Visual Comput (2008) 24: 787–796
DOI 10.1007/s00371-008-0260-x

ORIGINAL ARTICLE

Spectral mesh deformation

Guodong Rong
Yan Cao
Xiaohu Guo

Published online: 29 May 2008

Electronic supplementary material
The online version of this article (doi:10.1007/s00371-008-0260-x) contains supplementary material, which is available to authorized users.

G. Rong (✉) · X. Guo
Dept. of Computer Science, University of Texas at Dallas, USA
{guodongrong, xguo}@utdallas.edu

Y. Cao
Dept. of Mathematical Sciences, University of Texas at Dallas,
yan.cao@utdallas.edu

Abstract In this paper, we present a novel spectral method for mesh deformation based on manifold harmonics transform. The eigenfunctions of the Laplace–Beltrami operator give orthogonal bases for parameterizing the space of functions defined on the surfaces. The geometry and motion of the original irregular meshes can be compactly encoded using the low-frequency spectrum of the manifold harmonics. Using the spectral method, the size of the linear deformation system can be significantly reduced to achieve interactive computational speed for manipulating large triangle meshes. Our experimental results demonstrate that only a small spectrum is needed to achieve undistinguishable deformations for large triangle meshes. The spectral mesh deformation approach shows great performance improvement on computational speed over its spatial counterparts.

Keywords Spectral geometry · Manifold harmonics · Mesh deformation · Interactive manipulation

1 Introduction

Geometric model deformation is a very important topic in computer graphics and animation. The computation of physically correct deformation is intrinsically non-linear, and usually requires intensive computational load and lengthy computing time, to carry out the algorithm on regular PCs without any special hardware assistance (e.g. GPU-acceleration [13, 27]). However, in many interactive applications, e.g. model designing, physically correct deformation is not necessary. Instead, a physically "plausible" deformation could be sufficient for many of the interactive applications.

Linear approximations [5] have been adopted to accelerate the computation and get results that are still intuitively "correct". Linear approximations are usually achieved by simplifying the physical energy formula to a quadratic form, and minimizing the energy, which leads to solving a linear system. For triangle meshes, the unknowns of the linear system are typically the positions (or displacements) of the vertices of the geometric model. So for large meshes (e.g. the Armadillo model in Fig. 1), solving such a linear system at run-time could be still prohibitive for interactive applications.

From another viewpoint, model deformation usually requires some non-linear operations, especially for the fine surface details. For example, determining the required local rotations from positional constraints is a non-linear problem. Linear approximations will cause geometric details and protruding features to be distorted. A general technique to overcome these difficulties is to complement the linear deformation with multiresolution approaches [13]. A *geometric* hierarchy of smoother and smoother meshes can be built up by removing surface details [19]. The linear deformation is applied on the smoothed surface, and the details can be added back afterwards to preserve the local details with consistent orientations. A straightforward approach is to preserve the mesh connectivity information between the original and the smoothed meshes, in order for the details to be

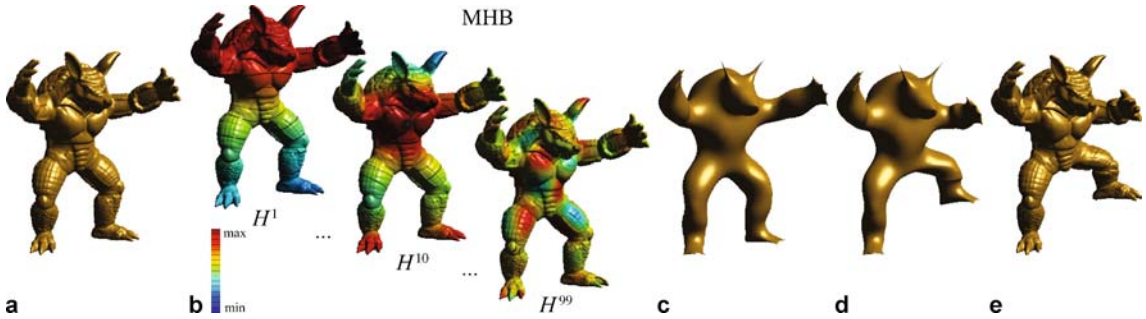

Fig. 1a–e. The spectral mesh deformation pipeline. **a** The original Armadillo model, **b** the manifold harmonics bases, **c** the smoothed model reconstructed using the first m bases, **d** the deformed smoothed model, and **e** the deformed model with details added back

added back easily [6]. However, the size of the linear system is not reduced since the number of unknowns (vertex positions) remains the same for the smoothed surfaces. Another approach is to use *topological* hierarchies of coarser and coarser meshes [8]. Subdivision surfaces provide a nice coupling of the geometric and topological hierarchy, and the number of unknowns in the coarse subdivision mesh could be significantly smaller than the original mesh [2, 27]. However, automatically building subdivision surfaces for arbitrary irregular meshes is a highly non-trivial task, since the original irregular mesh may not have coherent subdivision connectivity.

In this paper, we propose to use the spectrum of the Laplace–Beltrami operator defined on manifold surfaces, i.e. manifold harmonics, to compactly encode the deformation functions. The manifold harmonics can be precomputed on arbitrary irregular meshes. Compared with other subspace deformation techniques [9, 27], the computation of manifold harmonics is fully automatic, and these orthogonal bases provide a compact parametrization for the space of functions defined on the surfaces. We can use very small number (compared to the number of vertices) of frequency components to represent the geometry and motion of the smoothed model. So the number of unknowns in the linear system for the deformation can be greatly decreased, to allow interactive manipulation on large triangle meshes. The process of our algorithm is described as the following steps:

1. compute manifold harmonics bases for the original model with n vertices (Fig. 1b);
2. perform the inverse manifold harmonics transform using the first m ($m \ll n$) frequencies to get a smoothed model (Fig. 1c);
3. perform the deformation on the smoothed model by solving a linear system with m unknowns (Fig. 1d);
4. add the details back to the deformed smoothed model to get the final result (Fig. 1e).

The rest of this paper is organized as follows: Sect. 2 reviews some previous work related to mesh deformation and spectral methods. Section 3 introduces the concept of manifold harmonics. The details of our new algorithm are given in Sect. 4, and the experimental results are shown in Sect. 5. Section 6 concludes the paper with some possible directions of future work.

2 Related work

Mesh deformation is an active research area in computer graphics. There are numerous previous papers on this topic. Energy minimization has long been a general approach to deform smooth surfaces [3, 22]. A variational approach is introduced in [2] to deform subdivision surfaces. To preserve surface details, they optimize the energy of a deformation vector field instead of the deformation energy of vertex positions. Multiresolution mesh editing techniques [11, 28] have been developed for detail-preserving deformations by decomposing a mesh into several frequency bands. A deformed mesh is obtained by first manipulating the low-frequency mesh and later adding back the high frequency details as displacement vectors.

Yu et al. [23] apply the widely used Poisson equation on the 3D model deformation. They set the gradients before and after the deformation to be equal to get a Poisson equation, which is a linear system. The solutions of this equation give the deformed model. This class of algorithm is called *gradient domain deformation*. Gradient domain mesh deformation techniques [1, 9, 12, 17, 23, 24, 26, 27] have been intensively investigated for mesh editing. The main challenge is to handle nontrivial transformations which include rotations (especially large rotations) while preserving as much as possible the visual characteristic of the shape at interactive rates. The most important idea is to factor out the rotation from the deformation. For shape editing, the factorization and shape definition have to be solved simultaneously [17] since the target shape is not explicitly given. Instead of factoring out the rotation, a better solution is to represent the shape with rotation-invariant coordinates [12]. Zayer et al. [24] use a harmonic

scalar field to better propagate deformations to the entire mesh from the constraints. Huang et al. [9] use a subspace domain to reduce the problem dimensionality via mean value coordinates. Botsch et al. [4] introduce a local shape representation based on prisms and used a hierarchial multi-grid solver to reduce the problem complexity. A different research direction aims at preservation of the volume of a shape. Zhou et al. [26] develop a mesh deformation based on volumetric graph Laplacian to preserve the local volume.

Another type of mesh deformation algorithms is the *deformation transfer* introduced by Sumner and Popović [18]. They use the deformation of a *source* model to guide the deformation of a different target model. The idea is to match the deformation gradients of the corresponding triangles of the source model and the target model. By doing so, a linear system is generated, and its solution gives the deformed target model. The deformation transfer is shown as equivalent to gradient-based deformation by Botsch et al. [6]. And the deformation transfer method can be further improved by using the normal of every triangle, instead of an artificial fourth vertex, to compute the deformation gradients. The size of the linear system is thus reduced accordingly.

Generally speaking, the mesh deformation techniques can be classified into two categories: linear algorithms and non-linear algorithms. Non-linear algorithms require to solve a non-linear system, and thus are hard to achieve interactive rates. On the contrary, linear algorithms are usually much faster with some compromise in the quality of the results. A thorough survey on linear deformation algorithms can be found in [5]. For large mesh deformations, the size of the linear system could be still prohibitive for interactive manipulations. In this paper, we use manifold harmonics to help to reduce the size of the linear system.

Spectral geometry analysis is first used as a theoretical tool to characterize the classical approximations of low-pass filters [19]. This approach is based on the similarity between the eigenvectors of the graph Laplacian and the basis functions used in the discrete Fourier transform. Several variants of this approach are then suggested, such as combinatorial graph Laplacians [10], and a discrete Laplacian operator [16, 21] using cotan weights. The basic idea is to compute the eigenfunctions and eigenvalues of the Laplacian on a general manifold surface, i.e. the manifold harmonics.

3 Manifold harmonics

In this section, we briefly review the concept of manifold harmonics, and how to use it to convert the functions defined on a manifold surface from space domain to frequency domain, and vice versa. A more detailed introduction can be found in [10, 16, 19, 21, 25].

In Euclidean domain $\mathbb{R}^n$, the Laplace operator is defined as the divergence of the gradient:

$$\Delta = \operatorname{div}\operatorname{grad} = \nabla \cdot \nabla = \sum_i \frac{\partial^2}{\partial x_i^2}. \tag{1}$$

By using the exterior calculus (EC), the definition of the Laplacian can be generalized to functions defined over a manifold $\mathcal{M}$ with metric g, and is then called the Laplace–Beltrami operator:

$$\Delta = \operatorname{div}\operatorname{grad} = \sum_i \frac{1}{\sqrt{|g|}} \frac{\partial}{\partial x_i} \left(\sqrt{|g|} \sum_j g^{ij} \frac{\partial}{\partial x_j} \right), \tag{2}$$

where $|g|$ denotes the determinant of g and g^{ij} denote the components of the inverse of the metric tensor g. Hence we can define the eigenfunctions and eigenvalues of the Laplacian on a manifold surface $\mathcal{M}$ as all the pairs (H^k, λ_k) that satisfy :

$$-\Delta H^k = \lambda_k H^k. \tag{3}$$

The eigenfunctions of the continuous Laplace–Beltrami operator give orthogonal bases for the space of functions defined on the surface. The spectrum of this operator is isometry invariant, and continuous deformations of the manifold's geometry result in continuous changes of the spectrum. Smaller eigenvalues of the spectrum are correlated to coarser features of the manifold while higher eigenvalues represent finer structures. Hence the expansion coefficients provide a canonical parametrization of functions defined on the surface, and we can perform the deformation process in this coefficient domain.

Similar to [16, 21], we employ the finite element method (FEM) to discretize the above formula on a manifold surface with n vertices. By using the piecewise linear "hat" function ϕ_i $(i = 1, \ldots, n)$ which is associated with each of the n vertices, and $\phi_i(v_j) = \delta_{ij}$, solving the above formula becomes finding $H^k = \sum_{i=1}^n H_i^k \phi_i$ that satisfy:

$$\forall j \in \{1, \ldots n\}, \quad \langle -\Delta H^k, \phi_j \rangle = \lambda_k \langle H^k, \phi_j \rangle, \tag{4}$$

by projecting Eq. 3 onto each ϕ_i. Or we can write it in matrix form:

$$-\boldsymbol{Q}\boldsymbol{h}^k = \lambda_k \boldsymbol{B}\boldsymbol{h}^k, \tag{5}$$

where

$$\boldsymbol{h}^k = [H_1^k, H_2^k, \ldots H_n^k]^T,$$

$$\begin{cases} Q_{ij} = (\cot \beta_{ij} + \cot \beta'_{ij})/2 \\ Q_{ii} = \sum_j Q_{ij}, \end{cases}$$

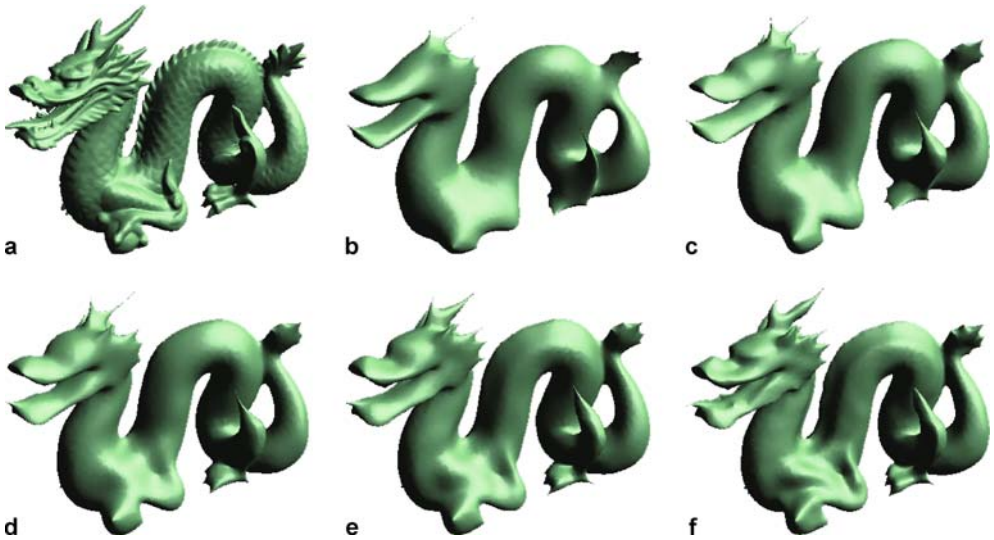

Fig. 2a–f. The inverse manifold harmonics transform for models of different numbers of bases. **a** is the original dragon model, and **b–f** are the results of the inverse manifold harmonics transform using $m = 100$, 200, 300, 500 and 900 bases, respectively

$$\begin{cases} B_{ij} = (|t| + |t'|)/2 \\ B_{ii} = \Big(\sum\limits_{t \in St(i)} |t| \Big)/6. \end{cases}$$

t, t' are the two triangles that share the edge (i, j), $|t|$ and $|t'|$ are their areas, $\beta_{i,j}$, $\beta'_{i,j}$ are the two angles opposite to the edge (i, j), and $St(i)$ is the set of triangles incident to vertex i.

The above formulation can be further simplified into:

$$-\boldsymbol{D}^{-1}\boldsymbol{Q}\boldsymbol{h}^k = \lambda_k \boldsymbol{h}^k, \tag{6}$$

where $\boldsymbol{D}$ is a diagonal matrix defined as follows:

$$D_{ii} = \sum_j B_{ij} = \Big(\sum_{t \in St(i)} |t| \Big)/3.$$

The solution to this eigenproblem yields a series of eigenpairs (H^k, λ_k) called the *manifold harmonics bases* (MHB). These bases are orthogonal, i.e. the functional inner product $\langle H^i, H^j \rangle = 0$ if $i \neq j$. We also ensure that the MHB is orthonormal, by dividing each basis vector H^k by its functional norm $\langle H^k, H^k \rangle$. By using the Arnoldi method, it is possible to compute eigenvectors band-by-band utilizing the shift-invert spectral transform. A detailed derivation of these formulae can be found in [21].

Using the MHB, we can define the *manifold harmonics transform* (MHT) to convert the geometry of the manifold surface $\mathcal{M}$ into the spectral domain. The geometry x (resp. y, z) of the triangulated surface can be considered as a piecewise linear function defined over the hat functions $\phi_i : x = \sum_{i=1}^{n} x_i \phi_i$ where x_i denotes the x coordinate at vertex i. The MHT is the projection of x onto the orthonormal MHB via the functional inner product, and the result is a vector $[\tilde{x}_1, \tilde{x}_2, \ldots \tilde{x}_m]$, with each item $\tilde{x}_k$ corresponding to each frequency basis H^k:

$$\tilde{x}_k = \langle x, H^k \rangle = \sum_{i=1}^{n} x_i D_{ii} H_i^k. \tag{7}$$

The inverse MHT maps the descriptor from frequency domain onto space domain by reconstructing x at vertex i using the first m frequencies:

$$x_i = \sum_{k=1}^{m} \tilde{x}_k H_i^k. \tag{8}$$

Figure 2 shows the inverse MHT for the Dragon model using 100, 200, 300, 500, and 900 bases, respectively.

4 Deformation process

The manifold harmonics transform can help us to transfer the geometric representation of a surface model and its deformation functions from the space domain to the frequency domain. When reconstructing geometric and deformation information using only the first m frequencies, we are in fact conducting a low-pass signal filtering. The high frequencies (details) are filtered, and we get a smoother approximation of the original model. In this paper we use a linear Laplacian-based variational algorithm for mesh deformation. It is well known that the gradient-based transformation cannot handle the local rotation of details [5]. We can overcome this problem by using a spectral multi-resolution approach. The multi-resolution here means different levels of resolution in

the frequency domain, which is different from the traditional level-of-detail technique in the space domain. The Laplacian-based deformation will be solved over the spectrum of m frequencies. Geometric details can be added back to the spectral reconstructed mesh after deformation using the *deformation transfer* technique [6, 18]. By using only the first m frequencies for solving the linear variational system, significant improvement on the computational speed can be achieved to allow interactive mesh manipulation for large mesh models. To make our paper self-contained, we briefly review the linear Laplacian-based variational algorithm for mesh deformation in the following section. For a more detailed survey in this field, please refer to [5].

4.1 Minimizing deformation energy

The 2-manifold surfaces can be considered as *thin-shells* [7] having *elastic energy* of stretching and bending terms. To perform the deformation on the smoothed surfaces, we minimize their elastic energy caused by the deformation process. Suppose $\mathcal{S}$ and $\mathcal{S}'$ are the manifold surfaces before and after the deformation. The elastic energy is defined as follows [20]:

$$E(\mathcal{S}') = \int_{\Omega} k_s \|\boldsymbol{I}' - \boldsymbol{I}\|_F^2 + k_b \|\boldsymbol{II}' - \boldsymbol{II}\|_F^2 \,\mathrm{d}u\,\mathrm{d}v, \tag{9}$$

where Ω is the parameterizing domain of the surface $\mathcal{S}$, $\boldsymbol{I}$ and $\boldsymbol{II}$ (resp. $\boldsymbol{I}'$ and $\boldsymbol{II}'$) are the first and the second fundamental forms of the surface $\mathcal{S}$ (resp. $\mathcal{S}'$), $\|\cdot\|_F$ denotes a Frobenius norm, and k_s and k_b are parameters to control the resistance of stretching and bending.

Denote $\boldsymbol{p}$ a point on $\mathcal{S}$, and $\boldsymbol{p}'$ the corresponding point on $\mathcal{S}'$, the displacement vector is then $\boldsymbol{d} = \boldsymbol{p}' - \boldsymbol{p}$. Using the first-order and second-order partial derivatives with respect to the surface parametrization (u, v), to approximate the first and second fundamental forms, the energy can be simplified into a quadratic form [22]:

$$\begin{aligned} E(\mathcal{S}') = \int_{\Omega} & k_s\big(\|\boldsymbol{d}_u\|^2 + \|\boldsymbol{d}_v\|^2\big) \\ & + k_b\big(\|\boldsymbol{d}_{uu}\|^2 + 2\|\boldsymbol{d}_{uv}\|^2 + \|\boldsymbol{d}_{vv}\|^2\big)\,\mathrm{d}u\,\mathrm{d}v. \end{aligned} \tag{10}$$

The minimization of this energy can be achieved by using variational calculus, which yields the Euler–Lagrange partial differential equation:

$$-k_s \Delta \boldsymbol{d} + k_b \Delta^2 \boldsymbol{d} = \boldsymbol{0}, \tag{11}$$

where Δ and Δ^2 are the Laplacian and bi-Laplacian operators:

$$\Delta \boldsymbol{d} = \mathrm{div}\nabla \boldsymbol{d} = \boldsymbol{d}_{uu} + \boldsymbol{d}_{vv},$$
$$\Delta^2 \boldsymbol{d} = \Delta(\Delta \boldsymbol{d}) = \boldsymbol{d}_{uuuu} + 2\boldsymbol{d}_{uuvv} + \boldsymbol{d}_{vvvv}.$$

On the manifold surface $\mathcal{S}$, the Laplacian operator Δ becomes the Laplace–Beltrami operator $\Delta_{\mathcal{S}}$, and Eq. 11 becomes:

$$-k_s \Delta_{\mathcal{S}} \boldsymbol{d} + k_b \Delta_{\mathcal{S}}^2 \boldsymbol{d} = \boldsymbol{0}. \tag{12}$$

A discretization based on finite differences leads to a discrete Laplace–Beltrami operator at a vertex v_i as:

$$\Delta_{\mathcal{S}} f(v_i) = w_i \sum_{v_j \in St(i)} w_{ij}\big(f(v_j) - f(v_i)\big).$$

Here, the weights for every vertex (w_i) and every edge (w_{ij}) are defined using the cotangent weights [14]:

$$w_i = \frac{1}{A_i}, \quad w_{ij} = (\cot\beta_{ij} + \cot\beta'_{ij})/2,$$

where A_i is the Voronoi area around vertex v_i [14], and β_{ij} and β'_{ij} are the two angles opposite to edge (v_i, v_j), the same as those defined for Q_{ij} in Eq. 5. With the discretization of the Laplace–Beltrami operator, the Euler–Lagrange equation 12 is then discretized to a sparse $n \times n$ linear system:

$$\big(-k_s \boldsymbol{L} + k_b \boldsymbol{L}^2\big)\boldsymbol{d} = \boldsymbol{0}, \tag{13}$$

where $\boldsymbol{L} = \boldsymbol{M}^{-1}\boldsymbol{L}_s$, $\boldsymbol{M}$ is a diagonal matrix with $\boldsymbol{M}_{ii} = A_i$, and $\boldsymbol{L}_s$ is a symmetric matrix defined as the following:

$$S_{ij} = \begin{cases} -\sum_{v_k \in St(i)} w_{ik}, & i = j; \\ w_{ij}, & i \neq j,\ v_j \in St(i); \\ 0, & \text{otherwise.} \end{cases}$$

4.2 Adding constraints

In an interactive mesh manipulation system, the user selects the constrained vertices and specifies their final positions explicitly. We use the Lagrange multiplier to integrate the constraints into our linear system Eq. 13. We set the target function as the energy E defined in Eq. 10. The minimum value of E occurs when Eq. 12 is satisfied. So we can convert the linear system of Eq. 12 to a minimizing problem with constraints.

Suppose we have n' constrained vertices, and denote by C the set of these vertices ($|C| = n'$). For vertex $v_i \in C$, its positions before the deformation (p_i) and after the deformation (p'_i) are both known. So the displacement constraint on the vertex v_i: $d'_i = p'_i - p_i$ is also known. The n' constraints can be written as: $d_i - d'_i = 0$, $v_i \in C$. We denote these constraints by g_k, $0 \le k < n'$. Finally, the constrained minimizing problem corresponding to Eq. 13 is:

$$\min E, \quad \text{s.t.} \quad g_k = 0,\ 0 \le k < n'.$$

Using the Lagrange multiplier method, we can build a new target function:

$$F = E + \sum_{k=0}^{n'-1} \gamma_k g_k,$$

where $\gamma_0, \ldots \gamma_{n'-1}$ are the unknown scalars associated with each constraint. Minimizing F is equivalent to solving the following system:

$$\begin{cases} \partial F/\partial d_0 = 0 \\ \ldots \\ \partial F/\partial d_{n-1} = 0 \\ \partial F/\partial \gamma_0 = 0 \\ \ldots \\ \partial F/\partial \gamma_{n'-1} = 0 \end{cases}$$

or in matrix mode:

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^{\mathrm{T}} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{\gamma} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{d}' \end{pmatrix}, \tag{14}$$

where $\boldsymbol{\gamma} = [\gamma_0, \ldots \gamma_{n'-1}]^{\mathrm{T}}$, and $\boldsymbol{d}'$ is the vector composed by all the displacement constraints d_i'. The coefficient matrix $\boldsymbol{A} = -k_{\mathrm{s}}\boldsymbol{L} + k_{\mathrm{b}}\boldsymbol{L}^2$, and the coefficient matrix $\boldsymbol{B} \subset \mathbb{R}^{n \times n'}$ with value 1 at B_{ij} if g_j is the constraint for vertex v_i, and value 0 otherwise.

By solving Eq. 14, we can get the deformed model which satisfy all the specified constraints. This equation is a linear system with the size of $(n+n') \times (n+n')$. When the number of vertices n is large, it is still very challenging to solve Eq. 14 at interactive speed.

4.3 Converting to frequency domain

We employ manifold harmonics to reduce the size of the above linear system, and thus achieve higher computational speed. As explained in Sect. 3, for every model, we can pre-compute the manifold harmonic bases (MHB) and get the first m frequency components. After having the MHB, we can convert the positions p_i into frequency domain using MHT (Eq. 7). So the displacements d_i can also be expanded in frequency domain as the following:

$$d_i = p_i - p_i' = \sum_{k=1}^{m} \tilde{d}_k H_i^k.$$

Substituting d_i into Eq. 14, we can have:

$$\begin{pmatrix} \boldsymbol{H}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{H} & \tilde{\boldsymbol{B}} \\ \tilde{\boldsymbol{B}}^{\mathrm{T}} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{d}} \\ \boldsymbol{\gamma} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{d}' \end{pmatrix}, \tag{15}$$

where $\tilde{\boldsymbol{d}}$ is the vector of $[\tilde{d}_0, \ldots \tilde{d}_{m-1}]^{\mathrm{T}}$, $\boldsymbol{H}$ is the $n \times m$ matrix $[H^0, \ldots H^{m-1}]$, and $\tilde{\boldsymbol{B}}$ is an $m \times n'$ matrix with $\tilde{B}_{kj} = H_i^k$ if g_j is the constraint for vertex v_i.

After this substitution, the size of the linear system is reduced to $(m+n') \times (m+n')$, which is far less than the original $(n+n') \times (n+n')$ if we select a small number of frequencies. Actually in most of our experiments we choose $m = 100$, while the number of vertices could be $n > 100\,000$ for the models in our interactive system.

After solving this linear system, we can use an inverse MHT (Eq. 8) to convert the spectral displacements $\tilde{\boldsymbol{d}}$ back to space domain to get $\boldsymbol{d}$, and thus $\boldsymbol{p}' = \boldsymbol{p} + \boldsymbol{d}$. Figure 3 plots the first 50 frequency coefficients $(\tilde{x}, \tilde{y}, \tilde{z})$ of the Dinosaur model before and after the deformation, together

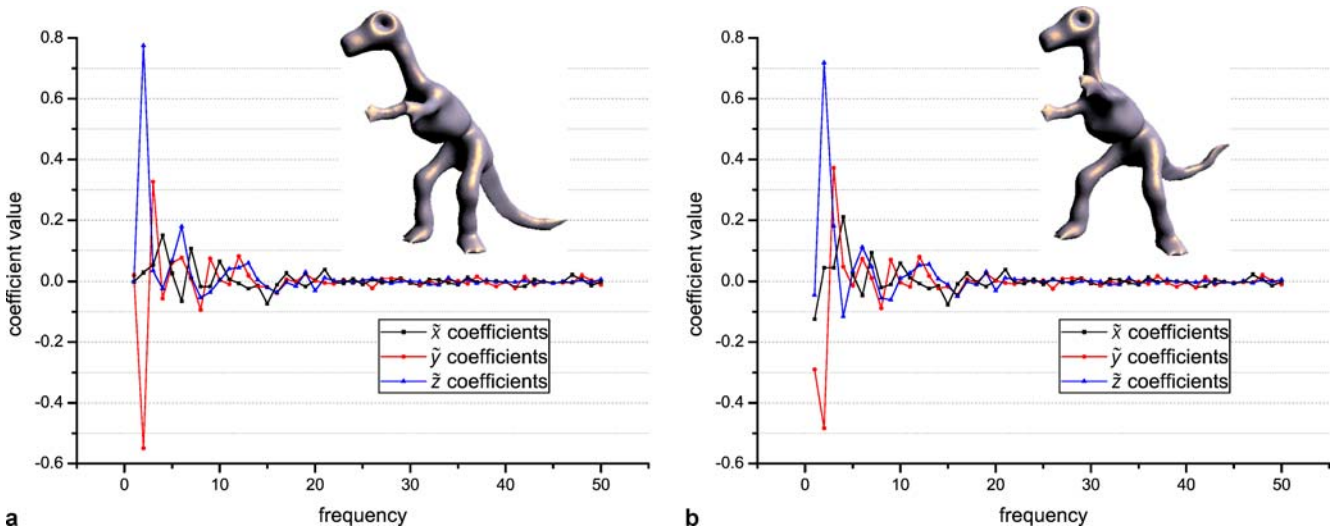

Fig. 3a,b. The frequency coefficients before (**a**) and after (**b**) deformation

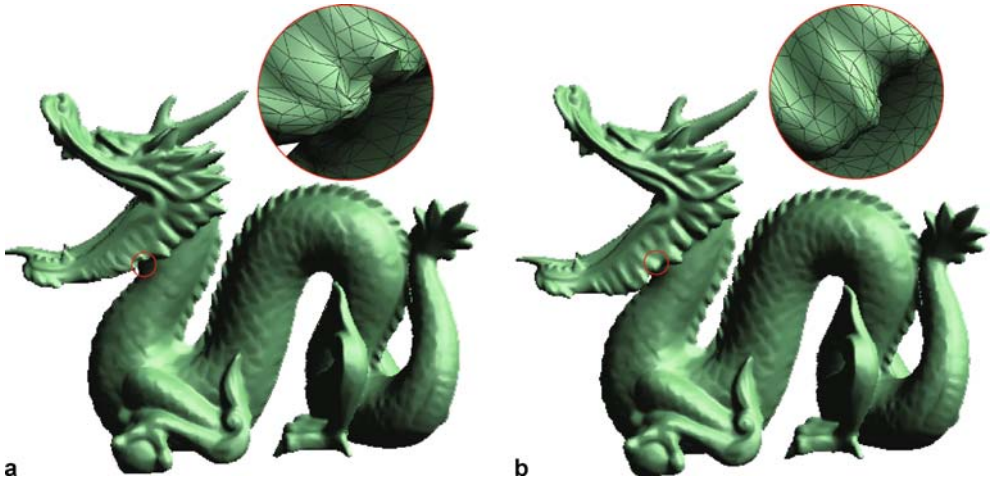

Fig. 4a,b. The comparison between different detail-adding approaches. Details are added back using **a** local frame approach and **b** deformation transfer approach. Some triangles in **a** erroneously intersect with each other, while no such problem happens in **b**

with the smoothed models rebuilt using the first 300 frequency coefficients by the inverse MHT.

By using only the first m frequencies to construct the model, we are in fact conducting a low-pass filtering on the model. So the deformation is applied on the smoothed model, instead of the original one. In this way, we can partially overcome the shortcoming of the linear system that cannot handle the intrinsic rotations of the local details, and focus on the deformation of the general shape of the model. After the deformation, we need to add the lost high-frequency details back to get the final deformation result.

4.4 Adding details

There are some different approaches to add the high-frequency details back to the smoothed model. The simplest way is to record the differences between the corresponding vertices of the original model and the smoothed model before the deformation, and directly add the differences back after the deformation. However, this naive approach cannot handle the local rotational effects.

Another efficient approach is to build a local coordinate system at every vertex. We use the normal ($\boldsymbol{n}$) of the vertex as the local z-axis ($\boldsymbol{z}$), and select the first edge of the first triangle incident to this vertex, normalize its direction as $\boldsymbol{e}$. Then, we have the local x- and y-axis: $\boldsymbol{x} = \boldsymbol{e} \times \boldsymbol{z}$, and $\boldsymbol{y} = \boldsymbol{z} \times \boldsymbol{x}$. A simple matrix multiplication can convert the difference vector between the corresponding vertices of the original model and the smoothed model into this local coordinate system. After the deformation, we rebuild such a local coordinate system at every vertex, and reconstruct the deformed shape by adding the difference vectors back to the deformed local coordinate systems. In this way the local rotation effects will be automatically achieved. However, this local-frame approach also has its own problem: under local bending deformations, some parts of the model may intersect with the other parts nearby.

A more elegant (yet less efficient) approach is introduced by Botsch et al. [6], by drawing ideas from the *deformation transfer* technique [18]. Using the smoothed model (spectral reconstructed) as the source model, and the original one with details as the target model, we can transfer the deformation gradient from the source model to the target model. When we deform each triangle using the spectral approach mentioned in Sect. 4.3, we compute the deformation gradient associated with the triangle. In practice, we follow Botsch et al.'s idea, which discard the fourth vertex in [18], and use the normal of every triangle instead. The deformation gradients can be applied back to the triangles in the original model, such that the original detailed mesh can be deformed consistently. This leads to a linear system with the size of $n \times n$. Fortunately the left part of the linear system is independent of the users' manipulation constraints, and can be pre-factorized when the model is loaded together with its manifold harmonics bases. During users' interactive manipulation process, only back-substitutions need to be performed for each frame. By solving this linear system, we can get the deformed model with details added back to the spectral reconstructed one.

A comparison of the results using local frame and deformation transfer is shown in Fig. 4. The local frame approach may erroneously generate some intersecting triangles for the parts with lots of details.

5 Experimental results

We implement our algorithm on a Windows XP PC with Intel Core2 Duo 2.93 GHz CPU and 2 GB DDR2 RAM.

Table 1. The computing time of different steps in our algorithm. In the pre-computing time, PT1 is for computing manifold harmonics bases, PT2 is for computing the left-hand matrix for the deformation of the smoothed model ($H^{\mathrm{T}}AH$ in Eq. 15), and PT3 is for computing the left-hand matrix for adding details and factorizing it. In the running time, RT1 is for the deformation of the smoothed model, and RT2 is for adding details. The last column is for the running time without spectral transform

Model	Model details		Pre-computing time (s)			Running time (s)		Running time
	#Vertex	#Triangle	PT1	PT2	PT3	RT1	RT2	w/o spectrum (s)
Lion	5000	9996	1.718058	0.228635	0.289104	0.000235	0.013313	12.203043
Dinosaur	28 098	56 192	19.680900	2.059899	7.115392	0.001208	0.076424	102.952112
Dragon	50 000	100 000	25.961076	4.495551	21.211791	0.002068	0.139505	396.493329
Armadillo	75 002	150 000	60.122655	7.692200	48.596778	0.003041	0.241282	1135.150139

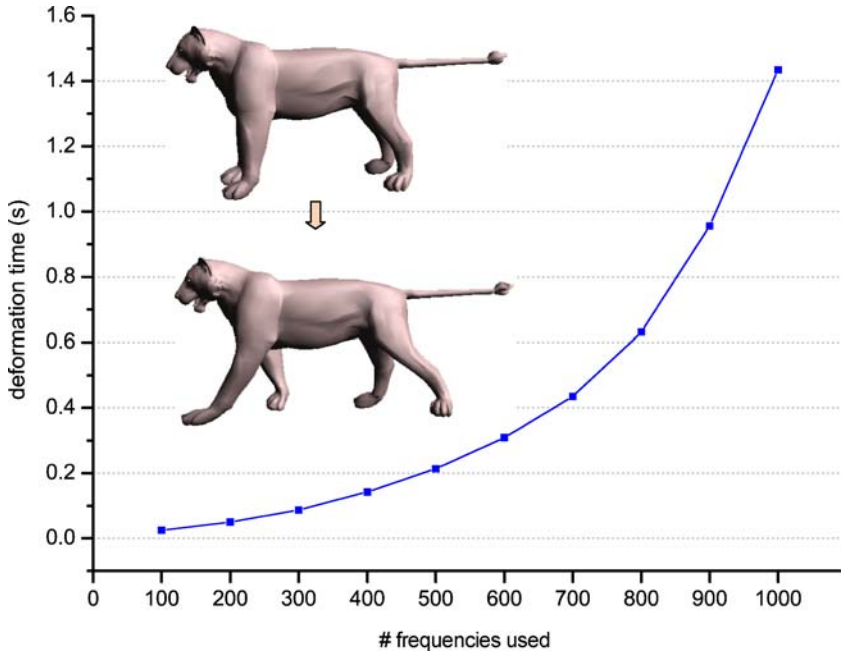

Fig. 5. The comparison of deformation timing using different numbers of frequency components

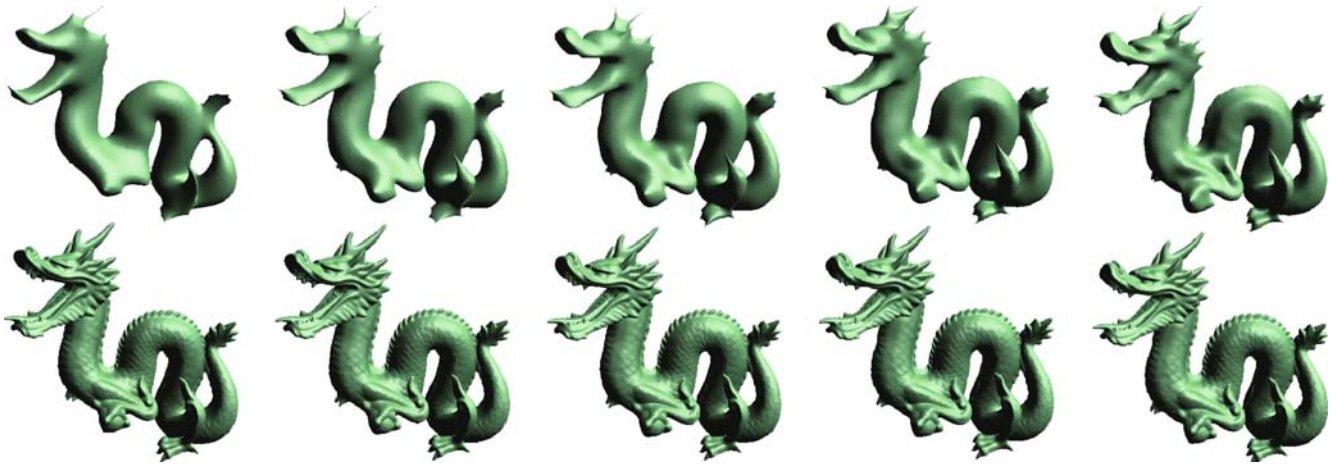

Fig. 6. The comparison of deformation results using different numbers of frequency components. From *left* to *right*, the number of frequencies used are 100, 200, 300, 500 and 900. The *top row* shows the results before the details are added back. The original pose of the dragon is shown in Fig. 2

The program is developed using Visual C++.NET 2005. For solving the dense linear system in Eq. 15, we use the LU-solver from *Numerical Recipe* [15]. For solving the sparse linear system of adding details, we use the CSparse library, which is good enough for our algorithm since the factorization can be pre-computed, and what we need is just a simple back substitution for adding details at the run-time.

Table 1 lists the details of the models we use in our experiments, and the timing of different steps in our algorithm, including both the pre-computation and the run-time deformation steps. For comparison, we also list the timing of the deformation without using MHT, i.e. by solving Eq. 14 directly. We use the bi-conjugate gradient solver in *Numerical Recipe* to solve this sparse linear system. It is evident that our new algorithm is orders of magnitude faster than the approach without using MHT.

The number of the frequencies (m) used to build the smoothed model is the key factor in our algorithm. It determines the speed of our algorithm. Figure 5 shows the speeds with different m for the Lion model. It is evident that with the increase of m, the deformation time of our algorithm increases accordingly. However, from our experiments, the difference between the deformation results of different m ($m \geq 100$) is indistinguishable (see Fig. 6 for the examples of the Dragon model). So a small m ($m = 100$) can be enough for most of our experiments.

6 Conclusion and future work

In this paper, we introduce a new spectral algorithm for geometric model deformation. The eigenfunctions of the Laplace–Beltrami operator give orthogonal bases for parameterizing the space of functions defined on the surface. Continuous deformations on the surfaces result in continuous changes of the spectrum. This algorithm uses the low-frequency components of the manifold surfaces to deform the smoothed model and add the high-frequency details back using deformation transfer techniques. With the help of manifold harmonics, we transfer the model deformation problem from space domain to frequency domain, and thus greatly reduce the size of the linear system to only hundreds of low-frequency components. So the computational speed is significantly enhanced over its spatial counterparts, to allow interactive manipulation of large triangle meshes on desktop PCs without any special hardware acceleration.

The graphics processing unit (GPU) is developing fast. The parallel architecture of the GPU suits the linear system very well. Some researchers have already utilized GPU-acceleration for model deformations [13, 27]. Currently, our algorithm is implemented on the CPU only. A part of our future work is to fit our spectral algorithms into the GPU pipeline to achieve even higher speed for deforming larger meshes.

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

References

1. Au, O.K.C., Tai, C.L., Liu, L., Fu, H.: Dual laplacian editing for meshes. IEEE Trans. Vis. Comput. Graph. **12**(3), 386–395 (2006)
2. Boier-Martin, I., Ronfard, R., Bernardini, F.: Detail-preserving variational surface design with multiresolution constraints. In: Proceedings of the 2004 Shape Modeling International, pp. 119–128. IEEE Computer Society, Washington, DC (2004)
3. Botsch, M., Kobbelt, L.: An intuitive framework for real-time freeform modeling. ACM Trans. Graph. **23**(3), 630–634 (2004)
4. Botsch, M., Pauly, M., Gross, M., Kobbelt, L.: PriMo: coupled prisms for intuitive surface modeling. In: Proceedings of the 4th Eurographics Symposium on Geometry processing, pp. 11–20. Eurographics Association, Aire-la-Ville, Switzerland (2006)
5. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Trans. Vis. Comput. Graph. **14**(1), 213–230 (2008)
6. Botsch, M., Sumner, R., Pauly, M., Gross, M.: Deformation transfer for detail-preserving surface editing. In: Proceedings of 11th International Fall Workshop Vision, Modeling & Visualization, pp. 357–364. Akademische Verlagsgesellschaft Aka, Aachen (2006)
7. Guo, X., Li, X., Bao, Y., Gu, X., Qin, H.: Meshless thin-shell simulation based on global conformal parameterization. IEEE Trans. Vis. Comput. Graph. **12**(3), 375–385 (2006)
8. Guskov, I., Sweldens, W., Schröder, P.: Multiresolution signal processing for meshes. In: Proceedings of ACM SIGGRAPH 99, pp. 325–334. ACM Press/Addison-Wesley Publishing Co., New York, NY (1999)
9. Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.Y., Teng, S.H., Bao, H., Guo, B., Shum, H.Y.: Subspace gradient domain mesh deformation. ACM Trans. Graph. **25**(3), 1126–1134 (2006)
10. Karni, Z., Gotsman, C.: Spectral compression of mesh geometry. In: Proceedings of ACM SIGGRAPH 2000, pp. 279–286. ACM Press/Addison-Wesley Publishing Co., New York, NY (2000)
11. Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.P.: Interactive multi-resolution modeling on arbitrary meshes. In: Proceedings of ACM SIGGRAPH 98, pp. 105–114. ACM Press/Addison-Wesley Publishing Co., New York, NY (1998)
12. Lipman, Y., Sorkine, O., Levin, D., Cohen-Or, D.: Linear rotation-invariant coordinates for meshes. ACM Trans. Graph. **24**(3), 479–487 (2005)
13. Marinov, M., Botsch, M., Kobbelt, L.: GPU-based multiresolution deformation using approximate normal field reconstruction. J. Graph. Tools **12**(1), 27–46 (2007)
14. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential geometry operators for triangulated 2-manifolds. In: Hege, H.C., Polthier, K. (eds.) Visualization and Mathematics III, pp. 35–57. Springer, Heidelberg (2002)
15. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes – The Art of Scientific Computing, 3rd edn. Cambridge University Press, New York, NY (2007)
16. Reuter, M., Wolter, F.E., Peinecke, N.: Laplace–Beltrami spectra as shape-DNA of surfaces and solids. Comput.-Aided Des. **38**(4), 342–366 (2006)
17. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.:

Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 175–184. ACM Press, New York, NY (2004)
18. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. ACM Trans. Graph. **23**(3), 399–405 (2004)
19. Taubin, G.: A signal processing approach to fair surface design. In: Proceedings of ACM SIGGRAPH 95, pp. 351–358. ACM Press/Addison-Wesley Publishing Co., New York, NY (1995)
20. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. Comput. Graph. (Proceedings of ACM SIGGRAPH 90) **21**(4), 205–214 (1987)
21. Vallet, B., Lévy, B.: Spectral geometry processing with manifold harmonics. Technical report, ALICE – INRIA Lorraine, Nancy, France (2007)
22. Welch, W., Witkin, A.: Variational surface modeling. Comput. Graph. (Proceedings of ACM SIGGRAPH 92) **26**(2), 157–166 (1992)
23. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with poisson-based gradient field manipulation. ACM Trans. Graph. **23**(3), 644–651 (2004)
24. Zayer, R., Rössl, C., Karni, Z., Seidel, H.P.: Harmonic guidance for surface deformation. Comput. Graph. Forum **24**(3), 601–609 (2005)
25. Zhang, H., van Kaick, O., Dyer, R.: Spectral methods for mesh processing and analysis. In: Proceedings of Eurographics State-of-the-art Report, pp. 1–22. Eurographics Association, Prague (2007)
26. Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H.Y.: Large mesh deformation using the volumetric graph laplacian. ACM Trans. Graph. **24**(3), 496–503 (2005)
27. Zhou, K., Huang, X., Xu, W., Guo, B., Shum, H.Y.: Direct manipulation of subdivision surfaces on GPUs. ACM Trans. Graph. **26**(3), 91 (2007)
28. Zorin, D., Schröder, P., Sweldens, W.: Interactive multiresolution mesh editing. In: Proceedings of ACM SIGGRAPH 97, pp. 259–268. ACM Press/Addison-Wesley Publishing Co., New York, NY (1997)

GUODONG RONG received his B.Eng. degree and M.Eng. degree both in Computer Science from Shandong University in 2000 and 2003 respectively, and his Ph.D. degree in Computer Science from National University of Singapore in 2007. Currently, he is a research scholar at the Department of Computer Science, University of Texas at Dallas. His research interests include computer graphics, visualization and image processing, especially general-purpose usage of graphics hardware (GPGPU) and geometry model processing. For more information, please visit http://www.utdallas.edu/∼guodongrong.

YAN CAO received her B.S. in Computational Mathematics from Peking University, China in 1996, M.S. in Mathematics from the University of Iowa in 1997 and Ph.D. in Applied Mathematics from Brown University in 2003. Currently, she is an assistant professor in the Department of Mathematical Sciences at the University of Texas at Dallas. Her research interests include computer vision and pattern theory, especially shape analysis and shape modeling.

XIAOHU GUO is an assistant professor of computer science at the University of Texas at Dallas. He received the Ph.D. degree in Computer Science from the State University of New York at Stony Brook in 2006. His research interests include computer graphics, animation and visualization, with an emphasis on geometric and physics-based modeling. Dr. Guo's research is funded by the National Science Foundation, and he is currently the principal investigator for projects related both to physical simulation of deformable models and to geometric mapping of surface and volumetric models. For more information, please visit http://www.utdallas.edu/∼xguo.