Laplacian Deformation on 2D/3D Mesh Editing

Chen Pan, Isabella Liu, Zhuoqun Robin Xu, Tongchuan Shen

March 2021

1 Abstract

In this project we explore the application of Laplacian deformation on mesh editing, as well as propose several possible ways to optimize the existing algorithm. Since Laplacian coordinate describes surface's local intrinsic property well, an optimization of Laplacian coordinate during non-rigid transformation on surface can greatly decrease the unreasonable distortion and preserve the local property of each point. We performed the experiment in both 2D and 3D and attached our result as a demo video here.[1]

2 Introduction

For surface editing in computer graphics, surfaces are usually presented in euclidean coordinates of points. However, such representation while explicitly specify their global positions, does not realize local properties for the surface. If we do mesh editing using such kind of global coordinate system, object's local geometric properties may not be preserved well and therefore cause unreasonable distortion. [Sor+] first proposed the method to use Laplacian coordinate to represent each points of a surface - Laplacian coordinates represent a point relative to its neighbors. This allows us to maintaining local geometric properties while editing the surface. In our project, we adopt the method from [Sor+] as our start point and then explored some possible ways to optimize current algorithm.

In the remaining parts we will first look at the problem definition, including the concept of the Laplacian coordinate and problem formulation. In the experiment part we will briefly introduce our experimental details and present part of our results. Finally, we examine the whole process, make conclusion and introduce some potential improvement to be achieved in the future.

[1] Please refer to our demo video: https://www.youtube.com/watch?v=R90fUuZJFXU

3 Problem Definition

3.1 Laplacian coordinate

Definition 1. *Given a mesh $\mathcal{M}$ described by (V, K), where V describes the vertices and K describes the connectivity of each pair of vertices, for a point $v_i \in V$, let $N(v_i)$ be the vertices connected to v_i, then the Laplacian coordinate is defined as:*

$$\delta_i = v_i - \frac{1}{d_i} \sum_{j \in N(v_i)} v_j \tag{1}$$

Where $d_i = |N(v_i)|$ is the degree of v_i.

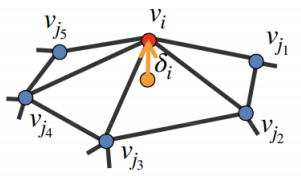

Figure 1: Illustration of Laplacian coordinate on a point

Note that the transformation between normal coordinate and the Laplacian coordinate is linear and can be described by a matrix.
Let A be the adjacency matrix of V and diagonal matrix D be the degree matrix, then from $\delta_i = v_i - \frac{1}{d_i} \sum_{j \in N(v_i)} v_j$, we can get

$$\Delta = (I - D^{-1}A)V \tag{2}$$

We define $L = I - D^{-1}A$ be the Laplacian operator [Tau], then

$$\Delta = LV \tag{3}$$

3.2 Problem Formulation

Our goal is to maintain the local geometric properties of original surface as mush as possible, while performing surface deformation and avoiding unreasonable distortion.
Given a mesh $\mathcal{M}$. We want to move a set of vertices to a new location, let $U = \{u_i\}$ be the constrain (target location), and δ_i' be the Laplacian coordinate of the transformed mesh $\mathcal{M}' = (V', K)$, we want to minimizing the change of Laplacian coordinate while moving the vertices to the desired location, the error functional is defined below and our problem is to minimize this error function.

$$E(V') = \sum_{i=1}^{n} \|\delta_i - \delta_i'\|^2 + \sum_{i=m}^{n} \|v_i' - u_i\|^2 \tag{4}$$

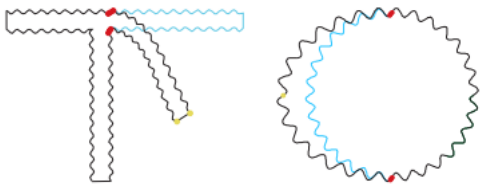

Figure 2: The yellow points are the *control points*, the red points are the *anchor points*. We hope to move the control points to the desired location while fix the anchor points as much as possible. Illustration pictures borrowed from [Sor+].

3.3 Other Optimization Details

3.3.1 Solving the Problem of Laplacian Coordinate's Sensitivity to Linear Transformation

As indicated by [Sor+], the Laplacian coordinates are invariant under translation, but they are not preserved under linear transformation such as rotation. In other word, they are sensitive to linear transformation and will cause some unwanted distortion while doing the deformation.

To solve such problem, we use the method from [Sor+], which is compute an appropriate transformation T_i for each vertex v_i base on the new vertexes V', i.e. T_i is a function of V_i, and we reformulate the error function as

$$E(V') = \sum_{i=1}^{n} \|T_i(V')\delta_i - \delta_i'\|^2 + \sum_{i=m}^{n} \|v_i' - u_i\|^2 \tag{5}$$

Since T_i is a linear function of V', solving for V' implicitly gives us T_i. We can view $E(V')$ as a quadratic function of V'.

To properly solve for V', we need to constrain T_i, otherwise the natural minimizer for $E(V')$ is a membrane solution and the geometric properties will be lost. So we want to limit T_i to **translations, rotations**, and **isotropic scales**. The class of isotropic scales and rotations can be expressed as $T = exp(H)$, where H is a skew symmetric matrix. And the translational part can be added by using homogeneous coordinates.

Since in 3D, skew-symmetric matrix can be represented by vector cross product, we can write T as:

$$T = s \exp H = s(\alpha I + \beta H + \gamma \mathbf{h}^T \mathbf{h}) \tag{6}$$

Where $Hx = \mathbf{h} \times x$.
Note that in 2D scenerios, T_i can be characterized with linear expression

$$T_i = \begin{pmatrix} a & w & t_x \\ -w & a & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{7}$$

While in 3D, we use linear approximation for $\mathbf{h}^T\mathbf{h}$ and get

$$T_i = \begin{pmatrix} s & -h_3 & h_2 & t_x \\ h_3 & s & -h_1 & t_y \\ -h_2 & h_1 & s & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

Now let $z_i = (s_i, \mathbf{h}_i, \mathbf{t}_i)^T \in \mathrm{R}^7$ be the vector of unknowns in T_i, we can rewrite equation (5) as

$$\|A_i z_i - b_i\|^2 \tag{9}$$

Where A_i contains the position of v_i and its neighbors. b_i contains position of v_i' and its neighbors. Such minimization problem can be solved by

$$z_i = (A_i^T A_i)^{-1} A^T b_i \tag{10}$$

We can also write the dual of our problem from equation (9), since $L = \|A_i z_i - b_i\|^2$, the dual problem is

$$\max_{\lambda} \min \|A_i z_i - b_i\|^2 = \min \|A_i z_i - b_i\|^2$$

(equation (9) is a unconstrained problem for z_i)

3.3.2 Enforce Rotation Matrix to $SO(3)$

While the method in section 3.3.1 gives a good representation of rotation matrix, in our experiment (mentioned in Section 4) we use cvxpy and want fewer optimization variable. Therefore in 3D, we use the Umeyama's method [Ume] to enforce any optimized 4 matrix to be a valid rotation matrix.

3.3.3 Reduce Computational Complexity

Another optimization we have performed here is to omit the computation for the vertices that won't be moved. Consider the following example, we will only move the vertices between the anchors. In our algorithm, we only compute the movement of these vertices, and concatenate the results with the positions of the unmoved vertices. In this way, we have largely reduced the computation cost.

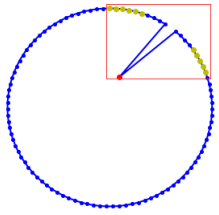

Figure 3: The original shape is a unit circle, red point is the control point. For illustration we have moved it to the desired location. Yellow points are the anchor points.

4 Experiments

We performed the experiment in both 2D and 3D. For 2D, we selected several hand-drawing pictures from the Internet, run basic image-processing steps to get 2D point cloud from the picture. Then calculate the degree matrix (D) and adjacency matrix (A) and compute the normalized Laplacian matrix (L). We then use the cvxpy package to do the optimization steps. Details of the deformation result can be found in the attached video file. In 3D, we choose Houdini 18.5 as our mesh processing tool and python + cvxpy as our optimization tool. We run our optimized deformation on the low-poly Stanford rabbit model. Details of the result can be found in the video as well.

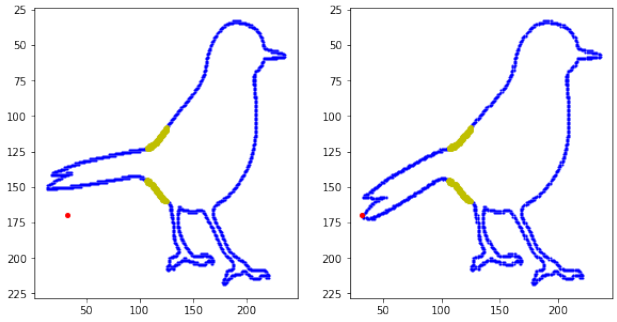

Figure 4: Illustration of our 2D experiment. After indicating the desired new location of the control point (red point) and anchor points (yellow points), the bird's tail is moved smoothly from the original pose to the new pose without any strange distortion. Local geometric properties are also preserved well during the deformation.

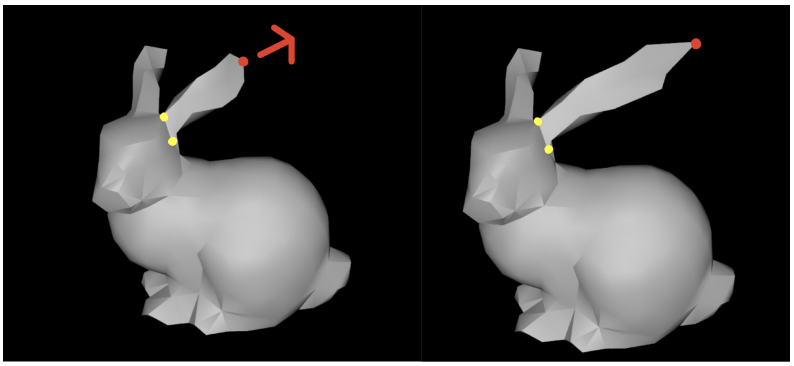

Figure 5: Screenshots from our video, red point is the control point, yellow points are the anchor points. All the intermediate steps are generated automatically by our algorithm. There is no unreasonable distortion, local properties of vertices are preserved well during the deformation.

5 Conclusion and Future Work

Our algorithm performs deformation on 2D shapes and 3D meshes smoothly while preserving the local geometric properties well. Our optimization makes it run quite efficiently. In the future, we can apply our 2D optimization technique in 3D scenarios, which means we can just compute part of the vertices that require deformation instead of calculating for the whole point set. This algorithm can also be applied to graphics processing tools. We can use this algorithm to implement a program such that we can freely deform the figure by dragging around the control point.

6 Individual Contribution

Chen implemented an optimization of this algorithm and a pipeline to generate experiment results. He also worked on the optimization detail, experiments, and conclusion and future work sections in this report.
Isabella implemented the algorithm in both 2D and 3D, generated the experiment results. She also worked on the final report revision.
Robin implemented the procedural visualization process of deforming experiment and animation generation. He also worked on the video demo of the final report.
Tongchuan is responsible for the introduction and problem formulation in this report.

References

[Sor+] Olga Sorkine et al. *Laplacian Surface Editing*. URL: `http://people.eecs.berkeley.edu/~jrs/meshpapers/SCOLARS.pdf`.

[Tau] Gabriel Taubin. *A Signal Processing Approach To Fair Surface Design*. URL: `http://mesh.brown.edu/taubin/pdfs/taubin-sg95.pdf`.

[Ume] Shinji Umeyama. *Least square Estimation of Transformation parameters Between Two point patterns*. URL: `http://web.stanford.edu/class/cs273/refs/umeyama.pdf`.