EUROGRAPHICS '97 / D. Fellner and L. Szirmay-Kalos
(Guest Editors)

Volume 16, (1997), Number 3

Selective Pixel Transmission for Navigating in Remote Virtual Environments

Yair Mann and Daniel Cohen-Or

Computer Science Department
School of Mathematical Sciences
Tel-Aviv University, Ramat-Aviv 69978, Israel
Email: daniel@math.tau.ac.il

Abstract

This paper presents a technique to improve the performance of a walkthrough in remote virtual environments, where a scene is rendered jointly by the server and the client, in order to reduce the network requirements as much as possible. The client generates novel views by extrapolating a reference view based on the locally available geometric model, while the server transmits data necessary to prevent an accumulation of errors. Within this concept, we show that by transmitting only a selected subset of pixels, the quality of the extrapolated views can be improved while requiring less bandwidth. We focus on the selection process in which the visibility gaps between the reference view and novel view are detected, packed and transmitted compressed to the client.

Keywords: virtual reality, virtual environment, image-based rendering

1. Introduction

With the increasing availability of texture mapping hardware, virtual environments can be represented by a relatively small number of textured polygons. The idea is that fine, detailed and complex geometries can be simplified into texture-based representations [2, 3, 7, 14, 15]. Texture-based rendering is similar to image-based rendering where images are used as basic primitives for generating new images [4, 5, 9, 13].

Both texture-based and image-based rendering aim at accelerating the rendering time of complex and realistic scenes. However, with the increasing popularity of the Web and recent advances in computation power, the network bandwidth and transmission latency become critical bottlenecks. Walkthrough Web-systems in which the virtual environment representation is dominated by textures are still extremely limited. Downloading the virtual environment from the server into the client to be rendered locally at the client's workstation is too slow if large textures are needed to be transferred quickly. The download-and-play paradigm fails to work interactively in remote, large and complex environments.

In [6] a new paradigm for interaction with remote virtual environments is presented. Here a scene is rendered jointly by the server and the client, in order to reduce the network requirements as much as possible. The assumption is that the server is a high-end graphics workstation and the client is a PC-based workstation. The client extrapolates novel views based on a textureless model and one reference view. Since the client images include increasing errors, the server computes the difference image between the high quality image and the client's extrapolated view, and transmits the compressed difference image to the client. The client uncompresses the difference image and adds it to the extrapolated view to yield a new reference view. The main advantage of the technique is that the difference image does not need to be transmitted in

every frame and the client can generate several views autonomously.

The role of the difference image is to avoid the accumulation of errors in the client's views as the reference view becomes quite distant from the current view. However, it is not obvious that the difference image is the best *correction data* to be transmitted to the client in the sense of the tradeoff between the bandwidth requirements and the image quality. Indeed, as presented in this paper, instead of transmitting the entire difference image, only a *selected* small number of pixels can be transmitted to perform corrections at the most significant areas. The *visibility gaps* between the reference view and novel view are detected, *packed* and transmitted compressed to the client.

In Section 2 we briefly review the relevant background and give an overview of the model-based view extrapolation method. Section 3 describes the visibility gap problem and the concept of *selective transmission*. Section 4 describes the packing algorithm of the selected pixels. Section 5 explains how the client uses the transmitted pixels to improve the extrapolated views. Results are presented in Section 6 and conclusions and suggestions for further research are given in Section 7.

2. Model-based view extrapolation

Standard video compression techniques [10] were developed for natural scenes. For a synthetic scene the available model can assist in computing the optical flow [17] or in compressing the frame-to-frame differences [1]. Instead of transmitting a synthetic animation it is possible to render it on-the-fly at both ends of the communication. The rendering task can be partitioned between the server and client [11]. Assuming the server is a high-end graphics workstation it can render high and low quality images, and transmit their difference compressed. The client needs to render only the low quality images and to add the transmitted image. It was shown that the overall compression ratio is better than conventional techniques. However, this technique requires the transmission of difference images in every frame.

In [6] another strategy is presented for the partition of the rendering task in which the client is able to generate several frames autonomously. Thus, the transmission of the difference images is not required in every frame, offering a significant overall bitrate reduction. In the proposed method the client generates a sequence of frames by *extrapolating* a reference view, based on the model. In other words, it renders a model by applying a perspective texture mapping to the reference view. It is assumed that the relevant portions (i.e., visible parts) are transmitted and are available at the client. From time-to-time the server renders both the exact novel view and the client-extrapolated view, subtracts them and transmits the compressed difference image to the client. The client receives the difference image and corrects the extrapolated view to produce a new updated reference view. As long as the reference view is close enough to the current view, the quality of the extrapolated view is satisfactory. Figure 1 shows an example of a reference view (a), an extrapolated view (b), and the difference image between the extrapolated view and the exact view (c).

With the view-extrapolation method it is possible to navigate in remote virtual environments through low bandwidth networks. In [6] it was shown that it is possible to walk through a virtual museum with a difference image transmission of less than once in every ten frames. It should be emphasized that the walkthrough does not require downloading the textures of the paintings, but only a single full frame view of the initial view and an online transmission of compressed difference images. Further detail can be found in [6].

3. The visibility gap

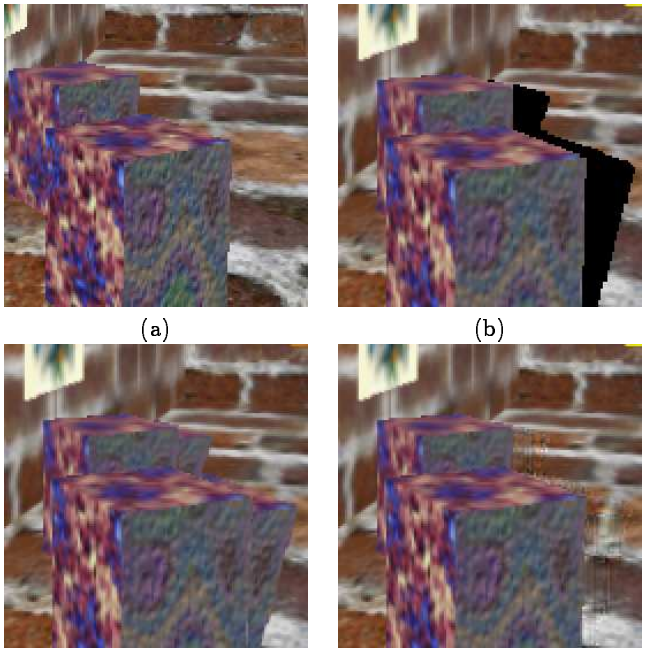
The difference image can be regarded as the error between the exact view and the extrapolated view. Observing the difference image it can be seen that the errors are not distributed evenly over the image, but that there are some areas where the extrapolation fails to texture the scene. The most severe errors are at areas which are visible from the novel view but hidden in the reference view. We call these areas the *visibility gap* which is demonstrated in Figure 2. When the camera moves away from the reference view, previously hidden parts become visible. During animation these gaps slowly grow and disappear whenever a new reference view is constructed. This temporal artifact is noticeable at depth discontinuities.

Increasing the reference update rate reduces the artifact, but on the other hand, increases the bitrate. The size of the compressed difference image is not a linear function of the distance between the reference view and the current view. Assuming reasonable walkthrough trajectories, using the JPEG standard compression, the difference images D_1 to D_k are compressed to an almost constant size, where D_j denotes the difference image j frames away from the reference image. Even if the camera does not really move, the difference image is not empty due to the inherent filtering of the texture mapping. This implies that it is not cost effective to transmit difference images more frequently to reduce temporal aliasing.

As an alternative, only a subset of the difference

(a) (b) (c)

Figure 1: *(a) the reference view. (b) the extrapolated view. (c) the difference image.*



(a) (b) (c) (d)

Figure 2: *(a) the reference view. (b) the visibility gap is filled with black pixels. (c) the gap extrapolated from the reference view. (d) the extrapolated view with the (selected) texture seen from the novel view.*

image pixels can be transmitted. Since the model is known it is possible to detect the visibility gaps and define the selected subset. Only the difference between the corresponding selected pixels needs to be transmitted, to "close" the visibility gaps. The selected pixels are received by the client who adds them to the reference view (this is elaborated in Section 5).

Since the model is known, the visibility gaps can be detected while mapping the pixels from the novel view back to the reference view. Let p be a given pixel in the novel view and q its corresponding pixel in the reference view. Whenever p belongs to the gap it disagrees with q on the depth value. The depth values of p and q can be calculated as part of the scanline backmapping. If their depth values disagree p is "selected". It is also possible to compare a unique ID index associated with each polygon. The IDs of p and q must match whenever p is visible in the reference view (see also [18]).

The client maintains only a partial subset of model polygons, and the server transmits polygons to the client according to the camera position. The detection of new visible parts can assist, as a byproduct, the transmission of new visible polygons. By rendering a

model with polygon IDs it is possible to collect the IDs from the frame buffer and maintain a dynamic list of the visible polygons. Let l_r be the reference visible list and l_n the novel visible list. Then if l_r and l_n are sorted, the subtraction lists $A = l_r - l_n$ and $B = l_n - l_r$ can be rapidly calculated. By transmitting A and B the client can maintain its dynamic list of visible polygons. For many scenes this saves much client work and accelerates client rendering time (see for example [8]).

4. The packing algorithm

Once the visibility gap has been detected and the selected subset of pixels has been marked, they need to be transmitted to the client. The selected pixels usually form several groups, scattered in the image (see Figure 2). By compressing the image which includes only the selected pixels, rather than the entire difference image, the compression rate is on average three times higher (3.6K vs. 1.1K). However, by first *packing* the selected pixels into a compact block of pixels, the compression rate is significantly further improved. For example, the colored pixels in Figure 3 have been packed into a 32-width block which is encoded by JPEG into 903 bytes. The unpacked image is encoded by JPEG into 2228 bytes. However, JPEG contains a header of 639 bytes which is not necessary in our application; excluding the header the packed pixels are compressed into 264 bytes and the unpacked image into 1589 bytes. That is, a six times better compression rate.

The compression rate of the packed block is higher because JPEG, like many other compression techniques, is block-based. The unpacked image contains many empty blocks which are compressed to a non zero size, and non empty blocks which need to be compressed. The packed block has dimensions which are a multiplication of the JPEG standard block size (i.e., 8x8 or 16x16). It should be noted that since the server and the client both know the selected pixels, the packing block does not need to include a header or a directory. In general, the packing mechanism is effective as a compact representation of the pixel colors only if their location is known to the encoder and the decoder.

Object-space optimal compaction is known to be NP-complete [16]. In image-space simple linear packing can be quite effective. Just scanning the image and storing all non empty pixels in a linear order folded into a packing block of width, say 32, can be a reasonable packing. However, with a little more effort, better packing can be achieved. A good packing algorithm preserves the pixel coherence as much as possible, assuming that the inherent pixel coherence improves the compression. Another criterion for successful packing is the percentage of empty space in the packing block. The linear packing is indeed compact since it fills the entire packing block modulo the last block column.

To exploit the inherent texture coherence we treat the selected pixels as a collection of groups. The visibility gaps formed at depth discontinuities, usually result in strips along polygon edges (see Figure 2). Therefore, the selected pixels are partitioned into strips. The width of each strip is completed to a power of 2 by adding pixels along its length. This excess of pixels is ignored by the client's unpacking algorithm. The image is scanned in image order and the selected pixels along the row are divided into spans. The spans of adjacent rows are aligned to form perfectly vertical or horizontal strips. The strips are sorted by size and then packed into a rectangular packing block with a high utilization of area. This packing preserves most of the pixel coherency, and may theoretically waste up to 50% of the packing area, due to the completion of the strip's widths to powers of 2 (for more detail see [12]). Figure 3(a) illustrates the results of the packing, where the selected pixels are colored in pseudo colors. Note that the packing block (on the lower left side of Figure 3(a)) is not fully occupied and part of its area is left empty with black pixels. In spite of this redundancy, it better preserves the original pixel-to-pixel and row-to-row spatial coherence. This coherent packing is more successful since JPEG is a lossy compression and, therefor, the decompressed pixels are better recovered and the image quality is improved. Unpacking is done by the client with an algorithm identical to the packing algorithm except that the read and write instructions are switched.

5. Closing the gaps

The selected pixels received by the client are used to texture those parts of the scene which are visible from the current client's viewing direction and not from the reference view. In other words, the transmitted pixels can be superimposed over the client-extrapolated view (see Figure 2) to reduce its error, that is, to close or fill the visibility gaps. Since the selected pixels are not transmitted on every frame, the following extrapolated views need to be corrected based on an old set of selected pixels. Instead, the selected pixels can be transformed to the reference viewing direction. Note that the selected pixels cannot be combined in the reference view since they would overwrite other pixels (the reference view contains no gaps). Thus, we use a *back-view* to store the selected pixels transformed to the reference viewing direction. During the generation of the extrapolated view the visibility of each pixel is tested. If the pixel is in the visibility gap it is

(a) (b)

Figure 3: *(a) the gap with pseudo colors; packed in the bottom left. (b) the textured gap. (c) the gap transformed to the reference viewing direction.*

backmapped to the back-view; otherwise to the reference front-view. Figure 3(b) shows the unpacked selected pixels as generated by the server and received by the client. Figure 3(c) shows the back-view which includes the selected pixels after they have been transformed to the reference viewing direction.

The bandwidth required to generate a fresh back-view is much less than the bandwidth to update the (front) reference view. Nevertheless, it is not cost effective to update the back-view in every frame. We have found that updating the back-view every five frames and the front-view every 20 frames gives good results in the sense of the image quality versus the bandwidth requirements.

6. Results

Several navigation sequences were generated to test the proposed method. It should be mentioned that the results are data dependent. The scenes (see for example Figure 1)) are virtual museums with textured walls and several paintings with some objects to cause occlusions. Each painting is a texture of over 550K bytes. For example, Escher's "Space birds" is 1083K Bytes and "Zeit" by Salvador Dali is 779K Bytes. The sum of the texture's size is up to about 10 Mega bytes. In the navigation the user advances along the hall and renders 241 views. The navigation sequences are available at http://www.math.tau.ac.il/~daniel/SPT/spt.html {a full screen is recommended for best display}).

In these sequences the reference views are updated every 15 frames while the others are extrapolated. The back-view (with the transformed textured visibility gap) is updated every five frames. The quality of these sequences is higher than those which were generated without the back-view and with a reference view updated every 10 frames. Moreover, as shown in Table 1, the bandwidth requirement is less. These results are compared to the MPEG sequences of the exact view.

The difference images and the packed blocks were compressed using JPEG. A typical difference image (256x256, 65K Bytes) is compressed to 3.5K Bytes. For the 241 frames of the "birds" sequences, a 10th frame reference view update results in a total of 85K (24 difference images). A 15th frame reference view update (15 difference images - 56K) and a fifth frame back-view update (32 unpacked frames - 32K) result in a total of 88K Bytes. However, the 32 back-view frames can be packed into 1K only. We get similar results for the other sequences (Table 1).

The client-extrapolation algorithm is the expensive part especially because the pixel-by-pixel ID check is hard to implement with OpenGl. Since the client is assumed to be a PC-based machine with no hardware texture mapping, we have implemented our own rasterization with a z-buffer, perspective texture mapping, ID check, and access to the back-view. Currently, our implementation, without optimization renders a 256x256 frame on a Pentium 100 in about 7 frames per second.

7. Conclusions

We have presented a method for navigating in remote virtual environments where the server and the client render the scene cooperatively. The main purpose is to trade off computation for bandwidth. In this paper we have shown how a selective transmission can further reduce bandwidth requirements and improve image quality. The server has to detect the visibility gap, pack the selected pixels, compress and transmit them more frequently. The client is required to unpack the selected pixels and transform them to back-view. Moreover, in every frame the client needs to map the pixels from either the front-view or the back-view which means an ID comparison per pixel. However, this has been proved to be cost effective in terms of bandwidth requirements. This is justified as in the foreseeable future, client low-end machines will become more and more powerful each year, while the cost of bandwidth will remain high.

Table 1: *The size of 241 frames from the virtual museum navigations. In "MBEX(10)" the reference view is updated every 10 frames. In the "MBEX(15) + Back(5)" column the reference view is updated every 15 frames and the back-view every 5 frames.*

animation	raw	MPEG	MBEX (10)	MBEX(15) + Back(5))
Zeit	47M	1536K	71K	50K
Race	47M	1589K	86K	61K
Birds	47M	852K	85K	57
Fish	47M	956K	77K	56K

We are presently considering trading more computation for bandwidth, by better defining the meaning of the selected pixels. In general terms, the idea is to transmit exactly those pixels that are necessary for the extrapolation. Except for those pixels in the visibility gap the extrapolation fails where the source texture is undersampled. By partitioning the areas which are dominated by a rigid transformation and those which scale above some threshold, we can define the fine selected subset of pixels to be transmitted.

Acknowledgments

This work was supported in part by a grant from the Ministry Of Science Israel, and the French Ministry of Research and Technology (AFIRST).

References

1. Maneesh Agrawala, Andrew Beers and Navin Chaddha. Model-Based Motion Estimation for Synthetic Animations. *Proc. ACM Multimedia '95.*
2. D. G. Aliaga. Visualization of Complex Models Using Dynamic Texture-based Simplification. *Proceedings of Visualization 96*, 101-106, October 1996.
3. A.C. Beers, M. Agrawala, and N. Chaddha. Rendering from Compressed Textures. *Computer Graphics (SIGGRAPH '96 Proceedings),* August 1996.
4. S. Eric Chen and L. Williams. View Interpolation for Image Synthesis. *Computer Graphics (SIGGRAPH '93 Proceedings),* 279-288, August 1993.
5. S. Eric Chen. QuicktimeVR - an Image-based Approach to Virtual Environment Navigation. *Computer Graphics (SIGGRAPH '95 Proceeding),* 29-38, August 1995.
6. D. Cohen-Or. Model-Based View-Extrapolation for Interactive VR Web-Systems, *Computer Graphics International*, 1997.
7. P.E. Debevec, C.J. Taylor and J. Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry - and Image-Based Approach. *Computer Graphics (SIGGRAPH '96 Proceedings),*
8. J.H, S. Jung and K.Y. Wohn, Exploiting Temporally Coherent Visibility for Accelerated Walkthoughs. *to appear in Computers and Graphics.*
9. T. Kaneko and S. Okamoto. View Interpolation with Range Data for Navigation Applications. *Computer Graphics International,* 90-95, June 1996.
10. D. LeGall, MPEG: A Video Compression Standard for Multimedia Applications. in *Communications of the ACM*, Vol. 34, No. 4, April, 1991, pp. 46-58.
11. Marc Levoy. Polygon-Assisted JPEG and MPEG Compression of Synthetic Images. *Computer Graphics (SIGGRAPH '95 Proceeding),* 21-28, August 1995.
12. Yair Mann. Low-bandwidth Navigation in Remote Virtual Environments. *Master's thesis, Computer Science Department, Tel-Aviv University*, 1996.
13. L. McMillan and G. Bishop, Plenoptic modeling: An Image-based Rendering System. *Computer Graphics (SIGGRAPH '95 Proceeding),* 39-46, August 1995.
14. G. Schauffler and W. Sturzlinger A Three Dimensional Image Cache for Virtual Reality *Eurographics '96, Computer Graphics Forum* Volume 15(3), 227-235, 1996
15. J. Shade, D. Lischinski, D.H. Salesin, J. Snyder and T. DeRose. Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments. *Computer Graphics (SIGGRAPH '96 Proceedings),*
16. P.E. Sweeney and E.R. Paternoster. Cutting and packing problems: A categorized application-oriented research bibliography. *Journal of Opl Res Soc.,* 43(7), 691-706, 1992.
17. D.S. Wallach, S. Kunapalli and M.F. Cohen. Accelerated MPEG Compression of Dynamic Polygonal Scenes. *Computer Graphics (SIGGRAPH '94 Proceedings),* 193-197, July 1994.
18. R. Yagel and Z. Shi. Accelerating Volume Animation by Space-Leaping. *Proceedings of Visualization'93*, October 1993, pp. 62-69.