Non-Rigid Shape Deformation

INF574 - Digital Representations and Analysis of Shapes

Chaussonnier Romain, Pilipyuk Alexandra

December 14, 2022

1 Presentation of the problem

A rigid modification of a mesh is one that applies only rotations or translations to its components, without any scaling. This definition refers to a rigorous shape preservation. However, for manifold meshes, this kind of transformation is very restrictive. In the real world, applying a rigid transformation only means moving the object around. But we may want to perform other kinds of shape deformations as well. Our goal is to find a non-rigid transformation, and in the same time, we would like it to be as-rigid-as-possible. With bending or stretching forced by a user, it should preserve as much as possible the properties of the initial shape. Hopefully the results would then correspond to how a real rigid object would react to a deformation with similar constraints.

Our implementation can be found at `https://github.com/30L2AN0/ARAP`.

2 Motivation

This work has several motivations. Most of them are linked to the purpose of simulating real life behaviours. For example, having virtual objects bending realistically is useful for the movie industry. It is used to create convincing special effects for the audience. It can also be used to predict with high precision how a company's product would be deformed. One could think of a car crash, for example.

Overall, The purpose of this implementation falls within the more global objective of reproducing reality in a virtual world.

3 Related work

Different approaches to the problem were proposed in the 2000s. We will cover in those paragraphs some of the fundamentally geometrical ones.

Non-rigid deformations can be modelled through the use of cages. The basic idea, introduced by Ju et al. in *Mean Value Coordinates for closed triangular meshes*, is to position vertices based on barycentric coordinates of the cage nodes [2]. In their paper, they also point out the limits of this method when using very concave meshes. It was then fixed in the work *Harmonic Coordinates for Character Articulation* by Joshi et al. [1]

A mesh can also be transformed using Poisson Editing. The idea is to try to stick a part of a mesh to another mesh. In order to make the piece fit, it is either shrunk or stretched.

Our work is mostly based on the paper *As-Rigid-As-Possible Surface Modeling*, written by Olga Sorkine and Marc Alexa [4]. They propose an approach based on Laplacian editing. Contrary to cage-based methods and Poisson editing, this allows a user to modify a mesh by directly manipulating its surface. The Laplacian editing spreads the deformation into the mesh from a few chosen constraints applied to vertices. Local displacements occur along normals, with local rotations being taken into account.

4 Implementation of ARAP

4.1 Setup

We used LibiGL and Eigen for our implementation. They offer easy to use data structures with straightforward matrix calculus. Being linked at its core to OpenGL, it also provides a window to display the results. There is simple zoom, moving around and scaling.

The HalfEdge builder and HalfEdge structure provided by Luca Castelli Aleardi in the XINF574 class labs were also used at first. Later, those files proved to be unnecessary in our approach.

4.2 Core iterative algorithm

ARAP method relies on an energy minimization, formalizing how each vertex will be displaced in relation to its direct neighbors. Consequently, for each vertex i of the mesh, each of its outgoing edges is considered. More precisely, the focus is on how much each edge is scaled compared to its original length. The difference is then added to a global sum. However, in order to focus on this scaling, the rotation R_i must be known. It is the rotation of the set of the vertex and its neighbors, called a cell.

$$E(S') = \sum_{i=1}^{n} \sum_{j \in N(i)} w_{ij} ||(p'_i - p'_j) - R_i(p_i - p_j)||^2$$

where S' is the mesh transformed from S and the w_{ij} are weights associated to each edge scaling. They are based on the edges relative lengths. To address this problem, the authors propose to minimize the energy with regards to each cell rotation as well, giving the following complete formulation:

$$E(S', R) = \sum_{i=1}^{n} \sum_{j \in N(i)} w_{ij} ||(p'_i - p'_j) - R_i(p_i - p_j)||^2$$

Numerically, the solving is iterative. The energy is minimized with regards to the rotations R_i, then with regards to the new vertices positions p'_i, and then back to the rotations, and so on and so on. We followed the same process.

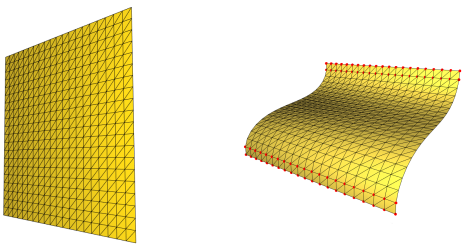

Figure 1: Demonstration of ARAP algorithm

4.2.1 Rotations estimation

First and foremost, the discrete Laplace-Beltrami operator needs to be computed. It corresponds to the weights contribution in our general energy equation. It also helps retrieving neighbors. We computed the operator as a sparse matrix, using an igl method:

```
1 igl::cotmatrix(vertices, faces, LB);
```

Then, to get the rotation R_i minimizing the energy contribution of a single cell, the covariance matrix $C_i = \sum_{j \in N(i)} w_{ij}(p_i - p_j)(pi' - pj')^T$ is also necessary. Indeed, if its SVD decomposition is written $C_i = U_i \Sigma_i V_i^T$, the wanted rotation simply is $R_i = V_i U_i^T$. The result is correct up to a small change of sign. From all this, using the Laplace-Beltrami operator comes naturally. Because of its nature and its being sparse, for each vertex i, retrieving all the neighbors is very quick. We simply visit all the elements in the column i that are not 0, and update C_i with them. At the end of the loop, we compute the SVD decomposition and retrieve the rotation.

4.2.2 Positions estimation

Minimizing the energy with regards to the new positions p'_i is slightly more complex. The method the authors used is to nullify all its partial derivatives. It was also our process:

$$\forall i \in \{1, \dots, n\}, \frac{\partial E(S')}{\partial p'_i} = 0$$

which can be rewritten:

$$\sum_{j \in N(i)} w_{ij}(p'_i - p'_j) = \sum_{j \in N(i)} \frac{w_{ij}}{2}(R_i + R_j)(p_i - p_j)$$

On the left side, the Laplace-Beltrami operator LB can be found again. It is applied to p'. The right side is a fixed vector b. Hence, the problem is reduced to solving a linear system.
Nevertheless, a slight change must be applied to some of LB's and b's rows. Some p'_i are going to be fixed by the user, meaning that they must not be deduced from the solving. This small obstacle can be bypassed easily. By putting 1 at (i, i) in LB, and by replacing with 0 all the other elements of the line, the left side is fixed. In b, the i-st element is replaced with the desired destination c_i for p_i. The update is then done. It is good to keep in mind that three systems are actually going to be solved. There is one for each of all p' components x, y, and z. In reality, what is put as the i^{th} element of b is $c_{i,x}$, $c_{i,y}$ or $c_{i,z}$ accordingly.

As it is done for rotations with the covariance matrix, the algorithm loops again on the Laplace-Beltrami matrix elements. This way b is computed. Setting the left side consists in changing a few of LB rows as described in the previous paragraph. Now, all that is left is to solve the three systems, using Eigen's LU sparse solver:

```
Eigen::SparseLU<SparseMatrix<double>, Eigen::
    COLAMDOrdering<int>> sparse_solver;
sparse_solver.analyzePattern(leftSide);
sparse_solver.factorize(leftSide);
new_vertices.col(0) = sparse_solver.solve(rightSide.
    col(0));
new_vertices.col(1) = sparse_solver.solve(rightSide.
    col(1));
new_vertices.col(2) = sparse_solver.solve(rightSide.
    col(2));
```

4.3 Initial rough guess

One would notice that in order to compute new rotations, new positions must be given, and in order to get new positions, rotations must be given. The snake is biting its own tail. There needs to be a first guess to start the core iterations.

While most of the algorithms online set rotations as identity matrices to compute the first positions, we chose to produce a rough but more accurate first guess. It is similar to the one used in the paper. Our guess also has the advantage of generating more coherent intermediate steps in the solving. If the user does not run the algorithm until convergence, the result will still be coherent.

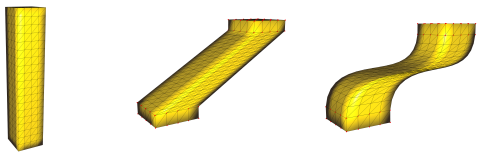

Figure 2: A bar (left) and its first guesses, using: identity rotations (middle), differential coordinates (right)

The method is based on *Differential Coordinates for Interactive Mesh Editing* by Lipman et al [3]. In this paper - in which Olga Sorkine also had a contribution - they use a scheme to approximate every vertex as a linear contribution of all its neighbors.

$$p_j \approx \sum_{i \in supp(j), i \neq j} \alpha_{ji} p_i$$

Then, the differential representation of the mesh V created by the scheme can be computed as:

$$D(V) = V - S(V)$$

In most cases, this is how far p_j is from its neighbors centroid.

The next step in order to solve a non-rigid transformation of V is to represent D as a matrix M of size $(|V|, |V|)$:

$$M_{ij} = \begin{cases} 1 & i = j \\ -\alpha_{ij} & j \in supp(i) \\ 0 & otherwise \end{cases}$$

We also compute D applied to the original mesh V:

$$\begin{aligned} D(V) &= M \cdot (V.col(0), V.col(1), V.col(2)) \\ &= M \cdot (p^{(x)}, p^{(y)}, p^{(z)}) \\ &=: (\delta^{(x)}, \delta^{(y)}, \delta^{(z)}) \end{aligned}$$

Finally, the idea is to force new vertices to obey the local displacements as much as possible. Formally it is equivalent to solving $Mp'^{(x)} = \delta^{(x)}$ (the same goes for y and z) in the least-square sense.

However, some constraints are also applied to the mesh, chosen by the user. Therefore, similarly to the "Positions estimation" step in ARAP method, some rows of M and δ must be changed. Or this is rather what could have been done if the resolution was precise, but it is an L2 approximation. Using this system of rows modifications means that those constraints would also be treated in the least square sense, and are not absolute. To ensure this, we instead add the $p_i = c_i$ equations to the system as definite ones. This can be done with igl.

In our code, because the Laplace-Beltrami operator was not needed at this step, we had chosen to use the HalfEdges structure of the mesh to visit all the neighbors. Nevertheless, the operator was then computed for the ARAP part of the algorithm. So we figured we may as well use it for this "First guess" part. The differential mesh operator we use is the simple Laplacian operator, corresponding to a regular centroid calculus:

$$LB(p_j) = p_j - \frac{1}{d_j} \sum_{i:(j,i) \in E} p_i$$

where d_j is the vertex valency. Once again the algorithm loops on LB non zero elements, but this time without even using them. The visit is done

solely to determine whether a vertex is a neighbor of j or not, and then M and δ are updated.
As mentioned, the solving is done using igl methods:

```
igl::min_quad_with_fixed_data<double> mqwf;
igl::min_quad_with_fixed_precompute(LB, constraintsIds
    , 0, true, mqwf);
igl::min_quad_with_fixed_solve(mqwf, delta.col(i),
    constraints.col(i), 0, newVerticesColumn);
newVertices.col(i) = newVerticesColumn;
```

5 Some 3D-printing

The goal is physically based at its core. Indeed, "rigid" is a word used to characterise real objects. We figured that comparing our results with real-world bending would be relevant, so we printed three objects with 3D printers:

Figure 3: From left to right: a wireframe human bust, a filled human body, and hollow parts of a human body

The first one is a human bust, printed using a wireframe technique. The idea was to mimic the grid aspect of edges. We used a solid plastic for the printing, which would make any stretching or bending difficult for a fully filled 3D-printed object. In the wireframe case, one layer can be easily detached from the previous one. The phenomenon of a wireframe layer being moved at the same level as the previous one, or being separated away from it, is similar to the phenomenon of edges shrinking or stretching along the vertical axis.
The second one is a full human body with inner filling, printed with an elastic material. Its behaviour as it is stretched is the one closest to a volume preserved mesh transformation. The expression "volume preservation" is used in contrast to "surface preservation", as it will be explained in "7.1 Limitations".
The third and last one is the same human body on a bigger scale, with the same elastic material as the small human body. We made it hollow in order to make its behaviour similar to that of a high resolution mesh.

6 Results

The algorithm we propose is efficient and gives the same visual results as the ones obtained by the authors. The results are also similar the ones given by the wireframe 3D bust or the hollow full body. Unfortunately, we were not able to print the meshes that were used in our code. Precise comparisons therefore can not be displayed.
The number of iterations needed to observe a convergence depends a lot on the mesh, ranging from three or four to a few dozens in some cases. However the changes applied at each step are immediate. Therefore, this number does not have any impact on the real-time requirement. Even if we were to imagine hundreds of iterations for some meshes, we could still choose a threshold for the number of iterations, whose presence would barely change the results.

6.1 Limitations

As expected, the method is not perfect.
First, high resolution meshes sometimes generate folding phenomena.

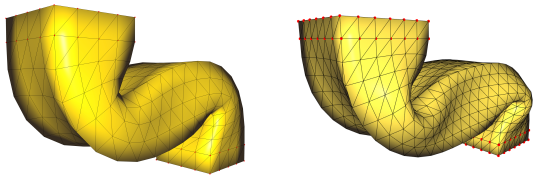

Figure 4: Differences in the folding phenomenon with slightly increasing resolutions

In order to stay as-rigid-as-possible, the vertices take advantage of the inner free room of the mesh. When it is subject to intense geometrical modifications, the mesh decides to fold on itself. One could think that this unrealistic behaviour could be the result of an approximation shortcut by the ARAP method. In the end, it could be considered as such. However, we also observed this phenomenon physically on our 3D printed models.
The main reason behind this issue is that what becomes rigid is not the entire object, but its surface. In the case of high detailed meshes, the

edges are less and less analogous to bones that should not be bent. They are more similar to infinitesimal portions of skin. So, applying ARAP results on having a very rigid skin for our models, that try not to bend, even if it means losing volume where the bending happens. This is still realistic in a way, if one manipulates hollow objects, and it explains why our 3D printed hollow leg behaves the same way.
Second issue: there can be self-intersection. This makes sense. No constraints were input to avoid it.

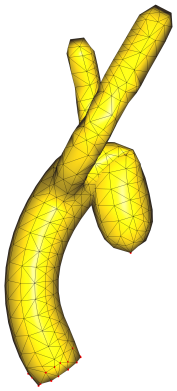

Figure 5: A cactus self-intersecting because of a hard constraint

6.2 Improvements

A solution to the first deflect would be to add inner edges, working as bones or as a wire-frame supporting structure. An arrangement similar to the one inside the Statue of Liberty could be imagined.

Figure 6: The Statue of Liberty seen from the interior

But the finer the surface mesh is, the finer the inner structure needs to be. This would be extremely costly to store. Alternatively, those edges could be theoretical. Their contributions could be implemented in the algorithm without them being stored at all. Or they could still be kept, but in a small number, with high weights w_{ij} associated to them.
In a way, in the process of 3D printing, inner edges are already used. Most of the objects are not printed fully. The inside only consists in a supporting grid with cube-shaped holes in it. We printed the same similar human body as the hollow one, but on a smaller scale and with this inner structure included. When moving around its extremities like the legs or the arms, no folding phenomenon appears. This reinforces our belief that inner edges would work.

Figure 7: Bending of the hollow (left) and the filled (right) elastic 3D-printed body, with and without folding phenomenon

References

[1] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. In *Pixar Technical Memo 06-02b*. Pixar Animation Studios, 2007.

[2] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. Rice University, SIGGRAPH, 2005.

[3] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, and David Levin. Differential coordinates for interactive mesh editing. In *Proceedings of the Shape Modeling International*, 2004.

[4] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, pages 109–116, 2007.