REPORT OF PROJECT INF574

NON-RIGID DEFORMATION BY USING THE TECHNIQUE AS-RIGID-AS-POSSIBLE

13 December 2022

Nikita DUDOROV - Vannvatthana NORNG

ÉCOLE POLYTECHNIQUE

IP PARIS

1 INTRODUCTION

In this project, we would like to implement the method As-Rigid-As-Possible (ARAP) that allows deforming a mesh in a natural way by dragging one of its parts (see an example at figure 1). ARAP technique was first published by Olga Sorkine and Marc Alexa in 2007.

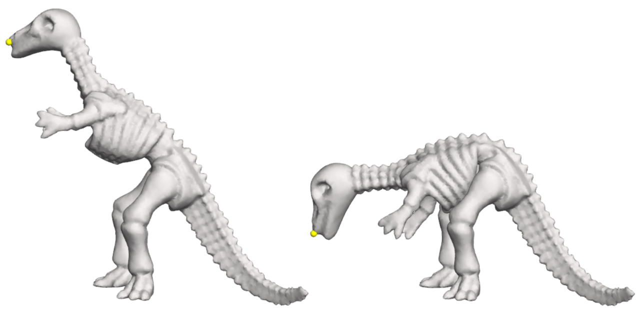

FIGURE 1 – A result shown in the paper using one vertex as the constraint (yellow)

ARAP method focuses on rigidity of each cell[1] (see Figure 2) of the shape. For each cell $\mathcal{C}_i$ of the shape, ARAP would like to find the deformation of this cell $\mathcal{C}'_i$ such that the Shell energy or the elastic energy that measures the difference between these two shapes $E(\mathcal{C}, \mathcal{C}')$ (1) is minimized. The intuition for this energy is the following : when the deformation is close to a rigid, in can be locally approximated with rotations R_i of each cell C_i.

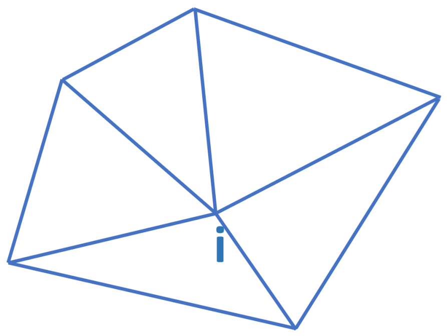

FIGURE 2 – Cell corresponding to vertex i

1. A cell corresponding to a vertex i is the ensemble of all the faces incident to the vertex i

2 IMPLEMENTATION

2.1 As-Rigid-As-Possible's algorithm

In the algorithm of ARAP, we start first by choosing the set of constraint points (let us denote it by $\mathcal{F}$) which consists of 2 types of points :

— **Stable points :** the points (or vertices) whose cells do not move nor deform during the deformation process which means that their positions stay the same all the time.

— **Handle points :** the points whose cells are dragged by the users to any positions they want in order to do the deformation.

Then, we try to find the unknown positions of the other vertices beside constraint points, but before that, the rotation matrix R_i of each cell i is required. Let $\mathcal{N}(i)$ be the set of all neighbors of the vertex i. For each cell $\mathcal{C}_i$ of the vertex i, we can find the optimal rotation matrix R_i that minimize the Shell energy given by :

$$E(\mathcal{C},\mathcal{C}') = \sum_{j\in\mathcal{N}(i)} w_{ij} \left\|(p'_i - p'_j) - R_i(p_i - p_j)\right\|^2 = \sum_{j\in\mathcal{N}(i)} w_{ij} \left\|e'_{ij} - R_i e_{ij}\right\|^2, \tag{1}$$

where p_i, p'_i are the original and the deformed positions of the vertex i, respectively (same for the vertex j), and w_{ij} in this case, we use the cotangent weight for the edge (i,j) given by :

$$w_{ij} = \frac{1}{2}\left(\cot(\alpha_{ij}) - \cot(\beta_{ij})\right) \tag{2}$$

where α_{ij} and β_{ij} are the angles opposite to the edge (i,j). R_i can be then derived from the singular value decomposition of the covariance matrix $S_i = U_i\Sigma_i V_i$ such that $S_i = P_i D P'_i$ where P_i, and P'_i are the matrices where the columns contain e_{ij} and e'_{ij}, D is the diagonal matrix contains w_{ij}. So

$$R_i = V_i U_i^T \tag{3}$$

After calculating all the R_i, they will be substituted into following the sparse linear system of equations in order to find the p', for all vertex i :

$$\sum_{j\in\mathcal{N}(i)} w_{ij}(p'_i - p'_j) = \sum_{j\in\mathcal{N}(i)} \frac{w_{ij}}{2}(R_i + R_j)(p_i - p_j) \tag{4}$$

$$\text{or in compact form : } \quad Lp' = b \tag{5}$$

As we can see in the equations (3) and (4), the problem is that we need to know $R_i, \forall i$ in order to calculate the new positions p'_i and vice versa. Therefore, one of them needs to be assumed to calculate the other one, in this case, we made the initial guess for p'_i to calculate R_i, then recalculate the p'_i using the R_i found earlier, and we do that back and forth for several iterations.

In our implementation we do iterations until the difference of meshes obtained at previous and current iterations gets low enough : $1e-3$: $\|V_k - V_{k+1}\|_2 \leq 1e-3$. This convergence rate seemed to be sufficient for a good estimation of the deformed mesh. To solve the system 5 we used LU decomposition for sparse matrices from Eigen [2].

In order to make the deformation smooth, the paper suggests to use as initial guess the shape position at previous frame. That might cause the code to run too slow as demands to do all computations at the screen's frequency. Instead we will move the handle points by a small distance each time we press a keyboard key. Precisely, we predefine small translations of handle points up,down,left,right by pressing keys W,S,A,D respectively and small rotations of handle points (around their geometric center) clockwise, counter-clockwise by pressing keys E,Q respectively. We note that we had to adjust the values of this translations and rotations to find the balance between too slow and very beautiful deformation and very fast but too bad deformation (if translation/rotation is too large, the initial guess is not precise enough and the mesh could be deformed badly). As for this approach we can only set fixed translations and rotations, we always scale the input mesh to a fixed size so that our code can work for any mesh.

2.2 Interface Implementation

For now, we've implemented interactive selection of constraint points (stable and handle points) by using mouse, and apply the deformation to the shape by using keyboards. To detect mesh vertices by mouse click we used mouse ray casting [1]. We've also tried to implement the option where users can move the handle points using the mouse, unfortunately, it still had a lot of bugs to fix, so we decided not to use it.

Finally, our implementation of the ARAP's method works really good now, even though it is a bit slow.

3 RESULTS

In this case, we use the mesh **bar2.off** among the mesh files given on the website of the paper. We chose the color yellow for the handle points and red for the stable points of the mesh, here is the result :

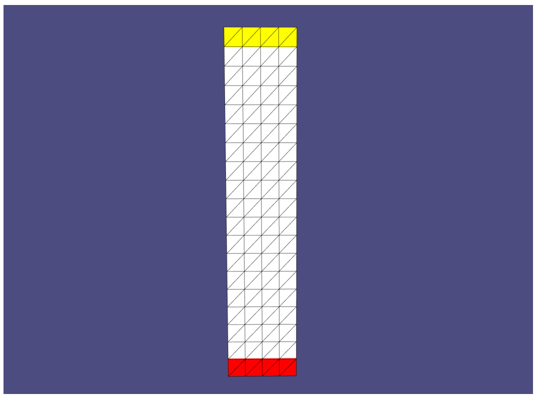

FIGURE 3 – Bar before the deformation

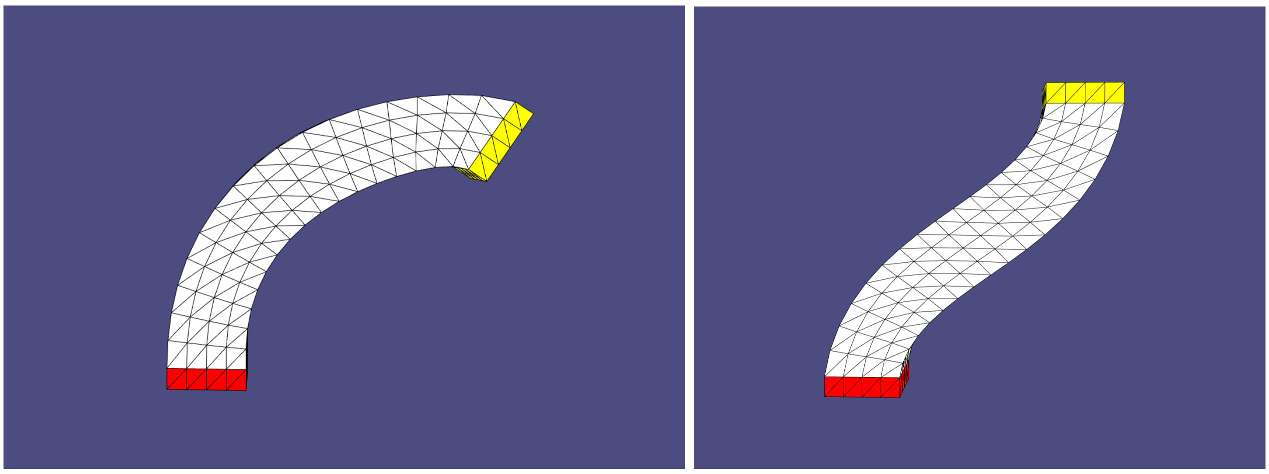

FIGURE 4 – Bar after deformation

As we can see in the Figure 4, the deformation of the bar looks really good.

4 CONCLUSION

We have successfully implemented the algorithm for rigid shape deformation proposed in the article. Despite the lack of interactiveness of our interface and of smoothness of the deformation, our implementation of the As-Rigid-As-Possible provides shape deformation which is very similar to that presented in the article. In our implementation the user can choose the constraint points with mouse clicking. Then, to deform the mesh in desired way, he can translate/rotate handle points using keyboard. As to calculate each small deformation we need to do about 25 iterations, computing at each iteration the rotation matrices R by equation (3) and updating p' by solving the sparse system of equations (5), our algorithm doesn't work fluently.

Our implementation is to be improved in terms of the time needed for calculating the deformation, and the interactiveness of the interface. For the interface, it can be improved by implementation of handle points dragging using the mouse. And maybe runtime could be improved by using another solver for sparse system (5) (we used LU decompositon).

RÉFÉRENCES

[1] *Code for raycasting with mouse.* URL : https://github.com/annareithmeir/AsRigidAsPossibleDeformer/blob/main/src/GUI.h.

[2] *LU solver for sparse matrices.* URL : https://eigen.tuxfamily.org/dox/classEigen_1_1SparseLU.html.