Université Grenoble Alpes - INP Ensimag

Master of Science in Informatics at Grenoble

3D Graphics : Project

Students :
Bastien ROURE

Professor :
Thibault TRICARD

April 25, 2025

Contents

1 Problem description

When processing meshes and 3D objects, particularly those produced by digitization such as 3D scanners, the resulting surfaces often exhibit noise and irregularities. These defects can be caused by measurement errors, inaccuracies in acquisition, or defects in mesh reconstruction. This noise interferes with the visual legibility of the model, as well as its use in technical applications. The aim of smoothing techniques is therefore to correct these imperfections while preserving the important geometric features of the mesh, such as the overall structure and important object information.

The presence of noise or irregularities can be measured or visualized in several ways. The most obvious is by visualizing the 3D object, observing defects and noise. There are also more advanced analyses, such as Mesh Spectral Analysis, where high frequencies are matched to noise, i.e. rapid variations in the shape of the mesh. Our main tool is direct object visualization. Indeed, this is the simplest method and is sufficiently accurate to visualize the various changes.

Many technical fields are impacted, such as surfaces reconstructed from medical data (MRI, CT scans), which may exhibit noise during measurement, due in particular to the precision of the tools used. Correcting this kind of inaccuracy is crucial, for example, to provide accurate anatomical modeling of the patient, or for the printing of prostheses.

We can also think of engineering or 3D printing problems where a clean, regular mesh is important. Take, for example, the 3D printing of an object or the design of a machining part. Rough or distorted surfaces can subsequently cause technical problems, inaccurate measurements, or an increase in manufacturing time.

Smoothing a mesh can therefore be very important, and in this paper we'll look at various techniques for achieving this.

2 Implemented methods

2.1 Laplacian

The first method implemented is directly the smoothing method *Laplacian*. By definition, the Laplacian is the second derivative of a continuous function. It is used to quantify the variation and curvature of this function in space, by measuring the divergence of its gradient. Here, in 3D modeling, we're working on meshes that don't belong to the continuous but to the discrete domain, so we can only approximate this Laplacian. To quantify this variation on the mesh, we use the following formula:

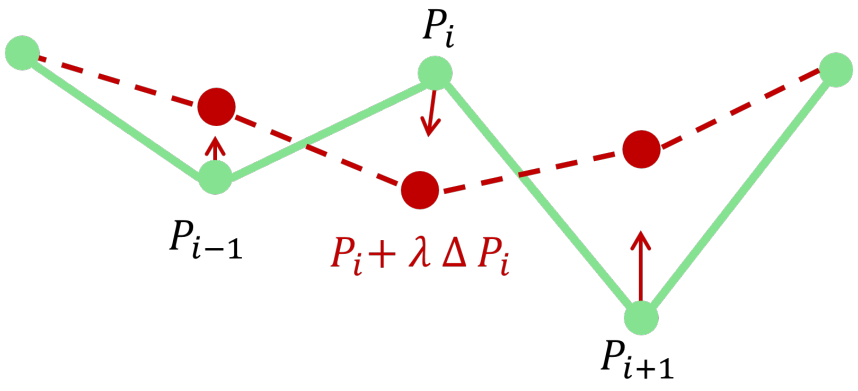
$$\Delta P_i = \left(\frac{1}{|adj(i)|} \sum_{j \in |ajd(i)|} P_j\right) - P_i, \quad \forall i \in V$$

With :

- V : All points of the mesh.
- ΔP_i : The Laplacian approximation.
- $adj(p)$: The set of neighbors of point p.

With this formula, we calculate the average distance between a point and its neighbors. To carry out this smoothing on the mesh, we need to move the point in the direction of variation minimization by adding a term weighting the displacement $0 \leq \lambda \leq 1$. In other words, the point's new coordinate will be :

$$P_i = P_i + \lambda \, \Delta P_i$$



2.1.1 Results

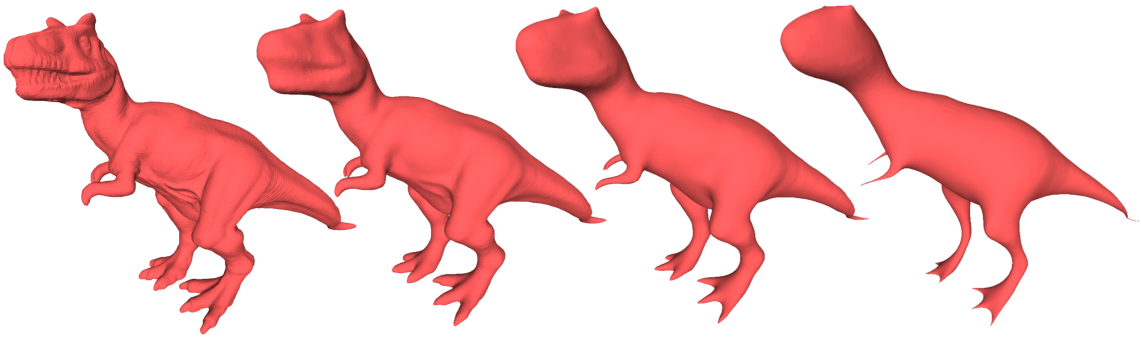


Figure 1: Laplacian with 0, 10, 50 and 200 iterations respectively, $\lambda = 1$

2.2 Taubin

From the results obtained with the previous method, we observe that after several iterations, the object becomes deformed. The reason for this is as follows : As mentioned previously, when applying this Laplacian formula, we try to minimize variations along the mesh at any point. Locally concave areas "soften", while locally convex areas "rise". The object will therefore tend towards a shape where curvature is minimal, i.e. a sphere. To limit this deformation, we can use the Taubin [1995] algorithm, which with a certain μ term compensates for this deformation by pushing the point in the opposite direction. Here's its 2-step definition:

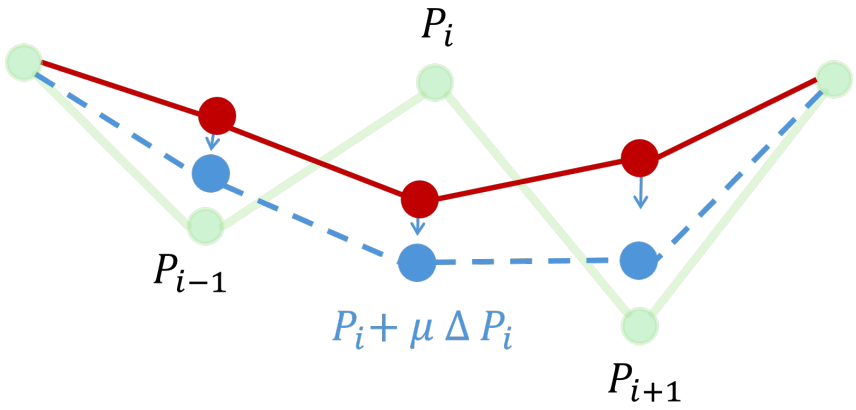
$$\textbf{(1)}\ P_i = P_i + \lambda\ \Delta P_i$$

$$\textbf{(2)}\ P_i = P_i + \mu\ \Delta P_i$$

(1) We apply the Laplacian method, which, as we've seen with the Figure 1 will contract the object.

(2) We apply the Taubin method which, by pushing the modified points in the opposite direction, "inflates" the mesh, respecting this desire to smooth out the initial shape.

> **Note :** The choice of values for λ and μ are initialized in the code to $\lambda = 0.5$ and $\mu = -\lambda$



2.2.1 Results

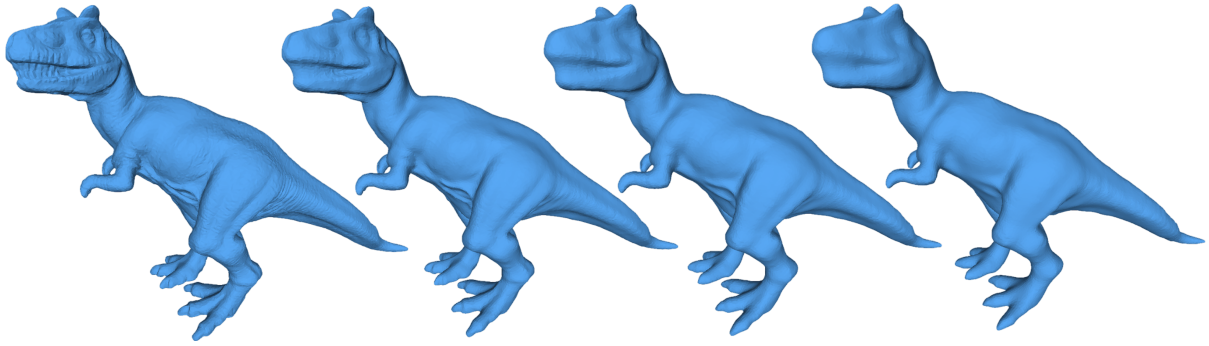


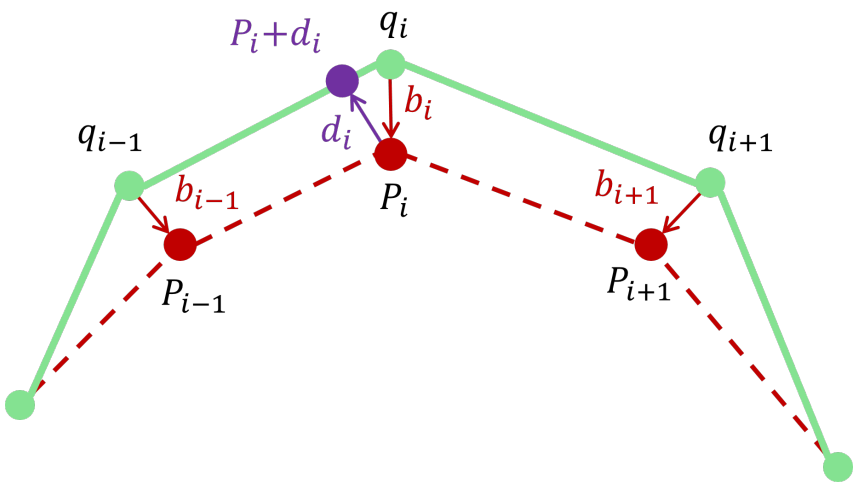
Figure 2: Taubin with 0, 10, 50 and 200 iterations respectively

2.3 HC-Algorithme

The **Taubin** algorithm takes us a big step closer to our goal of smoothing without distorting the structure. However, we can see from the Figure 2 results that we're still losing information. If we look at the dinosaur's mouth, the teeth have completely disappeared. Another algorithm allows us to retain more information while smoothing the mesh without losing the overall structure. Let's introduce HC-Algorithm Vollmer *et al.* [1999] which, after applying smoothing Laplacian, moderately pushes the points back to their previous coordinates (initial position, position just before smoothing). A vector is calculated to represent the point at which the point has "moved away" from its previous positions, and it is pushed in this direction to compensate for the deformation, taking into account the average displacement of its adjacent points.

- $B_i = p_i - \left(\alpha \cdot O_i + (1-\alpha) \cdot q_i\right)$
- $D_i = \beta \cdot B_i + \dfrac{1-\beta}{|adj(i)|} \sum_{j \in adj(i)} B_j$

With O_i the point i of Origin, before any deformation, q_i the point i before displacement by the Laplacian, p_i the point i after the Laplacian.

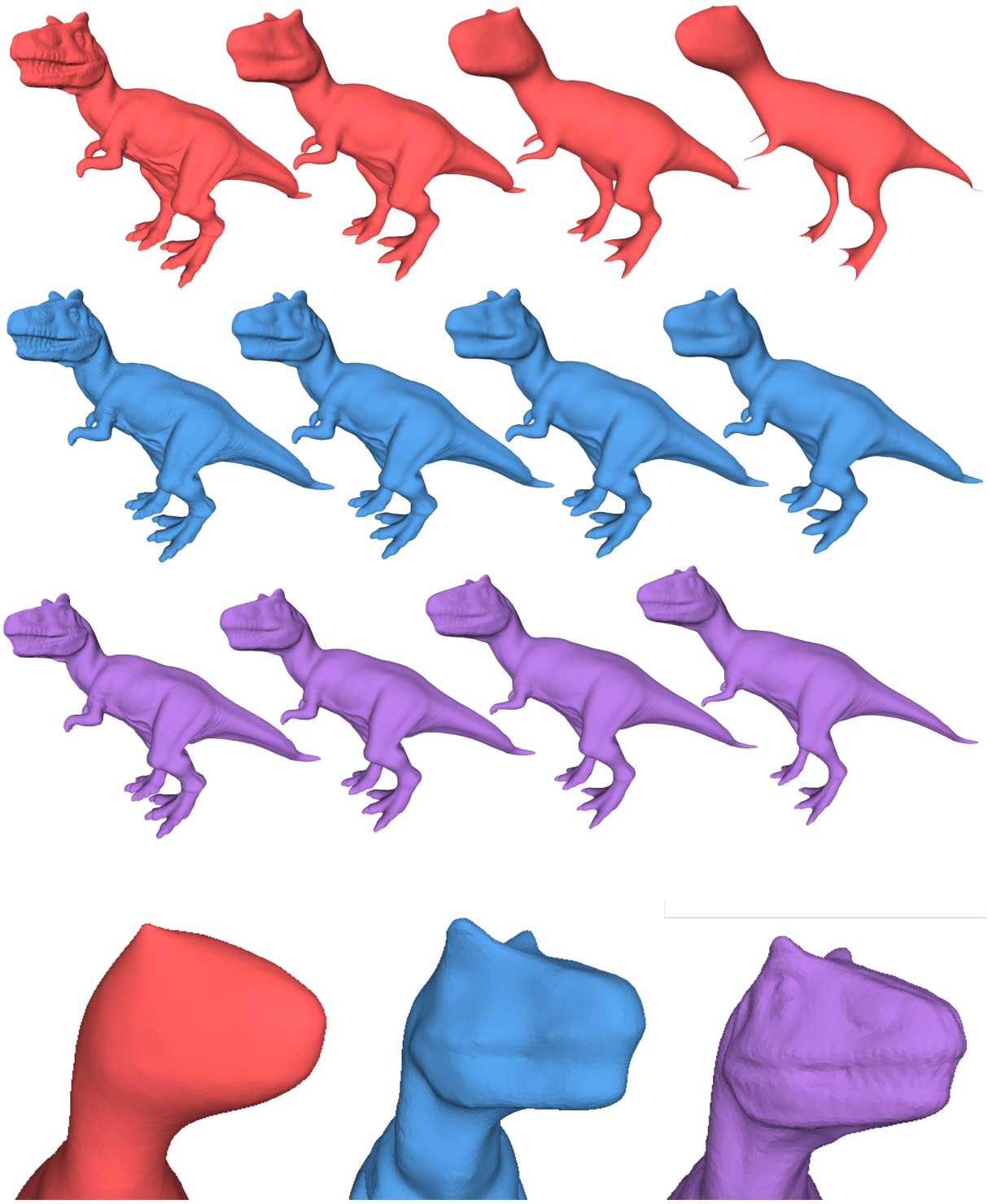


2.3.1 Results

Figure 3: HC with 0, 10, 50 and 200 iterations respectively, $\alpha = 0.5$ $\beta = 0.5$

3 Final results

Here are the results presented earlier :



We can observe a clear difference between these 3 algorithms, with a very great improvement in the stability of the mesh structure, but also in the preservation of details. However, all this comes at a cost :

- **Laplacian** : In this algorithm, we must first create the list of neighbors. Let n be the number of triangles belonging to the mesh, our loop traverses our n triangles. In this loop we check if the triangle is already in the list ($\rightarrow$ cost $O(1)$ for each lookup dict), if not we add it (operation $O(1)$). There are n triangles, and each triangle performs a constant number of O(1) operations, making the complexity $O(n)$. Now that the list has been constructed, we need to go through each point and look at its neighbors. Given that P is the number of points and V the number of neighbors locally associated with each point, we have a main loop in $O(P+V)$ (as the various calculations are considered $O(1)$), so a total algorithm with the creation of the list of neighbors in $O(n+P+V)$..

- **Taubin** : Here, too, we start by creating the list of neighbors $O(n)$, then apply the Laplacian which contracts the object, so $O(P+V)$, and finally the Taubin method which inflates the mesh (same cost as the previous operation), giving us: $O(n+2(P+V))$. In theoretical complexity, this result is similar to Laplacian, but not in practice.

- **HC-algorithm** : As before, the first step is to create the neighborhood at each point, $O(n)$, then also apply the Laplacian, $O(P+V)$, and finally the last step is to run through the neighbors for each point. So we have that HC-Algorithm and Taubin have the same theoretical complexity $O(n+2(P+V))$.

> **Note :** Complexities are calculated here over one iteration. By adding that in practice the tests are run over k iterations we have, in fact, a complexity $O(n+2k(P+V))$.

Method	1 iteration	10 iterations	50 iterations	200 iterations
Laplacian	1.7	3.9	13.8	50.9
Taubin	2.0	6.4	26.8	103.4
HC-Algorithm	1.9	8.8	37.1	148.0

Table 1: Comparison of smoothing methods in execution time (seconds)

> **Remarque :** In practice, in the HC-Algorithm, a copy of the points has to be made, so it takes longer.

4 Conclusion

Through these different algorithms, we have seen how methods have evolved to improve the smoothing of a 3D mesh. However, the latest result still has a few shortcomings. We don't totally avoid object shrinkage, which in practice can be disturbing if we attach importance to the starting shape (e.g. for machined parts). There are still many ways of improving this, such as the one presented by Desbrun *et al.* [1999], which mainly combines a scale-dependent Laplacian, which therefore respects the irregularity of the mesh, and a volume correction to avoid shrinkage.

References

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pages 317–324. ACM Press/Addison-Wesley Publishing Co., 1999.

Gabriel Taubin. A signal processing approach to fair surface design. *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 351–358, 1995.

J. Vollmer, R. Mencl, and H. Müller. Improved laplacian smoothing of noisy surface meshes. *Computer Graphics Forum*, 18(3):131–138, 1999.