A NEW APPROACH FOR REDUCING BLOCKINESS IN DCT IMAGE CODERS

Stephen A. Martucci

Scitex Digital Video
431 Crown Point Circle, Suite 150
Grass Valley, CA 95945

ABSTRACT

This paper presents a new approach for reducing the blockiness that occurs when using DCT image coders at high compression ratios. The method is simply the replacement of the inverse DCT-2 in the decoder by a larger inverse DCT-1 followed by overlapping and averaging of the enlarged blocks to reconstruct the image. The modified decoder can decode any bitstream generated by a standard encoder. Blockiness is reduced but there is no noticeable distortion or loss of sharpness in the image. There is also no significant increase in complexity when using this method.

1. INTRODUCTION

Many image and video compression standards, such as JPEG and MPEG, are based on discrete cosine transform (DCT) processing. In the JPEG standard, for example, the input image is first partitioned into nonoverlapping blocks of size 8×8. Each block is transformed by the DCT, then the coefficients are quantized and entropy coded. A decoder performs the inverse operations in reverse order (entropy decoding, dequantization, then inverse DCT), to produce a reconstructed approximation to the original image. Using the DCT, it is possible to achieve a fair level of compression without any noticeable artifacts in the reconstructed images. At higher levels of compression, however, artifacts become increasingly visible. A particularly disturbing artifact characteristic of DCT compression is blockiness.

Blockiness occurs because each block is independently transformed and quantized. There is no sharing of image data between the nonoverlapping blocks. At low to moderate levels of compression this is not a problem, but as the degree of quantization, and therefore level of compression, increases, blockiness becomes increasingly visible and detracting.

Solutions to the problem of blockiness have been reported in the literature [1, 2, 3]. A straightforward way to reduce blockiness would be to overlap the image data before the transform. Unfortunately, however, this would increase the amount of data that must be compressed and therefore would reduce the level of compression achievable. Another method is to filter the image after decompression to remove the high-frequency block edges. Simple filtering of the entire image, however, has the disadvantage of unwanted softening of the image. Adaptive filtering restricted to block boundaries adds significant complexity and can still lead to some distortions in the image. Methods that adaptively modify the input image during encoding and then undo those modifications during decoding work well but add significant complexity to both encoder and decoder and require that a matching nonstandard encoder and decoder be used. Iterative restoration and other post-processing techniques can add substantial complexity and delay.

In this paper we present a novel method of modifying a block-DCT-based image decoder that is simple to implement, adds little additional computational complexity, significantly reduces blockiness in reconstructed images, and preserves the ability of the decoder to decode any standard bitstream. No modifications need to be done to the encoder or the bitstream. The method is simply the replacement of the inverse DCT-2 in the decoder by a larger-than-conventional-size inverse DCT-1 and the overlapping of these larger-than-conventional-size blocks to reconstruct the decoded image. As we will show, using the DCT-1 in place of the DCT-2 has the effect of shifting the pixels by 1/2-sample in each dimension and enlarging each block by one row and one column without distorting the data. Because these blocks can now be overlapped, with averaging done in the regions of overlap, blockiness can be significantly reduced. The reduction in blockiness is especially noticeable when the image is enlarged.

2. THE METHOD

Assume we have received a bitstream, such as that produced according to the JPEG standard, that represents a DCT-compressed image of size $H \times V$. The steps to decode the bitstream by our new method are simply the following. It should be noted that the order of Steps 4 and 5 can be reversed and that both steps could be incorporated into Step 3 by a simple modification of the DCT-1 calculation.

1. Perform entropy decoding and dequantization to recover the 2D DCT-2 coefficients of each block of size 8×8.
2. Append a row and column of zeros to each block to increase the size to 9×9.
3. Compute the inverse 2D DCT-1 of each 9×9 block to recover a 9×9 block of pixels.
4. Scale the elements of the first and last rows of pixels by $\sqrt{2}$.
5. Scale the elements of the first and last columns of pixels by $\sqrt{2}$.
6. Reconstruct the image by combining the blocks so that each block overlaps each neighbor by one row or one column. Each pixel in a row or column of overlap is replaced by the average of the overlapping values. At the four corners of each block, there are four values that are averaged. For all other overlapping pixels, there are two values that are averaged.
7. Extract the desired $H \times V$ subimage from the size $(H+1) \times (V+1)$ image that resulted from the overlapping process.

3. THEORETICAL DISCUSSION

The theory behind our method is best presented in terms of the recently discovered properties of the DCT. The DCT used for image compression is just one member of a family of 16 discrete sine and cosine transforms, collectively called discrete trigonometric transforms (DTTs). These DTTs can be used to perform convolution, and therefore digital filtering, but the convolution is a special type called *symmetric convolution*. Details of symmetric convolution and its implementation using DTTs can be found in [4, 5, 6].

In order to simplify the notation and discussion, we discuss the DCTs in one dimension only. Because the transforms are separable, all results extend to two dimensions by simply applying the properties in each of the two dimensions consecutively.

The symmetric convolution of two finite sequences is equivalent to symmetrically extending the first sequence at both ends, symmetrically extending the second sequence on the left only, linearly convolving the two, then windowing the result. The second sequence is a special type, called a *filter-right-half*, that the symmetric convolution operation extends to create a symmetric filter. That symmetric filter is applied to the first sequence after it has been symmetrically extended at both ends to supply boundary values for the filtering.

Symmetric convolution can also be implemented by taking the appropriate inverse DTT of the element-by-element product of the forward DTTs of the inputs. The DTTs can therefore be used to implement digital filters where the type of filtering is that of symmetric convolution. There are 40 distinct types of symmetric convolution and 16 distinct DTTs. For this work, we use only one type of symmetric convolution and only two DTTs, the even DCT-1 and the even DCT-2.

Our method works by using the DCTs to perform a filtering operation on each block of pixels as it is being inverse transformed. The filter shifts the pixels of the block by 1/2-sample in each dimension, but causes practically no change to the frequency content of the data. In addition, each block increases in size by one row and one column. We are effectively filtering the reconstructed pixels $x(n)$, $n = 0, 1, \ldots, N-1$, where $N = 8$, with an even-length symmetric filter $h(n)$, $n = -L/2, \ldots, L/2-1$, where $L = 16$, having the frequency response shown in Fig. 1. The convolution performed is:

$$w(n) = \sum_{r=-L/2}^{L/2-1} h(r)\, \widetilde{x}(n-r) \qquad n = -1, 0, \ldots, N-1 \qquad (1)$$

where $\widetilde{x}(n)$ is $x(n)$ after half-sample symmetric extension (i.e., mirror-image reflection with the endpoint repeated) at both ends to the extent needed for computing the summation. But this filtering is implicitly performed as a symmetric convolution using DCTs, as explained next.

For what follows we define the weighting function:

$$k(p) = \begin{cases} 1/\sqrt{2} & p = 0 \text{ or } N \\ 1 & p = 1, 2, \ldots, N-1 \end{cases} \qquad (2)$$

The *orthogonal* form of the even DCT-2 ($\mathrm{C_{II}}E$) of the sequence $x(n)$, $n = 0, 1, \ldots, N-1$, is what is used for image compression and is computed according to:

$$X_{\mathrm{II}}(m) = \mathrm{C_{II}}E\{x(n)\} = \sqrt{\frac{2}{N}}\, k(m) \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi m(n+\frac{1}{2})}{N}\right) \qquad m = 0, 1, \ldots, N-1 \qquad (3)$$

The *orthogonal* form of the even DCT-1 ($\mathrm{C_{I}}E$) of the sequence $y(n)$, $n = 0, 1, \ldots, N$, is computed according to:

$$Y_{\mathrm{I}}(m) = \mathrm{C_{I}}E\{y(n)\} = \sqrt{\frac{2}{N}}\, k(m) \sum_{n=0}^{N} k(n)\, y(n) \cos\left(\frac{\pi mn}{N}\right) \qquad m = 0, 1, \ldots, N \qquad (4)$$

Suppose we are given an even-length symmetric filter $h(n)$, $n = -L/2, \ldots, L/2-1$. We define the filter-right-half by:

$$h^r(n) = \begin{cases} h(n) & n = 0, 1, \ldots, L/2-1 \\ 0 & n = L/2, \ldots, N-1 \end{cases} \qquad (5)$$

where $L/2 \leq N$. We compute the filter transform coefficients using the *convolution* form of the DCT-2 ($\mathcal{C}_{2e}$) according to:

$$H^r(m) = \mathcal{C}_{2e}\{h^r(n)\} = 2\sum_{n=0}^{N-1} h^r(n)\cos\left(\frac{\pi m(n+\frac{1}{2})}{N}\right) \quad m = 0, 1, \ldots, N-1 \quad (6)$$

Then, we can perform the convolution of (1) using DCTs according to the following two equations:

$$Y_{\rm I}(m) = \begin{cases} \mathcal{C}_{2e}\{h^r(n)\} \times {\rm C}_{{\rm II}E}\{x(n)\} & \\ \qquad\qquad\qquad m, n = 0, 1, \ldots, N-1 \\ 0 \qquad\qquad\quad m = N \end{cases} \quad (7)$$

$$w(n-1) = \frac{1}{k(n)}\,{\rm C}_{{\rm I}E}^{-1}\{Y_{\rm I}(m)\} \quad m, n = 0, 1, \ldots, N \quad (8)$$

where '$\times$' denotes element-by-element multiplication and ${\rm C}_{{\rm I}E}^{-1}$ is the inverse orthogonal DCT-1. Notice that $x(n)$ and the ${\rm C}_{{\rm II}E}$ are of length N and $w(n)$ and the ${\rm C}_{{\rm I}E}^{-1}$ are of length $N+1$.

We can now analyze, in one dimension, the steps of our method presented in Section 2. Step 1 recovers the $X_{\rm II}(m)$ needed by (7); the encoder had performed the DCT-2 and those (quantized) coefficients are available after Step 1. If we let $H^r(m)$ be a sequence of N ones, then Step 2 implements (7) and Steps 3, 4, and 5 implement (8) to compute the convolution of (1). But because $H^r(m)$ is all ones, the symmetric convolution is performed implicitly by merely computing the one-sample-longer inverse DCT-1 of DCT-2 coefficients, followed by scaling of the first and last pixels.

It is important to note that the DCT-2 is an N-point transform whereas the DCT-1 is an $(N+1)$-point transform. The 9-point inverse DCT-1 is actually no more complex than the 8-point inverse DCT-2. Both transforms can be computed with $\frac{1}{2}N\log_2 N$ or fewer real multiplications and less than $\frac{3}{2}N\log_2 N$ real additions [6], where we are using $N = 8$ for both transforms in the steps of Section 2.

Knowing that $H^r(m)$ is equal to a sequence of N ones, we can compute $h^r(n) = \mathcal{C}_{2e}^{-1}\{H^r(m)\}$ and then symmetrically extend on the left to find the effective even-length symmetric $h(n)$ of length $2N$ being implemented in (1). We have done this and have plotted the frequency response of $h(n)$ in Fig. 1. As can be seen, the filter passes practically all frequencies with no loss. The only effect of the filter is to shift the pixels by 1/2-sample and increase the sequence length by one.

Our method's effect on the pixel locations can be seen in Fig. 2. If we assume that we receive a block of 4×4 DCT-2 coefficients, the inverse DCT-2 would recover pixels located at positions of the image marked by a '2'. Now, if we append a row and column of zeros to the block and compute a 5×5 inverse DCT-1, we would generate interpolated pixels representing the image at the half-sample positions marked by a '1'. Because we are generating pixels at half-sample positions, the outermost rows and columns of each block represent the same positions as those of neighboring blocks. Therefore, we can overlap the decoded blocks, average the pixels in the areas of overlap, and thereby greatly reduce the blockiness artifacts, all without creating any noticeable new distortions in the image.

4. EXPERIMENTAL RESULTS

To test our new method, we compressed the 256×256 test image Lena about 17.5:1 using a standard JPEG encoder. Then we decoded the bitstream using the standard decoding algorithm and our new decoding algorithm. Because our new method creates a 1/2-sample shift in the reconstructed image, we could not compare it directly to the original Lena. Instead, we created a shifted version by computing a 256×256 DCT-2 of the original Lena followed by a 257×257 inverse DCT-1 (with scaling) and extraction of the first 256×256 pixels. This shifts the entire image the same way each block is shifted by our decoder method.

The PSNR of the standard decoder output compared to the original is 28.70 dB. The PSNR of our new decoder output compared to the shifted original is 29.20 dB. The visual improvement is much greater than the numerical improvement, especially when the image is enlarged. In Fig. 3, we show the enlarged standard JPEG result on the top and the enlarged new method result on the bottom. The visible blockiness of the standard result has been virtually eliminated by our new method.

5. CONCLUSION

We have presented a new approach to reducing blockiness in DCT image coders, whereby we simply replace the inverse DCT-2 of the decoder with a larger-than-conventional-size inverse DCT-1, then overlap those decoded blocks to reconstruct the image. The only effect of changing the DCT type is a 1/2-sample shift of the pixels. The shift and the overlap lead to a significant reduction in blockiness. Only the decoder is modified; any standard bitstream can be decoded. Our method is simple to implement and adds little computational complexity because it is an integral part of the decoder and not a separate post-processing step and because the DCT-1 has a fast algorithm comparable to the DCT-2.

6. REFERENCES

[1] H. C. Reeves and J. S. Lim, "Reduction of blocking effects in image coding," *Optical Engineering*, vol. 23, pp. 34–37, Jan./Feb. 1984.

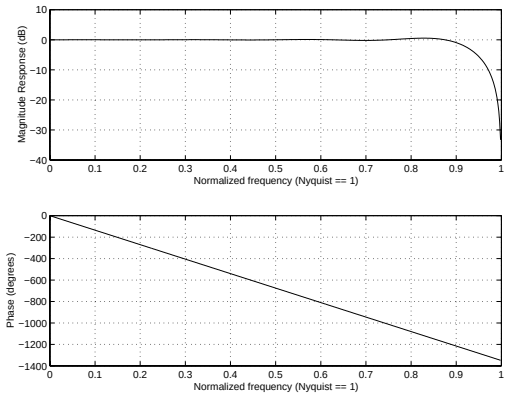

Figure 1. Frequency response of filter effectively being implemented by our method.

1	1	1	1	1
2	2	2	2	
1	1	1	1	1
2	2	2	2	
1	1	1	1	1
2	2	2	2	
1	1	1	1	1
2	2	2	2	
1	1	1	1	1

Figure 2. Sampling pattern for example 4×4 block of source pixels. Each '2' marks the location of a pixel represented by the DCT-2, each '1' marks the location of a pixel represented by the DCT-1.

[2] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, pp. 91–95, Mar. 1992.

[3] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Regularized reconstruction to reduce blocking artifacts of block discrete cosine transform compressed images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, pp. 421–432, Dec. 1993.

[4] S. A. Martucci, "Digital filtering of images using the discrete sine or cosine transform," *Optical Engineering*, vol. 35, pp. 119–127, Jan. 1996.

[5] S. A. Martucci, "Symmetric convolution and the discrete sine and cosine transforms," *IEEE Transactions on Signal Processing*, vol. 42, pp. 1038–1051, May 1994.

[6] S. A. Martucci, *Symmetric Convolution and the Discrete Sine and Cosine Transforms: Principles and Applications*. PhD thesis, Georgia Institute of Technology, 1993.

Figure 3. Enlargements of decoded test image Lena. Top is standard JPEG decoding (PSNR = 28.70 dB). Bottom is our new method of JPEG decoding (PSNR = 29.20 dB).