

---

# Perturbing Slivers in 3D Delaunay Meshes

Jane Tournois<sup>1</sup>, Rahul Srinivasan<sup>2</sup>, and Pierre Alliez<sup>1</sup>

INRIA Sophia Antipolis - Méditerranée, France

<sup>1</sup>firstname.lastname@sophia.inria.fr

<sup>2</sup>firstname.lastname@iitb.ac.in

**Abstract.** Isotropic tetrahedron meshes generated by Delaunay refinement algorithms are known to contain a majority of well-shaped tetrahedra, as well as spurious sliver tetrahedra. As the slivers hamper stability of numerical simulations we aim at removing them while keeping the triangulation Delaunay for simplicity. The solution which explicitly perturbs the slivers through random vertex relocation and Delaunay connectivity update is very effective but slow. In this paper we present a perturbation algorithm which favors deterministic over random perturbation. The added value is an improved efficiency and effectiveness. Our experimental study applies the proposed algorithm to meshes obtained by Delaunay refinement as well as to carefully optimized meshes.

## 1 Introduction

Delaunay refinement algorithms [9, 26, 23, 25] have been extensively studied in the literature. They are amenable to analysis, and hence are reliable algorithms. In addition, the robust implementations of Delaunay triangulations which are now available greatly facilitate the implementation of Delaunay-based mesh refinement algorithms. However, most Delaunay refinement algorithms fail at removing all badly-shaped tetrahedra, and a special class of almost-flat tetrahedra (so-called slivers) may remain in the triangulation. These slivers, with dihedral angles close to 0 and to  $\pi$ , are problematic for many numerical simulations.

### 1.1 Slivers

Many finite element methods require discretizing a domain into a set of tetrahedra. These applications require more than just a triangulation of the domain for simulation and rendering. The accuracy and the convergence of these methods depend on the size and shape of the elements apart from the fact that the mesh should conform to the domain boundary [28]. Both the

bad quality and the large number of the mesh elements can negatively affect the execution of a simulation. It is required that all elements of the mesh are well-shaped as the accuracy of the simulations and computations can be compromised by the presence of even a single badly shaped element. In general it is desirable to bound the smallest dihedral angle in the mesh, from below. The Delaunay refinement technique guarantees a bound on the radius-edge ratio of all mesh elements, which is the ratio of the circumradius to the shortest edge length of a tetrahedron. Although in 2D this translates into a lower bound on the minimum angle in the mesh, in 3D it does not: a bound on the radius-edge ratio is not equivalent to a bound on the smallest dihedral angle.

The only bad elements that remain after Delaunay refinement are slivers. A sliver tetrahedron is formed by almost evenly placing its 4 vertices near the equator of its circumsphere (see Figure 1), and has a bounded radius-edge ratio. In such a sliver the smallest dihedral angle can be very close to  $0^\circ$ , and a numerical simulation may be far from accurate in the presence of slivers.

## 1.2 Tetrahedron Quality

Several tetrahedron quality criteria have been defined and used in the literature depending on the application. The radius edge ratio  $\rho$  of a simplex is defined as the ratio of its circumradius to the length of the shortest edge. This measure, which is minimal for the regular tetrahedron, unfortunately cannot detect slivers, though it is used in Delaunay refinement algorithms to define bad simplices. The radius ratio, defined as the ratio of the inradius (insphere radius) to the circumradius (circumsphere radius), is another popular measure of tetrahedron quality. It is desired to ensure that radius ratio of all tetrahedra are bounded from below by a constant.

Another criterion for mesh generation is the minimum dihedral angle  $\theta_{min}$ . It can be shown that a lower bound on the radius ratio is equivalent to a lower bound on the minimum dihedral angle. In the sequel we choose this measure to evaluate the mesh quality as it is more intuitive and geometrically meaningful than, e.g., the radius ratio, which combines the six dihedral angles of a tetrahedron.

Consider an arbitrary tetrahedron  $\tau$  with triangular faces  $T_1, T_2, T_3, T_4$ . Let the areas of these triangles be denoted by  $S_1, S_2, S_3, S_4$  respectively, the dihedral angle between  $T_i$  and  $T_j$  by  $\theta_{ij}$  and the length of the edge shared by  $T_i$  and  $T_j$  by  $l_{ij}$ . The volume  $V$  of  $\tau$  is given by

$$V = \frac{2}{3l_{ij}} S_i S_j \sin \theta_{ij} \quad \text{for } i \neq j \text{ in } \{1, 2, 3, 4\}. \quad (1)$$

Let  $r_C, r_I$  be the circumradius and inradius of  $\tau$  and  $r_i$  be the circumradius of  $T_i$  for  $i$  in  $\{1, 2, 3, 4\}$ . We know that for any tetrahedron,  $r_i \leq r_C$ . This gives  $S_i \leq \pi r_i^2 \leq \pi r_C^2$ , and we also have a bound on the volume  $V \geq \frac{4}{3} \pi r_I^3$ .

Using Equation 1, for  $i \neq j$  we have

$$\frac{4}{3}\pi r_I^3 \leq \frac{2}{3l_{ij}}S_i S_j \sin\theta_{ij} \leq \frac{2}{3l_{ij}}\pi^2 r_c^4 \sin\theta_{ij},$$

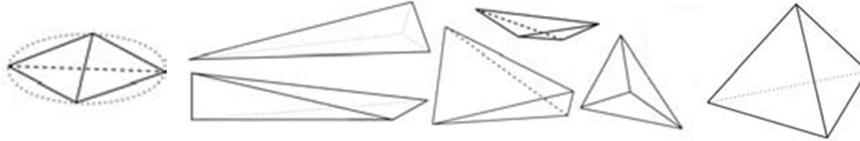
and we get

$$\sin\theta_{ij} \geq \frac{2}{\pi} \cdot \frac{r_I^3 l_{ij}}{r_c^3 r_C} \geq \frac{2}{\pi} \cdot \frac{a_0^3}{\rho_0},$$

where  $a_0$  is the radius-radius ratio and  $\rho_0$  is the radius-edge ratio. Finally,

$$\theta_{ij} \geq \sin^{-1}\left(\frac{2}{\pi} \cdot \frac{a_0^3}{\rho_0}\right).$$

Li [21] uses a different parameter of tetrahedron quality to define a sliver. Denote the volume of tetrahedron  $pqrs$  by  $V$  and its shortest edge length by  $l$ . The volume per cube of shortest edge length ( $\sigma = \frac{V}{l^3}$ ) is used as a measure of the shape quality along with the radius-edge ratio  $\rho$ , or on its own [7]. According to Li, a tetrahedron  $pqrs$  is called sliver if  $\rho(pqrs) \leq \rho_0$  and  $\sigma(pqrs) \leq \sigma_0$ , where  $\rho_0$  and  $\sigma_0$  are constant.



**Fig. 1.** Tetrahedron shapes. A sliver (left) has its four vertices close to a circle, four very small dihedral angles (close to  $0^\circ$ ), and two very large (close to  $180^\circ$ ). A regular tetrahedron (right) is well shaped and has its dihedral angles close to  $70.5^\circ$ . Each of the other tetrahedra (middle) present a different type of degeneracy.

### 1.3 Previous Work

The problem of removing slivers from a 3D Delaunay mesh has received some attention over the last decade. Delaunay refinement gets so close to providing a perfect output that removing the leftover slivers is generally performed as a post-processing step that is worth it. Previous work on removing and avoiding the creation of slivers can be classified into three parts: The Delaunay-based methods, the weighted Delaunay-based methods, and the non-Delaunay methods. For each part, post-processing steps and complete mesh generation algorithms can be studied. This paper focuses on a post-processing step, devised to take as input a Delaunay mesh and to improve its quality in terms of dihedral angles.

## Delaunay-based

### *Vertex Perturbation*

Li [21, 14] proposes to explicitly perturb the vertices incident to a sliver in an almost-good mesh, by locally relocating them so as to remove the incident slivers. The idea is based on the fact that, for any triangle  $qrs$ , the region of locations of the vertex  $p$  such that the tetrahedron  $pqrs$  is a sliver, is very small. Moving the point  $p$  out of this region ensures that the tetrahedron is not a sliver anymore, or has disappeared once the Delaunay connectivity is updated. This is achieved by moving the point  $p$  to a new location inside a small ball centered at  $p$ , whose radius is proportional to the distance from  $p$  to its the nearest neighbor. The author shows that for certain values of the involved parameters, there always exists some points in this ball which are outside all regions that form slivers with nearby triangles. Li uses the union graph concept to avoid circular dependencies on vertex perturbations. The following theorem [21] proves the existence of such a point that makes the mesh locally sliver-free.

**Theorem 1 (Sliver theorem).** *If every simplex in a Delaunay triangulation has radius-edge ratio of at least  $\rho_0$ , then there is a constant  $\sigma_0 > 0$  and a very mild perturbation  $S'$  with  $\sigma(\tau) \geq \sigma_0$  for each tetrahedron  $\tau$  in the perturbed triangulation.*

Based on this theorem, Li proposes an algorithm that applies mild random perturbations to the mesh until one which removes slivers is found. One drawback of the above result is the pessimistic theoretical estimate of the bounds on the involved parameters. These bounds are either too small or too large to have any significance. In practice, though this technique is very effective, when targeting a large bound on the minimum dihedral angle (e.g.  $15^\circ$ ), the average number of trials of random perturbations required is very large. In our experiments, it is not rare to apply hundreds of random perturbation trials on a single vertex before succeeding in removing a sliver. This number is not surprising when seeking a high minimum dihedral angle such as  $15^\circ$  since the corresponding tetrahedron is not a real sliver anymore. However the fact that the perturbation succeeds even for a high minimum dihedral angle is at the core of our motivation. Finally, the fact that this method always maintains the mesh as a true Delaunay triangulation makes it both robust and practical.

### *Sliver-free mesh generation*

Some mesh generation algorithms are designed to avoid creating slivers. For example, Delaunay refinement can be modified by choosing a new type of Steiner point which does not create any sliver [22, 23, 24]. As an example, Chew's algorithm [9] inserts Steiner points in a randomized manner, to

avoid the creation of slivers. This method has a theoretical lower bound of  $\arcsin 1/4 \approx 14.5^\circ$  on the angles of the triangular faces of the mesh.

## Weighted Delaunay-based

### *Sliver exudation*

First described by Cheng et al. [7], sliver exudation is a technique based on turning a Delaunay triangulation into a weighted Delaunay triangulation [3], devised to trigger flips so as to increase the minimal angle. Edelsbrunner and Guoy [13] provide an experimental study of sliver exudation, and show that it works pretty well in practice as a post-treatment applied to a triangulation obtained by Delaunay refinement [25]. The main strategy of the algorithm consists of assigning a weight to each vertex so that the weighted Delaunay triangulation is free of any slivers after connectivity updates, without any changes over the vertex locations. This method successfully increases all dihedral angles above  $5^\circ$  in the best configuration (see Section 3), but as admitted in [13], the theoretical bound on the dihedral angle is too small to be of any practical significance.

Beside being not strictly Delaunay anymore, the main disadvantage of sliver exudation is that the process often ends with leftover slivers near the boundary [13]. This is mainly due to the fact that sliver exudation is not allowed to modify the topology of the boundary of the mesh. Hence, weight assignments close to the boundary are constrained and do not always manage to remove the slivers.

### *Complete algorithm*

Cheng and Dey [6] propose a complete Delaunay refinement algorithm, combined with the sliver exudation technique. This type of weighted-Delaunay algorithm is also used to handle input domains containing sharp creases subtending small angles [8].

## Non-Delaunay

### *Local combinatorial operations*

Though a Delaunay-refined triangulation is known to have nice properties on its angles in 2D [12], there is no theoretical guarantee on the dihedral angles in 3D. One valid choice consists of leaving the Delaunay framework by flipping some well-chosen simplices [27, 18], either as a post-processing step to the meshing process [19], or during the whole process [10]. As long as the triangulation remains valid, flips can be performed on its edges and facets. Joe gives a description of all possible flips [17] that can be made in

a triangulation, and a triangulation improvement algorithm through these flips. Although each improvement in this algorithm is local, the complete algorithm succeeds in improving the overall quality of the mesh.

Dealing with non-Delaunay meshes can also be combined with optimization steps, such as Laplacian smoothing [15], which relocates each vertex to a new location computed as an average of the incident vertex positions. Laplacian smoothing can be applied to any valid triangulation.

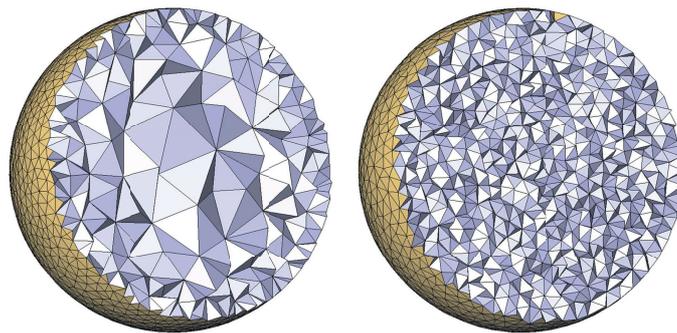
#### *Complete algorithm*

Some other types of triangulations, such as for example max-min solid angle triangulations [16] can be computed to improve the solid angles as compared to that in a Delaunay triangulation. This method generates a set of well-distributed points in the input polyhedral domain and first computes a Delaunay triangulation of these vertices. Then, local combinatorial transformations are applied to satisfy the local max-min angle criterion. These local transformations can in fact be applied to any triangulation as a post-processing step.

Instead of performing local improvements through flips in a Delaunay mesh, Labelle and Shewchuck [20] propose a fast lattice refinement technique which constructs a triangulation based on two nested regular or adapted grids. In its graded version this algorithm provides a theoretical bound on the dihedral angles which is much more practical than provided by other algorithms.

#### **1.4 Contribution**

We present a sliver removal algorithm inspired by Li's random perturbation algorithm [21]. Our algorithm is made more deterministic by choosing a favored perturbation direction for each vertex incident to one or more slivers,



**Fig. 2.** Sphere. (Left) Graded mesh with 3195 vertices, output angles are in  $[23.5; 142.5]$ . (Right) Uniform mesh with 7041 vertices, output angles are in  $[30.02; 138.03]$ .

*before* resorting to Li’s random perturbation if the favored perturbation fails at removing the incident slivers. Our experiments show that the chosen deterministic directions are sufficient to remove more than 80% of the slivers of a mesh, leading to shorter computational times. In addition, our approach reaches higher minimum dihedral angles in practice.

## 2 Algorithm

We describe a sliver perturbation algorithm which improves in a hill-climbing manner the dihedral angles of an input isotropic Delaunay mesh. This algorithm can be used as a post-processing step after refinement or optimization.

To improve the dihedral angles of the mesh tetrahedra, the rationale behind our approach is as follows: each vertex  $v$  incident to at least one sliver is repeatedly relocated through a perturbation vector  $\mathbf{p}_v$  such that when  $v$  moves to  $v + \mathbf{p}_v$ , the incident slivers get flipped. More specifically, the chosen direction for  $\mathbf{p}_v$  is not devised to improve the shape of the slivers, but rather to worsen them instead, so that they get flipped. Two directions are favored by the algorithm: the incident squared circumradius gradient ascent (see Section 2.1) and the sliver volume gradient descent (see Section 2.2). The length of the perturbation vector is heuristically chosen as a fraction (usually between 0.05 and 0.2) of the minimum incident edge length. If neither of these two perturbation vectors succeed in flipping a sliver we resort to random perturbations (see Section 2.3). If the whole sequence does not improve the local minimum dihedral angle then we restore the vertex to its original location before perturbation.

By construction, our combined perturbation algorithm is hill-climbing in the sense that the dihedral angles in the output mesh must be higher than the ones in the input mesh. The theoretical proofs of Li’s method [21] concerning random perturbation apply to this combined perturbation method as we resort to it in case of failure of the deterministic perturbation.

When more than one sliver is incident to a vertex  $v$ , all perturbation vectors must be compatible (i.e., pushing in a similar direction) to be effective. In our current algorithm, a set of perturbation vectors are said to be compatible if all their pairwise dot products are positive. The perturbation vector  $\mathbf{p}_v$  is then set to be the average of these vectors. When not compatible,  $v$  is perturbed only using random perturbations. The algorithm relies on a modifiable priority queue, built in a way such that vertices incident to fewer slivers are processed first. Hence, any “chain” of slivers (set of slivers sharing at least one vertex) is treated starting from its endpoints thereby minimizing the need to process vertices incident to more than one sliver.

---

**Algorithm 1.** Sliver perturbation

---

**Input:**  $T$ : a Delaunay triangulation, $\alpha$ : the angle bound defining slivers, and $N_{max}$ : the maximum number of random trials, or gradient steps.Let  $\mathcal{P}$  be a priority queue of Delaunay vertices.Fill  $\mathcal{P}$  with vertices incident to slivers,Compute perturbation vector  $\mathbf{p}_v$  for each vertex  $v$  in  $\mathcal{P}$ ,**while**  $\mathcal{P}$  non-empty **do**  Pop  $v$  from  $\mathcal{P}$ ,   $v' \leftarrow v$ ,  **while** relocating  $v$  to  $v'$  would not trigger a combinatorial change,  and  $\#loops < N_{max}$  **do**     $v' \leftarrow v' + \mathbf{p}_v$ ,    **if**  $\mathbf{p}_v$  is random, **then**      compute a new  $\mathbf{p}_v$ ,       $v' \leftarrow v$ .    **end if**  **end while**  Conditionally relocate  $v$  to  $v'$ .  **if**  $v$  is still incident to slivers and  $\mathbf{p}_v$  is not random, **then**

Compute a new perturbation vector (another type, if possible),

    and re-insert  $v$  into  $\mathcal{P}$ .  **end if**  Insert all vertices affected by relocation into  $\mathcal{P}$ ,

with their new perturbation vector.

**end while**

---

Note that each vertex relocation is conditional, as we want our algorithm to be hill-climbing in terms of dihedral angles. We need to check that the minimum dihedral angle of the triangulation does not decrease, and that the topology of the boundary is not affected. Otherwise, the relocation is canceled.

Each time a vertex is effectively relocated, the priority queue is updated. Moving  $v$  to  $v'$  in a Delaunay mesh makes combinatorial changes (and, hence, changes on incident dihedral angles) on the vertices incident to  $v$  before its removal, and the ones incident to  $v'$  after its insertion. We first compute the perturbations associated with all these vertices, and insert them into the priority queue.

The order in which the vertices are processed in the priority queue is related to the vertex type. Interior vertices are processed first, since they are more likely to be easily perturbable than boundary vertices. The boundary vertices are constrained to be located on the boundary, and their move must not break the topology of the mesh. These constraints make them more difficult to perturb. The other ordering criteria are discussed in Section 3.

## 2.1 Circumsphere Radius

In an almost-good isotropic tetrahedron mesh, the distribution of the mesh vertices is locally uniform. Hence, perturbing the vertex locations so as to make the radius of the sliver's circumsphere explode triggers many flips as the empty circumsphere property must hold after Delaunay connectivity update.

Let  $\tau$  be the sliver, and  $\{p_i\}_{i=0,1,2,3}$  its vertices. Without loss of generality, and since the sequel remains true by translation, we can assume that  $p_0 = 0_{\mathbb{R}^3}$ . We also assume that this vertex is fixed. Let  $c$  be  $\tau$ 's circumcenter. We have  $\|c\| = R$  the radius of  $\tau$ 's circumsphere. Then,  $\nabla R^2 = \nabla \|c\|^2$ . We aim at computing  $\nabla R^2$ .

Let  $p_i = (x_i, y_i, z_i)$  for  $i$  in  $\{1, 2, 3\}$  be  $\tau$ 's vertices, with  $p_0 = 0_{\mathbb{R}^3}$ . Also, let  $p_i^2$  be  $(x_i^2 + y_i^2 + z_i^2)$ . The center  $c$  of the circumsphere of  $\tau$  is given by

$$c = \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \begin{pmatrix} \frac{D_x}{2a} \\ \frac{D_y}{2a} \\ \frac{D_z}{2a} \end{pmatrix}, \text{ where}$$

$$a = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}, D_x = - \begin{vmatrix} p_1^2 & y_1 & z_1 \\ p_2^2 & y_2 & z_2 \\ p_3^2 & y_3 & z_3 \end{vmatrix}, D_y = + \begin{vmatrix} p_1^2 & x_1 & z_1 \\ p_2^2 & x_2 & z_2 \\ p_3^2 & x_3 & z_3 \end{vmatrix}, \text{ and } D_z = - \begin{vmatrix} p_1^2 & x_1 & y_1 \\ p_2^2 & x_2 & y_2 \\ p_3^2 & x_3 & y_3 \end{vmatrix}.$$

$$\text{Thus we have, } \nabla_{p_1} \|c\|^2 = \begin{pmatrix} \frac{\partial \|c\|^2}{\partial x_1} \\ \frac{\partial \|c\|^2}{\partial y_1} \\ \frac{\partial \|c\|^2}{\partial z_1} \end{pmatrix} \text{ with}$$

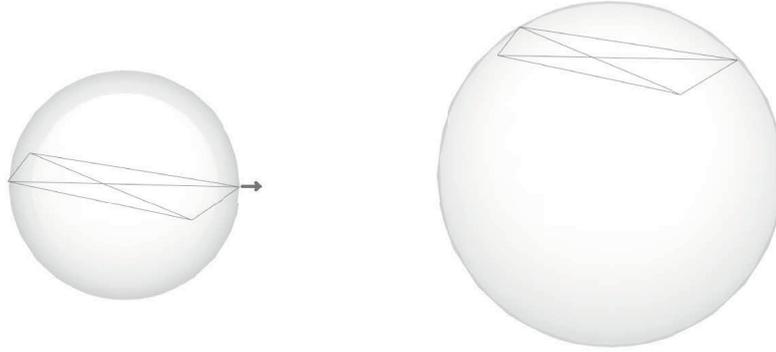
$$\begin{aligned} \frac{\partial \|c\|^2}{\partial x_1} &= \frac{\partial}{\partial x_1} \left( \frac{D_x^2 + D_y^2 + D_z^2}{4a^2} \right) \\ &= \frac{1}{2a^3} \cdot \left( a \cdot (D_x \cdot \frac{\partial D_x}{\partial x_1} + D_y \cdot \frac{\partial D_y}{\partial x_1} + D_z \cdot \frac{\partial D_z}{\partial x_1}) - \frac{\partial a}{\partial x_1} \cdot (D_x^2 + D_y^2 + D_z^2) \right), \\ \frac{\partial \|c\|^2}{\partial y_1} &= \frac{1}{2a^3} \cdot \left( a \cdot (D_x \cdot \frac{\partial D_x}{\partial y_1} + D_y \cdot \frac{\partial D_y}{\partial y_1} + D_z \cdot \frac{\partial D_z}{\partial y_1}) - \frac{\partial a}{\partial y_1} \cdot (D_x^2 + D_y^2 + D_z^2) \right), \\ \frac{\partial \|c\|^2}{\partial z_1} &= \frac{1}{2a^3} \cdot \left( a \cdot (D_x \cdot \frac{\partial D_x}{\partial z_1} + D_y \cdot \frac{\partial D_y}{\partial z_1} + D_z \cdot \frac{\partial D_z}{\partial z_1}) - \frac{\partial a}{\partial z_1} \cdot (D_x^2 + D_y^2 + D_z^2) \right). \end{aligned}$$

and

$$\begin{aligned} \nabla_{p_1} a &= \begin{pmatrix} \frac{\partial a}{\partial x_1} \\ \frac{\partial a}{\partial y_1} \\ \frac{\partial a}{\partial z_1} \end{pmatrix} = \begin{pmatrix} y_2 z_3 - y_3 z_2 \\ -(x_2 z_3 - x_3 z_2) \\ x_2 y_3 - x_3 y_2 \end{pmatrix}, \\ \nabla_{p_1} D_x &= \begin{pmatrix} \frac{\partial D_x}{\partial x_1} \\ \frac{\partial D_x}{\partial y_1} \\ \frac{\partial D_x}{\partial z_1} \end{pmatrix} = \begin{pmatrix} -2x_1 \frac{\partial a}{\partial x_1} \\ -2y_1 \frac{\partial a}{\partial x_1} + p_2^2 z_3 - p_3^2 z_2 \\ -2z_1 \frac{\partial a}{\partial x_1} - p_2^2 y_3 + p_3^2 y_2 \end{pmatrix}, \\ \nabla_{p_1} D_y &= \begin{pmatrix} \frac{\partial D_y}{\partial x_1} \\ \frac{\partial D_y}{\partial y_1} \\ \frac{\partial D_y}{\partial z_1} \end{pmatrix} = \begin{pmatrix} -2x_1 \frac{\partial a}{\partial y_1} - p_2^2 z_3 + p_3^2 z_2 \\ -2y_1 \frac{\partial a}{\partial y_1} \\ -2z_1 \frac{\partial a}{\partial y_1} + p_2^2 x_3 - p_3^2 x_2 \end{pmatrix}, \end{aligned}$$

$$\nabla_{p_1} D_z = \begin{pmatrix} \frac{\partial D_z}{\partial x_1} \\ \frac{\partial D_z}{\partial y_1} \\ \frac{\partial D_z}{\partial z_1} \end{pmatrix} = \begin{pmatrix} -2x_1 \frac{\partial a}{\partial z_1} + p_2^2 y_3 - p_3^2 y_2 \\ -2y_1 \frac{\partial a}{\partial z_1} - p_2^2 x_3 + p_3^2 x_2 \\ -2z_1 \frac{\partial a}{\partial z_1} \end{pmatrix}.$$

Following a gradient ascent scheme, the vertex position  $p_i$  evolves this way:  $p_i^{next} = p_i + \epsilon \nabla p_i R_\tau^2 / \|\nabla p_i R_\tau^2\|$ , where the step length  $\epsilon$  is taken as a fraction of the minimum incident edge length to  $p_i$ . A relocation is performed only if the new minimal dihedral angle in the tetrahedra impacted by the relocation is not smaller than it was before relocation. As shown by Figure 3, the squared radius of  $\tau$ 's circumsphere increases very fast for a small perturbation of one of its vertices' positions. The circumsphere, now huge, most probably includes other mesh vertices, which triggers a flip to maintain the empty sphere Delaunay property.



**Fig. 3.** Circumsphere of a sliver. Before perturbation (left), the sliver is close to the equatorial plane of its circumsphere. A very mild perturbation of one of the sliver vertices (right) makes its circumradius increase considerably.

### 2.2 Volume

One of the main characteristics of a sliver is that its volume is strictly positive albeit small with respect to its smallest edge length, and possibly arbitrarily small. This property can be exploited in order to apply a perturbation devised to generate a sliver with negative volume and hence to trigger a combinatorial change.

Let  $\{p_i\}_{i=1,2,3}$  be the three fixed points of  $\tau$ , and  $p_0$  the vertex to be perturbed. The volume of  $\tau$  is

$$V_\tau = \frac{1}{6} \begin{vmatrix} x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix}.$$

Then, we get the volume gradient:

$$\nabla p_0 V_\tau = \frac{1}{6} \begin{pmatrix} y_2 z_3 + y_1(z_2 - z_3) - y_3 z_2 - z_1(y_2 - y_3) \\ -x_2 z_3 - x_1(z_2 - z_3) + x_3 z_2 + z_1(x_2 - x_3) \\ x_2 y_3 + x_1(y_2 - y_3) - x_3 y_2 - y_1(x_2 - x_3) \end{pmatrix}.$$

Following a gradient descent scheme, the vertex position  $p_i$  evolves this way:  $p_i^{next} = p_i - \epsilon \nabla p_i V_\tau / \|\nabla p_i V_\tau\|$ , where the step length  $\epsilon$  is taken as a fraction of the minimum incident edge length to  $p_i$ . A relocation is performed only if the new minimal dihedral angle of the tetrahedra impacted by the relocation is not smaller than it was before relocation. A negative tetrahedron volume triggers a flip to maintain a valid Delaunay triangulation.

### 2.3 Random Perturbation

When both  $\nabla V$  and  $\nabla R^2$  fail at flipping the considered slivers by vertex perturbation, we use a random perturbation based on Li's approach [21]. A perturbation satisfying three conditions (flip sliver, improve minimum dihedral angles, preserve restricted Delaunay triangulation) is searched for randomly inside a sphere centered at  $v$ . In accordance with Li's algorithm, the magnitude of the perturbation vector is set to fraction of the minimum incident edge length.

## 3 Experiments and Results

The algorithm presented has been implemented with the 3D Delaunay triangulation of the *Computational Geometry Algorithms Library* [1]. Our implementation of Li's random perturbation algorithm is based upon Algorithm 1, with one single perturbation type: the random one, described in Section 2.3. For each of the following experiments we set 100 trials of random perturbations (in our combined version as well as in the purely random algorithm).

The order in which the vertices are processed in the priority queue has been chosen empirically as a result of many experiments. Interior vertices are processed first, with priority over boundary vertices. Boundary vertices are constrained so as to remain on the domain boundary and their relocation is invalid if they modify the local restricted triangulation. This makes boundary vertices more difficult to perturb than interior vertices. The second order criterion is the number of incident slivers to the processed vertex. The idea behind this choice is that a *chain* of slivers (several incident slivers) is more difficult to perturb than an isolated sliver as the directions of gradients may not be compatible. However, if the endpoints of the chain are successfully perturbed, we ideally would not have to deal with vertices incident to more than one sliver. Thirdly, the vertex incident to a smaller dihedral angle is processed first, as our first goal is to remove the worst tetrahedra.

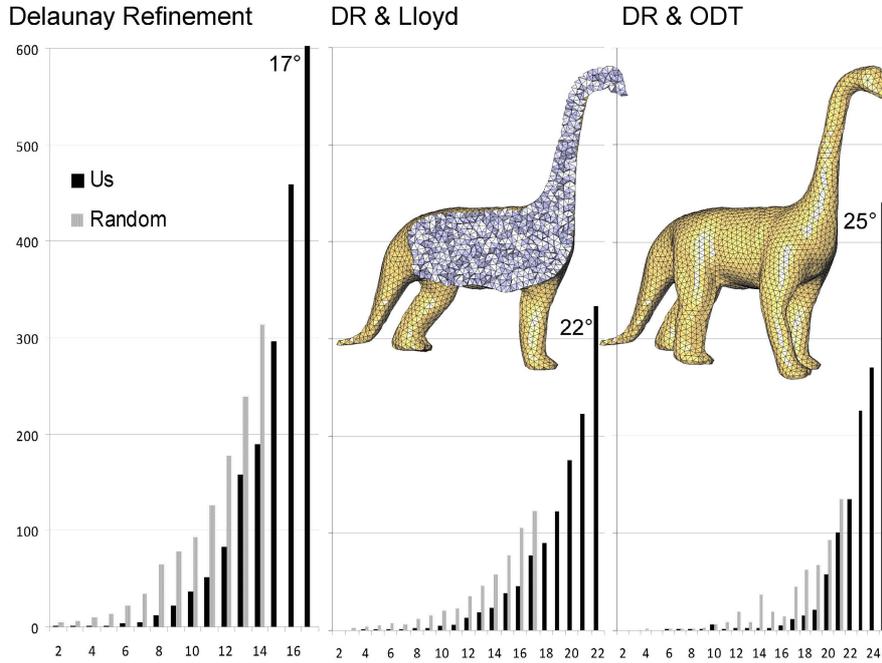
In our experiments, the  $\nabla R^2$  direction turns out to be more effective than  $\nabla V$  at perturbing a sliver. On average, this perturbation is responsible for

about 80% of all sliver flips. The  $\nabla V$  perturbation accounts for about 15% of the flips while the random perturbation counts for the remaining 5%. The priority given to  $\nabla R^2$  over  $\nabla V$  and random while picking the perturbation vector can be blamed for distorting these statistics, but we have chosen this order because it turns out to be the most effective. Giving priority to  $\nabla V$  results in an overall slowdown. Random perturbation always remains the last resort in the combined perturbation algorithm as the deterministic directions are favored.

The following experiments show what our combined algorithm can achieve on meshes generated by Delaunay refinement alone and on some meshes which have been optimized after refinement. A mesh optimization algorithm is in general devised to improve the mesh quality [2] while simpler algorithms aim at evenly distributing the vertices in accordance to a given mesh sizing function. Note that a mesh with well-spaced vertices does not mean an absence of slivers inside the mesh [29], and hence sliver removal is still required. The mesh optimization schemes used in our experiments are the centroidal Voronoi tessellation [11] using the Lloyd iteration, and the Optimal Delaunay triangulation (ODT for short) [5]. Both of these optimization methods have been implemented in a way that respects the local density of the mesh. It is important to not modify the density of a graded mesh, and to not decrease its quality.

Figures 4 and 5 provide the computation times and the best minimum dihedral angles obtained in our experiments. The same experiment has been carried out on many other models (not shown), giving similar results. Figures 4 and 5 emphasize that, for the same definition of a sliver (in terms of smallest dihedral angle), the combined algorithm is faster in removing all slivers by explicit perturbation compared to using Li's random perturbation alone. Moreover the combined algorithm reaches higher minimum dihedral angles.

The algorithm obtains fairly high minimum dihedral angles when the input is a mesh obtained by Delaunay refinement. Figures 4 and 5 illustrate that when the mesh is optimized prior to perturbation, the time taken for the algorithm to succeed in removing all slivers is shorter and that it can reach a higher minimum dihedral angle. As shown by histograms of Figure 4, the algorithm takes 611 seconds to perturb the mesh obtained after Delaunay refinement so that no dihedral angle is below  $17^\circ$ . If the same mesh is optimized prior to perturbations, the time taken goes down to 76 seconds for Lloyd and even further down to 11 seconds for ODT. Overall the same histograms show that a mesh optimized by ODT is easier to perturb and can reach a higher minimum angle ( $25^\circ$ ) than a mesh optimized by Lloyd ( $21^\circ$ ). However, optimization can be costly. The optimizations performed on Figure 4 meshes before applying perturbation took about 200 seconds. In spite of this additional cost, the combined perturbation algorithm remains more efficient than the random one. The same comments apply to Figure 5. The gradation of the mesh in Figure 5, along with the numerous high curvature



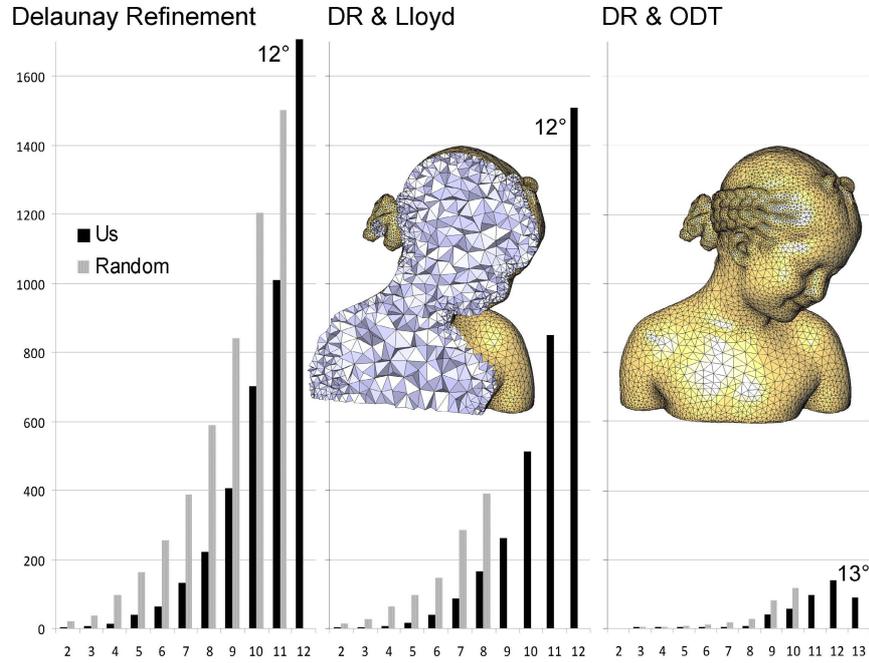
**Fig. 4.** Dinosaur. Comparison of the timings for our perturbation and random perturbation (in seconds) w.r.t. the sliver angle bound  $\alpha$  on the Dinosaur model meshes obtained by Delaunay refinement (left), followed by Lloyd optimization (middle) and ODT optimization (right).

regions, make it more difficult to perturb in a way that still preserves the gradation, even after optimization. Even in this case, ODT reaches a higher minimum angle.

For comparison we have also performed sliver exudation on meshes generated by Delaunay refinement and on meshes optimized after refinement. As expected sliver exudation performs better on the optimized meshes.

We performed two other experiments that were abandoned since they rarely succeeded in improving the mesh quality. While computing the perturbation of a vertex incident to more than one sliver, we tried combining  $\nabla V$  vector of one of the slivers and  $\nabla R^2$  of the other by using their average as perturbation direction if they were compatible. In practice such a combination was almost never successful removing the slivers. The other aborted experiment consisted of removing from the mesh the vertices that every explicit perturbation failed to perturb. In practice this never resulted in improving the minimum dihedral angle.

Moreover, our experiments show that successively applying our combined algorithm to the mesh several times while progressively increasing the angle



**Fig. 5.** Bimba. Comparison of the timings for our perturbation and random perturbation (in seconds) w.r.t. the sliver angle bound  $\alpha$  on the Bimba model meshes obtained by Delaunay refinement (left), followed by Lloyd optimization (middle) and ODT optimization (right).

**Table 1.** Angles. Minimum dihedral angles obtained by the different perturbation algorithms (combined perturbation, random perturbation, and sliver exudation). To achieve these maxima, combined perturbation takes about twice the exudation time, and random perturbation takes about six times the exudation time.

Mesh	input	combined	random	exudation
Dinosaur (DR)	0.65	25.0	24.2	2.62
Dinosaur (DR & Lloyd)	0.24	26.15	23.5	4.47
Dinosaur (DR & ODT)	2.26	28.55	22.0	4.55
Bimba (DR)	0.16	15.51	15.64	1.11
Bimba (DR & Lloyd)	0.11	16.02	15.63	3.84
Bimba (DR & ODT)	0.84	19.8	18.85	4.47

bound that defines a sliver provides higher minimal dihedral angles at the price of higher computation times. This amounts to giving priority to vertices incident to the worst slivers, cluster by cluster of minimum dihedral angles.

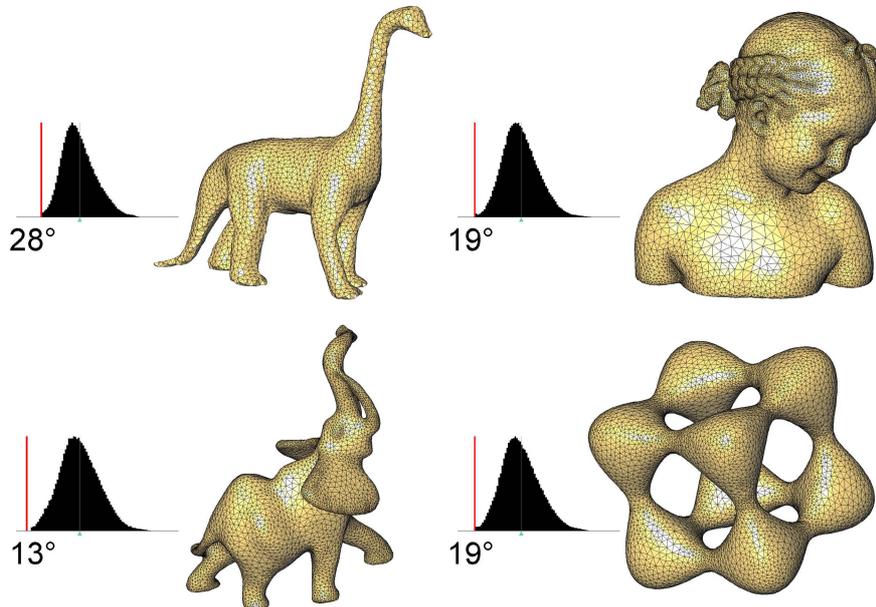


Fig. 6. Delaunay meshes perturbed with combined perturbation algorithm after ODT optimization.

Table 1 summarizes the best angles obtained in this way using combined perturbation, random perturbation and sliver exudation. In this labor-intensive experiment we only measure how far we can go in terms of dihedral angles and do not consider timing. Finally, Figure 6 shows some Delaunay meshes obtained by Delaunay refinement followed by ODT optimization and perturbed with the combined algorithm along with their dihedral angle histograms.

#### 4 Conclusion and Discussion

We have presented a practical vertex perturbation algorithm for improving the dihedral angles of a 3D isotropic Delaunay triangulation. The key idea consists of performing a gradient ascent over the sliver circumsphere radius as well as a gradient descent over the sliver volume. All vertices incident to slivers are processed, in an order devised to improve effectiveness and computation times. We compare our approach with pure random perturbation and sliver exudation.

Our experiments show that we are both faster and able to reach higher minimum dihedral angles. Our scheme is particularly well suited as a post-processing step after mesh optimization [30]. We also plan to use it in the context of mesh generation from multi-material voxel images [4].

In the cases where all vertices of a sliver are on the domain boundary, the perturbation can fail in removing a sliver as the boundary vertices are too constrained. One way to extend our approach would be to also perturb the vertices of the sliver's adjacent tetrahedra whose relocation can impact the sliver. Future work will focus on obtaining a proof of termination of our combined perturbation algorithm, and some tighter lower bounds on output dihedral angles.

## Acknowledgments

The authors wish to thank Mathieu Desbrun for helpful discussions and suggestions. We also thank Mariette Yvinec and Laurent Rineau for providing us with their implementation of sliver exudation.

## References

1. Cgal, Computational Geometry Algorithms Library, <http://www.cgal.org>
2. Amenta, N., Bern, M., Eppstein, D.: Optimal point placement for mesh smoothing. In: Proc. of the 8th ACM-SIAM Symposium on Discrete Algorithms, pp. 528–537. SIAM, Philadelphia (1997)
3. Boissonnat, J., Wormser, C., Yvinec, M.: Curved Voronoi diagrams. *Effective Computational Geometry for Curves and Surfaces*, 67–116 (2006)
4. Boltcheva, D., Yvinec, M., Boissonnat, J.-D.: Mesh generation from 3d multi-material images. In: MICCAI 2009. LNCS, Springer, Heidelberg (2009)
5. Chen, L., Xu, J.: Optimal Delaunay triangulation. *Journal of Computational Mathematics* 22, 299–308 (2004)
6. Cheng, S., Dey, T.: Quality meshing with weighted Delaunay refinement. In: Proc. of the 13th ACM-SIAM Symposium on Discrete Algorithm, pp. 137–146. SIAM, Philadelphia (2002)
7. Cheng, S.W., Dey, T.K., Edelsbrunner, H., Facello, M.A., Teng, S.H.: Sliver exudation. *Journal of the ACM* 47(5), 883–904 (2000)
8. Cheng, S., Dey, T., Ray, T.: Weighted Delaunay refinement for polyhedra with small angles. In: Proc. of the 14th Int. Meshing Roundtable (2005)
9. Chew, L.: Guaranteed-quality Delaunay meshing in 3D (short version). In: Proc. of the 13th Symposium on Computational Geometry, pp. 391–393. ACM Press, New York (1997)
10. Cutler, B., Dorsey, J., McMillan, L.: Simplification and improvement of tetrahedral models for simulation. In: Proc. of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 93–102. ACM, New York (2004)
11. Du, Q., Faber, V., Gunzburger, M.: Centroidal Voronoi tessellations: applications and algorithms. *SIAM review* 41(4), 637–676 (1999)
12. Edelsbrunner, H.: *Algorithms in Combinatorial Geometry*. Springer, Heidelberg (1987)
13. Edelsbrunner, H., Guoy, D.: An experimental study of sliver exudation. *Engineering with computers* 18(3), 229–240 (2002)

14. Edelsbrunner, H., Li, X., Miller, G., Stathopoulos, A., Talmor, D., Teng, S., Üngör, A., Walkington, N.: Smoothing and cleaning up slivers. In: Proc. of the 22nd ACM symposium on Theory of computing, pp. 273–277. ACM, New York (2000)
15. Forsman, K., Kettunen, L.: Tetrahedral mesh generation in convex primitives by maximizing solid angles. *IEEE Transactions on Magnetics* 30(5) (Part 2), 3535–3538 (1994)
16. Joe, B.: Delaunay versus max-min solid angle triangulations for 3-dimensional mesh generation. *Int. Journal for Numerical Methods in Engineering* 31(5) (1991)
17. Joe, B.: Construction of 3-dimensional improved-quality triangulations using local transformations. *SIAM Journal on Scientific Computing* 16(6), 1292–1307 (1995)
18. Klingner, B., Shewchuk, J.: Aggressive Tetrahedral Mesh Improvement. In: Proc. of 16th Int. Meshing Roundtable, pp. 3–23 (2007)
19. Krysl, P., Ortiz, M.: Variational Delaunay approach to the generation of tetrahedral finite element meshes. *Int. Journal for Numerical Methods in Engineering* 50(7), 1681–1700 (2001)
20. Labelle, F., Shewchuk, J.: Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. In: Int. Conference on Computer Graphics and Interactive Techniques. ACM Press, New York (2007)
21. Li, X.: Sliver-free 3-Dimensional Delaunay mesh generation. PhD thesis, University of Illinois (2000)
22. Li, X.: Spacing control and sliver-free Delaunay mesh. In: Proc. of the 9th Int. Meshing Roundtable, pp. 295–306 (2000)
23. Li, X., Teng, S.: Generating well-shaped Delaunay meshes in 3D. In: Proc. of the 12th ACM-SIAM Symposium on Discrete Algorithms, pp. 28–37. SIAM, Philadelphia (2001)
24. Liu, C.Y., Hwang, C.J.: New strategy for unstructured mesh generation. *AIAA journal* 39(6), 1078–1085 (2001)
25. Rineau, L., Yvinec, M.: Meshing 3D domains bounded by piecewise smooth surfaces. In: Proc. of the 16th Int. Meshing Roundtable, pp. 442–460 (2007)
26. Shewchuk, J.: Tetrahedral mesh generation by Delaunay refinement. In: Proc. of the 14th Symposium on Computational Geometry, pp. 86–95. ACM Press, New York (1998)
27. Shewchuk, J.: Two discrete optimization algorithms for the topological improvement of tetrahedral meshes (Unpublished manuscript) (2002)
28. Shewchuk, J.: What is a good linear element? interpolation, conditioning, and quality measures. In: Proc. of the 11th Int. Meshing Roundtable, pp. 115–126 (2002)
29. Talmor, D.: Well-spaced points for numerical methods. PhD thesis, University of Minnesota (1997)
30. Tournois, J., Wormser, C., Alliez, P., and Desbrun, M.: Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Transactions on Graphics* 28(3) (2009)