REDUCED DCT APPROXIMATIONS FOR LOW BIT RATE CODING

Ricardo L. de Queiroz
Xerox Corporation, 800 Phillips Rd., 128-27E, Webster, NY, 14580
queiroz@ieee.org

ABSTRACT

Transform approximations are explored for speeding up the software compression of images and video. Faster approximations are used to replace the regular DCT whenever only few DCT coefficients are actually encoded. In one instantiation we employ a data analyzer to drive a bank of coders by switching to the fastest coder depending on the image contents or on the output demand. The compression speed is adapted to the image contents in order to homogenize the data production rate, i.e. smoother areas would produce less bits than detailed ones but faster. The approximations in this paper are applicable to high compression (or constant output rate) environments and where lower complexity is required.

1. INTRODUCTION

In several image transmission systems, the image data is input or generated *on-the-fly* and transmitted immediately. This is the case in live transmission of video and stills. In color fax, image parameters are only determined after handshaking. Hence, as is the case of live production of stills and video, compression has also to be performed in real time. Compression standards such as JPEG [1] or MPEG [2] rely on sequential transform, quantization, and entropy coding. Typically, all image data is transformed and quantization is applied to every transformed coefficient. These two steps typically drain a significant fraction of the compression computation and are independent of the image data or of the compression target. Hence, while the contents change along the image, the number of bits produced per image block also changes, but there is little change in compression time. As a result, the rate in bits per second that the compressor produces would change significantly from smooth regions to active ones. The problem is that in several transmission systems such as modems for color fax, the transmission rate is fixed. A buffer can control and homogenize the speed for the modem, but, if the compressor is not fast enough, the buffer might underflow and the connection might time out.

In summary, we are concerned with systems in which software-based compression is performed *on-the-fly* and where it is desirable to reduce computational complexity either to homogenize the bit rate production or to reduce costs. Our goal is to speed up the compression by replacing the real transformation by a low cost approximation.

2. APPROXIMATING REDUCED TRANSFORMS

In this paper, we examine the case of the discrete cosine transform (DCT) and concentrate exclusively in the JPEG image coding standard, although results would also apply to MPEG. Let blocks have $M \times M$ pixels and assuming a separable 2D DCT, we can analyze a 1D M-point DCT. Imagine that some blocks have no high frequency content or that quantization would remove the high frequency coefficients in such a way that fewer than M coefficients are actually needed. In this paper's context, a *reduced transform* is the one wherein only N out of M coefficients are actually generated (see for example [3],[4]). This can be accomplished by pruning an M-point transform, but if one needs to retain only N lower frequency coefficients where $M = KN$, there is a "short-cut" for approximating the generation of these coefficients as shown in Fig. 1. In this figure the reduced $M : N$ DCT can be approximated by an N-point DCT, by first reducing the input vector by means of a K-to-1 reduction (e.g. pixel averaging). The expansion too, from the N low frequency coefficients to the M samples, can be approximated as in Fig. 1. The method is based on interpolation in the DCT domain and is further studied in [5]–[8]. Like in the forward case, the 1-to-K expansion can be the "nearest neighbor" or any other simple interpolation method. In the JPEG case, typical values are $N = 1, 2, 4$, i.e. reductions are $K = 8, 4, 2$, respectively. For example, in the case $N = 1$, only the DC is computed. This is done so by averaging all the 64 samples in the input block. Only one sample is quantized and there is no need for entropy coding the AC terms. For decompression, the DC term is directly dequantized and replicated for all the 64 block pixels. When $N = 2$, there is an average of 4×4 sub-blocks and a very simple DCT (DCT-2 is the Haar transform [5]).

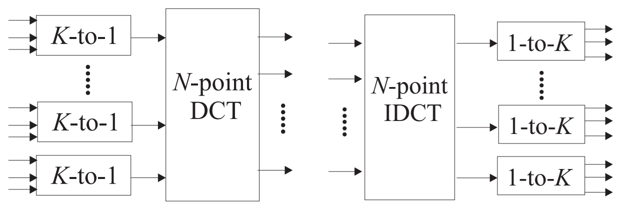


Figure 1. Approximated transform to approach a reduced M-point DCT ($M = NK$).

The use of the reduced transforms may imply in signal losses which can be further emphasized through their approximations. It is advisable to only apply the reduced transforms whenever there is no need for the high frequency information.

3. TRANSFORM ADAPTATION

Straight application of the reduced transforms can cause image degradation. In the best scenario, the image block is analyzed and one coder is selected based on image contents, so that it would save computation without compromising the image quality. The analyzer complexity overhead has to be

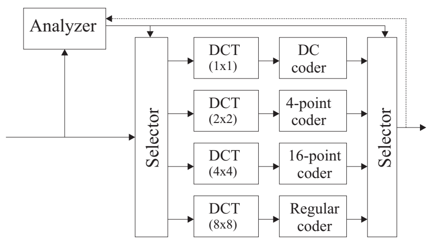
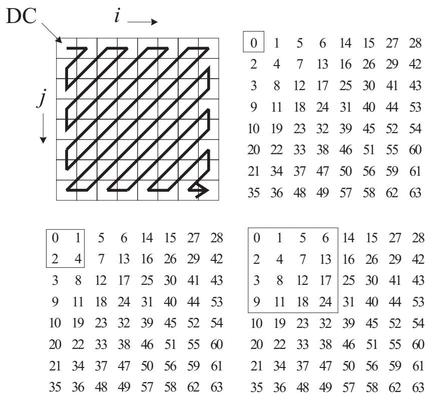


Figure 2. Fast DCT coder based on multiple reduced DCT and multiple coders.



0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Figure 3. JPEG's zigzag block scanning path and low frequency subblocks for $N = 1, 2, 4$.

such as to not offset the gains obtained by switching transforms. The overall diagram is depicted in Fig. 2 for the JPEG case, where the encoding system is shown for $N = 1, 2, 4$, along with the normal encoder ($N = M = 8$). Note that reduced transforms may also imply reduced compressors.

The key to select the coder is to determine if there are high frequency contents in a block or not. Actually, one may want to use the smallest transform that would encompass all the relevant high frequency data in a block.

3.1. Decoder adaptation

Decoder adaptation is trivial in the sense that one can observe the compressed data and compute the highest frequency non-zero DCT coefficient in a block. In JPEG the DCT coefficients are quantized and the block is scanned into a vector following a "zig-zag" pattern [1]. Let $\{v(n)\}$ ($0 \leq n < 64$) be the vector containing the 64 quantized samples of a given block. Referring to Fig. 3, one can easily identify which are the vector entries corresponding to the lowest frequency 1×1, 2×2 and 4×4 coefficients, e.g. $\{v(0), v(1), v(2), v(4)\}$ compose the lowest frequency 2×2 subblock.

The general process is to verify whether there are no non-zero coefficients outside the boxes in Fig. 3. However, testing too many coefficients can be costly and can partially offset the gains provided by reduced transforms. Let N_m be maximum index of a non-zero coefficient in $\{v(n)\}$ in a given block. We select $N = 1$ if $N_m = 0$; $N = 2$ if $N_m < 5$ and $v(3) = 0$; and $N = 4$ if $N_m < 14$ and $v(10) = 0$. If none of the conditions occur we use a regular transform $N = 8$. The above conditions are trivial, except for the case $N = 4$ where we did not test all the data within a 4×4 sub-block because it would require testing whether $N_m < 25$ plus checking if 9 individual coefficients are non-zero. In this method, only 3 out of 16 coefficients are kept "outside" the 4×4 box by testing the end of the vector at the 14th instead of the 25th element. Tests show that the proposed simplification does not impact selection significantly, despite reducing computation.

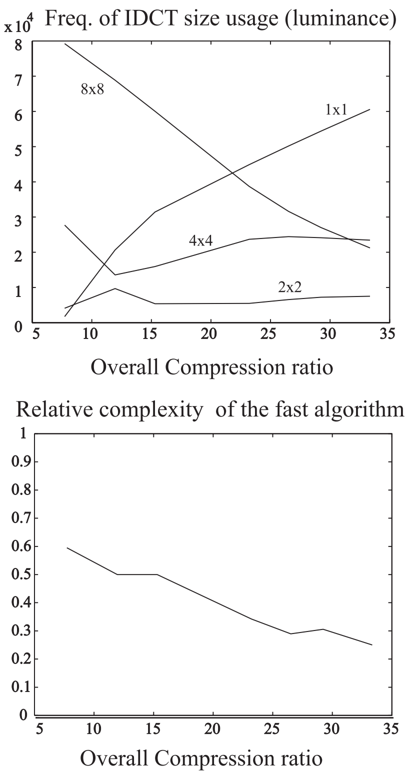


Figure 4. Top: typical frequency of reduced inverse DCT size usage in JPEG decompression. Bottom: relative complexity of the proposed algorithm compared to normal JPEG decompression.

Experiments were carried to compute how much time a real JPEG decoder would save by switching reduced transform approximations as we propose. Figure 4 shows the frequency of occurrence of blocks in each class as a function of the compression ratio for several images. The relative complexity, as measured in decompression speed in a Sun Ultra Sparc architecture machine, is also shown in Fig. 4. The overall adaptive decoder is typically more than twice as fast as the normal decoder. In our experiments the adaptive decoder did not produce any significant alteration in the image as compared to the normal JPEG decoder.

3.2. Adaptive encoding

The decoder adaptation is more trivial since the data is readily available and needs to be decompressed anyway. When compressing the image, if all the coefficients are computed and available, there is no need to use any reduced transform or coder. We have to estimate if there will be non-zero coefficients outside the boxes shown in Fig. 3, but without adding significant complexity. Otherwise it would be wiser to spend the extra computation to perform the full DCT.

3.2.1. Backward adaptive

If the compression system feeds a constant rate channel which can time out if there is no data to be transmitted, it is typical to buffer the encoder output in order to homogenize the rate fed to the channel. If the buffer tends to underflow one must speed up the generation of compressed bits. In baseline JPEG one cannot change the image quality (quantization) along the image. If the buffer is critically empty, the only method to increase the production of compressed bits is to speed up the compression itself.

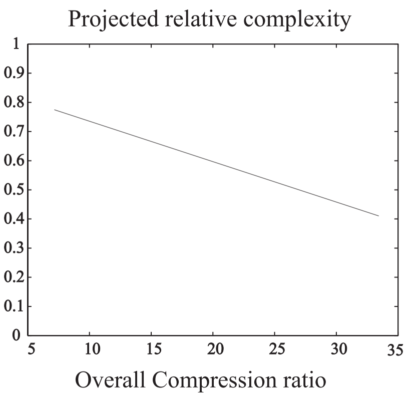


Figure 5. Projected relative complexity of the proposed system which is based an HVQ analyzer and multiple transforms in JPEG. The complexity is relative to standard JPEG coding of 8×8 blocks.

Let the buffer hold B bits and assume its status to be $b(n)$ where n is some time instant, e.g. a block count, and assume the system times out when the buffer gets empty, i.e. when $b(n) = 0$. Assume there are L compressors: C_1 through C_L. Let compressor C_k output data at a bit rate $R_k(n)$ bits/sec and let the channel draw data at a constant rate of R_0 bits/sec. Furthermore, without any loss of generality, we can assume the coders are labeled sequentially so that $R_k \geq R_{k+1}$. We assign coders according to the buffer status. The emptier the buffer, the faster the coder should be. Hence, we assign thresholds T_0 through T_L wherein $T_0 = 0$ and $T_L = B$ so that we assign coder C_k if $T_{k-1} \leq b(n) < T_k$. A detailed analysis of this setup allows us to determine the best T_k. First, note that the proposed system is only necessary when $R_1(n) > R_0 > R_L(n)$, because otherwise the buffer would be inevitably empty and there will be nothing we can do about it, or the buffer would be constantly full such that switching becomes unnecessary. Second, assuming a stationary process where $R_k(n)$ is constant and equal to R_k, threshold levels are unimportant. To see this note that if $R_k > R_0 > R_{k+1}$ the actual status $b(n)$ will oscillate around threshold T_k causing the selector to toggle between compressors C_k and C_{k+1}. Remember that with the stationary assumption, the system is up for a long time and each compressor outputs a constant rate. Hence, the system reached a balanced point: the buffer fills triggering compressor C_{k+1} which empties the buffer and, then, triggers compressor C_k.

In a transient state, where $R_k(n)$ changes shortly to settle down in another stationary point, a smooth region of the image will force $b(n)$ to decrease until a switch in compressors restores the balance or the image gets more active. Therefore, we might also like to have the thresholds as far apart as possible to allow the system to recover itself without switching for a faster compressor which might degrade the image (or, at least, to postpone the switch as much as possible). If we assume the bit-rates are piecewise constant, separated by transient states, the above considerations may hint that a good solution for the decision levels is a uniform buffer partition. In other words:

$$T_k = kB/L. \tag{1}$$

In reality the bit rates are not well behaved and vary depending on the image content in a very non-linear fashion. However, the approximation into stationary and transient states is a reasonable model for practical applications. In experiments the uniform buffer partition led to the best image quality results.

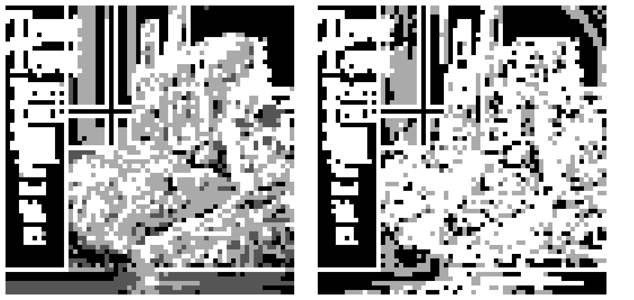


Figure 6. Comparison between actual and estimated maps of reduced transforms. From black to white: $N = 1, 2, 4, 8$. Left: actual; right: estimated via HVQ. ($Q = 3Q_d$ in both cases.)

3.2.2. Forward adaptive

Forward adaptation means that the analyzer should find the smallest N for a given block without losing high frequency information. We propose to use hierarchical vector quantization (HVQ) as the data analyzer [9],[10]. HVQ is based on look-up tables (LUT) and can easily map an 8×8 image block into a codeword with only 6 levels of LUT (63 LUT). As in any VQ system, this codeword maps to a representative block which is supposed to approximate the input data. Like in the HVQ-JPEG case [11], we can use HVQ and map the output codeword to something else than a reconstruction block. In our case the output codeword is mapped directly to a number between 1 and L indicating which coder should be used for the input block.

The LUT design is simple. The HVQ steps are designed using standard HVQ techniques. The mapping from codewords to encoder indices is done by correlating the HVQ output to a classifier's output for several images [12]. For each block and for a given quantizer table, the classifier design decides which is the smallest N such that the $N \times N$ lower frequency sub block contains all the non zero coefficients. The choice of N and the HVQ output for every block are correlated. At the end of training, each HVQ codeword is associated with the encoder selected by the classifier in the majority of the blocks in which such a codeword was produced.

The projected relative complexity of the proposed system is shown in Fig. 5. The proposed system is to be twice as fast as JPEG for high compression ratios. Image quality, however, is not granted. HVQ is far from being a perfect representation of the input system. For 8×8 blocks we used 6 stages of 10 bit codewords and yet the approximation is very rough. As a result the analyzer may misjudge the input data. To minimize error visibility it is a good idea to be conservative, i.e. bias towards the slower coder in the HVQ mapping design phase. A comparison between actual and estimated block classification maps is carried in Fig. 6. In that figure dark regions mark the blocks where only the DC is computed ($N = 1$). Lighter regions indicate blocks where $N = 2, 4$ or $N = 8$, for white pixels. It assumes a quantizer table $Q = 3Q_D$ (Q_d is JPEG's default quantizer table [1]).

The classification estimation errors are generally acceptable, mainly for high compression ratios. Figure 7 shows objective evaluations comparing the proposed system with regular JPEG. It shows the peak signal to noise ratio (PSNR) related to the compression ratio (or bit rate) for two examples. It also shows the average PSNR difference between methods for several images. The proposed fast method with an HVQ-based analyzer is typically comparable to the regu-

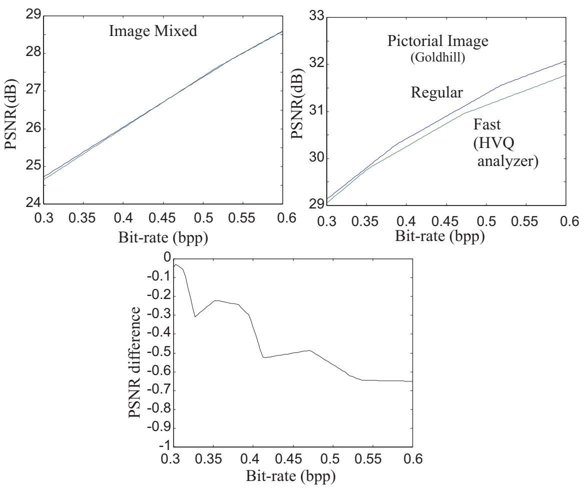


Figure 7. Compression results for both regular JPEG and the proposed system (using an HVQ analyzer). Average PSNR difference computed using 5 images.

lar compression method for high compression scenarios. As the compression ratio decreases, the proposed system yields images with lower relative quality. In any case, the decrease in objective performance is typically less than 0.7dB for the bit rates of interest. Figure 8 show decompressed versions of image "mixed" after compression to a bitrate of 0.52 bpp. Both images were decompressed at a PSNR around 27.6dB.

4. CONCLUSIONS

Approximations to the DCT and their enabling algorithms were proposed for use in image compression in order to save the overall compression computation. They are suitable to JPEG and MPEG in high compression ratio modes. The need for high compression is to mask the effects of the approximation which can typically half the overall computation for low bit rates and where JPEG/MPEG compression of the data blocks is already discarding too much visible information. In other words, savings come from not computing information that is not encoded. The method is applicable for environments where there is a need to produce a homogeneous bit rate or to environments where it is desirable to save computation while compressing the image or video to low bit rates. In one of the proposed compression methods, multiple reduced transforms and coders are employed and coders are selected according to an output buffer status in order to homogenize the compressed data bit production rate. In another method the coders are switched according to the results of a quick analysis of the input block. Such an analysis is performed very quickly via a LUT-based method derived from HVQ. All methods have shown to be efficient to substitute the DCT in JPEG/MPEG for low bit rates.

REFERENCES

[1] W. P. Pennebaker and J. L. Mitchell, *JPEG: Still Image Compression Standard,* Van Nostrand-Reinhold, 1993.

[2] A. M. Tekalp, *Digital Video Processing,* Upper Saddle River, NJ, Prentice-Hall, 1995.

[3] B. Girod and K. W. Stuhlmuller, "A content dependent fast DCT algorithm for low bit-rate video coding," *Proc. ICIP* Chicago, IL, Oct. 1998.

[4] I-M. Pao and M. T. Sun, "Modelling DCT coefficients for fast video encoding," *IEEE Trans. on Circuits and Systems for Video Tech.,* Vol. 9, pp. 608–616, June 1999.

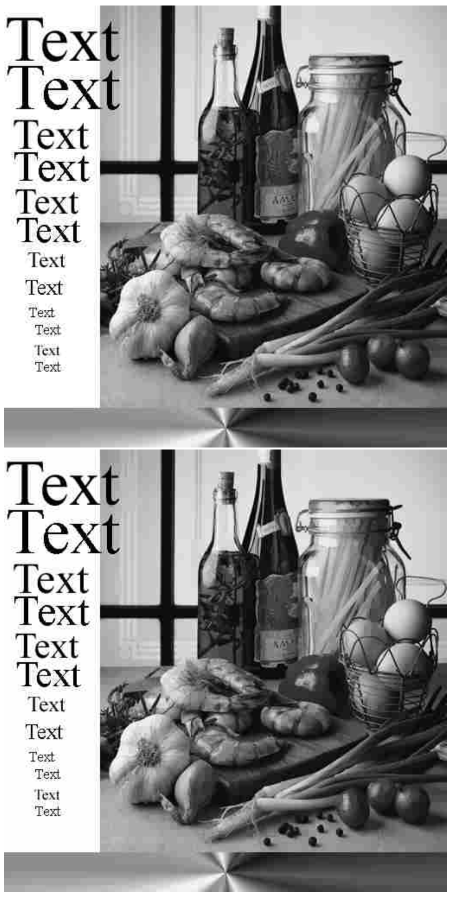


Figure 8. Image "mixed" after compression to 0.52 bpp. Top: regular; bottom: using proposed HVQ-analyser-based system.

[5] K. R. Rao and P. Yip *Discrete Cosine Transform : Algorithms, Advantages, Applications,* Academic Press, 1990.

[6] R. de Queiroz and R. Eschbach, "Fast downscaled inverses for images compressed with M-channel lapped transforms," *IEEE Trans. on Image Proc.,* Vol. 6, pp. 794–807, June 1997.

[7] K. N. Ngan, "Experiments on 2D decimation in time and orthogonal transform domains," *Signal Processing*, Vol. 11, pp. 249–263, Oct. 1986.

[8] S. Martucci, "Image resizing in the DCT domain," *Proc. ICIP'95,* Vol. II, pp. 244-247, 1995.

[9] P. C. Chang, J. May and R. Gray, "Hierarchical vector quantization with table look-up encoders," *Proc. Intl. Conf. Communications,* pp. 1452–1455, 1985.

[10] M. Vishwanath and P. Chou, "An efficient algorithm for hierarchical compression of video," *Proc. ICIP'94,* Vol. 3, pp. 275–279, 1994.

[11] R. de Queiroz and P. Fleckenstein, " Very fast JPEG compression using hierarchical vector quantization ," *IEEE Signal Proc. Letters,* Vol. 7, pp. 97-99, May 2000.

[12] A. Aiyer and R. M. Gray, "A fast table look-up algorithm for classifying document images," *Proc. ICIP'99,* 25PP4.10, 1999.