CD ROM Title and Abstract

A New Approach to Decoding and Compositing Motion-Compensated DCT-Based Images

Shih-Fu Chang and David G. Messerschmitt
University of California, Berkeley, CA

Multi-point network video services require compositing several video sources into a single displayed video stream. Compositing video directly in the compressed domain can save computations by processing less data and avoiding the conversion process back and forth between the compressed and uncompressed data formats. Earlier work has demonstrated the computational speedup of compositing DCT-compressed video directly in the DCT domain, compared to the straightforward spatial-domain approach. Typical compositing operations include overlapping, scaling, translation, filtering, etc. In this paper, we propose a new decoding algorithm for MC-DCT compressed video which converts MC-DCT compressed video to the DCT domain and enables video compositing in the DCT compressed domain. Computational complexity is analyzed. The idea of network compositing and its impacts on multimedia network implementations are also discussed.

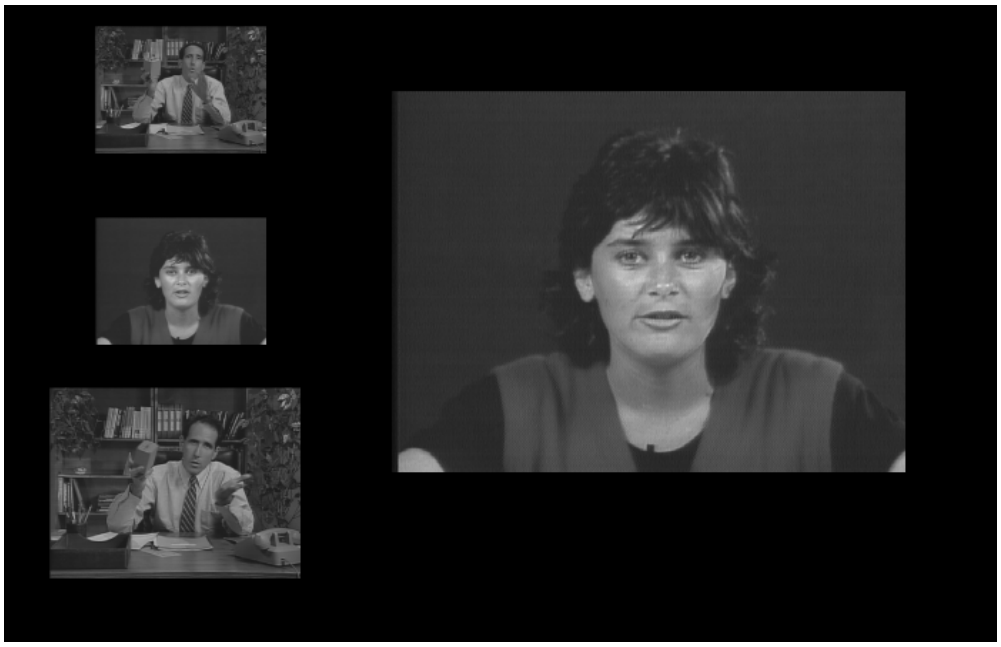


Fig. 6. A typical compositing scenario in video conferencing. The input images are scaled down with different ratios and translated to different positions. The image shown is reconstructed from the compressed output which is composited in the DCT domain. The input images are compressed with MC+DCT.

decompression, and increased latency for transmitting video to end users. Our experiments show that accumulation errors depends on the compositing operations performed, for example, more severe degradation for scaling, ignorable degradation for translation and overlapping.

We have simulated the DCT-domain compositing algorithms for different compositing scenarios and quality levels. For the compositing scenario shown in figure 6, our proposed DCT-domain algorithm can save the computations by 24%, compared to the straightforward spatial-domain approach. If the compression is without the MC algorithm (like JPEG), the reduction of computations is about 73%. The reconstructed images do not show clear subjective difference between our compressed-domain approach and the spatial-domain approach. (The PSNR is 42.8 dB vs. 43.2 dB.) The non-zero DCT coefficient percentage varies between 5%~22%, and the non-zero motion vector percentage ranges between 12%~90% in our simulations. The speedup can be increased if we raise the compression ratio by sacrificing some image quality.

4. Conclusions and Future Work

We design a new decoding algorithm for the MC-DCT based video, which performs inverse MC before inverse DCT. This algorithm can be applied in compositing compressed video within the network, which may take multiple compressed video sources and combine them into a single compressed output stream. The proposed algorithm converts all MC-DCT compressed video into the DCT domain and performs compositing in the DCT domain. This DCT-domain approach can reduce the required computations with a speedup factor depending on the compression ratio and the non-zero motion vector percentage. However, dropping some least-significant DCT coefficients maybe necessary for the worst case of high-motion video in real-time implementations.

Some issues of networked video compositing are also discussed. Another direct application of the proposed decoding algorithm is converting MC-DCT compressed video to the DCT compressed format directly in the DCT domain.

5. References

[1] S.-F. Chang and D.G. Messerschmitt, “Compositing Motion-Compensated Video within the Networks,” IEEE 4th Workshop on Multimedia Communications, Monterey, CA, April, 1992.

[2] S.-F. Chang, W.-L. Chen, and D.G. Messerschmitt, “Video Compositing in the DCT Domain,” IEEE Intern. Workshop on Visual Signal Processing and Communications, Raleigh, North Carolina, September, 1992.

[3] B.C. Smith and L. Rowe, “A New Family of Algorithms for Manipulating Compressed Images,” Submitted to IEEE Computer Graphics and Applications.

[4] J.B. Lee and B.G. Lee, “Transform Domain Filtering Based on Pipelined Structure,” IEEE Trans. on Signal Processing, pp.2061-4, Vol. 40, No. 8, Aug. 1992.

[5] W. Kou and T. Fjallbrant, “A Direct Computation of DCT Coefficients for a Signal Block from Two Adjacent Blocks,” IEEE Trans. on Signal Processing, Vol. 39, No. 7, pp. 1692-5, July 1991.

[6] CCITT Recommendation H.261, “Video Codec for Audiovisual Services at p$\times$64 kbits/s”

[7] Standard Draft, MPEG Video Committee Draft, MPEG 90/176 Rev. 2, Dec. 1990.

[8] W.-H. Chen, C.H. Smith, and S.C. Fralick, “A Fast Algorithm for the Discrete Cosine Transform,” IEEE Trans. on Communications, Vol. COM-25, No. 9, Sep. 1977, pp. 1004-9.

[9] B.G. Lee, “FCT-A Fast Cosine Transform,” IEEE ICASSP ‘84, San Diego, pp. 28A.3.1-4, March, 1984.

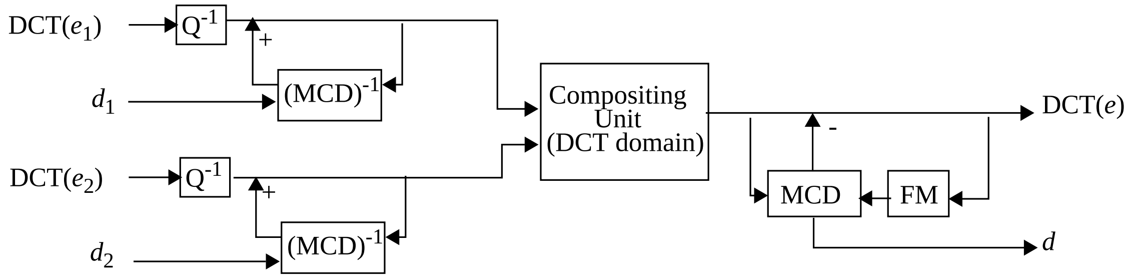


Fig. 4. Compositing two MC-DCT-based video sequences in the DCT domain. We convert input sequences to the DCT compressed domain, composite them in the DCT domain, and then convert the composited video to the original compressed format (i.e. the MC-DCT format).

them pixel by pixel. Our proposed method is to use techniques described in the previous section to convert images to the DCT domain and then composite them in the DCT domain, as shown in figure 4. The DCT and inverse DCT blocks in the traditional approach are removed, and the compositing unit needs to process less data in the DCT domain than the spatial domain. Note that in order to encode the composited video to the original full MC-DCT compressed format, the composited video still needs to be motion-compensated, as shown in the MCD block in figure 4. To prevent this block becoming the most computation-dominant in the whole compositing system, we can infer new motion vectors directly from those of original input images [1]. The required computations for motion measurement can thus be greatly reduced at the cost of little compression performance.

As mentioned above, the computational complexity for the proposed decoding algorithm (including the DCT-domain MC^{-1} only) depends on both the non-zero percentages of the motion vector and the DCT coefficients. We compare this complexity to that for the traditional decoding algorithm, which includes a fast DCT followed by an inverse MC in the spatial domain. We use the fast DCT proposed by Chen *et al* [8] in table 1. Other fast algorithms for DCT have similar or larger complexity [9]. In table 1, N is the block size, β represents the reciprocal

Table 1: Comparison of computational complexity between the traditional full decoder and the proposed decoder for MC-DCT video.

	total number of real multiplications and additions
traditional decoder	$2 \cdot \log_2 N + 8/N - 3$ (mul.) $3 \cdot (\log_2 N - 1) + 4/N + 1$ (add.)
proposed decoder (DCT-domain MC^{-1} only)	$(4/\beta + 2/\sqrt{\beta}) \cdot N \cdot \alpha_2 + (2/\beta) \cdot N \cdot \alpha_1$ (mul.) $[(4/\beta + 2/\sqrt{\beta}) \cdot N + 3] \cdot \alpha_2 + [(2/\beta) \cdot N + 1] \cdot \alpha_1 + 1$ (add.)

of the non-zero DCT coefficient percentage, α_2 represents the percentage of motion vectors which needs block alignment in both directions, and α_1 represents the percentage of motion vectors which requires block alignment in one direction only. We use efficient methods for calculating sparse matrix multiplication in equation 3. As we can see, the most significant computation

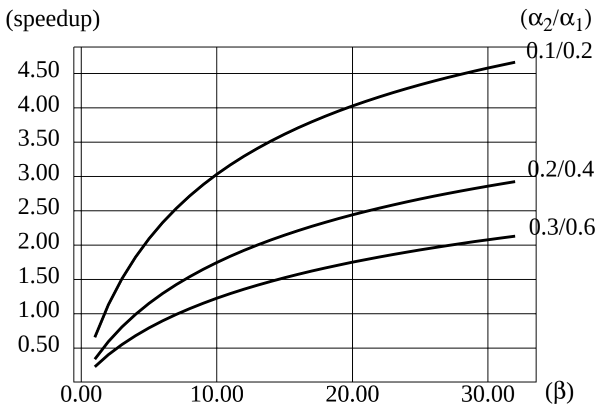
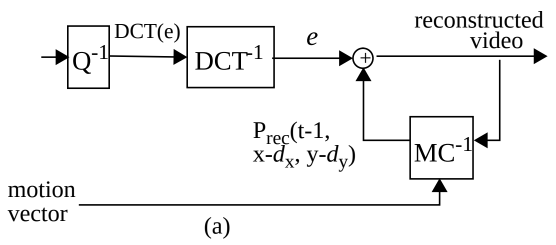


Fig. 5. Computational speedup by the proposed decoder (i.e. DCT-domain MC^{-1}), compared to the traditional full decoder for MC-DCT video.

component for the proposed decoding algorithm is proportional to α_2, which indicates the frequency of full computation of equation 3. In figure 5, we illustrate the speedup of using the proposed decoding algorithm vs. the traditional spatial-domain decoder for some different compression parameters. For a typical β value of about 10, the speedup ranges between 1.2 and 3.0 for low-motion video sequences, e.g. head-and-shoulder video.

However, for high-motion video sequences (i.e. high α_1/α_2 values), the DCT-domain approach could become more complicated than the spatial-domain approach. This variable throughput will become an obstacle for real-time implementation which should be able to handle the worst-case situation. Fortunately, in the DCT domain, the compositing unit has the flexibility to ignore some high-order DCT coefficients during the worst case with less image quality degradation. This freedom of ignoring low-priority data is not available in the spatial domain. However, its effect on the recovered image quality needs to be investigated.

Besides the needs for compressed-domain compositing, compositing within the network has other impacts on designs of multimedia networks. The transmission cost can be reduced if multiple video sources are composited together before they are transmitted to the end users. Composited scenes can be shared among different users who subscribe the same service. Possible drawbacks for compositing video within the network are accumulation of quantization errors after repeated compression and



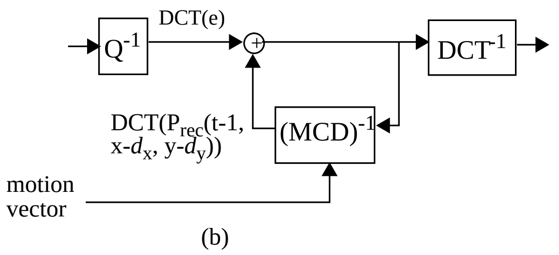


Fig. 1. (a) Traditional decoders for MC-DCT-based videos. (b)A new decoding algorithm, in which the inverse MC algorithm is performed in the DCT domain. MCD: MC in the DCT domain.

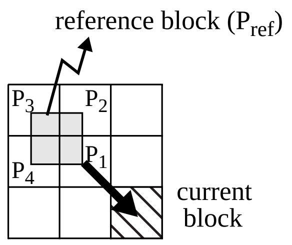


Fig. 2. The MC optimal reference block may overlap with four neighboring blocks in the DCT block structure.

all coefficients are grouped into a block-based format. The optimal reference image block may overlap with four neighboring blocks in the DCT block structure, as shown in figure 2. Kou [4] proposed an algorithm to compute DCT coefficients of a new block from two adjacent blocks, but it becomes very complicated when the overlap length is not equal to half of the block width. In [2], we have designed an algorithm to calculate the DCT coefficients of a new arbitrary-position image block directly from the DCT coefficients of four original neighboring blocks. The operations include multiplications with pre-matrices and post-matrices —

$$DCT\left(P_{ref}\right) = \sum_{i=1}^{4} DCT\left(H_{i1}\right) DCT\left(P_i\right) DCT\left(H_{i2}\right) \qquad \text{(EQ 3)}$$

where P_i are original neighboring image blocks and H_{ij} are special sparse matrices like

$$\begin{bmatrix} 0 & 0 \\ I_h & 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 & I_w \\ 0 & 0 \end{bmatrix}$$

whose DCT coefficients can be pre-computed and stored in the memory (h and w are overlap width). The net effects of matrix multiplication with H_{ij} are combination of *windowing* and *shifting*.

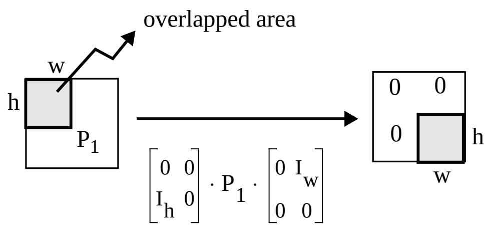

overlapped area
w
h
P_1
$\begin{bmatrix} 0 & 0 \\ I_h & 0 \end{bmatrix} \cdot P_1 \cdot \begin{bmatrix} 0 & I_w \\ 0 & 0 \end{bmatrix}$
0 0
0
h
w

Fig. 3. Use simple matrix multiplications to perform windowing and shifting operations on image blocks.

As shown in figure 3, the upper-left corner of P_1 is extracted (windowing) and shifted to the lower-right corner of the desired block, P_{ref}. The four summation components of equation 3 account for contributions from four original overlapped blocks, whose DCT coefficients are known. Due to the distributive property of matrix multiplication to DCT, we can perform the windowing and shifting operations directly in the DCT domain, as shown in equation 3. Therefore, the DCT coefficients of the arbitrary-position optimal reference block, i.e. DCT(P_{rec}(t-1, x-d_x, y-d_y)) can be calculated directly in the DCT domain. The DCT coefficients of the current frame can be obtained by adding the DCT of the prediction errors to the DCT of the reference block. The whole inverse MC in the DCT domain is represented by the MCD^{-1} block in figure 1. Fully decoded video can be obtained after applying the inverse DCT. Note, besides the numerical round-off errors, the proposed decoding approach is mathematically equivalent to the traditional decoding approach.

The required computations of equation 3 can be greatly reduced when many DCT coefficients of P_i are zero, which can be indicated by the run length code (RLC). Further, if the motion vectors are zero or integer multiples of the block width, the above block adjustment procedure can be avoided, and the inverse MC procedure requires simple additions only. Therefore, the computational complexity of the proposed DCT-domain inverse MC algorithm increases with the percentage of the non-zero motion vectors. We will discuss the complexity further in the next section. The storage overhead for storing the pre-computed DCT coefficients of H_{ij} matrices is very small. For a block size of 8×8, a memory of 896 words (one word for each DCT coefficient) is needed.

One immediate application of this new decoding algorithm is compression format conversion, e.g. from MPEG to JPEG. The DCT coefficients of every frame can be obtained by applying the DCT-domain inverse MC directly. The conversion process back and forth between the DCT domain and the spatial domain is avoided. Another application is to composite images in the DCT domain, which will be described in the next section.

3. Compositing Networked Video in the DCT Domain

In a multi-point networked video service like video conference, multiple video sources can be composited within the network. The compositing unit takes the compressed video inputs, composites them, and then produces the compressed composited video output. The traditional straightforward approach reconstructs all images fully back to the spatial domain and composite

A New Approach to Decoding and Compositing Motion-Compensated DCT-Based Images

Shih-Fu Chang and David G. Messerschmitt

Dept. of EECS, University of California, Berkeley, CA 94720

Abstract

Multi-point network video services require compositing several video sources into a single displayed video stream. Compositing video directly in the compressed domain can save computations by processing less data and avoiding the conversion process back and forth between the compressed and uncompressed data formats. Earlier work has demonstrated the computational speedup of compositing DCT-compressed video directly in the DCT domain, compared to the straightforward spatial-domain approach. Typical compositing operations include overlapping, scaling, translation, filtering, etc. In this paper, we propose a new decoding algorithm for MC-DCT compressed video which converts MC-DCT compressed video to the DCT domain and enables video compositing in the DCT compressed domain. Computational complexity is analyzed. The idea of network compositing and its impacts on multimedia network implementations are also discussed.

1. Introduction

Advanced networked video services require real-time video compression, compositing and transmission. Examples of such services are multi-point video conferencing and interactive networked video. Multiple video sources are compressed, transmitted through the network, and composited into a single displayed video stream. The compositing hardware can be located at the user site or an intermediate node (within or outside the network). In the latter case, the composited video needs to be compressed again for further transmission. It is our goal to design compositing algorithms directly in the compressed domain.

Many video compression methods, like H.261 and MPEG, include both the Discrete Cosine Transform (DCT) and the Motion Compensation (MC) algorithms, in which the former removes the spatial redundancy and the latter removes the temporal dependance between frames. It has been shown that compositing the DCT-compressed images directly in the DCT domain can save computations for many compositing operations (e.g. overlapping, scaling, translation and linear filtering), compared to the straightforward spatial-domain approach which composites images pixel by pixel [2,3]. However, obstacles exist preventing image compositing in the MC domain [1]. An example obstacle is that the reference image blocks of the background image in an overlapping scene could be replaced by the foreground image. Video sequences need to be inverse-motion-compensated before composition, and may need to be re-motion-compensated afterwards. This may make real-time high-speed implementations difficult. In [1], we propose some heuristics for calculating new motion vectors of the composited video based on those of the original input video. Thus, the most computation-intensive operation — motion measurement, can be avoided with little performance degradation. For MC-DCT-compressed video, the DCT algorithm is applied on the prediction errors of the MC algorithm. The traditional decoder decodes DCT first and then computes inverse MC. Due to the above mentioned obstacle of compositing video in the MC domain, this decoding algorithm requires full decompression of the MC-DCT algorithm all the way back to the spatial domain and compositing in the spatial domain.

To reduce the computational complexity, we propose a new decoding algorithm for MC-DCT-based video, which performs the MC and inverse MC algorithms in the DCT domain rather than in the spatial domain. We then use this new algorithm to convert all MC-DCT-based input images to the DCT domain and use efficient algorithms proposed in [2] to composite them in the DCT domain. Another application of the proposed decoding algorithm is efficient image format conversion between different compression standards, like JPEG and MPEG. Note if the input video is further compressed with the Huffman code like that in most video compression standards, the front-stage Huffman decoder is still necessary.

2. New Decoding Algorithms for MC-DCT Compressed Video

Figure 1(a) shows the block diagram for the traditional decoder for the MC-DCT compressed video. The operations can be described as

$$P_{rec}(t, x, y) = DCT^{-1}(DCT(e(t, x, y))) + P_{rec}(t-1, x-d_x, y-d_y) \qquad \text{(EQ 1)}$$

where P_{rec} is reconstructed video signal, e is the prediction error, and d is the motion vector. With simple reordering, we can change it to

$$DCT(P_{rec}(t, x, y)) = DCT(e(t, x, y)) + DCT(P_{rec}(t-1, x-d_x, y-d_y)). \qquad \text{(EQ 2)}$$

Namely, we switch the order of the inverse DCT and the inverse MC, and perform the inverse MC algorithm in the DCT domain. The new decoder block diagram is shown in figure 1(b). However, one technical issue is that the MC algorithm can have a motion vector of arbitrary number of pixels, while in the DCT algorithm,