Tzou, K. H., [1987] "Progressive image transmission: a Review and Comparison," *Optical Engineering*, *vol. 26*, pp. 581-580, July 1987.

Vetterli, M. and H. Nussbaumer, [1984] "Simple FFT and DCT Algorithms with Reduced Number of Operations," *Signal Processing*, *vol. 6*, pp. 267-278, Aug 1984.

Wang, L. and M. Goldberg, [1988] "Progressive Image Transmission by Transform Coefficient Residual Error Quantization," *IEEE Trans. on Communication*, *vol. 36*, pp. 75-87, Jan 1988.

Wallace, G. K., [1991] "The JPEG Still Picture Compression Standard", *Com. of the ACM*, *vol. 34*, **no. 4**, pp. 30-44, April 1991.

Westover, L. A., [1990] "Footprint Evaluation for Volume Rendering," *Computer Graphics*, *vol. 24*, **no. 4**, pp. 367-376, Aug 1990.

Westover, L. A., [1991] *SPLATTING: A Parallel, Feed-Forward Volume Rendering Algorithm.* Ph.D. Dissertation, Technical Report TR91-029, University of North Carolina, Chapel Hill, NC, 1991.

Wolberg, G., [1990] *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, CA, 1990.

Appendix

The follow C code fragment demonstrates the forward-mapping evaluation of the $k(4,2)$ reconstruction kernel:

```
short key[4][128];
short table[4][128][2];

splat42(tile, value, qindex)

short tile[8][8];
short value;
short qindex;
{
    register short hash = value & 127;
    register short *tptr = table[qindex][hash];
    register short c0, c1, c2, c3;
    register int coeff;

    if (key[qindex][hash] != value) {
        /* dequantization step */
        coeff = value * quantizer[qindex][32];
        key[qindex][hash] = value;
        tptr[ 0] = c0 = (coeff * COS_03 + 128) / 256;
        tptr[ 1] = c1 = (coeff * COS_07 + 128) / 256;
        tptr[ 2] = c2 = (coeff * COS_01 + 128) / 256;
        tptr[ 3] = c3 = (coeff * COS_05 + 128) / 256;
    } else {
        c0 = tptr[ 0];
        c1 = tptr[ 1];
        c2 = tptr[ 2];
        c3 = tptr[ 3];
    }
    tile[0][0] += c0; tile[0][1] -= c1; tile[0][2] -= c2; tile[0][3] -= c3;
    tile[0][4] += c3; tile[0][5] += c2; tile[0][6] += c1; tile[0][7] -= c0;
    tile[1][0] -= c0; tile[1][1] += c1; tile[1][2] += c2; tile[1][3] += c3;
    tile[1][4] -= c3; tile[1][5] -= c2; tile[1][6] -= c1; tile[1][7] += c0;
    tile[2][0] -= c0; tile[2][1] += c1; tile[2][2] += c2; tile[2][3] += c3;
    tile[2][4] -= c3; tile[2][5] -= c2; tile[2][6] -= c1; tile[2][7] += c0;
    tile[3][0] += c0; tile[3][1] -= c1; tile[3][2] -= c2; tile[3][3] -= c3;
    tile[3][4] += c3; tile[3][5] += c2; tile[3][6] += c1; tile[3][7] -= c0;
    tile[4][0] += c0; tile[4][1] -= c1; tile[4][2] -= c2; tile[4][3] -= c3;
    tile[4][4] += c3; tile[4][5] += c2; tile[4][6] += c1; tile[4][7] -= c0;
    tile[5][0] -= c0; tile[5][1] += c1; tile[5][2] += c2; tile[5][3] += c3;
    tile[5][4] -= c3; tile[5][5] -= c2; tile[5][6] -= c1; tile[5][7] += c0;
    tile[6][0] -= c0; tile[6][1] += c1; tile[6][2] += c2; tile[6][3] += c3;
    tile[6][4] -= c3; tile[6][5] -= c2; tile[6][6] -= c1; tile[6][7] += c0;
    tile[7][0] += c0; tile[7][1] -= c1; tile[7][2] -= c2; tile[7][3] -= c3;
    tile[7][4] += c3; tile[7][5] += c2; tile[7][6] += c1; tile[7][7] -= c0;
}
```

One could also further reduce the control circuitry required by recognizing that there are fewer than 384 actual configurations of a given quadrant, as implied by Table 1.

Thus, with a highly regular structure requiring a single input of approximately 16-bits and 10-12 controls bits, a low cost and high-speed implementation of a 2-D 8×8 IDCT can be realized. Futher enhancements might include combining the look-up tables and the accumulator array on the same chip along with a suitable state-machine which would automatically step through each of the kernel values corresponding to a given quantized coefficient value.

5. Conclusions

We have presented an algorithm for the efficient evaluation of the IDCT which uses a forward-mapping approach. It typically requires 2 to 6 times less computation than other fast algorithms. Our approach takes unique advantage of the decorrelation properties of the DCT, in that it requires no work for zero valued coefficients. Multiplies are essentially eliminated through the use of table look-ups and caching. The FMIDCT also provides a unique tradeoff that exchanges quality for further reductions in computation. Thus, in addition to being amenable to a simple and regular hardware realization, this technique allows for an efficient software-only realization that provides performance previously only achievable via hardware assisted approaches, yielding a low-cost playback-only solution for DCT based compression standards.

6. References

Ahmed, N., T. Natarajan, and K. R. Rao, [1974] "Discrete Cosine Transform," *IEEE Trans. on Computers, vol. 23*, pp. 90-93, Jan 1974.

Chen, W. H., C. H. Smith, and S. C. Fralick, [1977] "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. on Communication, vol. 25*, pp. 1004-1009, Sept 1977.

Jain, A. K., [1979] "A Sinusoidal Family of Unitary Transforms," *IEEE Trans. on Pattern Analysis and Machine Intelligence, vol 1*, **no. 4**, pp. 356-365, Oct 1979.

Kamangar, F. A. and K. R. Rao, [1982] "Fast Algorithms for the 2-D Discrete Cosine Transform," *IEEE Trans. on Computers, vol. C-31*, **no. 9**, pp. 899-806, Sept 1982.

Knuth, D. E., [1973] *Art of Computer Programming, Volume 3: Sorting and Searching* Addison-Wesley, Reading, Mass., 1973.

LeGall, D., [1991] "MPEG: A Video Compression Standard for Multimedia Applications", *Com. of the ACM, vol. 34*, **no. 4**, pp. 46-58, April 1991.

Lee, B. G., [1984] "A New Algorithm for the Discrete Cosine Transform," *IEEE Trans. on Acoustics, Speech, and Signal Processing, vol 32*, pp. 1243-1245, Dec 1984.

Ligtenberg, A. and J. H. O'Neil, [1987] "A Single Chip Solution for an 8 by 8 Two Dimensional DCT," *International Symposium on Circuits and Systems*, **ISCAS 87**, pp. 1128-1131, Philadelphia, Pa, May, 5-7, 1987.

Liou, M. [1991] "Overview of the px64 kbit/s Video Coding Standard", *Com. of the ACM, vol. 34*, **no. 4**, pp. 59-63, April 1991.

Narasimha, M. J., and A. M. Peterson, [1978] "On the Computation of the Discrete Cosine Transform," *IEEE Trans. on Com., vol. 26*, pp. 934-946, June 1978.

Tseng, B. D. and W. C. Miller, [1978] "On Computing the Discrete Cosine Transform," *IEEE Trans. on Computing, vol. 27*, pp. 966-968, Oct 1978.

levels and the variance of the coefficient's distribution varies for each coefficient, the system could size each cache accordingly.

4.2 Hardware Realization Potential

Another advantage of the FMIDCT is its suitability for VLSI implementation. The FMIDCT can be accomplished using a simple accumulator block repeated $N \times N$ times as shown in Figure 1.

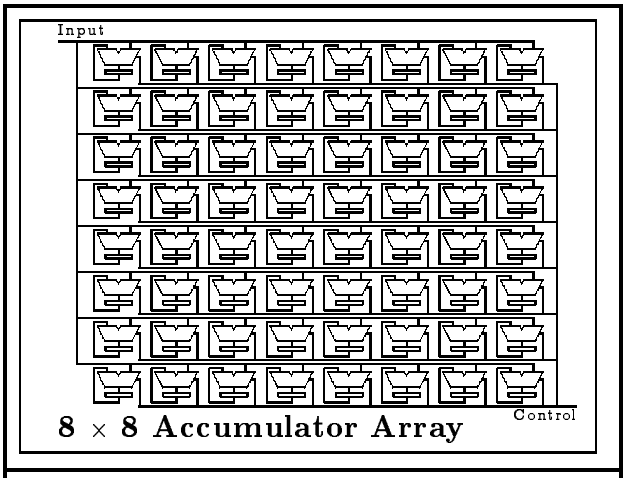

Figure 1: Hardware Implementation

It is possible to precede the data-path described with a single multiplier to perform both dequantization and scaling by the unit-reconstruction kernel. However, it is probably more efficient for a supporting controller to perform this operation using the caching techniques described earlier, given the infrequency of multiplies. Every accumulator in the array could be made to share a common data input I, with distributed control for each accumulator. Each accumulator is only required to perform four distinct operations. $A_{xy} \leftarrow I, A_{xy} \leftarrow A_{xy}, A_{xy} \leftarrow A_{xy} + I, A_{xy} \leftarrow A_{xy} - I$. The IDCT applys the scaled unique values of the reconstruction kernel to the array along with an appropriate control word, for each non-zero coefficient.

The amount of control circuitry can be reduced by recognizing that each reconstruction kernel can be categorized into one of four possible forms.

$$\begin{bmatrix} [Q] & [H] \\ [V] & [D] \end{bmatrix}, \qquad \begin{bmatrix} [Q] & -[H] \\ [V] & -[D] \end{bmatrix}, \qquad \begin{bmatrix} [Q] & [H] \\ -[V] & -[D] \end{bmatrix}, \qquad \begin{bmatrix} [Q] & -[H] \\ -[V] & [D] \end{bmatrix}$$

where:

$$H = QR, \quad V = RQ, \quad D = RQR \qquad \text{and } \mathbf{R} \text{ is defined as} \qquad R = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Therefore, independent control is only required for the accumulators within a single quadrant of the array, with an addition two-bits to select the appropriate symmetry.

routines. About one fifth of a sequence's playback time is spent in the bit-stream parsing and Huffman decoding routines, while the remaining time is spent in various bookkeeping routines.

Table 2 gives some statistics on a few sample image sequences. The "target" sequence is the 30 second Sun Microsystems, Inc. client-server computing commercial. The "mifkin" sequence is a 60 second sequence with 19 scene changes. The f16, Lenna, mandrill, and Zelda images are commonly studied in the image processing field.

image	Resolution	Frames	IDCTs	non-zero Coeffs	Mults	$\frac{Mults}{IDCT}$
f16	512×496	1	5952	4.90	4183	0.703
Lenna	512×496	1	5952	4.67	3726	0.626
mandrill	512×496	1	5952	11.28	5328	0.895
Zelda	512×496	1	5952	4.33	3336	0.560
target	320×224	300	504000	5.08	5425	0.011
mifkin	320×224	600	1008000	3.80	5167	0.006

Table 2:Result Table

4.1 Software Implementation Details

The FMIDCT uses table look-ups for most of its multiplications. The current implementation has a separate table or cache for each DCT coefficient. The lower bits of the quantized input value act as the index into the cache. If the last key-value stored in the cache is the same as the current input value, the system simply uses the results that are stored in the cache. If the values disagree, the system dequantizes the value, performs the required multiplies needed for this DCT coefficient and loads these results in the cache. Choosing the least significant bits of the quantized value guarantees that the most probable values, (1, -1, 2, and -2) do not map to the same cache location, and increases the likelihood that these values may stay in the cache once they are calculated.

The values in Table 2 illustrate the effectiveness of this approach. For the sample images, each DCT block has between 4 and 12 non-zero AC coefficients on average. Each of these coefficients require between 1 and 10 multiplies to scale their respective reconstruction kernels. Yet, the average number of multiplies per block is below 1 for all cases, and is approaching 0 for the long sequence.

The current cache implementation uses approximately 112 kilobytes of storage. Since we use the least significant 7 bits as an index to the cache, there are 128 different cache locations for each of the 64 different caches. The size of the cache can be reduced by using fewer bits as the index. Since the number of quantization

would require 63 adds per result, typically the algorithm requires less than 20% of that number, and in the best case, where the DC coefficient is the only non-zero coefficient, requires no adds. The highly uncorrelated data sequences which are necessary to generate a large number of adds are rare in practice and furthermore, they are generally ill-suited for DCT coding.

One of the most important properties of the DCT that is often exploited in compression algorithms is the high probability, after appropriate quantization, of zero valued coefficients [Wallace91][LeGall91][Liou91]. These coefficients imply no work for the FMIDCT. Table 2 gives the average number of non-zero AC coefficients occurring in a variety of images out of a possible 63. Since each non-zero coefficient implies the accumulation of a scaled reconstruction kernel, this number indicates the average number of additions required for each pixel. This data suggests that the FMIDCT required only 6% to 17% of the additions implied by the worst case. Futhermore, based on the multiplier equivalence arguments presented earlier, this algorithm is between 2 to 6 times faster than other well known approachs.

3.3 Quality versus Speed Trade-off

This formulation of the IDCT is unique in providing a quality versus computation-time trade-off. Since each input is handled independently, we can accumulate the various spatial frequency inputs in a priority order. Such an ordering can be approximated by zig-zag ordering. The accumulation of reconstruction kernels can be stopped at any point in order to achieve a given performance level. We call this approach *incremental evaluation*. This notion is similar to that of progressive transmission of images [Wang88][Tzou87], where an approximate reconstruction of an image is gradually built up with increasing fidelity while a viewer controls how long the process should continue. The motivation for progressive transmission is to make effective use of channel bandwidth at the cost of requiring many invocations of the IDCT. In contrast, the incremental evaluation approach places a upper limit on the computational cost for the IDCT, trading off image quality for decoding speed. These two techniques may be combined, where each successive component in the progressive scheme implies one additional accumulation in the FMIDCT.

4. Results

We have implemented a baseline JPEG decoder in software that utilizes the FMIDCT which can perform in excess of 50,000 8×8 transforms per second on a 40 Mhz Sun SparcStation 2. This implementation is capable of decoding and displaying 320×240 image sequences at rates in excess of 10 frames per second. In our implementation, each 16×16 pixel macroblock is represented by 4 8×8 blocks of luminance values and 2 8×8 blocks of subsampled chrominance values. These frames have 280 macroblocks, so each image requires 1680 8×8 IDCTs. Less than one third of a sequence's playback time is spent in the IDCT and related routines. Another third of a sequence's playback time is spent in the conversion of the luminance/chrominance representation to red, green, blue and dithering

of unique coefficients of the unit-valued reconstruction kernels in each of the 8×8 cases.

v/u	0	1	2	3	4	5	6	7
0	1	4	2	4	1	4	2	4
1	4	10	8	10	4	10	8	10
2	2	8	3	8	2	8	3	8
3	4	10	8	10	4	10	8	10
4	1	4	2	4	1	4	2	4
5	4	10	8	10	4	10	8	10
6	2	8	3	8	2	8	3	8
7	4	10	8	10	4	10	8	10

Table 1:Unique entries for each DCT coefficient

This suggests that an 8×8 IDCT can be accomplished with just 384 multiplies instead of the 4096 implied by the matrix equation. This reduction is less impressive than the 128 required multiplies achievable through other fast 2-D algorithms [Kamangar82], however, the computation requirements of the FMIDCT can be further reduced using techniques which are not amenable to other fast algorithms.

In most applications the IDCT is preceded by a dequantization process. This leads to a finite limit on the actual number of reconstruction levels. Generally, dequantization is performed prior to the IDCT. In the FMIDCT, the dequantization and the scaling may be combined into a single step. Since there are a finite number of reconstruction levels for each coefficient and each coefficient is treated independently, this step may be realized as a table look-up, as described in the Software Implementation Detail section.

The fact that existing fast implementations of the IDCT do not treat each input value independently limits the usefulness of performing multiplications by table-look-ups. In these algorithms, linear combinations of several input values, generated by the matrix products of the previously applied matrix decompositions, are multiplied by scalar constants. While these linear combinations are also constrained to take on a finite number of unique values, this number quickly becomes impractical for implementation by table look-up, since it is proportional to the product of the number of reconstruction levels for each of its constituent inputs.

3.2 Special Case of Zero Coefficients

Well known fast IDCT algorithms [Kamangar82] require as few as 400 adds and 128 multiplies per 8×8 block. If we assume each multiply is equivalent to 5 adds, (allowing 12-bits per scalar constant and assuming the use of Booth's algorithm), this results in more than 1024 adds per block or approximately 16 adds per result. More commonly used approaches, where the 2-D IDCT is separated into 2N 1-D IDCTs, require on the order of 128 additional multiplies resulting in 26 adds per output. While, in the worst case, a FMIDCT that utilizes table-driven multiplies,

domain coefficients are frequently quantized to a small number of reconstruction levels, this multiply can be replaced by a table-look-up. Second, zero-valued coefficients will make no contribution to the output vector, thereby eliminating the scaling and accumulation steps. Since there is a high incidence of these zero-valued coefficients, due to the decorrelation properties of the DCT and the subsequent quantization, this results in a significant reduction in computation. Finally, the FMIDCT provides a continuous quality versus computation time trade-off.

3.1 Table Driven Multiplication

Although the unit-valued reconstruction kernels are composed of N^2 values, the magnitude of at most $\frac{N^2+2N}{8}$ of these values is unique. This repetition allows the scaling multiply to be calculated only once and accumulated into the output vector at each repeated position. This is demonstrated by the following example 8×8 IDCT kernels. Let

$$C_{xy} = f(x)f(y)\cos\left(\frac{x\pi}{16}\right)\cos\left(\frac{y\pi}{16}\right),$$

$$k(0,1) = \begin{bmatrix} C_{01} & C_{03} & C_{05} & C_{07} & -C_{07} & -C_{05} & -C_{03} & -C_{01} \\ C_{01} & C_{03} & C_{05} & C_{07} & -C_{07} & -C_{05} & -C_{03} & -C_{01} \\ C_{01} & C_{03} & C_{05} & C_{07} & -C_{07} & -C_{05} & -C_{03} & -C_{01} \\ C_{01} & C_{03} & C_{05} & C_{07} & -C_{07} & -C_{05} & -C_{03} & -C_{01} \\ C_{01} & C_{03} & C_{05} & C_{07} & -C_{07} & -C_{05} & -C_{03} & -C_{01} \\ C_{01} & C_{03} & C_{05} & C_{07} & -C_{07} & -C_{05} & -C_{03} & -C_{01} \\ C_{01} & C_{03} & C_{05} & C_{07} & -C_{07} & -C_{05} & -C_{03} & -C_{01} \\ C_{01} & C_{03} & C_{05} & C_{07} & -C_{07} & -C_{05} & -C_{03} & -C_{01} \end{bmatrix},$$

$$k(1,1) = \begin{bmatrix} C_{11} & C_{13} & C_{15} & C_{17} & -C_{17} & -C_{15} & -C_{13} & -C_{11} \\ C_{13} & C_{33} & C_{35} & C_{37} & -C_{37} & -C_{35} & -C_{33} & -C_{13} \\ C_{15} & C_{35} & C_{55} & C_{57} & -C_{57} & -C_{55} & -C_{35} & -C_{15} \\ C_{17} & C_{37} & C_{57} & C_{77} & -C_{77} & -C_{57} & -C_{37} & -C_{17} \\ -C_{17} & -C_{37} & -C_{57} & -C_{77} & C_{77} & C_{57} & C_{37} & C_{17} \\ -C_{15} & -C_{35} & -C_{55} & -C_{57} & C_{57} & C_{55} & C_{35} & C_{15} \\ -C_{13} & -C_{33} & -C_{35} & -C_{37} & C_{37} & C_{35} & C_{33} & C_{13} \\ -C_{11} & -C_{13} & -C_{15} & -C_{17} & C_{17} & C_{15} & C_{13} & C_{11} \end{bmatrix} \quad \text{and}$$

$$k(4,4) = \begin{bmatrix} C_{44} & -C_{44} & -C_{44} & C_{44} & C_{44} & -C_{44} & -C_{44} & C_{44} \\ -C_{44} & C_{44} & C_{44} & -C_{44} & -C_{44} & C_{44} & C_{44} & -C_{44} \\ -C_{44} & C_{44} & C_{44} & -C_{44} & -C_{44} & C_{44} & C_{44} & -C_{44} \\ C_{44} & -C_{44} & -C_{44} & C_{44} & C_{44} & -C_{44} & -C_{44} & C_{44} \\ C_{44} & -C_{44} & -C_{44} & C_{44} & C_{44} & -C_{44} & -C_{44} & C_{44} \\ -C_{44} & C_{44} & C_{44} & -C_{44} & -C_{44} & C_{44} & C_{44} & -C_{44} \\ -C_{44} & C_{44} & C_{44} & -C_{44} & -C_{44} & C_{44} & C_{44} & -C_{44} \\ C_{44} & -C_{44} & -C_{44} & C_{44} & C_{44} & -C_{44} & -C_{44} & C_{44} \end{bmatrix}.$$

Notice that only 4 unique multiples are necessary for the accumulation of the $k(0,1)$ kernel, 10 for the $k(1,1)$ kernel, and only 1 for $k(4,4)$. Table 1 depicts the number

As a matrix equation, this is written as:

$$\begin{bmatrix} o_{00} \\ o_{01} \\ \vdots \\ o_{xy} \\ \vdots \\ o_{LL} \end{bmatrix} = \begin{bmatrix} & & & & \\ r_0^{xy} & r_1^{xy} & r_2^{xy} & \dots & r_M^{xy} \\ & & \vdots & & \end{bmatrix} \begin{bmatrix} i_{00} \\ i_{01} \\ \vdots \\ i_{uv} \\ \vdots \\ i_{LL} \end{bmatrix}.$$

When using the forward-mapping approach, the output vector is formed by the successive accumulation of each system matrix column scaled by the corresponding input value.

$$\bar{O} = \sum_{uv} i_{uv} C^{uv} \qquad \text{where} \qquad C^{uv} = \begin{bmatrix} c_0^{uv} \\ c_1^{uv} \\ \vdots \\ c_M^{uv} \end{bmatrix}$$

with

$$c_k^{uv} = f(u)f(v) \times \cos\left(\frac{\pi(2(k \ mod \ N)k+1)u}{2N}\right) \cos\left(\frac{\pi(2(k \ div \ N)+1)v}{2N}\right).$$

As a matrix equation, this is written as:

$$\begin{bmatrix} o_{00} \\ o_{01} \\ \vdots \\ o_{LL} \end{bmatrix} = i_{00} \begin{bmatrix} c_0^{00} \\ c_1^{00} \\ \vdots \\ c_M^{00} \end{bmatrix} + i_{01} \begin{bmatrix} c_0^{01} \\ c_1^{01} \\ \vdots \\ c_M^{01} \end{bmatrix} + \ldots + i_{LL} \begin{bmatrix} c_0^{LL} \\ c_1^{LL} \\ \vdots \\ c_M^{LL} \end{bmatrix}$$

[Westover90] [Westover91] has demonstrated the efficiency of the forward-mapping approach for volumetric reconstruction. The advantages are pronounced when the input vector is sparse. The forward-mapping process incrementally accumulates the energy contributed from each input element into the output vector. Each transform domain coefficient scales the corresponding matrix column. We call this matrix column vector the *reconstruction kernel* of the input coefficient. The scaled reconstruction kernel is then accumulated with the output vector, $\bar{O}$.

3. Properties of FMIDCT

Previous algorithms have been formulated to be computed in some minimal, yet constant, number of operations. This leads to constant time evaluation, which is independent of the input sequence applied. However, the FMIDCT is input sequence dependent, since the amount of computation required is proportional to the number of non-zero coefficients. This situation is similar to that of the well known quicksort algorithm [Knuth73], in which its worst case behavior varies significantly from its average case behavior. In such cases it is useful to not only discuss the average behavior, but also the likelihood of the worse case scenario.

There are three ways in which the FMIDCT reduces computation. First, the reconstruction kernels exhibit considerable symmetry thereby reducing the number of unique multiplies required in the scaling process. Furthermore, since the transform

elements independently. This formulation is referred to as a *forward-mapping* system [Wolberg90]. A system can calculate a weighted average by either gathering energy from each contributing input to calculate each output, or by spreading the energy from each input to all affected outputs. The final result is independent of the evaluation order of these energy contributions.

We will introduce a forward-mapping derivation of a fast IDCT algorithm (FMIDCT) which exploits the statistical properties of its input sequence. Although the discussion will concentrate on the 2-D IDCT algorithm, the techniques described are easily adapted to other dimensions and other orthogonal transforms with similar statistical properties. We also discuss a further optimization to the FMIDCT which takes advantage of the quantization of the transform domain coefficients. Finally, we will demonstrate the unique capability of the FMIDCT to trade-off reconstruction quality in exchange for reductions in computation.

2. FMIDCT Evaluation

A type II, $N \times N$, 2-D IDCT, which is commonly used in image compression, is expressed as

$$o(x,y) = \sum_{v=0}^{N-1}\sum_{u=0}^{N-1} f(u)f(v)i(u,v) \times \cos\left(\frac{\pi(2x+1)u}{2N}\right)\cos\left(\frac{\pi(2y+1)v}{2N}\right), \qquad (1)$$

$$x, y: \rightarrow [0, N-1] \qquad \text{and} \qquad f(i) = \begin{cases} \frac{\sqrt{2}}{2}, & \text{for } i = 0; \\ 1, & \text{otherwise.} \end{cases}$$

This equation expressed as a linear system is $\bar{O} = \boldsymbol{C}\bar{I}$, where $\bar{I}$ and $\bar{O}$ are N^2-dimensional vectors constructed from the row-ordered enumeration of the $N \times N$ transform domain input sequence, $i(u,v)$, and the reconstructed output sequence, $o(x,y)$, respectively. The $N^2 \times N^2$ system matrix, $\mathbf{C}$, is composed of the input weighting terms and is defined as follows:

$$c(yN+x, vN+u) = f(u)f(v)\cos\left(\frac{\pi(2x+1)u}{2N}\right)\cos\left(\frac{\pi(2y+1)v}{2N}\right).$$

Equation (1) may be evaluated using either matrix decompositions, inverse-mapping procedures, or forward-mapping procedures. Previously, fast algorithms [Chen77][Lee84][Ligtenberg87][Kamangar82] have concentrated almost exclusively on matrix decomposition evaluation procedures, where the system matrix, $\mathbf{C}$, is factored into a set of sparse matrices.

Inverse-mapping procedures are equivalent to calculating each element of the output vector, $\bar{O}$, by taking the inner-product of a corresponding row in the system matrix, $\mathbf{C}$ and the input vector, $\bar{I}$. Let $L = N-1$ and $M = N^2-1$, then

$$\bar{O} = \bar{R}\bar{I} \qquad \text{or} \qquad O_{xy} = R^{xy}\bar{I}, \qquad \text{where} \qquad R^{xy} = [r_0^{xy} r_1^{xy} \ldots r_M^{xy}]$$

with

$$r_k^{xy} = f(k \ mod \ N)f(k \ div \ N)\cos\left(\frac{\pi(2x+1)(k \ mod \ N)}{2N}\right)\cos\left(\frac{\pi(2y+1)(k \ div \ N)}{2N}\right).$$

A Forward-Mapping Realization of the Inverse Discrete Cosine Transform

Leonard McMillan

Lee Westover

Sun Microsystems, Inc.
Research Triangle Park, NC 27709

Abstract

This paper presents a new realization of the Inverse Discrete Cosine Transform (IDCT). It exploits both the decorrelation properties of the Discrete Cosine Transform (DCT) and the quantization process that is frequently applied to the DCT's resultant coefficients. This formulation has several advantages over previous approaches, including the elimination of multiplies from the central loop of the algorithm and its adaptability to incremental evaluation. The technique provides a significant reduction in computational requirements of the IDCT, enabling a software-based implementation to perform at rates which were previously achievable only through dedicated hardware.

1. Introduction

Since its introduction, the DCT [Ahmed74] has found widespread use in the field of image processing. [Jain79] has demonstrated that for data exhibiting high correlation, the DCT performs close to the ideal Karhunen-Loeve Transform. The DCT representation of a data sequence tends to concentrate the most variance (energy) into the fewest transform coefficients. This results in a transform domain description which is sparse compared to the original input sequence.

The application of the DCT to many interesting classes of data, particularly continuous tone images, has motivated the search for fast and efficient algorithms. The earliest fast algorithms were based on approaches originally developed for the Fast Fourier Transform, in which the periodicity and recursive nature of the underlying basis functions were exploited [Narasimha78][Tseng78][Vetterli84]. Later, other fast algorithms were developed by considering various factorizations of the DCT's basis matrix [Chen77][Lee84][Ligtenberg87].

The structural similarities of the DCT to its inverse, the IDCT, has enabled each of the fast DCT algorithms to be easily adapted to their dual IDCT formulation. As a consequence, there has been little concentration on specific formulations of the IDCT and the unique statistical properties of this transform domain description.

Both IDCT and the DCT are easily expressed as a constant-coefficient linear system in which each output element is expressed as a finite weighted sum of input elements. Systems where each output value is directly evaluated in this fashion are called *inverse-mapping* systems [Wolberg90]. An alternate evaluation approach is to express each input element's contribution to the entire set of output