A Server-based Interactive Remote Walkthrough

Daniel Cohen-Or[1,2], Yuval Noimark[3], and Tali Zvi[1]

[1] Enbaya Ltd.
[2] Computer Science Department, Tel-Aviv University
[3] IBM Research Lab in Haifa

Abstract. This paper presents a server-based remote walkthrough system. The client is assumed to be a thin client, like a handset or a mobile device, with no strong processor but with some embedded video chip. The server holds the large environment, generates the frames, encodes and transmits them to the client. The encoded frames are transmitted as a video stream to the client, which then decodes the stream and displays it. We show how the computer generated frames can be efficiently encoded using layering techniques to yield a lighter stream, which enables its transmission over narrow bandwidth channels and minimizes the communication latency. To enable the interactivity of the system, the rendering engine generates the frames in real-time according to the client input, and feeds the frames to an accelerated video encoder based on the available optical flow.

1 Introduction

With the increasing availability of the Web and the advances in graphics hardware, the network bandwidth becomes a critical bottleneck. Remote walkthrough in complex computer-generated environments is an emerging challenge in computer graphics, since the entire model cannot be downloaded from the server to the client. Even dynamically downloading relevant portions of the model on demand with respect to the client position is not possible as it may consist of a prohibitive number of polygons or textures. There are different techniques that simplify the representation of the objects to accelerate the walkthrough [6, 14, 18, 17, 3]. Others have focused on Web-based remote walkthrough [13, 7, 20]. Hybrid-rendering methods, where the client and the server both commonly participate in the rendering of the frames are a possible solution that various researchers have taken [11, 15, 23, 5]. Streaming technologies can alleviate the problem by reducing the bandwidth requirement, but require extensive computational power from the client.

For most applications, the server should effectively serve as many clients as possible simultaneously. Thus, it is desirable to shift most of the computation to the client, and to ease the server load. However, there are scenarios where the client is thin and most or all of the computation load should be on the server. This is especially relevant to commercial handsets (e.g. cellphone or PDA) that contain a weak processor for economic reasons. Typically, a handset is built with

an old processor that costs much less and whose computation power cannot support graphical 3D applications. In such a case the server should generate the frames and transmit them to the thin client, which then needs only to display them. However, naively transmitting the frames is too expensive since the current available network bandwidth is extremely narrow.

The proposed solution, presented in this paper, is to use a server-based approach. The server renders the sequence and encodes the frames into a video stream. The stream is received by the client, which displays the stream using an MPEG-4 video chip. Fast online encoding is essential for real-time experience of the remote walkthrough, and therefore special techniques are needed to reduce the encoding complexity.

The rest of this paper is organized as follows: In Section 2 we give a brief overview of the system. In Section 3 we describe the MPEG-4 encoding scheme. Section 4 describes our per-layer encoding technique. The real-time encoding is presented in Section 5. We finalize this paper with some results (Section 6), conclusions and future work (Section 7).

2 System Overview

Our system is server-based, as illustrated in Figure 1. The 3D model of the scene is held on the server. Once interaction starts, the Interactive Rendering Engine renders the frames. It provides scene information to the Macro-block Analysis module for generating layering information and motion hints. The layering information is used to vary the quantization level locally over the frame (see Section 4), and motion hints are used for fast encoding (see Section 5).

An interaction back-channel is used for real-time walkthrough in the model. A Control Data back-channel is used for bandwidth adaptation and for packet loss information. The Control Data enables the Frame Encoder to retransmit a lost packet or to adapt the compression ratio and the quality to the available network bandwidth. The frames generated by the rendering engine are fed into the Frame Encoder as plain RGB frames. We choose to encode the frames with the MPEG-4 video standard since today there are already devices that provide MPEG-4 video decoding. Recently more and more companies have announced that cellphones capable of decoding MPEG-4 video will soon be available. We can also expect that in the near future, PDAs and other mobile devices will have such capabilities in hardware.

3 MPEG-4 Encoding Scheme

MPEG-4 is the new ISO/IEC multimedia standard developed by MPEG (Moving Picture Experts Group); its format designation is ISO/IEC 14496 [2]. The standard defines media objects as the primitives in the scene. These objects can be of synthetic or natural origin and can be visual or aural ones. It also defines how to multiplex and synchronize these objects to enable their transportation over the network. One of the media objects defined is MPEG-4 video [1]. As we

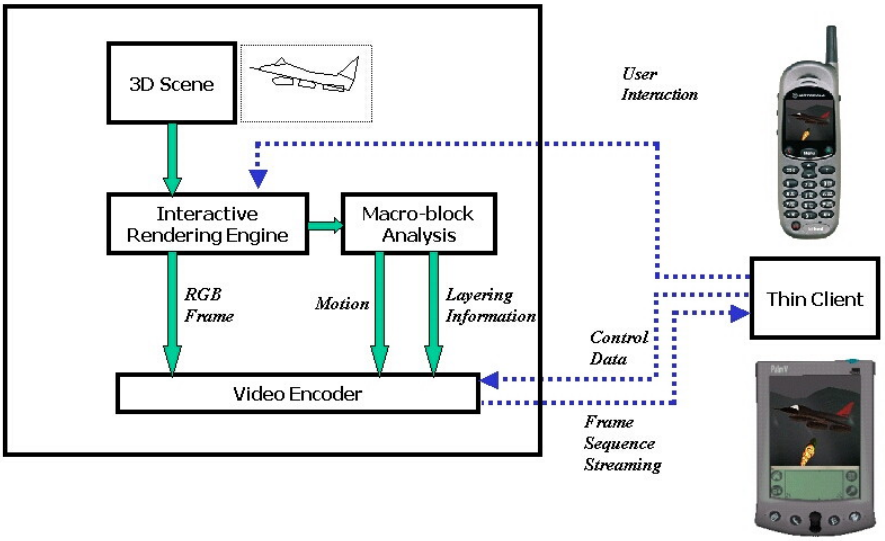


Fig. 1. The systemoverview

mentioned in the Introduction, we have implemented our Frame Encoder with the MPEG-4 video standard as more and more of today's devices are enabled to display MPEG-4 video content using on-chip decoders [16, 19, 4]. These devices can display real-time video according to the user's preferences and the device's location. However, the technique introduced here is not limited to this standard and could be applied to other block-based video compression schemes.

3.1 Frame Coding Scheme

An MPEG-4 video stream consists of three types of frames: intra frames I, predictive frames P, and bi-directional predictive frames B. The I frames are encoded independently of the other frames in the sequence, while the P and B frames are encoded using motion estimation and interpolation techniques. The P and B frames are substantially smaller than the I frames. The motion estimation of the P frames is based on preceding P or I frames. The motion estimation of B frames is based on preceding P or I frames and/or on successive P or I frames. The P or I frames used for motion estimation are referred to as reference frames.

More specifically, a frame is encoded at the macro-block (MB) level. Each frame is partitioned into macro-blocks of 16x16 pixels that are encoded independently of the other macro- blocks. An Intra macro-block (I-MB) is one whose data is fully encoded. An Inter macro-block (P-MB) is one whose data is encoded using motion estimation. The I-MB is encoded by applying DCT (Discrete Cosine Transform) to its data, quantizing the resulting coefficients, and encoding them

using variable length coding. The P-MB is first motion-estimated by searching the reference frame for the best match for the macro-block to be encoded. The co-located macro-block is the macro-block in the reference frame, which has the same relative location as the P-MB to be encoded. The search for the best match is done in an area around the co-macro-block. Once a motion vector is found, the difference between the matching area and the P-MB is DCT-transformed. The resulting coefficients are quantized and then encoded using variable length coding.

The quantization level determines the compression ratio and controls the degree of the lossy compression. Clearly there is a trade-off between the qualities of the macro-blocks and the degree of compression. The more the macro-block is compressed (i.e. higher quantizer), the more its quality is degraded. The frame types are usually defined using a predefined repeating pattern of a sequence of I, P and B frames, like IBBPBBP or IPPBIP. However, during encoding of a P frame, each macro-block can be defined as an I-MB or P-MB independently of the global definition of the entire frame. Identifying different types of encoding per macro-block imposes an overhead of predicting the estimated error. When the estimation error is too large, prediction is not used and an I-MB is used instead.

Encoding a P (or B) macro-block imposes a search for the best candidate macro-block in the reference frame. A benefit of MPEG-4 is that is does not restrict the search area. However, searching for a candidate is a complex and time-consuming task. In Section 8 we describe a technique we use to reduce the computation complexity of the search to enable a real-time encoding.

3.2 Macro-block Quantization

An important parameter in the encoding process is the quantization value. The DCT coefficients are quantized and encoded into a variable size length of 3-26 bits. A successful choice of the quantizer is critical for an efficient encoding. Usually the quantization value is defined globally aiming at generating some given frame rate. The value is chosen according to the trade-off between better quality and higher bitrate; the higher the quantizer the better compression to be achieved and the more degrading quality. The quantization factor varies from 1 to 31 in steps of 1. A step change in the quantizer can improve the compression ratio while the degradation in quality is almost unnoticeable. MPEG-4 defines two types of quantization. In the first method, the same quantization value is used for all DCT coefficients of the macro-block. In the second, a matrix of quantization values for the macro-block is used where each coefficient can be quantized with a different value according to the matrix (see [1], Section 7.4.4.1). Further in this paper we refer to the quantization method in which one quantizer is used for all coefficients. Choosing a quantizer and a quantization method is an active area of research. Quantizing the correct coefficients with the correct value gives the best compression ratios while maintaining quality as far as possible. Another usage of the quantizer is for bit-rate control purposes. A bit-rate mechanism usually modifies the quantizer on a per-frame base. Current

approaches make use of the human visual system (HVS) to modify the quantizer on a per macro-block base (see [1], Annex L.).

Figure 2 is an example of the effect of different quantization values. In (a), one can see the quality of a sequence encoded with a quantizer value of 2. In (b), the quantization value is 18 and the quality degrades significantly.

(a) (b)

Fig. 2. The effect of applying different quantization values. The quantization value in (a) is 2, and 18 in (b).

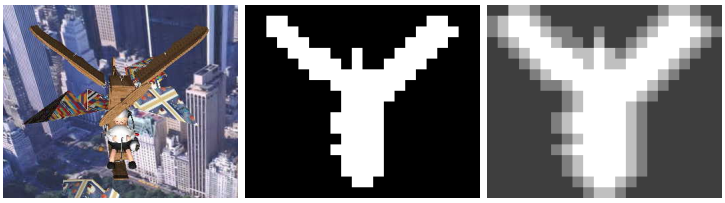
4 Per-layer Quantization

Efficient frame encoding can be achieved by choosing the correct quantization value per macro-block according to its visual importance. This necessarily requires some image understanding of the scene represented in the sequence.

If depth information is given or some other knowledge about the model in the image, a frame can be segmented into layers, and each layer can then be assigned a different visual importance value. However, automatic segmentation of an image, or a sequence of images, into layers is not simple and still an active area of research (e.g., [21, 8]) Moreover, real-time segmentation of video sequences is very complex and is not feasible yet.

When the frame sequence encodes views of a computer-generated model, the layering problem is trivial. For example, one can easily segment the image into background and foreground macro-blocks as shown in Figure 3. The figure shows an image segmented and classified into two layers (see Figure 3(b)). The macro-blocks containing the foreground objects are colored in white; the less visually important macro-blocks are in black. Our server-based system is based on the ability to properly select different quantization values on a per macro-block level to yield better compression ratios, while degrading only the background macro-blocks.

The quantization value for background and foreground can differ in several levels. For example, one can assign a quantization value of 10 to the background and a quantization value of 2 to the foreground. However, such an approach can cause artifacts, as the transition from foreground to background might be noticeable. Our approach uses a gradual change of the quantization value, so that the transition from foreground to background is not abrupt and noticeable (see Figure 3(c)).



(a) (b) (c)

Fig. 3. A frame from the Studio sequence in (a), and its foreground/background layering in (b). In (c) a smoothing of the layering

Using a different quantization value for each layer, our method yields better compression ratios. The quality is degraded only in areas that are defined as background. These areas are not in the viewer focus and therefore less visually important. There are other methods which assign different quantization values on a per macro-block base [10]. However, these methods do not take into consideration regions of interest. Our approach is based on the knowledge of the model and on the visual importance of its different parts. Thereby, regions that are less visually important are compressed with a higher ratio, while preserving the appropriate quality for regions of higher importance to the viewer.

5 Real-time Encoding

5.1 Selecting the Encoding Pattern

The computer-generated model can be further exploited to compress the frame sequence by selecting an I frame when it is known to be needed. In a conventional encoder, a pre-defined pattern is used to determine the type of frame to encode. Patterns, such as IPPPP or IPBBBPBBI, are used to determine which frame should be independently encoded and which should be encoded using motion estimation. Playing a video sequence from a new location in time is termed seek. I frames (except for the first frame) are used in the encoding process to enable seek capability at the end viewer and to reduce the propagated error of P frames over error-prone networks.

Most of the frames types, as defined in the encoding pattern, are motion estimated ones. In cases of scene cuts, where the whole scene changes in a given

frame, no motion vectors can be found. A scene cut, encoded as a P frame, imposes a redundant search for each of its macro-blocks. The search does not find a suitable motion vector since the motion estimation error is too high and the macro-block ends up being encoded as an I-MB.

Using the scene's model, one can use I frames only when necessary. The frame encoding pattern is not pre-defined but determined in real-time as a function of the walkthrough. The system uses its ability to detect scene-cuts and informs the Frame Encoder to avoid the search for motion, and to encode the current frame as an I frame. The Frame Encoder's complexity is considerably reduced for these frames and the processing power can be diverted to other time-consuming tasks. Another case, in which the system informs the Frame encoder to encode a frame as an I frame, is when the network is error-prone. As back-channel exists, the client can transmit to the server the statistics of lost frames. As the number of lost frames increases, the server may increase the number of I frames to reduce the propagated error of P frames.

5.2 Accelerating Motion Estimation

Searching for a motion vector is the most time-consuming task in the encoding, and several search algorithms have been proposed to accelerate its computation [12, 9]. There is a tradeoff between the complexity of the search and its resulting motion vectors. Typically, there is a direct relation between the complexity of the search algorithm and quality of the match. An exhaustive search yields the best possible match, but of course, it is overly expensive, since algorithms with relatively low complexity can provide acceptable results for real-time applications.

Wallach et al. introduced in [22] a technique to find the motion vectors based on the available optical flow information using common graphics hardware. The Macro-block Analysis module also exploits the known model and accelerates the search algorithm by providing approximate motion vectors to the Encoder module. In our implementation we use the Logarithmic motion estimation algorithm [12]. This is a hierarchical search that checks the corners, sides, and centers of a 2Nx2N pixel square, for the best match and continues recursively on the best result as the new center. This means that in each iteration there are nine possible motion vectors (4 corners, 4 sides, 1 center). Based on the available scene model and the available camera parameters, the macro-block analysis module can compute an approximate optical flow for a given macro-block. This is used as the initial location of a Logarithmic search by the Encoder. This saves the first level search and gains a speed-up of nine in encoding and yields a real-time encoding throughput. The speed-up gained in the Logarithmic search time does not come without cost, since the computation of the optical flow is also time-consuming. However, in our system this overhead is hidden since the Macro-block Analysis and the Frame Encoder work simultaneously in a pipeline fashion. Thus, the acceleration of the encoding improves the overall performance of the system and enables its usage as in real-time scenarios.

6 Results

We have implemented our technique within the framework of MPEG-4. The encoder is implemented in C, the decoder is pure Java and can be used on any Java-enabled platform, while the rendering engine is based on a proprietary implementation. Table 1 compares the stream size of the frame sequences encoded with a constant quantization value and those encoded with our method. The size of each frame in the sequence is CIF (Common Intermediate Format; 352x288). The figures show that our method provides better compression ratios while only degrading the quality of the background regions. The background regions have less importance to the viewer and therefore the degradation in quality is acceptable. We use the same quantization value for the foreground as in the constant quantization and a higher quantization value for the background. In the table, the left column shows the quantization range used (high-low). The lower value is used for foreground quantization and the higher value is used for background quantization. The "2-levels" column shows our resulting stream sizes which are lower than the sizes of the streams in the "low" columns. The "high" column shows the sizes of streams encoded with the same value we used for the background. Although its size is lower, its quality is worse than in our 2-levels approach. This is especially noticeable in the areas of the foreground object itself. We have repeated the same analysis for a QCIF (Quarter CIF; 176x144) size type. The results are similar, however the compression gained is somewhat less than in CIF size. This is because the macro-block size is fixed and therefore the ratio between the foreground and the background macro-blocks is a bit higher for QCIF.

Figure 4 compares images encoded by our technique and naive quantization, in which the same quantization value is used for all macro-blocks. The two sequences have been encoded with different ranges. The left column is from the "Pilot" sequence, encoded in the range of 10-4. The right column is from the "Studio" sequence, encoded in the range of 12-6. In the upper row, the lowest value is used as the constant quantization value. While in the lowest row, the higher value is used. The middle row shows frames of our two-layer technique. Note that the quality of the foreground object is as high as that in the upper row. The movies and a player, are available for download at: http://www.cs.tau.ac.il/~dcor/mpeg4/

7 Conclusions and Future Work

We have introduced a server-based system that alleviates the processing power required by a thin client to render an interactive remote walkthough. The technique exploits the fact that a thin client has the ability to decode video sequences independently of its computational power. Since we are streaming computer-generated video, we benefit from the available model of the scene to better compress the sequence and reduce the required bandwidth and communication latency. Current bandwidth issues might delay the immediate adoption of this

Table 1. The stream sizes (KBytes) of CIF (352x288 frame) in different quantization levels. Each line represents a different range of quantization values. The column "low" shows the size of a stream quantized with the lowest value in the range; the "high" shows the size of the stream with the highest value, and the "2-level" column shows the size using our per-layer technique .

Quantization	**Studio Sequence**			**Pilot Sequence**		
	high	2-levels	low	high	2-levels	low
8-2	1,813	3,038	7,307	1,657	3,404	6,947
10-4	1,388	1,980	3,851	1,249	2,121	3,596
12-6	1,105	1,459	2,516	979	1,514	2,326
14-8	908	1,143	1,813	788	1,151	1,657
16-10	767	932	1,388	652	912	1,249
18-12	660	783	1,105	548	744	979

Table 2. The stream sizes (KBytes) of QCIF (176x144 frame) in different quantization levels as in Table 1.

Quantization	**Studio Sequence**			**Pilot Sequence**		
	high	2-levels	low	high	2-levels	low
8-2	512	1,160	2,257	430	1,190	1,788
10-4	378	699	1,158	319	698	938
12-6	291	485	735	244	476	609
14-8	230	359	512	191	347	430
16-10	188	278	378	152	264	319
18-12	157	223	485	124	207	244

technique. We expect this technique to be adopted as emerging networks gain more bandwidth. However, it does not require broadband networks. This technique can be applied in interactive network games where the server needs to serve several clients playing in a common virtual environment.

The method we have presented in this paper focuses on the basic set of MPEG-4 Video [1]. MPEG-4 defines various types of other "media objects" rather than video objects. However, today's devices are still limited to MPEG-4 video decoding and do not provide the ability to display a whole MPEG-4 scene. Moreover, the devices cannot compose several video objects overlaid. Such a capability would enable us to enhance this technique and further improve its performance.

8 Acknowledgment

The authors wish to thank the following people for their various contributions: Efi Fogel and Amit Shaked from Enbaya Ltd., Zohar Sivan and Konstantin Kupeev from IBM Research Lab in Haifa.

References

[1] Information Technology - Coding of Audio-Visual Objects - Part2: Visual. ISO/IEC 14496-2, Dec. 1999.

[2] Overview of the MPEG-4 Standard (ISO/IEC JTC1/SC29/WG11 N3747). http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.html, Oct. 2000.

[3] D. G. Aliaga and A. A. Lastra. Architectural walkthroughs using portal textures. In R. Yagel and H. Hagen, editors, *IEEE Visualization '97*, pages 355–362, Nov. 1997.

[4] M. Budagavi, W. Rabiner, J. Webb, and R. Talluri. Wireless mpeg-4 video on texas instruments dsp chips. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 4, pages 2223–2226, 1999.

[5] D. Cohen-Or, Y. Mann, and S. Fleishman. Deep compression for streaming texture intensive animations. In *Computer Graphics* Proceedings, Annual Conference Series, pages 261–268, Aug. 1999.

[6] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Computer Graphics* Proceedings, Annual Conference Series, pages 247–254, Aug. 1993.

[7] G. Hesina and D. Schmalstieg. A network architecture for remote rendering. In A. Boukerche and P. Reynolds, editors, *Proceedings of Second International Workshop on Distributed Interactive Simulation and Real-Time Applications*, pages 88–91, July 1998.

[8] H. Huang, W. Lin, Y. Hung, and C. Fuh. New video object segmentation technique based on flow-thread features for mpeg-4 and multimedia systems. In *Image and Video Communications and Processing 2000*, pages 204–212, 2000.

[9] A. kHI RN, O. Au, and M. Liou. Fast motion estimation using circular zonal search, 1999.

[10] H.-J. Lee, T. Chiang, and Y.-Q. Zhang. Scalable rate control for very low bit rate (vlbr) video. In *Proceedings of the 1997 International Conference on Image Processing (ICIP '97)*, volume 3, 1997.

[11] M. Levoy. Polygon-assisted jpeg and mpeg compression of synthetic images. *Proceedings of SIGGRAPH 95*, pages 21–28, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[12] R. Li, B. Zeng, and M. Liou. A new three-step search algorithm for fast motion estimation. *IEEE Trans. Circuits Syst. Video Technol.*, 4:438–442, 1994.

[13] M. R. Macedonia, D. P. Brutzmann, M. J. Zyda, D. R. Pratt, P. T. Barham, J. Falby, and J. Locke. NPSNET: A multi-player 3D virtual environment over the internet. In P. Hanrahan and J. Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 93–94. ACM SIGGRAPH, Apr. 1995.

[14] P. W. C. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *1995 Symposium on Interactive 3D Graphics*, pages 95–102, Apr. 1995.

[15] Y. Mann and D. Cohen-Or. Selective pixel transmission for navigating in remote virtual environments. *Computer Graphics Forum*, 16(3):201–206, August 1997.

[16] T. Nishikawa, M. Takahashi, M. Hamada, T. Takayanagi, H. Arakida, N. Machida, H. Yamamoto, T. Fujiyoshi, Y. Maisumoto, O. Yamagishi, T. Samata, A. Asano, T. Terazawa, K. Ohmori, J. Shirakura, Y. Watanabe, H. Nakamura, and S. Minami. A 60 mhz 240 mw mpeg-4 video-phone lsi with 16 mb embedded dram. In *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International , 2000*, pages 230 –231, 2000.

[17] G. Schaufler and W. Stürzlinger. A three dimensional image cache for virtual reality. *Computer Graphics Forum*, 15(3):227–236, August 1996.

[18] J. Shade, D. Lischinski, D. H. Salesin, T. DeRose, and J. Snyder. Hierarchical image caching for accelerated walkthroughs of complex environments. In *Computer Graphics* Proceedings, Annual Conference Series, pages 75–82, 1996.

[19] H. Stolberg, M. Berekovic, P. Pirsch, H. Runge, H. Moller, and J. Kneip. The mpire mpeg-4 codec dsp and its macroblock engine. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 2, pages 192–195, 2000.

[20] E. Teler and D. Lischinski. Streaming of complex 3d scenes for remote walkthroughs. *to appear in Computer Graphics Forum (Proceedings of Eurographics 2001)*, 2001.

[21] K. Vaithianathan and S. Panchanathan. Analysis of object segmentation methods for vop generation in mpeg-4. In *Image and Video Communications and Processing 2000*, pages 191–203, 2000.

[22] D. S. Wallach, S. Kunapalli, and M. F. Cohen. Accelerated mpeg compression of dynamic polygonal scenes. *Proceedings of SIGGRAPH 94*, pages 193–197, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

[23] I. Yoon and U. Neumann. Web-based remote rendering with ibrac (image-based rendering acceleration and compression). *Computer Graphics Forum*, 19(3):321–330, August 2000.

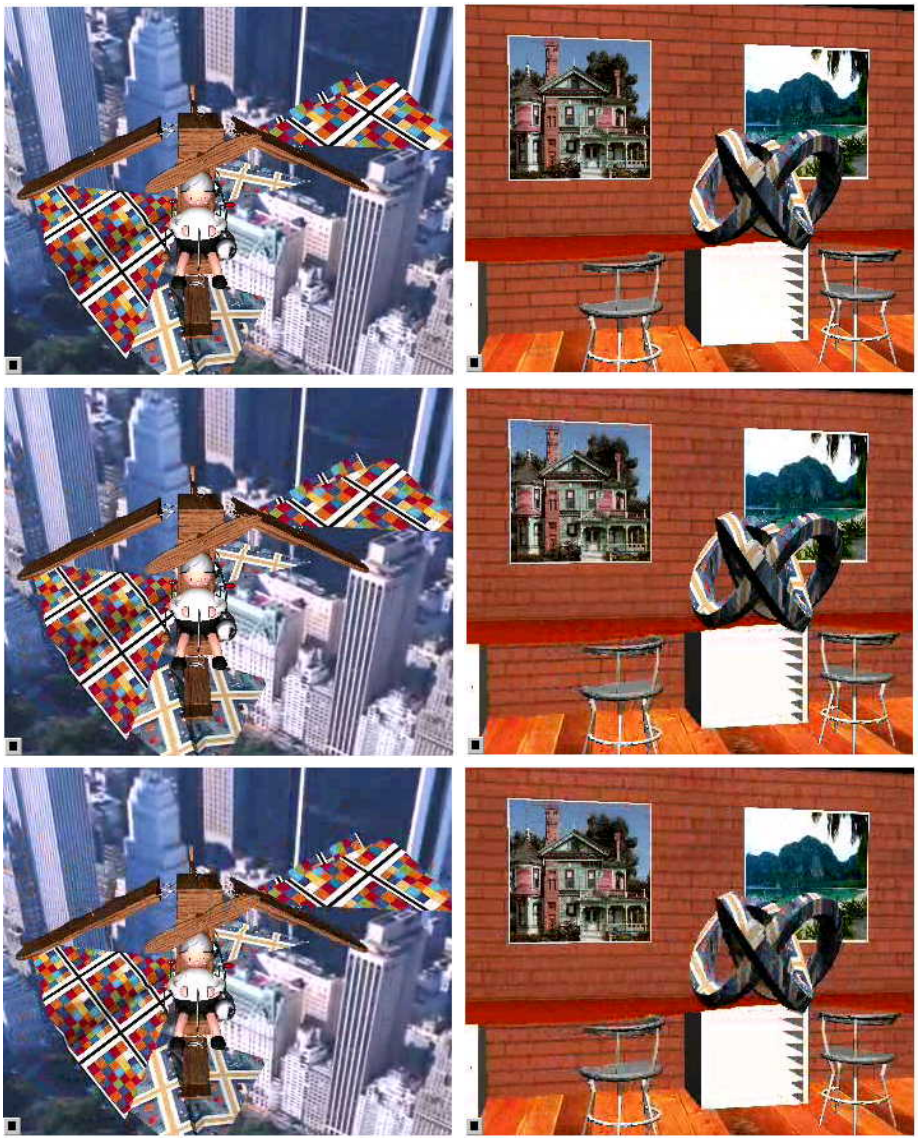


Fig. 4. A comparison of images encoded by our technique and naive quantization. In the upper row, the lowest value is used as the constant quantization value. While in the lowest row, the higher value is used. The middle row shows frames of our two-layer technique.