Low-Complexity Transform and Quantization in H.264/AVC

Henrique S. Malvar, *Fellow, IEEE*, Antti Hallapuro, Marta Karczewicz, and Louis Kerofsky, *Member, IEEE*

***Abstract*—This paper presents an overview of the transform and quantization designs in H.264. Unlike the popular 8×8 discrete cosine transform used in previous standards, the 4×4 transforms in H.264 can be computed exactly in integer arithmetic, thus avoiding inverse transform mismatch problems. The new transforms can also be computed without multiplications, just additions and shifts, in 16-bit arithmetic, thus minimizing computational complexity, especially for low-end processors. By using short tables, the new quantization formulas use multiplications but avoid divisions.**

***Index Terms*—Integer transforms, periodic quantization, quantization, transforms, video coding, video standards.**

I. Introduction

THE NEW H.264 video coding standard provides a compression gain of $1.5\times - 2.0\times$ over previous standards such as $\mathrm{H.263+}$ and MPEG-4 Part 2 [1]. The H.264 architecture has many innovations when compared to $\mathrm{H.263+}$ [2], [3], such as hybrid predictive/transform coding of intra frames and integer transforms. In particular, in this paper we review the new low-complexity transform and quantization approaches that are unique to H.264 [4]–[6]. The transforms employ only integer arithmetic without multiplications, with coefficients and scaling factors that allow for 16-bit arithmetic computation on first-level transforms. These changes lead to a significant complexity reduction, with an impact in peak signal-to-noise ratio (PSNR) of less than 0.02 dB [4]–[7].

In Section II we present the design requirements for the new H.264 transform. In Section III we review basic ideas for integer transform design and present the length-4 transform adopted in H.264. In Section IV, we consider the quantization procedures, and in Section V we present a design that allows for transform and quantization computations in 16-bit arithmetic. Additional aspects are considered in Section VI.

II. Design Requirements of the H.264 Transform

The structure of H.264 imposes several requirements on the design of residual coding. In traditional work, residual decoding contains the possibility of drift (mismatch between the decoded data in the encoder and decoder). The drift arises from the fact that the inverse transform is not fully specified in integer arithmetic; rather it must satisfy statistical tests of accuracy compared with a floating point implementation of the inverse transform (e.g., the inverse discrete cosine transform (IDCT) accuracy specification in H.261 Annex A). On one hand, this freedom gives the implementer flexibility in adapting to a particular architecture, but in the other hand the cost of this flexibility is the introduction of prediction drift. Several methods have been introduced to control the accumulation of drift, ranging from the forced intra refresh requirements to different oddification techniques [13], [14]. H.264 makes extensive use of prediction, since even the intra coding modes rely upon spatial prediction. As a result, H.264 is very sensitive to prediction drift. In prior standards, prediction drift accumulation can occur once per P-frame. In contrast, in H.264 prediction drift can occur much more frequently. As an illustration, in an I-frame, 4×4 blocks can be predicted from their neighbors. At each stage prediction drift can accumulate. For a CIF image, which has a width of 88 4×4 blocks, prediction drift can accumulate 88 times in decoding one row of an I-frame. Thus, it is clear that as a result of the extensive use of prediction with H.264, the residual coding must be drift-free.

A drift-free decoding design places exacting requirements upon a decoder implementation, raising the issue of complexity. Since all decoders must implement the drift-free inverse transform exactly, the implementation complexity on any expected decoder architecture (ASIC, media processor, DSP, general CPU) must be considered. The algorithm must not place excessive burdens on any expected architecture. The main bottlenecks with the inverse transform in the initial drafts [2] were the need for 32-bit multiplication and 32-bit memory access. A set of criteria were developed to restrict the complexity of the inverse transform [13]. Requirements were using only 16-bit multiplication and 16-bit memory access. Two desirable features were an entirely 16-bit implementation; even the arithmetic and logic unit (ALU) is 16-bits, and the possibility of alternate implementations giving mathematically exact algorithms. Additionally, memory requirements were also considered, particularly for inverse quantization and scaling.

An early design feature of H.264 was the variation of the quantization step size. The quantization step size increases by approximately 12% for each increase in quantization parameter, so that each increment of six in the quantization parameter doubles the quantization step size. This allows a wide range of quality levels to be addressed efficiently and succinctly. At low quantization, fine quantization control is possible while at high quantization, coarse quantization is not burdened. During the

Manuscript received April 15, 2003; revised May 9, 2003.

H. S. Malvar is with Microsoft Research, Redmond, WA 98052 USA (e-mail: malvar@microsoft.com).

A. Hallapuro is with the Nokia Research Center, FIN-3372, Tampere, Finland (e-mail: antti.hallapuro@nokia.com).

M. Karczewicz is with the Nokia Research Center, Irving, TX 75039 USA (e-mail: marta.karczewicz@nokia.com).

L. Kerofsky is with Sharp Laboratories of America, Camas, WA 98607 (e-mail: lkerofsky@sharplabs.com).

Digital Object Identifier 10.1109/TCSVT.2003.814964

development of the transform, the quality range was extended at the low end by making the smallest quantization step size one quarter of its original value.

Currently, H.264 supports 8-bit pixel data. It was expected during the design that support for 10- or 12-bit data will be needed. Initial proposals for higher bit depth were presented in [4]. At the time of this writing, there is a call for extensions of the bit-depth of H.264 [16].

Compression efficiency is the ultimate reason for introducing a transform. Thus, the transform must efficiently exploit spatial correlation to aid in compression. As mentioned above, H.264 relies heavily on prediction before the transform. The use of 4×4 motion segmentation or spatial prediction [2] significantly reduces the spatial correlation between 4×4 blocks, motivating the choice of a 4×4 transform. In H.264, the transform codes prediction error signals, which differ from the statistics of natural images often used to justify the selection of a transform. The compression performance of the transform design must be evaluated on segmented prediction error signals. As we discuss briefly in Section III, a coding gain analysis [6] has shown that despite the increased complexity, the discrete cosine transform (DCT) does not give better compression than the transform selected.

As we mentioned above, one of the new aspects in H.264 is the use of a 4×4 transform block size, whereas previous video coding standards used the 8×8 DCT. This smaller block size leads to a significant reduction in ringing artifacts. Compression gain is improved by using inter-block pixel prediction for intra-coded frames [2], so that the transform is applied to prediction residuals. With that approach, H.264 intra-frame coding leads to better compression than systems based on 8×8 DCT blocks, and also better compression than state-of-the-art image coders such as JPEG2000 [8].

The length-4 transform originally proposed in [2] is an integer orthogonal approximation to the DCT, allowing for bit-exact implementation for all encoders and decoders, thus solving the drift problem discussed above. The new transform, proposed in [4] and adopted in the standard, has the additional benefit of removing the need for multiplications.

For improved compression efficiency, H.264 also employs a hierarchical transform [9] structure, in which the DC coefficients of neighboring 4×4 transforms are grouped in 4×4 blocks and transformed again by a second-level transform. This hierarchical transform is discussed further in Section VI.

III. Integer Transform Design

The DCT [10] is commonly used in block transform coding of images and video, e.g., JPEG and MPEG, because it is a close approximation to the statistically optimal Karhunen–Loève transform, for a wide class of signals [9], [10]. The DCT maps a length-N vector $\mathbf{x}$ into a new vector $\mathbf{X}$ of transform coefficients by a linear transformation $\mathbf{X} = \mathbf{H}\,\mathbf{x}$, where the element in the kth row and nth column of $\mathbf{H}$ is defined by

$$\mathbf{H}_{kn} = H(k,n) = c_k\sqrt{\frac{2}{N}}\cos\left[\left(n+\frac{1}{2}\right)\frac{k\pi}{N}\right] \tag{1}$$

for the frequency index $k = 0, 1, \ldots, N-1$, and sample index $n = 0, 1, \ldots, N-1$, with $c_0 = \sqrt{2}$ and $c_k = 1$ for $k > 0$. The DCT matrix is orthogonal, that is $\mathbf{x} = \mathbf{H}^{-1}\mathbf{X} = \mathbf{H}^T\mathbf{X}$ (where the superscript T denotes transposition).

A disadvantage of the DCT is that the entries $H(k,n)$ are irrational numbers. Thus, in a digital computer, when we compute the direct and inverse transform in cascade, we may not get exactly the same data back. When we compute $\mathbf{X} = \mathbf{H}\,\mathbf{x}$ and $\mathbf{u} = \mathrm{round}\{\mathbf{H}^T\mathbf{X}\}$, we may not get $u(n) = x(n)$ for all n if the direct and inverse transforms are implemented in different machines with different floating-point representations and rounding. If we introduce appropriate scale factors β and γ and define $\mathbf{X} = \mathrm{round}\{\gamma\,\mathbf{H}\,\mathbf{x}\}$ and $\mathbf{u} = \mathrm{round}\{\beta\mathbf{H}^T\mathbf{X}\}$, then we can make $u(n) = Gx(n)$, where G is an integer, for almost all n by choosing β large enough and γ appropriately. Nevertheless, we cannot guarantee an exact result unless we standardize on rounding procedures for intermediate results.

Thus, it is desirable to replace $\mathbf{H}$ by an orthogonal matrix with integer entries. For that, two basic approaches can be used: one is to build $\mathbf{H}$ with just a few integers, with symmetries similar to those of the DCT, to guarantee orthogonality and approximate a uniform frequency decomposition [11], for example [5]

$$\mathbf{H} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}. \tag{2}$$

For the original H.264 design [2], the choices are $a = 13$, $b = 17$, and $c = 7$. That makes $\mathbf{H}$ quite close to a scaled DCT, and also ensures all rows have the same norm, because $2 \times 13^2 = 17^2 + 7^2$.

Another approach is to round the scaled entries of the DCT matrix to nearest integers [6]

$$\mathbf{H} = \mathrm{round}\left\{\alpha\mathbf{H}_{\mathrm{DCT}}\right\} \tag{3}$$

where $\mathbf{H}_{\mathrm{DCT}}$ is the DCT matrix. If we set $\alpha = 26$, then we get exactly the same solution as in the original TML design in [2].

The main problem with the $\{a = 13, b = 17, c = 7\}$ choice is the increase in dynamic range. If $\max\{|x(n)|\} = A$, then $\max\{|X(k)|\} = 52$ A, i.e., the transform has a dynamic range gain of 52. Since we compute two-dimensional transforms by transforming rows and columns, the total gain is $52^2 = 2704$. Since $\log_2(2704) = 11.4$, we need 12 more bits to store $X(k)$ than to store $x(n)$. That would lead to the need of 32-bit arithmetic to compute transforms and inverse transforms in the original design [2]. To overcome that limitation, we proposed the matrix obtained by setting $\alpha = 2.5$ in (3), which leads to the new set of coefficients $\{a = 1, b = 2, c = 1\}$, that is

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}. \tag{4}$$

That way, the maximum sum of absolute values in any row of $\mathbf{H}$ equals 6, so the maximum dynamic range gain increase for a

2-D transform is $\log_2(6^2) = 5.17$, i.e., storage of $X(k)$ needs only six more bits than $x(n)$.

With the design in (4), the rows of $\mathbf{H}$ are orthogonal but do not have the same norm. However, that can be easily compensated for in the quantization process, as we discuss in Section IV. From a compression standpoint, the transform coding gain of $\mathbf{H}$ is 5.38 dB, whereas the original TML-1 design or the DCT have a coding gain of 5.39 dB, for a stationary Gauss-Markov input with correlation coefficient $\rho = 0.9$. Since the input to the transform are prediction residuals, in practice the correlation coefficient is less than 0.9, so the loss in coding gain is even less than 0.01 dB, which is negligible in practice. Even if inter-block prediction is not used, the 0.01 dB coding gain difference means that there is no noticeable performance penalty in using the new design in (4).

A. *Inverse Transform*

In the decoder, we could use just the transpose of $\mathbf{H}$ in (4), as long as we take care of scaling the reconstructed transform coefficients appropriately, to compensate for the different row norms. However, in order to minimize the combined rounding errors from the inverse transform and reconstruction, we need to reduce the dynamic range gain. The problem is in the odd-symmetric basis functions, whose peak value is two.

Thus, we proposed in [4] scaling the odd-symmetric basis functions by 1/2; that is, replacing the rows $[2 \;\; 1 \;\; -1 \;\; -2]$ and $[1 \;\; -2 \;\; 2 \;\; -1]$ by $[1 \;\; 1/2 \;\; -1/2 \;\; -1]$ and $[1/2 \;\; -1 \;\; 1 \;\; -1/2]$, respectively. That way, the sum of absolute values of the odd functions is halved to three. Thus, the maximum sum of absolute values for any basis function now equals four (the sum for the even functions), which reduces the dynamic range gain for the 2-D inverse transform from 6^2 to 4^2. Since $\log_2(4^2) = 4$, we reduce the increase in dynamic range from 6 bits to 4 bits. The inverse transform matrix is then defined by [4]

$$\tilde{\mathbf{H}}_{\text{inv}} = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix}. \tag{5}$$

where the tilde indicates that $\tilde{\mathbf{H}}_{\text{inv}}$ is a scaled inverse of H, i.e.,

$$\tilde{\mathbf{H}}_{\text{inv}} \text{diag}\left\{\frac{1}{4}, \frac{1}{5}, \frac{1}{4}, \frac{1}{5}\right\} \mathbf{H} = \mathbf{I}. \tag{6}$$

The multiplications by 1/2 can be implemented by sign-preserving 1-bit right shifts [4], so all decoders produce identical results. A key observation is that the small errors caused by the right shifts are compensated by the 2-bit gain in the dynamic range of the input to the inverse transform.

Fig. 1 shows flowgraphs of the direct and inverse transforms as described above, which are applied to rows and columns of each 4×4 block. The complexity of these transforms is so low that the only way to reduce complexity any further would be to remove the shifts, which would turn the transforms into Hadamard transforms $(a = b = c = 1)$ [9], [10], with a significant reduction in coding gain and a significant increase in visual coding artifacts.

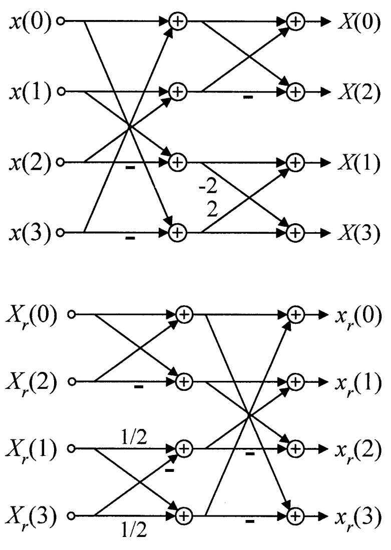

Fig. 1. Fast implementation of the H.264 direct transform (top) and inverse transform (bottom). No multiplications are needed, only additions and shifts.

IV. Quantization

In lossy compression, quantization is the step that introduces signal loss, for better compression. For a given step size Q_s, usually an integer, the encoder can perform quantization by

$$X_q(i,j) = \text{sign}\{X(i,j)\} \frac{|X(i,j)| + f(Q_s)}{Q_s} \tag{7}$$

where i and j are the row and column indices and $f(Q_s)$ controls the quantization width near the origin (the "dead zone"). The decoder can perform inverse quantization (reconstruction) by simply scaling the quantized data by Q_s

$$X_r(i,j) = Q_s X_q(i,j). \tag{8}$$

A disadvantage of the quantization formula (7) is that it requires integer divisions at the encoder. To avoid divisions, the original H.264 proposal in [2] replaces the formulas by

$$\begin{aligned} X_q(i,j) =& \text{sign}\{X(i,j)\}[(|X(i,j)|A(Q) + f2^L) \gg L] \\ X_r(i,j) =& X_q(i,j)B(Q) \\ \mathbf{x}_r =& (\mathbf{H}^T \mathbf{X}_r + 2^{N-1}\mathbf{e}) \gg N \end{aligned} \tag{9}$$

where $\mathbf{e} = [1\;1\;1\;1]^T$, the new parameter Q varies from zero to $Q_{\max}$, and the association of quantization parameters $A(Q)$ and $B(Q)$ are such that zero corresponds to the finest quantization and $Q_{\max}$ the coarsest quantization. Thus, the parameter $Q_{\max}$ is chosen for fine enough granularity. In the initial proposal in [2], $Q_{\max} = 31$. Note that the dead-zone control parameter f can be set differently for different encoders; it typically lies in the range 0 to 1/2. The integer values in the quantization table $A(Q)$ and reconstruction table $B(Q)$ must satisfy

$$A(Q)B(Q)G^2 \cong 2^{L+N} \tag{10}$$

where G is the squared norm of the rows of $\mathbf{H}$.

In (9), the L-bit right shift is equivalent to a division by 2^L, thus avoiding an actual division operation. The N-bit right shift approximates closely a division by 2^N, since an N-bit shift is actually equivalent to division with rounding toward minus infinity instead of toward zero (the term $2^{N-1}\mathbf{e}$ is an offset that minimizes the effect of rounding toward minus infinity). The values of L and N are chosen by a compromise: larger values reduce the approximation error in (10), whereas small values reduce the dynamic range of X_r and $|X(i,j)|A(Q)$. In the original H.264 design [2], $L = N = 20$. These values are large enough so that the error in the *AB* product is negligible, while keeping all variables within the limits of 32-bit signed integers.

V. 16-Bit Arithmetic and Periodic Quantization Tables

Although the quantization formulas in (9) are relatively simple, we can simplify the implementation complexity further by using formulas that allow for 16-bit arithmetic, with no penalty in PSNR performance. To achieve that goal, we reduce the values of $B(Q)$ and the parameters L and N.

Another aspect of the original H.264 quantization design in (9) is that the values of $B(Q)$ increase in approximately equal steps in an exponential scale, roughly doubling for every increase of six in Q. That allows for a closer to linear relationship between PSNR and the step size control parameter Q.

By forcing $B(Q)$ to exactly double for every increase of 6 in Q [12], we can reduce the size of the quantization and reconstruction tables. That also helps to compensate for the need of using three different tables in view of the different row norms of $\mathbf{H}$ (that need comes from the fact that the 2-D version of (6) has three different scaling factors: $1/4^2$, $1/5^2$, and 1/20).

Thus, in our proposal to H.264 [4] we proposed the use of the following quantization formula:

$$X_q(i,j) = \operatorname{sign}\{X(i,j)\}\left[\left(|X(i,j)|\,A(Q_M,i,j) + f2^{17+Q_E}\right) \gg (17+Q_E)\right] \tag{11}$$

where $Q_M \equiv Q$ mod 6 and $Q_E \equiv Q/6$. We see that for every increase of one in the "exponent" Q_E, the denominator in (11) doubles, with no change in the scaling factor multiplying $|X(i,j)|$. This periodicity enables us to define a large range of quantization parameters without increasing the memory requirements. The quantization range has been extended relative to the original H.264 design. The parameter f is chosen by the encoder, and is typically in the range 0 to 1/2. The corresponding reconstruction formula that we proposed [4] is

$$X_r(i,j) = X_q(i,j)\,B(Q_M,i,j) \ll Q_E \tag{12}$$

where we have explicitly indicated the use of a shift operator to replace multiplication by 2^{Q_E}.

We note that the quantization and reconstruction factors $A(Q_M,i,j)$ and $B(Q_M,i,j)$ depend on the transform coefficient position $\{i,j\}$ inside the block. That is necessary to compensate for the different scaling factor in the 2-D version of (6). Their values are given by $A(Q_M,i,j) = M(Q_M,r)$ and $B(Q_M,i,j) = S(Q_M,r)$, where $r = 0$ for $(i,j) = \{(0,0),(0,2),(2,0),(2,2)\}$, $r = 1$ for $(i,j) = \{(1,1),(1,3),(3,1),(3,3)\}$ and $r = 2$ otherwise, with

$$M = \begin{bmatrix} 13\,107 & 5243 & 8066 \\ 11\,916 & 4660 & 7490 \\ 10\,082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix}, \quad S = \begin{bmatrix} 10 & 16 & 13 \\ 11 & 18 & 14 \\ 13 & 20 & 16 \\ 14 & 23 & 18 \\ 16 & 25 & 20 \\ 18 & 29 & 23 \end{bmatrix}. \tag{13}$$

These matrices were designed to maximize dynamic range and to satisfy a similar relationship to that in (10), namely

$$M(Q_M,r)S(Q_M,r)v(r) \cong 2^{21} \tag{14}$$

where $v(r) = \{4^2, 5^2, 4\times 5\}$. The final scaling after reconstruction becomes

$$\mathbf{x}_r = \left(\tilde{\mathbf{H}}_{\text{inv}}\mathbf{X}_r + 2^5\mathbf{e}\right) \gg 6 \tag{15}$$

where $\mathbf{e} = [1\ 1\ 1\ 1]^T$.

Note that in the final draft standard [3] only the reconstruction formulas (12) and (15) are specified, since the standard specifies only the decoder, but not the encoder. Thus, we can look at (11) as a preferred way of performing quantization, given the requirement of reconstruction via formula (12), which corresponds to formula (8-267) in the draft standard [3]. Similarly, (15) corresponds to formula (8-278) in the draft standard [3].

With the transform operators defined in Section IV and the quantization and reconstruction formulas above, we see that all operations can be computed in 16-bit arithmetic, for input data with 9-bit dynamic range. We recall that the inputs to the transform are prediction residuals, and thus they have a 9-bit range for 8-bit pixel data. There is one exception, though: in the quantization (11), the product $|X(i,j)|\,A(Q_M,i,j)$ has a 32-bit dynamic range, but the final quantized value is guaranteed to fall within a 16-bit range. Analysis of the transform dynamic range is provided in [17].

VI. Additional Aspects

The inverse transform and reconstruction specifications in H.264 cover additional aspects. For blocks with mostly flat pixel values, there is significant correlation among transform DC coefficients (i.e., $i = j = 0$) of neighboring blocks. Therefore, DC coefficients can be grouped in blocks of size 4×4 for the luminance channel and blocks of size 2×2 for the luminance channels. This two-level transformation is usually referred to as a hierarchical transform [9]. In the original H.264 design the second-level 4×4 transform was the same as the first-level transform. The final standard specifies just a Hadamard transform (that is, the transform in (2) with $a = b = c = 1$), though, because no performance loss was observed over the standard video test sets 0, and dynamic range and complexity are reduced.

With respect to chrominance coding, usually the same step size as that for luminance is used. However, to avoid visible color quantization artifacts at high quantization step sizes, the current draft limits the maximum value of Q for chrominance to about 80% of the maximum value for luminance; according to the final draft of the specification the maximum value of Q is 51 for luminance, and 39 for chrominance [3].

In some applications, it is desired to reduce the quantization step size to improve PSNR to levels that can be considered visually lossless. To achieve that, the current H.264 draft extends the quantization step sizes by two additional octaves when compared to the original proposal in [2], redefining the tables and allowing Q to vary from 0 to 51. Compared to H.263+, for example, H.264 allows for finer quantization; the minimum $QP = 0$ in H.263+ corresponds approximately to $Q = 6$ in H.264. For small Q, additional care must be taken to avoid exceeding the 16-bit dynamic range. Thus the quantization equation may need to be rescaled, as noted in [3].

During the transform specification, there was an interest in having a matrix-multiply implementation to take advantage of the efficient multiply/accumulate architecture of many processors. Such a definition allows both the efficient shift/add implementation in Fig. 1 and a separable matrix-multiply implementation that produces identical results. In particular, for mathematical agreement for low values of the quantization parameters, the matrix multiply implementation requires inclusion of shift and rounding offsets [4], [18].

VII. Conclusion

We presented the new transform and quantization procedures that have been adopted in the current H.264 draft standard [3]. The new transform is a scaled integer approximation to the DCT, which allows computation of the direct or inverse transform with just additions and a minimal number of shifts, but no multiplications. The basis functions of the new transform do not have equal norm, which leads to an increase in the size of the quantization tables. By using an exact exponential scale for the effective quantization step size as a function of the quantization parameter Q, the table size is reduced to just 36 16-bit entries.

The quantization tables are designed to avoid divisions at the encoder, and to ensure that data can be processed in 16-bit arithmetic. In that way, we achieve a minimal computational complexity, with no penalty in PSNR performance.

References

[1] T. Wiegand and G. Sullivan, "The emerging JVT/H.26L video coding standard," presented at the Tutorial at ICIP 2002, Rochester, NY, Sept. 2002.

[2] G. Bjontegaard. Response to call for proposals for H.26L. presented at ITU-T SG16. [Online]. Available: ftp://standard.pictel.com/video-site

[3] T. Wiegand and G. Sullivan. Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T rec. H.264 | ISO/IEC 14 496-10 AVC. presented at Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts

[4] A. Hallapuro, M. Karczewicz, and H. Malvar, "Low complexity transform and quantization," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Jan. 2002, Docs. JVT-B038 and JVT-B039.

[5] A. Hallapuro and M. Karczewicz, "Low complexity (I)DCT," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Sept. 2001, Docs. VCEV-N43.

[6] H. S. Malvar, "Low-Complexity length-4 transform and quantization with 16-bit arithmetic," in ITU-T SG16, Sept. 2001, Doc. VCEG-N44.

[7] G. Bjontegaard, "Calculation of average PSNR differences between RD curves," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Mar. 2001, Doc. VCEG-M33.

[8] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression*. Boston, MA: Kluwer, 2002.

[9] H. S. Malvar, *Signal Processing With Lapped Transforms*. Boston, MA: Artech House, 1992.

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

[11] W. Cham, "Development of integer cosine transforms by the principle of dyadic symmetry," *Proc. Inst. Elect. Eng.*, pt. 1, vol. 136, pp. 276–282, Aug. 1989.

[12] L. Kerofsky and S. Lei, "Reduced bit-depth quantization," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Sept. 2001, Doc. VCEG-N20.

[13] Y. Katayama, "Protection From IDCT Mismatch," Tech. Rep. MPEG 93/283, ISO/IEC JTC1/SC2/WG11, 1993.

[14] Yagasaki, "Oddification Problem for IDCT Mismatch," Tech. Rep. MPEG 93/283, ISO/IEC JTC1/SC2/WG11, 1993.

[15] L. Kerofsky, "H.26L transform/quantization complexity reduction Ad Hoc Report," in Joint Video Team(JVT) of ISO/IEC MPEG and ITU-T VCEG, Nov. 2001, Doc. VCEG-O09.

[16] G. Beakley, "Requirement for bit depths above 8 bits," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Mar. 2003, Doc. JVT-G044.

[17] L. Kerofsky, "Notes on JVT IDCT," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, May 2002, Doc. JVT-C24.

[18] ——, "Matrix IDCT," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Oct. 2002, Doc. JVT-E033r2.

Henrique S. Malvar (M'79–SM'91–F'97) received the B.S. degree from Universidade de Brasília, Brazil, in 1977, the M.S. degree from the Universidade Federal do Rio de Janeiro, Brazil, in 1979, and the Ph.D. degree from the Massachusetts Institute of Technology (MIT), Cambridge, in 1986, all in electrical engineering.

From 1979 to 1993, he was on the faculty of the Universidade de Brasília. From 1986 to 1987, he was a Visiting Assistant Professor of Electrical Engineering at MIT and a Senior Researcher at PictureTel Corporation, Andover, MA. In 1993, he rejoined PictureTel, where was Vice President of research and advanced development. Since 1997, he has been a Senior Researcher at Microsoft Research, Redmond, WA, where he heads the Communication, Collaboration, and Signal Processing research group. His research interests include multimedia signal compression and enhancement, fast algorithms, multirate filter banks, and wavelet transforms. He has several publications in these areas, including the book *Signal Processing with Lapped Transforms* (Boston, MA: Artech House, 1992).

Dr. Malvar is an Associate Editor for the journal *Applied and Computational Harmonic Analysis*, an Associate Editor of the IEEE Transactions on Signal Processing, and a member of the Signal Processing Theory and Methods Technical Committee of the IEEE Signal Processing (SP) Society. He received the Young Scientist Award from the Marconi International Fellowship and Herman Goldman Foundation in 1981. He also received the Senior Paper Award in Image Processing in 1992 and the Technical Achievement Award in 2002, both from the IEEE SP Society.

Antti Hallapuro was born in Vimpeli, Finland, in 1975. He is currently working toward the M.Sc. degree at Tampere University of Technology, Tampere, Finland.

He joined the Nokia Research Center, Tampere, Finland, in 1998. At Nokia, he works in the Advanced Video Coding Group, where his main concern is real-time implementation of video coding algorithms on various processor architectures. He has participated in the H.264/MPEG-4 AVC video codec standardization effort since 2001 and is author or co-author of several technical contributions submitted to the standardization group.

Marta Karczewicz received the M.S. degree in electrical engineering in 1994 and the Dr. Technol. degree in 1997, both from Tampere University of Technology, Tampere, Finland.

During 1994–1996, she was Researcher in the Signal Processing Laboratory of Tampere University of Technology. Since 1996, she has been with the Visual Communication Laboratory, Nokia Research Center, Tampere, Finland, where she is currently a Senior Research Manager. Her research interests include image compression, communication, and computer graphics.

Louis Kerofsky (S'94–M'95) received the B.S. degree in physics and in mathematics from Arizona State University, Tempe, in 1990, and the M.S. and Ph.D. degrees in mathematics from the University of Illinois, Urbana-Champaign (UIUC), in 1992 and 1995 respectively.

At UIUC, he was a Department of Defense Advanced Research Fellow during 1991–1994 and a Department of Education Teaching Fellow during 1990–1991 and 1994–1995. In 1996, he joined MultiLink, Inc. Andover, MA, as a Software Engineer, where he contributed to the development of a continuous-presence video processing system for an H.320-based multipoint conferencing server. In 1997, PictureTel Corp, Andover, MA acquired MultiLink he worked as a Senior Engineer on digital video processing for server products. In 1999, he joined Sharp Laboratories of America, Camas, WA, in the Digital Video Department. He has represented Sharp in ITU-T standardization activities leading to the standardization of H.264/MPEG-4 AVC where he chaired the *ad hoc* group on Transform and Quantization Complexity Reduction. He is currently a Senior Researcher with interests in video compression, image processing, and fast algorithms. He has one granted and several pending patents.