COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds (2013)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1562

RESEARCH ARTICLE

Streaming 3D deforming surfaces with dynamic resolution control

Lin Zhang[1,2], Fei Dou[1,2], Zhong Zhou[1,2]* and Wei Wu[1,2]

[1] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
[2] School of Computer Science and Engineering, Beihang University, Beijing, China

ABSTRACT

Real-time streaming of shape deformations in a shared distributed virtual environment is a challenging task due to the difficulty of transmitting large amounts of 3D animation data to multiple receiving parties at a high frame rate. In this paper, we present a framework for streaming 3D shape deformations, which allows shapes with multi-resolutions to share the same deformations simultaneously in real time. The geometry and motion of deforming *mesh* or *point-sampled* surfaces are compactly encoded, transmitted, and reconstructed using the spectra of the manifold harmonics. A receiver-based multi-resolution surface reconstruction approach is introduced, which allows deforming shapes to switch smoothly between continuous multi-resolutions. On the basis of this dynamic reconstruction scheme, a frame rate control algorithm is further proposed to achieve rendering at interactive rates. We also demonstrate an efficient interpolation-based strategy to reduce computing of deformation. The experiments conducted on both *mesh* and *point-sampled* surfaces show that our approach achieves efficient performance even if deformations of complex 3D surfaces are streamed.

KEYWORDS

streaming media; 3D shape deformation; distributed virtual environment; multi-resolution; point-sampled surface; progressive surface

***Correspondence**

Zhong Zhou, State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China.
E-mail: zz@vrlab.buaa.edu.cn

1. INTRODUCTION

With the development of surface deformation and manipulation techniques, as well as computer hardware, the amount of high-quality 3D shape animations is increasing at a relatively high rate. Techniques such as freeform deformation are powerful and flexible tools for interactive 3D shape editing. Some of these techniques, such as space deformation, can be used to deform shapes represented in various forms—polygon meshes or point clouds. Compared with traditional media content (e.g., video and audio) and static/rigid 3D model, sharing 3D shape animations in distributed virtual environment (DVE) can facilitate various exciting activities such as telemedicine, collaborative learning, and performing arts for a large number of remotely distributed participants.

Shape animations generated by 3D shape deformation tools can be simply divided into two kinds, offline and real time. Offline shape animation constructs a list of fixed animated shapes without responding to users, meaning that the animator has exact control of the animation. Shape animation specified while running is called real-time shape animation, in which users are allowed to interact with the deformable objects. Streaming and rendering of both real-time and offline animations in a shared DVE are challenging because of bandwidth limitations for the transmission of 3D motion data between multiple recipients at high frame rate. For offline shape animation, it is possible to download the preprocessed animation frames for offline viewing. However, because of the increase of the model resolution, the processing and rendering require massive storage space, and additional synchronization strategy is required to achieve frame synchronization among all distributed nodes.

A straightforward method of sharing 3D deformations is streaming the deformed shapes directly, namely model-based approach, which leads to a heavy transmission load. Another way is streaming control handles directly, meaning that each node needs to have very high computing capability to compute not only local but also all remote 3D deformations. The image-based technique is an efficient way to share 3D dynamics remotely, which has been fully studied in remote rendering. The main disadvantage of the image-based approach is the high interaction delay,

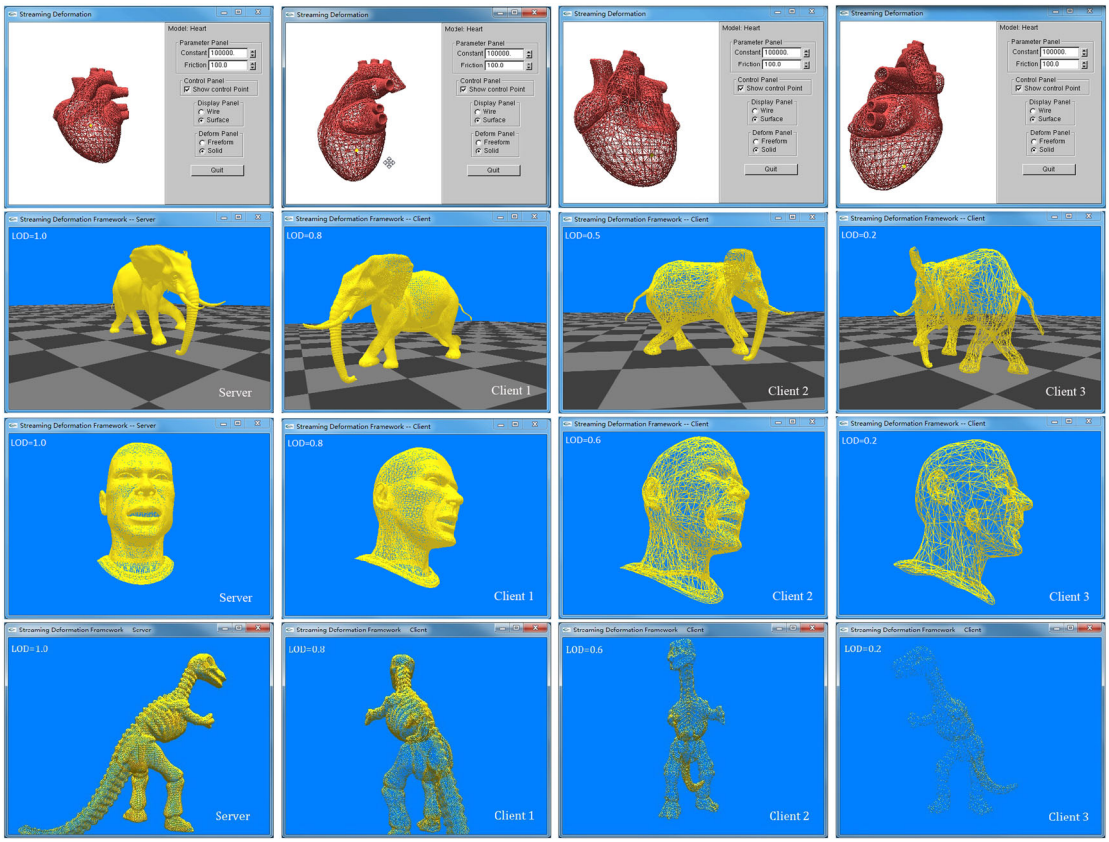


Figure 1. Streaming real-time interactive deformation (heart and dinosaur) and offline animations (elephant and head). The dinosaur model is used as a point-sampled surface. Clients run models with different resolutions but sharing the same deformations over network.

which has great impacts on the user experience of interactive applications. The image-based approach still requires substantial bandwidth to ensure sufficient quality of the geometry.

The purpose of our work is to provide an effective solution for streaming 3D shape deformations in DVE systems. We propose for the streaming scheme to be independent of deformation algorithms so that different shape-editing tools can be applied for future enhancements. Furthermore, we aim at the deformable shapes to be able to switch between multi-resolutions during run time so that each node can dynamically optimize its model resolution to achieve interactivity. Finally, we aim to retain a high frame rate even on network clients with very low rendering capabilities.

Our solution is a multi-resolution streaming framework based on a decomposing-reconstruction approach, which combines both spectral and progressive surfaces. The advantages of our method include the following: (i) arbitrary surface deformation tools are supported, and the bandwidth consumption is very low; and (ii) the method has the ability to continuously evaluate and optimize the resolution of mesh according to a tradeoff between visual quality and performance, achieving the rendering at interactive rates (Figure 1 and accompanying video). Our framework can be readily used in most DVEs.

2. RELATED WORK

3D model streaming has long been an active research topic and has been used in many applications. In general, the main goal of model streaming is to provide 3D contents in real time for users over network, such that the interactivity and visual qualities of the contents may match as closely as when they were stored locally.

To reduce the data size of 3D model for processing or network transmission, a fair amount of research on surface simplification techniques have been developed. Progressive mesh, first proposed by Hoppe [1], is a multi-resolution technique to progressively stream and render 3D static meshes. The progressive mesh approach simplifies models by iteratively contracting edges, namely edge collapse. Among numerous edge collapse algorithms, the quadric error metric (QEM) [2], which estimates the error introduced by a pair collapse as the distance from a vertex to a quadratic surface, represented as a symmetric

matrix, is the one capable of rapidly producing high fidelity approximations of models. To improve the inadequacy of PM for deformable surfaces, Kircher and Garland [3] proposed a hierarchy adaptation method, which provides very good approximations for a deforming mesh over all frames of an animation. There are also surface simplification methods for point-sampled geometry, such as [4,5]. These algorithms create a simplified point cloud that is a true subset of the original point set, by ordering iterative point removal operations according to a surface error metric. Pauly *et al*. [6] adapted and generalized different heuristics that have been presented in the polygonal mesh setting to point-based surfaces. Iterative simplification is achieved by using an adaptation of the QEM presented for polygonal meshes in [2], which is fast, robust, and creates surfaces of high quality. However, these iterative simplification methods for static surfaces are inadequate for real-time streaming of mesh deformations.

To exploit the dynamic visual importance induced by camera movement, view-dependent streaming strategies to effectively transmit meshes via network have been proposed in the literature [7–13]. They suggest to transmit the particular part of the mesh that is visible for clients. These view-dependent methods organize a given mesh into a hierarchy and continuously query the hierarchy to generate a set containing only visible primitives. They work well when dealing with mesh data without entire coding. This scheme is also fit for implementation with out-of-core algorithms.

The data size of the 3D model for network transmission can also be reduced by compressing. Compressed PM method [14] and progressive forest split method [15] groups edge collapses into batches to achieve a higher compression ratio. He *et al*. [16,17] introduced the spectral geometry images, an image format for the representation of 3D progressive model. There have also been publications describing spectral compression of static and dynamic meshes. For spectral compression of static meshes, Karni *et al*. [18] proposed to apply manifold harmonics (MH) to 3D mesh data to obtain compact representations. To reduce complexity, the mesh is partitioned into a number of balanced submeshes, each of which is compressed independently. Liu *et al*. [19] introduced a spectral shape compression method for triangular mesh based on the spectral analysis tool called Dirichlet MH. User-specified feature lines can be preserved while achieving a high compression ratio.

Techniques closest to our approach are those for spectral compression of dynamic mesh deformations. Alexa and Müller [20] introduced an approach for representing animation sequences based on the principal components of key-frame geometries. Karni and Gotsman [21] proposed a compression scheme for 3D animation sequences that applies linear prediction coding to the principle component analysis scheme, which achieves good compression ratios while keeping very small distortion. Rong *et al*. [22,23] proposed a spectral mesh deformation that compactly encodes the deformation functions in the frequency domain. By using spectral method, the size of the linear system is reduced significantly. Tang *et al*. [24,25] proposed to compress and stream the model and its motion using the spectra of the MH. Low resolution mesh can be used on the client by building a mapping matrix between a pair of the high and low resolution meshes.

Our primary contribution is an approach of streaming 3D deforming shapes with dynamic resolution control. The key idea is combining both the spectral and progressive representations of 3D shapes. Our approach is inspired by earlier spectral compression approaches of dynamic meshes. Tang *et al*. [24] have described an approach similar to ours. But it should be noted that our method supports dynamic resolution control, meaning that deforming shapes can switch smoothly between continuous multiresolutions to achieve frame rate optimization. Moreover, both surfaces represented in polygon meshes and point clouds are supported in our methods. Our framework has been designed so that the shape-editing and streaming modules are almost completely self-contained so that future enhancements can be easily adopted.

3. SURFACE SPECTRA

In this section, we briefly review the framework of the MHT and related concepts. More details can be found in [26,27].

Manifold harmonic is a spectral tool converting a function from spatial domain into frequency domain. MH bases (MHB) are defined as the eigenfunctions of the Laplace operator Δ. To ensure that MHB are orthogonal to each other, a symmetrizable discrete Laplace–Beltrami Operator (LBO) is usually built over the surface. Existing methods for discretizing LBO can be simply divided into two main groups, mesh-based and point-based methods.

Mesh-based: Let $\mathcal{M} = (V, E)$ be a given manifold triangular mesh. V denotes the set of vertices, and E denotes the set of edges. Let $\mathbf{e}_{ij}$ be the edge linking two distinct vertices $\mathbf{v}_i$ and $\mathbf{v}_j$. Then, the mesh-based symmetric LBO of mesh $\mathcal{M}$ is defined as

$$\begin{cases} \Delta_{ij} = \frac{\cot\alpha + \cot\beta}{\sqrt{A_i A_j}} & \text{if } \{\mathbf{v}_i, \mathbf{v}_j\} \in E \\ \Delta_{ij} = 0 & \text{if } \{\mathbf{v}_i, \mathbf{v}_j\} \notin E \\ \Delta_{ii} = -\sum_j \Delta_{ij} \end{cases} \tag{1}$$

where A_i and A_j are the areas of the two triangles that share the edge $\mathbf{e}_{ij}$ and α and β are the two angles opposite to that edge. The eigenvalues and eigenfunctions of the LBO are all the pairs (λ_k, H^k) that satisfy

$$\Delta H^k = \lambda_k H^k. \tag{2}$$

Because Δ is a symmetric matrix, its eigenvalues are real and eigenfunctions are orthonormal. These eigenfunctions are the basis functions, namely the MHB, and any scalar function defined on $\mathcal{M}$ can be projected onto them.

Using MHB, we can define the MH transform (MHT) to convert the geometry of mesh surface $\mathcal{M}$ into the spectral domain, and *vice versa*. The geometry x (respectively y and 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 n denotes the number of vertices and x_i denotes the x coordinate of $\mathbf{v}_i$. The projection of x onto MHB is the functional inner product

$$\tilde{x}_k = \langle x, H^k \rangle = \sum_i x_i H_i^k, k = 1, \ldots, m, \qquad (3)$$

where $\tilde{x}_k$ is the coefficient of the frequency basis H^k and m ($m \ll n$) is the user-specified number of MH used for transformation. Similarly, given the m $\tilde{x}_k$ and H^k in Equation 3, we can transform the object back into geometric space by reconstructing x, namely the inverse MHT.

In current context, we sort the eigenvalues in ascending order and use the first m eigenfunctions to capture the general shape of the model, and the next ones correspond to the details. As a result, the surface $\mathcal{M}$ is split into a smooth low-frequency base model $\mathcal{B}$ and high-frequency geometric details $\mathcal{D} = \mathcal{M} \ominus \mathcal{B}$. High-frequency details $\mathcal{B}$ should be added back to the reconstructed base model such that the original detailed model can be deformed consistently. Refer to [26] for more details.

Point-based: Besides the mesh-based LBO, there have also been publications describing specific methods for discretizing the LBO directly from point sets. Belkin *et al*. [28] proposed a converging discrete LBO for point clouds, but the resulting matrix is not symmetrizable in general. To tackle the lack of symmetry, a discrete LBO over the point-sampled surface that is guaranteed to be symmetrizable has recently been proposed by Liu *et al*. [27]. By solving the eigen problem related to this symmetrizable discrete LBO, a set of orthogonal point-based MHB (PB-MHB) for spectral analysis over point-sampled manifold surfaces is built. Using PB-MHB, similarly to mesh-based MHB, we can define the point-based MHT (PB-MHT) to convert the geometry of point-sampled surface into the spectral domain, and vice versa.

4. FRAMEWORK OVERVIEW

Our framework consists of several major function modules (Figure 3), the deformation module and spectral transformer on the sender and the level of detail (LOD) resolver and geometric reconstructor on the receiver. Each module has been designed so that it is almost completely self-contained and future enhancements can be easily adopted.

Our solution is mainly a two-stage process, a decomposing approach on the sender and a reconstruction approach on the receiver. Firstly, the original large model $\mathcal{M}_L = (V_L, E_L)$ is submitted to a deformation module on the sender side to perform a deformation reconstruction $\mathcal{M}_L^* = (V_L^*, E_L^*)$. The resulting model $\mathcal{M}_L^*$ can then be decomposed into frequency domain by an MHT transform. Then, the transformed spectral coefficients are streamed to the receivers instead of the whole deformed model. On the receiver, an optimized version of simplified model $\mathcal{M}_S = (V_S, E_S)$ is created by a geometric controller according to its LOD, determined by an adaptive LOD resolver. When the transformed spectral coefficients are received from the sender, a receiver-based multi-resolution mesh reconstruction approach described in the succeeding text is implemented on V_S. The resulting vertex geometry V_S^* together with the set of edges E_S leads to a frame of transformation of the simplified model $\mathcal{M}_S^*$. It is worth noting that the set of edges E of a point-based surface is an empty set.

5. MULTI-RESOLUTION SURFACE RECONSTRUCTION

We consider the deformation over multi-resolution models because of the different computational power and display capability of users in practical distributed systems. There are two straightforward methods to achieve this goal, but we argue that neither of them is good enough for practical application. In the first method, deformations of all required resolutions are computed on the sender, and the deformed meshes are then transmitted to remote receivers. Nevertheless, this method leads to a very high computational overhead on the sender side. In the second method, a mapping matrix between the high and low resolution surfaces is built, and then the deformed low resolution mesh can be reconstructed on the receiver. However, mapping matrices are needed to be computed offline; as such, users cannot change the model resolution accordingly during the running time.

To address these challenges, we develop a receiver-based multi-resolution shape reconstruction approach that allows multi-resolution models to share the same deformation and allow deformable surfaces to switch smoothly between continuous multi-resolutions in real time. The most straightforward approach to the multi-resolution solution is to take the advantage of progressive shape representations, such as [1,2,4–6], to obtain continuous-resolution representations of the original large shape. Although the progressive shape representation techniques work well for static models, they may not work for dynamic models because of their inabilities to define the new geometry of vertices of model when the model is deformed. Nevertheless, we can still take advantage of the sequence of collapses of point pairs (i.e., the renumbering information of vertices and faces) from the progressive shape representation.

The key of progressive surfaces is how to define the order of collapses and where a new point should be located. In fact, the new point is usually located on the one position on the collapse edge defined by the user. According to the choice of the new vertex, edge collapses in PM

can be divided into two categories: full-edge and half-edge collapse. The full-edge collapse requires additional calculation to optimize the position of new vertex, resulting in high computation complexity. The half-edge collapse, on the contrary, can achieve similar results as other vertex placement schemes with no new points generated, contributing to saving storage, reducing bandwidth, and accelerating compression. Similarly, we can define full and half point-pair contractions on point-sampled surfaces, which are extensions of the common edge collapse operator. The full point-pair contraction replaces two points P_1 and P_2 by a new point P'. The half point-pair contraction uses simple point removal, that is, points are removed from the point cloud in order, resulting in a simplified point cloud that is a subset of the original point set.

As previously discussed, the geometry of the full-detail dynamic surface is compressed on the sender side and then reconstructed on the receiver side by performing transformation between geometry and spectral domain, meaning that the reconstructed point cloud on the receiver side is a subset of the original point set. Therefore, we use half contraction of a surface, which creates a simplified mesh or point cloud that is a true subset of the original surface. The half contraction data structure that stores the renumbering information of vertices and faces is defined in the succeeding text.

vertices and faces is defined as below.

```
struct HalfContraction {
    int vFrom;
    int vTo;
    array<int> trisRemoved;
    array<int> trisAffected;
}
```

where *vFrom* is the vertex removed by this contraction and it is renumbered to vertex *vTo*. The array *trisRemoved* stores indices of triangular faces vanish in this contraction. The Array *trisAffected* stores indices of triangular faces with their vertex *vFrom* renumbered to *vTo*. For point-sampled surfaces, sets *trisRemoved* and *trisAffected* are empty sets.

Implementing the progressive surfaces proceeds, a sequence of contraction transformations are acquired as a list of the *HalfContraction* records. The given contraction list is then processed sequentially, and an optimized version of mesh $\mathcal{M}_S = (V_S, E_S)$ is produced according to a tradeoff between visual quality and performance. However, here, $\mathcal{M}_S$ only provides the renumbering information of vertices and faces, with no up-to-date geometry of vertices on the deformed model. Therefore, we further reconstruct the geometry of $\mathcal{M}_S$ by adopting the MH spectra of the model, because using this scheme, we only need to know the new transformed m spectral coefficients and the *a priori* MHB. Therefore, for each vertex $\mathbf{v}_i \in V_S$, we reconstruct its updated geometry x_i^* (respectively, y^* and z^*) from the frequency domain as the following formula.

$$x_i^* = \sum_{k=1}^{m} \tilde{x}_k^* H_i^k, \tag{4}$$

where $\tilde{x}_k^*$ is the transformed spectral coefficient of the frequency basis H^k, which is streamed from the remote sender. After the reconstruction in the frequency domain, we get the transformed geometry of vertices V_S^* and then the deformed base surface $\mathcal{B}_S^* = (V_S^*, E_S)$. Finally, the modified fine-scale surface is reconstructed from $\mathcal{B}_S^*$ and the corresponding high-frequency geometric details as $\mathcal{M}_S^* = \mathcal{B}_S^* \oplus \mathcal{D}_S$.

The advantage of such an approach is obvious. The progressive surfaces technique is extended to deformable model; therefore, dynamic models can switch smoothly between continuous multi-resolutions in real time according to a local optimization on the receiver. The deformation server needs only to compute and transmit the m ($m \ll n$) transformed spectral coefficients of the input large model $\mathcal{M}_L$, then all users in the system will experience the same state changes by viewing models with their own proper resolutions.

6. DETAIL-PRESERVING

After reconstructing the modified base model $\mathcal{B}_S^*$, the geometric details $\mathcal{D}$ should be added back such that the original detailed model can be deformed consistently. There are some different approaches to add the geometric details back. The simplest way is to record the differences between $\mathcal{M}$ and $\mathcal{B}$ before the deformation and directly add the differences back to the modified base model $\mathcal{B}^*$ during running. However, this naive approach cannot handle the local rotational effects [22].

To be able to rotate the geometric details, Rong *et al.* described an efficient method that builds a local coordinate system at every vertex [22]. Each local coordinate system is built from the normal of the vertex and the first edge of the first triangle incident to this vertex. After the deformation, they rebuild such a local coordinate system at every vertex of the modified base model $\mathcal{B}^*$, and the modified fine-scale surface is reconstructed by adding the difference vectors back to the deformed local coordinate systems. In this way, the local rotation effects will be automatically achieved. But this approach may also cause local self-intersections. Moreover, because some specific edge

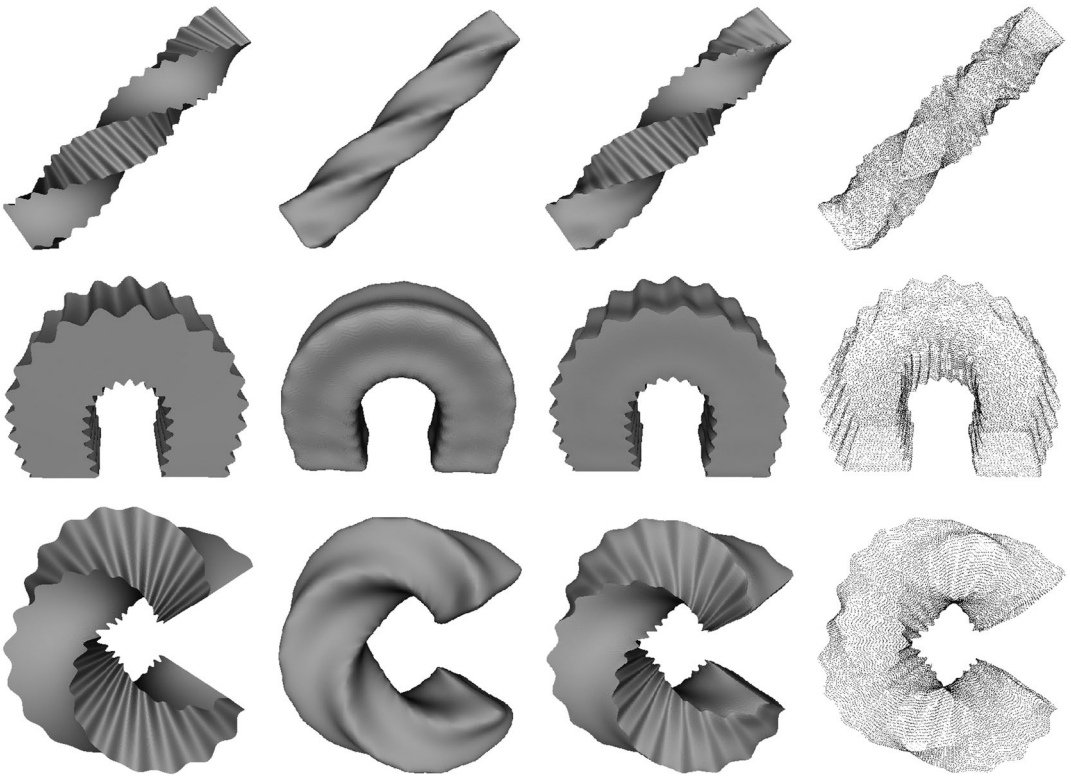


Figure 2. Deformed bars (left) and the corresponding modified base models (middle left). Result mesh models (middle right) and point-sampled surfaces (right) are reconstructed using normal displacements rotates details. Although there are slight distortions, the result models are still useful in applications where slight loss can be tolerated.

information are used to build local coordinate systems, this approach is not applicable to PM and point-sampled surfaces.

Botsch *et al*. [29] proposed to improve the detail reconstruction by using an adaptation of the *deformation transfer* technique. The idea here is to transfer the deformation gradient for each triangle of the base model $\mathcal{B}$ to the fine-scale surface $\mathcal{M}$, which yields the result $\mathcal{M}^*$. The method by Botsch *et al*. performs well on avoiding local self-intersections. However, it is only applicable to manifold triangle meshes but not to point-sampled surfaces. Moreover, because this method requires to compute deformation gradients for all triangles from $\mathcal{B}$ and $\mathcal{B}^*$, PM are not supported because only a simplified model $\mathcal{B}_S^*$ is reconstructed instead of $\mathcal{B}^*$.

Another efficient approach that can achieve local rotation effects is encoding the details as displacement vectors parallel to the normal field of $\mathcal{B}$, and the modified fine-scale surface is reconstructed from $\mathcal{B}^*$ and the original normal displacements [30]. Although the resulting modified surface may still have local self-intersections, this method is applicable to both PM and point-sampled surfaces, whereas the others are not. Therefore, to keep details for both mesh and point-sampled surfaces, we adopt the normal displacement in [30] to encode the geometric details $\mathcal{D}$ and add them back to $\mathcal{B}^*$, resulting in the result $\mathcal{M}^*$. In practice, the smaller the difference between $\mathcal{M}$ and $\mathcal{B}$, the better the result of detail-preserving.

Our lossy multi-resolution surface reconstruction is useful in applications where precise accuracy is not important, for example, rapid previewing and remote dance teaching [31]. Figure 2 gives examples where a bar is twisted, bended, or both. The corresponding geometric distortions of the result models in Figure 2 are shown in Table II.

7. FRAME RATE CONTROL

One of the most important indicators when viewing 3D animation is the frame rate. But in many cases, the complexity of 3D models exceeds the capability of graphics hardware to render them interactively. Dynamic approaches such as LOD and data filtering are usually used to alleviate this problem. However, dynamic 3D rendering of real-time 3D deformation in a distributed system is a challenging task, and we argue that part of the reason is the difficulty of streaming deformations and sharing deformations between multi-resolution models.

To make our framework viable, we require a frame control approach to be robust enough to deal with real-time 3D

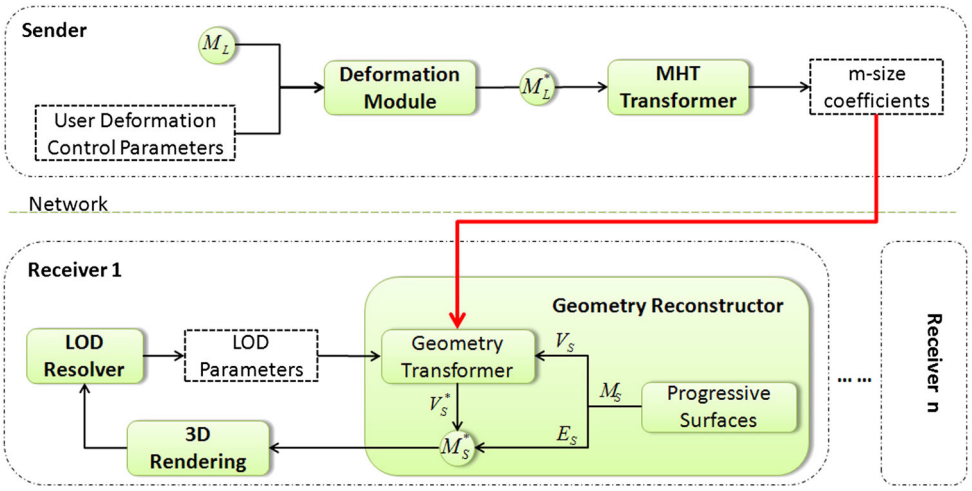


Figure 3. Overview of our proposed framework.

shape deformations as well as keeping a consistently high and smooth frame rate. Moreover, our approach should be scalable and modular.

Our frame rate control approach for 3D shape deformations is based on our receiver-based progressive mesh reconstruction algorithm previously described. It consists of two function modules, the *LOD resolver* and *geometric controller* shown in Figure 3.

LOD resolver: The LOD resolver controls switches between levels of detail. Importantly, switching between the different levels of details during simulations should be smooth and unnoticeable. LOD is selected according to mainly two criteria: the frame rate of the system and the distance from the camera.

Let $L(o_i)$ denote the LOD for the i-th object o_i and d_i denote the distance from the object o_i to the viewpoint (we use the barycenter of the object to represent the position of it). In this paper, the proportion of vertices reserved after mesh o_i' simplification is used to quantize $L(o_i)$. Therefore, we define the computation of $L(o_i)$ as the formula in the succeeding text.

$$L(o_i) = \mathcal{G}(d_i) + \mathcal{F}(f, d_i) \tag{5}$$

$$\text{subject to:} \quad L(o_i) \in [0,1] \qquad \mathcal{G}(d_i) \propto 1/d_i,$$

where the first term $\mathcal{G}(\cdot)$ is a basic location-based LOD function and the second term $\mathcal{F}(\cdot)$ is an optimal function that helps to keep the frame rate smooth and high. We, in this paper, adopt a straightforward linear function $\mathcal{G}(d_i) = 1 - d_i'$, where $d_i' \in [0,1]$ is the rescaled d_i. Clearly, the key of frame rate control in formula 5 is the selection of $\mathcal{F}(\cdot)$. We define $\mathcal{F}(\cdot)$ as a linear optimization formulation that satisfies formula 5 as in the succeeding text.

$$\mathcal{F}(d_i, f) = \begin{cases} \kappa \cdot d_i', & \kappa \geq 0 \\ \kappa \cdot (1 - d_i'), & \kappa < 0 \end{cases} \tag{6}$$

$$\text{subject to} \quad \kappa \in [-1, 1],$$

where κ is a dynamic penalty factor for frame rate control. Intuitively, there is a direct relationship between the penalty factor κ and frame rate f, and the optimization of κ is given by the following algorithm: where f_0 is the user-defined objective frame rate, $T_f > 0$ is the triggering threshold, and δ is the step size of κ. Algorithm 1 leads to a smooth movement of κ in the range from -1 to 1. During runtime, each $L(o_i)$ switches dynamically between the different levels of details smoothly and unnoticeably, resulting in a satisfactory switch between multi-resolution models (see the experimental results later in this paper).

Algorithm 1: adjust the penalty factor κ

Input: $f, f_0, T_f, \kappa, \delta$
Output: adjusted κ

```
Initialization: κ ← 0
for each interval do
    if f > f0 + Tf then
        κ ← (κ + δ) < 1 ? (κ + δ) : 1;
    end
    else if f < f0 − Tf then
        κ ← (κ − δ) > −1 ? (κ − δ) : −1;
    end
end
```

Geometric controller: The geometric controller encapsulates all the information necessary to describe the object's appearance within the current frame. It is composed of two submodules (Figure 3), *progressive surfaces module* and *geometry transformer module*,

which performs the multi-resolution shape reconstruction approach described in the previous chapter. To rate the contraction operation of progressive surfaces, we adopt the QEM [2] error function presented for polygonal meshes to build the contraction list of point pairs. For point-sampled surfaces, we use an adaptation of QEM to quantify the error caused by the contraction operation as described in [6]. The idea is to approximate the surface locally by a set of tangent planes and to estimate the geometric deviation of a mesh vertex from the surface by the sum of the squared distances to these planes. The only difference here from [6] is that we do not create new points during contraction of point pairs, which leads to a half contraction.

At run time, the *progressive surfaces* switches model from one LOD to another according to the dynamic $L(o)$ chosen by the LOD resolver, resulting in the set of vertex indices V_S and the set of edges E_S of the simplified model $\mathcal{M}_S$ within the current frame. Then, submodule *geometry transformer* reconstructs the geometry of each vertex in V_S according to the transformed spectral coefficients (i.e., $\tilde{x}^*$, $\tilde{y}^*$, and $\tilde{z}^*$) received from the server side, resulting in the transformed simplified model $\mathcal{M}_S^* = (V_S^*, E_S^*)$.

8. INTERPOLATION-BASED OPTIMIZATION

Considering that the computation of 3D shape deformation requires intensive computation overhead and lengthy computing time, we advocate the use of shape interpolation for further improving the system efficiency.

The powerful properties of Laplacian coordinates for mesh representation have been exploited in various ways. Lipman *et al*. [32] show that the Laplacians are linear functions of the Cartesian coordinates and thus their linear interpolation is equivalent to simple Cartesian interpolation. Sorkine [33] shows that by interpolating the differential representations of several meshes, it is possible to blend between different shapes. When simple linear interpolation of the differential representations is performed, the result is identical to linear interpolation of Cartesian coordinates, which is known for its artifacts. Alexa *et al*. [34] show that Laplacian coordinates can be effective for morphing and proposed to use varying interpolation weights, which lead to better results. For more detailed surveys in this field, please refer to [32,33].

Therefore, we mimic motions of deformable model by interpolating spectral representations of meshes, because the spectral method is an inherent part of our approach. In current context, interpolation of the spectral coefficients $\tilde{x}$, $\tilde{y}$, and $\tilde{z}$ is implemented. Importantly, because we are working on real-time streaming of 3D shape deformations, an essential assumption is that there is no large deformation between two continuous poses. Hence, to interpolate the geometry x (respectively, y and z), the simplest linear interpolation scheme is adopted using the following formula:

$$I(\tilde{x}^i, \tilde{x}^j) = \tilde{x}^i + \lambda(\tilde{x}^j - \tilde{x}^i) \tag{7}$$

where $\tilde{x}^i$ and $\tilde{x}^j$ are spectral coefficients of the two input poses, $I(\tilde{x}^i, \tilde{x}^j)$ is the interpolation between them, and λ $(0 \leq \lambda \leq 1)$ is the step factor controlling the rate of approximation to $\tilde{x}^j$. When completed, the inverse MHT is implemented on $I(\tilde{x}^i, \tilde{x}^j)$ to get the geometry of this interpolation. Refer to [33] for more interpolation schemes for deformation propagation.

Figure 4 shows examples of the interpolation. As the examples confirm, spectral scheme can yield very intuitive interpolations and can hence be used for exploring the natural space shape spanned by a set of key poses. Therefore, high frame rate real-time deformations could be achieved

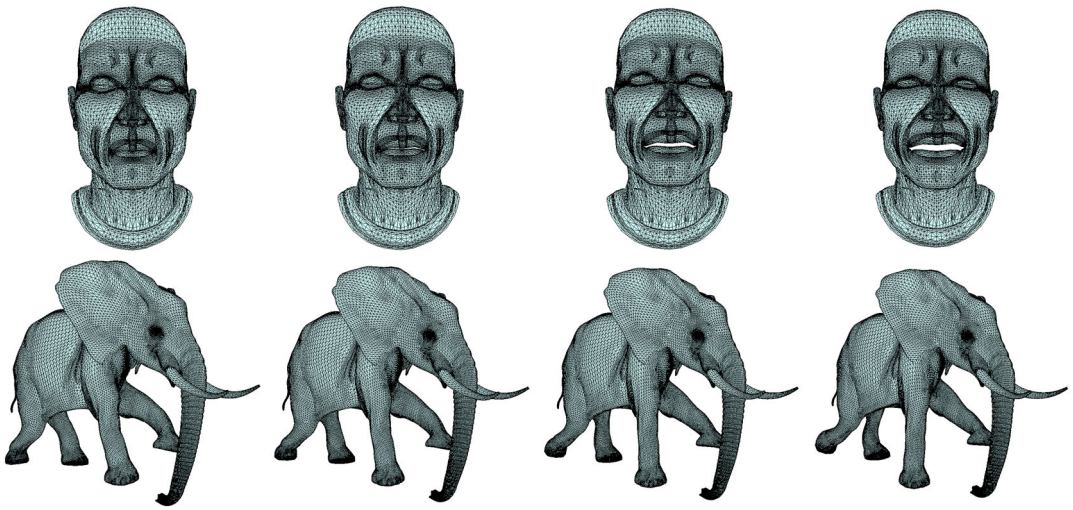


Figure 4. Interpolations between two input poses (leftmost and rightmost columns). Parameter values: λ = 0.33, 0.66.

by only computing the key poses. Consider the elephant example in Figure 4, an animation at a frame rate $3 \cdot f_n$ could be achieved by only computing f_n key poses per second during run time (see demo video). This deformation propagation allows for different interesting optimizations such as reducing the computational and transmission load. To achieve smooth animation, there should be no large deformations between two subsequent key poses.

9. EXPERIMENTS

Each node in DVE could be either a sender or receiver of shape deformation, or both. In our experiments, a node performed as both deformation sender and receiver. Our experiments were conducted on several Windows 7 Lenovo PCs with Intel Core2 Xeon 3.07GHz CPU and 4GB RAM, and an apple iPad2. We used 100 Mbps LAN between PC nodes, and the data are transmitted under the transmission control protocol/internet protocol. The iPad client connects with the PC server through 3G wireless with bandwidth less than 2 Mbps. We use the publicly available library ARPACK (used in [35]) to compute the solutions of large sparse eigenproblems and use Liu's implementation of the method proposed in [27] to compute PB-MHB. The number of MH m is set to 200 for an optimal tradeoff between visual quality and performance. Two kinds of mesh animations, real-time and offline animations, are tested in our experiments. For real-time interactive shape manipulation, we have implemented the *green coordinates* of [36], which is a method of space deformation, because of its simplicity and high speed. The offline animation data used in this paper was made available by Robert Sumner and Jovan Popović from the computer graphics group at Massachusetts Institute of Technology. For more details, please refer to [37].

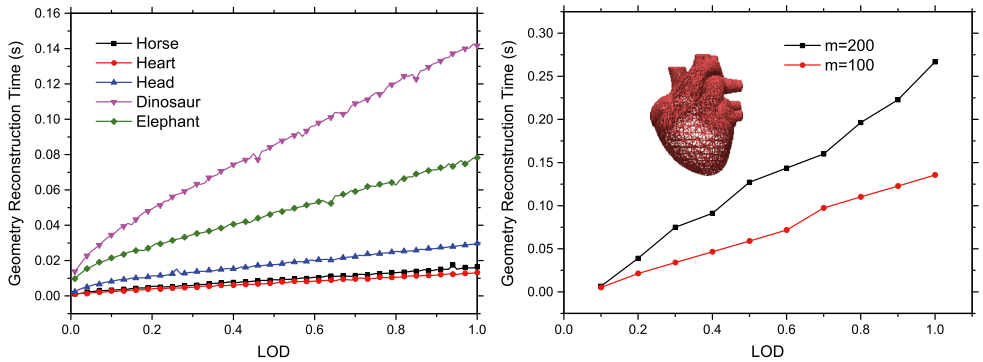
Table I. Model information and running time (in seconds) for computing mesh-based (t^m_{eigen}) and point-based (t^p_{eigen}) MHB and MHT transformation (t_{MHT}).

Model	# of vertices	# of faces	t^m_{eigen}	t^p_{eigen}	t_{MHT}
Heart	7,349	14,676	7.065	451	0.031
Horse	8,431	16,843	7.817	672	0.035
Head	15,941	31,620	14.759	1,534	0.065
Bar	32,802	65,600	31.763	3,148	0.142
Elephant	42,321	84,638	37.594	4,091	0.275
Dinosaur	56,194	112,384	51.137	5,065	0.415

We list in Table I the statistics of meshes throughout the paper and computing time on them. The timings in Table I report that the computing of MHs is the computationally most intense part of the preprocessing step but still reasonable even for large surfaces. In fact, this step can be precomputed offline as the prior knowledge of the model. Once the MHs are precomputed, the computational overhead of MHT is small and can be computed in real time by using a powerful centralized server if necessary. Figure 5 shows the computing time of the receiver-based shape reconstruction on both PC and iPad clients. As we can see, lowering the object's LOD can obviously reduce the cost of shape reconstruction.

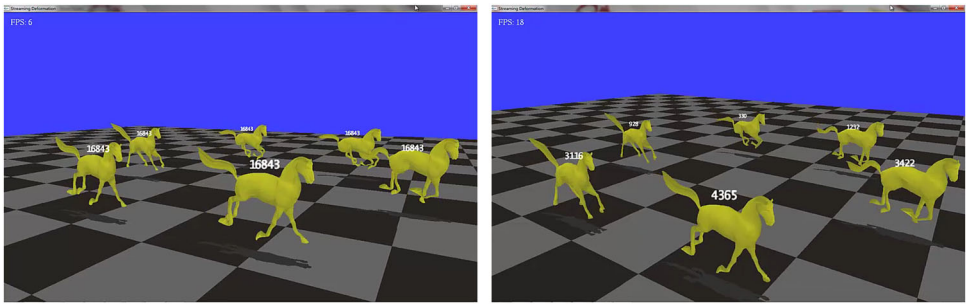
Figure 1 gives snapshots of sharing interactive deformations, as well as offline mesh animations, with multiresolution models. Remote users are viewing the same animation with models of different resolutions simultaneously in real time. Moreover, the sender only needs to compute and transmit deformations of the original high-resolution model, which could benefit both computation and network a lot.

In order to demonstrate the effectiveness and flexibility of the proposed frame rate control approach, as well as the system efficiency, a dynamic scenario with large movement of camera is tested. Figure 6 shows such a scenario, where six horses are galloping in the virtual environment. When streaming such animation via the network in our experimental setup, we achieved a low frame rate of only about 5 fps (Figure 6(a)) because of high computational overheads on these full-detail models, resulting in an unpleasant visual experience. Figure 6(b) shows a corresponding snapshot of the implementation of our approach with dynamic frame rate control. The models switch between low-resolutions and high-resolutions dynamically



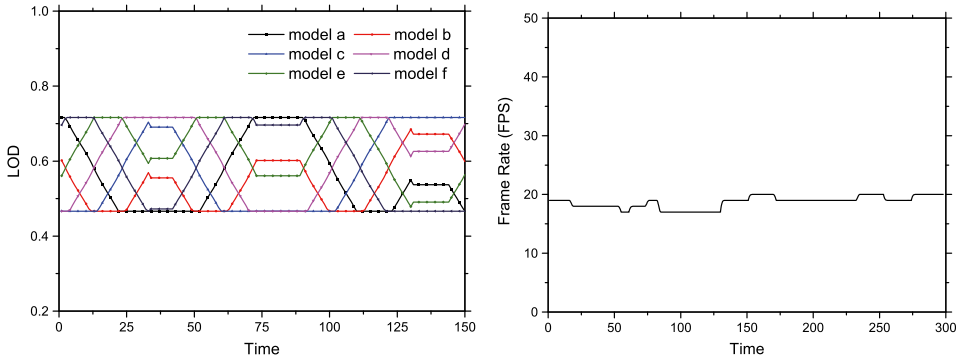
(a) pc client (b) iPad client

Figure 5. Computing time for geometry reconstruction at a different level of detail (LOD) on PC and iPad.



(a) full-detail rendering (b) view-dependent rendering

Figure 6. Receiver-based view-dependent rendering and its comparison with full-detail rendering. For parameter values of $f_0 = 20$, $T_f = 3$, and $\delta = 0.01$ in (b).



(a) plot of LODs of different models (b) plot of frame rate

Figure 7. Plots of level of detail (LOD) and frame rate of the scenario shown in Figure 6(b).

according to their levels of detail while the camera is rotating around them at a high speed. By reducing resolutions of distant models, overheads of rendering and reconstruction of deformed geometry are significantly reduced, therefore higher frame rate can be achieved. As shown in Figure 7, as well as the accompanying video, switching between multi-resolutions is very smooth even when the user moves the camera at a very high speed, resulting in very good visual effect. And, the frame rate keeps stable at around the desired f_0.

In our approach, the bandwidth usage of transmitting an animation with a set of objects O to a group of clients C can be estimated as

$$B = |C| * sizeof(double) * \sum_{i=1}^{|O|} m_i f_i \qquad (8)$$

where B is bandwidth usage, $|C|$ is the number of clients, and $|O|$ is the number of objects. Parameter m_i is the number of eigenvectors used for compression of the $i\text{-}th$ object, which is usually in the order of a few hundred [22,24,25], and parameter f_i is the update rate per second of the $i\text{-}th$ object. Figure 8 shows a comparison of bandwidth usage of streaming a virtual elephant animation between different streaming approaches. As displayed

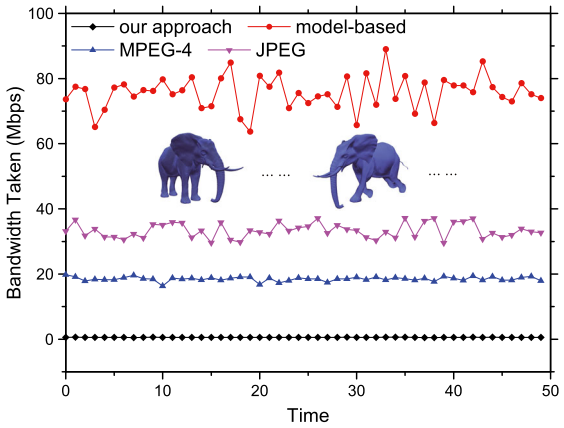


Figure 8. Comparison of bandwidth usage between our spectral-based, the model-based, and the remote-rendering approaches. For frame rate at around 30 fps and image resolution of 1440 × 900.

in Figure 8, in our experimental setup, bandwidth usage of the model-based approach is around 70 Mbps, whereas that of our approach keeps very low. The remote rendering approaches usually require substantial bandwidth to ensure

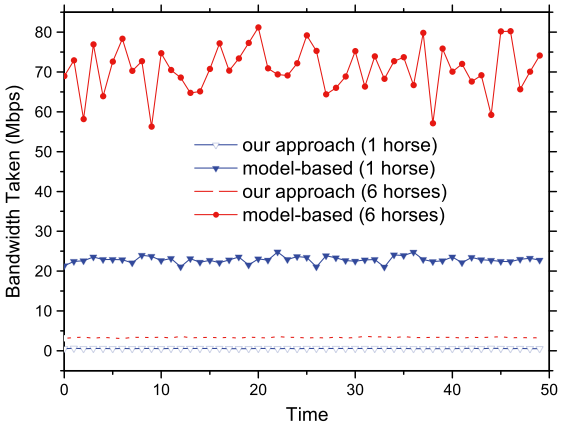


Figure 9. Comparison of bandwidth usage when different numbers of mesh animations (galloping horses) are streamed.

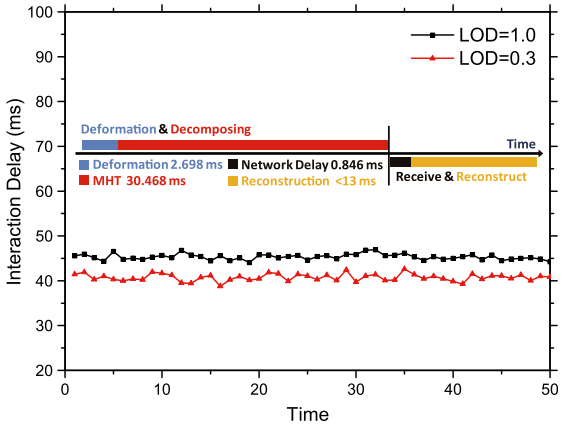


Figure 11. Interaction delay between the sender and the receivers.

Table II. The geometric distortion of the reconstructed bar models in Figure 2, based on Hausdorff distance.

Example	Hausdorff Distance	
	Mesh Surface	Point-sampled Surface
Twisted	0.0032	0.0029
Bended	0.0093	0.0101
Twisted and bended	0.0046	0.0054

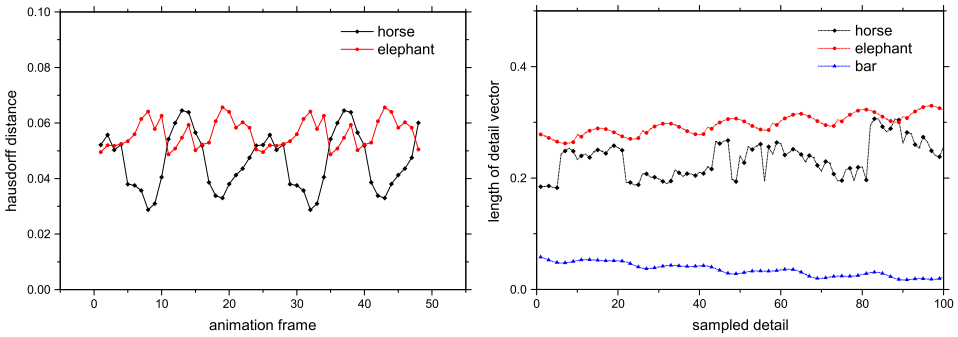
sufficient quality of the rendered image, and such two-dimensional solutions are not applicable in many DVEs. Figure 9 shows a comparison of bandwidth usage of the two three-dimensional solutions when different numbers of animations are streamed. We can see that our approach can support more deformable models than the straightforward model-based approach in a DVE due to its low bandwidth requirement.

In our experiments, the geometric distortions of the reconstructed bar models (Table II) are much smaller than that of the horse and elephant models (Figure 10(a). One of the main reasons here is the smaller details $\mathcal{D}$ of the bar (Figure 10(b)).

Figure 11 shows delay of streaming real-time manipulation of the heart model shown in Figure 1. As shown in the figure, although the local user gets a shape update in about 2.7 ms after the user input, remote users have to wait for about 40–45 ms to view the deformation under different model resolutions. Note that this waiting time can be much longer if it is in an intricate network such as the Internet. Therefore, the network delivery delay is the bottleneck.

There are two ways to implement shape interpolation in our framework, server-based and client-based. In our experiment, the server-based approach is implemented (Figure 12). Two interpolations were computed on the server by using Equation 7, resulting in a smoother move and a faster frame rate. As shown in Figure 12, the frame rate on the client is around three times as high as the server's (see also the demo video).

In fact, when streaming 3D animation, a pose estimation method could also be adopted. Figure 13 shows such an example. Given spectral coefficients $\tilde{x}^{i-1}$ and $\tilde{x}^{i}$ of the latest two pose updates, the pose estimation is computed by



(a) geometric distortion (b) size of normal displacements

Figure 10. The geometric distortion of the reconstructed animations on the receiver side, and plot of length of some sampled normal displacement vectors.

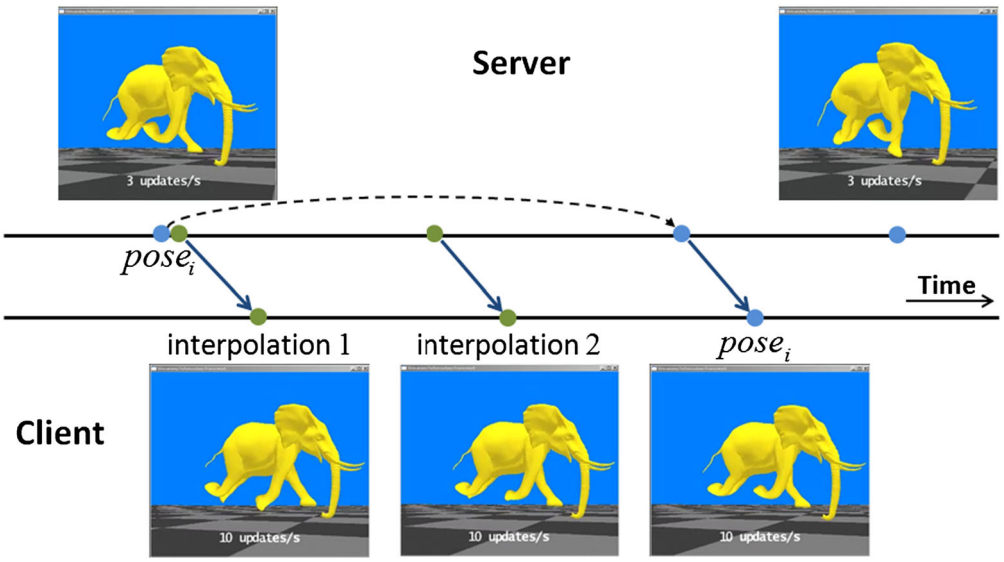
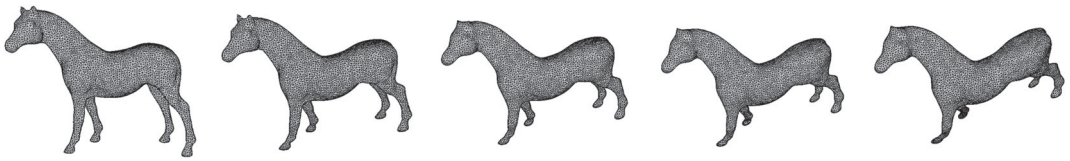


Figure 12. Improve the frame rate using spectral-based shape interpolation.



(a) pos_{i-1} (b) pos_i (c) Est_1 (d) Est_2 (e) Est_3

Figure 13. Illustration of pose estimation for parameter values $\lambda = 1, 2, 3$.

a simple generalization of formula 7 as in the succeeding text:

$$Est(\tilde{x}^{i-1}, \tilde{x}^i) = \tilde{x}^i + \lambda(\tilde{x}^i - \tilde{x}^{i-1}) \tag{9}$$

where $Est(\tilde{x}^{i-1}, \tilde{x}^i)$ is the spectral representation of the estimation and $\lambda \geq 0$ is the step factor. Figure 13 demonstrates that we can get smooth poses by the pose estimation.

10. CONCLUSION AND FUTURE WORK

We have proposed a novel approach for streaming 3D deforming surfaces with dynamic resolution control. Our framework allows the transmission of deformations at multiple resolutions in real time. Deformable meshes can switch smoothly and unnoticeably between continuous multi-resolutions in real time without affecting the streaming at all. A dynamic frame rate control approach is introduced, which makes a good tradeoff between visual quality and performance even when the camera moves at a very high speed. Furthermore, an interpolation-based optimization is introduced to further improve the computation and frame rate. Our framework is designed so that future enhancements on deformation tool and the progressive surfaces step can be easily adopted; therefore, it can be used for a wide range of applications such as telemedicine, collaborative design, and performing arts. One limitation of our results is the geometric distortion of the result model because of the lossy spectral compression and errors generated by the step of adding details.

Interactive space deformation is a powerful approach for editing geometric models and animated characters, which has led to an abundance of methods seeking to improve interactive deformation with intuitive use. Some efficient methods are known to compute high-quality surface deformations [38–40], and each has its own features. Currently, only a space deformation method [36] is adopted in our framework. A part of our future work is using several existing interactive deformation techniques to enhance the deformation tool for efficient and intuitive surface deformations.

ACKNOWLEDGEMENTS

Our thanks to Dr. Gregorij Kurillo for the recommendation and paper polishing. This work is supported by the Natural Science Foundation of China under Grant

No.61170188, the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No.20121102130004, and the National Key Technology R&D Program of China under Grant No.2012BAI06B01.

REFERENCES

1. Hoppe H. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96. ACM, New York, NY, USA, 1996; 99–108.
2. Garland M, Heckbert PS. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1997; 209–216.
3. Kircher S, Garland M. Progressive multiresolution meshes for deforming surfaces. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05. ACM, New York, NY, USA, 2005; 191–200.
4. Alexa M, Behr J, Cohen-Or D, Fleishman S, Levin D, Silva CT. Point set surfaces. In *Proceedings of the Conference on Visualization '01*, VIS '01. IEEE Computer Society, Washington, DC, USA, 2001; 21–28.
5. Linsen L. *Point Cloud Representation*. Univ., Fak. für Informatik, Bibliothek, Technical Report, Faculty of Computer Science, University of Karlsruhe, 2001.
6. Pauly M, Gross M, Kobbelt LP. Efficient simplification of point-sampled surfaces. In *Proceedings of the Conference on Visualization '02*, VIS '02. IEEE Computer Society, Washington, DC, USA, 2002; 163–170.
7. Guan W, Cai J, Zheng J, Chen CW. Segmentation-based view-dependent 3-D graphics model transmission. *IEEE Transactions on Multimedia* 2008; **10**(5): 724–734.
8. Alregib G, Altunbasak Y, Rossignac J. Error-resilient transmission of 3-D models. *ACM Transactions on Graphics* 2005; **24**(2): 182–208.
9. Yan Z, Kumar S, Kuo C. Error-resilient coding of 3-D graphic models via adaptive mesh segmentation. *IEEE Transactions on Circuits and Systems for Video Technology* 2001; **11**(7): 860–873.
10. Cheng W, Ooi WT, Mondet S, Grigoras R, Morin G. An analytical model for progressive mesh streaming, In *ACM MM'07*, Augsburg, Germany, 2007; 737–746.
11. Li H, Tang Z, Guo X, Prabhakaran B. Loss tolerance scheme for 3d progressive meshes streaming over networks, In *ACM MM'08*, Vancouver, BC, Canada, 2008; 501–504.
12. Jihad E, Chiang Y. External memory view-dependent simplification. *CGFComputer Graphics Forum* 2000; **19**(3): 139–150.
13. DeCoro C, Pajarola R. Xfastmesh: fast view-dependent meshing from external memory, In *In IEEE Vis'02*, Boston, Massachusetts, USA, 2002; 363–370.
14. Pajarola R, Rossignac J. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics* 2000; **6**(1): 79–93.
15. Taubin G, Guéziec A, Horn W, Lazarus F. Progressive forest split compression. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98. ACM, New York, NY, USA, 1998; 123–132.
16. He Y, Chew BS, Wang D, Hoi S, Chau LP. Streaming 3D meshes using spectral geometry images. In *Proceedings of the 17th ACM International Conference On Multimedia*, MM '09. ACM, New York, NY, USA, 2009; 431–440.
17. Chew BS, Chau LP, He LP, Wang D, Hoi S. Spectral geometry image: image based 3D models for digital broadcasting applications. *TBC* 2011; **57**(3): 636–645.
18. Karni Z, Gotsman C. Spectral compression of mesh geometry. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH'00. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000; 279–286.
19. Liu Y, Prabhakaran Y, Guo X. *Dirichlet harmonic shape compression with feature preservation for parameterized surfaces*, Vol. 29(7): 2039-2048. Computer Graphics Forum, Blackwell Publishing Ltd, 2010.
20. Alexa M, Müller W. Representing animations by principal components. *Computer Graphics Forum,* Blackwell Publishers Ltd 2000; **19**(3): 411–418.
21. Karni Z, Gotsman C. Compression of soft-body animation sequences. *Computers & Graphics* 2004; **28**: 25–34.
22. Rong G, Cao Y, Guo X. Spectral mesh deformation. *The Visual Computer* 2008; **24**(7): 787–796.
23. Rong G, Cao Y, Guo X. Spectral surface deformation with dual mesh, In *Proceedings of International Conference on Computer Animation and Social Agents (CASA 2008)*, Grand Hilton Seoul, Korea, 2008; 17–24.
24. Tang ZY, Rong G, Guo X, Prabhakaran B. Streaming 3d shape deformations in collaborative virtual environment, In *IEEE VR 2010*, Waltham, Massachusetts, USA, 2010; 183–186.
25. Tang Z, Ozbek O, Guo X. Real-time 3D interaction with deformable model on mobile devices, In *ACM MM'11*, Scottsdale, Arizona, USA, 2011; 1009–1012.

26. Vallet B, Lévy B. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum,* Blackwell Publishing Ltd, 2008, 27(2): 251–260.
27. Liu Y, Prabhakaran B, Guo X. Point-based manifold harmonics. *IEEE Transactions on Visualization and Computer Graphics* 2012; **18**(10): 1693–1703.
28. Belkin M, Sun J, Wang Y. Constructing Laplace operator from point clouds in Rd. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009; 1031–1040.
29. Botsch M, Sumner R, Pauly M, Gross M. Deformation transfer for detail-preserving surface editing, In *Proceedings of Vision, Modeling, and Visualization (VMV)*, Aachen, Germany, 2006; 357–364.
30. Kobbelt L, Vorsatz J, Seidel H. Multiresolution hierarchies on unstructured triangle meshes. *Computational Geometry* 1999; **14**(1C3): 5–24.
31. ZHAO QP. A survey on virtual reality. *SCIENCE CHINA Information Sciences* 2009; **52**(3): 348.
32. Lipman Y, Sorkine O, Cohen-Or D, Levin D, Rössl C, Seidel HP. Differential coordinates for interactive mesh editing, In *SMI'04*, Palazzo Ducale, Genova, Italy, 2004; 181–190.
33. Sorkine O. Differential representations for mesh processing. *Computer Graphics Forum* 2006; **25**(4): 789–807.
34. Alexa M. Differential coordinates for local mesh morphing and deformation. *The Visual Computer* 2003; **19**(2): 105–114.
35. Dong S, Bremer PT, Garland M, Pascucci V, Hart JC. Spectral surface quadrangulation. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06. ACM, New York, NY, USA, 2006; 1057–1066.
36. Lipman Y, Levin D, Cohen-Or. D. Green coordinates. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08. ACM, New York, NY, USA, 2008; 78:1–78:10.
37. Sumner RW, Popović J. Deformation transfer for triangle meshes. *ACM Transactions on Graphics* 2004; **23**(3): 399–405.
38. Jacobson A, Baran I, Popović J, Sorkine O. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)* 2011; **30**(4): 78:1–78:8.
39. Botsch M, Sorkine O. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics* 2008; **14**(1): 213–230.
40. Rivers AR, James DL. Fastlsm: fast lattice shape matching for robust real-time deformation. In *ACM Transactions on Graphics (TOG)* Vol. 26(3): 82, SIGGRAPH '07. ACM, New York, NY, USA, 2007.

AUTHORS' BIOGRAPHIES

Lin Zhang is a PhD student at Beihang University, Beijing, China. He received the Master's degree in Computer Software Engineering from the Ocean University of China, Qingdao city, in 2008. His research interests include distributed simulation, tele-immersion, multimedia, and virtual reality.

Fei Dou was born in 1988. He graduated from the College of Computer Science and Technology at Jilin University of China, Changchun city, in 2012. Currently, he is studying at Beihang University for his Master's degree. He is a student of the State Key Lab of Virtual Reality Technology and Systems.

Zhong Zhou is an associate professor at State Key Lab of Virtual Reality Technology and Systems, Beihang University, Beijing, China. He received his BS degree from Nanjing University and PhD degree from Beihang University in 1999 and 2005, respectively. His main research interests include tele-immersion, natural phenomena simulation, distributed virtual environment, and Internet-based virtual reality technologies. He is a member of China Computer Federation, Association for Computing Machinery, and Institute of Electrical and Electronics Engineers.

Wei Wu is a professor in the School of Computer Science and Engineering at Beihang University; he is currently the chair of the Technical Committee on Virtual Reality and Visualization of China Computer Federation. He received the PhD degree from Harbin Institute of Technology, China, in 1995. He has published more than 90 papers, 33 issued patents, and one book. His current research interests involve real-time 3D reconstruction, remote immersive system, and augmented reality.