Ukrainian Catholic University

Faculty of Applied Sciences

Computer Science Programme

As-Rigid-As-Possible Mesh Transforamtion

MMML final project report

Authors:

Mykhailo Pasichnyk

Jan 2023

Abstract

Understanding a surface's local behavior is crucial for deformation and editing in the surface modeling field. I will implement a modeling approach that defines operations based on the rigidity of local transformations. We will observe non-linear energy and consider this problem a minimization task. So such mesh editing would be solved via a simple and effective algorithm. The proposed method preserves fine details and gives intuitive elastic deformations. I implemented such an algorithm by means of Python and NumPy from scratch as a MMML course project. Source code [1].

1 Introduction

Skinning and other linear methods for deformation are inherently limited. Difficult areas, especially when large rotations are imposed by the handle constraints. The As-Rigid-As-Possible (ARAP) algorithm aims to preserve the local shape of a deforming object as much as possible, similar to how physical objects deform when subjected to smooth, large-scale transformations. The ARAP algorithm achieves this by dividing the surface S into small overlapping cells and optimizing the rigid transformations to S' within each cell.

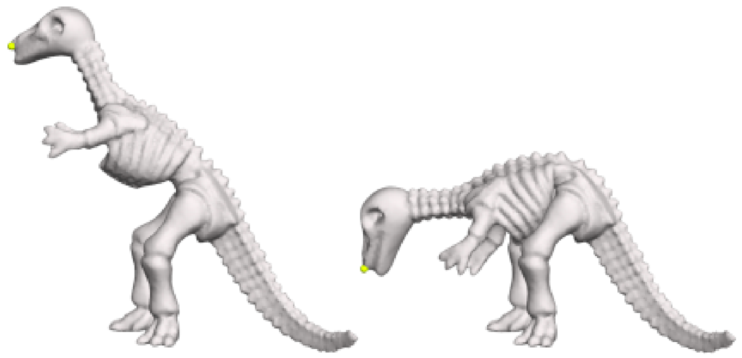

Figure 1: Large deformation obtained by translating a single vertex constraint (in yellow) using ARAP technique. [2]

Brief algorithm:

1. **Compute initial cell transformations**: copy the given mesh. We have constraints for some points, so we set constraints vertexes at these positions, and the rest of points do not move.
2. **Optimization**: iteratively updates the vertex positions to minimize the energy function, which measures the deviation from the desired rigid transformations within each cell.
   (a) **Compute triangles rotations**: The optimal rotation matrix is computed using each cell's current vertex positions and the initial and deformed edges. This step ensures that the deformation within each cell is as rigid as possible.
   (b) **Update vertex positions**: The vertex positions are updated by solving linear system of equations. Each vertex's location is determined by the weighted average of its neighbors' positions, with the weights depending on the computed cell rotations.

(c) **Repeat optimization**: Steps [2a] and [2b] repeat ittractively until convergence. The stopping criteria we define by simply fixing the number of iterations or the energy reduction tolerance.

The crucial mathematical concept in the ARAP algorithm is the notion of local rigidity. The algorithm aims to find the optimal rigid transformations for each cell such that the deformed shape locally preserves the initial shape. The energy function quantifies the deviation from rigidity, and the optimization process seeks to minimize this energy by updating the vertex positions and recomputing the cell rotations. By dividing the surface into cells and optimizing locally within each cell, the ARAP algorithm can handle complex, detailed surfaces while preserving fine-scale details. It provides an intuitive and efficient framework for shape deformation in interactive modeling applications.

2 Math derivations

2.1 Energy model

Given the cell $\mathcal{C}_i$ corresponding to vertex i, and its deformed version $\mathcal{C}_i'$, we define the approximate rigid transformation between the two cells by observing the edges emanating from vertex i in S and S'. If the deformation $\mathcal{C} \rightarrow \mathcal{C}'$ is rigid, there exists a rotation matrix $\mathbf{R}_i$ such that

$$\mathbf{p}_i' - \mathbf{p}_j' = \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j), \forall j \in \mathcal{N}(i)$$

When the deformation is not rigid, we can still find the best approximating rotation $\mathbf{R}_i$ that fits the above equations in a weighted least squares sense, i.e., minimizes

$$E(C_i, C_i') = \sum_{j \in N(i)} w_{ij} \quad \|\mathbf{p}_i' - \mathbf{p}_j - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2$$

Thus our mean deal now is to minimize $\sum_i \mathcal{E}(\mathcal{C}_i, \mathcal{C}_i')$.

2.2 Rotation estimating

The optimal rotation $\mathbf{R}_i$ for fixed $\mathcal{C}_i, \mathcal{C}_i'$. Let the edge $\mathbf{e}_{ij} = \mathbf{p}_i - \mathbf{p}_j$, and similar for $\mathbf{e}_{ij}'$ for the deformed cell $\mathcal{C}_i'$. We denote summation over j as a shorthand for $j \in \mathcal{N}(i)$. Then we can rewrite the energy function as

$$\begin{aligned}
&\sum_j w_{ij} \left(\mathbf{e}_{ij}' - \mathbf{R}_i\mathbf{e}_{ij}\right)^T \left(\mathbf{e}_{ij}' - \mathbf{R}_i\mathbf{e}_{ij}\right) \\
&= \sum_j w_{ij} \left(\mathbf{e}_{ij}'^T\mathbf{e}_{ij}' - 2\mathbf{e}_{ij}'^T\mathbf{R}_i\mathbf{e}_{ij} + \mathbf{e}_{ij}^T\mathbf{R}_i^T\mathbf{R}_i\mathbf{e}_{ij}\right) \\
&= \sum_j w_{ij} \left(\mathbf{e}_{ij}'^T\mathbf{e}_{ij}' - 2\mathbf{e}_{ij}'^T\mathbf{R}_i\mathbf{e}_{ij} + \mathbf{e}_{ij}^T\mathbf{e}_{ij}\right)
\end{aligned}$$

Terms without $\mathbf{R}_i$ are constant in the minimization so eliminate them.

$$\underset{\mathbf{R}_i}{\operatorname{argmin}} \sum_j -2w_{ij}\mathbf{e}'^T_{ij}\mathbf{R}_i\mathbf{e}_{ij} = \underset{\mathbf{R}_i}{\operatorname{argmax}} \sum_j w_{ij}\mathbf{e}'^T_{ij}\mathbf{R}_i\mathbf{e}_{ij}$$
$$= \underset{\mathbf{R}_i}{\operatorname{argmax}} \operatorname{Tr}\left(\sum_j w_{ij}\mathbf{R}_i\mathbf{e}_{ij}\mathbf{e}'^T_{ij}\right)$$
$$= \underset{\mathbf{R}_i}{\operatorname{argmax}} \operatorname{Tr}\left(\mathbf{R}_i\sum_j w_{ij}\mathbf{e}_{ij}\mathbf{e}'^T_{ij}\right)$$

Let:

- $\mathbf{D}_i$ - a diagonal matrix containing the weights w_{ij}
- $\mathbf{P}_i$ is the $3 \times |\mathcal{N}(v_i)|$ containing $\mathbf{e}_{ij}$'s as its columns
- $\mathbf{P}'_i$ - the same
- $\mathbf{S}_i$ - the covariance matrix

$$\mathbf{S}_i = \sum_{j \in \mathcal{N}(i)} w_{ij}\mathbf{e}_{ij}\mathbf{e}'^T_{ij} = \mathbf{P}_i\mathbf{D}_i\mathbf{P}'^T_i$$

The rotation matrix $\mathbf{R}_i$ maximizing $\operatorname{Tr}(\mathbf{R}_i\mathbf{S}_i)$ is obtained when $\mathbf{R}_i\mathbf{S}_i$ is symmetric psd (positive semi-definite). If $\mathbf{M}$ is a psd matrix then for any orthogonal $\mathbf{R}$: $\operatorname{Tr}(\mathbf{M}) \geq \operatorname{Tr}(\mathbf{RM})$. So we can derive $\mathbf{R}_i$ from the singular value decomposition of $\mathbf{S}_i = \mathbf{U}_i\Sigma_i\mathbf{V}^T_i$:

$$\mathbf{R}_i = \mathbf{V}_i\mathbf{U}^T_i$$

Up to changing the sign of the column of $\mathbf{U}_i$ corresponding to the smallest singular value, such that $\det(\mathbf{R}_i) > 0$.

There is no complex logic behind updating vertex positions, we just apply $\mathbf{R}_i$ for the respective cells.

2.3 Weights

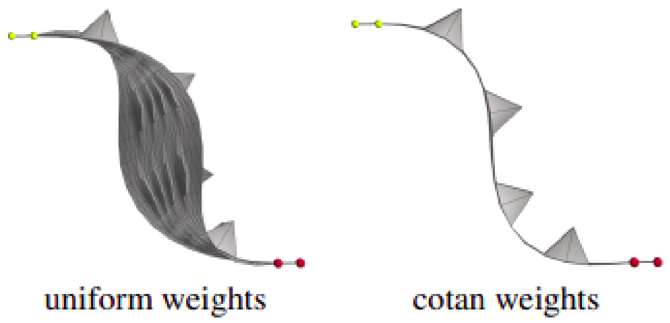

Figure 2: Demonstration of the importance of proper edge weighting in the energy formulation. Uniform weighting ($w_{ij} = 1$) - to asymmetrical results. Cotangent weighting - eliminate the influence of the meshing bias using ARAP. [2]

We measure the rigidity of the whole mesh by the sum of the rigidity per cell. Thus, we obtain the energy function in [2.1]. w_i, w_{ij} - some fixed cell and edge weights. Since the input mesh is fixed, the only variables in $E(\mathcal{C}', \mathcal{C})$ are the deformed vertex positions $\mathbf{p}'_i$. This is because the optimal rotations $\mathbf{R}_i$ are well-defined functions of $\mathbf{p}'$, as was shown in the previous section.

The choice of per-edge weights w_{ij} and per-cell weights w_i is important for making our deformation energy as mesh-independent as possible, as demonstrated in Figure 2. The weights should compensate for non-uniformly shaped cells and prevent discretization bias. Therefore we use the cotangent weight formula for w_{ij} [4]:

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

α_{ij}, β_{ij} are the angles opposite of the mesh edge (i, j) (for a boundary edge, only one such angle exists). We further note that the deviation from rigidity, as defined by (3), is an integrated quantity, so that the cell energy is proportional to the cell area, and we can set $w_i = 1$. An alternative explanation for this would be using the area-corrected edge weights $w'_{ij} = (1/A_i)\, w_{ij}$, where A_i is the Voronoi area of cell $\mathcal{C}_i$, and then also setting the cell weights to be the Voronoi area: $w'_i = A_i$. The area term simply cancels out, and we are left with the symmetric cotangent weights w_{ij}.

3 Implementation and results

The algorithm was implemented in Python via NumPy from scratch [1]. Moreover, I provide interactive visualizations using Open3D.

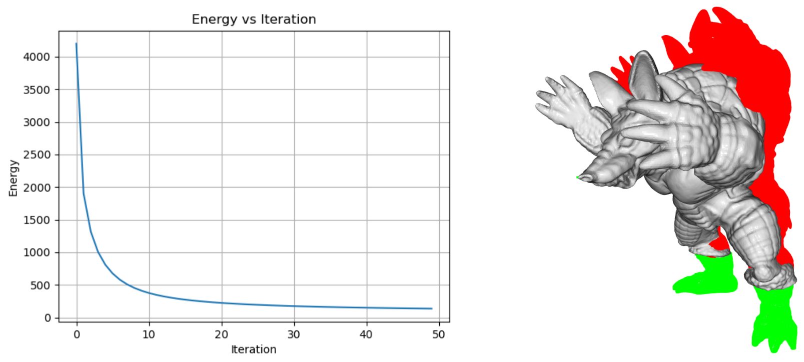

Figure 3: Demonstration of ARAP algorithm on armadillo mesh transformation.

My implementation on Python for huge meshes works quite slowly. For armadillo mesh (345K triangles and 172K vertices), 40s per iteration on Intel(R) Core(TM) i5-1135G7 @ 2.40GHz. But there is implementation by Open3D [3] on C++ with the multithreaded pipeline, which took for 1 iteration took 1s. This algorithm converges really fast. After 20 iterations changes of the energy function are tiny.

4 Conclusions

The As-Rigid-As-Possible (ARAP) Mesh Transformation algorithm provides a simple and efficient method for deforming and editing surfaces. Its local stiffness concept, reduced energy strategy, and appropriate loading scheme help it to handle complex materials and preserve fine-scale information and can also be applied to 2D meshes. Although the implementations may differ, the algorithm converges quickly and shows efficiency, making it a valuable tool for modifying interaction patterns in a variety of applications.

References

[1] https://github.com/fox-flex/ARAP

[2] https://igl.ethz.ch/projects/ARAP/arap_web.pdf

[3] https://github.com/isl-org/Open3D

[4] https://www.cs.jhu.edu/~misha/Fall09/Pinkall93.pdf