Computer Aided Geometric Design 35–36 (2015) 56–68

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

HIRM: A handle-independent reduced model for incremental mesh editing

Yirui Wu [a], Oscar Kin-Chung Au [b], Chiew-Lan Tai [c], Tong Lu [a,*]

[a] State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China
[b] School of Creative Media, City University of Hong Kong, Hong Kong
[c] Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Hong Kong

ARTICLE INFO

Article history:
Available online 24 March 2015

Keywords:
Incremental editing
Mesh deformation
Reduced model

ABSTRACT

Most existing handle-based mesh deformation methods require costly re-computation for every handle set updating, namely, adding or removing of handles on the mesh surface. In this paper, we propose a reduced deformation model that is independent of handle configuration, allowing users to dynamically update the handle set without noticeable waiting time. We represent the deformation space of a mesh as propagation fields defined by only the mesh geometry, independent of the handle set. We define the propagation fields as selected eigenvectors of the Laplacian operator and adopt the transformations of isolines sampled from the fields as the deformation descriptors. In this way, the deformation descriptors are pre-computed before handle specification. During interactive manipulation, constraints generated from the handles are incorporated into the deformation system in real time. Our method therefore supports incremental mesh editing where the user can freely define different handle sets to edit different parts of the shape without waiting for long re-computation. Our reduced model is scalable since the updating time per iteration is independent of the mesh size and the number of handles. We demonstrate the effectiveness of the proposed deformation method and compare its performance with related reduced deformation models.

1. Introduction

Handle-based surface editing (Sorkine et al., 2004; Botsch and Sorkine, 2008) is an intuitive editing metaphor as it supports specifying arbitrary handles directly on the surface and a simple click-and-drag user interface. During editing, the surface vertices are transformed to compose the desired deformation along the mesh surface. The main challenge is to find the proper transformations which induce plausible deformation respecting the handle constraints. Most handle-based deformation methods, such as Zayer et al. (2005), Popa et al. (2006), Au et al. (2007), rely on deformation propagations that are constrained along paths connecting the user-specified handles. These methods require precomputation of the handle-driven propagation fields or functions, which means the entire deformation system has to be rebuilt whenever the user updates the handle set. This long rebuilding time (e.g., in Au et al., 2007, 15 s for a model with 25 000 vertices and 4 handles) is the major bottleneck of these systems and discourages users from adopting an incremental editing workflow, whereby

* Corresponding author.
E-mail addresses: wuyirui1989@163.com (Y. Wu), kincau@cityu.edu.hk (O. Kin-Chung Au), taicl@cse.ust.hk (C.-L. Tai), lutong@nju.edu.cn (T. Lu).

http://dx.doi.org/10.1016/j.cagd.2015.03.001
0167-8396/

different handle sets can be freely specified to edit different parts of the shape. This limits the usability of the handle-based deformation metaphor in general.

In this paper, we propose a novel and efficient handle-independent reduced model, which we call HIRM, to enable instant handle set updating. The user is able to dynamically modify the handle set on the fly and the deformation system is updated without noticeable waiting time (<1 s). Our key idea stems from the hypothesis that the global deformation space of a given model can be well described by a small number of *propagation fields* defined only by the mesh geometry, independent of the handle configuration and the number of vertices. Essentially, the deformation propagation is described by the low-frequency geometry components, rather than depend on the constraints defined by the handles. We represent such geometry-based propagation as *deformation descriptors* and design our mesh deformation framework in two stages.

In the pre-processing stage, we adopt a specific subset of the eigenvectors of Laplacian matrix to capture the deformation space of the input shape. Selecting a subset of the eigenvectors for deformation propagation is necessary because using all the eigenvectors (theoretically equivalent to the number of vertices) would potentially requires a large computation cost. We select the first few eigenvectors sorted in ascending eigenvalues, which capture the global geometric and topological information of the shape with little loss. A small set of isolines is then sampled on the selected propagation fields, and the transformations of the sampled isolines constitute the descriptors of local deformation. The deformation descriptors are the variables to be solved in our reduced deformation system, and the deformed vertex positions are computed via interpolation of the deformation descriptor transformations.

During the interactive handle manipulation stage, constraints generated from the handles are incorporated into the deformation system. Unlike previous methods, the handles are specified by the user only to introduce movements that induce the deformation, rather than to describe the deformation propagation. By positioning and orientating the handles, the Laplacian energy minimization is subjected to the handle's vertices, enforced as soft constraints (Huang et al., 2006). Note that the propagation fields and deformation descriptors, which capture the low-frequency spectral geometry, remain unchanged during incremental deformation. This is reasonable as they represent the deformation space which are mostly determined by the model's geometry structure, rather than the embedding of the model. We pre-compute them before handle specification, resulting in efficient updating of the Laplacian deformation system with any new handle configuration. Our system thus allows dynamic handle set updating and incremental mesh editing.

The main contributions of the paper are:

- Introduction of the handle-independent propagation fields and deformation descriptors based on eigenvectors of Laplacian matrix, which successfully capture the deformation space of the given shape.
- An interactive, efficient and scalable mesh deformation framework that supports instant handle set updating and incremental editing.

We show that HIRM is faster in system re-computation time than the previous handle-based surface deformation methods (Au et al., 2007; Xu et al., 2009). Specifically, HIRM requires time linear in the number of vertices for system reconstruction and the per-iteration updating cost is independent of the number of handles or vertices, thus allowing scalable incremental editing.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 presents an overview of the algorithm. Details of the proposed propagation fields, deformation descriptors, and the integration of the handle-independent reduced model into a Laplacian editing framework are discussed in Section 4. Section 5 presents the experimental results and discussions. Finally, Section 6 concludes the paper.

2. Related works

Surface-based deformation (Au et al., 2006; Sorkine and Alexa, 2007; Botsch and Sorkine, 2008) directly operates on mesh surfaces, allowing users to intuitively deform a mesh through manipulating the handles specified on the mesh surface. Typical surface-based methods encode shape features by some kind of deformation energy and formulate deformation as a global optimization problem to solve for a proper transformation to each vertex. This deformation energy can be represented as a function of shape descriptor defined on the surface. Laplacian coordinates are one of widely used shape descriptors, which describe the local shape characteristics through differential properties of the surface, transforming the original problem to the minimization of the changes of Laplacian coordinates.

Existing methods for solving the minimization problem can be categorized into two classes: linear methods and non-linear methods. Early *linear methods* (Botsch and Kobbelt, 2004; Sorkine et al., 2004; Botsch and Sorkine, 2008) were proposed to achieve low computation costs by linearizing the problem using either uniform weights (Yu et al., 2004) or non-uniform weights (Popa et al., 2006). However, linear methods can only achieve approximate results, making these methods unsuitable for large deformations.

Non-linear methods are able to achieve more precise results. For example, the method of Sorkine and Alexa (2007) iteratively minimizes the elastic energy function to construct transformations, explicitly enforcing local rigid rotations to improve deformation quality. In a similar way, Chao et al. (2010) minimize the L2 distance for the rotation group to achieve elastic deformation. However, adopting non-linear energies for surface modeling often leads to time consuming energy minimization calculations. To avoid this, the Laplacian iterative framework, as adopted in Au et al. (2006), Huang et al. (2006), is

Fig. 1. We develop a handle-independent reduced model for interactive incremental mesh editing. Users can dynamically update the handle set in real time and edit different parts of the shape in any order, without a long waiting time for reconstructing the deformation system. Left: original model with initial handle set. Middle left: intermediate deformation results. Middle right: new handle set (new handles in green). Right: final deformed model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

proposed, which obtains proper transformations via alternatively updating vertex positions and Laplacian coordinates. Au et al. (2007) further sample the handle-driven harmonic fields to obtain isoline sets as a reduced model and perform iterative updating in the reduced domain, thus achieving fast convergence and low iterative updating cost. However, these handle-driven Laplacian methods (Zayer et al., 2005; Popa et al., 2006; Au et al., 2007) do not address the problem of real-time updating of handle constraints, and the re-computation of the handle-driven propagation fields or deformation energy still remains a major bottleneck for these methods.

As an alternative to using differential coordinates, physical models, which are composed of elastic shells, have been proposed to achieve numerical robustness and physically plausible results. The PriMo method (Botsch et al., 2006) extends the triangles of a mesh into elastic prisms, forcing the mesh to transform rigidly under the constraints of virtual volumetric prisms. The drawback of these physical approaches lies in that the optimization of the formulation is still very challenging especially during interactive editing. Recently, Hildebrandt et al. (2011) construct a reduced low-dimensional shape space for deformations. With energy approximation and an efficient quasi-Newton solver, they search the reduced subspace to get optimized convergence results and successfully construct an interactive physical deformation system. However, it has to adopt a large number of vectors for constructing the shape subspace, which results in time consuming pre-computations. Later, von Tycowicz et al. (2013) propose a model reduction technique which approximates the reduced forces by constructing the whole reduced shape space of deformable objects in the first iteration. Their technique can generate deformation space with smaller approximation error than Hildebrandt et al. (2011). Even though it requires a thousand of training poses for each input mesh, it replaces the expensive computation of modal derivatives in Hildebrandt et al. (2011) by a much faster scheme such that the cost of the subspace construction is reduced to a level that is comparable to the cost of the subspace construction of our method.

Most relevant to our work, Fisher et al. (2007) propose an interactive framework for designing harmonic vector fields, which allows dynamic changes of user constraints during interaction. However, this approach is not suitable for handle-based deformations since users are required to define the desired propagation fields manually, which is tedious and sometimes difficult even for experienced users. Xu et al. (2009) adopt a penalty method to convert the handle constrained problem into an unconstrained one, which is solved in the least-square sense. Then they use the supernodal algorithm (Davis and Hager, 2009) to update the corresponding Cholesky factorization, resulting in dynamic updating of harmonic fields. However, to facilitate interactive deformations, their deformation system with the new reduced model has to be built with updated harmonic fields, which essentially is still a costly task for large meshes.

Eigenvectors of Laplace operator have been proved effective in characterizing meaningful global geometric information of the input mesh for multiresolution editing (Rong et al., 2008a, 2008b; Rustamov, 2009). Dey et al. (2012) use eigenvectors to create an implicit skeleton and shape deformation is achieved by adding properly transformed geometry details to the deformed skeleton. However, it requires up to 300 eigenvectors to represent the skeleton and geometry details, which is a costly pre-computation task for large meshes.

3. System overview

In this paper, we propose a handle-independent reduced model for incremental mesh editing (see Fig. 1). Our method is surface-based, preserving local mesh geometry and allowing shape deformation through handle manipulation in an incremental editing manner. To build a reduced model of the input shape, we first select a small subset of eigenvectors of the Laplacian matrix as the propagation fields which capture the desired low-frequency deformation space of the shape. Like Au et al. (2007), we consider the isolines sampled on the fields as a reduced structure to describe the deformation space, such that the transformation of each isoline describes the local deformation of the local geometry and they are the unknowns to be solved in our deformation system. We therefore refer to the transformations of the isolines as the deformation descriptors. The final deformed vertex positions are interpolated from the isoline transformations. We introduced a geometry-aware weighting scheme based on the spectral coefficients of the input mesh geometry to blend the transformations of the isolines smoothly, producing plausible deformation results.

Fig. 2 shows the flow of computation of our deformation system. The computation of the propagation fields, isoline sampling and the interpolation process only depends on the input shape but not the handle configuration, leading to a reduced linear system with only isoline transformations as the unknowns. Therefore we could construct this reduced linear system in the pre-processing step, before specifying any handle or constraint. During interactive manipulation, constraints generated from the users-specified handles are incorporated into the reduced linear system on the fly, resulting in a deformation sys-

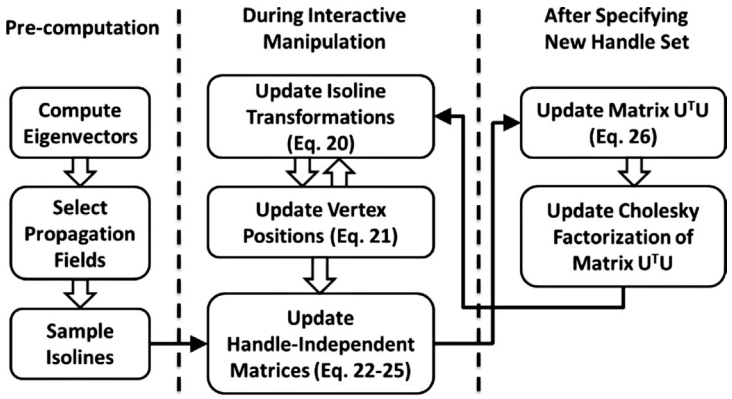

Fig. 2. The flow of the computation of our deformation system. Our system uses a two-levels of pre-computation to achieve instant handle updating. First, our system pre-computes the propagation fields and samples the isolines sets before the editing, this information is unchanged during the whole editing process. During interactive manipulation, our system updates the isoline transformation and vertex positions alternatively to fit with the manipulated handles, also the geometry-aware handle-independent matrices of the deformation system are computed with the current geometry in background (Eqs. (22)–(25)). These matrices are the pre-computed parts of the new deformation system when the handle set is updated. Once a new handle set is specified, the new deformation system can be constructed by integrating the new handle constraints with those pre-computed matrices (Eq. (26)), thus minimizing the computation cost and achieving instant handle updating.

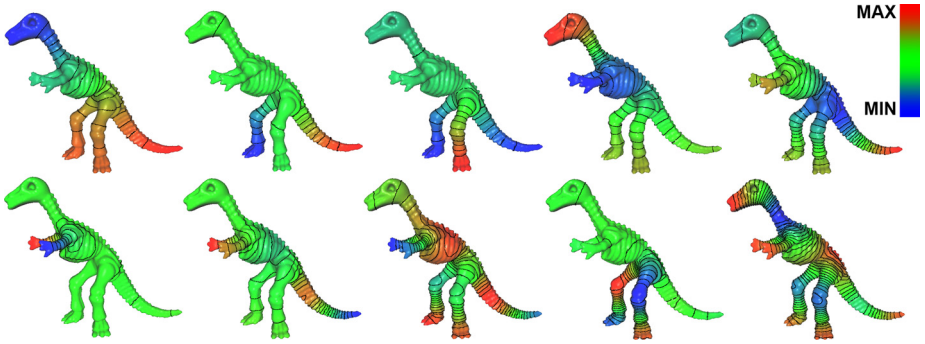

Fig. 3. Plots of the first 10 non-constant eigenvectors of the dinosaur model with 20 sampled isolines. Values from maximum to minimum are mapped to corresponding colors from red to blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

tem that supports dynamic and instant updates. This means users are able to freely add or remove handles during editing, thus achieves an incremental editing work flow. To achieve instant handle updating, we introduce a background computation approach which reduces users' waiting time and ensures that the newly constructed deformation system reflects the current edited geometry.

4. Handle-independent reduced model

We first introduce a strategy for selecting a subset of eigenvectors to construct the propagation fields, and then describe in detail how we construct the deformation descriptors. Next, we describe the interpolation process of computing the deformed vertex positions from the deformation descriptors. Finally, we present the iterative Laplacian deformation that facilitates incremental mesh editing.

4.1. Propagation fields

The key idea of HIRM is to represent the potential deformation space of the input model as a small set of propagation fields defined by specific components of the mesh geometry. We consider the mesh geometry as a discrete mesh signal and apply spectral analysis to discover the desirable components. Namely, we select a small subset of eigenvectors of the discrete Laplacian operator as the desirable propagation fields.

The discrete Laplacian operation of a surface mesh is represented as an $N \times N$ sparse matrix L, where N is the vertex count of the mesh (Sorkine, 2006). The Laplacian matrix has N eigenvectors of size N, where each eigenvector e_i represents a scalar function on the mesh with the value $e_i^{(v)}$ assigned to each vertex v. The eigenvectors of the Laplacian matrix form a spectrum representing the mesh geometry. Their corresponding eigenvalues are the natural frequency representing the scales of the geometry features encoded by the eigenvectors. For example, in Fig. 3, we can see that the eigenvectors with

smaller eigenvalues have a smoother variation in the global shape while the eigenvectors with larger eigenvalues have rapid variation respecting geometry features of smaller scales.

A major issue to address is to select a proper subset of the eigenvectors that is able to sufficiently represent the geometry information and capture the desired low-frequency deformation space while keeping its size small enough for efficient computation.

We select the eigenvectors based on their eigenvalues. Specifically, we assign scores based on the eigenvalues and select the first $i-1$ eigenvectors with scores smaller than a threshold φ:

$$E = \{e_2, \ldots, e_i | \arg\limits_i S_i < \varphi\} \tag{1}$$

where S_i is the score of the eigenvector e_i defined as:

$$S_i = \frac{\sqrt{\lambda_i}}{\sqrt{\lambda_2}} \tag{2}$$

Note that λ_2 is the first non-zero eigenvalue and its corresponding eigenvector, also called the Fielder vector, represents the global shape component with the smoothest shape variation and the smallest natural frequency. Therefore we use the ratio between the eigenvalue and λ_2 as the score of the corresponding eigenvector such that an eigenvector with a smaller score means represents a more global shape component and plays a more important role in deformation.

4.2. Deformation descriptors

In this subsection, we describe how we construct the deformation descriptors that represent the local transformations of shapes. Our goal is to construct deformation descriptors that are independent of the vertex distribution of the mesh model and the handle configuration specified by the user, thereby supporting efficient and incremental editing with different handle sets.

Recall that each propagation field, as the nature of eigenvectors of Laplacian matrix, represents the variation of shape along the mesh surface in different orientation and frequency. Like in Zayer et al. (2005), Au et al. (2007), we assume that all surface points on a specific isoline of a propagation field receive the same influence of transformation propagated in that field. Therefore, to represent the local transformation being applied to the mesh in a particular orientation specified by a propagation field, we uniformly sample each field as a set of isolines, and consider the transformation associated to each isoline as a deformation descriptor. These descriptors approximate the deformation space induced by the geometry of the input shape. We denote the transformation set

$$\{r_{i,j} \mid 1 \le i \le n_{field},\ 1 \le j \le n_{iso},\} \tag{3}$$

as constituting the deformation descriptors of the deformation system, where $r_{i,j}$ is a 3×4 transformation matrix associated with the j-th isoline of the propagation field e_i, with isovalue $s_{i,j} = (j-1)/(n_{iso}-1)$; $n_{field} = |E|$ is the number of propagation fields selected and n_{iso} is the number of isolines sampled in each propagation field. We use $n_{iso} = 20$ for all examples shown in this paper, which we found to be a good balance between deformation quality and computation cost.

The sampled isolines form a set of sparse but well-distributed control variables, serving as a generalized structure such that their associated transformations implicitly represent the smooth and low-frequency deformation space of the mesh. This allows us to encode the possible mesh deformation using a small set of transformation variables, without explicitly solving for individual deformed vertex positions. In the next subsection we will describe how we represent the deformed vertex positions in term of the deformation descriptors.

Note that our idea of the deformation descriptors is similar to the isoline-based deformation framework (Au et al., 2007) in that the transformations of the isolines of the scalar harmonic fields are considered as the reduced variables in the mesh deformation system. However, in Au et al. (2007) the isolines are sampled from harmonic fields defined by the user-specified handle constraints. Thus the reduced model (constructed with the handle-based isolines) is only able to describe the deformation space corresponding to a particular handle set. In contrast, the eigenvector-based isolines capture the general deformation space in a global manner in the sense that the propagation fields and the isolines remain unchanged during incremental editing with successive updates of handle configuration.

4.3. Deformation interpolation

We now describe the interpolation from the deformation descriptors to compute the vertex positions of the deformed mesh. Like Au et al. (2007), we apply a two-level linear interpolation which first interpolates the transformations of isolines of the same propagation field, and then average the interpolated transformations among all fields to form the final vertex transformation.

We first compute the transformation $r_i^{(k)}$ of vertex k corresponding to the propagation field e_i by linearly interpolating the transformations of the neighboring isolines $r_{i,j}$ and $r_{i,j+1}$:

$$r_i^{(k)} = (1-\alpha_i^{(k)})r_{i,j} + \alpha_i^{(k)} r_{i,j+1}, \tag{4}$$

with the interpolation coefficients

$$\alpha_i^{(k)} = \frac{e_i^{(k)} - s_{i,j}}{s_{i,j+1} - s_{i,j}}, \tag{5}$$

where $e_i^{(k)}$ is the field value associate with the field e_i at vertex k, such that $s_{i,j} \le e_i^{(k)} \le s_{i,j+1}$. As each propagation field describes the geometry information in different frequency and magnitude, next we need to define a proper weighting scheme to combine the vertex transformations corresponding to each field.

As the Laplacian matrix is symmetric, its eigenvectors are orthogonal, which means they can serve as the vector basis to represent the input mesh geometry. Specifically, the x-coordinates of the mesh geometry, denoted as the mesh signal X, can be encoded into the *spectral transform coefficients* with respect to the eigenvector $X = \sum_{i=1}^{n} \xi_{x,i} e_i$, where the coefficients ξ_x are obtained by projecting the original mesh geometry onto the corresponding eigenvector, i.e., $\xi_{x,i} = X \cdot e_i$. Similarly, the y- and z-coordinates of the mesh geometry are encoded in the same way. It is well known that such encoding is dominated by the first few spectral transform coefficients, that is, an approximate spectral reconstruction X' of the original geometry X can be obtained by using only the first m eigenvectors and spectral coefficients (Karni and Gotsman, 2000; Sorkine, 2006) as $X' = \sum_{i=1}^{m} \xi_{x,i} e_i$ with $m \ll n$. The L_2 error of the reconstruction is given by:

$$\|X - X'\| = \left\| X - \sum\nolimits_{i=1}^{m} \xi_{x,i} e_i \right\| = \sqrt{\sum\nolimits_{m+1}^{n} \xi_{x,i}^2}. \tag{6}$$

In other words, the spectral coefficients indicate how important the corresponding eigenvectors are with respect to the original geometry.

It is natural to use the scale of the spectral coefficients as a criteria to quantify the importance ω_i of eigenvector e_i, i.e.,

$$\omega_i = \sqrt{\xi_{x,i}^2 + \xi_{y,i}^2 + \xi_{z,i}^2}. \tag{7}$$

Therefore, it is reasonable to adopt the importance values ω_i as the weights for integrating the transformations associated to different propagation fields. Hence, we define the deformation transformation of vertex k as:

$$r^{(k)} = \sum\nolimits_{i=1}^{n_{field}} \bar{\omega}_i r_i^{(k)}, \tag{8}$$

where $\bar{\omega}_i$ are the normalized importance values such that $\sum_{i=1}^{n_{field}} \bar{\omega}_i = 1$. The deformed vertex positions x_k are then computed as

$$x_k = r^{(k)} x_k^0, \tag{9}$$

where x_k^0 is the original vertex position (represented in homogeneous coordinates).

Eq. (9) can be written in matrix form $X = WR$, where X is one of the deformed geometry coordinates (x-, y- or z-coordinates of the mesh geometry), R is the column vector constructed from the transformations $\{r_{i,j}\}$, and W is the matrix constructed from the $\{\bar{\omega}_i\}$, $\{\alpha_i^{(k)}\}$ and $\{x_i^0\}$. Note that W can be pre-computed before specifying handle specification. Essentially, W serves as a generic reduced model that simplifies the global deformation problem from solving for all the vertex positions to finding the transformations for a small set of deformation descriptors.

4.4. Iterative Laplacian deformation

Recall that in Laplacian editing framework the deformation task is achieved by minimizing the difference of Laplacian coordinates before and after deformation:

$$\arg\min_X \|LX - \sigma(X)\|^2, \tag{10}$$

where $\sigma(X)$ is the Laplacian coordinates which are non-linearly dependent on the deformed vertex position X. By replacing the unknown vertex position X with the reduced model, we can write the minimization of difference of the Laplacian coordinates before and after deformation as

$$\arg\min_R \|LWR - \sigma(WR)\|^2 + \mu^2 \|MR\|^2, \tag{11}$$

where the latter term is a smooth energy that enforces the smoothness of the deformation interpolation, which is constructed from the following soft constraints:

$$r_{i,j} - r_{i,j+1} = 0, \quad 1 \le i \le n_{field}, 1 \le j \le n_{iso} - 1. \tag{12}$$

This enforces the transformations associated with neighboring isolines to be similar to one another, and μ is a weight that balances the two energy terms (we use $\mu = 0.01$ in our experiment). With the user-specified handle constraints, the deformed mesh can be updated by solving the resulting linear system from Eq. (11) in a least-squares sense:

$$\begin{bmatrix} LW \\ \mu M \\ \Phi W \end{bmatrix} R = \begin{bmatrix} \sigma(WR) \\ 0 \\ H \end{bmatrix}, \tag{13}$$

where $\Phi WR = \Phi X = H$ indicates the constraints of the handle positions. Thus the deformed vertex positions can be computed by solving Eq. (13) as follows:

$$U = [W^T L^T \quad W^T \Phi^T \quad \mu M^T]^T, \tag{14}$$

$$\tilde{R} = (U^T U)^{-1}[W^T L^T \sigma(WR) + W^T \Phi^T H], \tag{15}$$

$$\tilde{X} = W\tilde{R}, \tag{16}$$

where $\tilde{R}$ and $\tilde{X}$ are the vector representations of the deformation descriptors and vertex positions of the deformed mesh. Since the local features on a mesh are deformed and rotated during editing, the Laplacian coordinates $\sigma(WR)$ must somehow be updated to match the new deformed surface. We adopt an iterative Laplacian editing framework (Au et al., 2006; Huang et al., 2006) to achieve the rotation invariant deformation by iteratively and alternatively updating $\sigma(WR)$, $\tilde{R}$ and $\tilde{X}$, according to the current deformed surface. Note that the updating of $\tilde{R}$ involves solving a linear system of size $n_{field} \times n_{iso}$, which is small and can be computed very efficiently.

Similar to the deformation interpolation which encodes the deformed vertex positions into the deformation descriptors, we also encode the Laplacian coordinates $\sigma(WR)$ into the local rotation of the sampled isolines, which can be expressed in matrix form as $\sigma(WR) = W_\sigma R_\sigma$, where R_σ is the column matrix consisting of local rotation of isolines $\{\widehat{r}_{i,j}\}$, and the W_σ is a matrix constructed from the original Laplacian coordinates $\{\sigma(X^0)\}$, $\{\bar{\omega}_i\}$ and $\{\alpha_i^{(k)}\}$. To compute $\{\widehat{r}_{i,j}\}$, we only need to know the local rotation of the isolines. Let $\Gamma_{i,j}$ denote the set of triangles that the j-th isoline of propagation field e_i pass through. We compute the local rotation $\{\widehat{r}_{i,j}\}$ (represented by a 3×3 matrix without translation) by first solving the following system in a least-squares sense:

$$\tilde{r}_{i,j}(v_{k,l} - c_k) = \tilde{v}_{k,l} - \tilde{c}_k, \quad \text{for } k \in \Gamma_{i,j}, l \in \{1, 2, 3\} \tag{17}$$

where $v_{k,l}$ and $\tilde{v}_{k,l}$ are the original and current deformed vertex positions of triangle k, respectively, and c_k and $\tilde{c}_k$ are the original and current deformed vertex center of the triangle, respectively. The isoline rotation can then be computed as:

$$\widehat{r}_{i,j} = s_{i,j} * orth(\tilde{r}_{i,j}), \tag{18}$$

where $orth()$ represents the orthogonal normalization that extracts the rigid rotation from the transformation $\tilde{r}_{i,j}$ (computed by SVD decomposition $\tilde{r}_{i,j} = U\Sigma V$), and $s_{i,j}$ is the scale factor for canceling the shrinking effect due to the linear interpolation of transformation matrix we adopted in HIRM. The local scaling of the current deformed shape is canceled by implicitly adjusting the scales of the updated Laplacian coordinates with the scale factor $s_{i,j}$. More specifically, $s_{i,j}$ is defined by the trace of diagonal matrix in the SVD decomposition of $\tilde{r}_{i,j}$:

$$s_{i,j} = 3/trace(\Sigma). \tag{19}$$

By replacing the Laplacian coordinates $\sigma(WR)$ with $W_\sigma R_\sigma$, the iterative updating of the deformation system become:

$$\tilde{R} = (U^T U)^{-1}[W^T L^T W_\sigma R_\sigma(X) + W^T \Phi^T H], \tag{20}$$

$$\tilde{X} = W\tilde{R}. \tag{21}$$

Note that the computation of R_σ only involves a small subset of vertices corresponding to the triangle set $\Gamma_{i,j}$, therefore we do not need to update all vertex positions until the intermediate deformed mesh is needed for rendering. This leads to efficient updating for each iteration and a scalable mesh deformation framework.

4.5. Incremental mesh editing

To facilitate mesh editing in an incremental manner, we first pre-compute the propagation fields. The deformation system is constructed and updated with the new handle constraints whenever the user specifies a new set of handles. The pre-computation cost for finding the propagation fields mainly depends on the mesh size and the shape complexity, which could be costly for large meshes but the propagation fields can be computed and stored in advance before interactive editing.

Whenever the user updates the handle configuration, the current editing result has to be retained. This means the matrices W and U have to be recomputed using the vertex positions and Laplacian coordinates of the current deformed

mesh in order to reflect the new geometry in the deformation system. We observe that the construction of W and U can be decomposed into two parts: the handle-dependent submatrix ΦW and the handle-independent part that only depends on the current deformed geometry. Therefore, in order to reduce users waiting time, once a user finishes manipulating a handle and completes a step of editing, our system starts to compute the new values of the handle-independent part using the most updated geometry in the background.

Specifically, we use a background thread to compute the following handle-independent matrices using the current updated vertex positions X once the user finishes a handle manipulation step:

$$W = W(X) \tag{22}$$

$$\hat{U} = [W^T L^T \quad \mu M^T]^T, \tag{23}$$

$$T_1 = \hat{U}^T \hat{U} \tag{24}$$

$$T_2 = W^T L^T W_\sigma R_\sigma(X). \tag{25}$$

Since we already have the pre-computed propagation fields, the matrix construction of these handle-independent part can be done in time linear in the mesh size and the number of propagation fields. We observe that users typically need to spend at least a few seconds changing the viewpoint to examine the intermediate editing result and specify a new handle set, therefore the handle-independent part can typically be completely pre-computed in background before users finish specifying a new handle set.

When the user finishes specifying a new handle set, only the submatrix ΦW of the handle constraints is related to the new handle configuration and is the only part that needs to be updated. With the small size of ΦW, we can update the soft constraints caused by the new handle set very efficiently. Specifically, the new deformation system is constructed as follows:

$$U^T U = T_1 + (\Phi W)^T (\Phi W) \tag{26}$$

$$\tilde{R} = (U^T U)^{-1} [T_2 + W^T \Phi^T H] \tag{27}$$

$$\tilde{X} = W \tilde{R}. \tag{28}$$

The cost of this background pre-computation approach is that our system has to keep re-computing the handle-independent part while the user is editing the mesh model, as the most-recent geometry is needed to build the new deformation system. However, most of the computation is done in the background and hidden from users, hence the system updating can be done dynamically without any noticeable delay.

Note that even though the global shape and pose of the editing model is modified during incremental deformation, the system always adopts the eigenvectors defined by the *original* shape as the propagation fields throughout the editing process. This makes sense since these fields capture the low-frequency spectral geometry and represent the deformation space which is mostly determined by the model's geometry structure, rather than the embedding of the model.

5. Experiments and discussions

We demonstrate that our method can support effective deformation in an incremental manner. Fig. 4 shows several editing examples using our incremental deformation framework. The user specifies new handle sets on intermediate deformation results (middle left and middle right columns of Fig. 4) on the fly without long system rebuilding time (see accompanying video for interactive editing scenarios). The intermediate deformation is retained during the incremental editing process and different parts of the model can be edited by specifying different sets of handles. The deformations are naturally propagated in all the results due to our properly selected propagation fields, which are able to capture the low-frequency deformation space, allowing the movement of handles to dictate global deformation while the high-frequency details are well preserved by the iterative Laplacian deformation process.

Our deformation system supports both region handles and point handles (Au et al., 2006, 2007). Users can specify positional and/or orientation constraints by translating and rotating the region handles (e.g. the Armadillo and the hand models in Fig. 4). For point handles, the system is able to automatically determine natural orientations of local features based on the new positions of the point handles (e.g. orientations of the mouth corners of the Max Planck model are updated automatically).

5.1. Deformation quality analysis

Selecting a small number of propagation fields might lead to artifacts when editing local details or when the handles are placed too closed to each other. This problem could be solved by adopting more propagation fields. Fig. 5 shows an example where two handles are specified at both hands of the dinosaur model, and the right handle is translated away from the other hand. It is observed that when only 4 propagation fields are used, the deformation system is unstable and produces shearing artifacts. Using 6 propagation fields leads to a more stable deformation system and natural deformation

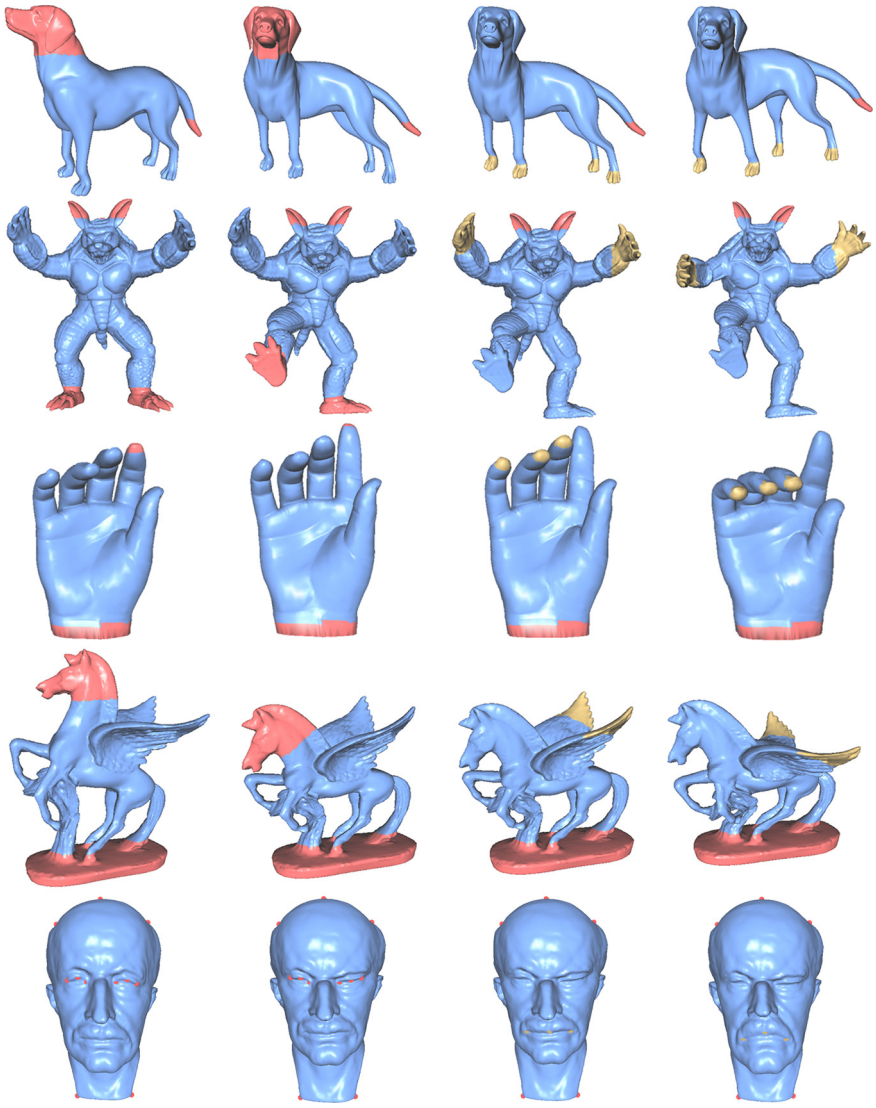

Fig. 4. Incremental editing results with the handle-independent reduced model. Left: original model with initial handle set. Middle left: intermediate deformation results. Middle right: a new user-specified handle set. Right: final deformation result.

result. The extra fields properly capture the geometry between the two hands of the model (Fig. 3 bottom left). Note that our algorithm automatically selects 6 propagation fields for the dinosaur model based on the ratio of the corresponding eigenvalues. The deformation results by using 8 and 20 propagation fields are also shown, demonstrating that the selection of more propagation fields does not always improve deformation quality.

In Fig. 6, we show the results for some extreme deformations used in Botsch and Sorkine (2008). We compare the deformation results with Botsch et al. (2006), Hildebrandt et al. (2011) and Au et al. (2007) by using the same handle constraints and transformations. The results of Hildebrandt et al. (2011) and Botsch et al. (2006) are taken from the respective papers for comparison. The Modal Analysis method (Hildebrandt et al., 2011) and the PriMo algorithm (Botsch et al., 2006) are physically-based methods and produce desirable results even with large rotations of handles. However, the PriMo algorithm involves more computation than the other methods in terms of running time since it has to perform prisms updates in each iteration and the number of prisms is the same as the number of triangles. The Modal Analysis method (Hildebrandt et al., 2011) is fast, but requires a long pre-computation time (reported 1067 s for generating a reduced space of 67 dimensions for a dragon model with 130K vertices). Like HIRM, the Modal analysis method uses eigenvectors to construct a subspace for deformation; however, the Modal Analysis method only assigns a weight for each eigenvector, while our method samples the selected eigenvectors as isolines, and assigns individual weights to the sampled isolines for the interpolation process. Therefore HIRM does not need as many eigenvectors for constructing the possible deformation space. Our method produces desirable deformation results similar to Au et al. (2007), while our deformation system allows dynamic handle set updating and supports incremental mesh editing.

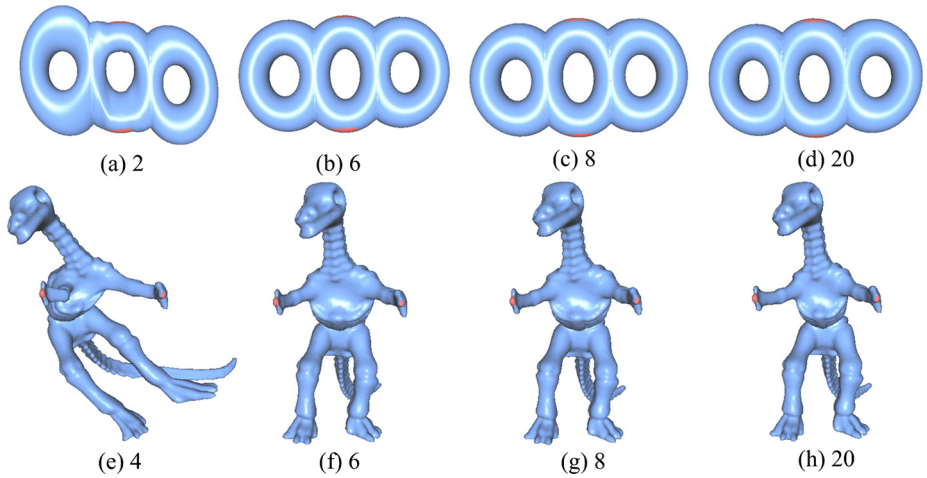

Fig. 5. Deformation results of using different number of eigenvectors for constructing the propagation fields. Observe that there are unnatural artifacts in (a) and (e) where too few eigenvectors are used, and using more eigenvectors does not necessarily improve the deformation quality. Our algorithm automatically selects sufficient number of eigenvectors shown in (b) and (f) so that plausible deformation is attained efficiently.

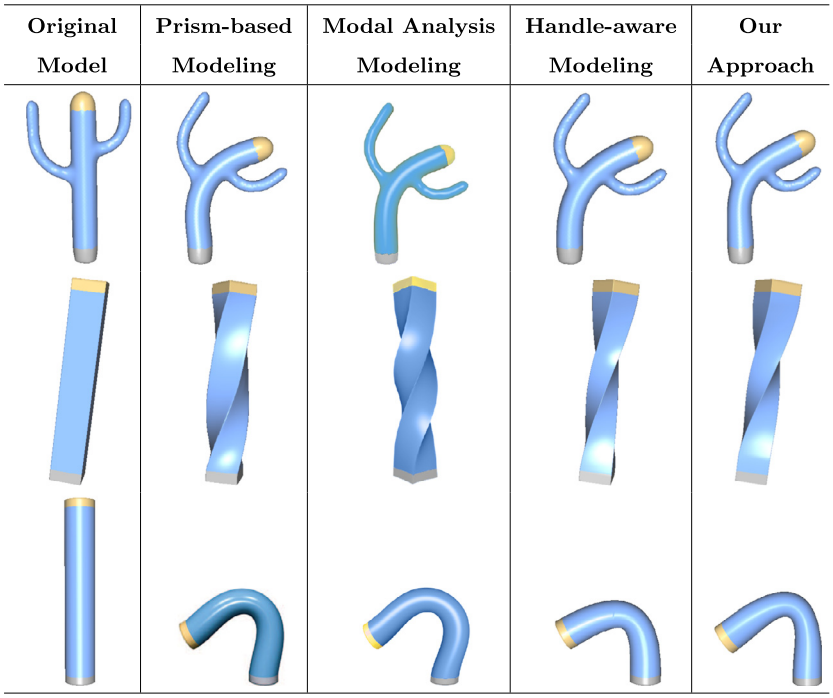

Fig. 6. Comparison with related methods. Physically-based methods (Botsch et al., 2006; Hildebrandt et al., 2011) preserve volume better; however, they need costly computation. Our method produces desirable results similar to Au et al. (2007).

5.2. Performance analysis

We divide the computing time of HIRM into the eigenvectors pre-computing time t_e, the background pre-computing time t_b, the handle updating time t_h and the per-iteration update time t_i. Table 1 gives the detailed statistics and the time comparison between our handle-independent reduced model and Au et al.'s handle-aware method (Au et al., 2007) (with t_{h2} as the system rebuild time of Au et al., 2007), measured on a 1.7 GHz i5 core2 PC with 6 GB of RAM.

Recall that the updating of our deformation system involves the background pre-computation and integration of the handle constraints. As shown in Table 1, the handle update time of our method is almost constant, independent of the number of handles n_h. In addition, the background pre-computation only takes a few seconds for mesh models of moderate sizes, which can be performed completely in the background in most cases before a user finishes specifying a new handle set. Note that the handle update time t_{h2} of Au et al.'s method is equal to its pre-computation time since once the handle

Table 1
Performance comparison between HIRM and Au et al.'s handle-aware method (Au et al., 2007). n_v and n_h are the number of vertices of the model and the number of handles used, respectively, n_{field} is the number of selected eigenvectors as propagation fields, and t_e and t_b are the eigenvector pre-computation time and background pre-computation time of our method, respectively. The subscripts 1 and 2 correspond to the our method and Au et al.'s method respectively. n_{t1} and n_{t2} are the number of triangles passed through by the sampled isolines and involved in the iterative Laplacian coordinates updating; t_{h1} and t_{h2} are the system update time for a given new handle set; t_{i1} and t_{i2} are the per-iteration updating time. P refers to the percentage of handle updating time of our method and that of Au et al.'s algorithm. Display time is excluded for fair comparison.

Model	n_v	n_h	n_{field}	n_{t1}	n_{t2}	t_e (s)	t_b (s)	t_{h1} (ms)	t_{h2} (ms)	P (%)	t_{i1} (ms)	t_{i2} (ms)
Dinosaur	14 000	2	6	1306	458	**1.701**	**1.947**	**73.79**	1350	**5.466**	**11.09**	1.922
Dinosaur	14 000	6	6	1306	1281	**1.707**	**1.949**	**81.34**	2984	**2.726**	**10.41**	10.07
Dinosaur	14 000	10	6	1306	2095	**1.702**	**1.952**	**86.59**	4622	**1.873**	**10.96**	17.84
Dragon	50 002	2	8	1713	418	**5.627**	**8.069**	**108.5**	6023	**1.801**	**35.66**	6.448
Dragon	50 002	6	8	1713	1270	**5.612**	**8.031**	**118.3**	12 800	**0.9242**	**35.83**	30.52
Dragon	50 002	10	8	1713	2133	**5.621**	**8.003**	**123.5**	19 910	**0.6203**	**36.35**	51.95
Pegasus	32 046	2	8	1766	448	**3.361**	**5.219**	**121.8**	3970	**3.068**	**27.39**	3.788
Pegasus	32 046	6	8	1766	1264	**3.373**	**5.241**	**143.3**	6725	**2.131**	**26.26**	17.84
Pegasus	32 046	10	8	1766	2086	**3.290**	**5.303**	**154.4**	11010	**1.402**	**26.35**	30.58
Max Planck	49 132	11	14	2979	2208	**5.787**	**19.71**	**878.5**	24 490	**3.587**	**81.67**	44.73
Dog	18 114	7	10	2159	1491	**1.830**	**3.991**	**168.4**	4165	**4.043**	**22.45**	11.93
Armadillo	60 000	10	9	2441	2105	**6.784**	**12.78**	**191.1**	24 620	**0.7762**	**48.89**	55.97
Hand	7609	7	11	2441	1497	**0.8848**	**2.778**	**204.9**	2129	**9.624**	**22.44**	9.376
Bar	6084	2	5	1108	418	**0.9497**	**0.5982**	**75.41**	812.9	**9.277**	**9.148**	1.340
Torus	6846	4	6	1336	818	**1.103**	**1.026**	**124.0**	1613	**7.688**	**12.32**	7.054
Cactus	5261	4	6	1328	847	**0.5219**	**0.6436**	**93.82**	929.9	**10.09**	**9.269**	3.569
Cylinder	4802	2	3	608	570	**0.6742**	**0.2405**	**65.65**	366.3	**17.92**	**2.866**	1.054

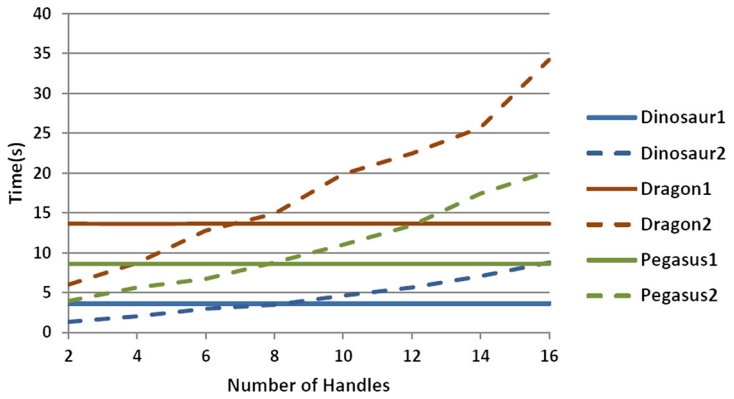

Fig. 7. Comparison of pre-computation time between HIRM and Au et al.'s method (Au et al., 2007) for different number of handles. The subscripts 1 and 2 refer to our method and Au et al.'s method, respectively. Note that the pre-computational cost of our method is independent of the number of handles.

set is modified, their method has to compute a new set of handle-aware harmonic fields. Such an expensive re-computation is what we aim to avoid for interactive incremental mesh editing.

The per-iteration updating cost t_{i1} of our deformation framework largely depends on the number of deformation descriptors n_e (i.e. the total number of isolines) and the number of triangles n_t crossed by the sampled isolines. These quantities are small compared to the mesh size, and is again independent of the number of handles n_h being used. Therefore, our method achieves a fast and constant iterative updating cost.

The plot in Fig. 7 compares the pre-computation time between HIRM and Au et al.'s method in terms of the number of handles for three specific models (Dragon, Dinosaur and Pegasus). Note that the pre-computational cost of our method $(t_e + t_b)$ is independent of the number of handles, while Au et al.'s method exhibits linear growth, making deformation inefficient when a user specifies a large number of handles. In other words, our method encourages users to freely specify any number of handles for a more flexible control. Recall that the propagation fields can be precomputed and stored in advance when a new model is first loaded, and the fields can be reused for any handle set configurations. This property allows instant editing of any pre-processed meshes, and greatly improves the efficiency when a user needs to frequently edit the same mesh model.

Fig. 8 shows the computational costs of using different numbers of propagation fields on several different meshes. It can be observed that the computational time increases sharply with more propagation fields. Obviously, selecting a relatively large number of propagation fields is infeasible for interactive editing. As shown in Table 1 and Fig. 5, our automatic selection algorithm keeps a reasonable short pre-computation time thus successfully speeds up the interactive process for real-time deformations, and simultaneously well guarantees the deformation quality by avoiding both undesired artifacts and rigid deformations.

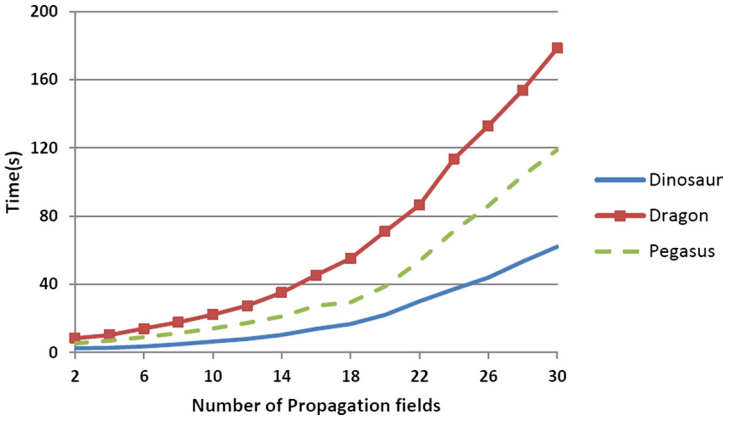

Fig. 8. Comparing pre-computation time for different numbers of propagation fields on various meshes.

5.3. Implementation and limitations

Our system uses the ARPACK solver to compute the first few eigenvectors without requiring a full eigen-decomposition, and we solve Eq. (20) using Cholesky factorization for efficient iterative updating. Similar to Au et al. (2007), we adopt the discrete dual Laplace operator with cotangent weighting scheme for the iterative Laplacian deformation since it gives a better approximation for irregularly sampled surfaces and maintains the local parameterization during editing by preserving local angles and triangle shapes. After experimenting on different geometric models and editing scenarios, we found that using $\varphi = 3.5$ in the selection of propagation fields is sufficient in generating desirable deformation results for most situations. Lastly, for rotation angles larger than π, like other iterative Laplacian editing frameworks (Huang et al., 2006; Au et al., 2006), our system requires specifying extra in-between handles to achieve plausible deformations.

6. Conclusion

In this paper, we develop a handle-independent reduced model for interactive incremental mesh editing. The deformation space of the input shape is captured by the propagation fields and deformation descriptors, which are defined by only the mesh geometry and are independent of the handle configuration. The handle-independent property of our reduced model allows the updating of the deformation system to be done mostly in the background, therefore users can dynamically update the handle set in real time and edit different parts of the shape in any order, without a long waiting time for the reconstruction of the deformation system. Our proposed deformation framework achieves constant per-iterative updating time, independent of the number of handles and vertices, leading to interactive and scalable mesh editing. We believe our reduced model has great potential for other applications, such as volumetric shape deformation and skeleton-less animation.

Acknowledgements

We thank the reviewers for their constructive comments. This work was partially supported by grants from the Natural Science Foundation of China (Grant Nos. 61272218, 61321491), Research Grants Council of HKSAR, China (Project No. 619611) and CityU Strategic Research Grant (Project No. 7004155).

Appendix A. Supplementary material

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.cagd.2015.03.001.

References

Au, O.K.-C., Tai, C.-L., Liu, L., Fu, H., 2006. Dual Laplacian editing for meshes. IEEE Trans. Vis. Comput. Graph. 12 (3), 386–395.
Au, O.K.-C., Fu, H., Tai, C.-L., Cohen-Or, D., 2007. Handle-aware isolines for scalable shape editing. ACM Trans. Graph. 26 (3), 83.
Botsch, M., Kobbelt, L., 2004. An intuitive framework for real-time freeform modeling. ACM Trans. Graph. 23 (3), 630–634.
Botsch, M., Sorkine, O., 2008. On linear variational surface deformation methods. IEEE Trans. Vis. Comput. Graph. 14 (1), 213–230.
Botsch, M., Pauly, M., Gross, M.H., Kobbelt, L., 2006. PriMo: coupled prisms for intuitive surface modeling. In: Symposium on Geometry Processing, pp. 11–20.
Chao, I., Pinkall, U., Sanan, P., Schröder, P., 2010. A simple geometric model for elastic deformations. ACM Trans. Graph. 29 (4).
Davis, T.A., Hager, W.W., 2009. Dynamic supernodes in sparse Cholesky update/downdate and triangular solves. ACM Trans. Math. Softw. 35 (4).
Dey, T.K., Ranjan, P., Wang, Y., 2012. Eigen deformation of 3d models. Vis. Comput. 28 (6–8), 585–595.
Fisher, M., Schröder, P., Desbrun, M., Hoppe, H., 2007. Design of tangent vector fields. ACM Trans. Graph. 26 (3), 56.
Hildebrandt, K., Schulz, C., von Tycowicz, C., Polthier, K., 2011. Interactive surface modeling using modal analysis. ACM Trans. Graph. 30 (5), 119.
Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.-Y., Teng, S.-H., Bao, H., Guo, B., Shum, H.-Y., 2006. Subspace gradient domain mesh deformation. ACM Trans. Graph. 25 (3), 1126–1134.

Karni, Z., Gotsman, C., 2000. Spectral compression of mesh geometry. In: SIGGRAPH, pp. 279–286.
Popa, T., Julius, D., Sheffer, A., 2006. Material-aware mesh deformations. In: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006. SMI06. IEEE Computer Society, Washington, DC, USA, p. 22.
Rong, G., Cao, Y., Guo, X., 2008a. Spectral mesh deformation. Vis. Comput. 24 (7–9), 787–796.
Rong, G., Cao, Y., Guo, X., 2008b. Spectral surface deformation with dual mesh. In: Proceedings of International Conference on Computer Animation and Social Agents, pp. 17–24.
Rustamov, R.M., 2009. On mesh editing, manifold learning, and diffusion wavelets. In: Mathematics of Surfaces XIII, 13th IMA International Conference, Proceedings. York, UK, September 7–9, 2009, pp. 307–321.
Sorkine, O., 2006. Differential representations for mesh processing. Comput. Graph. Forum 25 (4), 789–807.
Sorkine, O., Alexa, M., 2007. As-rigid-as-possible surface modeling. In: Symposium on Geometry Processing, pp. 109–116.
Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.-P., 2004. Laplacian surface editing. In: Symposium on Geometry Processing, pp. 175–184.
von Tycowicz, C., Schulz, C., Seidel, H., Hildebrandt, K., 2013. An efficient construction of reduced deformable objects. ACM Trans. Graph. 32 (6), 213.
Xu, K., Zhang, H., Cohen-Or, D., Xiong, Y., 2009. Dynamic harmonic fields for surface processing. Comput. Graph. 33 (3), 391–398.
Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.-Y., 2004. Mesh editing with Poisson-based gradient field manipulation. ACM Trans. Graph. 23 (3), 644–651.
Zayer, R., Rössl, C., Karni, Z., Seidel, H.-P., 2005. Harmonic guidance for surface deformation. Comput. Graph. Forum 24 (3), 601–609.