3D Model Streaming Based on JPEG 2000

Nein-Hsien Lin, Ting-Hao Huang, and Bing-Yu Chen, *Member*, IEEE

Abstract — *For PC and even mobile device, video and image streaming technologies, such as H.264 and JPEG/JPEG 2000, are already mature. However, the streaming technology for 3D model or so-called mesh data is still far from practical use. Therefore, in this paper, we propose a mesh streaming method based on JPEG 2000 standard and integrate it into an existed multimedia streaming server, so that our mesh streaming method can directly benefit from current image and video streaming technologies. In this method, the mesh data of a 3D model is first converted into a JPEG 2000 image, and then based on the JPEG 2000 streaming technique, the mesh data can then be transmitted over the Internet as a mesh streaming. Furthermore, we also extend this mesh streaming method for deforming meshes as the extension from a JPEG 2000 image to a Motion JPEG 2000 video, so that our mesh streaming method is not only for transmitting a static 3D model but also a 3D animation model. To increase the usability of our method, the mesh stream can also be inserted into a X3D scene as an extension node of X3D. Moreover, since this method is based on the JPEG 2000 standard, our system is much suitable to be integrated into any existed client-server or pear-to-pear multimedia streaming system*[1].

Index Terms — Mesh Streaming, JPEG 2000, Geometry Image, X3D.

I. INTRODUCTION

Recently, 3D graphics over the Internet has attracted a lot of attention, such as web-based virtual shopping malls, on-line 3D games, etc. For supporting these applications, the demand of transmitting 3D models increased significantly. Being able to view or play a 3D model or a 3D scene composed of many sophisticated 3D models over the Internet is one of the goals of VRML (Virtual Reality Modeling Language) [16] and X3D (eXtensible 3D) [20]. However, due to the increase of the model complexity and file size, even with the increase of network bandwidth, to download the 3D models would still take a lot of time. To reduce the waiting time for downloading the 3D models, mesh streaming mechanism must be available as what has been done in the video and image streaming.

Therefore, in this paper, we propose a mesh streaming method by utilizing the benefits of JPEG 2000 (J2K) [17]. This method takes the advantages of geometry image [5] which can convert the mesh data of a 3D model into an image. This could reduce the problem of 3D mesh streaming and transfer it to a 2D image streaming problem. There are also many 2D image compression methods that can be used to further downsize the file of the geometry image. This paper made use of the JPEG 2000 compression due to some of its good characteristics, such as ROI (Region Of Interest), progressive compression, multiple components, etc. Moreover, since this method is based on JPEG 2000, which is a famous standard, our system is much suitable to be integrated into any existed client-server or pear-to-pear multimedia streaming system.

Based on our mesh streaming method, during the downloading process, the user can first obtain an approximate shape and then the 3D model will become clearer when more data is received. Besides the progressive transmission, the view-dependent issue is also taken into consideration. Hence, the most significant part of the 3D model or the part faced to the user will be refined earlier. Furthermore, since the 3D animation models or so-called deforming meshes are used more frequently than static 3D models, we also extent our mesh streaming method for transmitting a 3D animation model as a deforming meshes streaming by converting a 3D animation model into a Motion JPEG 2000 video. To increase the usability of our mesh streaming method, the 3D model or 3D animation model represented by a JPEG 2000 image or a Motion JPEG 2000 video can be inserted into a X3D scene as an extension node of X3D.

This paper is organized as following. In Section II, the related work is introduced. In Sections III and IV, we describe the details of our mesh streaming method and the network architecture of our mesh streaming system. Sections V and VI illustrate the results and a short discussion on the future work of this method.

II. RELATED WORK

In this session, we first introduce some background related to the main themes of this paper including 3D streaming, geometry images, and image compression.

A. Mesh Streaming and Compression

In general, the main goal of mesh streaming is to provide 3D contents in real-time for users over the network links, such that the interactivity and visual qualities of the contents may match as closely as when they were stored locally. The

[1] This work is partially supported by the National Science Council of Taiwan under the numbers: NSC92-2218-E-002-056, NSC93-2622-E-002-033, NSC94-2622-E-002-024, and NSC95-2221-E-002-273.

Nein-Hsien Lin is with the National Taiwan University (e-mail: ppbb@cmlab.csie.ntu.edu.tw).

Ting-Hao Huang is with the National Taiwan University (e-mail: richardg@ cmlab.csie.ntu.edu.tw).

Bing-Yu Chen is with the National Taiwan University (e-mail: robin@ntu.edu.tw).

Manuscript received January 15, 2007

resource bottleneck is often assumed to be the bandwidth and not rendering or processing power [14]. To achieve this, simplification and progressive transmission are two dominant strategies [12]. Existing mesh streaming techniques may be categorized into four main types [3], and in this paper the main point we focus is about the object streaming using geometry images.

Hoppe introduced the concept of progressive meshes (PM) [6], which store an arbitrary triangular mesh $\mathbf{M}$ as an appearance-preserving but much coarser base mesh $\mathbf{M}_0$ and a number of refinement pieces constructed by using iterative edge collapse operations. Then, the 3D model can be represented as $\mathbf{M}_0$, $\mathbf{M}_1$, $\mathbf{M}_2$, ..., $\mathbf{M}_n = \mathbf{M}$, where $\mathbf{M}_1$, $\mathbf{M}_2$, ..., and $\mathbf{M}_n$ are the refined multi-resolution meshes by splitting one vertex in the previous meshes $\mathbf{M}_0$, $\mathbf{M}_1$, ..., $\mathbf{M}_{n-1}$. A user in the client may view or interact immediately with the 3D model once the base mesh $\mathbf{M}_0$ is downloaded. Streaming additional parts incrementally refines the mesh and restores the original mesh $\mathbf{M}_n = \mathbf{M}$ exactly. Geometrical meshes thus can be streamed from servers to clients, making interactions with 3D data possible without a complete download [2].

A different concept introduced by Isenburg and Lindstrom is streaming meshes (SM) [9]. They reorganize the triangle mesh, so that highly spatial related triangles are packed with each other in the file. While rendering a triangle, other triangles close to it will be found locally in the file, thus avoids most file seek system calls. So we can see this mesh clearly from some parts to entire. The main goal of this scheme is designed for memory out-of-core situation.

The third concept is to compress the 3D model [15]. Originally this technique is designed for the limited bandwidth between CPU and GPU, but it can also be used for network transmission. There are two categories of mesh compression: geometry compression and connectivity compression, where the geometry compression dominates the final file size. To compress a 3D model, the vertex data is first quantized, and then some predictions based on observations are made. These predictions can provide clues for entropy/arithmetic coding.

In this paper, we use the properties of JPEG 2000 to combine the concepts of PM and SM, so that a remote user can view the 3D model not only progressively but also clearly from user's viewpoint.

B. *Parameterization and Geometry Image*

Surface parameterization is to find a mapping function $\mathrm{F}: \mathbf{R}^3 \leftrightarrow \mathbf{R}^2$ which maps 3D coordinates (x, y, z) to 2D coordinates (s, t), i.e., $\mathrm{F}(x, y, z) = (s, t)$. Geometry images [5] provided by Gu *et al.* is one of the surface parameterization methods. In this method, given a 3D model, we can use a 2D image to represent it, where the R, G, B values (R_i, G_i, B_i) of each pixel i of the image are used to represent the 3D coordinates (x_i, y_i, z_i), normal vectors $(N_{x_i}, N_{y_i}, N_{z_i})$, or texture coordinates (u_i, v_i) of the vertex i. Hence, the number of vertices N of the reconstructed 3D model will be the same as the resolution of the geometry image $w \times h$, i.e., $N = w \times h$.

To reconstruct the 3D model from a geometry image, we first convert the R, G, B values (R_i, G_i, B_i) of each pixel i of the geometry image to be vertex positions (x_i, y_i, z_i) of vertex i. Then, we connect the vertex i to its horizontal and vertical neighbors to be several quadrangles, and then we choose the shorter distance between the two diagonals of each quadrangle and connect the opposite vertices. Hence, the reconstructed 3D model can be represented as a triangular mesh. Finally, the vertex attributes, such as normal vectors $(N_{x_i}, N_{y_i}, N_{z_i})$ or texture coordinates (u_i, v_i), can also be assigned to the vertex i.

To extend the geometry images for 3D animation model or so-called deforming meshes, Briceño *et al.* present geometry videos [1] to provide a new data structure to encode deforming meshes. Their data structure provides a way to treat deforming meshes as a video sequence and is well suitable for network streaming. They also offer the possibility to apply and adapt existing mature video processing and compression techniques, such as MPEG.

C. *JPEG 2000 and JPIP*

Fig. 1. The four types of progression provided by JPEG 2000: (a) progression by quality; (b) progression by locality; (c) progression by component; (d) progression by resolution.

The JPEG 2000 standard supports lossy and lossless compression of single-component (e.g., grayscale) and multi-component (e.g., color) imagery. In addition to this basic compression functionality, however, numerous other features are provided, including: (a) progressive recovery of an image

by resolution, image quality, component, and locality; (b) ROI coding, whereby different parts of an image can be coded with differing fidelity; (c) random access to particular regions of an image without needing to decode the entire code stream. Due to its excellent coding performance and many attractive features, JPEG 2000 has a very large potential application base and can be used for image archiving, Internet, web browsing, document imaging, digital photography, medical imaging, remote sensing, and desktop publishing. Fig. 1 illustrates how the four types of progression work.

The high flexibility of JPEG 2000 code stream provides many possible applications, especially for browsing image file on the Internet. JPIP (JPEG 2000 Internet Protocol) [18] is a protocol which is implemented on top of HTTP (HyperText Transfer Protocol) and facilitates the flexibility of JPEG 2000. Suppose that a user is browsing a large image file, for example the satellite image of a city, by using a JPIP compatible browser. The user may first view the image block which contains only one building. In this situation, only the code stream which corresponds to the building is needed to be transferred, and the server does not have to transmit the remaining code stream. After the user scroll the image, a new request for JPEG 2000 code stream is sent from the client, and the server will return the corresponding image data that the user wants to see.

In this paper, since the mesh data of a 3D model is converted into a JPEG 2000 image, the mesh streaming mechanism can then be integrated into a JPIP server. Hence, the JPIP server can not only be used to transmit image and video streaming, but also the mesh streaming.

III. JPEG 2000-Based Mesh Streaming

A. Overview

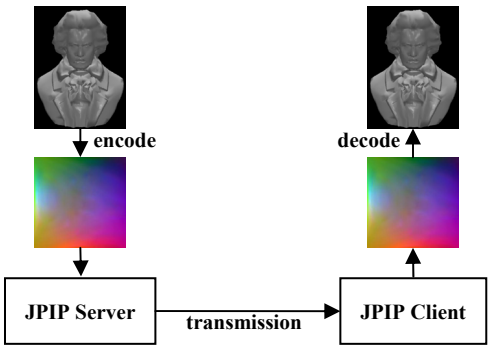

Fig. 2. The system hierarchy of the JPEG 2000-based mesh streaming.

The system hierarchy of our JPEG 2000-based mesh streaming method is shown in Fig. 2. To transmit a 3D model, we first parameterize it and encode it as a JPEG 2000 image (Sections B and C), then the JPEG 2000 geometry image can be transmitted through any JPIP server. To support the view-dependent and multi-resolution schemes, we modify the transmission sequence of an existed JPIP server (Section D). Hence, through our modified JPIP server, the system can perform not only the view-dependent, quality-scalable, and multi-resolution mesh streaming, but also the view-dependent and multi-resolution JPEG 2000 image streaming. Besides, we also modify an existed X3D browser to make the browser can show the JPEG 2000-based mesh streaming as showing a normal 3D model (Section E). Furthermore, to support the 3D animation model, as we encode a 3D model as a JPEG 2000-based mesh streaming, we also encode a 3D animation model as a Motion JPEG 2000-based deforming meshes streaming (Section F), i.e., the 3D animation model is encoded as a Motion JPEG 2000 (MJ2K) code stream [18].

B. Parameterization

Our mesh streaming method first converts a 3D model to a geometry image [5]. A geometry image is an extension of surface parameterization; more specifically, it uses the R, G, B values of an image to represent the 3D model's attributes, such as vertex positions, normal maps, texture coordinates, etc. The surface parameterization, on the other hand, is to find an one-to-one mapping function $F: \mathbf{R}^3 \leftrightarrow \mathbf{R}^2$, such that $F(x_i, y_i, z_i) = (u_i, v_i)$, where (x_i, y_i, z_i) means the position of the vertices and (u_i, v_i) is its corresponded position in the parameterization space, such as the pixels of an image.

Thus, while processing a model, if the 3D model is closed manifold, we first use the cut method in [5] to find the proper boundary of the model. Then, we use Floater's surface parameterization method [4] to flatten the model which has been cut to be a surface of 2-manifold with boundary or originally is a 2-manifold with boundary. Hence, the model is flattened to a 2D surface and the boundary of the model is mapped to a square. We then resample the grid points in the image and calculate the attributes with bi-linear interpolation, such as vertex positions, normal maps, texture coordinates, etc. Finally, we normalize the interpolated attributes to get the corresponding R, G, B values.

Hence, when the client side receives this JPEG 2000 encoded geometry image, the system will first decode the JPEG 2000 image and then use the decoded image to reconstruct the 3D model as described in Section II.B.

C. JPEG 2000 Coding

The geometry image is not the actual image that we transmit over the Internet, due to its file size and the fact that we have to transmit a number of images to recover all of the attributes. JPEG 2000 compression method was employed to address this problem, which is chosen for the reasons below:

- JPEG image is the most famous image compression format on the Internet, and so is JPEG 2000.
- JPEG 2000 has a lossless compression mode.
- JPEG 2000 has a greater compression rate compared to the older methods.
- JPEG 2000 supports multiple layer compression, thus we can compress the data of all attributes into one image.

- JPEG 2000 supports progressive compression and decompression, which allowed us to achieve progressing transmission over the Internet.
- Arbitrary image block can be retrieved from a JPEG 2000 image, which is very suitable for view dependent transmission.

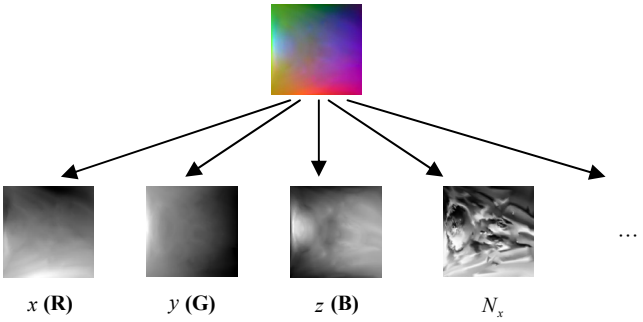

Fig. 3. The JPEG 2000 geometry image of the Beethoven model as shown in Section V. This JPEG 2000 image includes the vertex positions which are encoded as the R, G, B information of the JPEG 2000 image and the normal map which is encoded as an image and is hided as a layer of the JPEG 2000 image

To encode the geometry image as a JPEG 2000 image or code stream, we utilize the above features of JPEG 2000 to store a 3D model and its all attributes. For each vertex attribute, such as vertex position (x,y,z), normal vector (N_x,N_y,N_z), or texture coordinates (u,v), each component is represented by one image layer, i.e., there are three image layers for representing x, y, and z components of vertex position (x,y,z). Since JPEG 2000 supports multi-layer compression, it allows us to easily compress all sets of information into only one image. Hence, from one JPEG 2000 image as shown in Fig. 3, we can obtain all of the x, y, z (vertex position), N_x, N_y, N_z (normal vector), and u, v (texture coordinate) information.

Because JPEG 2000 uses wavelet transform to compress the image into multiple quality layers (set to 12), in the process of transmission, we can decide the quality layer to be transmitted. This mechanism allows us to achieve the effect of progressive transmission and the users can obtain the multi-resolution images. To achieve the view-dependent issue, the concept of ROI of JPEG 2000 is adopted. The geometry image is first divided into several blocks (with the size of 8×8), and then the blocks which are close to the user's viewpoint can be selected to transmit first. When the viewpoint changed, we can easily recalculate the new blocks to be transmitted according to the new viewpoint.

D. JPEG 2000 Image Transmission

To support the view-dependent and multi-resolution schemes, we have to modify the transmission sequence of an existed JPIP server. Although our JPIP server can support both of the view-dependent and multi-resolution schemes at the same time, to make the explanation clear, we first describe the view-dependent scheme and then explain the multi-resolution scheme (also with view-dependent) as an additional functionality.

To describe how we request the JPEG 2000 code stream from a JPIP server, we use the term of *focus window* as described in [13]. A *focus window* specifies the user's current spatial region, the image resolution, and the image layer that the user required. In this paper, we use $\mathrm{W}(l,r,c)$ to indicate a *focus window*, where $l=((x_w,y_w),(w_w,h_w))$ is the location (x_w,y_w) and the window size (w_w,h_w) of the *focus window*, r is the image resolution level, and c is the index of the image layer.

In the view-dependent scheme, since the image resolution is fixed during the entire transmission process and we assume there is only one image layer for easy explanation, we hence can only use $\mathrm{W}(l)$ to indicate the *focus window*. To support the view-dependent scheme (with only one layer and full resolution image), our system is able to find the pixel (in other words, a vertex of the 3D model) which the user is viewing, so that the currently viewing part (*focus window*) can be transmitted first. Then, the following sequence of the *focus windows* are defined as shown in Fig. 5 which will be described later.

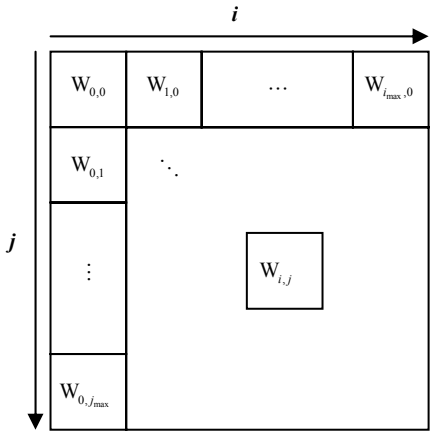

Fig. 4. To transmit a JPEG 2000 image with the view-dependent and multi-resolution schemes, we divide the image into several image blocks according to their location and resolution level.

As shown in Fig. 4, we first divide an image into several blocks. Each block size is $w_b\times h_b$, where w_b and h_b are the width and height of a block, and this division is applied for every resolution level and every image layer. Hence, the *focus window* of a block (i,j) is defined as $\mathrm{W}_{i,j}(l_{i,j})$, where $i=0..i_{\max}$ $(i_{\max}=w/w_b)$ and $j=0...j_{\max}$ $(j_{\max}=h/h_b)$ denote the block indices along the column and row directions, w and h are the width and height of the whole image, and $l_{i,j}=((i\Box w_b,j\Box h_b),(w_b,h_b))$.

When the transmission starts, suppose that the pixel (the vertex of the 3D model) which the user is viewing falls on $\mathrm{W}_{i,j}$, as depicted in Fig. 5 (a), then the sequence of the *focus windows* is decided in a swirl fashion. Hence, as shown in Fig.

5 (a), the sequence of the *focus windows* starts from $W_{i,j}$, and the following *focus windows* are $W_{i,j+1}$, $W_{i+1,j+1}$, $W_{i+1,j}$, $W_{i+1,j-1}$,... , etc. After the *focus windows* (blocks) at the lowest resolution level have been transmitted, the blocks at the next resolution level will be transmitted as the same sequence. While the user changes his/her viewpoint, the sequence will be recomputed in the same manner to fulfill the demand of the user. Therefore, during the transmission, the user's viewing part will be transmitted first and the visual quality will become better and better while more blocks at finer resolution levels are transmitted.

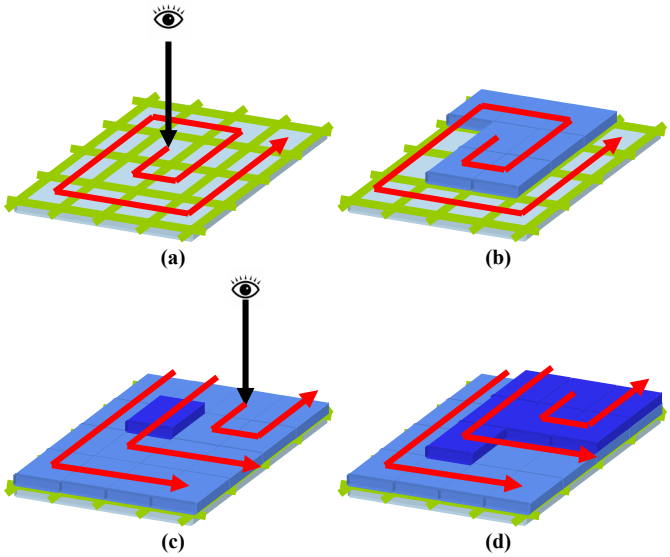

Fig. 5. (a) Given the image with lowest resolution, we find a block which contains the user's viewing pixel and decide the sequence of the following blocks. (b) After the blocks at the lowest level have been transmitted, the blocks at the next level will be transmitted as the same sequence. (c) & (d) If the user changes the viewpoint, the sequence will be recomputed.

In the view-dependent and multi-resolution scheme, similarly, we use $W(l,r)$ to specify a *focus window*, since we also assume there is only one image layer for easy explanation. As described in the view-dependent scheme, we also divide the image into several blocks for each resolution level. The size of each block is the same for all resolution levels. Moreover, the images at different resolution levels are stored as a pyramid manner, since the pixel at lower resolution level is generated by averaging four pixels at higher resolution level. If the pixels in the image are used to represent the vertices of a 3D model, the pixel averaging works like the vertex grouping. Hence, the *focus window* of a block (i,j) at resolution level $r = 0...r_{\max}$ is defined as $W_{i,j,r}(l_{i,j},r)$, where the maximum resolution $r_{\max} = \log_2(\max(w/w_b, h/h_b))$ denotes the number of resolution levels, $i = 0...i_{\max,r}$ $(i_{\max,r} = w_r/w_b)$ and $j = 0...j_{\max,r}$ $(j_{\max,r} = h_r/h_b)$ are block indices along the column and row directions, $w_r = w/2^{(r_{\max}-r)}$ and $h_r = h/2^{(r_{\max}-r)}$ denote the image size at resolution level r, and $l_{i,j} = ((i\square w_b, j\square h_b),(w_b,h_b))$.

If we set the image and the block as the squares, i.e., $w = h$ and $w_b = h_b$, the first *focus window* should be $W_{0,0,0}(l_{0,0},0)$. That means there is only one block at the lowest resolution level, so it should be transmitted first. Otherwise, the first *focus window* should be decided by taking the view-dependent issue into consideration. The following *focus windows* are decided in a recursive manner.

In the recursive manner, given a *focus window* $W_{i,j,r}$, we can define its four child *focus windows* as $W_{2i,2j,r+1}$, $W_{2i+1,2j,r+1}$, $W_{2i,2j+1,r+1}$, and $W_{2i+1,2j+1,r+1}$. After the client receives the *focus window* $W_{i,j,r}$, suppose the user's viewing pixel always falls on the upper-left corner of $W_{i,j,r}$, the next *focus window* will be $W_{2i,2j,r+1}$ as shown in Fig. 6 (a). While the *focus window* reaches the highest resolution level $r_{\max}$, the other child *focus windows* will be transmitted like the DFS (Depth First Search) manner as shown in Fig. 6 (b).

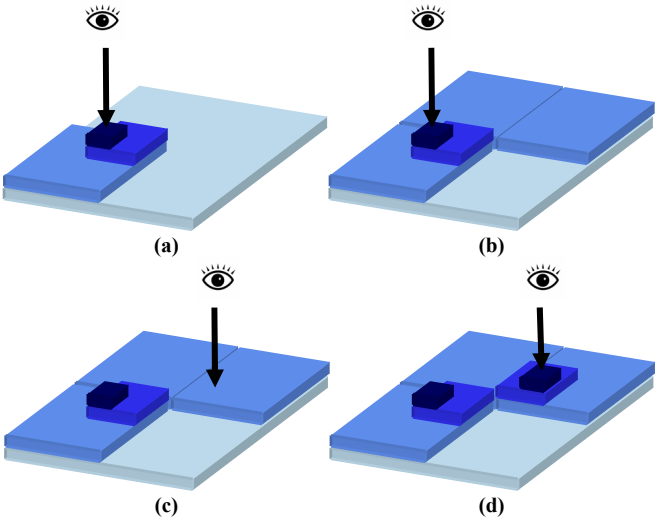

Fig. 6. (a) The sequence of the *focus windows* is decided in a recursive manner. (b) If a *focus window* reaches the highest resolution, the sequence of the following *focus windows* is decided as the DFS manner. (c) & (d) If the user changes the viewport, the sequence will be recomputed to fit user's demand.

To combine the recursive manner and the view-dependent scheme, instead of reaching the maximum resolution level $r_{\max}$, a relative resolution threshold r_ε is used. Hence, after the client receives the *focus window* $W_{i,j,r}$ the next *focus windows* are decided as the recursive manner until the *focus window* reaching the resolution level $\max(r + r_\varepsilon, r_{\max})$. Therefore, if $r_\varepsilon = r_{\max}$, the server will work as the DFS manner; if $r_\varepsilon = 1$, the server will work as the BFS (Breadth First Search) manner.

If the user changes his/her viewpoint, the new viewing information will be sent to the server and the sequence of the *focus windows* is recomputed to fit the user's demand as shown in Fig. 6 (c) and (d). Therefore, at first, an image (3D model) with fewer pixels (vertices) and lower visual quality

will be shown. As more code stream is received, the number of pixels (vertices) will increase and the visual quality will become better.

E. X3D Extension

Our new streaming method can be employed by X3D and serves as a replacement to the node `IndexedFaceSet`. The Syntax of the X3D extension is shown in Fig. 7. In the extension node, the `url` field specifies the location of the geometry image. `PosMin` and `PosMax` fields represent the bounding box of the 3D model. Similarly, `NormalMin` and `NormalMax` field specify the range of the normal vectors of the 3D model. `BlockSize` field informs the height and width of each image block separated by the server.

```
GeometryImage:X3DComposedGeometryNode {
  SFString [in,out] url [][url or urn]
  MFInt32 [in]   PosMin    [0 0 0] (-∞,∞)
  MFInt32 [in]   PosMax    [0 0 0] (-∞,∞)
  MFInt32 [in]   NormalMin [0 0 0] (-∞,∞)
  MFInt32 [in]   NormalMax [0 0 0] (-∞,∞)
  SFInt32 [in]   BlockSize [0]     [0,∞)
}
```

Fig. 7. The X3D extension node syntax of the 3D model encoded as a JPEG 2000-based mesh streaming.

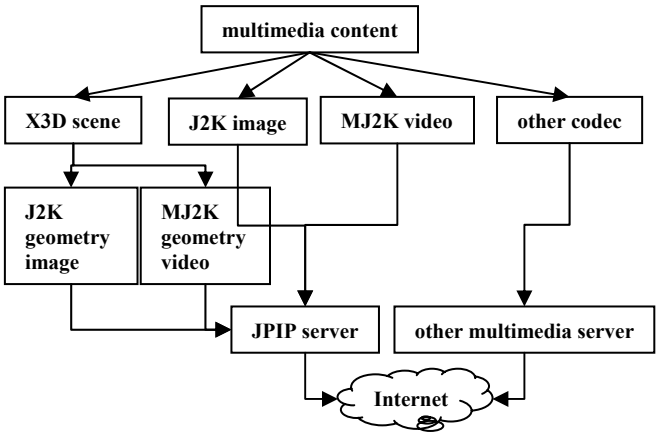

Fig. 8. The system hierarchy of the multimedia content transmission through our JPEG 2000-based mesh streaming method.

Through our X3D extension, as shown in Fig. 8, a multimedia content which contain images, videos, and 3D scenes can be encoded as JPEG 2000 images, Motion JPEG 2000 videos, and X3D scenes. Since the 3D models and 3D animation models in the X3D scenes are encoded as a JPEG 2000-based mesh streaming and a Motion JPEG 2000-based deforming meshes streaming (will be described in the next section), the 3D contents can therefore be transmitted by JPIP server with the JPEG 2000 images and the Motion JPEG 2000 videos. Of course, if the user wants to use other codec to encode the multimedia content, he/she can use other multimedia server for transmission.

F. Deforming Meshes Streaming

Since a 3D model can be encoded as a JPEG 2000 code stream, a 3D animation model which can be treated as deforming meshes can then be encoded as a Motion JPEG 2000 code stream. Hence, to convert a 3D animation model to be a Motion JPEG 2000 geometry video, we first convert each frame of the animation to be one JPEG 2000 geometry image. As described in Section B, to convert a 3D model to be a geometry image, we have to cut and parameterize it. To process a 3D animation model, if the 3D animation model is closed manifold in all frames, we first use the cut method in [5] to find the proper boundary of the model for all frames, i.e., to find a uniform cut path for all frames in order to keep the consistency. Then, we map the boundary to a square. To keep the consistency, the vertices at the four corners of the square should be the same in all frames. Finally, we parameterize the 3D models in all frames and guarantee the parameterization of all frames is consistent. Hence, the 3D models in all frames are converted to be consistent JPEG 2000 geometry images.

After converting the all frames of a 3D animation model to several JPEG 2000 geometry images, we use intraframe coding method similar to Motion JPEG 2000 [18]. Hence, a 3D animation model can be converted as a Motion JPEG 2000 geometry video, and also can be transmitted through the JPIP server.

IV. TRANSMISSION ARCHITECTURE

To transmit the JPEG 2000 encoded mesh streaming or Motion JPEG 2000 encoded deforming meshes streaming through the Internet, we provide two kinds of transmission architectures. One is the traditional client-server architecture and the other is a peer-to-peer (P2P) architecture. Since our modified JPIP server is modularized and our block-based image transmission scheme does not need to transmit one by one sequentially, our system can support simple client-server architecture or a more efficient peer-to-peer architecture

A. Client-Server Architecture

Fig. 9 shows the concept of the transmission under the client-server architecture. The further explanations are as the following:

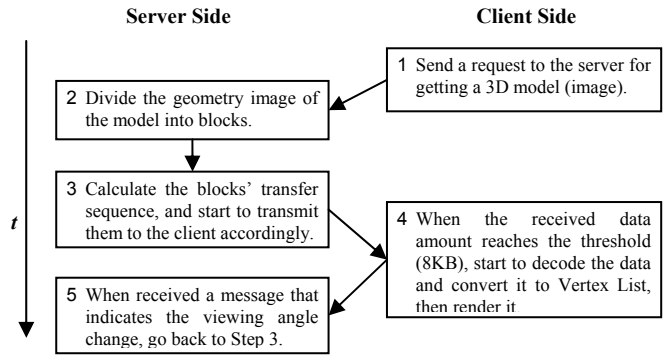

Fig. 9. The concept of the client-server transmission.

1. When the client requests the server for getting a certain 3D model, the initial viewing information will also send to the server at the same time. The viewing information is a 4×4 transformation matrix. From this matrix, we can

calculate the vertex that the user is currently viewing according to the viewpoint and the viewing direction.

2. When the server receives the transmission request and the viewing information, the server first divides the geometry image of the 3D model into several blocks. The default block size is 8×8 according to the block size used in DCT (Discrete Cosine Transform) of JPEG. Hence, for the image with resolution of 32×32, it is divided into 4×4 blocks.
3. With the number of blocks and the viewing information, the server can calculate which block is currently viewed by the user and then decide the first block to be sent.
4. After deciding the first block to be sent, we have to decide what are succeeding blocks. To transmit the blocks as near as possible to the first block, we send the blocks in a swirl fashion. The first one being the eye of the swirl, the next block in line would be the one at its immediate top, then the one next to the second in a counter-clockwise fashion. After obtaining the sequence, we can place the blocks into the sending buffer according to this sequence. In the process of transmission, the server will keep track of the block indices, so that when the viewpoint changes, there is no need to resend the blocks that has been sent already.
5. After the client receives the block, it can start to decode and place the information into an array prepared beforehand according to the block index. Squares with incomplete vertices will be set aside first. With this, we can decode while transmission and achieve the mesh streaming.
6. When the viewpoint changes, the client will resend the viewing information to the server as described in Step 1. When the server receives the message, it will stop the transmission of the current blocks, then proceed to Step 1 to re-compute the new transmission sequence and start to transmit again.

B. Peer-to-Peer Architecture

Although the JPIP server provides excellent functionality for interactive image browsing over the Internet, it cannot be exploited in a peer-to-peer network architecture directly. As described in the last subsection, the JPIP server should contain the entire code stream of a JPEG 2000 image, so it can handle all kinds of request from the client. Each node in our peer-to-peer network framework, however, does not always have the entire JPEG 2000 code stream. To get better performance in a peer-to-peer network, each node should provide any information it has already received to other neighbor nodes. Each node should start spreading code stream as soon as it receives any of them, rather than waiting the completion of downloading whole code stream.

To adapt the JPIP server for a peer-to-peer network architecture, a pre-processing is needed [7]. First, we generate a sequence of the *focus windows*, which specify the location of the image block, the image layer, and the resolution we need. These *focus windows* are then passed to a modified server program which, instead of transmitting the data pieces, stores the corresponding data pieces onto disk. These result data pieces are then suitable for peer-to-peer transmission. Since the peer-to-peer transmission can not guarantee the sequence of the transmitted blocks, the modified client program will store the received blocks and check if the blocks can pass to the higher level application or not. Although there is no entire information for the transmission content, since we encode the block numbers (i, j of the *focus window*) as a pyramid manner as described in the previous section, the block dependency can be easily checked by checking the block indices.

V. RESULT

To show the mesh streaming result, we use a Beethoven model as shown in Fig. 10 - Fig. 14. The original file size is 314KB in .obj file format and the file size of its 256×256 JPEG 2000 geometry image is 105KB. Fig. 10 shows the view-dependent result. With the locality of JPEG 2000 image, we can see the model being rendered in more parts with our viewing information as more data received by the client as time goes. Fig. 11 shows the multi-resolution result. With the multi-layer JPEG 2000 image, we can see the model being rendered in greater details as more data received by the client as time goes. Fig. 12 shows the result by combining the view-dependent and multi-resolution schemes.

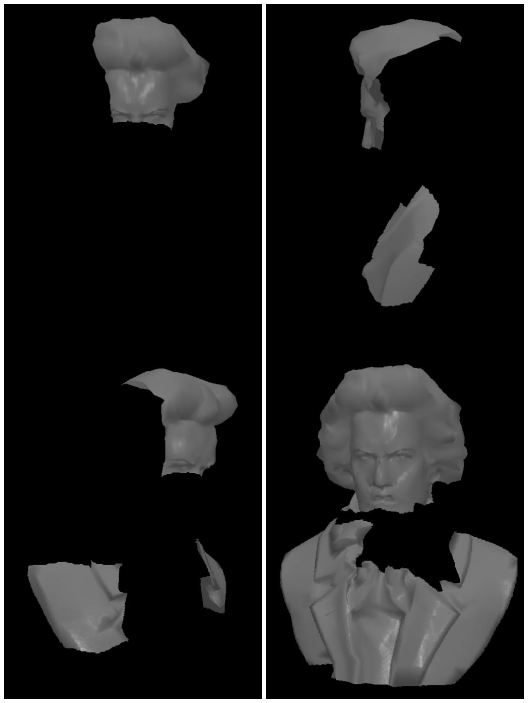

Fig. 10. The view-dependent result.

After integrating our JPEG 2000-based geometry image into X3D, our method can have more extensions and variations. Fig. 13 and Fig. 14 show the example code and the result of combining our JPEG 2000-based geometry image into an X3D library. In Fig. 14, the image at the left side has

additional material information; the one at the right side has texture information. Fig. 15 shows some frames of the animation result. The dog animation model is transmitted as a Motion JPEG 2000 code stream.

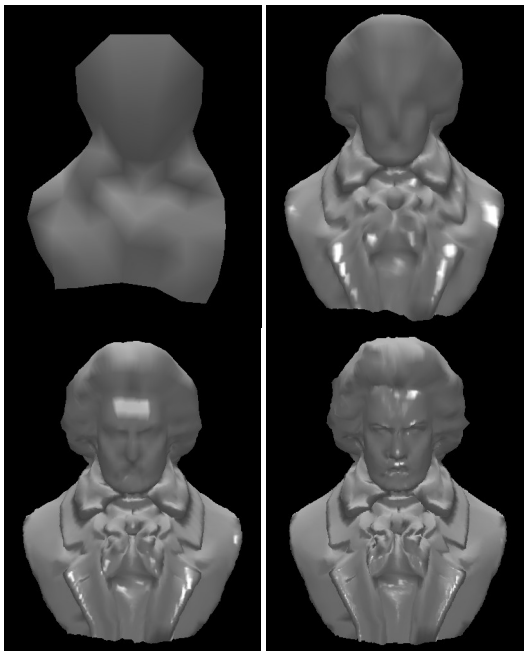

Fig. 11. The multi-resolution result

Fig. 12. The view-dependent and multi-resolution result

Fig. 16 shows an X3D scene composes of three ball models, two triceratops models, and one Isis model. The six models are all encoded as a JPEG 2000-based mesh streaming. To perform this mesh streaming, we first convert the six models into six JPEG 2000-based geometry images. During the transmission, since the JPIP server will divided the geometry images into several blocks. By using the viewpoint information, all of the blocks are reshuffled to achieve a good view-dependent and multi-resolution result. Hence, there is only one code stream to transmit the whole X3D scene.

```
<Shape>
  <Appearance>
    <Material diffuseColor=".7 0.9 0.7"
              emissiveColor="0.1 0.1 0.1"
              specularColor=".9 .9 0.9"
              shininess="1.0"/>
    <ImageTexture url="tex03.jpg"/>
  </Appearance>
  <GeometryImage url="Beethoven_256.j2c"
                 PosMin="-46.49 -57.88 -24.55"
                 PosMax="46.49 57.88 24.55"
                 NormalMin="-0.99 -0.99 -0.85"
                 NormalMax="0.99 0.98 0.99"
                 BlockSize="8"/>
</Shape>
```

Fig. 13. The example code of the X3D extension.

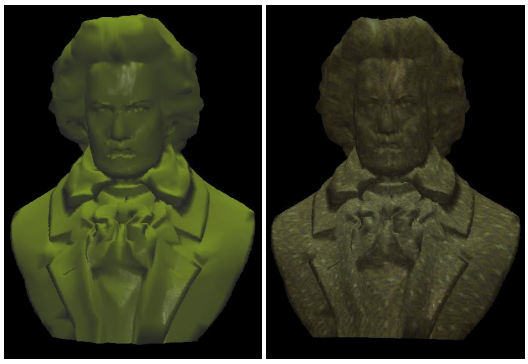

Fig. 14. The X3D extension result.

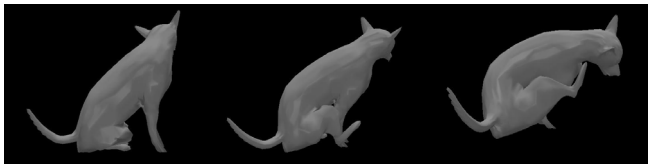

Fig. 15. The animation result.

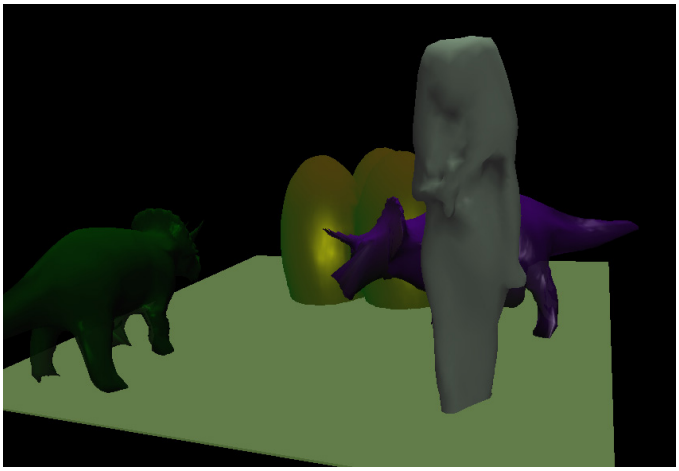

Fig. 16. An X3D scene composes of six 3D models which are transmitted as a JPEG 2000-based mesh streaming.

VI. Conclusion and Future Work

In this paper, we propose a new 3D model streaming method which uses a JPEG 2000 image to store all information of a 3D model. Since the 3D model is encoded as

a JPEG 2000 image, we can compress the geometry image without data loss (lossless mode) or compress it in high compression rate with a little bit data loss (lossy mode). With JPEG 2000's support for progressive compression and decompression, we can progressively transmit and render a 3D model to reduce the user waiting time. Besides the progressive transmission, the view-dependent issue is also taken into consideration. Furthermore, a 3D animation model can also be encoded as a Motion JPEG 2000 video.

Hence, by using our modified JPIP server, we can achieve mesh streaming, deforming meshes streaming, JPEG 2000 image streaming, and Motion JPEG 2000 video streaming at the same time with only one server. Since we divided the JPEG 2000 image into several blocks, our JPIP server can support not only the client-server mode, but also the peer-to-peer transmission. Besides integrating our mesh streaming method with the JPIP server, our method is also integrated with X3D.

However, even the lossless mode of JPEG 2000 can compress the data without loosing the quality; there are still some inherent problems. To convert the 3D model to a geometry image, we have to cut the model first to find the boundary and this actually may cause the distortion or data loss. Moreover, when using Floater's method for surface parameterization, there is a tendency to loss the information at sharp features of a 3D model. Hence, the cut and surface parameterization method should be enhanced to lessen the data loss.

Most 3D models produced by range scanners are able to be parameterized and converted into geometry image. However, many 3D models made by artist are not easy to be parameterized, since they are not closed manifold or 2-manifold with boundary meshes. As a result, a new algorithm for converting none-manifold meshes into a geometry image or geometry images is also needed. In the network virtual environments such as on-line games and virtual tour system, there are many 3D models to be rendered at a time. One of our future works is to extend our current system to a 3D scene streaming system for more practical applications.

ACKNOWLEDGMENT

The authors would like to thank Shun-Yun Hu for his great support on peer-to-peer architecture.

REFERENCES

[1] H. M. Briceño, P. V. Sander, L. McMillan, S. Gortler, and H. Hoppe, "Geometry videos: a new representation for 3d animations," in *Proceedings ACM SIGGRAPH/Eurographics 2003 Symposium on Computer Animation*, 2003, pp. 136-146.

[2] B.-Y. Chen and T. Nishita, "Multiresolution streaming mesh with shape preserving and qos-like controlling," in *ACM Web 3D 2002 Conference Proceedings*, 2002, pp. 35-42.

[3] S. Deshpande and W. Zeng, "Scalable streaming of jpeg2000 images using hypertext transfer protocol," in *ACM Multimedia 2001 Conference Proceedings*, 2001, pp. 372-281.

[4] M. Floater, "Parametrization and smooth approximation of surface triangulations," *Computer-Aided Geometric Design*, vol. 14, no. 3, 1997, pp. 231-250.

[5] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," *ACM Transactions on Graphics (SIGGRAPH 2002 Conference Proceedings)*, vol. 21, no. 3, 2002, pp. 355-361.

[6] H. Hoppe, "Progressive meshes," in *ACM SIGGRAPH 1996 Conference Proceedings*, 1996, pp. 99-108.

[7] S.-Y. Hu, "A case for peer-to-peer 3d streaming," in *ACM Web 3D 2006 Conference Proceedings*, 2006, pp. 57-63.

[8] F.-C. Huang, B.-Y. Chen, and Y.-Y. Chuang, "Progressive deforming meshes based on deformation oriented decimation and dynamic connectivity updating," in *Proceedings ACM SIGGRAPH/Eurographics 2006 Symposium on Computer Animation*, 2006, pp. 53-62.

[9] M. Isenburg and P. Lindstrom, "Streaming meshes," in *IEEE Visualization 2005 Conference Proceedings*, 2005, 231-238.

[10] S. Kircher and M. Garland, "Progressive multiresolution meshes for deforming surfaces," in *Proceedings ACM SIGGRAPH/Eurographics 2005 Symposium on Computer Animation*, 2005, pp. 191-200.

[11] N.-H. Lin, T.-H. Huang, and B.-Y. Chen, "3D model streaming based on a jpeg 2000 image," in *Proceedings of IEEE 2007 International Conference on Consumer Electronics*, 2007.

[12] S. Rusinkiewicz and M. Levoy, "Streaming qsplat: a viewer for networked visualization of large, dense models," in *ACM Interactive 3D 2001 Conference Proceedings*, 2001, pp. 63-69.

[13] D. Taubman and R. Prandolini, "Architecture, philosophy, and performance of jpip: internet protocol standard for jpeg2000," in *Proceedings of 2003 International Symposium on Visual Communications and Image Processing*, 2003, pp. 791-805.

[14] E. Teler and D. Lischinski, "Streaming of complex 3d scenes for remote walkthroughs," *Computer Graphics Forum (Eurographics 2001 Conference Proceedings)*, vol. 20, no. 3, 2001, pp. 17-25.

[15] S. Yang, C.-S. Kim, and C.-C. J. Kuo, "A progressive view-dependent technique for interactive 3-d mesh transmission," *IEEE Transactions on Circuit and System for Video Technology*, vol. 14, no. 11, 2004, pp. 1249-1264.

[16] *The Virtual Reality Modeling Language -- Part 1: Functional specification and UTF-8 encoding*, ISO/IEC 14772-1:1997, Web3D Consortium, Inc., 2003.

[17] *JPEG 2000 image coding system: Core coding system,* ISO/IEC 15444-1:2004, JPEG Committee, 2004.

[18] *JPEG 2000 image coding system -- Part 3: Motion JPEG 2000,* ISO/IEC 15444-3:2002, JPEG Committee, 2004.

[19] *JPEG 2000 image coding system: Interactivity tools, APIs and protocols*, ISO/IEC 15444-9:2005, JPEG Committee, 2005.

[20] *Extensible 3D (X3D) -- Part 1: Architecture and base components*, ISO/IEC 19775-1:2004, Web3D Consortium, Inc., 2006.

Nein-Hsien Lin received the B.S. degree in Mathematics from the National Taiwan University, Taipei, in 2004. He is currently a M.S. student in the Graduate Institute of Networking and Multimedia of the National Taiwan University, since 2004. His research interests includes Geometric Modeling and Web Graphics.

Ting-Hao Huang received the B.S. degree in Computer Science from the National Tsing-Hua University, Hsinchu, Taiwan, in 2002. He is currently a M.S. student in the Department of Computer Science and Information Engineering of the National Taiwan University, Taipei, since 2005. His research interests include Geometric Modeling, Mobile Graphics, and Virtual Reality.

Bing-Yu Chen (M'00) received the B.S. and M.S. degrees in Computer Science and Information Engineering from the National Taiwan University, Taipei, in 1995 and 1997, respectively, and received the Ph.D. degree in Information Science from the University of Tokyo, Japan, in 2003. He is currently an assistant professor in the Department of Information Management and the Graduate Institute of Networking and Multimedia of the National Taiwan University since 2003. He is a member of IEEE, ACM, ACM SIGGRAPH, IEICE, and Eurographics.