).

Figure 4.1: Visual overview of the Adaptive Remeshing steps. The color of the edges illustrates their length and the color of the vertices illustrates their valence. *Long* edges are in red and *short* edges are in green. Edges in blue have lengths that satisfy the interval. For vertices, the color red indicates a valence greater than 6 and the color green indicate a valence lower than 6.

The ideal amount of edges depends on the the average edge length m of the interval. However, being m_i the average edge length of the mesh at the i-th iteration, if m_i is much smaller than m, the model will have to be strongly simplified, what can lead to the degeneration of the mesh. In order to prevent it, the method computes intermediate values for l^{min} and l^{max} that only allow smooth transformations. These values are:

$$\begin{aligned} l_i^{min} &= MIN(2 \cdot m_i, m) - \tfrac{l^{max} - l^{min}}{2} \\ l_i^{max} &= MIN(2 \cdot m_i, m) + \tfrac{l^{max} - l^{min}}{2} \end{aligned} \quad .$$

Edges whose length is greater than l_i^{max} become candidates to be split, and edges whose length is lower than l_i^{min} become candidates to be collapsed. As in any series of local modifications, the operations order impact on the results. The work of Peçanha et al. (2013) proposes to create a priority list of edges as a way to ensure the method stability and improve its convergence. The priority d_j associated to each edge $\overline{e}_j$ is the deviation $d_j = ||\overline{e}_j| - m_i|$. A list L_p is created as a set of edges $\{e_1, ..., e_c\} \subset \mathcal{E} \subset \mathcal{M}$, with $d_1 \geq d_2 \geq ... \geq d_{c-1} \geq d_c$, where $\overline{e}_j \in L_p$ if $|\overline{e}_j| \notin [l_i^{min}, l_i^{max}] \in \mathbb{R}$ (Peçanha et al., 2013).

Algorithm 2: CreatePriorityList($\mathcal{M}', l_i^{min}, l_i^{max}$)

```
1 foreach ē_j ∈ E ⊂ M' do
2     if |ē_j| < l_i^min then
3         priorityList.Add(ē_j)
4     end if
5     else if |ē_j| > l_i^max then
6         priorityList.Add(ē_j)
7     end if
8     SortDescent(priorityList)
9 end foreach
```

After the list is set, the algorithm traverses it processing each edge, in order. Edges with length smaller than l_i^{min} are collapsed, and edges with length greater than l_i^{max} are split. Each edge processed has its both vertices marked as visited. Edges for which both vertices were marked are not processed. This prevents a vertex from being moved too many times in one single iteration, improving the vertices distribution per iteration.

Both the vertex generated by the edge split operation and the remaining vertex of the edge collapse operation can be placed in an arbitrary position over the operated

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

```
foreach ē_j ∈ L_p do
    if bothVerticesVisited(ē_j) then
        continue
    end if
    else if |ē_j| < l_i^min then
        EdgeCollapse(ē_j)
    end if
    else if |ē_j| > l_i^max then
        EdgeSplit(ē_j)
    end if
    MarkVerticesAsVisited(ē_j)
end foreach
```

edge. Hauck et al. (2014) proposes to choose this position based on the length of the edges affected by the corresponding stellar operation. The work proposes an error measurement S for the edges connected to the vertex of interest $\overline{v}_i$, given by the equation:

$$S = \sum_{\overline{e}_j} (|\overline{v}_i - \overline{v}_j| - m)^2, \tag{4.1}$$

where $\overline{v}_j$ are the vertices connected to $\overline{v}_i$. The vertex $\overline{v}_i$ is placed in the position that minimizes Equation 4.1.

4.2 Valence optimizer

For a fully equalized triangular mesh, the projection of all triangles to the tangent plane should be equilateral. In this scenario, their internal angles would be 60°, and the valence of each vertex would be 6. Hence, in order to achieve more equalized edge lengths, the algorithm performs an optimization of the valence of the vertices, in a way that they are closer to the ideal value 6.

Surazhsky and Gotsman (2003) describe a simple procedure, based on edge flips, to optimize the valence. For each edge $\overline{e}_j$, let v_1^j and v_2^j be the vertices connected by $\overline{e}_j$, and let v_3^j and v_4^j be the opposite vertices of $\overline{e}_j$. Denoting by $vl(v)$ the valence of the vertex v, if the desirable edge length is ϕ, we can describe the valence error ψ_j associated

to the edge $\overline{e}_j$ as:

$$\psi_j = |vl(v_1^j) - \phi| + |vl(v_2^j) - \phi| + |vl(v_3^j) - \phi| + |vl(v_4^j) - \phi| \tag{4.2}$$

For each edge in the model, the method computes its ψ_j, and then applies the edge flip operation. If the value of ψ_j is reduced, the operation is maintained. Otherwise, the operation is undone.

For this work, an improvement of this strategy is proposed. Since the final mesh should not present polygons with fewer than five sides nor more than seven sides, it is necessary to increase the penalty for higher deviations in the valence. So, the error of each vertex on Equation 4.2 is modified to be quadratic, and the new error formula is given by:

$$\Psi_j = (vl(v_1^j) - \phi)^2 + (vl(v_2^j) - \phi)^2 + (vl(v_3^j) - \phi)^2 + (vl(v_4^j) - \phi)^2 \tag{4.3}$$

The algorithm proceeds as previously described, using the error proposed in Equation 4.3.

In addition, this step is now performed iteratively. Each iteration process all the vertices in the model. If any edge flip operation is performed and not undone, the method proceeds to a new iteration. When no improvement is achieved, the step is finished. Algorithm 4 illustrates the process.

4.3 Global optimization

Once the method achieves a good amount of vertices and improves their valences, it is necessary to improve the vertices distribution over the mesh to obtain equalized edge lengths. There are two main strategies for doing so. The first one is a global optimization strategy, in which the distribution of the vertices of the model is improved as a whole. In contrast, there is the local optimization strategy, in which the distribution of the vertices is improved only in small regions at each time.

For the Adaptive Remeshing algorithm (Hauck et al., 2015), the global strategy

Algorithm 4: ValenceOptimizer($\mathcal{M}$)

```
improved = true
while improved do
    improved = false
    foreach ē_j ∈ E ⊂ M do
        preDev = (vl(v_1^j) − 6)^2 + (vl(v_2^j) − 6)^2 + (vl(v_3^j) − 6)^2 + (vl(v_4^j) − 6)^2
        EdgeFlip(ē_j)
        posDev = (vl(v_1^j) − 6)^2 + (vl(v_2^j) − 6)^2 + (vl(v_3^j) − 6)^2 + (vl(v_4^j) − 6)^2
        if preDev < posDev then
            EdgeFlip(ē_j)
        end if
        else
            improved = true
        end if
    end foreach
end while
```

is applied at earlier stages and the local strategy is applied at later stages, controlled by the parameter s. If the percentage of *long* and *short* edges is greater than s, the method applies the global strategy. Otherwise, it applies the local strategy. The local strategy is useful to correct troubling regions of the model, but since it ignores the edges around those regions, it tends to decrease the final quality of the mesh as a whole (Hauck et al., 2015). Because of this, it is recommended to use a value of s as low as possible, while still enough to ensure the convergence.

The global optimization strategy tries to obtain a mesh with a more uniform distribution of vertices. Taubin (1995) demonstrates that this problem can be reduced to a low-pass filtering over the signal surface. Hence, this step performs an application of the Laplacian operator over the surface.

Each vertex $\overline{v}_j$ is moved to the center of mass of its neighbors until the k-star, weighted by their ring number. The contribution of closer vertices (vertices with lower ring number) is greater than the contribution of distant vertices, as proposed by Peçanha et al. (2013). First, the new position $\overline{v}_j^*$ of each vertex $\overline{v}_j$ is obtained. Following that, a constraint is added to the Laplacian in a way that the vertices displacements are limited to the tangent plane. To do this, the displacement vector $\delta_j = \overline{v}_j^* - \overline{v}_j$ is projected onto the plane tangent to $\overline{v}_j$ by removing the normal component. Therefore, being δ_j^N the normal component of δ_j, then the displacement δ_j^T over the tangent plane is given by

$\delta_j^T = \delta_j - \delta_j^N$. Hence, the new position v_j^T for the vertex $\overline{v}_j$ is $v_j^T = \overline{v}_j + \delta_j^T$. After all the new positions are computed, the vertices are updated. The vertices on the borders, if any, remain unchanged to prevent the shrink of the model.

Hauck et al. (2014) proposes to approximate the discrete Laplacian with an iterative algorithm (Algorithm 5), rather than solving the linear system resulting from Equation 3.1. Although not exact, this algorithm presents very similar results with a much lower computational cost.

Algorithm 5: LowPassFiltering($\mathcal{M}'$, k)

foreach $\overline{v}_j \in \mathcal{V}' \subset \mathcal{M}'$ **do**
  $kStar$=`getKStar` ($\overline{v}_j$,k)
  fat=0
  foreach $\overline{v}_i \in kStar$ **do**
    $\overline{v}_j^*$=$\overline{v}_j^*$+$\frac{\overline{v}_i}{star}$
    fat=fat+$\frac{1}{star}$
  end foreach
  $\overline{v}_j^*$=$\frac{\overline{v}_j^*}{fat}$
  δ_j=$\overline{v}_j^*$-$\overline{v}_j$
  δ_j=δ_j -`projection` (δ_j,N_j)
end foreach
foreach $\overline{v}_j \in \mathcal{V}' \subset \mathcal{M}'$ **do**
  if $\overline{v}_j \notin \mathcal{B}$ **then**
    $\overline{v}_j$=$\overline{v}_j$+δ_j
  end if
end foreach

4.4 Local optimization

The effectiveness of the global optimization is reduced as the iterations progresses. At certain point, some regions cannot be corrected by the Laplacian operator. In order to solve this issue, a local optimization was proposed. First, Hauck et al. (2014) propose an error measurement for a region around an edge:

$$\sum_{\overline{v}_i} \sum_{\overline{v}_j} (|\overline{v}_i - \overline{v}_j|^2 - m^2)^2, \tag{4.4}$$

where $\overline{v}_i$ are the vertices in the forth star of the edge and $\overline{v}_j$ the vertices in the first star of $\overline{v}_i$.

If we find the positions $\overline{v}_i^*$ and $\overline{v}_j^*$ for $\overline{v}_i$ and $\overline{v}_j$, respectively, that minimize Equation 4.4, we obtain a very uniform model. Nonetheless, as the vertices have three degrees of freedom, the geometric distortions caused by these displacements are severe. In order to prevent it, Hauck et al. (2014) proposes to constrain the displacement of the vertices to the tangent plane.

To do this, first consider an orthonormal base $< \mathrm{N}_i, \mathrm{T}_i, \mathrm{T}'_i >$ for the plane tangent to the vertex $\overline{v}_i$, where N_i is the normal and T_i, $\mathrm{T'}_i$ are orthogonal directions over the plane. If we move $\overline{v}_i$ to a new position $\overline{v}_i^*$, the displacement $\delta_i = |\overline{v}_i^* - \overline{v}_i|$ can be described as the sum:

$$\delta_i = \alpha_i \cdot T_i + \beta_i \cdot T'_i + \gamma_i \cdot N_i, \tag{4.5}$$

where α, β and γ are the displacements in the directions T_i, T'_i and N_i, respectively. So, to find positions $\overline{v}_i^*$, $\overline{v}_j^*$ for $\overline{v}_i$ and $\overline{v}_j$ that minimize Equation 4.4 is equivalent to find the values of α, β and γ that minimize:

$$\sum_{\overline{v}_i} \sum_{\overline{v}_j} [|\overline{v}_i + \alpha_i \cdot T_i + \beta_i \cdot T'_i + \gamma_i \cdot N_i - (\overline{v}_j + \alpha_j \cdot T_j + \beta_j \cdot T'_j + \gamma_j \cdot N_j)|^2 - m^2]^2. \tag{4.6}$$

If we want to limit the displacements to the tangent plane, we make $\gamma = 0$ in Equation 4.6. Then, the final error measurement to minimize is:

$$\sum_{\overline{v}_i} \sum_{\overline{v}_j} [|\overline{v}_i + \alpha_i \cdot T_i + \beta_i \cdot T'_i - (\overline{v}_j + \alpha_j \cdot T_j + \beta_j \cdot T'_j)|^2 - m^2]^2. \tag{4.7}$$

Both α_i and β_i are set zero when the index i does not exist.

The Equation 4.7 is minimized with the conjugate gradient method, as seen in Press et al. (1992). The minimization is performed around one edge at each time, to improve the performance and avoid numerical issues.

In a similar way to what occurs in the stellar operations, as this step is a series of local modifications, the order in which they will be applied impact on the results. As a consequence, a new priority list must be created. This list set the higher priorities to

the regions in which the operation will impact in the greatest amount of *long* and *short* edges.

For this new list, the priority P_j assigned to each edge $\overline{e}_j$ is defined as $P_j = \overline{e}_j^l + \overline{e}_j^s$, where $\overline{e}_j^l$ and $\overline{e}_j^s$ are the amount of *long* and *short* edges in the neighborhood of $\overline{e}_j$, respectively. This neighborhood is considered to be all the edges until the forth star of $\overline{e}_j$. The list L_p^o is the set $\{e_1, ..., e_t\} \subset \mathcal{E} \subset \mathcal{M}$, with $P_1 \geq P_2 \geq ... \geq P_t$, where $\overline{e}_j \in L_p^o$ if $|\overline{e}_j| \notin [l^{min}, l^{max}] \in \mathbb{R}$ (Hauck et al., 2015). After L_p^o is set, the minimization of Equation 4.7 is performed for each edge on the list, in order.

4.5 Projection

After the relaxation of the mesh, the vertices assume positions that are not found in the original surface $\mathcal{M}$. Those vertices need to be projected over $\mathcal{M}$ if we expect to minimize the geometric distortions. However, being $\{v'_1, ...v'_u\}$ the vertices of $\mathcal{M}'$, the problem of finding its corresponding vertices over the surface of $\mathcal{M}$ can be reduced to the problem of finding the nearest points between two sets (Botsch et al., 2010), which is a problem of high computational cost. Instead of finding an optimal solution, Peçanha et al. (2013) propose a method of low cost to find a good candidate position.

The method finds the vertex $v^m \in \mathcal{M}$ that is nearest to v'_j. Next, v'_j is projected orthogonally on every triangle incident to v_m, obtaining the candidates $\{v_1^m, ...v_t^m\}$. The candidate v_c^m for which we obtain lower value of $|v'_j - v_c^m|$ is chosen as the new v'_j.

For some cases, the nearest vertex does not provide the best correspondence. In this work, as a way to improve this approximation, the method is processed for the five nearest vertices, instead of only one. This improvement also does not guarantee that an optimal result is found, but can find better correspondences in some cases.

The mesh $\mathcal{M}_p$ in which we perform the projection can be updated with the progressing of the algorithm. At each p iterations, the projection mesh $\mathcal{M}_p$ is replaced with the current mesh $\mathcal{M}'$. This procedure relax the next projections, accelerating the convergence of the method. However, it also increases the geometric losses, as the smoothing caused by the Laplacian become more permanent. If the input p is set zero, then $\mathcal{M}_p$ is never replaced and all the projections are performed over the original mesh, achieving

Algorithm 6: Projection($\mathcal{M}'$,$\mathcal{M}_p$)

```
1  foreach v̄_j ∈ M' do
2      Nearest = getNearestVertices(v̄_j,M_p,5)
3      v^m = getFirstVertex(Nearest)
4      foreach v_i ∈ Nearest do
5          Triangles = getIncidentTriangles(v_i)
6          foreach f̄_i ∈ Triangles do
7              candidate = OrthogonalProjection(v̄_j, f̄_i)
8              if |v̄_j − candidate| < |v̄_j − v^m| then
9                  v^m = candidate
10             end if
11         end foreach
12     end foreach
13     v̄_j = v^m
14 end foreach
```

higher fidelity of geometry and reducing the convergence.

4.6 Post processing

For complex models, some edge lengths may not satisfy the constraining interval after the limit of n iterations is reached. For these cases, the algorithm proceeds to the minimization of Equation 4.4 without any constraints, using the conjugate gradient method (Press et al., 1992). This resort solve most problems. However, since there are no constraints about the displacement of the the vertices, it also increases the geometric distortions.

4.7 Dual computing

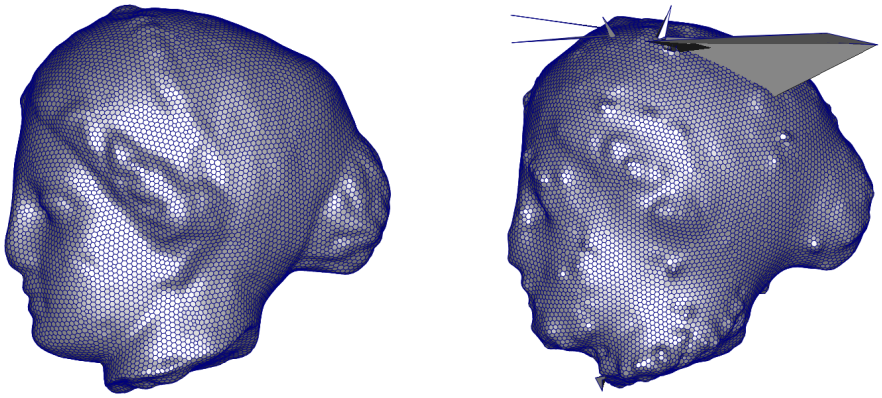
After the method successfully constrains the edge lengths to the desirable interval, or the limit of iterations is reached, we transform the resulting model in a trivalent mesh by computing its dual. The dual transformation is very simple and it is illustrated in Algorithm 7. It creates for each face $\overline{f}_j$ of the primal mesh a corresponding vertex v''_j on the dual mesh, in the barycenter of $\overline{f}_j$. Next, for each vertex $\overline{v}_j$ in the primal it creates a face f''_j, whose vertices v''_i are the ones corresponding to the faces $\overline{f}_i$ in which the vertex $\overline{v}_j$ participates.

Algorithm 7: Dual($\mathcal{M}$)

```
1  count = |F|
2  V'' = {v''_1, ..., v''_count}
3  count = |V|
4  F'' = {f''_1, ..., f''_count}
5  insertVertices(V'', M'')
6  insertFaces(F'', M'')
7  foreach f̄_j ∈ F ⊂ M do
8  |   v''_j = newVertexPosition(centroid(f̄_j))
9  end foreach
10 foreach v̄_j ∈ V ⊂ M do
11 |   foreach {f̄_i | v̄_j ∈ f̄_i} do
12 |   |   addVertex(v''_i, f''_j)
13 |   end foreach
14 end foreach
15 return M''
```

5 Experimental results

This chapter describes the experiments performed and discusses the obtained results. Different models were used for results generation. All the models were scaled, putting the average edge length to the desirable value of 1.5 angstrons.



(a) The output model from adaptive remeshing

(b) The model after 5000 iterations of molecular dynamics, showing the atoms that were repelled.

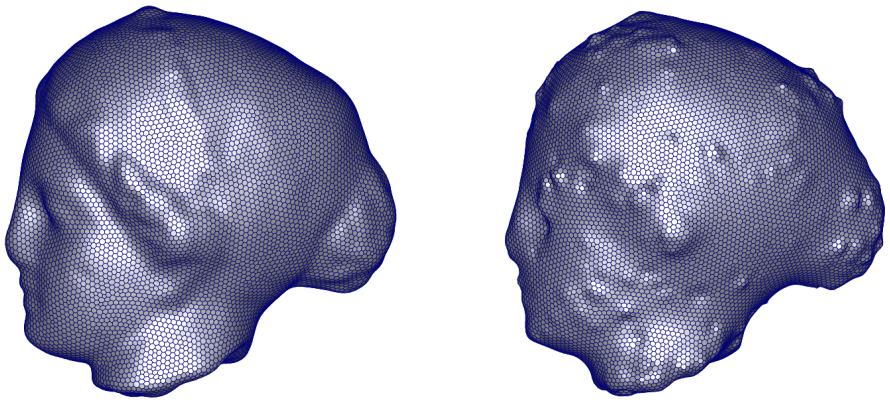
Figure 5.1: *Egea* model generated with parameters $p = 0$ and $s = 2$. The time step of each iteration of the simulation was 0.001 ps. For the visualization, the bonds depicted in the last picture correspond to the bonds formed in the first iteration, so we can see the displacement of the repelled atoms.

The tests were performed with varying values for the parameters p and s. All the tests assign the value 2 to the parameter k, since it gives the better results for the great majority of cases (Peçanha et al., 2013; Hauck et al., 2014, 2015). The maximum number of iterations n for these tests is 150.

The first experiments show the different results obtained with varying parameters of the adaptive remeshing for the model *egea*, and the relationship between the algorithm convergence and the stability of the simulations. Firstly, Figure 5.1 show the results achieved when the projection mesh is never replaced and $s = 2$, a scenario for which the remeshing algorithm did not converge. In the very early iterations, some atoms are

repelled from the molecule.

In contrast to this result, with the parameters $p = 10$ and $s = 5$, for which the method converge, the model maintain stability, as depicted in Figure 5.2. One may notice that the stable model presents several geometric distortions, once many knobs arise in the model. This effect is probably a consequence of the non-regularization of the angles formed by the edges incident to a same vertex, suggesting that the adaptive remeshing could be improved to consider this kind of constraining.



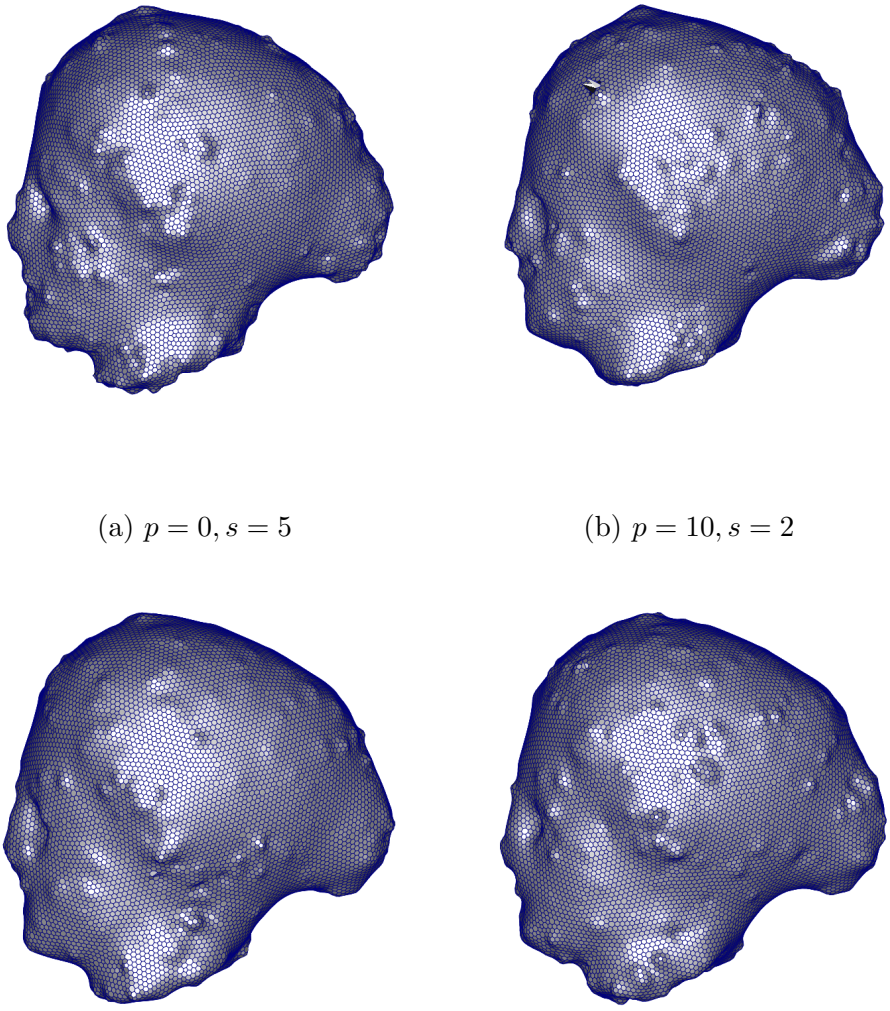
(a) The output model from adaptive remeshing

(b) The model after 5000 iterations of molecular dynamics

Figure 5.2: *Egea* model generated with parameters $p = 10$ and $s = 5$. The molecule presents geometric distortions, but maintain the main structure without losing any atoms.

The convergence of the adaptive remeshing is much slower in this work if compared to the version presented in (Hauck et al., 2015). This difference comes from the improvement in the valence optimizer step, which reduces the convergence in order to obtain meshes of better quality. Consequently, the impact of parameter p in the results was greatly increased. The same goes for the parameter s, since the efficacy of the Laplacian operator is reduced more quickly. For the *egea* model, the method did not converged for $s = 2$ in any case, and converged for $s = 5$ for all cases, presenting better results for tests with more replacements of the projection mesh. Figure 5.3 illustrates the results for different scenarios.

For different models, the method presents similar results. Figure 5.4 illustrates the efficacy of the method for molecules of different shapes.

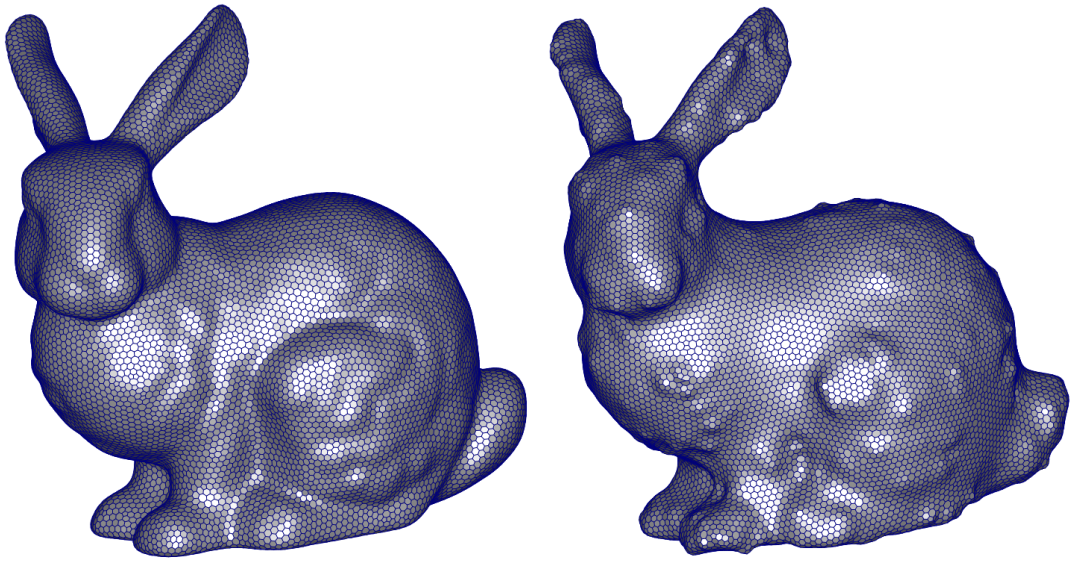


(a) $p = 0, s = 5$ (b) $p = 10, s = 2$

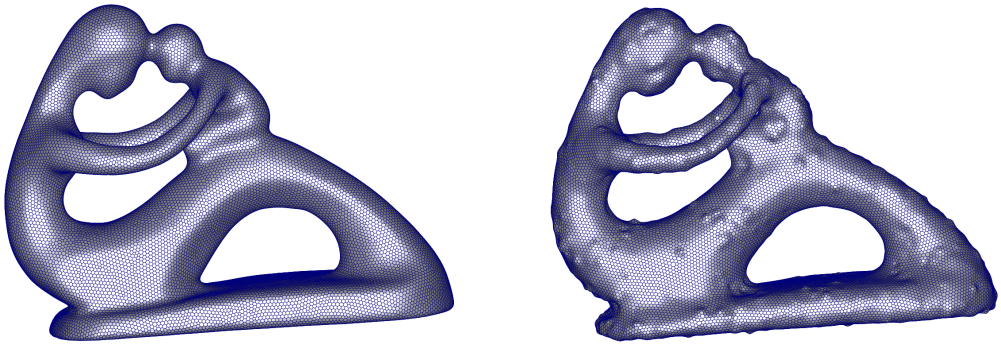
(c) $p = 25, s = 2$ (d) $p = 25, s = 5$

Figure 5.3: *Egea* model at the end of 5000 iterations of molecular dynamics simulations for different parameters.

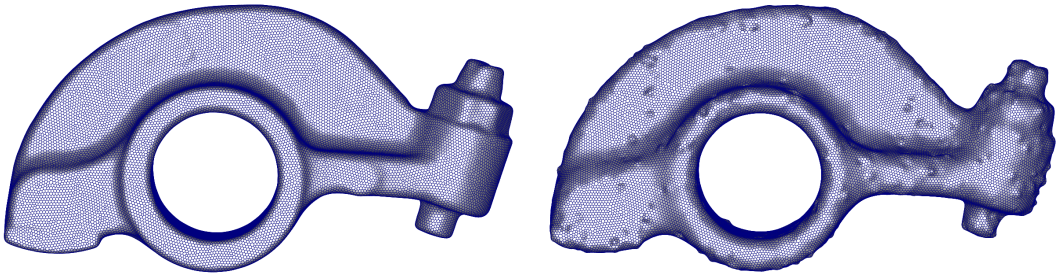
Visually, one may notice that the resulting models shrink before achieving stability, suggesting that the ideal average edge length should be lower than 1.5. In order to verify the best average length for simulation, an exhaustive test was performed with the *rockerarm* mesh. The model was submitted to 200000 iterations of molecular dynamics, until it became very stable. Following that, the average bond length for this *rockerarm* was calculated, resulting in the value 1.43 angstrons. Next, models were generated with an average edge length closer to this value. The models were constrained to the interval of 1.1 to 1.7 angstrons. This experiment failed, as all of these models submitted to simulation degenerated. This fact, in addition to the previous results, in which the models that



(a) *Bunny* model generated with $p = 25$ and $s = 5$. The model in the right was submitted to 3000 iterations of molecular dynamics.



(b) *Fertility* model generated with $p = 10$ and $s = 5$. The model in the right was submitted to 3000 iterations of molecular dynamics.



(c) *Rockerarm* model generated with $p = 25$ and $s = 5$. The model in the right was submitted to 3000 iterations of molecular dynamics.

Figure 5.4: Resulting molecules of different models. The method was capable of achieving very stable structures.

had *short* edges presented repulsion of some atoms, lead us to conclude that, despite the average edge length should be lower than 1.5, the lower bound for the interval should not

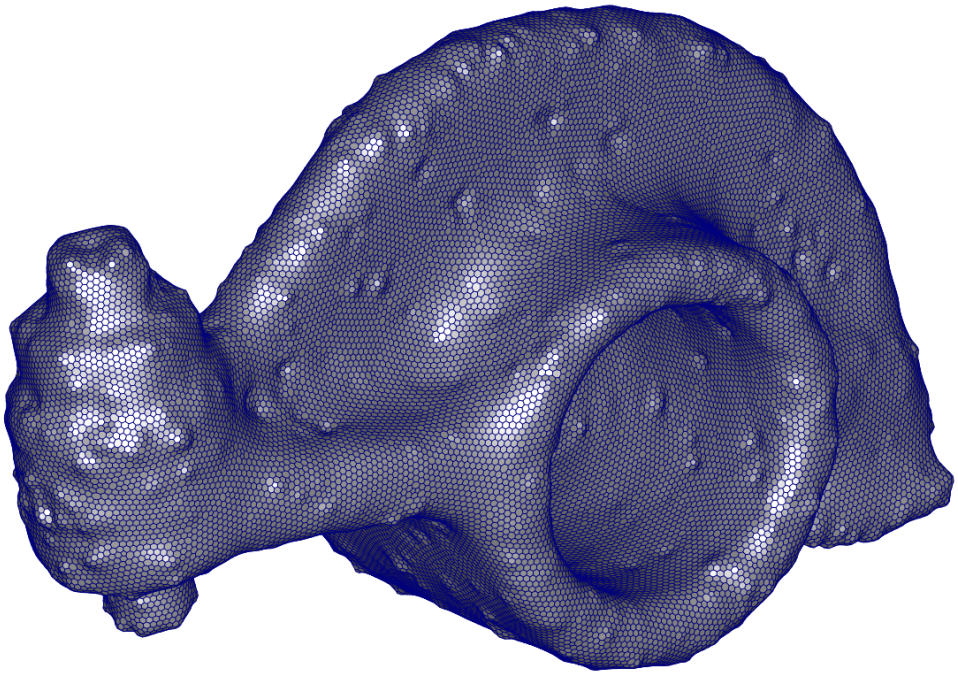


Figure 5.5: Rockerarm

be lower than 1.2. Since longer edges did not present the same problem (the model could stabilize even in the presence of some *long* edges), it seems like the standard deviation of lengths bellow the average should be lower than the standard deviation above the average. As an improvement of the remeshing process, this peculiarity could be taken into account.

Another aspect verified was the application of energy minimization approaches. The problem with these approaches is that they need a minimum of previous stability to work properly. As a validation test, we performed the energy minimization of the potential REBO2 with the conjugate gradient method (Press et al., 1992) for the models previously illustrated. The models degenerated in every case. However, if we want to make use of this tool for arbitrary models, the adaptive remeshing can be applied as an intermediate step. It is important to highlight that sometimes the output model of the adaptive remeshing does cannot be optimized either. However, if the model do not degenerate in the molecular dynamics, it is possible to let it achieve a certain stability in the simulation and then submit it to the energy minimization.

As an example, Figure 5.5 depicts the resulting *rockerarm* model optimized through a minimization of its REBO2 potential. The output *rockerarm* model from the adaptive remeshing did not converge, so we used the right model on Figure 5.4c as input

instead.

6 Conclusion

This work presents the description of an algorithm to remesh an arbitrary model into a nanocarbon structure, proposed by Hauck et al. (2015), and verify the quality of their results in an application scenario. It illustrates the behavior of the method for different models and parameters, exemplify its advantage over an energy minimization approach and illustrates the possibility of combining both strategies.

While not perfect, the method have presented good results, generating stable structures of many different shapes. However, it is still very sensible to variations in its parameters, specially the parameters s, that dictates the moment in which the global optimization strategy is replaced with the local optimization strategy, and p, that informs the rate of updating of the projection mesh.

As future works, many improvements for the method can be proposed. It could be interesting to make the method reactive, eliminating the need of parameter fitting. Moreover, the method should not optimize the edge lengths in a Gaussian distribution, since the simulations do not admit a standard deviation of interatomic bonds bellow the average as high as the deviation above the average. Finally, an optimization strategy could be proposed for the dual mesh, since that the optimal features of the primal mesh do not imply that the dual will present them.

Bibliography

Alliez, P.; Meyer, M. ; Desbrun, M. Interactive geometry remeshing. **ACM Trans. Graph.**, v.21, n.3, p. 347–354, Jul 2002.

Bommes, D.; Zimmer, H. ; Kobbelt, L. Mixed-integer quadrangulation. **ACM Trans. Graph.**, v.28, n.3, p. 77:1–77:10, Jul 2009.

Botsch, M.; Kobbelt, L.; Pauly, M.; Alliez, P. ; Lévy, B. **Polygon Mesh Processing**. A K Peters, 2010.

Brenner, D. W.; Shenderova, O. A.; Harrison, J. A.; Stuart, S. J.; Ni, B. ; Sinnott, S. B. A second-generation reactive empirical bond order (rebo) potential energy expression for hydrocarbons. **Journal of Physics: Condensed Matter**, v.14, n.4, p. 783, Jan. 2002.

Guillemin, V.; Pollack, A. **Differential topology**. Prentice-Hall, 1974.

Hamada, N.; Sawada, S.-I. ; Oshiyama, A. New one-dimensional conductors: Graphitic microtubules. **Phys. Rev. Lett.**, v.68, p. 1579–1581, Mar. 1992.

Hauck, J. V. d. S.; Silva, R. N.; Vieira, M. B. ; Silva, R. L. S. **Iterative remeshing for edge length interval constraining**. In: Computational Science and Its Applications–ICCSA 2014, p. 300–312. Springer, 2014.

Hauck, J. V. d. S.; Silva, R. N.; Vieira, M. B. ; Silva, R. L. S. Adaptive remeshing for edge length interval constraining. **Journal of Mobile Multimedia**, v.11, 2015.

Lászlo, I.; Rassat, A. The geometric structure of deformed nanotubes and the topological coordinates. **Journal of Chemical Information and Computer Sciences**, v.43, p. 519–524, Jan. 2003.

Lennard-Jones, J. E. Cohesion. **Proceedings of the Physical Society**, v.43, n.5, p. 461–482, Set. 1931.

Yong-Kui, L. The generation of straight lines on hexagonal grids. **Computer Graphics Forum**, v.12, p. 27–32, Mar. 1993.

Liu, Y.; Pan, H.; Snyder, J.; Wang, W. ; Guo, B. Computing self-supporting surfaces by regular triangulation. **ACM Trans. Graph.**, v.32, n.4, p. 92:1–92:10, Jul 2013.

Narain, R.; Samii, A. ; O'Brien, J. F. Adaptive anisotropic remeshing for cloth simulation. **ACM Transactions on Graphics (TOG)**, v.31, n.6, p. 152, 2012.

Pampanelli, P. C. P.; Peçanha, J. P.; Campo, A. M.; Vieira, M. B.; Lobosco, M. ; Dantas, S. O. **Rectangular hexagonal mesh generation for parametric modeling**. In: SIBGRAPI Conference on Graphics Patterns and Images, volume 0, p. 120–125, 2009.

Peçanha, J. P.; Souza Filho, J. L.; Vieira, M. B.; Lobosco, M. ; Dantas, S. O. **Iterative method for edge length equalization**. In: International Conference on Computational Science, p. 481–490, Barcelona,Spain, 2013.

Press, W. H.; Teukolsky, S. A.; Vetterling, W. T. ; Flannery, B. P. **Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing**. New York, NY, USA: Cambridge University Press, 1992.

Surazhsky, V.; Gotsman, C. **Explicit surface remeshing**. In: Proceedings of Eurographics Symposium on Geometry Processing, p. 17–28, Aachen, Germany, Jun 2003.

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, p. 351–358, New York, NY, USA, 1995. ACM.

Taubin, G. Dual mesh resampling. **Graphical Models**, v.64, n.2, p. 94 – 113, 2002.

Wang, Z. **Reactive empirical bond-order (rebo) potential**. Jan. 2006.

Zsoldos, I.; Kakuk, G.; Réti, T. ; Szasz, A. Geometric construction of carbon nanotube junctions. **Modelling and Simulation in Materials Science and Engineering**, v.12, n.6, p. 1251, 2004.