Effects of Dynamic Quantization Noise on Video Quality

Thom Carney[1], Yuan-Chi Chang[2], Stanley A. Klein[1] and David G. Messerschmitt[2]
[1]Neurometrics Institute, and School of Optometry, University of California, Berkeley, Berkeley, CA 94720, USA
[2]Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA 94720, USA

Abstract

Packet transmissions over the Internet incur delay jitter that requires data buffering for resynchronization, which is unfavorable for interactive applications. Last year we reported results of formal subjective quality evaluation experiments on delay cognizant video coding (DCVC), which introduces temporal jitter into the video stream. Measures such as MSE and MPQM indicate the introduction of jitter should degrade video quality. However, most observers actually preferred compressed video sequences with delay (DCVC) to sequences without. One reason for this puzzling observation is that the delay introduced by DCVC suppresses the dynamic noise artifacts introduced by compression, thereby improving quality. This observation demonstrates the possibility of reducing bit rate and improving perceived quality at the same time. We have been characterizing conditions in which dynamic quantization noise suppression might improve video quality. A new battery of video test sequences using simple stimuli were developed to avoid the complexity of natural scenes. These sequences are cases where quantization noise produces bothersome temporal flickering artifacts. We found the significance of artifacts depend strongly on the local image content. Pseudo code is provided for generating these test stimuli in the hope that they lead to the development of future video compression algorithms which take advantage of this technique of improving quality by dampening temporal artifacts.

Keywords: delay cognizant video coding, video compression, network video

1 Introduction

To compress or not to compress, that is the question.

This paper examines the effects of dynamic quantization noise on video quality, especially in the cases where small changes in uncompressed image sequences incur strong, noticeable artifacts after compression. In a video sequence, regions of the image may undergo small changes over several frames. Depending on the context of the video, these changes may be unnoticed under normal viewing conditions. While modern compression algorithms remove video signal redundancy to reduce the demanding bit rate (bandwidth) requirement, this process results in quantization noise. Quantization noise can be very dynamic even when the original signal changes little over time. Dynamic quantization noise, in the form of flickering, is often bothersome to the observer and draws the observers' attention. For regions that change little over time, it may arguably be better not to compress those regions for each frame and just repeat the region from the previous frame. The associated dynamic noise with frame by frame compression could be more bothersome than the static quantization noise of compressing just once and repeating the frame. Suspending compression (reducing bit rate) and simultaneously achieving better picture quality seem to be conflicting goals from the signal processing point of view. However, when visual perception is considered, they are not necessarily mutually exclusive. In this paper, we address this issue by proposing a battery of artificial stimuli that serve as good examples of what not to compress frame by frame. The long-term goal of this project is to develop rules for detecting those ill-compressioned conditions and answer the question, compress or not to compress.

The work is motivated from results of subjective quality evaluation experiments on delay cognizant video coding (DCVC), which introduces temporal jitter to video. The results were reported in [2] and presented in the conference Human Vision and Electronic Imaging III. DCVC is new type of layered coding algorithm that generates two compressed video flows with differential delay requirements, a delay-critical flow and a delay-relaxed flow [3]-[5]. The delay-critical (low-delay) flow has to synchronize with audio and it carries the portion of video data that affects the perceptual delay[1] most. The delay-relaxed (high-delay) flow car-

Part of the IS&T/SPIE Conference on Human Vision and Electronic Imaging IV • San Jose, California • January 1999
SPIE Vol. 3644 • 0277-786X/99/$10.00

ries the rest of video to gradually improve the quality. A DCVC decoder applies a different mode of video reconstruction from the traditional, synchronous approach. Since video packets from the transmitter are carried by differential delay flows (streams with differential delay requirements), they arrive at the receiver with different latencies. Video data is rendered immediately upon arrival. Figure 1 illustrates an example of this asynchronous reconstruction, in which two video blocks acquired at the same instant in the transmitting terminal are assigned to different delay flows, arrive at the receiver at different instants, and are displayed immediately.

Asynchronous rendering of DCVC was predicted to degrade video quality over the traditional approach of synchronous rendering. Surprisingly, last year's subjective quality evaluation experiments revealed that for some compressed sequences and most observers, DCVC appeared to improve rather than degrade video quality [2]. Ideally, delaying any part of an 'optimally' compressed video will cause quality degradation, because an optimal compressor should have kept only visually significant information. Any additional changes, either spatially or temporally, should cause a noticeable degradation. The compression of DCVC was not optimal in the visual quality sense and thus our findings suggest there might be ways to improve quality and save bits at the same time. This surprising finding prompted us to examine this issue.

The paper is organized as follows. Section 2 briefly discusses the main results from the subjective experiments of DCVC. Section 3, 4, and 5 introduce artificial stimuli we constructed to mimic the natural scenes which favor DCVC video with long delay. Section 6 summarizes the paper. Pseudo code for generating the stimuli is provided in appendices.

2 Subjective Testing of DCVC

2.1 Review of DCVC

Delay cognizant video coding (DCVC) is a new type of layered coding algorithm for delay critical network video applications. Abolishing the implicit assumption in conventional video coding that every bit in a video stream has the same delay requirement, a DCVC encoder segments video information into multiple flows (layers) with differential delay requirements. The DCVC encoder shown in Figure 2 is divided into two stages: segmentation and compression, each of which is framed in the figure. A video frame is first processed by the segmentation stage to extract the low-delay information. High-delay data are obtained by subtracting low-delay data from the video frame. Extracted flow data are then passed to the compression stage to remove spatial and temporal redundancy. We describe the algorithm in the order of the processing: segmentation first followed by compression.

The first stage of encoding applies block-based segmentation. A captured video frame is first divided into blocks of size 8 by 8. Each block is then independently assigned to either the low- or high-delay flow, based on the spatio-temporal frequency variation in the block. The flow diagram of this stage is marked and shown in the left box in Figure 2.

The segmentation algorithm identifies visually significant blocks by measuring the similarity of a block and its predecessor in time. There are a total of 128 test

[1]Perceptual delay is a subjective measure of delay in video communications. It can be quantified by the end-to-end latency of perceiving a motion event, such as nodding or gestures, happening at the transmitting end. It can also be characterized by the relative delay to the associated audio signal for maintaining lip sync. The former definition may apply without the presence of audio channel. Both definitions are measured by subjective evaluations. The longer the perceptual delay, the poorer the interactivity of the application.

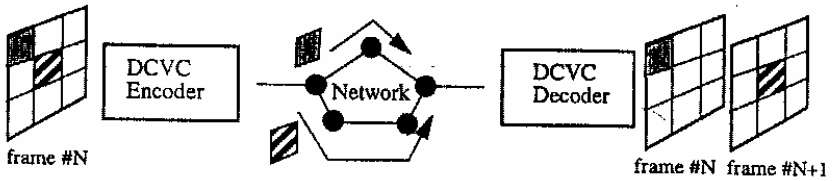

FIGURE 1. An illustration of differential delay flows and asynchronous reconstruction; the top path is shorter and thus the block arrives earlier than the other block taking the bottom path

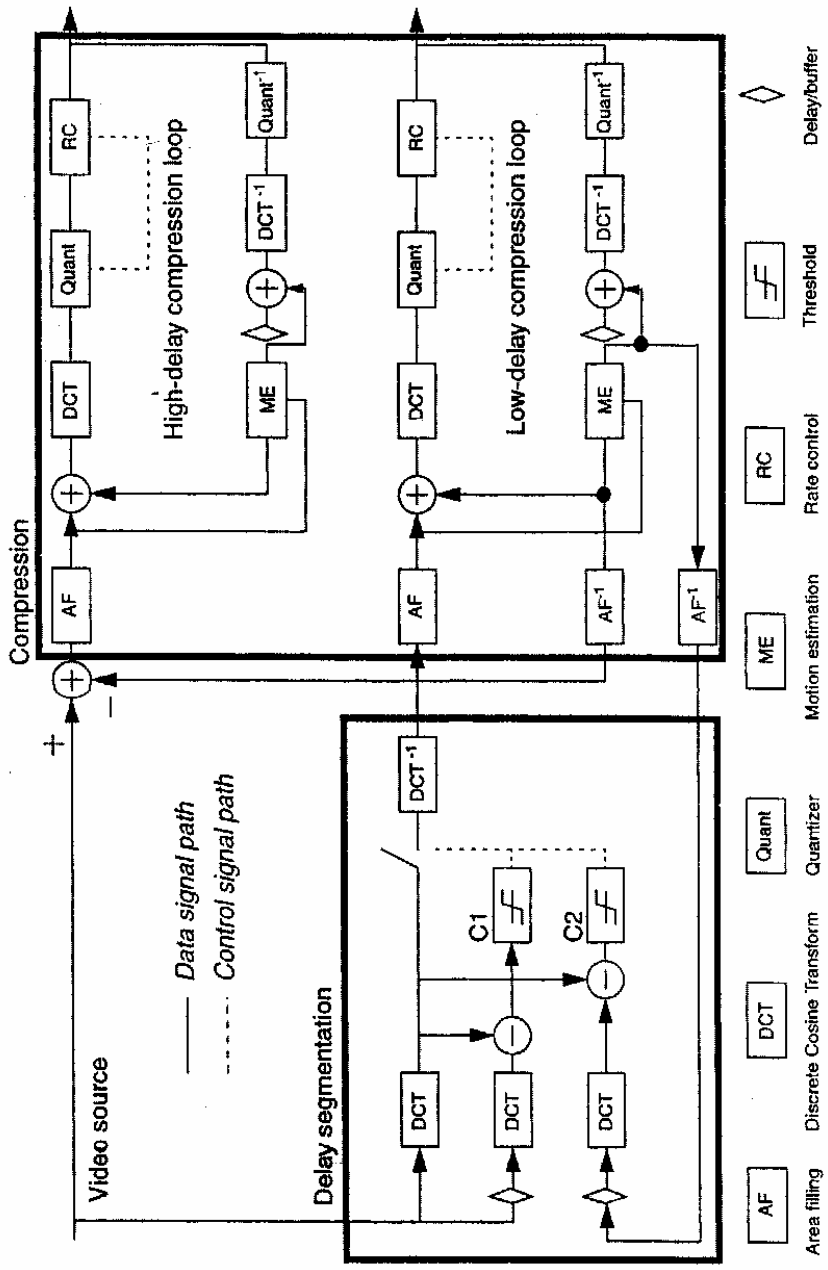

FIGURE 2. Delay cognizant video encoder block diagram

conditions, all of which must be satisfied for a block to be assigned to the high-delay flow. The 128 conditions are composed of 2 conditions each for every Discrete Cosine Transform (DCT) coefficient of the tested block, which has 64 coefficients. Since each coefficient is independently tested, it suffices to look at just one pair of such conditions:

Condition 1: $|P_{i,j,n,t} - P_{i,j,n,t-1}| < V_{i,j}$

Condition 2: $|P_{i,j,n,t} - P_{i,j,n,update}| < V_{i,j}$

In the above expressions, $P_{i,j,n,t}$ is the (i, j)th DCT coefficient for block n at time t; $P_{i,j,n,update}$ is the (i, j)th coefficient of block n stored in a buffer for the latest update; $V_{i,j}$ is a fixed preset threshold for the (i, j)th coefficient. Reasons behind these conditions are detailed in [5].

The compression stage as shown at the right frame of Figure 2, removes spatial and temporal redundancies from the segmented video flows. Our design adopts the motion estimation (ME) with discrete cosine transform (DCT) architecture [8][9][10]. The upper (lower) ME loop in the diagram corresponds to the compression of the high- (low-) delay flow. The reconstructed low-delay image plane in the ME loop is fed back to the segmentation stage. The feedback is stored as the latest update for the second segmentation condition described previously. The high-delay image plane is obtained by subtracting the reconstructed low-delay image plane from the input video frame. This allows quantization errors in the low-delay ME loop to be passed as a part of the input to the high-delay ME loop.

The DCVC decoder is also divided into two stages: the independent decompression of flows and their video composition. The two video flows are combined for final rendering by the following method. Compressed bit streams from both flows are tagged with temporal references (frame numbers). The decoder maintains one temporal reference table for each flow, in which each entry stores the frame number of the received block at the coordinates. The tables are initiated to zero and blocks from earlier frames are replaced with those from later frames. By comparing $TR_{n,L}$, temporal reference of the nth block from the low-delay flow, and $TR_{n,H}$, temporal reference of the nth block from the high-delay flow, the decoder makes the following decision:

1. $TR_{n,L} > TR_{n,H}$, display the block from the low-delay flow;
2. $TR_{n,L} = TR_{n,H}$, display the sum of two blocks;
3. $TR_{n,L} < TR_{n,H}$, display the block from the high-delay flow.

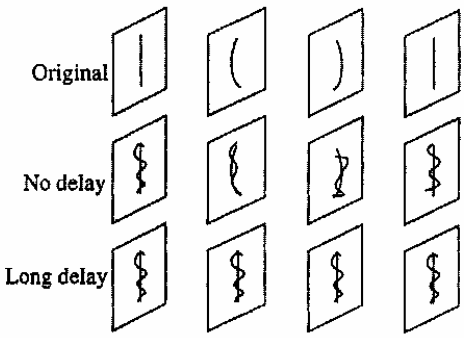

FIGURE 3. A qualitative illustration of the condition when delayed video looks better.

2.2 Subjective testing results

We conducted video viewing experiments with the participation of 11 human subjects. Seven H.263 test sequences were used. We evaluated video quality at two compression levels and two delay levels. Comparing video clips with no delay (synchronous rendering) and long delay (asynchronous rendering with 400 msec lagging in the high-delay flow), most subjects preferred the video quality of long delay sequences.

How can delay improve video quality, even by a small amount? Figure 3 schematizes an example where delay should improve video quality by reducing dynamic quantization noise. The figure illustrates the condition when an original, uncompressed block (8 by 8 pixels) is varying slowly in time. Under high compression, the compressed block contains quantization noise. Upon rendering the video sequence, the block closely follows the luminance variation of the original block. However, the quantization noise changes from frame to frame as shown in the second row of Figure 3. The noise results in flickering and is very annoying. For delayed video, however, the encoder sends the second to the fourth block to the high delay flow, which will arrive at the receiver after 400 msec. In the mean time, the decoder simply keeps showing the first block received as shown in the third row of Figure 3. The quantization noise seen by our subjects is thus static. The static noise seems to be preferred to the dynamic noise. The static noise might even be attributed to the

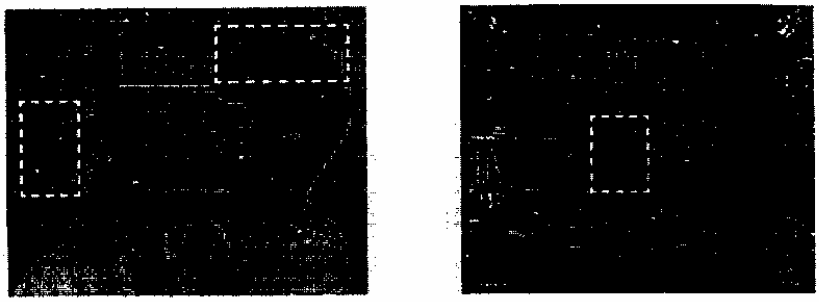

FIGURE 4. Image regions that incur strong flickering noise under heavy compression are marked by dotted boxes.

original image but dynamic noise is clearly not a part of the original scene.

Our goal is to isolate the causes of quality augmentation to improve the compression algorithm. As illustrated in Figure 3, small variations 'modulate' accompanied quantization noise, which often causes flicker. Dynamic, flickering noise attracts human viewers' attention and leads to lower quality ranking. On the other hand, if the compressor ignores those small variations, quantization noise becomes static and is often overlooked if it does not occur in key areas. After studying the test sequences, we found attention-attracting flickering occurred in mostly static, medium to high textured areas. Examples are marked in Figure 4.

As we argued in [11], natural images are much harder to analyze than simple lines, edges and gratings used in basic vision research. Natural scenes in video are even harder to analyze because of the temporal dimension. Marked areas in Figure 4 do not always flicker and localizing its occurrence both spatially and temporally is rather difficult. Our approach is to design artificial stimuli which mimic these scenes in creating dynamic noise. Simple artificial stimuli are easier to study in order to develop new guidelines for compression. We recognize, however, flickering noise does depend on video content. For example, flickering of densely leaved trees are hard to identify as artifacts. A weakness of artificial stimuli is that they do not take into account context dependency.

3 Stimulus I: pulsing circles

Using artificial stimuli, we noticed that stimuli periodic in space and time emphasized the importance of context and anticipation in assessing how bothersome compression artifacts are. Large area periodic textures resulted in periodic artifacts that tended to be attributed to the original image rather than perceived as artifact. This was less common when the texture area was greatly reduced in size. Similarly, dynamic noise artifacts that repeated in time were easily anticipated and consequently perceived as less bothersome. The test stimuli described below avoided these problems by limiting the texture size and using non-repetitive motions

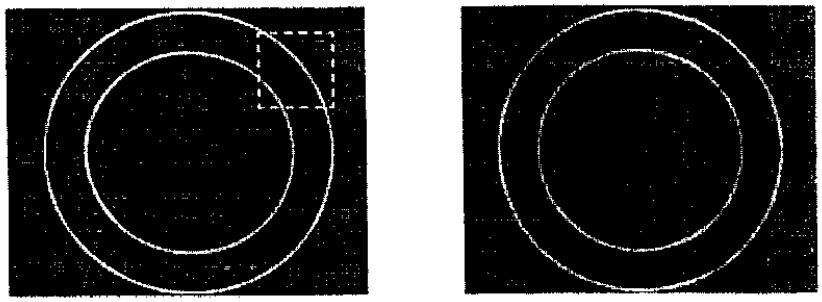

FIGURE 5. Uncompressed (left) and compressed (right) video frames of the pulsing circles stimulus; the dotted box is magnified in 3D shown in Figure 6.

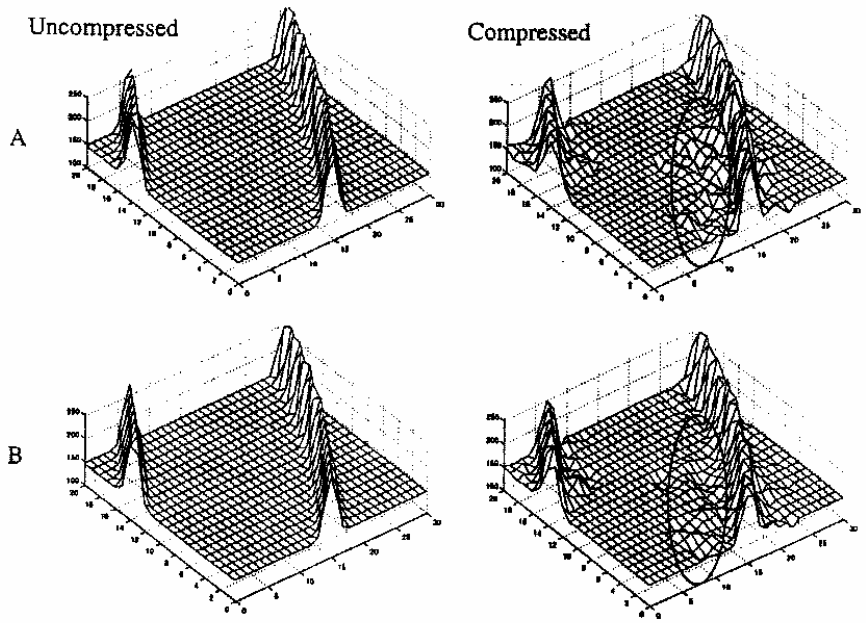

FIGURE 6. A magnified view of the dotted box in Figure 5; temporal variation of quantization noise causes flickers.

(motion was given by the pattern matrix in the appendix).

Our first artificial stimulus is pulsing circles (change in radius). The amount of the radius change is 0.2 pixel (one pixel is 2.16 min wide in our viewing environment). Fractional distance change is achieved by plotting a virtual ring with Gaussian distribution as the ring width. Pixel values are then mapped from the distribution. The stimulus appears to be static perceptually in uncompressed video while in fact the two rings expand and contract synchronously. We applied the H.263 compression algorithm to process the sequence [1][6]. Under heavy compression, the small pulsing movement is *visually amplified* by quantization noise around the rings. The dynamic quantization noise (flickering) draws viewers' attention. Pseudo code for generating the stimulus is provided in Section 7.

While it is difficult to show flickering, which involves temporal changes, on paper, an uncompressed and its associated compressed images are plotted in Figure 5 for reference. To help readers visualize the dynamic noise, Figure 6 shows a small segment of the rings in three dimensions, where z-axis represents pixel intensity. The figure shows uncompressed and compressed images of two consecutive frames. For uncompressed frames, there is a small, hard-to-notice change on the shape of the rings. Under heavy compression, however, the change of pixel values in neighboring areas of both rings from the frame A to B are more noticeable (the area in the ellipses).

The pulsing circles stimulus best characterizes scenes where the boundary between uniform and textured regions has small movements, which may be caused by unsteady camera motion.

4 Stimulus II: luminance modulated noise

The second stimulus is a realization of the case illustrated in Figure 3. The stimulus is created by adding a static white noise on a luminance ramp, which slowly varies its intensity. The noise pattern does not change. Only the height of the ramp varies. This stimulus mimics the movement of shadow over textured regions. Its pseudo code is given in Section 8.

As shown in Figure 7, image areas of three consecutive frames are plotted. The left column is the uncom-

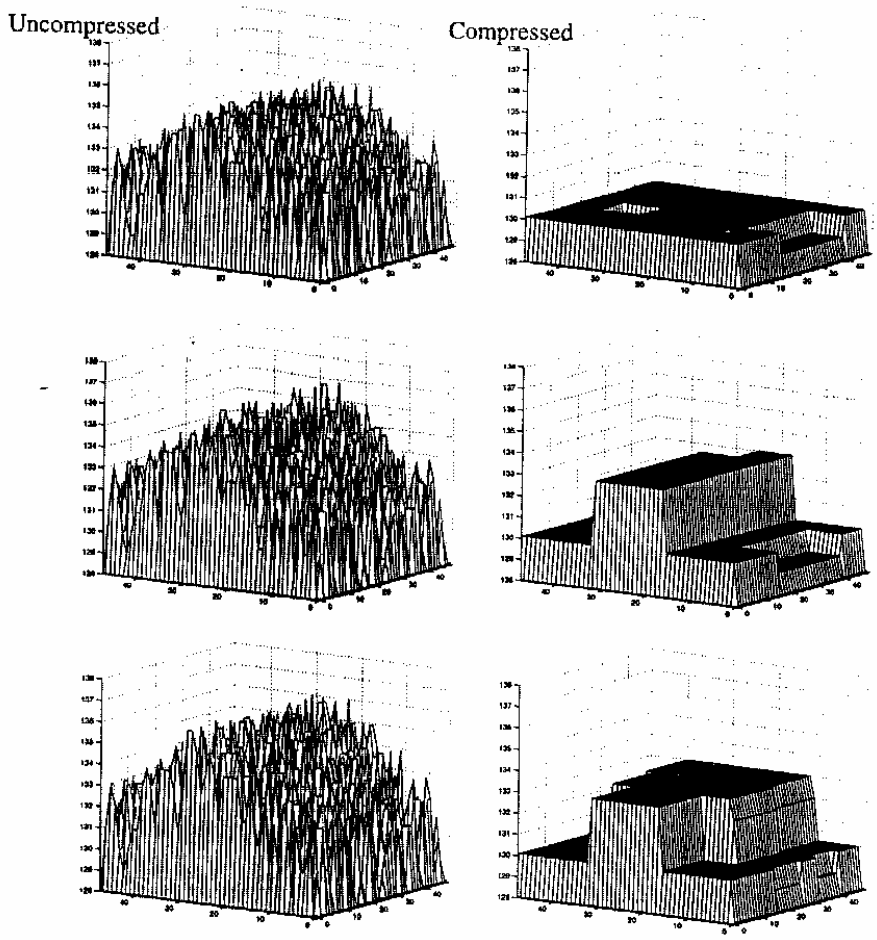

FIGURE 7. A magnified view of the noisy ramp stimulus before and after compression.

pressed frames while the right shows the compressed ones. Distinct features (edges and blocks) appear in compressed images, as they are caused by motion compensation and quantization. As the compressed video is played back, strong flicker is observed due to the movements of edges.

5 Stimulus III: moving noise fields

The third stimulus represents another type of scene that creates flicker. In this stimulus, two random noise fields are moved relative to the other. This stimulus mimics scenes containing slow motion of transparent objects, such as glass bottles. Its pseudo code is provided in Section 9.

Figure 8 plots uncompressed and compressed image areas of two consecutive frames. The moving noise fields are marked by arrows in this figure. It is interesting to see the 8 by 8 block at the center bottom of the figure transforms from blank to texture by a single pixel movement (the first arrow from the right). This is because the movement of a pixel from the neighboring block contributes enough signal energy in the frequency domain. Due to the property of block DCT, quantized

signal spreads out to the whole block. The flashy transition is very noticeable on screen.

6 Summary

We identified some of the natural scenes in which delayed video looked better and constructed artificial stimuli to mimic these scenes. Although in our experiments we applied the H.263 compression standard, these stimuli should be equally applicable to other compression algorithms based on motion compensation and DCT. We hope these simpler stimuli will help video coding researchers to develop better coding control rules and improve compression algorithms. Since our findings will only affect the compression end, decompression algorithms and bit stream formats need not change. Output streams of the improved compressor will be compatible with existing standards.

Acknowledgments

This research was supported by the University of California MICRO program and by grant from the Air Force Office Scientific Research (F49620-95) to the Neurometrics Institute.

References

[1] "Video coding for low-bit rate communications: draft recommendation ITU-T H.263," International Telecommunications Union - Telecommunication Standardization Sector, May 1996.

[2] Y. C. Chang, T. Carney, S. A. Klein, D. G. Messerschmitt, and A. Zakhor, "Effects of temporal jitter on video quality: assessment using psychophysical methods," *Proceedings of the SPIE - Human Vision and Image Processing*, San Jose, CA, 1998.

[3] Y. C. Chang and D. G. Messerschmitt, "Improving network video quality with delay cognizant video coding," *Proceedings of IEEE International Conference On Image Processing*, Chicago, IL, 1998.

[4] Y. C. Chang, T. Carney, S. A. Klein, and D. G. Messerschmitt, "Delay cognizant video coding: architecture, applications and quality evaluation," submitted to *IEEE Transactions on Image Processing*.

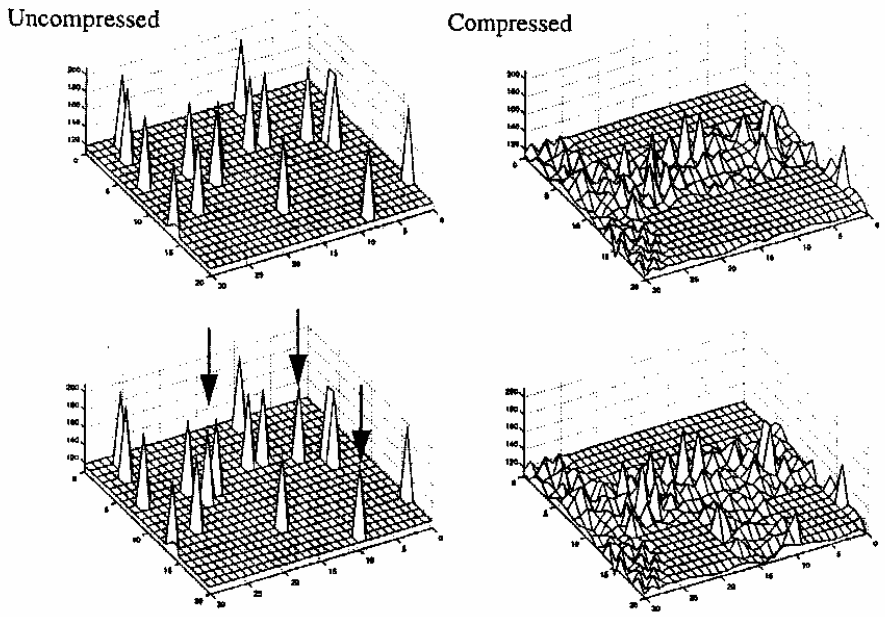

FIGURE 8. A magnified view of the moving noise fields stimulus before and after compression

[5] Y. C. Chang, "Delay cognizant video coding," *Ph. D dissertation*, University of California, Berkeley, 1998. See also http://ptolemy.eecs.berkeley.edu/~yuanchi/DCVC_papers.html.

[6] K. Olav Lillevold and R. Danielson, H. 263 encoding and decoding software, Telenor Research and Development, Norway, 1996.

[7] T. Carney, "PC-MatVis", Neurometrics Institute, Berkeley, CA. See also www.neurometrics.com.

[8] R. J. Clarke, *Digital compression of still images and video*, published by Academic Press, 1995.

[9] A. K. Jain, *Fundamentals of digital image processing*, published by Prentice Hall, 1989.

[10] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG video compression standard*, Chapman & Hall, 1997.

[11] S. A. Klein, "Image quality and image compression: a psychophysicist's viewpoint," *Digital Images and Human Vision*, A. B. Watson, ed. MIT Press, 1993.

[12] B. Girod, "What's wrong with mean-squared error?" *Digital Images and Human Vision*, A. B. Watson, ed. MIT Press, 1993.

7 Appendix A: pseudo code of pulsing circles stimulus

The following codes generate a sequence of 64 QCIF-sized frames (176 by 144). The standard H.263 compression algorithm is then applied with quantization step size set to 30. The playback rate is 30 frames per second using PC-MatVis [7]. Pixel values are gamma corrected before display to ensure luminance linearity.

```
int pattern[64] = {                    // this matrix governs the circle change radius
      0, 0, 0, 1, 0, 1, 1, 0,
      0, 0, 0, 1, 0, 0, 0, 1,
      0, 0, 0, 1, 0, 1, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 1,
      1, 1, 0, 0, 0, 0, 1, 0,
      0, 0, 0, 0, 0, 0, 0, 1,
      0, 1, 0, 1, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 1, 0};
radius = 70;                           // radius of the outer circle pixels
scaling = 200; // intensity            // the width of thin circles has a
spread = 0.3; // sigma                 // Gaussian shape; scaling and spread
                                       // determine the shape
PI = 3.1416;

for (frame = 0; frame < 64; frame++)
      {
      if (pattern[frame])
            {
            r = radius;                // the variation of radius is 0.2 pixels
            }
      else
            {
            r = radius+0.2;
            }

      for (i = 0; i < 176; i++)
      for (j = 0; j < 144; j++)
            {
            {
            dist1 = sqrt((i-88)^2+(j-72)^2))-r;
            dist2 = sqrt((i-88)^2+(j-72)^2))-r+20;
            pixelvalue[i][j] = scaling*exp(-dist1/2/spread)/sqrt(2*PI*spread) + scal-
ing*exp(-dist2/2/spread)/sqrt(2*PI*spread)+128;
```

```
            }
            }
      write pixel values
      }
```

8 Appendix B: pseudo code of luminance modulated noise stimulus

The following codes generate a sequence of 64 QCIF-sized frames (176 by 144). The standard H.263 compression algorithm is then applied with quantization step size set to 18.

```
int pattern[64] = {                    // the height of the ramp increases by one unit
      1, 1, 1, 1, 1, 1, 1, 1,          // when the entry is one; it decreases by one unit
      0, 0, 0, 0, 0, 0, 0, 0,          // when the entry is zero
      1, 1, 1, 1, 1, 1, 1, 1,
      0, 0, 0, 0, 0, 0, 0, 0,
      1, 1, 1, 1, 1, 1, 1, 1,
      0, 0, 0, 0, 0, 0, 0, 0,
      1, 1, 1, 1, 1, 1, 1, 1,
      0, 0, 0, 0, 0, 0, 0, 0};

noisestrength = 5;                     // amount of added noise
ramp = 0;                              // height of the ramp
stepsize = 1;                          // ramp size increases or decreases by this amount

for (i = 0; i < 48; i++)               // noise pattern is static
      for (j = 0; j < 48; j++)
            noise[i][j] = (double)rand()*noisestrength/RAND_MAX;

for (frame = 0; frame < 64; frame++)
      {
      if (pattern[frame] == 1)
            ramp += stepsize;
      else
            ramp -= stepsize;

      for (i = 0; i < 176; i++)
      for (j = 0; j < 144; j++)
            pixelvalue[i][j] = 128;
      for (i = 0; i < 48; i++)
      for (j = 0; j < 48; j++)
            {
            if (i < 24)
                  pixelvalue[i+48][j+64] = ramp*i/48+noise[i][j]+128;
            else
                  pixelvalue[i+48][j+64] = ramp*(48-i)/48+noise[i][j]+128;
            }
            write pixel values
      }
```

9 Appendix C: pseudo code of moving noise fields stimulus

The following codes generate a sequence of 64 QCIF-sized frames (176 by 144). The standard H.263 compression algorithm is then applied with quantization levels set to 20.

```
int pattern[64] = {
      0, 1, 1, 1, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 1, 1,
```

```
        0, 0, 0, 1, 1, 1, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0,
        0, 1, 1, 1, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1
        };

for (i = 0; i < 176; i++)
      for (j = 0; j < 144; j++)
            {
            if (rand()/RAND_MAX > 0.98)       // noise field 1 is static
                  noise1[i][j] = 208;
            else
                  noise1[i][j] = 128;

            if (rand()/RAND_MAX > 0.999)      // noise field 2 moves based on the
                  noise2[i][j] = 208;         // the pattern matrix
            else
                  noise2[i][j] = 128;
            }

for (frame = 0; frame < 64; frame++)
      {
      if (pattern[frame] == 1)   // noise field 2 moving distance: one or two pixels
            xoffset = 2;
      else
            xoffset = 1;

      for (i = 0; i < 176-xoffset; i++)
      for (j = 0; j < 144; j++)
            {
            if (noise1[i][j] > noise2[i+xoffset][j])
                  pixelvalue[i][j] = noise1[i][j];
            else
                  pixelvalue[i][j] = noise2[i+xoffset][j];
            }
          write pixel values
      }
```