Minimum-Drift Digital Video Down-Conversion[1]

Osama Alshaykh
PacketVideo Corporation
10350 Science Center Dr. STE 140
San Diego, CA 92121
osama@packetvideo.com

Homer Chen
Rockwell Science Center
1049 Camino Dos Rios
Thousand Oaks, CA 91320
homer@risc,rockwell.coml

ABSTRACT

This paper presents a new technique for decoding a full-resolution video bitstream at low memory cost and displaying the signal at a lower resolution. Existing techniques solve the problem by storing the down-converted blocks into memory instead of the full-resolution blocks. While the memory is reduced, these techniques introduce drift errors because the decoder does not have the same pixels as the encoder in performing motion-compensated prediction. The approach proposed here alleviates the problem by tracking the drift at the decoder. It improves the video quality without any increase in decoder complexity. The effectiveness of the approach is evaluated using both objective and subjective tests. This minimum-drift approach is very simple to implement and can also be applied for memory reduction of a full resolution HDTV decoder.

Keywords

MPEG-2, HDTV, SDTV, Format Conversion.

1. INTRODUCTION

The Federal Communications Commission's requirement to start broadcasting digital television (DTV) in Fall 1998 will expedite the development of affordable DTV receivers that can accommodate the transition from conventional television to DTV. The cost of DTV receivers is driven, in part, by the large memory needed to buffer the decoded pictures for motion-compensated prediction. To enable widespread acceptability of DTV in the consumer electronics market, the decoder memory must be effectively reduced.

This paper addresses the memory reduction issue of DTV down conversion. A DTV down- converter decodes a full-resolution digital video signal and displays it at a lower resolution. Such down-conversion is needed, for example, for viewing high definition television (HDTV) materials with a standard definition television (STDV) monitor or for generating the picture-in-picture special effect. The goal here is to achieve the highest possible memory reduction while preserving the picture quality. The down-conversion problem can be solved by fully decoding the HDTV signal and then post-processing and down-sampling the resulting images. Although this approach is able to achieve the best picture quality, it requires full-resolution memory to store the decoded images and extra computations to down-sample the images.

The down-conversion problem has been investigated in the past, and various approaches have been proposed. In most approaches, the memory reduction is achieved by storing the down-converted images in memory. For example, Vetro *et al.* [1], [2] proposed to store the lower-resolution frame and process the residual by either masking the high frequency coefficients of a block (the cut approach) or by combing the DCT coefficients of neighboring blocks (the synthesis approach). Since the motion compensation is based on the down-converted images, the decoder does not have the same pixels as the encoder for motion-compensated prediction. As a result, the quality of decoded images degrades (the drift problem).

We present a new technique to reduce the drift. The basic idea is to minimize the accumulation of drift errors by tracking the decoded pixels. The technique is more effective than existing approaches available in the literature without any increase in complexity. (In fact, the complexity is sometimes slightly decreased.) The effectiveness of the proposed approach is tested objectively and subjectively. The subjective test shows that 95% of the 80 viewers (non-experts in video compression) participating in the test rank this approach better than the frequency-domain cut and synthesis approaches. The other 5% are unable to tell the difference.

This paper is organized as follows: Section 2 describes the drift problem caused by memory reduction in an MPEG decoder. Section 3 describes our new technique for solving the drift problem. Section 4 demonstrates the effectiveness of the technique and provides a complexity analysis. A comparison of the performance of the technique with other existing methods is also presented. Finally, Section 5 concludes the paper.

2. PROBLEM DESCRIPTION

Figure 1 shows the simplified block diagram of a typical video decoder with a decoding loop that performs motion compensation and prediction. For the decoder to produce the same video as

[1] This work was performed while the first author was with Rockwell Science Center.

intended by the encoder, each reference frame used by the decoder for motion-compensated prediction must be the same as the one used by the encoder. This is how a motion-compensated digital video system works. Thus the decoder memory should be large enough to store the reference frames. In a typical MPEG-2 decoder, for example, 3 frame stores are needed. For the purpose of cost reduction, however, a down-converter reduces the memory and only stores the down-converted version of the reference frames in the memory instead of the full-resolution frames. The resulting mismatch between the decoder and encoder penalizes the decoded video quality and causes a drift problem. An illustration of the drift problem is given in Figure 2, where a video frame generated by perfect motion-compensated prediction is compared against a decoded video frame generated by using a sub-sampled reference frame. As can be seen, the down converted picture quality can be seriously degraded.

3. DOWN CONVERSION USING DRIFT TRACKING

This section describes the drift-tracking approach to DTV down-conversion. The approach involves two basic operations: image subsampling and drift tracking. The image subsampling is done within the motion-compensated prediction loop to reduce the memory required, while the drift tracking is performed to preserve the reference frame and thereby reduce the impact of image subsampling on the integrity of motion-compensated prediction.

For the purpose of discussion, denote the decoded current frame, the decoded previous frame, the residual error, and the motion compensation operation by *f(n)*, *f(n-1)*, *r(n)*, and *MC(.)*, respectively. For a full-frame decoder, we have

$$f(n) = MC(f(n-1)) + r(n).$$

For a down converter that performs in-loop subsampling (that is the down-converted frame instead of the full frame is stored in memory), we have

where *SUB(f)* denotes the sub-sampled version of frame *f*. Clearly, the output picture of the down-converter will be the same as the subsampled version of the output picture of the full frame decoder (and hence drift-free) if

For this equation to hold, the reference image block used for motion compensation in the down-converter must be identical to the subsampled version of the corresponding image block in the full-frame decoder. Unfortunately, this relationship does not hold in reality.

The difference between the two sides of the above equation is the drift error. Suppose the image is subsampled by a factor of 1/2 in each dimension, three quarters of the image pixels are thrown away. Note that an encoder computes the motion vector of an image block based on the full frame. In the decoding, if the reference pixels that are needed for reconstructing a pixel of the current frame are among the thrown pixels, the pixel will start to drift. The drift error accumulates through all predictive frames until the next intra-coded frame or block, where the decoding loop is re-initialized.

We propose to solve the drift problem by compensating the drift error in decoding each pixel. This is achieved by computing the drift error of each reference frame pixel and storing it in memory. The drift error can be tracked by, for example, a procedure similar to Σ–Δ modulation [3]. Let x be the frame to be downsampled by using an anti-aliasing filter h and a sub-sampler. We first store the down-sampled frame y, where

$$y(m,n) = \sum_{k,l} h(k,l)x(m-k,n-l).$$

Then, for each 2×2 block of pixels (three of which are to be thrown away), we compute the following values:

$$y(m,n) = \sum_{k,l} h(k,l)x(2m-k,2n-l),$$
$$\partial y(2m,2n) = 0,$$
$$\partial y(2m+1,2n) = Q(y(m,n) - x(2m+1,2n)),$$
$$\partial y(2m,2n+1) = Q(y(m,n) - x(2m,2n+1)), \text{ and}$$
$$\partial y(2m+1,2n+1) = Q(y(m,n) - x(2m+1,2n+1)).$$

The function Q is a quantizer. That is, we store the quantized differences between the pixel to be stored and the three pixels to be thrown away. In our current implementation, we choose to use very simple quantizer and entropy coder to reduce the complexity. The scalar quantizer shown in Figure 3 is used to quantize the drift error. It is important to note that, for visual quality, a trailing-edge reconstruction quantizer is recommended.

However, note that the proposed technique does not depend on any particular type of filters. For simplicity, we use box averaging in our implementation. Then the above equations become

Figure 4 shows a block diagram of the overall down conversion algorithm. The decoder decodes and down-samples a full block, one at a time. The down-sampled block is stored into the frame memory. In parallel to that, the decoder also stores the quantized differences between the stored pixels and the thrown ones in the drift memory. The pixels and the quantized differences in the two memory buffers are used to construct motion-compensation blocks. In other words, the new decoding loop performs both motion compensation and drift compensation.

$$y(m,n) = (x(2m,2n) + x(2m,2n+1) + x(2m+1,2n) + x(2m+1,2n+1))/4,$$
$$\partial y(2m,2n) = 0,$$
$$\partial y(2m+1,2n) = Q(y(m,n) - x(2m+1,2n)),$$
$$\partial y(2m,2n+1) = Q(y(m,n) - x(2m,2n+1)), \text{ and}$$
$$\partial y(2m+1,2n+1) = Q(y(m,n) - x(2m+1,2n+1)).$$

The extra storage of this algorithm is the memory required for storing the quantized differences for the thrown pixels. In our approach, the quantized differences are coded using a variable length coder (VLC). In effect, the quantizer controls the quality of the picture and the extra storage, and the entropy coder determines the amount of memory required by the down converter. Overall, the quantizer and the entropy coder determine the overhead complexity of the down converter.

To further increase the coding efficiency, a bit is used for each 8×8 block to indicate if all the pixels in the block are zero or not. This is because most background blocks are zero blocks. The addition of this additional bit also simplifies the decoding of the quantized differences.

4. RESULTS

The algorithm is implemented on an MPEG-2 decoder. The effectiveness of the algorithm is demonstrated by decoding different CCIR image sequences. Moreover, the performance of the algorithm is compared to the cut and synthesis methods described in [1], [2]. It is also compared with the post-processing approach that decodes and stores the full video frame. In all approaches, we use the box averaging as a down-sampling filter. Other filters can be used, but the conclusion drawn in this section holds for different filter choices. The measured PSNR reflects the difference with the non-drift case, i.e., the box averaging method.

The cut and the synthesis algorithms need one-fourth of the memory required by a full-frame decoder. Both algorithms filter the residual DCT block before adding it to the motion compensated block. In the cut method, a mask is used to mask the high frequency coefficients of a block, while the synthesis method combines the DCT coefficients of four neighboring blocks to produce a single DCT block.

4.1 Quality

The performance of the proposed approach is demonstrated by decoding the CCIR video sequences to one quarter of their original size. In other words, the image is subsampled by a factor of one-half horizontally and vertically. In all experiments, the additional memory (compared to the cut and synthesis methods) required by the drift-tracking algorithm is at most 1 bit per pixel (bpp) for tracking the drift. Table 1 shows the average memory required and the resulting PSNR for the CCIR sequences tested. The PSNR is measured against the results of the post-processing approach, which is drift-free. The drift-tracking algorithm used a scalar quantizer with quantization value of 16 and a fixed, non-trained 5-element VLC table. Table 1 shows that with small extra memory the objective quality improves. The subjective quality also improves as shown in Figure 5.

4.2 Memory Requirements

The system can set the maximum memory allowance for the drift-tracking algorithm. For all experiments in Table 1, the maximum memory used is 0.83 bpp for the Mobile-calendar sequence. Suppose we allow a maximum of 1 bpp for the drift memory. The drift-tracking algorithm would require a total of 11.0592 Mbits of memory for 4:2:0 format, 720-lines×1280-pixels (720P) sequences, whereas the post-processing decoders would require 33.1776 Mbits of memory. The cut and synthesis methods need one quarter of that amount, i.e., 8.2994 Mbits. For 4:2:2 format, 720-lines×1280-pixels progressive sequences, the post-processing method needs 44.2368 Mbits of memory, the cut and synthesis methods need 11.0592 Mbits, and the drift-tracking algoirthm needs 13.824 Mbits.

The quantizer controls the memory requirement for the drift-tracking algorithm. To see the effect of the quantizer on the subjective quality, Figure 6 shows a down-converted Mobile-calendar video frame generated by using different quantizers, while the effect on objective quality is shown in Figure 7.

4.3 Computational Requirements

The algorithm is quite simple. The complexity of the algorithm depends on the quantizer, the entropy coder, and the filter used for down-sampling [4], [5]. In our implementation we use the box averaging filter. We use a power-of-2 scalar quantizer (16). These two steps require 12 additions and 8 shifts. The algorithm also needs a parser and de-parser to code the quantized differences. We use a 5-element VLC table, so it is very simple. However, the actual computational cost is image dependent; it depends on how many zero blocks exist.

Table 2 summarizes the complexity of all algorithms. The following assumptions are made in the comparison:

1. For the post-processing approach, a 7-4 separable filter is used.
2. Video sequences are of 4:2:0, 720×1280 progressive format.
3. The memory cost of the drift-tracking approach is 1bpp.

To provide a fair comparison, the numbers of arithmetic operations (additions, shifts, and multiplies) are calculated on the pixel basis. These numbers are obtained by dividing the total number of operations by the number of pixels in a down-sampled frame

5. CONCLUSIONS

We have described a new technique for DTV down-conversion. The down-conversion is performed inside the motion compensation loop of a video codec to achieve memory reduction, while the pixels discarded by the down-conversion are tracked to preserve the integrity of motion-compensated prediction. The algorithm is compared against two competing methods and a baseline method. The results show that on average, the proposed algorithm outperforms the competing methods by 7.6 dB in PSNR. This drift-tracking approach, which is simple and computationally efficient, can also be easily applied to reduce the memory of full-resolution HDTV or MPEG decoders [6], [7].

6. ACKNOWLEDGMENTS

The authors thank Anthony Vetro and Huifung Sun of Mitsubishi for helpful discussions on the cut and synthesis techniques for DTV down conversion. They also thank Alex Wang for implementing these two techniques on an MPEG-2 decoder.

7. REFERENCES

[1] A. Vetro and H. Sun, "Frequency domain down conversion of HDTV using optimal motion compensation," *Int'l Journal*

Imaging Systems and Technology, vol. 9, no. 4, pp. 274-282, 1998.

[2] A. Vetro and H. Sun, "On the motion compensation within a down conversion decoder," *Journal of Electronic Imaging*, vol. 7, no. 3, July 1998.

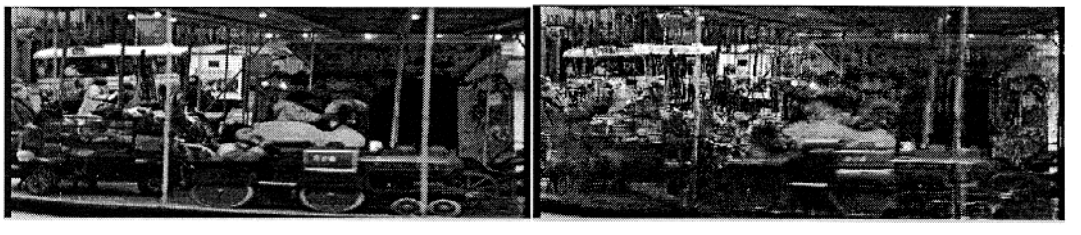
[3] A.V. Oppenheim and R.W. Schafer, *Discrete Time Signal Processing*, Prentice Hall, 1998.

[4] W.B. Pennebaker and J.L. Mitchel, *JPEG: Still Image Compression Standard*, Van Nostrand Reinhold, New York, NY, 1993.

[5] S. Merril Weiss, *Issues in Advanced Television Technologies*, Focal Press, Newton, MA, 1996.

[6] R. Bruni, A. Chimienti, M. Lucenteforte, D. Pau, and R. Sannino, "A novel adaptive vector quantization method for memory reduction in MPEG-2 HDTV decoders," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 537-544, August, 1998.

[7] U. Bayazit et al., "A novel memory compression system for MPEG-2 decoders," *Proc. Int'l Conf. Consumer Electronics*, pp. 56-57, 1998.



(a) (b)

Figure 2: Illustration of the drift problem: (a) The image is subsampled after the full-resolution video frame is generated, and (b) The image is generated by a down converter using subsampled reference frames in the decoding loop.

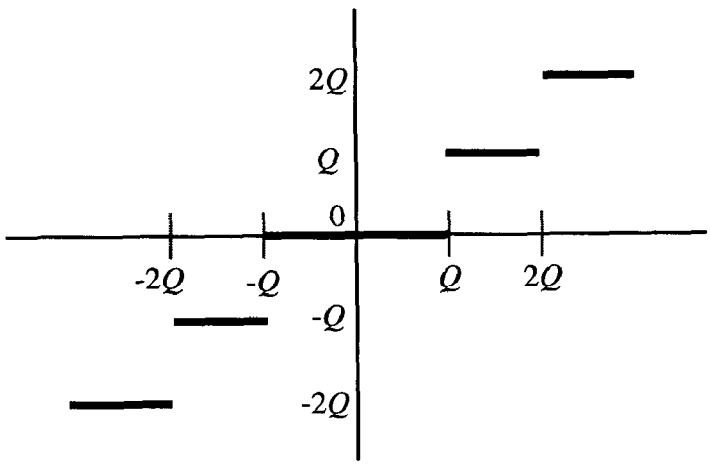


Figure 3. The quantizer for quantizing the drift error.

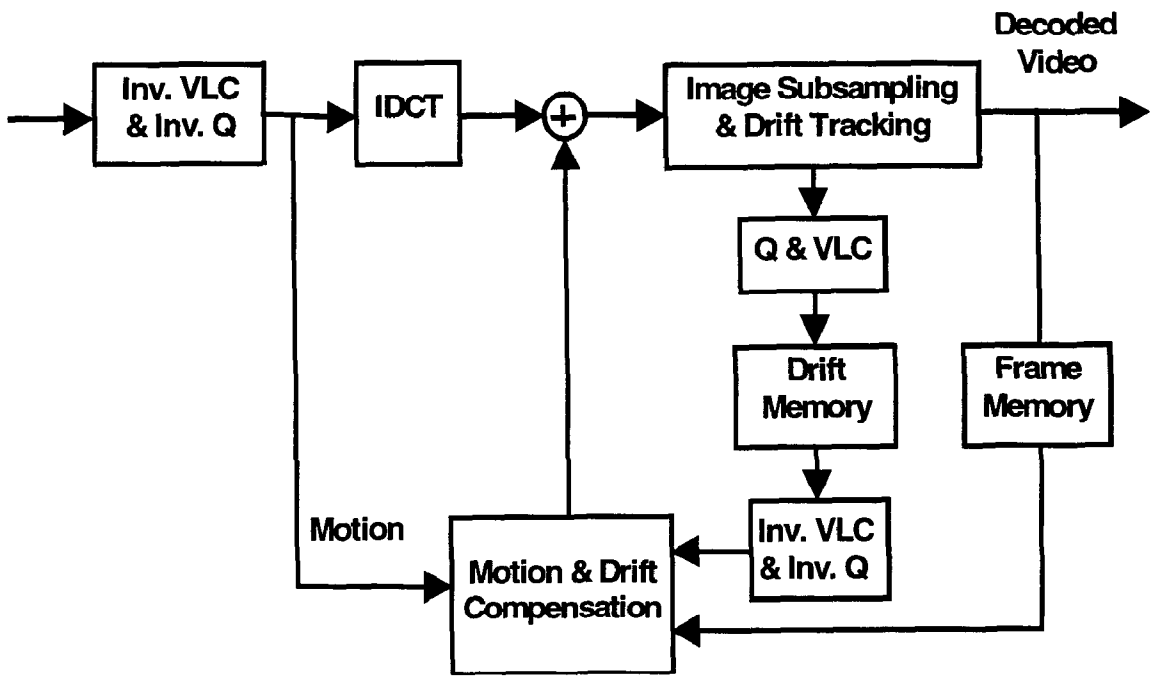


Figure 4. A block diagram of the drift-tracking approach for DTV down conversion.

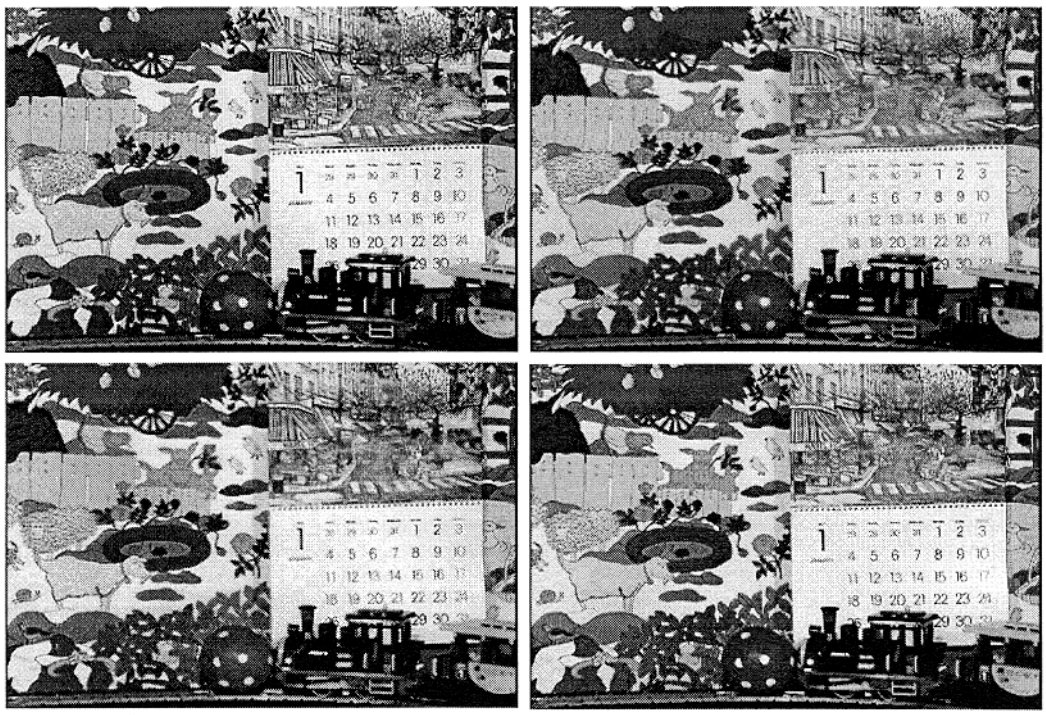


Figure 5: Frame 89 of the Mobile-calendar sequence generated by the post-processing (top-left), cut (top-right), synthesis (bottom-left), and drift-tracking (bottom-right) methods (with Quantizer=16 and a 5-element VLC table).

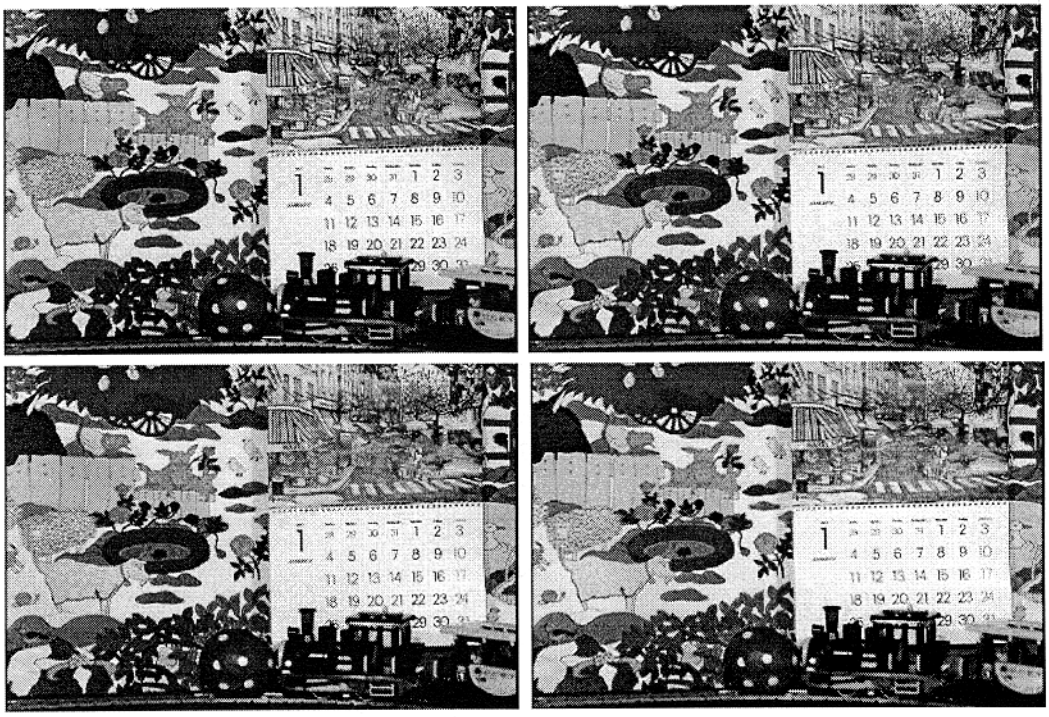


Figure 6: Frame 89 of the Mobile-calendar sequence decoded using the drift-tracking algorithm. From top to bottom, left to right: Q=6 (1.89 bpp), Q=13 (0.988 bpp), Q=25 (0.556 bpp), and Q=32 (0.289 bpp), where Q is the quantizer.

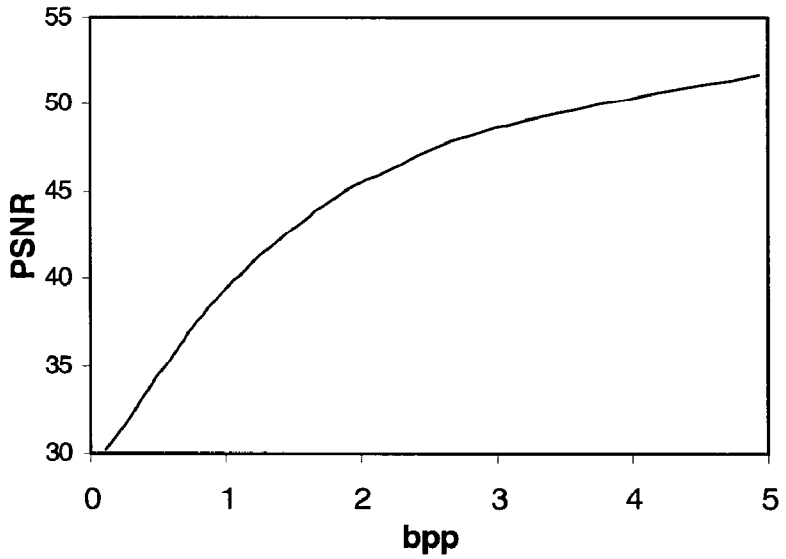


Figure 7: PSNR versus bits needed per pixel and per frame for the Mobile-calendar experiment. Total memory needed is equal to bpp*720*480*3+4.147 Mbits. Uniform quantizers are used with operation range [-128, 128].

Table 1: PSNR and memory comparisons.

Sequence	Rate	PSNR (dB)			Memory (Mbits)		
		Cut	Syn.	Tracking	Post-Proc.	Cut/Syn.	Tracking
Akiyo	5Mbps	42.53	41.97	48.26	16.796	4.199	4.337
Coast-guard	5Mbps	35.05	34.65	40.68	16.589	4.147	4.532
Container-ship	5Mbps	35.56	34.48	45.13	16.589	4.147	4.454
Mobile-calendar	5Mbps	28.93	28.70	37.24	16.589	4.147	5.005
Stefan	3Mbps	33.17	33.00	41.86	16.589	4.147	4.614

Table 2: Complexity comparison.

Measure	**Post-Proc.**	**Cut**	**Synthesis**	**Drift-Track**
Memory (Mbits)	33.1766	8.2994	8.2994	11.0592
Additions	6	11	267	12
Shifts	0	7	7	8
Multiplies	9	0	64	0
VLC/DVLC	No	No	No	Yes