Accelerating Non-photorealistic Video Effects using Compressed Domain Processing

Wen-Hung Liao[1]

[1] Department of Computer Science, National Chengchi University,
Taipei, Taiwan
whliao@cs.nccu.edu.tw

Abstract. Recently, various non-photorealistic rendering (NPR) techniques have been developed for computers to generate images of different artistic styles automatically. Due to the complexity of the algorithms, however, most NPR methods are limited to the processing of still images. It is the objective of this paper to extend and improve existing NPR techniques to enable near real-time processing of video. The enhancement will be achieved by compressed domain processing. By exploiting the relationship among I, P, and B frames, different strategies can be developed to increase the efficiency of the NPR algorithm. Experimental results have demonstrated the efficacy of the proposed methods and validated the near real-time creation of NPR video effects.

Keywords: Non-photorealistic rendering, compressed domain processing, MPEG standards.

1 Introduction

In computer graphics, photorealistic rendering techniques are aimed at generating artificial images of simulated environments that look "just like the real world." Conversely, non-photorealistic rendering (NPR) can be defined as any technique that produces images in a style other than realism. Recently, NPR has been become practical on personal computers thanks to rapid advances in hardware technology. Commercial software packages that can convert images into different artistic styles are abundant. Common examples of NPR effects include water color, oil paint and charcoal. However, current solutions are limited to the processing of still images due to the complexities of the NPR algorithms. Attempts to extend the creation of NPR effects to video have been made only recently [1][2], and most of the proposed methods are concerned about the realistic synthesis of artistic styles. That is, most algorithms are aimed at extending the methods designed for images to arrive at a consistently looking video. Speed is an issue of lower priority, since the computation is usually carried out off-line.

With the prevalence of low-cost recording devices, digital video acquisition and analysis is no longer beyond the reach of average users. Whereas fidelity may be of primary concern in commercial-grade film creation, in certain desktop applications, interactivity may be a more significant requirement. For example, we have developed

an interactive multimedia software named 'DDPlayCam'[3] that combines face detection with special effects to create amusing scenes in real-time. At present, the maximum resolution is limited to 320x240 to ensure a smooth operation. We have traded spatial resolution for higher frame rates since we believe an application with low latency can retain the user's interest and promote its usability. Application of NPR effects is restricted for the same reason. It is therefore the main objective of this paper to explore methods to speed up the creation of NPR video to make the interactive multimedia more versatile.

An intuitive approach to generate NPR video is to apply the NPR algorithm for images frame by frame and combine the individual pictures to form a dynamic scene. Not only is such a method inefficient, it also suffers from temporal incoherence where random flickering may appear in the output video. In [1], the 'paint over' technique was introduced to reduce the flickering. Optical flow was also taken into account to warp the brush stroke between frames. Since both improvements were achieved by exploiting temporal relationship, it is reasonable to expect that similar results can be obtained by undertaking the NPR generation problem in the compressed domain. Moreover, boost in performance can be realized as NPR effects will be applied to certain frames in a sequence and then propagated accordingly.

The rest of this paper is organized as follows. In Section 2 we review the MPEG standard and outline possible directions for manipulating the video in the compressed domain. Section 3 formulates the strategy for compressed domain processing. A generic method to create oil-paint effect is utilized to demonstrate the process. We then present some experimental results, along with discussions on the improvements in processing speed. A brief conclusion and outlook for future improvements are given in Section 4.

2 Compressed Domain Processing

With the standardization of video formats such as MPEG-1, 2 and 4, much of the multimedia content available nowadays is in compressed format already. Compressed domain features can be extracted directly from the data stream without conversion. They haven been used for video segmentation and object detection [4]. However, there is no previous work on enhancing NPR performance using compressed domain processing.

In MPEG-1 and 2 video, the compressed data encode the frequency domain coefficients as well as motion vectors. Every linear operator in the spatial domain has an equivalent in the frequency domain. Manipulating the compressed DCT coefficients can only yield linear operations. As such, no significant NPR effects can be created in this manner. To illustrate, Fig.1 depicts a high-pass filtered image where the computation is carried out purely in the DCT domain. It has a 'mosaicing' effect. Applying low-pass or band-pass filters, however, does not yield any interesting 'style'. As a result, our subsequent discussion will be centered at the utilization of motion vector information to speed up the NPR generation.

(a) (b)

Fig. 1. (a) original frame (b) Butterworth high-pass filtered.

MPEG standards exploit temporal redundancy to reduce the data required to code the image sequences. Three types of frames: I, P and B have been defined. Only I frames need to be coded independently. P frame is computed from the previously coded I- or P-frame by prediction based on motion compensation. B-frames can be recovered using bi-directional motion compensation. A typical MPEG frame sequence depicting the relationship among I-,P- and B-frames is given in Fig. 2.

Potential speed enhancement arises from the readily available motion vectors in successive frames. In Fig. 2, each group of pictures (GOP) contains nine frames. The proposed compressed-domain NPR algorithm works as follows: We will first apply still image NPR effect to the I-frame. We then propagate results obtained in I-frame to other frames in the group using motion compensation. This will reduce the flickering since inter-frame motion has been considered. In addition, because full-frame computation has to be done only once, the proposed method will be computationally efficient. Of course, we also need to consider the situation when predictive coding fails. This happens when the prediction error exceeds a certain threshold. In this case, the macroblock itself has to be encoded, and NPR on this portion needs to be performed independently.

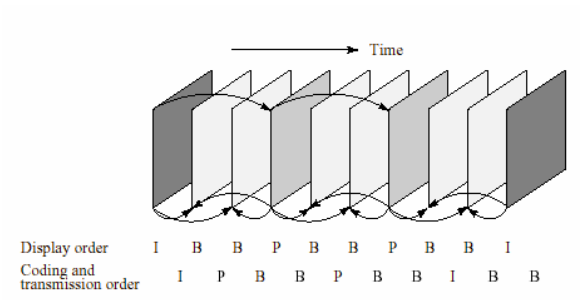

Fig.2. MPEG sequence.

To summarize, there are four important issues we need to take into account in compressed domain processing:

1. Manipulating DCT coefficients of the macroblocks of I-frame is straightforward. However, most NPR operations are non-linear in nature. They can be performed only in the spatial domain. Very limited NPR effects can be created by changing the DCT coefficients.
2. The routine process is to decompress I-frame and apply NPR in the spatial domain. Effects in other frames of the same GOP can be produced by propagation
3. If the content of successive frames changes drastically, motion compensation will not work well. We may have to apply NPR to P- or B-frames since propagation from I-frame can not compensate for the discontinuity.
4. If the difference between the I-frames of successive GOPs is small, there is no need to re-apply the NPR. NPR of the previous I-frame will serve as the source for effect propagation.

3 NPR in the Compressed Domain

In this section, we will use MPEG-1 video to illustrate NPR creation and propagation processes in the compressed domain. The picture resolution in MPEG-1 is 352x240 for NTSC video at 30 fps. Each frame is partitioned into macroblocks of size 8x8. We will employ a generic histogram-based algorithm outlined in Fig. 3 to create oil-paint effect. Notice that this algorithm is chosen mainly due to its simplicity so that we can focus on the framework of compressed-domain analysis instead of the details of a particular NPR creation process.

```
  For y= 1: height
    For x=1:width
       find most frequent value M in (x,y)'s
n*n neighborhood
            oil_paint_image(x,y)= M;
```

Fig. 3. Algorithm used to generate oil-paint effect.

3.1 Selective Application of NPR Effects

For the oil-paint effect and many other NPR algorithms, if the region to be processed is uniform, efforts spent on NPR calculation will be futile. We can prevent these redundant operations by checking the variance of each macroblock. And the test can be done in the compressed domain. Let m denotes the mean of the pixel values in a block. In the Discrete Fourier Transform (DFT) domain, we have:

$$m = F(0,0) = \frac{1}{MN} \sum_{x=0}^{N-1} \sum_{y-0}^{M-1} f(x, y) \tag{1}$$

where F is the DFT of f. The variance can be computed according to:

$$\begin{aligned}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1}\left[f(x,y)-m\right]^2 &= \sum_{x=0}^{N-1}\sum_{y=0}^{M-1}f^2(x,y)-2m\sum_{x=0}^{N-1}\sum_{y=0}^{M-1}f(x,y)+\sum_{x=0}^{N-1}\sum_{y=0}^{M-1}m^2 \\ &= \sum_{u=0}^{N-1}\sum_{v=0}^{M-1}F^2(u,v)-2MNm^2+MNm^2 \\ &= \sum_{u=0}^{N-1}\sum_{v=0}^{M-1}F^2(u,v)-MNm^2 \\ &= \sum_{u=0}^{N-1}\sum_{v=0}^{M-1}F^2(u,v)-F^2(0,0)\end{aligned} \tag{2}$$

In other words, the variance is the sum of the squares of the AC coefficients in the compressed domain. Even though Equation (2) is derived using DFT definition, the same conclusion holds in the DCT domain as the basis functions in DCT are orthonormal and Parseval's theorem is valid. Therefore, we can use the rule shown in Fig. 4 to check if NPR computation needs to be performed in a particular block:

```
Variance =sum of all AC^2 in the macro block
if( Variance < threshold )
  ApplyNPR(macro block)
else
  macro block = macro block of source image
```

Fig. 4. Applying NPR selectively

The macroblock in MPEG-1 is of size 8x8. The corresponding DCT coefficients are stored in a zigzag manner. We can regard it as a 1D array. The first element is DC. The others (2-64) are AC coefficients. In the actual implementation, we use Y-channel (in YUV system) information to compute the variance.

Fig. 5 depicts an image before and after applying the oil-paint effect. Figs. 6 and 7 demonstrate the results of selective NPR by setting the threshold to 100 and 1000, respectively. For this particular type of effect, no remarkable visible differences can be detected if the slowly-varying blocks are not processed. Table 1 summarizes the improvements in computation time by selective application of the oil-paint effect.

(a) (b)

Fig. 5. (a) original image (b) with oil-paint effect.

(a) (b)

Fig. 6. (a) sum of AC^2 in green blocks < 100 (b) applying oil-paint effect selectively.

(a) (b)

Fig. 7. (a) sum of AC^2 in green blocks < 1000 (b) applying oil-paint effect selectively.

Table 1. Performance enhancement by selective NPR application

Threshold	0	100	200	1000
Time (sec)	1.09	0.875	0.719	0.65
% of blocks not processed	0%	21.5%	34.5%	45%
Speedup	0%	20%	33%	40%

3.2 Propagation of NPR Effects

We now turn to the discussion of initiating the NPR in I-frame and relying on motion compensation to spread out the effect. To evaluate the performance of different approaches, we use a video segment containing 419 frames with a GOP structure defined according to: IBBPBBPBBPBBPBB. It takes approximately 400 seconds to apply the oil-paint algorithm frame-by-frame. The frame rate is 1.05.

Fig. 8 shows the first frame of the video before and after processing[1]. The spatial domain NPR computation is performed on the I-frame only. Afterwards, the result is

[1] The video is available for download at: http://140.119.164.91/liang/NPR.htm

converted back to the compressed domain. P- and B-frames are reconstructed using the normal decoding process. The total computation is drastically reduced. It requires only 28.4 seconds to process the video, a speedup of approximately 15, corresponding to the size of the GOP group. In addition, flickering has been reduced significantly compared to the frame-by-frame approach.

(a) (b)

Fig. 8. (a) original video (b) I-frame oil-paint effect

Propagation from I-frame does not always work, however. For example, if significant changes occur between consecutive frames, as depicted in Fig. 9, the effect in Fig. 9(a) will not show up in Fig. 9(b). Under such circumstances, we will have to recover the P- and B-frames and apply the NPR accordingly, as demonstrated in Fig. 10. Again, the condition can be checked in the compressed domain since:

$$\|f - g\|^2 = \|F - G\|^2 \tag{3}$$

where f,g are spatial domain and F,G are frequency domain representations, respectively. If we set the difference threshold to be 60% (i.e., 60% of the pixels have changed values), the computation time is increased to 56.5 sec, or 7.7 fps.

(a) (b)

Fig. 9. (a) (b) Consecutive frames with abrupt changes in video content.

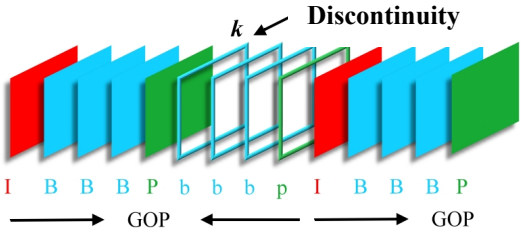

Fig. 10. b and p frames will require independent NPR.

Finally, to investigate the effect of I-frame propagation, we define the percentage difference between two color images I_1 and I_2 as:

$$\%\,Diff \;=\; 100\;x\,\frac{\left|I_1(R)\text{-}I_2(R)\right| + \left|I_1(G)\text{-}I_2(G)\right| + \left|I_1(B)\text{-}I_2(B)\right|}{I_1(R) + I_1(G) + I_1(B)} \tag{4}$$

where R, G, and B refer to the individual color channels. Fig. 11 shows the percentage difference between all-frames and I-frame only NPR (black) as well as all-frames and selective frames NPR (pink). In most frames, both methods give approximately the same result. However, at places where abrupt scene changes occur (frame 255 and frame 417 as indicated by two arrows), selective P,B frame NPR does a better job.

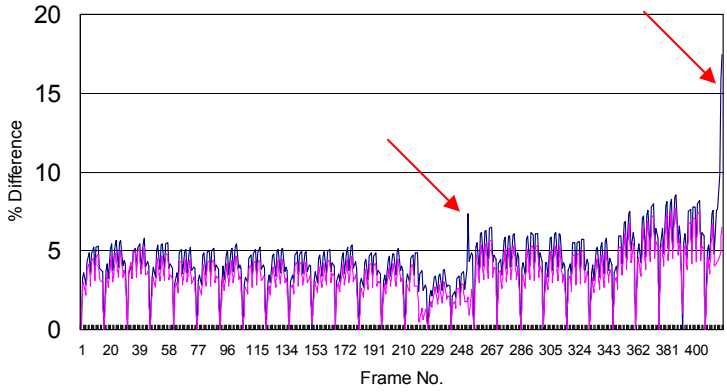

Fig. 11. Percentage difference between all frames and I-frame only NPR (black), all frames and selective frames NPR (pink).

4 Conclusions

In this paper, we have proposed a novel approach to accelerate NPR video effects using compressed domain processing. The enhancement is two-fold: flickering reduction and speedup. We have used MPEG-1 video to demonstrate the NPR generation and propagation process. We have also developed a selective NPR scheme to further reduce the computation time. Even though only a generic NPR algorithm has been investigated, we believe that similar improvements on other types of NPR can be obtained using the proposed methodology.

5 References

[1] A. Hertzmann, K. Perlin, “Painterly rendering for video and interaction”, Proceedings of NPAR 2000: 7-12.

[2] J. Wang, Y. Xu, H. Shum, M. F. Cohen , “Video Tooning”, ACM Trans. on Graphics (Proc. of SIGGAPH 2004), Vol. 23, No. 3, p.574-583, 2004.

[3] DDPlayCam: http://www.ddplaycam.tv

[4] Wang, H., Divakaran, A., Vetro, A., Chang, S.F., Sun, H., "Survey of compressed-domain features used in audio-visual indexing and analysis", Journal of Visual Communication and Image Representation(14), No. 2, June 2003, pp. 150-183.