Streaming 3D Shape Deformations in Collaborative Virtual Environment

Ziying Tang, Guodong Rong, Xiaohu Guo and B. Prabhakaran
Computer Science Department, University of Texas at Dallas

ABSTRACT

Collaborative virtual environment has been limited on static or rigid 3D models, due to the difficulties of real-time streaming of large amounts of data that is required to describe motions of 3D deformable models. Streaming shape deformations of complex 3D models arising from a remote user's manipulations is a challenging task. In this paper, we present a framework based on spectral transformation that encodes surface deformations in a frequency format to successfully meet the challenge, and demonstrate its use in a distributed virtual environment. Our research contributions through this framework include: i) we reduce the data size to be streamed for surface deformations since we stream only the transformed spectral coefficients and not the deformed model; ii) we propose a mapping method to allow models with multi-resolutions to have the same deformations simultaneously; iii) our streaming strategy can tolerate loss without the need for special handling of packet loss. Our system guarantees real-time transmission of shape deformations and ensures the smooth motions of 3D models. Moreover, we achieve very effective performance over real Internet conditions as well as a local LAN. Experimental results show that we get low distortion and small delays even when surface deformations of large and complicated 3D models are streamed over lossy networks.

KEYWORDS: Collaborative Interaction, Distributed Virtual Reality, 3D Shape Deformation.

INDEX TERMS: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems — Artificial, augmented, and virtual realities.

1 INTRODUCTION

With the rapid advances of networks, computer-assisted collaborative virtual environment has been profoundly involved into many aspects of our daily life. Traditional collaborative virtual environments mainly employ video, audio, or text to communicate, and extensive research has helped technologies to transmit these media to remote sites in real-time. Compared with traditional media, 3D models can represent objects in a more realistic and comprehensive manner. Therefore, they have received increasing attentions in collaborative virtual environments. However, the current state-of-the-art technologies cannot facilitate a real-time collaborative virtual environment where large and complicated 3D models and their related shape deformations are involved. Prior researchers have proposed algorithms to effectively transfer large and complex 3D models over network under different conditions, nevertheless these methods focus on *static*/*rigid* 3D models only. Some researchers generate 3D models' shape deformations offline, and stream them as animations. But these methods work off-line instead of real-time.

Real-time streaming of 3D models' deformable motions faces two grand challenges, namely high computational cost and requirement of transmitting a large amount of data in real-time. For a deformable 3D model, the number of DOFs could be as large as the size of the 3D model. Therefore describing the shape deformations of a 3D deformable model requires a large amount of data. For instance, if a 3D model with size 2MB is included in a collaborative environment and we share its shape deformations with remote users in real time. To achieve a smooth motion of the 3D model, we require a transmission rate of 20fps. Thus, we need to deliver 40MB data every second, which is too heavy for current network speed.

There are two straightforward ways to share 3D shape deformations remotely, but we argue neither of them is good enough to achieve real-time streaming smoothly. First method is calculating the shape deformation in one high performance machine, and then transmitting the deformed 3D mesh. However, this method leads to a very heavy transmission load, so that it is not possible to reach real-time speed. Another method is streaming control points/handles directly. Nevertheless, it requires all the remote users to have very high computational power to locally complete the deformation computation, which cannot be performed by users with low-end devices.

In this paper, we address these challenges by proposing an interactive collaboration framework that supports streaming of 3D shape deformations. Adopting a spectrum-based representation of shape deformation, we achieve a real-time streaming speed. Building over a server-client-based distributed environment, our framework offers users from remote locations to manipulate the same 3D model and share its deformation. Our system supports computations over multi-resolution models, as well as streaming over lossy networks. Therefore, the framework can be used for a wide range of applications. There are several novelties in our work: 1) we use manifold harmonic transformation [10] to convert surface deformations into a reduced frequency subspace; thus the DOFs in our spectrum-driven deformable models can be effectively minimized, in order to achieve high streaming speed; 2) we propose a mapping method to allow models with "spatial multi-resolutions" to have the same deformations simultaneously, which allows users with low-end computation and display capabilities to see lower resolutions of the same model with the same deformation behaviour; 3) our spectrum representation of deformable models provides a natural "spectral multi-resolution" structure to adaptively encode their deformation behaviour in different scales, which can tolerate packet loss with only small distortion errors when streamed over lossy networks. The stream load is only several KB in general, independent of the 3D models' data sizes. In addition, the performance of our system over real Internet conditions is comparable to that over a local LAN.

Currently, we do not consider the physical properties of the models. However, extending our spectrum representation and streaming framework to other physics-based deformation methods will be straight-forward.

{zxt061000 | guodongrong | xguo | praba}@utdallas.edu

IEEE Virtual Reality 2010
20 - 24 March, Waltham, Massachusetts, USA
978-1-4244-6236-0/10/$26.00 ©2010 IEEE

2 Related Works

In this section, we briefly review some previous researches in the fields of both 3D mesh streaming and shape deformation.

Static 3D model streaming includes progressive compression and its related transmission techniques. Progressive Mesh (PM), as a multi-resolution technique to progressively render 3D meshes is suggested by Hoppe [5] in 1996. In a similar way, Progressive Forest Split (PFS) method [12] and Compressed Progressive Meshes (CPM) method [8] group edge collapses into batches to achieve a higher compression ratio. Another research branch is from the network's point of view to find efficient streaming strategies. Consider clients' viewpoint information, researchers suggest to transmit the particular part of the mesh that is visible for clients [4]. Yan et al. [14] and AlRegib et al. [1] separately propose different approaches using redundant bits to minimize the distortion caused by packet loss. Cheng et al. [3] suggest an analytical model to measure streaming of progressive meshes. Li et al. [7] suggest a prediction method to tolerate packet loss.

For 3D shape deformation applications in interactive surface design rather than simulation, a merely physically *plausible* result is sufficient. A detailed survey on linear mesh deformation algorithms can be found in [2]. Energy minimization has been a general approach to deform smooth surfaces [13]. Smoother and smoother meshes can be built up by removing surface details [11]. The deformation is applied on the smoothed surface, and the details are added back afterwards. Topological hierarchy of coarser and coarser meshes, e.g. subdivision surfaces [15], has been used in multi-resolution editing. The number of unknowns could be significantly decreased. However, automatically building subdivision for arbitrary irregular meshes is non-trivial, since the original irregular mesh may not have coherent subdivision connectivity.

Rong et al. [10] propose a spectral deformation method, which is based on manifold harmonic transformation to achieve interactive speed for manipulating large and complex triangle mesh surfaces. Similar to other subspace deformation method [6], spectral deformation can be considered as performing the deformation computation in a reduced spectral subspace. The advantage is that the manifold harmonic transformation can be easily computed on any polygonal meshes automatically, without any manual construction of the embedded subspace [6]. In our current framework, we use manifold harmonics to encode shape deformations in the frequency domain.

3 Collaborative System and Algorithm

The spectrum-based surface deformation method [10] provides us a platform to quickly complete computations of the deformation. However, when we apply it into a distributed collaborative system, we would like to include users with different display capabilities and real network conditions where data loss may occur.

3.1 Deformation of Multi-resolution Models

In a general distributed system, users may not have the same computational power and display capability. Besides it is not necessary to use high resolution all the time for some applications. Thus, we consider the deformation over multi-resolution surfaces.

Mesh surfaces in different levels of resolution have different numbers of vertices but keep a similar geometric shape. Multi-resolution refers to different levels of resolution in the spatial domain. However, the Laplacian-based deformation is solved over the frequency domain instead of spatial domain to reach a real time speed; thus we need to build a spectral relationship between different resolutions of surfaces.

In order to represent the relationship between a high resolution surface S with n vertices and a low resolution surface $\hat{S}$ with n' vertices, we build a mapping matrix $\mathbf{M}$ with the size of $n' \times n$. For each of the vertex $\hat{v}_i$ (i= 1… n') in surface $\hat{S}$, we find out the nearest triangle t_i on surface S. The barycentric coordinates of vertex v_i' on triangle t_i are put into the i^{th} row of matrix $\mathbf{M}$ according to the columns corresponding to vertices of triangle t_i. Figure 1 illustrates the construction procedure.

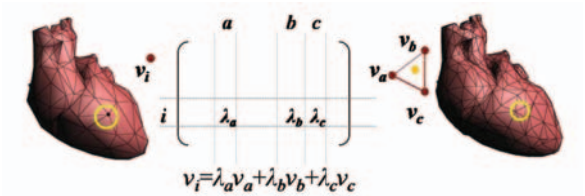


Figure 1. Mapping between different resolutions. v_i is a vertex of a low resolution model. It maps to a triangle $v_a v_b v_c$ on a high resolution model with barycentric coordinates λ_a, λ_b, and λ_c.

By doing so, we obtain the mapping matrix $\mathbf{M}$ which is used to link the two mesh surfaces in different resolutions. According to [10], we can easily get the *manifold harmonic bases* (MHB) matrix $\mathbf{H}$ of a high resolution surface S, where $\mathbf{H}$ is an $n \times m$ matrix and m is number of frequencies we use. We use it to define the *manifold harmonic transformation* (MHT) and its inverse [10]. The MHT describes the conversion of the manifold surface from spatial domain into frequency domain. Therefore, we can use a $n' \times m$ matrix $\mathbf{K}=\mathbf{MH}$ to approximate the MHB of the low resolution surface $\hat{S}$ and use it during the Laplacian-based deformation. Thus, we redefine the inverse MHT of the low resolution surface $\hat{S}$ as following:

$$\hat{\mathbf{x}} = \mathbf{M}\mathbf{x} = \mathbf{M}\mathbf{H}\tilde{\mathbf{x}} = \mathbf{K}\tilde{\mathbf{x}} \quad (1)$$

where $\hat{\mathbf{x}}$ and $\mathbf{x}$ are x-coordinates of the meshes $\hat{S}$ and S, and $\tilde{\mathbf{x}}$ is x-coordinates of mesh S in frequency domain.

By applying the redefined MHT, we only need to solve linear system once for a high resolution surface, and results can be used to perform deformation process of low resolution meshes. Moreover, it is not necessary to compute the MHB for the low resolution mesh surface, so we can save some pre-computing time.

3.2 Deformation Transmission with Data Loss

In order to make our system more robust, we consider a general network environment in which data loss may occur during transmission. There is no need for our system to deal with data loss; using only partial of the linear system solution, we can still reconstruct the deformed model. Specifically, solving the linear system with m unknown variables gives us $[\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_m]$, which is a list of $\tilde{x}$-coordinates ($\tilde{y}$- and $\tilde{z}$- coordinates respectively) in frequency domain. When data loss occurs, we get parts of the m solutions and reconstruct the deformed mesh using the inverse MHT. We explain this using an example. Suppose we choose frequencies m = 100 and loss ratio is 20%. So the client gets 100*(1-20%) =80 solutions, for example $[\tilde{x}_1, \tilde{x}_2, \ldots \tilde{x}_k, \tilde{x}_{k+20} \ldots \tilde{x}_{100}]$ ($\tilde{y}$- and $\tilde{z}$- coordinates respectively) are arrived at the client. Following the inverse MHT, we can use 80 frequencies instead of 100 to reconstruct deformed smooth model in the spatial domain. As mentioned in [10], low frequency components carry the model structure while high frequencies represent geometric details. Since we have data loss during transmission, and loss of low frequencies will affect results more significantly, so in our system, we do not allow loss of the first few, say 10, frequencies (The number can be determined based on the allowable distortion). There are many transmission protection methods. One simply way is to insert redundant low frequency data to ensure a high probability of successful transmission. Another obvious way is to use a reliable transport mechanism such as TCP for the first few frequencies.

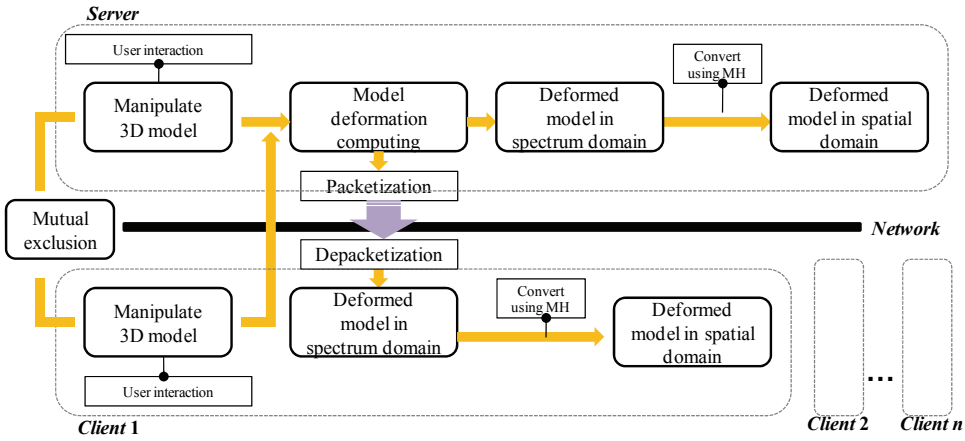


Figure 2. The proposed interactive deformation transmission system.

3.3 System and Algorithm

In order to maintain a consistent state all the time, we adopt the client-server architecture in our system implementation. We use a centralized server to perform complex computation, and communicate related state changes to all the connected clients. Therefore, all users in the environment will experience the same outcome.

We consider the following aspects when designing our distributed system. First, 3D deformable models can be very large and complicated. Second, realistic shape deformations must be computed quickly to achieve a real-time speed. Third, global consistency should be maintained to prevent divergent simulations at clients. In our system, no deformation computation is performed locally. User interactions are sent directly to the server, and the clients update their local representations when they receive a feedback message from the server. The deformation is encoded as frequencies and sent back to clients. In our experiments, we only use the first few frequencies, e.g. 100, so the data communicated between the server and clients is very small. For example, we take first 100 frequencies to compute deformation, and the visual update rate is about 20 fps, so less than 10Kbps will be communicated in the system. Thus, our framework reaches a real-time transmission and guarantees a smooth deformation of the 3D model as well. Additionally, a distributed mutual exclusion module is included to ensure the success of multiple users' operations. Figure 2 illustrates the proposed collaborative system.

The general process of our algorithm for deformation transmission is described as the following steps:

1. Compute MHB for the high resolution mesh with n vertices.
2. Compute the mapping matrix **M** and matrix **K** for low resolution mesh with n' vertices.
3. Compute the deformation on the high resolution model by solving a linear system with m unknown variables.
4. Transfer the solution of m unknowns to clients.
5. Update the 3D model locally to get the final result.

Steps 1 and 2 belong to pre-processing which is performed on the server where high computation power is available, and then results are streamed to clients. Note that the first two steps are completed once offline and do not affect running time. Step 3 is finished by the server and step 5 is the task of clients.

4 Experimental Results

Our interactive distributed virtual environment and deformation transmission algorithm are implemented on several Windows XP PCs with Intel Core2 Duo 2.93GHz CPU and 2GB DDR2 RAM. The system is communicated over various network conditions simulated using FreeBSD and Dummynet [9]. Our algorithm is developed using Visual C++.NET 2005 and we use the LU-solver to solve the linear system. The number of frequencies (m) is set to 100 in our experiments.

We use normalized *Hausdorff Distance* (HD) as an error metric to evaluate distortions of our deformation transfer. The following equation gives the definition of the *Hausdorff Distance* between two mesh surfaces S and S':

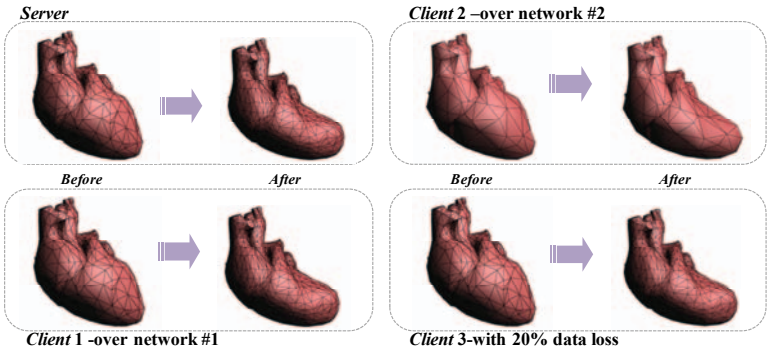
$$H(S,S') = \max\{h(S,S'),\ h(S',S)\} \qquad (2)$$

where $h(S,S') = \max_{v\in S}\min_{v'\in S'}\|v-v'\|$, and $\|v-v'\|$ represents the Euclidean distance of v and v'.

We list in Table 1 the details of the models and the computational time. The last 3 steps of our algorithm are included in run–time. Table 1 contains computing time costs of experiments running in the high resolutions (HR) and low resolution (LR). Obviously, our new deformation transmission algorithm is remarkably fast and works at real time speed. Moreover, for supporting multi-resolution, we only need to spend a little longer time during the pre-computing process. The running time consists of both the running time at the server side and at the client side. The computational burden of the client is very light, especially for the low resolution case.

Table 1. The computing time of our algorithm on both server and clients for high resolution (HR) and low resolutions (LR)

Model	Model details H/L		Pre-computing time (s)		Running time (s) - Server	Running time (s) - Client	
	#Vertex	#Triangle	HR	LR		HR	LR
Heart	1752/252	3500/500	0.613097	0.652426	0.005026	0.003659	0.00176
Lion	5000/2502	9996/5000	1.961939	2.186335	0.013542	0.023987	0.01195
Brain	26077/5002	52150/10000	25.48497	25.949494	0.067058	0.058571	0.030488
Dragon	50000/2500	100000/5000	49.67642	49.93512	0.131773	0.170512	0.026368



Hausdorff Distance	Before	After
Client 1	0.0000	0.0000
Client 2	0.013104	0.024350
Client 3	0.0000	0.003602

Figure 3. Deformation transmission results: Client 1 runs a high resolution model over network #1 with 7Mb/s bandwidth and 1% data loss; client 2 runs a lower resolution model over network #2 with 1Mb/s bandwidth and 2% data loss; and client 3 runs over networks with 20% data loss. The numbers in the table show the *Hausdorff Distance* between the results on server and clients.

In addition to a real-time speed, our system ensures successful deformation transmission as well as smooth and natural motion of the 3D models. We test our system over two different Internet conditions and show results in Figure 3. We demonstrate the results of the deformation transmission over both single resolution and multi-resolutions in this figure. Moreover, to better illustrate the tolerance of packet loss in our system, we simulate transmission with 20% data loss in Figure 3 as well. In this figure, we present the geometric distortions between two surfaces using *Hausdorff Distance*. As we can see, the results of the deformation transmission in our framework are very good. We are able to get deformed 3D model successfully in real Internet conditions and even with high percentage of data loss.

We show our analysis results on the geometric distortion according to different loss ratio in the Figure 4. As expected, distortions are very limited, even though with the increasing of data loss, the distortion raises as well.

5 Conclusion and Future Works

In this paper, we have proposed an interactive collaborative framework, where 3D deformable models are deformed in real-time according to users' manipulations. Adopting a spectrum-based surface deformation method, we describe deformation using frequencies in order to obtain a high streaming speed. We support users with different display capacities, and we can tolerate packet loss with only small distortion errors when streaming over lossy networks. The experimental results show that 3D surface deformation can be transmitted efficiently. In addition, all the deformation is computed by the server, which saves users' computational cost and gives more flexibility on users. In the future, we would like to extend our deformable model streaming framework to physics-based deformations in order to simulate either elastic or plastic materials and achieve more realistic deformation results for different materials. Moreover, haptic devices can be integrated into our framework to provide more intuitive manipulation experiences for remote users.

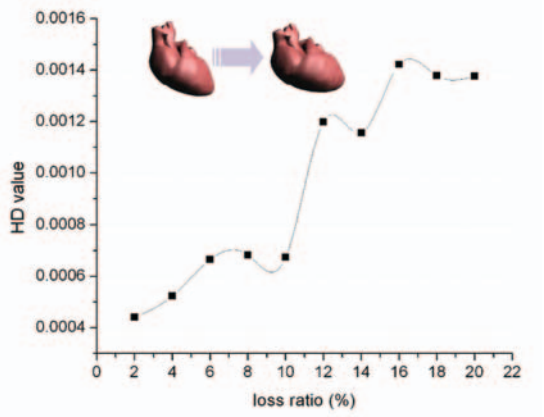


Figure 4. The geometric distortion of the deformed 3D model with different data loss ratio, based on Hausdorff Distance. Loss ratio is from 2% to 20%.

References

[1] G. AlRegib, Y. Altunbasak, and J. Rossignac. Error-resilient transmission of 3D models. *ACM Trans. Graph*. 24, 182–208, 2005

[2] M. Botsch, and O. Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1), 213–230, 2008.

[3] W. Cheng, W. T. Ooi, S. Mondet, R. Grigoras, and G. Morin. An analytical model for progressive mesh streaming. In *Proceedings of 15th ACM International Multimedia Conference*. 24–29, 2007.

[4] W. Guan, J. Cai, J. Zheng, and C. Chen. Segmentation-based view-dependent 3-D graphics model transmission. *IEEE Transaction on Multimedia*, 10(5). 724–734, 2008.

[5] H. Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH'96*. 77–108, 1996.

[6] J. Huang, X. Shi, X. Liu, K. Zhou, L.Y. Wei, S.H. Teng, H. Bao , B. Guo, and H.Y. Shum. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics*, 25(3). 1126-1134, 2006.

[7] H. Li, Z. Tang, X. Guo, and B. Prabhakaran. Loss tolerance scheme for 3D progressive meshes streaming over networks. In *Proceedings of IEEE International Conference on Multimedia & Expo*. 501–504, 2008.

[8] R. Pajarola, and J. Rossignac. Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1). 79–93, 2000.

[9] L. Rizzo. Dummynet. http://info.iet.unipi.it/~luigi/ip_dummynet/.

[10] G. Rong, Y. Cao, and X. Guo. Spectral mesh deformation. *The Visual Computer*, 24(7-9). 787– 796, 2008.

[11] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of ACM SIGGRAPH 1995*. 351–358, 1995.

[12] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus. Progressive forest split compression. In *Proceedings of SIGGRAPH 1998*. 123–132, 1998.

[13] W. Welch, and A. Witkin. Variational surface modeling. *Computer Graphics*, 26(2). 157–166, 1992.

[14] Z. Yan, S. Kumar, and C. Kuo. Error resilient coding of 3D graphic models via adaptive mesh segmentation. *IEEE Trans. Circuits and Systems for Video Technology*, 11(7). 860–873, 2001.

[15] K. Zhou, X. Huang, W. Xu, B. Guo, and H.Y. Shum. Direct manipulation of subdivision surfaces on GPUs. *ACM Transactions on Graphics*, 26(3). 91, 2007.