

Spectral Surface Deformation with Dual Mesh

Guodong Rong Yan Cao Xiaohu Guo

University of Texas at Dallas

{guodongrong | yan.cao | xguo}@utdallas.edu

Abstract

In this paper, we introduce a new surface deformation algorithm by combining the spectral method with the dual Laplacian editing algorithm. The dual Laplacian editing algorithm tries to maintain the Laplacian coordinates of all the vertices before and after deformation in dual space as much as possible. By using an iteration process, the local rotational effects can be achieved to get very natural results. In our new algorithm, we exploit manifold harmonics to perform the dual Laplacian editing in frequency domain. By using a small number of frequencies, the size of the linear system in each iteration is greatly decreased, and thus the speed is increased accordingly. Our experiments shows that our algorithm is particularly suitable for large meshes, and the speed of our algorithm is much faster than the spatial counterpart.

Keywords: Spectral Geometry, Surface Deformation, Dual Mesh, Manifold Harmonics

1 Introduction

Deformation of geometric meshes is a very basic topic in computer graphics, especially in areas such as computer animation and computer aided design. Usually, the user select some vertices (constraints) on a mesh and explicitly specify their final positions. The positions of the other vertices are calculated to achieve a “natural” result which satisfies these constraints. Generally speaking, a physically correct result often requires solving a complex non-linear system, and thus costs enormous computation time. However, for many real-world applications, a merely physical “plausible” result is good enough.

Previous work tries to approximate the physical deformation procedure with a linear system [1]. A common way to attain such a linear approximation is to maintain certain geometry properties, e.g. Laplacian coordinates [2, 3]. The Laplacian coordinate of every vertex can be represented by a linear combination of its one-ring neighbors and itself. So it is easy to build a linear system equalizing the Laplacian coordinates of all the vertices before and after deformation. Solving this linear system (in a least-square sense) gives the final positions of all the vertices after deformation. However, the local rotational effects cannot be well captured by a simple linear system. So the results tend to be unnaturally sheared for the parts with large displacement. The reason is that this method solves the final mesh using the final Laplacian coordinates. But the final Laplacian coordinates are determined by the final mesh, which is unknown. So this is a chicken-and-egg problem. To bypass this problem, Au et al. [4] use an iteration process to gradually correct Laplacian coordinates, and thus get a more natural final mesh. Later, they introduce the dual mesh of the original mesh to obtain a more robust algorithm and guarantee the convergency of the algorithm [5].

Besides the Laplacian-based algorithms, there are also some other approaches for the linear approximation of the physical deformation procedure, such as gradient-based algorithms [6], energy-based algorithms [7], local-frame-based algorithms [8], etc. For all these algorithms, the size of the linear system is $O(n)$, where n is the number of vertices in the mesh to be deformed. This prohibits them to be applied on large meshes. Subdivision surfaces can be used to build a hierarchy of coarser and coarser meshes to reduce the size of the linear system [9, 10]. However, the lack of automatic methods to build subdivision surfaces for arbitrary irregular meshes limits the usage of

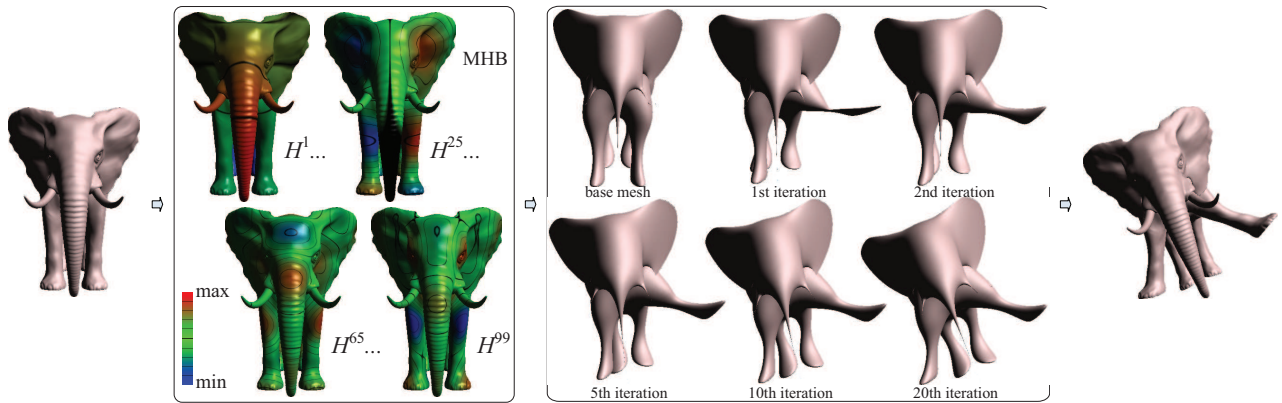


Figure 1: Pipeline of our algorithm. From left to right: the original mesh, the manifold harmonics bases, the smoothed mesh and the iteration process, the final mesh with details added back.

such approach. Another way to reduce the size of the linear system is to convert the problem into frequency domain, and use a small number of spectral bases to represent the deformation space [11]. But this method cannot handle rotational effects in large deformation due to the linear approximation.

In this paper, we introduce a new algorithm combining the spectral method with the dual Laplacian editing algorithm. This new algorithm inherits the advantages of both: It greatly reduces the size of the linear system and thus increase the speed, while is still able to generate very natural results with large rotational effects. The manifold harmonics bases can be automatically built for arbitrary irregular meshes. Using these manifold harmonics bases, the mesh can be converted into frequency domain, and the size of the linear system in the dual Laplacian editing is greatly reduced from $O(n)$ to $O(m)$, where m is the number of frequencies used in the algorithm. According to our experiments, a small number of frequencies (say $m = 100$) are enough to generate very natural results. So the size of the linear system in every iteration step becomes very small, and thus the system can be very efficiently solved.

Figure 1 illustrates the pipeline of our algorithm. Starting from the original mesh, we first compute the manifold harmonics bases (MHB). Next, we convert the deformation problem into frequency domain using the first 100 spectral bases. We use the dual Laplacian editing with several iterations to gradually refine the solutions. In every iteration, a smoothed mesh is rebuilt from the solutions in frequency domain. Finally, the details are added back to get the final result.

The rest of this paper is organized as follows: Section 2 briefly reviews some previous researches. Sec-

tion 3 explains the details of the dual Laplacian editing algorithm. Our new algorithm is introduced in Section 4, and some experimental results are given in Section 5. Finally, Section 6 concludes the paper with some possible directions of future work.

2 Previous Work

The study of surface deformation has a long history in computer graphics. It is impossible and unnecessary to review all the previous work here. Even for the researches using linear approximation, there are numerous related papers. Most recently, Botsch and Sorkine [1] give a detailed review of the linear algorithms. In this section, we only briefly review some previous work closely related to our algorithm.

Alexa [12] first introduces the Laplacian coordinates into mesh deformation. He uses the Laplacian coordinates as the representation of the local geometry properties of the mesh, and tries to maintain these properties before and after deformation. This method only needs to solve a simple linear system, but it fails to deal with local rotations of the details on the mesh.

Later, Lipman et al. [3] extend the definition of the Laplacian coordinates by expanding the neighborhood of every vertices from its one-ring neighbors to a larger set. By doing so, they can achieve more natural results, but still cannot handle large rotation very well.

In a more recent paper, Au et al. [4] further improve this idea to handle large rotational effects. They use the Laplacian coordinates of the initial mesh as a guess of the final Laplacian coordinates, and then use an iteration process to refine the Laplacian coordinates gradually. The iteration process is later performed in dual space to make the algorithm more robust and to

guarantee the convergency of the algorithm [5].

Another approach using linear system to handle large rotation is multi-resolution editing introduced by Botsch et al. [7]. They generate a smoothed mesh, perform the deformation on it, and apply deformation transfer using the smoothed mesh as a reference to get the final deformed mesh. Note that the multi-resolution here means a hierarchy of meshes with different smoothness but the same connectivities. This is different to the multi-resolution in level of details, where it means a hierarchy of meshes with different number of vertices. The quality of this approach is determined by the way to deform the smoothed mesh. In [7], they use an energy-based method, which may generate unnatural shearing effects for large rotations.

Compared to the more complicated non-linear approaches, all the linear approaches above are very computationally efficient. However, for a mesh with n vertices, the sizes of the linear systems are $O(n)$ for all the algorithms. So they are still not suitable for large meshes. People usually address this problem by pre-factorize the matrix in the linear system and only perform the back-substitution in the running time. Thus the real-time speed can be achieved. However, the matrix of the linear system is fixed only after the constrained vertices are selected by the user. There are many cases where such time-costing pre-factorization is not tolerated. The user would like to select the constrained vertices, specify their final positions, and see the result quickly. There are also some applications where the constrained vertices change frequently, such as collision handling. All of these require to find a more efficient way to solve the linear system for large meshes.

Shi et al. [13] introduce a multigrid linear solver specially designed for Poisson systems. They use a graph coarser to build a multigrid of meshes with fewer and fewer vertices, and then recursively solve the linear system on different levels. This can greatly increase the speed of solving the linear system.

Another approach introduced by Rong et al. [11] uses spectral method to directly reduce the size of the linear system. They use manifold harmonics transform to build a smoothed mesh with a very small number (m) of frequencies. The linear system is converted into frequency domain, and the size of the system becomes $O(m)$, which is usually much smaller than the original size $O(n)$ in spatial domain. After the deformation of the smoothed mesh, the details are added back using deformation transfer as in [7]. Since this method is a linear approximation to the non-linear thin-shell

energy, it cannot generate natural results for large rotational deformation.

Our new algorithm also uses the spectral method, particularly manifold harmonics, as a powerful tool to reduce the size of the linear system. Different to the algorithm in [11], we use dual Laplacian editing for the deformation of the smoothed mesh. So we can handle large rotational effects and get more natural results while keep the high speed of the algorithm. To make this paper self-contained, we first introduce the dual Laplacian editing algorithm in the next section.

3 Dual Laplacian Editing

In this section, we briefly review the framework of the dual Laplacian editing algorithm. More details can be found in [4, 5].

Assume the mesh to be deformed is a triangle mesh with n vertices, and let $\mathbf{V} = \{\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_{n-1}\}^T$ be the set of the vertices. For a vertex \mathbf{v}_i , its *Laplacian coordinate* is a linear combination of itself and its one-ring neighbors, defined as follows [2, 3]:

$$\mathbf{l}_i = \sum_{\mathbf{v}_j \in \mathcal{N}(i)} w_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

where w_{ij} is the weight associated with the edge $\mathbf{v}_i\mathbf{v}_j$, and $\mathcal{N}(i)$ is the set of the one-ring neighbors of \mathbf{v}_i . Different choices of the weights w_{ij} lead to different Laplacian coordinates. The computation of Laplacian coordinates for all the vertices can be formulated in a matrix mode as $\mathbf{l} = \mathbf{L}\mathbf{V}$, where $\mathbf{l} = \{l_0, l_1, \dots, l_{n-1}\}^T$ and \mathbf{L} is an $n \times n$ sparse matrix composed of all the weights w_{ij} . Note that both \mathbf{V} and \mathbf{l} are $n \times 3$ matrices, where the three columns correspond to x -, y - and z -coordinates. For simplicity, from now on, we only consider x -coordinates, and thus assume both \mathbf{V} and \mathbf{l} are vectors with lengths of n . The other two coordinates can be computed in the same way.

The main idea of the Laplacian-based deformation is to maintain the Laplacian coordinates before and after deformation as much as possible. Denote the final positions by \mathbf{V}' . They can be computed by solving the equation $\mathbf{L}\mathbf{V}' = \mathbf{l}$ in least-square sense, or in another word, minimizing the value $\|\mathbf{L}\mathbf{V}' - \mathbf{l}\|^2$. In the same time, the algorithm also wants to satisfy the user-specified constraints, which can also be written in a matrix mode as $\mathbf{C}\mathbf{V}' = \mathbf{c}$, where \mathbf{C} is a $k \times n$ matrix with every row having only one non-zero element (with value 1) corresponding to a certain constrained vertex, \mathbf{c} is a vector contains the final positions of all

the constrained vertices, and k is the number of constraints. Combining these two conditions, the final positions can be computed by solving the following equation in least-square sense:

$$\begin{pmatrix} \mathbf{L} \\ \mathbf{C} \end{pmatrix} \mathbf{V}' = \begin{pmatrix} \mathbf{1} \\ \mathbf{c} \end{pmatrix} \quad (1)$$

In the dual Laplacian editing algorithm, the Laplacian coordinates are computed on the dual mesh of the original mesh. Every vertex in the dual mesh corresponds to a face in the original (primal) mesh. Suppose there are f faces in the primal mesh, the positions in dual space $\hat{\mathbf{V}} = \{\hat{\mathbf{v}}_0, \hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_{f-1}\}$ can be computed using an $f \times n$ matrix \mathbf{D} as $\hat{\mathbf{V}} = \mathbf{D}\mathbf{V}$, where each row of \mathbf{D} has values of 1 at the positions corresponding to the vertices of a certain face, and values of 0 at other positions [14]. In the dual mesh, the one-ring neighbors of every vertex $\hat{\mathbf{v}}_i$ comprise exactly three vertices $\hat{\mathbf{v}}_{i0}$, $\hat{\mathbf{v}}_{i1}$ and $\hat{\mathbf{v}}_{i2}$. These three neighbors decide a plane π_i . Denote by $\hat{\mathbf{q}}_i$ the projection of $\hat{\mathbf{v}}_i$ on the plane π_i . If we chose the barycentric coordinates of $\hat{\mathbf{q}}_i$ in $\triangle \hat{\mathbf{v}}_{i0}\hat{\mathbf{v}}_{i1}\hat{\mathbf{v}}_{i2}$ as the weights, the *dual Laplacian coordinate* of $\hat{\mathbf{v}}_i$ is exactly along the normal ($\hat{\mathbf{n}}_i$) of π_i :

$$\hat{\mathbf{l}}_i = \sum_{j \in \{1,2,3\}} \hat{w}_{ij}(\hat{\mathbf{v}}_{ij} - \hat{\mathbf{v}}_i) = -h_i \hat{\mathbf{n}}_i$$

where h_i is the norm of $\hat{\mathbf{l}}_i$. This good property makes the later iteration process more robust, and the convergence of the iteration guaranteed.

In dual space, eq. (1) becomes:

$$\begin{pmatrix} \hat{\mathbf{L}}\mathbf{D} \\ \mathbf{C} \end{pmatrix} \mathbf{V}' = \begin{pmatrix} \hat{\mathbf{1}} \\ \mathbf{c} \end{pmatrix} \quad (2)$$

Solving eq. (2) is equivalent to solving the following linear system:

$$\mathbf{A}\mathbf{V}' = \mathbf{b} \quad (3)$$

where $\mathbf{A} = \mathbf{D}^T \hat{\mathbf{L}}^T \hat{\mathbf{L}} \mathbf{D} + \mathbf{C}^T \mathbf{C}$ and $\mathbf{b} = \mathbf{D}^T \hat{\mathbf{L}}^T \hat{\mathbf{1}} + \mathbf{C}^T \mathbf{c}$.

Until now, the solution of the final positions cannot reflect the local rotational effects because of the linear approximation. Several steps of iterations need to be performed to get a more natural result. In every iteration step, we compute the normal direction (in dual space) using the newly computed positions, and set the new Laplacian coordinates having the same directions to these normals and the same lengths as the original Laplacian coordinates. Then, we substitute the new

Laplacian coordinates into eq. (2) to compute the positions again. This process is repeated until the difference between the results of two iterations is small enough.

The size of the matrix \mathbf{A} in eq. (3) is $n \times n$. Every time the user changes the constraints, this matrix as well as \mathbf{b} need to be updated. And the new matrix needs to be factorized for the using in later iterations. If the original mesh is too large, i.e. n is too big, this process would require too much computation time. Next, we use manifold harmonics to convert the problem into frequency domain, and apply the dual Laplacian editing algorithm. By doing so, we can greatly decrease the size of the linear system.

4 Spectral Surface Deformation

4.1 Manifold Harmonics

Manifold harmonics is a spectral tool converting a function from spatial domain into frequency domain. The Laplacian operator over a manifold \mathcal{M} (a.k.a. Laplacian-Beltrami operator) is defined using the exterior calculus (EC) as follows:

$$\Delta = \text{div grad} = \sum_i \frac{1}{\sqrt{|g|}} \frac{\partial}{\partial x_i} \left(\sqrt{|g|} \sum_j g^{ij} \frac{\partial}{\partial x_j} \right).$$

The eigenfunctions and eigenvalues of the Laplacian operator on \mathcal{M} are thus a set of pairs (H^k, λ_k) that satisfy the equation $-\Delta H^k = \lambda_k H^k$. This equation can be discretized using the Finite Element Method [15, 16]. Using a piecewise linear ‘‘hat’’ function ϕ_i defined on all the vertices as $\phi_i(\mathbf{v}_j) = \delta_{ij}$, the above eigenfunctions H^k can be discretized as $H^k = \sum_{i=0}^{n-1} H_i^k \phi_i$. So the eigenproblem can be written in a matrix form as: $-\mathbf{D}^{-1} \mathbf{Q} \mathbf{h}^k = \lambda_k \mathbf{h}^k$. For the detailed derivation of this equation, please refer to [16].

The solution of this eigenproblem is called the *Manifold Harmonics Bases* (MHB). The *Manifold Harmonics Transform* (MHT) is defined using the MHB to convert the original coordinates into frequency domain. Again, for simplicity, we only consider the x -coordinates of the vertices in the following discussion. Using the hat function mentioned above, the x -coordinates over the whole manifold \mathcal{M} can be discretized as: $x = \sum_{i=0}^{n-1} x_i \phi_i$. The MHT projects x -coordinates onto the orthonormal MHB leading to m \tilde{x} -coordinates in frequency domain $\tilde{x}_0, \tilde{x}_1, \dots, \tilde{x}_{m-1}$, where m is the number of the frequencies used in

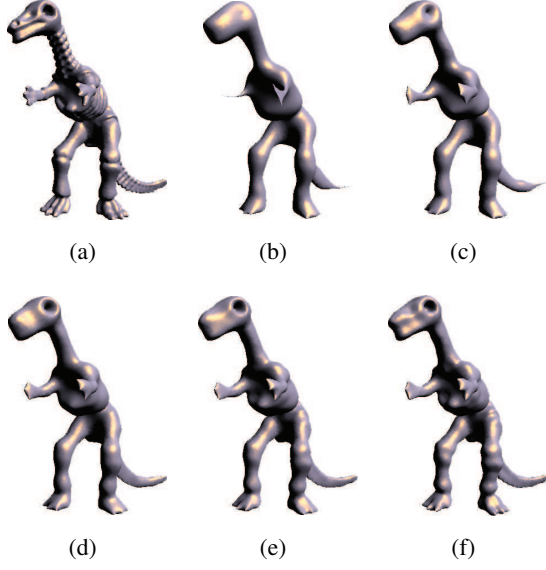


Figure 2: Results of the inverse MHT. (a) is the original mesh, and (b)-(f) are the results of the inverse MHT using 100, 200, 400, 700 and 1000 frequencies respectively.

MHT. More specifically, the \tilde{x} -coordinate corresponding to the frequency basis H^k is defined as:

$$\tilde{x}_k = \langle x, H^k \rangle = \sum_{i=1}^n x_i D_{ii} H_i^k. \quad (4)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product of functions. Similarly, the *inverse MHT* rebuilds the x -coordinates using the m frequencies as follows:

$$x_i = \sum_{k=1}^m \tilde{x}_k H_i^k. \quad (5)$$

Figure 2 shows some results of the inverse MHT of the Dinosaur mesh using different number of frequencies. Figure 2(a) is the original mesh, and Figure 2(b)-2(f) are the results of the inverse MHT using 100, 200, 400, 700 and 1000 frequencies respectively.

4.2 Spectral Deformation

With the help of Manifold harmonics, we can greatly reduce the size of the linear system in Section 3 by only use the first m frequencies to build a smoothed mesh. Substituting eq. (5) into eq. (2), we can get a new equation in frequency domain:

$$\begin{pmatrix} \hat{\mathbf{L}}\mathbf{D}\mathbf{H} \\ \tilde{\mathbf{C}} \end{pmatrix} \tilde{\mathbf{V}}' = \begin{pmatrix} \hat{\mathbf{1}} \\ \mathbf{c} \end{pmatrix} \quad (6)$$

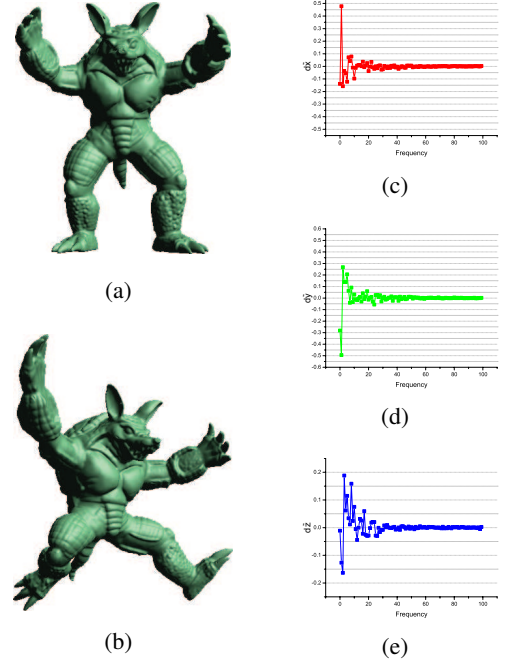


Figure 3: Displacements of coordinates in frequency domain for Armadillo. (a) and (b) are the original and deformed Armadillo mesh. (c)-(e) are the displacements of \tilde{x} -, \tilde{y} - and \tilde{z} -coordinates in frequency domain.

where \mathbf{H} is a matrix composed by MHB: $H_{ij} = H_i^j$, and $\tilde{\mathbf{C}}$ is composed by the MHB corresponding to the constrained vertices. Solving this equation in least-square sense leads to a linear system:

$$\tilde{\mathbf{A}}\tilde{\mathbf{V}}' = \tilde{\mathbf{b}} \quad (7)$$

where $\tilde{\mathbf{A}} = \mathbf{H}^T \mathbf{D}^T \hat{\mathbf{L}}^T \hat{\mathbf{L}} \mathbf{D} \mathbf{H} + \tilde{\mathbf{C}}^T \tilde{\mathbf{C}}$ and $\tilde{\mathbf{b}} = \mathbf{H}^T \mathbf{D}^T \hat{\mathbf{L}}^T \hat{\mathbf{1}} + \tilde{\mathbf{C}}^T \mathbf{c}$.

The size of the matrix $\tilde{\mathbf{A}}$ in eq. (6) is only $m \times m$, which is usually much smaller than the matrix \mathbf{A} in eq. (3). As the result, the factorization process can be performed much faster than that of eq. (3). Although the computation of $\tilde{\mathbf{A}}$ seems to be more complicated, it can also be performed fast. This is because all the matrices except $\tilde{\mathbf{C}}$ are determined only by the original mesh, and thus can be pre-computed. Every time the user changes the constraints, only the later part, i.e. $\tilde{\mathbf{C}}^T \tilde{\mathbf{C}}$ needs to be updated. Usually, the number of constraints k is relatively small, so this update does not cost much computation time.

At the end of every iteration step, we get a new set of $\tilde{\mathbf{V}}$, with coordinates in frequency domain. To visually present the displacements in frequency domain, we plot them in Figure 3 for the deformation of the

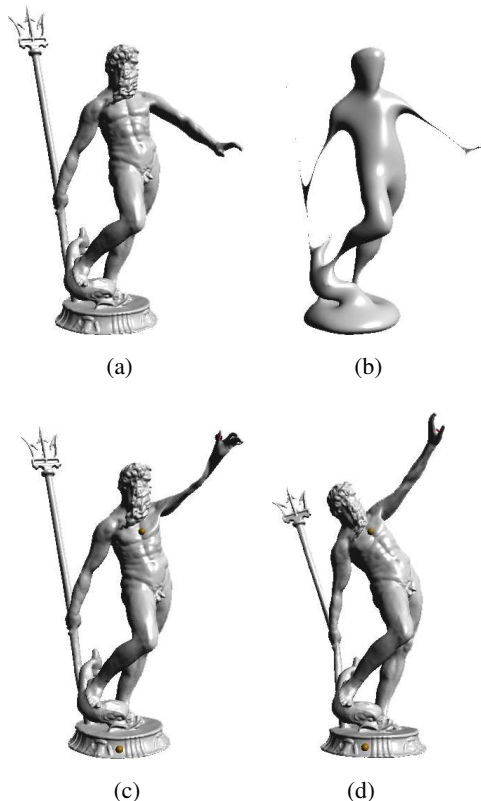


Figure 4: Comparison of the results of Neptune. (a) is the original mesh; (b) is the smoothed mesh using 100 frequencies; (c) is the deformed mesh using the algorithm in [11]; (d) is the deformed mesh using our algorithm.

Armadillo mesh from Figure 3(a) to Figure 3(b). Figure 3(c) is for the \tilde{x} -coordinates in frequency domain: $d\tilde{x} = \tilde{x}' - \tilde{x}$, where \tilde{x}' and \tilde{x} are \tilde{x} -coordinates after and before deformation. Figure 3(d) and 3(e) are for the other two coordinates respectively.

The coordinates in frequency domain can be easily converted back into spatial domain using eq. (5). Then a new set of Laplacian coordinates of the dual mesh can be calculated as explained in Section 3 for the use of the next iteration step. This iteration process stops when the solutions converge.

After the iteration stops, we use deformation transfer, as in [11], to add the details back to get the final deformed mesh. We use the smoothed mesh as the source and transfer the deformation of it to the target mesh (the one with details). Following Botsch et al.'s idea [7], for every triangle of the smoothed mesh, we use its normal to build a deformation gradient matrix. These deformation gradient matrices are then applied on the triangles of the original mesh to get a deformed mesh with details.

5 Experimental Results

Our experimental environment is Intel Core2 Duo 2.93GHz CPU and 2GB DDR2 RAM. We use Microsoft Visual C++.NET 2005 to develop our algorithm. The solver of the linear system is the CSspares library [17].

The solving of the linear system is divided into two parts: updating and factorizing the left matrix of the linear system (i.e. the matrix $\tilde{\mathbf{A}}$ in eq. (7)) and back-substitution for solving the linear system. Note that updating $\tilde{\mathbf{A}}$ only requires to update the later part, i.e. $\tilde{\mathbf{C}}^T \tilde{\mathbf{C}}$, because the former part can be pre-computed. Table 1 lists the time required by these two parts for all the examples in this paper. We list the time for our algorithm as well as for Au et al.'s algorithm [5] for comparison reason. It is evident that the time of updating and factorizing the matrix is much faster for our algorithm than Au et al.'s algorithm, due to the big difference in the size of the matrix ($m \ll n$).

The conversion from frequency domain back into spatial domain in every iteration step leads to a slight overhead in the solving time. But in total, we can achieve in our current experiments about 3-7 times faster speed over the original algorithm. This factor will increase further with the increase of the number of vertices of the mesh.

Figure 4 compares the results of our algorithm with the algorithm in [11] using the Neptune mesh. In this simple example, we only use three point constraints. The two yellow points are fixed at their original positions, while the red one on the left arm is lifted. Figure 4(c) is generated using the algorithm in [11], and Figure 4(d) our algorithm. It is evident that the algorithm in [11] cannot handle such a large rotation very well. The left arm in this result becomes a twisted thin plate. On the contrary, our algorithm generates a very natural result. The torso of Neptune and the Trident also rotate accordingly in our result.

The number of frequencies (m) used to build the smoothed mesh is an important factor in our algorithm. With the increase of m , the time required by both the factorization and the back-substitution increase accordingly. Figure 5 shows such relationship between the time and the number of frequencies for the deformation of the same Dinosaur mesh using the same constraints. According to our experiments, a very small m (say $m = 100$) is good enough for most applications, since increasing m would barely affect the final result. Figure 6 demonstrates two results of deformation of Dragon mesh using the same constraints, but with 100

Model	Number of Vertex	Our algorithm (Sec.)		Au et al.'s algorithm (Sec.)	
		Factorization	Back-Substitution	Factorization	Back-Substitution
Dinosaur	28098	0.061465	0.124726	1.731504	0.062399
Elephant	42321	0.101218	0.180489	4.123936	0.112704
Dragon	50000	0.131981	0.214095	3.943385	0.111215
Armadillo	75002	0.158395	0.369953	11.355873	0.208035
Neptune	99996	0.208901	0.506052	23.263234	0.310035

Table 1: Time for pre-factorization and back-substitution.

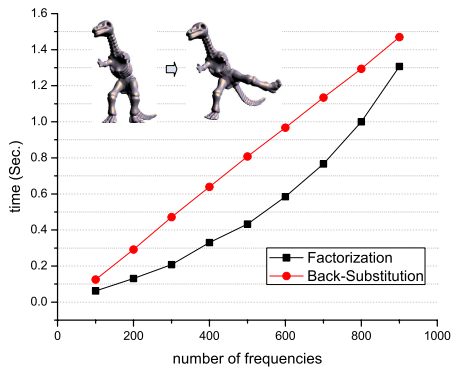


Figure 5: Time with different number of frequencies.

(Figure 6(c) and 6(e)) and 500 (Figure 6(d) and 6(f)) frequencies respectively. It is clear that the difference between these results is indistinguishable. Figure 6(b) also shows the result using the algorithm in [11] as a comparison. Again, our algorithm can generate much better results for such large deformation.

6 Conclusion and Future Work

In this paper, we introduce a new spectral algorithm for surface deformation. With the help of the dual Laplacian editing, our algorithm can handle large rotational effects very well. We use manifold harmonics transform to convert the linear system into frequency domain, so that the size of the linear system is greatly reduced and the speed is thus accordingly increased. Our experiments shows several times speedup over the original dual Laplacian editing algorithm.

Manifold harmonics is a very powerful spectral tool to reduce the size of linear system used in mesh deformation. Since mesh morphing is a closely related topic to mesh deformation, this method could be naturally applied on mesh morphing. One possible future work is to combine the spectral method with the dual Laplacian morphing [18].

Currently, the linear system in our algorithm is solved totally by the CPU. Graphics processing unit (GPU) has seen fast development in recent years. More and more researchers have tried to use the GPU to solve general-purpose problems. Some GPU-based mesh deformation algorithms have already been proposed [10, 19]. Using the parallel computing ability of the GPU to further accelerate mesh deformation is another interesting direction for the future work.

Acknowledgements

This research is supported by the National Science Foundation under Grant No. CCF-0727098.

References

- [1] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.
- [2] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 175–184, 2004.
- [3] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190, 2004.
- [4] Oscar Kin-Chung Au, Chiew-Lan Tai, Hongbo Fu, and Ligang Liu. Mesh editing with curvature flow laplacian operator. Technical Report HKUST-CS05-10, Hong Kong University of Science and Technology, 2005.

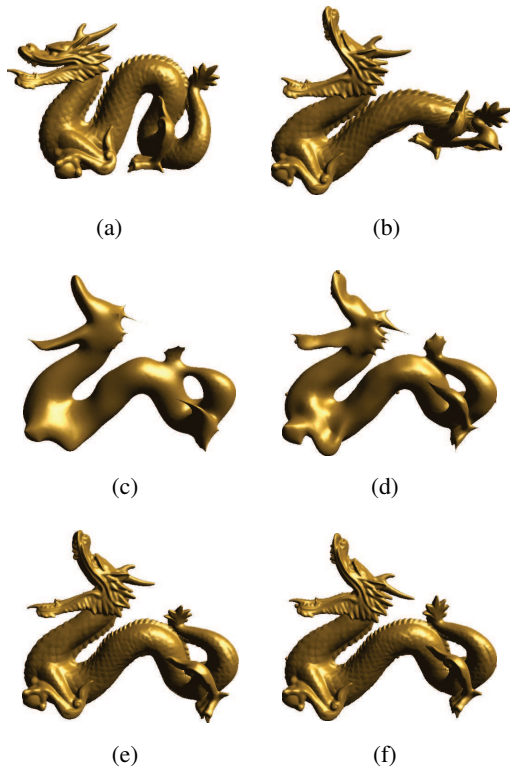


Figure 6: Results using different number of frequencies. (a) original mesh; (b) the result of the algorithm in [11]; (c)-(f) results of our algorithm for the smoothed mesh (2nd row) and the mesh with details (3rd row) with 100 ((c), (e)) and 500 ((d), (f)) frequencies.

- [5] Oscar Kin-Chung Au, Chiew-Lan Tai, Ligang Liu, and Hongbo Fu. Dual laplacian editing for meshes. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):386–395, 2006.
- [6] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with Poisson-based gradient field manipulation. *ACM Transactions on Graphics*, 23(3):644–651, 2004.
- [7] Mario Botsch, Robert Sumner, Mark Pauly, and Markus Gross. Deformation transfer for detail-preserving surface editing. In *Proceedings of 11th International Fall Workshop Vision, Modeling & Visualization*, pages 357–364, 2006.
- [8] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics*, 24(3):479–487, 2005.
- [9] Ioana Boier-Martin, Remi Ronfard, and Fausto Bernardini. Detail-preserving variational surface design with multiresolution constraints. In *Proceedings of the 2004 Shape Modeling International*, pages 119–128, Washington, DC, USA, 2004. IEEE Computer Society.
- [10] Kun Zhou, Xin Huang, Weiwei Xu, Baining Guo, and Heung-Yeung Shum. Direct manipulation of subdivision surfaces on GPUs. *ACM Transactions on Graphics*, 26(3):91, 2007.
- [11] Guodong Rong, Yan Cao, and Xiaohu Guo. Spectral mesh deformation. *The Visual Computer (CGI 2008 special issue)*, 24(7-9):787–796, 2008.
- [12] Marc Alexa. Local control for mesh morphing. In *Proceedings of the International Conference on Shape Modeling and Applications*, pages 209–215, 2001.
- [13] Lin Shi, Yizhou Yu, Nathan Bell, and Wei-Wen Feng. A fast multigrid algorithm for mesh deformation. In *SIGGRAPH '06*, pages 1108–1117, 2006.
- [14] Gabriel Taubin. Dual mesh resampling. In *Proceedings of Pacific Graphics*, pages 94–113, 2001.
- [15] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace-Beltrami spectra as shape-DNA of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.
- [16] Bruno Vallet and Bruno Lévy. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum*, 27(2):251–260, 2008. (Proceedings of Eurographics 2008).
- [17] Timothy A. Davis. *Direct Methods for Sparse Linear Systems*. Number 2 in Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, 2006.
- [18] Jianwei Hu, Ligang Liu, and Guozhao Wang. Dual laplacian morphing for triangular meshes. *Computer Animation and Virtual Worlds (Proceedings of CASA 2007)*, 18(4-5):271–277, 2007.
- [19] Martin Marinov, Mario Botsch, and Leif Kobbelt. GPU-based multiresolution deformation using approximate normal field reconstruction. *Journal of Graphics Tools*, 12(1):27–46, 2007.