# Reproduction Report of Laplacian Mesh Editing

Qu Yi(Albert)
The Chinese University of Hongkong, Shenzhen
Yi Qu@link.cuhk.edu.cn

November 30, 2024

## Abstract

This report presents the implementation[1] of a mesh editing tool based on the Laplacian operator[2]. The tool allows users to deform 3D meshes (such as PLY format meshes) by specifying boundary control points and manipulation handles. The program constructs a graph representation of the mesh, computes the Random Walk Laplacian matrix, and solves a linear system to perform the mesh deformation. The edited mesh can be exported as a new PLY file and visualized alongside the original mesh for comparison.

## 1 Introduction

Mesh editing based on Laplacian systems is a technique used for manipulating 3D surface meshes while preserving the smoothness and local geometry of the surface. This project implements a mesh editing tool that allows users to interactively specify control points and manipulation handles on a mesh, and then apply the Laplacian system to deform the mesh accordingly. The algorithm utilizes the Random Walk Laplacian (RW-Laplacian) to construct a linear system and solve for the new vertex positions that respect the boundary conditions.

## 2 Project Overview

### 2.1 Functionality

The main functionality of the project includes:

- Loading 3D mesh models in PLY format.
- Defining mesh editing regions using control points and boundary points.
- Constructing a linear system for mesh editing using the Laplacian matrix.
- Solving the system using least squares and updating the mesh vertex positions.
- Exporting the edited mesh to a new PLY file.
- Visualizing the results by overlaying the original and edited meshes.

### 2.2 Key Algorithms

- **Graph Representation and Laplacian Matrix**: The mesh is represented as a graph, where vertices correspond to nodes and edges to graph edges. The Laplacian matrix is used to describe the relationships between nodes and is essential for solving the mesh deformation problem.
- **Constraints and Linear Solving**: The program incorporates boundary control points and manipulation handles as constraints in the linear system. The system is solved using a least squares method to compute the new vertex positions.
- **Visualization and Output**: The program utilizes the ‘trimesh‘ library for 3D mesh visualization, enabling users to see the effects of the mesh deformation before and after applying the Laplacian editing.

## 2.3 Installation and Dependencies

This project requires the following Python libraries:

- `numpy`: For numerical computations.
- `scipy`: For linear algebra and sparse matrix operations.
- `networkx`: For graph operations and calculating the graph representation of the mesh.
- `trimesh`: For loading, manipulating, and exporting 3D meshes.
- `time`: For generating timestamps and naming files.

## 2.4 Usage

### 2.4.1 Prepare the Mesh File

Ensure you have a valid PLY format mesh file (e.g., `bunny.ply`). The program defaults to loading the mesh file from `./meshes/bunny.ply`, but you can modify the file path as needed.

### 2.4.2 Edit Parameters

In the code, define the control points and manipulation handles. Control points define the boundary of the mesh, while manipulation handles are used to edit the mesh. Modify the following dictionary as per your requirements:

```
control_handles = {
    24092: [-0.01134, 0.151374, -0.0242688]
    # Manipulation handle ID and purpose position
}
boundary_control_points = [15617, 24120, 30216, 11236, 6973]
# Boundary control point IDs
```

### 2.4.3 Run the Program

To run the program, simply execute the following script:

```
python mesh_editing.py
```

### 2.4.4 Output File

The program will save the edited mesh as a new PLY file, named with a timestamp. For example:

```
./exports/bunny_20231129124523.ply
```

### 2.4.5 Visualizing Results

The program also generates a ‘trimesh‘ scene containing the original mesh, the edited mesh, and the positions of the control points and manipulation handles. You can visualize the overlaid original and edited meshes to compare the deformation effect.

### 2.4.6 Optional Parameters

If you need to modify the output path or control point settings, you can edit the following section in the code:

```
mesh = trimesh.load('./meshes/bunny.ply', process=False)
output_filename = './exports/bunny_%s.ply' %
(time.strftime("%Y%m%d%H%M%S", time.localtime()) )
```

You can replace the file paths and names as needed.

# 3 Mathematical Derivation

In this section, we provide the mathematical formulation of the Laplacian coordinates used in the mesh editing process.

## 3.1 Fitting Transformed Laplacian Coordinates

Let the mesh $M$ be described by a pair $(K, V)$, where $K$ describes the connectivity of the mesh, and $V = \{v_1, \ldots, v_n\}$ represents the geometric positions of the vertices in $\mathbb{R}^3$. We use the following terminology:

- The neighborhood ring of vertex $i$ is the set of adjacent vertices $N_i = \{j|(i, j) \in K\}$.
- The degree $d_i$ of vertex $i$ is the number of elements in $N_i$.
- We assume that the mesh is connected.

Instead of using the absolute coordinates $V$, we wish to describe the mesh geometry using a set of differentials $\Delta = \{\delta_i\}$. Specifically, the coordinate $i$ will be represented by the difference between $v_i$ and the average of its neighbors:

$$\delta_i = L(v_i).$$

For simplicity, we define $L$ with uniform weights:

$$L(v_i) = v_i - \frac{1}{d_i} \sum_{j \in N_i} v_j.$$

These weights have been shown to be sufficient in all our experiments. However, our approach is not dependent on the particular choice of $L$. For instance, cotangent weights (see, e.g., [**?**]) could accommodate extremely non-uniform tessellations, and their application is straightforward.

The transformation between $V$ and $\Delta$ can be described in matrix algebra. Let $A$ be the adjacency matrix of the mesh, and $D = \text{diag}(d_1, \ldots, d_n)$ be the degree matrix. Then:

$$\Delta = LV,$$

where $L = I - D^{-1}A$ for the uniform weights. The matrix $L$ is commonly considered as the Laplacian operator of the mesh with connectivity $A$ [**?**, **?**], which is why we call $\delta_i$ the Laplacian coordinate of vertex $i$.

Laplacian coordinates are invariant under translation but sensitive to linear transforms. The matrix $L$ has rank $n - 1$, which means that $V$ can be recovered from $\Delta$ by fixing one vertex and solving a linear system.

### 3.2 Fitting Laplacian Coordinates

To perform mesh editing using Laplacian coordinates $\Delta$, the approach is to fix the absolute position of several vertices $\{u_i\}$ and solve for the remaining vertices $\{v'_i\}$ by fitting the Laplacian coordinates of the geometry $V'$ to the given Laplacians $\Delta$. Specifically, we aim to minimize the following error functional:

$$E(V') = \sum_{i=1}^{n} \|\delta_i - L(v'_i)\|^2 + \sum_{i=m}^{n} \|v'_i - u_i\|^2.$$

This quadratic minimization problem results in a sparse linear system of equations, which is solved to obtain the new vertex positions.

### 3.3 Transformation of Laplacian Coordinates

The main idea of our approach is to compute an appropriate transformation $T_i$ for each vertex $i$ based on the new configuration of vertices $V'$. The transformation $T_i(V')$ is a function of $V'$, and the error functional becomes:

$$E(V') = \sum_{i=1}^{n} \|T_i(V')\delta_i - L(v'_i)\|^2 + \sum_{i=m}^{n} \|v'_i - u_i\|^2.$$

Here, both $T_i$ and $V'$ are unknown, but if the coefficients of $T_i$ are linear functions in $V'$, then solving for $V'$ implies finding $T_i$ (though not explicitly), as $E(V')$ is a quadratic function in $V'$.

To define $T_i$, we minimize the following expression for each vertex $i$:

$$\min_{T_i} \left( \|T_i v_i - v'_i\|^2 + \sum_{j \in N_i} \|T_i v_j - v'_j\|^2 \right).$$

This minimization problem ensures that the transformation is a linear function of $V'$. The translational part of $T_i$ is introduced using homogeneous coordinates, and the linear part should satisfy the constraints for isotropic scaling and rotation. The transformation matrix $T$ is of the form $T = s\exp(H)$, where $H$ is a skew-symmetric matrix.

### 3.4 Linear System and Solution

The algorithm constructs a linear system using the Laplacian matrix $L$, which describes the relationship between the vertices and their neighbors. The system is solved using least squares optimization to find the new vertex positions that minimize the deformation energy.

## 4 Results

The following figures show the results of the mesh editing algorithm applied to a sample bunny mesh.

### 4.1 Test

Source Mesh Edited Mesh

Figure 1: Source Mesh and Edited Mesh

The Test example shows an edit to a bunny(Source Mesh) by assigning the control point `24092` in the PLY format mesh file `bunny.ply` to the position `[-0.01134, 0.151374, -0.0242688]`. And the Edited Mesh is the result after assigning.

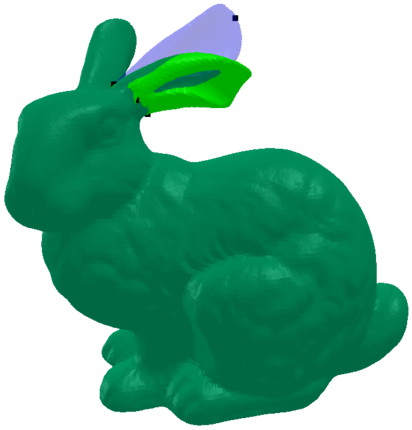

Figure 2: The front of The Overlay of Before and After Editing

Figure 3: The back of The Overlay of Before and After Editing

In Figures 2 and 3, the transparent blue part represents the original mesh, while the green part represents the edited mesh. The points marked at the top of the bunny's ear in the original mesh are the handle point `24092`, and the points marked at the top of the ear in the edited mesh are the target points(whose position is `[-0.01134, 0.151374, -0.0242688]`). The ring of points marked at the base of the bunny's ear represents the boundary control points `15617, 24120, 30216, 11236, 6973`.

## 5 Conclusion

The Laplacian Surface Editing algorithm successfully deforms 3D meshes while preserving the overall geometry. The method allows users to specify control points and boundary conditions, ensuring that the mesh deformation is smooth and consistent with the original structure. This tool can be applied to a wide range of 3D mesh editing tasks, from simple deformations to more complex mesh manipulations.

# References


[1] Shitong Luo. laplacian-surface-editing. `https://github.com/luost26/laplacian-surface-editing`, 2019. Accessed: 2024-11-30.

[2] Olga Sorkine, Daniel Cohen-Or, Yuval Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Eurographics Symposium on Geometry Processing*. Eurographics Association, 2004. Conference proceedings.