

COMP5411 Project Report: Naive Laplacian Deformation

1. Introduction

Mesh deformation is the practice of manipulating shapes of objects via deformation of the given surface of objects. Common examples include, but not restricted to, deformation of elastic balls when bounced, or simple movement of characters' limbs, etc. While there may exist a direct approach that changes the displacements vertex-by-vertex, the computation overhead is too expensive for immediate and continuous animated response. Another approach is to create in-between shape linear interpolation for deformations. However, in many cases, deformations and distances are non-linear, e.g. curved planes, leading to unnatural deformation results. This project aims to implement the Laplacian-based surface editing algorithm, proposed by O. Sorkine et. al., which utilizes the intrinsic surface representation that preserves local surface details during surface editing. The methodology section explains the algorithm and its mathematical theory behind. In the result section, we will show the screenshots of the implementation results of some deformed mesh shapes. Lastly, we will discuss some of the limitations of this algorithm in the discussion section.

2. Methodology

Let V denote all the vertices in the closed mesh, where $V = \{v_1, \dots, v_n\}$ describes all the geometric positions of vertices in \mathcal{R}^3 . Let N describe the connectivity of the set of vertices V in the mesh, where $N(i) = \{j \mid (v_i, v_j) \text{ is an edge in the mesh}\}$. During the mesh deformation, instead of keep updating with the absolute position of each vertex, we store the geometric relationship between vertices by pre-computing a set of $\Delta(V)$, where $\Delta = \{\delta_1, \dots, \delta_n\}$ is the differential coordinates, and each δ_i corresponds to each $v_i \in V$. The computation of $\delta_i(V)$ is given by the following:

$$\delta_i(V) = v_i - \sum_{j \in N(i)} w_j v_j$$

Specifically, $\delta_i(V)$ describes the sum of the weighted difference with respect to its neighbors. The intuition behind represents the local shape information of each vertex. In our implementation, we use the cotangent weight function to calculate the weight for each neighbor v_j . The following is the cotangent weight function:

$$w_j = \frac{1}{2} (\cot \alpha + \cot \beta)$$
$$\alpha = \angle v_i v_{j+1} v_j, \beta = \angle v_i v_{j-1} v_j$$

For simplicity, we will use a matrix L to describe the Δ since each $\delta_i(V) \in \Delta$ is a linear combination of a set of $v_j \mid j \in N(i)$, i.e. $\delta_i = L(v_i)$.

In practice, we assume that there is a set of user-specified fixed points that acts as “handles” during the mesh transformation. We represent the set of fixed points as U , where $U = \{v_c \mid \forall v_c \in V, v_c \text{ is fixed}\}$, $|U| \leq |V|$. Let \tilde{L} be the augmented L that encodes the fix point information, $(\Delta \mid U)^T = \tilde{L}(V^T)$. As a result, the deformation of mesh is sufficient to solving the following problem:

$$\text{Solve for } V': (\Delta \mid U')^T = \tilde{L}(V'^T)$$

In detail, U' represents the updated set of fixed points which is specified by the user's moving of handles. We aim to solve for a new set of deformed vertices V' such that it preserves the local shape information of fixed points and differential coordinates. To solve this linear equation, it suffices to solve the following two variants:

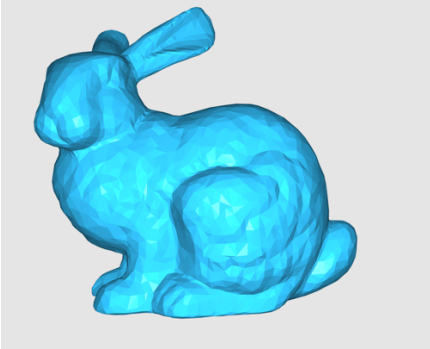
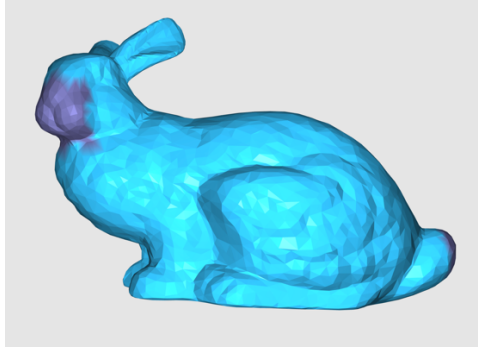
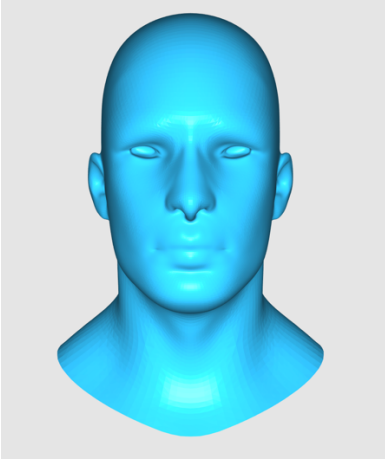
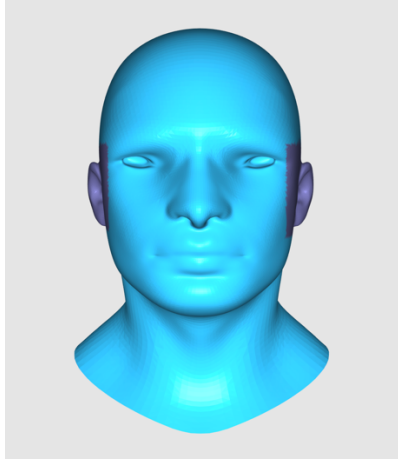
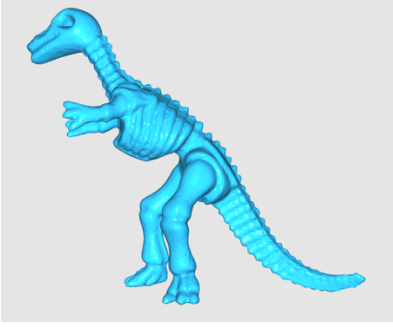
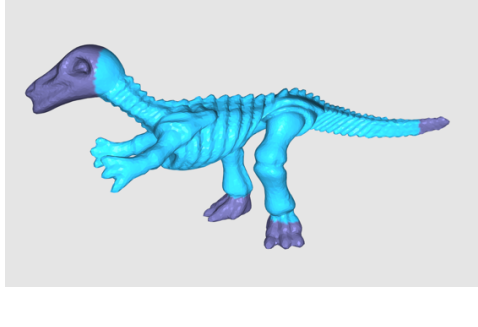
$$\text{Solve for } V': \tilde{L}^T (\Delta | U')^T = \tilde{L}^T \tilde{L} (V'^T)$$

$$\text{i. e. Solve for } V' = \operatorname{argmin} \left(\sum_{v'_i \in V'} |L(v'_i) - \delta_i|^2 + \sum_{v_c \in U} |v'_c - v_c|^2 \right)$$

From observation, since the matrix $\tilde{L}^T \tilde{L}$ is sparse and symmetric positive definite, there exists a solution for the above least square error equation, which can be solved by Cholesky factorization. We use the Eigen package in our C++ implementation to solve this linear equation.

3. Results

The following are some results of the mesh deformation using the Laplacian-based surface editing algorithm that is proposed by O. Sorkine et. al. We will show the screenshotted deformation results of the three object meshes, namely, “bunny.obj”, “face.obj” and “dinosaur.obj”. The purple regions are the user-specified fixed points, which are used as deformation handles in the application.

	Original	After deformation
“bunny.obj”		
“face.obj”		
“dinosaur.obj”		

4. Discussions

According to the above results, our naïve Laplacian deformation achieves translation invariance well. As our algorithm aims to preserve the global orientation of each vertex in the mesh, the transition will be smooth. However, this also implies that the normals, i.e. differential coordinates, for each vertex is fixed. Consequently, this algorithm does not support rotation invariance when there is larger deformation. This bright side is that, in some cases we do want to achieve rotation invariance, and this naïve Laplacian-based algorithm is a simple approach. For example, we want the dinosaur object's head to lower along with its body, instead of just turning its head down. The opposite side is that rotation operations cannot be achieved since the equation assumes that $\delta_i, \forall i$ is fixed.

5. Conclusion

We have discussed the mathematical model and implemented the code of intrinsic geometry representation for mesh deformation. Since all the differential coordinates are fixed at the beginning, the algorithm can reconstruct the new mesh conveniently by solving a least squared equation. In practice, first, users have to set up handles as fixed-point constraints. Then, the deformation can be performed on the region of interests to the direction of handles accordingly.

Reference

[SCLA04] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., & Seidel, H. P. (2004, July). Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (pp. 175-184).