Manipulation and Compositing of MC-DCT Compressed Video[1]

Shih-Fu Chang *   *David G. Messerschmitt* **

*Department of Electrical Engineering
& Center for Telecommunications Research
Columbia University
New York, NY 10027

**Department of EECS
University of California, Berkeley
Berkeley, CA 94720

(For publication in IEEE JSAC Special Issue on Intelligent Signal Processing, 1994)

Abstract

Many advanced video applications require manipulations of compressed video signals. Popular video manipulation functions include overlap (opaque or semi-transparent), translation, scaling, linear filtering, rotation, and pixel multiplication. In this paper, we propose algorithms to manipulate compressed video in the compressed domain. Specifically, we focus on compression algorithms using the Discrete Cosine Transform (DCT) with or without Motion Compensation (MC). Compression systems of such kind include JPEG, Motion JPEG, MPEG, and H.261. We derive a complete set of algorithms for all aforementioned manipulation functions in the transform domain, in which video signals are represented by quantized transform coefficients. Due to a much lower data rate and the elimination of decompression/compression conversion, the transform-domain approach has great potential in reducing the computational complexity. The actual computational speedup depends on the specific manipulation functions and the compression characteristics of the input video, such as the compression rate and the non-zero motion vector percentage. The proposed techniques can be applied to general orthogonal transforms, such as Discrete Trigonometric Transform. For compression systems incorporating MC (such as MPEG), we propose a new decoding algorithm to reconstruct the video in the transform domain and then perform the desired manipulations in the transform domain. The same technique can be applied to efficient video transcoding (e.g., from MPEG to JPEG) with minimal decoding.

1 Introduction

Advanced video services have emerged as a focus of interest as the technologies of digital signal processing, VLSI, and broadband networks advance. Examples of such services include multi-point video conferencing, interactive networked video, video editing/publishing, and advanced multimedia workstations.

1. Part of this work has been presented at IEEE ICASSP, Minneapolis, Minnesota, 1993.

Usually, video signals are compressed when transmitted over networks or stored in databases. After video signals are compressed, there are still many situations where further manipulations of such compressed video are needed. For example, in a multi-point video conferencing, multiple compressed video sources may need to be manipulated and composited within the network at the so-called *video bridge* [26]. Figure 1 shows a typical video conferencing scene, in which input video signals are arbitrarily manipulated (e.g., scaled to arbitrary size, translated to arbitrary locations) and composited into a single output video sequence. Typical video manipulation functions include translation (block-wise or pixel-wise), scaling, linear filtering, rotation, overlapping (opaque or semi-transparent[1]), and pixel multiplication. Many applications require only a subset of these features.

There are two possible ways to manipulate compressed video. The first approach fully decodes each compressed input video and then manipulates them in the spatial domain [26]. The output video needs to be re-encoded again if the compressed format is required. Alternatively, we can derive equivalent manipulation algorithms in the compressed domain and manipulate compressed video directly in the compressed domain [4,5,6,32]. Due to a much lower data rate and the removal of the unnecessary decoding/coding pair, the compressed-domain approach has great potential in reducing the computational complexity. In addition, manipulation in the compressed domain provides the flexibility to accommodate dynamic resources and heterogeneous Quality of Service (QoS) requirements. Users with low-end computing/communication resources can process the signal components with the highest significance only (e.g. lower-order DCT coefficients, or lower bands in subband coding) to maintain as high video quality as possible within the resource limit. This prioritized significance of signal components is not available in the uncompressed domain.

This paper shows that many video manipulation functions can be performed in the compressed domain more efficiently (i.e., with less computations) than in the uncompressed domain. This statement is particularly true when both input and output video signals need to be compressed, as in the case of network video compositing. We focus on the Motion Compensated Discrete Cosine Transform (MC-DCT) format, which is widely used in many image/video compression standards (e.g. H.261, MPEG, HDTV [2,13,16,35]). In specific, we derive one set of algorithms to perform all above mentioned manipulations in the transform domain, in which video signals are represented by transform coefficients (such as quantized DCT coefficients) [5]. Our derivations are based on the linearity of the manipulation functions and the orthogonality of the transform algorithms. The proposed techniques can be applied to general orthogonal transform coding algorithms, such as Discrete Sine Transform (DST) and DFT.

1. By semi-transparent overlapping, we mean that the background image pixels can be partly seen through the foreground image pixels (e.g., fade-in and fade-out special effects).

For the case where MC is incorporated (such as MPEG and H.261), we will discuss the obstacles preventing the MC-domain manipulation [4]. To avoid these difficulties, we propose a new decoding technique to convert the MC-DCT video to the DCT domain and then perform desired manipulation functions in the DCT domain. The same approach can be applied to efficient video transcoding (e.g., from MPEG to JPEG) with minimal decoding.

Video manipulation in the compressed domain is new. There is some independent related work. Smith and Rowe studied a subset of manipulation functions in the DCT domain [32], such as linear combination and pixel multiplication. However, they used a coefficient mapping approach without deriving underlying mathematical formulae. Based on a little different derivations, Lee and Lee derived the transform-domain filtering algorithms and proposed a pipelined hardware architecture [21]. Martucci derived the symmetrical convolution routines for the Discrete Trigonometric Transform [36]. The concept of compressed-domain manipulation can also be applied to other compression algorithms, such as subband coding. Lee and Woods proposed some subband-domain algorithms for simple operations such as picture in picture and text overlay [22]. However, border boxes are used along the overlap boundary to cover some artifacts.

This paper is organized as follows. Section 2 includes review of MC-DCT compression systems and some terminology definitions. We derive the proposed DCT-domain manipulation algorithms in Section 3. We explain the difficulties for video manipulation in the MC domain and propose some techniques to reduce computations of MC recalculation in Section 4. Manipulation of MC-DCT compressed video in the transform domain is proposed in Section 5. Performance analysis is presented in Section 6.

2 Background and Domain Definitions

Figure 2 shows the block diagram for hybrid compression systems based on the MC-DCT algorithm. Input images are segmented into small blocks, each of which is motion compensated and transformed into DCT coefficients. The *Motion Estimation* (ME) procedure finds the *optimal reference block* from the previous frame and also outputs the motion vector. The DCT coefficients are quantized and then run-length coded (RLC) to redundance in long sequences of zeroes. In addition, the statistically-based variable-length code (VLC), such as the Huffman code or arithmetic code [24], is applied to exploit any remaining data redundancy. For compression systems using intra-frame coding only, the ME block is not needed.

In this paper, we use the ***spatial*** domain to refer to the raw pixel data format before encoding or after decoding. Sometimes for the purpose of contrast, we also refer to it as the ***uncompressed*** domain. The ***DCT*** or ***DCT-compressed*** domain refers to the quantized DCT coefficients[1], which can be obtained after the inverse quantizer in the decoder of Figure 2. Note that a video signal can be transformed into the DCT domain either by applying the DCT on raw image data or by partially decoding the interframe MC-DCT encoded video, as will be described later. Through the context, we will also use the ***"transform*** domain" to refer to the frequency domain of general orthogonal transforms, such as DST, DFT, and DCT.

The ***MC*** or ***MC-compressed*** domain refers to the encoded data format after the MC algorithm. Basic components in the MC domain include the motion vectors ($\vec{d}$) and the prediction errors ($\hat{e}$), as shown in Figure 2. The ***MC-DCT-compressed*** or ***MC-DCT*** domain refers to the hybrid interframe encoded format shown in Figure 2. It contains the motion vectors and the DCT coefficients of the prediction errors.

3 Video Manipulation in the DCT Domain

We describe one set of video manipulation primitives — overlap, scaling, translation, linear filtering, rotation, and pixel-multiplication, and their DCT-domain equivalents in this section. Most video services require only a subset of these primitives.

3.1 Overlap

Opaque overlapping of two video objects requires substituting pixels of the foreground video object for those of the background object. ***Semi-transparent overlapping*** requires a linear combination of the foreground and background pixels, i.e.,

$$P_{new}(i,j) = \alpha \cdot P_a(i,j) + (1-\alpha) \cdot P_b(i,j)\ , \tag{1}$$

where P_{new}, P_a, P_b, and α are new pixels, foreground pixels, background pixels, and the transparency factor [30]. Since it's a linear operation, we can apply the same technique in the DCT domain, i.e.,

$$DCT\left(\bar{P}_{new}\right) = \alpha \cdot DCT\left(\bar{P}_a\right) + (1-\alpha) \cdot DCT\left(\bar{P}_b\right)\ , \tag{2}$$

where $\bar{P}$ represents the block-wise data format in DCT.

3.2 Pixel Multiplication

If the α coefficients vary from pixel to pixel, semi-transparent overlapping becomes a pixel-wise operation, i.e.,

$$P_{new}(i,j) = \alpha(i,j) \cdot P_a(i,j) + (1-\alpha(i,j)) \cdot P_b(i,j)\ . \tag{3}$$

This operation involves a basic operation called **pixel-multiplication**, i.e.,

1. Without otherwise specified, we assume the transform coefficients are by default quantized, so that we can take advantage of the fact that many coefficients are truncated to zero after quantization.

$$P_{new}(i,j) = P_a(i,j) \cdot P_b(i,j) \,. \tag{4}$$

Pixel-multiplication is also required in situations like *subtitl*ing (adding text on top of an image), *anti-aliasing* (removing the jagged artifacts along the boundaries of irregularly-shaped video objects), and special-effect *masking* (for special graphic patterns). To compute the pixel-wise multiplication in the DCT domain, we derive a multiplication-convolution relationship for the DCT similar to that for the DFT, except that the order for the convolution is increased to 2·N points. We leave the proof in [7] and present the final result here. Similar symmetric convolution routines were studied in [36] independently.

Suppose X_c is the DCT of image block X. First, we form an extended symmetrical version of the DCT coefficients as the following

$$\hat{X}_c(k_1, k_2) = \begin{cases} X_c(k_1, k_2) / C(k_1, k_2) & k_1, k_2 = 0, \ldots, N-1 \\ X_c(-k_1, k_2) / C(-k_1, k_2) & k_1 = -N+1, \ldots, -1,\ k_2 = 0, \ldots, N-1 \\ X_c(k_1, -k_2) / C(k_1, -k_2) & k_1 = 0, \ldots, N-1,\ k_2 = -N+1, \ldots, -1 \\ X_c(-k_1, -k_2) / C(-k_1, -k_2) & k_1, k_2 = -N+1, \ldots, -1 \\ 0 & k_1 = N, \text{ or } \quad k_2 = N \end{cases}$$

$$\text{where} \qquad C(k_1, k_2) = \begin{cases} \frac{1}{2} & k_1 = k_2 = 0 \\ 1 & \text{otherwise} \end{cases}$$

Then, the 2D *multiplication-convolution theorem* can be described as follows.

If

$$y(i,j) = x(i,j) \cdot h(i,j), \qquad i, j = 0, \ldots, N-1$$

then

$$\hat{Y}_c(k_1, k_2) = \frac{1}{2N} \cdot \sum_{l_1, l_2 \in [-N, N-1]} \sum \left[\hat{X}_c(l_1, l_2) \cdot \hat{H}_c\Big(((k_1 - l_1))_{2N}, ((k_2 - l_2))_{2N} \Big) \right] \times \alpha(k_1 - l_1)\, \alpha(k_2 - l_2)$$

$$k_1, k_2 = -N, \ldots, N-1$$

$$\alpha(k) = \begin{cases} 1 & k \in [-N, N-1] \\ -1 & \text{otherwise} \end{cases}$$

(5)

For convenience we use the notation "$((n))_{2N}$" to denote (n modulo $2N$). If we ignore the α term, the above equation is equivalent to a 2D 2N-point circular convolution. In other words, we expand the N×N DCT coefficients of an image block to an extended symmetric 2N×2N block. The pixel-wise multiplication of two image blocks in the spatial domain corresponds to the two-dimensional 2N-point circular convolution-like operation of the extended blocks in the DCT domain.

3.3 Translation (block-wise and pixel-wise)

In a video scene containing multiple video objects, as in the case of multi-source video conferencing, users may want to move each video object around flexibly. We refer to this general position control operation as **translation**. Two types of translation should be considered separately— ***block-wise*** (i.e., fixed block boundary positions) and ***pixel-wise*** (i.e. arbitrary positions). If we restrict both the horizontal and vertical translation distance to be an integral multiple of the block width, then the DCT coefficients are always aligned with the same block structure. (By *block structure*, we refer to the grid lines used to segment the images into small blocks.) Moving a video object around requires updating the origin point of the video object only. Aforementioned manipulation functions such as overlapping and pixel multiplication can be performed in the DCT domain as described if the block structures of input video objects are aligned.

However, if we allow translation by an arbitrary number of pixels, the block structures of the input video objects could be mismatched. Figure 3 illustrates mismatched block structures of objects A and B. In the spatial domain, this mismatch problem is easy to solve. In the transform domain, due to the rigid block structure, it's no longer a trivial issue. Suppose we want to composite object A and object B as shown in Figure 3 and we choose the block structure of object A as the final reference block structure. Therefore, we need to re-segment object B with respect to the block structure of object A. A new image block of object B, say B′, contains contributions from four original neighboring blocks (B_1-B_4), namely the lower-left corner (B_{13}) of block B_1, the lower-right corner (B_{24}) of block B_2, the upper-right corner (B_{31}) of block B_3, and the upper-left corner (B_{42}) of block B_4. We supplement these four contributions with zeroes as illustrated in Figure 3 to form N pixels by N pixels image blocks. Then, the new block can be calculated as

$$B' = B_{13} + B_{24} + B_{31} + B_{42} \, . \tag{6}$$

The same method cannot be applied in the DCT domain. We cannot simply assembly four subblocks from the original DCT blocks to form the DCT of the new block. Instead, it should be calculated as

$$\mathrm{DCT}\,(B') \;=\; \mathrm{DCT}\left(B_{13}\right) + \mathrm{DCT}\left(B_{24}\right) + \mathrm{DCT}\left(B_{31}\right) + \mathrm{DCT}\left(B_{42}\right) \, . \tag{7}$$

Therefore, if we can find the relationship between the DCT of subblocks (B_{13} - B_{42}) and the DCT of the original blocks (B_1 - B_4), then we can calculate the DCT of the new block directly from the DCT of the original blocks.

In other words, the conversion processes back and forth between the DCT domain and the spatial domain can be eliminated. Kou and Fjallbrant [18] proposed an algorithm to compute the DCT of a signal block from two original adjacent blocks when the overlap length is one half of the block size. However, the complexity becomes quite high for an arbitrary overlap length. Here, we propose a direct computation method applicable to any arbitrary overlapping length.

Figure 4 shows a mathematical model for obtaining a subblock from an original block (e.g., B_{42} from B_4) in the spatial domain. Note that the upper-left corner of B_4 is extracted, supplemented by zeroes, and moved to the lower-right corner, as required in Figure 3. A matrix-form equation for this subblock extraction procedure is

$$B_{42} = H_1 B_4 H_2 \quad \text{, where} \quad H_1 = \begin{bmatrix} 0 & 0 \\ I_h & 0 \end{bmatrix} \qquad H_2 = \begin{bmatrix} 0 & I_w \\ 0 & 0 \end{bmatrix} \text{,} \tag{8}$$

where I_h and I_w are identity matrices with size h×h and w×w respectively; h and w are the number of rows and columns extracted. As shown in Figure 4, multiplying B_4 with a pre-matrix H_1 extracts the first h rows and translates them to the bottom; multiplying B_4 with a post-matrix H_2 extracts the first w columns and translates them to the right.

It can be shown that all unitary orthogonal transforms such as the DCT are distributive to matrix multiplication [15], i.e.,

$$DCT(AB) = DCT(A)DCT(B) . \tag{9}$$

Using this property, we can compute the DCT of B_{42} directly from the DCT of B_4, i.e.,

$$DCT(B_{42}) = DCT(H_1)DCT(B_4)DCT(H_2) . \tag{10}$$

Summing all contributions from four corners, we can obtain the DCT coefficients of the new block B′ directly from the DCT of old blocks B_1 - B_4. In other words,

$$DCT(B') = \sum_{i=1}^{4} DCT\left(H_{i1}\right)DCT\left(B_i\right)DCT\left(H_{i2}\right) . \tag{11}$$

The DCT of H_{i1} and H_{i2} can be pre-computed and stored in memory. With a block width equal to N, only 2·N-2 H matrices need to be stored. Given DCT of B_i and H_{ij}, Equation 11 can be implemented by matrix multiplications. The required computation can be reduced by using sparse matrix multiplication techniques since many DCT coefficients are zeroes. Detailed complexity analysis will be presented in Section 6.1.

3.4 Linear Filtering

Two-dimensional separable **linear filtering** can also be done in the DCT domain [5, 9, 21, 29]. Linear filtering of images in the horizontal direction can be achieved by multiplications with post-matrices:

$$Y = \sum_i X_i H_i \quad , \tag{12}$$

where X_i is the input image block, H_i is the filter coefficients represented in the block form, and Y is the output image block. Each output image block has contributions from several input blocks. The number of contributing input blocks depends on the length of the filter kernel. Since the DCT algorithm is distributive to matrix multiplication, we can calculate DCT(Y) in the following way:

$$DCT(Y) = \sum_i DCT\left(X_i\right) DCT\left(H_i\right) \tag{13}$$

Similarly, image filtering in the vertical direction can be achieved by multiplication with pre-matrices. Detailed derivation of the transform-domain filtering algorithm can be found in [7,21].

3.5 Scaling

Another important image manipulation technique is **scaling**. Each pixel in the final scaled image is a linear combination of several neighboring pixels in the original image [12]. Thus, it can be treated in a way similar to linear filtering. For example, if we use the simple box area averaging method to implement the 1/2×1/2 down scaling operation [34], a new block can be computed as $H_1B_{11}W_1 + H_2B_{21}W_1 + H_1B_{12}W_2 + H_2B_{22}W_2$, where B_i are the original neighboring blocks, H_i are the vertical scaling matrices, and W_i are the horizontal scaling matrices. For example,

$$H_1 = \left(W_1\right)' = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad H_2 = \left(W_2\right)' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix} \tag{14}$$

for a block width of 4 pixels. The same linear operations can be performed in the DCT domain as described above.

3.6 Other manipulation functions

In addition to the above operations, *rotation* and *shearing* are also useful in creating visual special effects. Strictly speaking, both these two operations are linear geometrical transformations and should have equivalent counterparts in the transform domain. For example, the horizontal shearing can be implemented as

$$Y = \sum_i W_i X S_i \quad , \tag{15}$$

where X is the input image block, W_i extracts the i_{th} row and supplements the remaining rows with zeroes, and S_i performs the horizontal 1D shifting and interpolation [12]. Again, applying the distributive property of DCT, we can derive the shearing algorithm in the DCT domain. For rotation, a typical implementation is based on a column-reserving shearing and a row-reserving shearing, plus some appropriate scaling [12]. Thus, based on the derived DCT-domain scaling and DCT-domain shearing, the DCT-domain rotation algorithm can be derived as well. However, for such complicated operations as rotation, the matrix structure in the DCT domain could become too cumbersome and thus their DCT-domain algorithms may not be attractive, compared to the conventional spatial-domain approach.

4 Manipulation of MC-Compressed Video

Video signals in the MC-compressed domain consist of two components: motion vector (d), and prediction error (e), denoted as the *MC data*. Given the MC data of input video streams, it would be desirable to compute the MC data of the output video of the manipulation functions directly in the MC domain without any decoding. Unfortunately, the MC algorithm does not have the same linear and orthogonal property as that for the transform algorithms discussed above, thus making the MC-domain video manipulation impossible. In this section, we illustrate this obstacle by using two examples: overlapping and scaling. Given the MC data of the input video streams, we need to convert them back to the uncompressed domain before any manipulations. Recalculation of the MC data is required if the output video needs to be encoded in the MC format again. In order to avoid the intensive computations involved in the *MC data recalculation* process, we propose an *inference principle* to calculate the new motion vector with minimal computations.

4.1 Obstacles for Video Manipulation in the MC domain

Figure 5 shows **overlapping** of two video objects. Assume both the foreground and background objects are MC-encoded and their block structures are aligned. The MC data of the foreground object can be kept in the composited video since it is not affected by the background object. However, part of the background object is obscured by the foreground object. We need to consider two different areas of the background object separately— the *directly affected area* (DAA) and the *indirectly affected area* (IAA). The difference has to do with the *search area*[1] in the MC algorithm. In the DAA, part of its search area is replaced by the foreground object. If the motion vector is from the obscured area, the original prediction block is destroyed. The MC data thus becomes invalid and needs to be recalculated. In the IAA, though the search area is not overlapped by the foreground object, the MC data may still need to be recalculated since its prediction block could be modified through *error propagation*. In other words, the MC data of the composited video cannot be directly obtained

1. The search area is the area in the previous frame where the optimal prediction block is searched.

from the MC data of the input video streams. Both the foreground and background video need to be fully decoded so that new MC data can be calculated.

Another example is *down scaling*. Suppose every four original image blocks are scaled down to one new image block. The motion vectors of these four original blocks in general are different. Although the new motion vector can be inferred from the original four motion vectors, decoded images still need to be reconstructed so that the correct prediction errors can be recalculated.

4.2 Simplification of MC Data Recalculation

As discussed above, all MC-encoded video needs to be decoded first so that the new MC data of the composited video can be calculated. But the MC process is very computation-intensive. In order to keep low-cost manipulations of MC-coded video feasible, we propose some techniques here to simplify the indispensable MC recalculation. The first approach is to reduce the frequency of MC recalculation. The second approach is to minimize the computations associated with the ME process, by inferring the new motion vectors from the original input motion vectors. We will use the overlap example in Figure 5 to verify the effectiveness of these two approaches. The underlying concept can be applied to other situations as well. For example, new motion vectors can be interpolated from old motion vectors in the down scaling scenario mentioned above.

Approach I: Reducing the Frequency of Recalculation

The first principle for simplifying the MC recalculation process is to reduce the number of image blocks requiring MC recalculation. For example, in the opaque overlapping situation in Figure 5, we can assume that only image blocks in the DAA may need recalculation of the motion vector. If the motion vector comes from the foreground area, then the original prediction block is obscured and a new motion vector is needed. Our experiments on some test sequences show that only 5-15% of the directly affected blocks require MC recalculation [4]. Compared to blind recalculation of every block in the background object, the frequency of MC recalculation is greatly reduced. Of course, this number varies with video sequences and depends on the specific manipulation scenario as well.

Approach II: Obtaining New Motion Vectors by Inference

To further reduce the computation cost associated with the MC recalculation, we apply the second principle — *inferring new motion vectors from the old ones*. Our objective is to simplify the ME procedure, which is the most computation-demanding process in the MC algorithm.

To reduce the computational complexity of ME, Jain and Jain [14] have proposed a two-dimensional binary search approach by assuming that the block distortion function is monotonically increasing along the horizontal or vertical direction when the search position moves away from the optimal prediction point. For the overlap scenario considered. if we adopt the same assumption, we can reduce the number of search positions to two only! Figure 6 shows the spatial relations of these search positions, where B represents the current block, D the optimal prediction point, and D_1, D_2 the crossing points when the search position moves away from D horizontally and vertically respectively. Jain and Jain's assumption assures that any search position to the right of D_1 has a larger distortion than D_1, and any search position lower than D_2 has a larger distortion than D_2. So, if we assume that the new optimal prediction location is still from the background object, then either D_1 or D_2 should have the minimal distortion and be the new optimal prediction. Compared to the full-search ME (which needs $(2\cdot d\_max+1)^2$ search positions), the computational complexity is much lower.

4.2.1 Impact on Video Quality

Whenever MC recalculation (including quantization) is performed, video quality will be further degraded even the full-search ME is used. This issue is very similar to the error accumulation issue in repetitive coding of video signals [37]. In the overlap scenario of Figure 5, the average SNR among the directly affected blocks which require MC recalculation is decreased by 2.88 ~ 5.45 dB if the full-search ME is used. This quality loss depends on the image content and the distribution of the distortion function. The additional SNR loss due to our proposed two-point ME procedure is usually small (within 1 dB). Significant loss occurs in very few frames, where the assumption about the monotonic distribution of the distortion function does not seem to hold.

It is worth mentioning that there is also quality degradation for background pixels outside the directly affected area. For these pixels, we can reasonably assume their motion vectors are unchanged. But their prediction errors may need to be updated because their prediction blocks in the previous frame may be located inside the DAA and need to be recalculated. The change of prediction values will propagate to more outside area as the video sequence proceeds. Our simulations show the average SNR loss among pixels affected by the error propagation effect ranges from 2.38 to 6.19 dB. The percentage of background pixels affected by this error propagation effect remains low in our test, though in reality it could vary with different video streams. Some sparse spots with significant SNR loss are visually noticeable. We have proposed techniques to partially rectify the error propagation problem by updating the prediction errors only, without recalculating the motion vectors [4]. These techniques are proved to be effective in removing abrupt severe quality loss in many frames.

5 Manipulation of MC-DCT Compressed Video

The obstacles preventing the MC-domain video manipulation also exist for the MC-DCT encoded video. In order to keep the benefits of the compressed-domain video manipulation, we propose to keep the video signals as much in the compressed form as possible. In specific, we propose to partially decode the MC-DCT video to the DCT domain first, and then perform desired manipulations in the DCT domain. One potential problem with this approach is that conventional MC-DCT decoding first decodes the DCT part and then the MC part. The non-linear MC is at the bottom of the compression stack. Our solution is to swap the order of the decoding stack, namely inverse MC first, then inverse DCT. Note that the encoding stack doesn't have to change, meaning that regular MC-DCT encoders can be used. Figure 7 illustrates the proposed transform-domain manipulation system.

As shown in Figure 2, the decoding process for MC-DCT video can be described as

$$P_{rec}(t, x, y) = DCT^{-1}(DCT(e(t, x, y))) + P_{rec}(t\text{-}1, x\text{-}d_x, y\text{-}d_y)\ , \quad (16)$$

where P_{rec} is the reconstructed image, e is the prediction error, and d is the motion vector. We skip quantization of DCT(e) for simplicity here. With a simple reordering, we can rewrite it as

$$DCT(P_{rec}(t, x, y)) = DCT(e(t, x, y)) + DCT(P_{rec}(t\text{-}1, x\text{-}d_x, y\text{-}d_y))\ . \quad (17)$$

Now, the inverse MC is performed before the inverse DCT. In other words, the inverse MC is performed in the DCT domain, denoted as "***(MCD)***$^{\boldsymbol{-1}}$" in Figure 7. The $(MCD)^{-1}$ procedure first locates the optimal reference block in the previous frame (by using the received motion vector) and then adds the DCT of the reconstructed optimal reference block to the DCT of the prediction errors. By doing so, the received video is reconstructed in the DCT domain.

However, the DCT of the optimal reference block may not be immediately available. In general, the motion vector of each block is an arbitrary number of pixels. Therefore, the optimal reference block generally overlaps with four blocks, whose DCT coefficients are already available. The situation is the same as that for the pixel-wise translation discussed in Section 3.3. In both places, we need to calculate the DCT of a new arbitrary-position block from the DCT of four original neighboring blocks. Therefore, the same formula in Equation 11 can be used here to obtain the DCT of the optimal prediction block, i.e., $DCT(P_{rec}(t\text{-}1, x\text{-}d_x, y\text{-}d_y))$.

The same approach can be used in the forward MC as well when the manipulation output needs to be MC encoded again. The DCT of the prediction error can be calculated as

$$DCT(e(t, x, y)) = DCT(P_{rec}(t, x, y)) - DCT(P_{rec}(t\text{-}1, x\text{-}d'_x, y\text{-}d'_y))\ , \quad (18)$$

where d' is the new motion vector for the output video sequence.

If the motion vectors are zero (as in DPCM interframe coding) or integral multiples of the block width, the block structure alignment procedure is not needed, and motion compensation in the DCT domain requires simple additions only, as in the spatial domain. If only one motion vector component (d_x or d_y) is zero or integral multiples of the block width, the DCT coefficients of the new block can be computed from the DCT coefficients of two original overlapping blocks, instead of four.

6 Performance Analyses

We analyze performance of the proposed transform-domain video manipulation techniques both analytically and numerically in this section. Two major performance factors considered are *computational complexity,* and *video quality.* Other considerations such as hardware implementation are also briefly discussed. Numerical simulations are done by non-real-time software prototyping. Three different test scenarios are considered. Scene 1 is the scenario with multiple small inputs and a large output display, as shown in Figure 1. Input videos are down scaled, translated, and then composited together. Scene 2 has multiple input videos and an output display of the same size as the input. Every input video object needs to be scaled down in this case. Scene 3 represents the picture-in-picture scenario. One input video is scaled down and overlapped on top of another video stream, which does not require any scale change. Video sequences used are head-and-shoulder images. We consider the case where both input and output are MC-DCT encoded. Different quantization tables are tested, including those listed in the JPEG and MPEG standards.

6.1 Computational Complexity

The computational complexity of the transform-domain video manipulation techniques strongly depends on the number of zero DCT coefficients, which in turn depends on the compression rate of the input video. When the MC algorithm is included, the complexity is also significantly depends on the motion vector distribution. As far as manipulation functions are concerned, in general, block-wise operations benefit more from the transform-domain approach, compared to the pixel-wise operations. But since the decoding/coding conversion is avoided, the DCT-domain approach may still provide a net efficiency gain for many cases involving pixel-wise operations.

In Table 1, we list the number of multiplications and additions required in each major operation, such as the DCT, DCT^{-1}, MCD, MCD^{-1}, pixel-wise translation, scaling, quantization, and inverse quantization. Interested readers are referred to [7] for detailed derivations. An important property we assume in deriving the complexity is that the run-length-code (RLC) of the quantized DCT coefficients can indicate the locations of the non-zero DCT coefficients so that the redundant operations associated with zero coefficients can be skipped. In addition, we exploit optimization in spare matrix multiplication, which is the basic computation module in various transform-domain manipulation functions, such as linear filtering and translation. A straightforward implementation of matrix multiplication, $A \cdot P \cdot B$ (where A and B are $N \times N$ matrices, and P represents an N pixels by N pixels image block), requires $2 \cdot N^3$ multiplications and $2 \cdot (N-1) \cdot N^2$ additions. Based on some assumption of the non-zero DCT coefficient distribution and optimization techniques for sparse matrix multiplication, the same operation needs $(1/\beta + 1/\sqrt{\beta}) \cdot N^3$ multiplications and $(1/\beta + 1/\sqrt{\beta}) \cdot N^3$ additions only, where β is the *compression rate*[1] of the input image. Since matrix multiplication is the basic building block in many DCT-domain operations, many complexity figures in Table 1 are closely related to the above formula.

Table 1 shows that the MCD (MCD^{-1}) operation has the highest computational complexity. This is due to the need of block structure alignment in both directions. If one of the motion vector components is an integral multiple of the block width, then the associated computation can be reduced, as discussed in Section 5. The overall computations required for MCD depends on the non-zero motion vector percentage, α_1 and α_2 (defined in Table 1). Typical values of (α_2, α_1) are shown in Figure 8. The composited scenes usually have lower (α_2, α_1) values than input video signals due to the involved down-scaling operations. The compression ratio (β) is about 7-8 for the salesman sequence, 20 for the Miss USA sequence (high compression ratio due to its flat background), and 7-9 for the composited sequence.

Pixel-wise translation in the DCT domain also requires block structure adjustment in both directions. But since some matrix multiplications can be shared, its complexity is lower than that of MCD. The scaling operation, like linear filtering, can be implemented by multiplications with a pre-matrix and a post-matrix in the DCT domain. In general, the DCT-domain scaling operation has a lower computational complexity than MCD and pixel-wise translation. Also, the $1/3 \times 1/3$ down scaling in the DCT domain requires a little bit less computation than $1/2 \times 1/2$ down scaling because more image blocks share the same matrix multiplication.

For the spatial-domain operations, the major computations are from the coding/decoding conversion, i.e., the DCT and its inverse. Chen's fast algorithm [8] has a complexity of $2 \cdot \log_2 N - 3 + 8/N$ multiplications per pixel, which is about one half of that using the 2N-point FFT [10]. There have been many new fast DCT computation methods reported in the literature [3,27, 20], but all have a similar complexity order. The spatial-domain $1/2 \times 1/2$ down-scaling operation needs 1/4 multiplication per pixel. (i.e., $P_{new} = (P_{11}+P_{12}+P_{21}+P_{22})/4$

1. Strictly speaking, β defined here is the ratio between the total number of the DCT coefficients and the number of non-zero DCT coefficients. It's more or less proportional to the commonly used *compression ratio*. We use them interchangeably in the context, except in the detailed calculation of computational complexity.

by using the simple box area averaging algorithm described in [34]). It becomes 1/9 multiplication per pixel for $1/3 \times 1/3$ down scaling.

One interesting note is that the major component for the transform-domain operations increases linearly with the block width, N, while it increases with the order of $\log_2 N$ for the spatial-domain approach (although when the block size increases, the image compression ratios may change as well.) Therefore, the transform-domain approach is more suitable for cases using a small block size, which is usually true in image compression.

Simulation Results:

Comparisons of computational complexity for three test scenarios are shown in Table 2. Manipulation features demonstrated here include overlapping, scaling with various factors, and block-wise translation. Compression characteristics such as β, α_2, α_1 are collected through simulations and substituted into the analytical formulae in Table 1. Results show that for these scenarios (with both input and output MC-DCT compressed), the DCT-domain approach is faster than the spatial-domain approach by about 10% to 30%. We also compare the speed of the software implementations. The CPU time reported in Table 2 approximately confirms the above speedup with the DCT-domain approach. However, note that we assume translations in these scenarios are all block-wise. If pixel-wise translation is allowed, then the DCT-domain approach will suffer from the overhead of block structure alignment. One trick to minimize the overhead of pixel-wise translations is to perform down-scaling before translation, if size reduction is needed. Thus, the relative complexity contribution from pixel-wise translation can be reduced.

If the original input videos are compressed without the MC algorithm, namely DCT-encoded only as in the JPEG or Motion JPEG standards, the computational speedup by using the DCT-domain approach will increase significantly since the complicated MCD process is not required. Table 3 shows computational complexity for the case where only DCT coding (without MC) is used. The net computational speedup factor ranges from 3 to 6.

6.2 Image Quality

One would think that since the DCT-domain approach does not need a second coding pass after manipulation, the video quality should be better than that of the spatial-domain approach. However, this is not true in general. The DCT-domain manipulation algorithms are mathematically equivalent to their counterparts in the spatial domain. Even we keep all operations in the DCT domain, the final DCT coefficients still need to be quantized. Therefore, the quality degradation suffered in the second quantization still applies to the DCT-domain approach.

The second quantization will introduce quality degradation whenever the image content is modified (e.g., scaled or filtered) in the intermediate operations between two coding passes. This problem is described as the *error accumulation* problem in multi-pass image coding [37]. Table 4 shows the SNR values of reconstructed videos before and after the second coding process. The extent of quality degradation depends on the manipulation functions between two coding passes. For example, modification of image content in Scene 2 is more dramatic than that in Scene 3. Therefore, the SNR loss caused by re-coding is much larger in Scene 2 (7.7 dB) than in Scene 3 (1.2 dB).

In our implementations, the DCT-domain approach suffers an additional minor quality degradation due to the need of thresholding DCT coefficients after each intermediate manipulation. The thresholding process is used to reconstruct the RLC structure of the DCT coefficients, which would be destroyed in each intermediate manipulation. The RLC structure is important because it can indicate the positions of the non-zero DCT coefficients so that redundant computations can be skipped. From Table 4, we can see that this additional quality degradation is very minor, about 0 ~ 0.7 dB. Our simulations show that images obtained both by the DCT-domain approach and by the spatial-domain approach suffer some subjective quality degradation due to lossy quantization twice. But there is no noticeable quality difference between images reconstructed from these two approaches.

6.3 Other Considerations

6.3.1 Worst-Case Throughput

The DCT-domain operations produce variable throughput, as opposed to the constant throughput of the spatial-domain approach. The higher the compression ratios and the lower the non-zero motion vector percentages the input videos have, the faster the manipulation functions can be executed in the DCT domain. For real-time implementations which need to consider the worst-case situation, this variable throughput may be a shortcoming. But in the DCT domain, we have the flexibility to skip high-order DCT coefficients whenever the maximal processing delay bound is exceeded. The image quality degradation can thus be minimized since the high-order coefficients are usually less subjectively important. In essence, the amount of non-trivial DCT coefficients requiring processing is determined based on the hardware processing capability, as opposed to the rate-based criterion used in the constant-rate video encoders. The relationship between the compression rates and computational complexity shown in Table 1 can assist in allocation of required computing capacity when we design real-time video manipulations.

6.3.2 Techniques for Computation Reduction

As we can see in Table 1, the most complex operation in the transform domain is the MCD (MCD^{-1}) algorithm. This is mainly due to its need of block structure realignment.We also notice that the computational complexity of the MCD algorithm strongly depends on the non-zero motion vector percentages, i.e. α_2 and α_1. If α_2 and α_1 are both equal to zero, the MCD algorithm can be easily implemented by simple additions. However, with the regular MC algorithm, these percentages vary with different video sequences, as shown in Figure 8. One way to reduce the non-zero motion vector percentages is to modify the MC encoder to give some preference to zero motion vectors. For example, we can add a fractional weighting factor to the distortion associated with the zero motion vector. We refer to this algorithm as the *weighted MC algorithm*. Our simulations show that if we set the weighting factor associated the zero motion vector to 80%, the non-zero motion vector percentages (α_2, α_1) can be greatly reduced (from (52%, 38%) to (30%, 19%) for “Miss USA” video, and from (8%, 19%) to (6%, 7%) for the “salesman” video). The SNR values of reconstructed images using the weighted MC algorithm are very close to those using the regular MC algorithm. In terms of the computational efficiency, the weighted MC can raise the computational speedup by 10% to 20%, compared to the regular MC.

Another possible technique to reduce the computational complexity in the DCT domain is to combine a sequence of operations into a single operation. For example, for MC-DCT compressed videos, we may need to perform $F(DCT(e) + G(DCT(P_{ref})))$, where G represents the MCD^{-1} operation and F represents the scaling operation. Since scaling is a linear operation, we can apply the distributive law to change the above formula to $F(DCT(e)) + H(DCT(P_{ref}))$, where H represent the composite function $F \cdot G$. The computations associated with function F can thus be reduced since it can operate on a much more sparse matrix now (DCT(e) vs. $(DCT(e) + G(DCT(P_{ref}))$)).

7 Conclusions

Many advanced video applications require manipulations of compressed video. We have explored the freedom of performing video manipulation in the compressed domain. In specific, we have derived efficient algorithms in the transform domain for many useful video manipulation functions such as overlapping, translation, scaling, pixel multiplication, rotation, and linear filtering. These algorithms can be applied to general orthogonal transforms, like DCT, DFT, and DST. Compared to the spatial-domain approach which converts compressed video back to the spatial domain and manipulates video data in the spatial domain, the proposed transform-domain approach can increase the computation efficiency, with a factor depending on the compression characteristics of the input videos. For hybrid MC-DCT encoded video, we have proposed a new decoding algorithm to convert the encoded video to the DCT domain and perform manipulation functions in the DCT domain. This new decoding algorithm can also be applied to efficient transcoding from MPEG to JPEG.

We have derived the formulae for estimating the computational complexity of major DCT-domain manipulation algorithms. We have also evaluated the efficiency of the DCT-domain techniques by software simulations. The DCT-domain compositing approach can reduce the required computations by 60% ~ 75% for DCT-compressed images (without MC), 10% ~ 23% for MC-DCT-compressed images in various simulated scenarios without complicated pixel-wise operations (such as pixel-wise translation). If pixel-wise manipulations are incorporated, the efficiency of DCT-domain operations will drop to some extent, depending on the specific manipulation scenarios. Considerations of the recovered video quality and some real-time implementation issues have been discussed as well.

As shown in this study, compression algorithms have significant impact on the efficiency of the video manipulation techniques. Orthogonal transforms provide great flexibility for pursuing manipulation techniques in the transform domain, but non-linear coding algorithms such as MC do not. The close interplay between the compression algorithms and the manipulation techniques strongly motivates a joint approach to optimal algorithm designs for video compression and manipulation. For example, we may want to compromise some compression performance in order to provide greater flexibility in video compositing and manipulation in the compressed domain. Currently, we are actively working in this area.

8 References

1. N. Ahmed, T. Natarajan, and K.R. Rao, “Discrete Cosine Transform,” IEEE Transactions on Computers, Vol. C-23, pp.90-93, Jan. 1974.
2. J. A. Bellisio and K.-H. Tzou, “HDTV and the Emerging Broadband ISDN Network,” SPIE Vol. 1001, Visual Communications and Image Processing ‘88, pp. 772-86.
3. S.C. Chan and K.L. Ho, “A New Two-Dimensional Fast Cosine Transform Algorithms,” IEEE Transactions on Signal Processing, Vol. 39, No.2, Feb. 1991, pp.481-5.
4. S.-F. Chang and D.G. Messerschmitt, “Compositing Motion-compensated video within the Network,” IEEE 4th Workshop on Multimedia Communications, Monterey, CA, April, 1992.
5. S.-F. Chang, W.-L. Chen and D. G. Messerschmitt, “Video Compositing in the DCT domain,” *IEEE Workshop on Visual Signal Processing and Communications*, Raleigh, NC, pp. 138-143, Sep. 1992.
6. S.-F. Chang and D. G. Messerschmitt, “A New Approach to Decoding and Compositing Motion Compensated DCT-Based Images,” IEEE ICASSP, Minneapolis, Minnesota, April, 1993.

7. Shih-Fu Chang, *Compositing and Manipulation of Video Signals for Multimedia Network Video Services*, Ph.D. Dissertation, University of California at Berkeley, August, 1993.
8. W.-H. Chen, C.H. Smith, and S.C. Fralick, "A Fast Algorithm for the Discrete Cosine Transform," IEEE Transactions on Communications, Vol. COM-25, No. 9, Sep. 1977, pp. 1004-9.
9. B. Chiptrasert and K.R. Rao, "Discrete Cosine Transform Filtering," *Signal Processing*, Vol. 19, No. 3, pp. 233-45, Mar. 1990.
10. R.J. Clarke, "Transform Coding of Images," Academic Press, 1985.
11. T. Duff, "Compositing 3-D Rendered Images," *Siggraph*, vol. 19, November 1985, pp. 41-44.
12. J.D. Foley, A. V. Dam, S. K. Feiner and J.F. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed., Addison-Wesley, 1990.
13. CCITT Recommendation H.261, "Video Codec for Audiovisual Services at px64 kbits/s", 1990.
14. J.R. Jain and A.K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," IEEE Transactions on Communications, Vol. COM-29 (12), pp.1799-1808, Dec. 1981.
15. A. Jain, "Fundamentals of Digital Image Processing," Prentice-Hall Inc., 1989.
16. Standard Draft, JPEG-9-R7, Feb. 1991
17. Ronald K. Jurgen, "The Challenges of Digital HDTV," IEEE Spectrum, April, 1991, pp.28-30.
18. W. Kou and T. Fjallbrant, "A Direct Computation of DCT Coefficients for a Signal Block from Two Adjacent Blocks," IEEE Transaction on Signal Processing, Vol. 39, No. 7, pp. 1692-5, July 1991.
19. M. Kunt, A. Ikonmopoulos, and M. Kocher, "Second-Generation Image Coding Techniques," Proceedings of IEEE, Vol. 73, pp. 549-574, Apr. 1985.
20. B.G. Lee, "FCT - A Fast Cosine Transform," IEEE ICASSP, San Diego, March, 1984, pp.28A.3.1-4.
21. J.B. Lee and B.G. Lee, "Transform Domain Filtering Based on Pipelining Structure," IEEE Transactions on Signal Processing, pp. 2061-4, Vol. 40, No. 8, Aug. 1992.
22. Y.Y. Lee and J.W. Woods, "Video Production with Compressed Images," submitted to the SMPTE journal.
23. D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," Communications of the ACM, Vol. 34, No.4, April, 1991.
24. G.G. Langdon and J. Rissanen," Compression of Black-White Images with Arithmetic Coding," IEEE Transactions on Communications, June 1981, pp.858-67.
25. M. Liou, "Overview of the p$\times$64 kbits/s Video Coding Standard," Communications of the ACM, Vol. 34, No. 4, April 1991.

26. M. Lukacs, "An Advanced Digital Network Video Bridge for Multipoint with Individual Customer Control," Private Communication, Bell Communications Research, NJ, May 1992.

27. M. Narasimha and A. Peterson, "On the Computation of the Discrete Cosine Transform," IEEE Transactions on Communications, Vol. COM-26, No. 6, June 1978, pp.934-6.

28. A.N. Netravali and J.D. Robbins, "Motion-Compensated Television Coding: Part I," The Bell System Technical Journal, Vol. 58(3), pp.631-670, Mar. 1979.

29. K.N. Ngan and R. J. Clarks, "Lowpass Filtering in the Cosine Transform Domain," Intern. Conference on Communications, pp.31.7.1-31.7.5, Seattle, WA, June 1980.

30. T. Porter and T. Duff, "Compositing Digital Images," Computer Graphics, Vol. 18, pp. 253-259, 1984.

31. P.V. Rangan, H.M. Vin, and S. Ramanathan, "Communication Architectures and Algorithms for Media Mixing in Multimedia Conferences," IEEE/ACM Transactions on Networking, Vol.1, No.1, Feb., 1993.

32. B.C. Smith and L. Rowe, "Algorithms for Manipulating Compressed Images," IEEE Computer Graphics and Applications, pp. 34-42, Sept. 1993.

33. G.K. Wallace, "The JPEG Still Picture Compression Standard," Communications of the ACM, Vol. 34, No. 4, April 1991.

34. C.F.R. Weiman, "Continuous Anti-Allseed Rotation and Zoom of Raster Images," SIGGRAPH 80, 286-293.

35. Standard Draft, MPEG Video Committee Draft, MPEG 90/ 176 Rev. 2, Dec. 1990.

36. S.A. Martucci, "Symmetric Convolution and the Discrete Sine and Cosine Transforms," IEEE Transactions on Signal Processing, Vol. 42, No. 5, pp. 1038-51, May, 1994.

37. S.-F. Chang and A. Eleftheriadis, "Error Accumulation in Repetitive Image Coding," IEEE International Conference on Circuits and Systems, London, May, 1994.