Laplacian Mesh Editing

HE Qihao

October 25, 2022

1 Introduction

This project implemented the Laplacian mesh editing technique based on the Laplacian Surface Editing [1].

2 Methodology

The deformer class was implemented using a least square equation overall, which is in the form:

$$A^T Ax = A^T b$$

where the original equation takes the form $Ax = b$ and the two sides are multiplied by A^T.

The first "Vertex" number of rows of matrix A consists of the Laplacian Matrix L storing topology of the mesh, corresponding to the RHS of the equation, first "Vertex" number of rows of vector b, which stores the Differential Coordinates calculated in the similar way as the moved distance was calculated in Laplacian Smoothing, where a cotangent weight is used.

The left rows in both matrix A and vector b indicate the specified "handle" vertices that will be manipulated by users when doing editing. And A stores the indices for such vertices (each row one index with 1), and b stores the position of the corresponding vertices at each remaining rows.

The vector x, after all, is the solution to our least-squares problem, which defines the vertices' positions after deformation, this is reasonable since we try to maintain the vertices' local geometry using the Laplacian matrix and differential coordinates, and also match the deformed shape using some "handle" vertices. The equation in essential tries to find the solution that fulfill the two constraints we give simultaneously.

After having defined the terms in the equation, we need to solve it efficiently, which introduces the CholeskySolver from the Eigen library. The solver will first factorize $A^T A$ into MM^T where M is an upper-triangular matrix in Deformer::buildSystemMat. Then we can call the solver in Deformer::deform to solve for vector $A^T b$ to obtain the desired deformed positions x, which is the solution to the least-squares problem.

One thing noticeable is that, the first "Vertex" number of rows in both A and b are filled in buildSystemMat, same for the left rows on handle indices in A, while the left rows corresponding to handle vertices' positions in b are filled in deform, since they get changed whenever the user is deforming the object by changing handle vertices' positions.

3 Results

Below are the results of for different objects after Laplacian editing:

Figure 1a shows deformed knight.
Figure 1b shows deformed Dinosaur.
Figure 1c shows deformed Bunny.
Figure 1d shows deformed Dinosaur.
Figure 1c shows deformed Bunny.
Figure 1d shows deformed Dinosaur.
And link to Demo Video.

4 Discussions

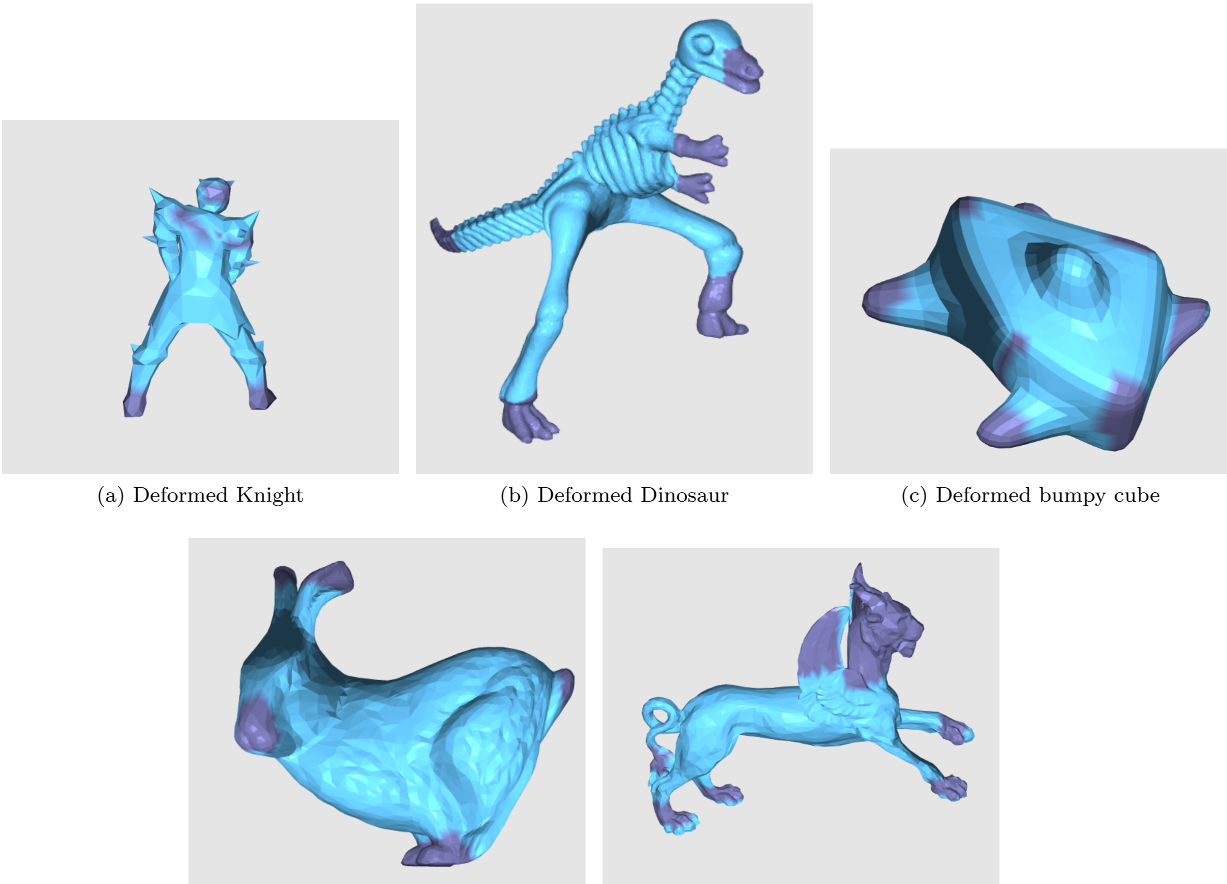
It is noticed that some further improvements on Laplacian editing of certain objects are desirable, take the dinosaur for example, its editing session takes a lot of time, for the complexity of its geometry, whenever a handle is dragged, it takes around 1 2 seconds for the deformation to take effects. Unlike the Knight or the Bumpy Cube which respond to editing very smoothly. As a result, it is believed that if mesh simplification is applied on certain complex meshes, their editing cost will be lowered significantly, thus resulting in a smoother editing session.

5 Conclusion

In conclusion, the Laplacian Mesh Editing technique is not very difficult to implement based on Laplacian smoothing, the additional thing that we need to figure out is the Least-Squares problem concept and how to solve it quickly with the help of matrix decomposition. Nevertheless, there is still space for further improvements when the mesh involved has a complex structure with many vertices. But for sure everything can now be deformed and moved around easily which is very cool.

References

[1] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of the EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. ACM Press, 2004.



(a) Deformed Knight (b) Deformed Dinosaur (c) Deformed bumpy cube

(d) Deformed bunny (e) Deformed feline

Figure 1: Deformations.