Independent Study-2

Vraj Thakkar

Prof. Aditya Tatu
(Mentor)

Dec 2024

Contents

Chapter 1

Mesh Deformation Algorithms

1.1 Background

We cannot represent surfaces as a homeomorphic function in a computer. A computer requires a discretized representation of the surface and that is known as mesh. There are various ways to represent a mesh, using the signed distance function, polygon mesh, etc. Usually, a laser camera is used to pinpoint the location of different points on the mesh and then these 3-D points are used to generate mesh by using different interpolating schemes. These raw points are known as 0^{th} approximation of the surface whereas when they are interpolated with straight lines to form a polygon, the resulting mesh is known as the first approximation to the surface. In different mesh representations, triangle meshes are the most widely used for geometry processing tasks. One of the main reasons is that deforming a planar triangle results in a planar triangle whereas deforming a higher-order polygon such as a quadrilateral may not result in a planar quadrilateral.

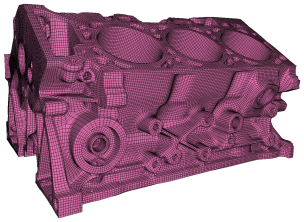

Figure 1.1: Quad mesh representation(geometryfactory.com)

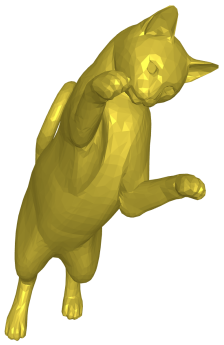

Figure 1.2: Triangular mesh of a cat

1.1.1 Triangular Meshes

Triangular meshes are the most widely used representations of surfaces in geometry processing. To represent a surface as a triangular mesh we need to know how the vertices are connected to form triangles and also how the vertices are placed in the 3-D space. Hence, we need to store geometry as well as topological information. Any triangular mesh M is represented by coordinates $V_{n\times 3} \in \mathbb{R}^3$ and faces as $F_{n\times 3} = \{(i, j, k)\}$ where i, j, k are the indices of the respective vertices and $i \neq j \neq k$ for any face.

1.1.2 Rigid Transformations

When we talk about deforming a mesh we have to talk about transformations. There are two broad kinds of transformations rigid and non-rigid. The rigid transformations are those which preserve the Euclidean distance between any two points. Examples of rigid transformations are rotation, translation, etc. Global rigid transformations on meshes preserve the shape of the mesh completely. Let's talk about rotation matrices for 3-D shape modeling. Rotation matrices are a subset of 3×3 matrices such that $det(R) = 1$ and $RR^T = I$. These properties imply that these transformations preserve angles and lengths. Let's prove that these matrices preserve length $||Rv|| = \sqrt{(Rv)^T Rv} = \sqrt{v^T R^T Rv} = \sqrt{v^T v}$. Similarly, we can also prove that these matrices preserve angles between any two vectors in the Euclidean space.

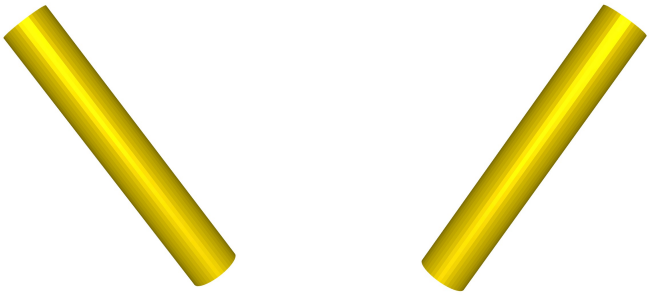

Figure 1.3: Global rigid transformation(rotation) on the bar mesh

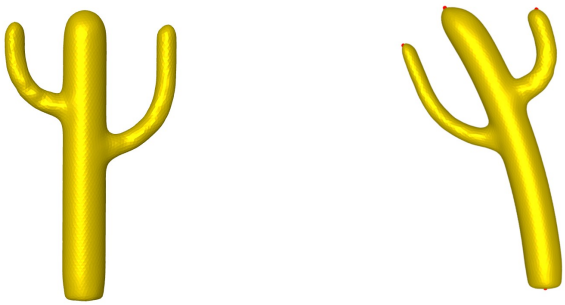

Figure 1.4: Cactus mesh(Left), Global non-rigid deformation on cactus(Right)

1.1.3 How do we deform a mesh?

In most cases, global rigid transformations of the mesh are not enough. Our task is to get a non-rigid deformation of the mesh given rigid transformations at some points such that the deformation is intuitive or aesthetic. So we are given a mesh $M(V, F)$, few points $v_h \in V, h \in \{1, 2, 3...m\}$ as handle points and transformations for each handle point point $T_h, h \in \{1, 2, 3...m\}$. We want a mesh that contains the transformed vertices and preserves the object's shape as best as possible. There are two analogies in mesh deformation. The time when we are specifying handle points and the weights at each vertex w.r.t. handles is known as bind time. Whereas the time at which we are allowing the mesh to generate the new deformed version is known as pose time. There are three classes of deformation methods that are found in the literature.

- Skinning-based Methods:- Skinning-based methods are extremely fast and the idea of these methods is to interpolate rotation matrices of handles to other points. There are variants like Linear Blend Skinning, Quaternion Blend Skinning, etc. These methods are highly used in animation tasks.

- Optimization-based Methods:- These methods are typically slower than Skinning-based methods because they use optimization at pose time. However, they are used to produce more accurate and intuitive deformations, typically in cases where skinning-based methods would fail.

- Physics-based Methods:- We can model the deformations by encoding actual physics information like elasticity, plasticity, etc. of the material and then perform deformations on it. These methods are extremely slow and use a lot of computational power but give the best results. They are used in problems like cloth draping, etc.

In the next chapter, we introduce a mesh deformation method based on optimization which gives very good results.

Chapter 2

As-Rigid-As-Possible(ARAP) surface modeling

We want to perform a global non-rigid deformation for tasks like animation. Skinning-based methods are fast but often fail to give very intuitive results. Physics-based methods give very good results but they are very slow for deforming a mesh in real time. ARAP is an optimization-based method that gives good results. It is one of the most famous mesh deformation methods. The key idea of ARAP is approximating a global non rigid transformation through locally rigid transformation and improving the approximation iteratively. ARAP is best understood as a two-step process.

- Locally, we make the transformations as rigid as possible. These transformations cannot be directly applied to the local neighborhood because the surface can just break!
- So the second step is that we use these local rotations and do a global

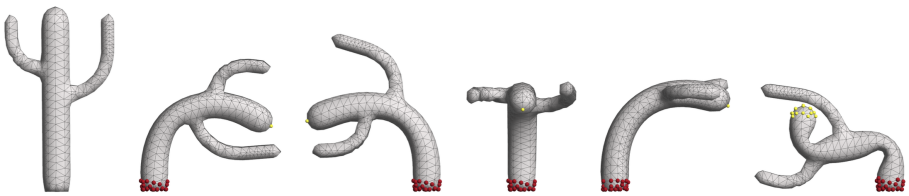

Figure 2.1: O. Sorkine and M.Alexa, As-Rigid-As-Possible Surface Modeling(2004)

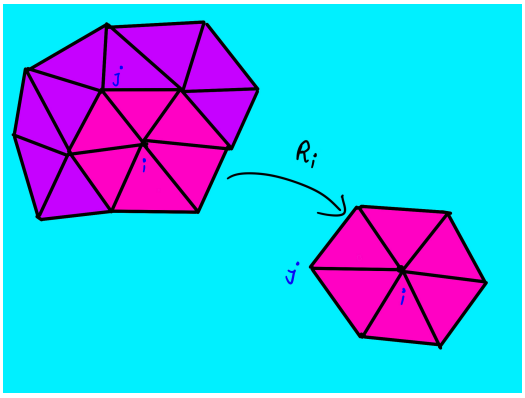

Figure 2.2: Local rigid transformation may not preserve mesh structure

step to get a connected surface.

2.0.1 Local Step: Estimating rigid transformations

So let's first look at the local step where we are estimating a local rigid rotation between the deformed mesh and the original template mesh. We want to find a rotation $\mathbf{R}_i$ for the cell(1-ring neighborhood) transformation $\mathcal{C}_i \rightarrow \mathcal{C}'_i$ such that $(\mathbf{p}'_i - \mathbf{p}'_j) = \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)$, $\forall j \in \mathcal{N}(i)$.
We can convert this to an energy function

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

The cell weights come from the discrete Laplace-Beltrami Operator on triangle meshes. For this case, we are using the cotan Laplacian for defining the cell weights as $w_{ij} = \frac{1}{2A_i}(cot\alpha_{ij} + cot\beta_{ij})$. Here each edge is an indicator of how much close it is to being a part of a good triangle for the triangular mesh. A good triangle is a triangle which satisfies the delaunay triangulation property, or basically if the triangles to which it is adjacent should be close to being equilateral for the edge to be weighted more. So we can look at this energy function as a means of approximating the transformation for each vertex. Let's look at steps to find these rotations. We want to minimize the energy function $E\left(\mathcal{C}_i, \mathcal{C}'_i\right)$ over $\mathbf{R}_i$ such that $\mathbf{R}_i\mathbf{R}_i^T = \mathbf{I}$. Putting $\mathbf{p}'_i - \mathbf{p}'_j = \mathbf{e}'_{\mathbf{ij}}$ and $\mathbf{p}_i - \mathbf{p}_j = \mathbf{e}_{\mathbf{ij}}$ and removing the terms which do not contain $\mathbf{R_i}$ we can

reduce the problem to

$$\underset{\mathbf{R}_i}{\operatorname{argmax}} \operatorname{Tr}\left(\mathbf{R}_i \sum_j w_{ij} \mathbf{e}_{ij} \mathbf{e}_{ij}'^T\right)$$

Putting X and Y as the matrices with $\mathbf{e}_{ij}$'s and $\mathbf{e}'_{ij}$'s as row vectors we have

$$\underset{\mathbf{R}_i}{\operatorname{argmax}} \operatorname{Tr}\left(\mathbf{R}_i X W Y^T\right)$$

$S = XWY^T$ is a covariance matrix and we want to find R_i such that it is maximally aligned with the basis of S. We do a Singular Value Decomposition of $S = U\Sigma V^T$. Rearranging the matrices we have

$$\underset{\mathbf{R}_i}{\operatorname{argmax}} \operatorname{Tr}\left(\Sigma V^T \mathbf{R}_i U\right)$$

The above equation is maximized when $\mathbf{R}_i = VU^T$. We should also take care about the property that $det(R) = 1$ for rotation matrices as we have not explicitly added that constraint anywhere. For that we will add a matrix M in $\mathbf{R}_i = VMU^T$ where M is a diagonal matrix with all entries 1 except $M(-1,-1) = det(VU^T)$ which will make sure that the determinant of the matrix is 1. For more information we refer the reader to [3]

2.0.2 Global Step: Finding new coordinates given rotations

We will have to devise a method to find global transformation given rigid transformations. The idea for this is to minimize the energy function over the whole mesh. We have the new energy function over the whole mesh as

$$\begin{aligned} E\left(\mathcal{S}'\right) &= \sum_{i=1}^{n} w_i E\left(\mathcal{C}_i, \mathcal{C}'_i\right) = \\ &= \sum_{i=1}^{n} w_i \sum_{j \in \mathcal{N}(i)} w_{ij} \left\| \left(\mathbf{p}'_i - \mathbf{p}'_j\right) - \mathbf{R}_i \left(\mathbf{p}_i - \mathbf{p}_j\right) \right\|^2 \end{aligned}$$

Cells with larger areas are more important hence $w_i = A_i \ \forall i$, where A_i is the Voronoi area of the cell $\mathcal{C}'_i$. We find the gradient of $E(\mathcal{S}')$ w.r.t the points $\mathbf{p}'$ to find the optimal vertices given rotations.

$$\begin{aligned}\frac{\partial E\left(\mathcal{S}'\right)}{\partial \mathbf{p}_i'} &= \frac{\partial}{\partial \mathbf{p}_i'}\left(\sum_{j\in\mathcal{N}(i)} w_{ij}\left\|\left(\mathbf{p}_i'-\mathbf{p}_j'\right)-\mathbf{R}_i\left(\mathbf{p}_i-\mathbf{p}_j\right)\right\|^2+\right.\\ &\left.+\sum_{j\in\mathcal{N}(i)} w_{ji}\left\|\left(\mathbf{p}_j'-\mathbf{p}_i'\right)-\mathbf{R}_j\left(\mathbf{p}_j-\mathbf{p}_i\right)\right\|^2\right)\\ &= \sum_{j\in\mathcal{N}(i)} 2w_{ij}\left(\left(\mathbf{p}_i'-\mathbf{p}_j'\right)-\mathbf{R}_i\left(\mathbf{p}_i-\mathbf{p}_j\right)\right)+\\ &+\sum_{j\in\mathcal{N}(i)} -2w_{ji}\left(\left(\mathbf{p}_j'-\mathbf{p}_i'\right)-\mathbf{R}_j\left(\mathbf{p}_j-\mathbf{p}_i\right)\right)\end{aligned}$$

We also have $w_{ij} = w_{ji}$, therefore

$$\frac{\partial E\left(\mathcal{S}'\right)}{\partial \mathbf{p}_i'} = \sum_{j\in\mathcal{N}(i)} 4w_{ij}\left(\left(\mathbf{p}_i'-\mathbf{p}_j'\right)-\frac{1}{2}\left(\mathbf{R}_i+\mathbf{R}_j\right)\left(\mathbf{p}_i-\mathbf{p}_j\right)\right)$$

Setting the partial derivatives to zero w.r.t. each $\mathbf{p}_i'$

$$\sum_{j\in\mathcal{N}(i)} w_{ij}\left(\mathbf{p}_i'-\mathbf{p}_j'\right) = \sum_{j\in\mathcal{N}(i)} \frac{w_{ij}}{2}\left(\mathbf{R}_i+\mathbf{R}_j\right)\left(\mathbf{p}_i-\mathbf{p}_j\right)$$

This left side of the equation is the **discrete Laplace-Beltrami Operator** applied to $\mathbf{p}_\mathbf{i}'$. We can we write the equations for all vertices as

$$\mathbf{L}\mathbf{p}' = \mathbf{b}$$

The problem with this framework is that we don't have either $\mathbf{R_i's}$ or $\mathbf{p_i's}$. For that initially, we assume the template mesh to be the original mesh and the deformed mesh as the mesh with just the handle points shifted. We then find optimal rotations for each vertex, solve the abovementioned equations, obtain the new deformed vertices, and iterate. At a glance we would follow these steps

Algorithm 1 Iterative Mesh Deformation Algorithm

1: Compute the Cotan Laplacian $\mathbf{L}$ for the mesh.
2: Using the original mesh M and modified mesh M' with changes at handle points, compute the rotation matrices for each cell.
3: Use the rotation matrices to compute the vector $\mathbf{b}$.
4: Solve the system of equations $\mathbf{Lp'} = \mathbf{b}$ to obtain the new vertices $\mathbf{p'}$.
5: Update M' with $\mathbf{p'}$ as the new mesh for the next iteration.
6: Repeat steps 2 to 5 until convergence.

Chapter 3

Results

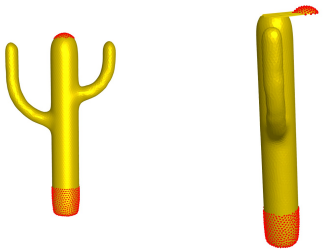

Figure 3.1: Cactus mesh with handle points in red (Left), Cactus mesh with handle deformations (Right).

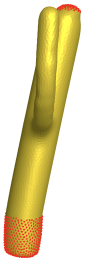

Figure 3.2: Deformed mesh after the 5th iteration of ARAP.

Chapter 4

Future Work

Making deformations realistic is a harder and more important task for most applications. There have been various improvements to ARAP to deform the mesh in a realistic manner like [2] which tries to look at a larger neighborhood instead of just 1-ring for each vertex deformation this sort of averages the edge transformation and works as if locally this new part(r-ring neighborhood) is a rigid part. This also does not change the deformation framework but requires additional computational resources. Estimating rigidity then is a simpler task, we look at different deformations and try to approximate the r for the r-ring neighborhood at each vertex by minimizing the ARAP energy. [4] proposed using

$$E\left(\mathcal{C}_k, \mathcal{C}'_k\right) = \sum_{(i,j)\in\mathcal{E}(k,r)} w_{ij} \left\| \left(\mathbf{p}'_i - \mathbf{p}'_j\right) - \mathbf{T_k}\left(\mathbf{p}_i - \mathbf{p}_j\right) \right\|_2^2 + \lambda \left\| \mathbf{T_k}^{\mathbf{T}} \mathbf{T_k} - \mathbf{I} \right\|_F^2$$

Adding to the rigidity-informed ARAP technique we can allow a finer rigidity control using this energy function. Instead of allowing just rotations, we allow all the linear transformations and then penalize them according to how non-rigid they are. However, in this case, finding local rotations is not a trivial task and requires an iterative gradient descent algorithm. Estimating λ in this case is something we would like to explore in future.

Bibliography

[1] Olga Sorkine and Marc Alexa. (2007). As-rigid-as-possible surface modeling. In Proceedings of the fifth Eurographics symposium on Geometry processing (SGP '07). Eurographics Association, Goslar, DEU, 109–116.

[2] Chen, Shu-Yu Gao, Lin Lai, Yu-Kun Xia, and Shihong. (2017). Rigidity Controllable As-Rigid-As-Possible Shape Deformation. Graphical Models. 91. 10.1016/j.gmod.2017.02.005.

[3] Olga Sorkine-Hornung and Michael Rabinovich. (2017). Least-Squares Rigid Motion Using SVD.

[4] Darshil Patel, Material-based Mesh Deformation, 2022