Proxy-Based Approaches for IDCT Acceleration

Wendi Pan *[a], Antonio Ortega[a], Ibrahim Hajj-Ahmad[b] and Roberto Sannino[b]

[a]Integrated Media Systems Center, Department of Electrical Engineering-Systems
University of Southern California, Los Angeles, CA 90089-2564
[b]Advanced System Technology Lab, ST Microelectronics, San Diego, CA 92127

ABSTRACT

In the boundary between wired and wireless worlds, proxy-based processing has been recognized as an efficient approach to provide client-specific adaptation to improve the overall performance. For the portable video decoding devices with wireless access ability, fast, low-power decoding is desirable. In this paper, we propose a novel proxy-based framework that is capable of accelerating the IDCT operations performed at the client. We introduce a proxy-specific IDCT algorithm that is not only suitable for the proxy framework, but has fine-grained complexity scalability as well. We then demonstrate the effectiveness of the proxy by two examples. The first example is a simple proxy that is assigned the task of IDCT block classification, which is performed at the client along with the IDCT operation conventionally. Experimental results clearly show the complexity advantage of this method. The second example is a more active proxy. Adaptation to the image statistics is introduced. Simulations of the optimization process based on the average complexity criteria show that the client complexity can be further reduced, while maintaining a low side information overhead. Both examples provide the trade-off between the client-proxy bandwidth increment and the IDCT complexity reduction at the client.

Keywords: Proxy, client-server, entropy, side information, complexity/bandwidth trade-off, fast IDCT algorithms, JPEG

1. INTRODUCTION

Today's Internet sees a trend of increased heterogeneity, both in terms of access devices (from high end PCs, hand held PDA's to cell phones) and of access bandwidth (from cable modems, DSL to 28.8k dial-up). Application adaptation can be provided by proxies, which are placed between clients and servers to perform computation and services on behalf of the clients.[1–4] Existing examples of the proxy approach include transcoding techniques developed by such companies as Spyglass and IBM. In wire/wireless multimedia communication, the broad potential of the proxy-based processing has been recognized, which is demonstrated by on-going standardization efforts in the industry, for example, the Internet Content Adaptation Protocol(ICAP),[5] as well as the Wireless Application Protocol (WAP).[6]

On the wire/wireless interface, a considerable amount of work featuring the client-proxy-server model has been done. For instance, functions such as data-type specific lossy compression are performed at the proxy so that latency can be reduced in accessing the web over the CDMA cellular networks.[7] Another example is in mobile video access, where adaptation in joint source and channel coding is applied at the proxy in order to ensure high quality video over error-prone wireless connections.[8] In these applications, the source to be transmitted is either modified or reformatted at the proxy before it is transmitted to the client.

In wireless applications where the power is at a premium for the portable image-decoding clients, it would be desirable for the proxy to be able to communicate with the client and enable fast, low-power decoding based on the client's power needs. Moreover, the proxy can trade-off different resources such as bandwidth, power consumption, latency and memory depending on the client needs or the user preferences.

In this work, we focus on the inverse discrete cosine transform (IDCT), which is used in image/video decoders of various international image and video coding standards such as JPEG, H.263, MPEG-1, MPEG-2, etc.[9] The IDCT is a major component in terms of computation and power consumption. However, since many DCT coefficients are quantized to zero at the encoder of these standards, there exist many methods to exploit the sparsely populated, non-zero transform DCT coefficients to speedup the IDCT computations. Hung and Meng[10,11] identify and count zero values in coefficient matrices in order to select the best IDCT algorithms. Froitzheim and Wolf used reduced

*Correspondence: Phone: (213)740-4679, Fax: (213)740-4651, Email: {dwpan, ortega}@sipi.usc.edu

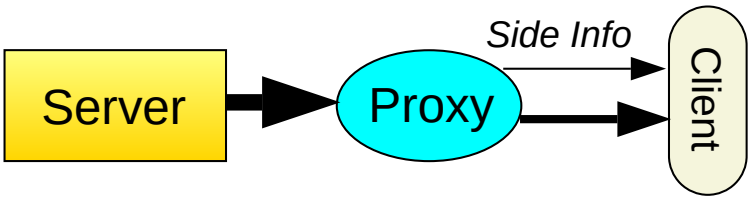

Figure 1. The proxy IDCT framework.

IDCT that saves computation for sparse inputs.[12] Nevertheless, it is not aimed at taking advantage of the full sparseness of the input. We have introduced a tree-structured classification scheme, with the goal of achieving a minimum average complexity after the zero test overheads has been taken into account,[13–15] but this tree structure was not designed for the proxy-client setting. Therefore, it would be desirable to design a fast DCT algorithm that not only has fine granularity in exploiting the sparseness of the input blocks, but also is amenable to the proxy-specific processing, as well as the proxy-client bandwidth constraints.

In this paper, we present a novel proxy-based framework for accelerating IDCT computations: the zero patterns of DCT coefficients are tested and IDCT blocks are classified accordingly. The related class information is then sent from the proxy to the client, where IDCT is actually performed. The client can use this information to accelerate the IDCT computation and reduce power consumption. We assume that the power consumption of the proxy is unconstrained, but the client has a tight power budget. Since the proxy-client bandwidth is constrained, those IDCT algorithms that require large overhead bits for the block class information would not fit well into our framework. On the other hand, we would not be able to harness the full power of the proxy in classifying the blocks, as well as helping the client in reducing the complexity, if the block classification is overly coarse-grained. In order to achieve a good balance, a proxy-specific IDCT algorithm is called for. Based on the dyadic IDCT algorithm,[14] we propose such an IDCT algorithm that is amenable to the proxy framework . We show that through adaptation, an active proxy can further decrease the client's complexity, subject to a slight bandwidth increase constraint.

The remaining of the paper is organized as follows. Section 2 presents the new proxy-based framework that is able to provide the useful trade-off between client-proxy bandwidth and client complexity. Section 3 describes a proxy-specific IDCT algorithm and its complexity. Next, Section 4 discusses a case where the IDCT operation can be made faster if we shift the block classification task from the client to the proxy. Section 5 further demonstrates the effectiveness of the proxy framework. An active proxy can help lower the client's complexity to a greater extent when adaptation to the image statistics is introduced. Finally, conclusion is drawn in Section 6, where our ongoing work is also briefly mentioned.

2. PROXY PROCESSING TO SPEED UP IDCT AT THE CLIENT

In the existing image/video coding standards such as JPEG, H.263, MPEG-1, MPEG-2, etc, the inverse discrete cosine transform (IDCT) has long been known to account for a significant portion of the total decoding time budget. Lowering the complexity of IDCT can contribute to the overall power consumption reduction of the client where image/video decoding is performed. There exist already many fast IDCT algorithms, however, one can do even better by designing *variable complexity* algorithms that take advantage of the fact that many DCT coefficients are quantized to zeros at the encoder, especially when the image quality is low. An additional 15% speed-up is reported to be realizable by exploiting the sparseness of the IDCT blocks.[16] These IDCT algorithms skip computing the zeros in the input blocks, and thus achieve a variable degree of complexity reduction, in accordance with the zero patterns of the IDCT input.

Nevertheless, in such algorithms, the classification of zero patterns in the IDCT input block, if performed at the client, turns out to be a nontrivial overhead. When the zero testing takes an excessive amount of complexity, the return of computing less inputs will be diminished. In a client-proxy setting, however, we may be much better off if the task of zero testing is moved to the proxy.

We thus propose a new framework based on the proxy. In Figure 1, the server stores the image/video data in compressed form. When the data is sent to the client for decoding, it passes through the proxy, which acts as a kind

of intermediary between the server and the client. The proxy can process the data for the purpose of helping the client decode faster. For example, in Sec.4, zero patterns of IDCT inputs are tested and the resulting block class index is obtained by the proxy. These indices are then sent as side information alongside with the image data, to the client where IDCT is actually performed. The client decodes this side information and uses it to invoke the appropriate reduced IDCT subroutines. As a result, the complexity (power consumption) of the IDCT computation is reduced.

We can see that this framework introduces an interesting trade-off between a larger proxy-client bandwidth and the reduced client complexity. This trade-off becomes valuable when client power consumption is more of a concern than a slightly increased bandwidth. On the other hand, we want to design an IDCT algorithm that is more suitable to the distributed processing between and proxy and the client, so that the side information will be kept as low as possible. This issue is addressed in Sec.3, where we discuss a proxy-specific fast IDCT algorithm.

Unlike common proxy schemes, we do not change or reformat the source data, and thus we do not introduce any impairment to the source data. Rather, such structural information as block class index is extracted from the data and sent to the client. Hence, the decode image quality remains unchanged even if the speed of performing the IDCT changes. Note, however, that nothing would prevent the proxy from also changing the coded data while preserving compatibility with the decoder.

Note that here we assume that the power consumption of the proxy is unconstrained, but the client has a tight power budget. Depending on the amount of work it carries out, a proxy can be categorized as *light* (simple) and *heavy* (active). In Sec.5, we show that an active proxy is capable of further reducing the IDCT complexity at the client, by performing a more aggressive processing.

3. PROXY-SPECIFIC FAST IDCT ALGORITHM

In the proxy-based IDCT framework we have described, IDCT inputs are tested and side information about the sparseness is sent from the proxy to the client. Thus we want to select an IDCT architecture that requires low side information overhead, while at the same time helps the client most in reducing the IDCT complexity.

In the case of extremely sparse IDCT inputs, the proxy can test the total number N, of non-zero entries in an 8×8 IDCT block, if N is below a threshold point B, then the client adopts the so called forward mapping IDCT.[17,18] Otherwise, the client falls back to the baseline 8×8 IDCT algorithm. Thus 1 bit/block is needed for the proxy to convey this binary decision to the client. The spatial information of the non-zero entries is not necessary since the client computes the IDCT of each non-zero coefficient independently. The intermediate IDCT results of the non-zero DCT coefficients are simply superimposed at the end. The threshold point B is determined by comparing in terms of complexity the forward mapping IDCT with a fast IDCT algorithm such as AAN[19] on a full 8×8 block. Our study shows that B is very small - around 8. This limits the practical use of this scheme.

There have been many finer-grained approaches in testing the zeros present in the IDCT input blocks.[10–13] However, if these algorithms were incorporated in our proxy framework, the number of different classes would tend to place a heavy burden on the proxy-client bandwidth.

3.1. Dyadic 2D IDCT

The *dyadic* IDCT proposed by Lengwehasatit and Ortega[14] appears to be a good compromise. At the proxy, each 8×8 DCT block is classified into DC-only, low-2×2, low-4×4 and full 8×8 classes . The reason behind this classification is that high frequency components in a typical image are more likely to be quantized to zero, and non-zero coefficients tend to cluster around the DC component.

Only 2 bits/block are required to be transmitted to the client to indicate the class index. At the client, for each of the four possible classes, a corresponding 1D pruned IDCT is applied along the nonzero rows and then all columns.

The advantage of this scheme lies in the inexpensive 2D classification, as well as the low side information (bandwidth) requirement.

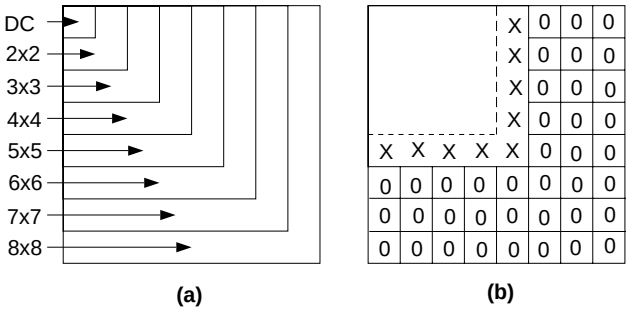

Figure 2. IDCT blocks of different classes. (a) Reduced $k \times k$ IDCT on class k block. DC is a special case, where all entries except the DC coefficient are zero, the output of the 1×1 IDCT is simply the scaled version of the DC coefficient. The other extreme is the full 8×8 IDCT, which is the least sparse case. (b) A block of class 5. Not all entries are zero in the boundary region marked by "X".

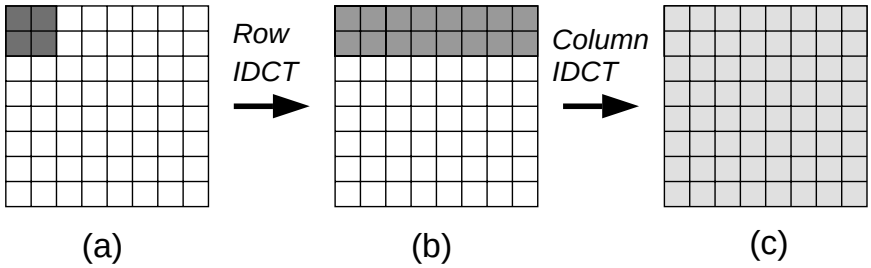

Figure 3. Separable 2D IDCT. Entries in non-shaded regions are all zero. (a)Class-2 block input; (b) A snapshot of the intermediate storage of the block, after the same 2-point 1D reduced IDCT is applied twice, one on each of the upper two rows. (c) After the same 2-point 1D reduced IDCT is applied 8 times, one for each of the column in (b). This completes the whole 2×2 reduced IDCT.

3.2. Proxy-Specific 2D IDCT

One disadvantage of the dyadic 2D IDCT is that it fails to yield satisfactory complexity reduction when a mismatch between the model and the statistics of data occurs. For example, for some sources, blocks of class 5×5 may be more likely to exist than those of class 4×4. But all 5×5 blocks are forced to map to the full 8×8 IDCT, thereby failing to capitalize the feasible complexity gain by using a 5×5 reduced (pruned) IDCT butterfly instead. We extend the dyadic IDCT to a proxy-specific 2D IDCT (PS-IDCT) scheme that is not only suitable for the proxy framework, but also provides a general scalable approach with excellent complexity performance. The details are given below.

For an 8×8 IDCT block $\mathbf{Y}$, where $\mathbf{Y}(i,j)$ is the IDCT input value at (i,j), with $i,j = 1,2,...,8$, we choose one out of 8 possible classes.

Definition A block $\mathbf{Y}$ is said to be of class k if $\mathbf{Y}(m,n) = 0$, $\forall\ m > k$ and $n > k$.

In PS-IDCT, a reduced $k \times k$ IDCT is applied to a block of class k (see Figure 2). A reduced $k \times k$ IDCT on class-k block can be decomposed into two separable operations, either row-wise followed by column-wise, or the other way equivalently. In each operation, the same k-point reduced 1D IDCT is invoked. This is a nice attribute since the penalty for instruction cache misses can be avoided by adhering to the same subroutine in the locality of an IDCT program.[16] Note that although an IDCT algorithm can choose to carry out the row/column operations in different orders, depending on whether there are more zero columns or more zero rows, our PS-IDCT does not distinguish the row/column order since it would incur extra side information overhead from the proxy to the client.

We choose the *AAN* 1D IDCT[19] as the baseline algorithm, which is supported in the independent JPEG software.[20] AAN IDCT is an efficient algorithm, requiring only 5 multiplications and 29 additions. It pre-scales all input DCT coefficients and combines the scaling factors with the de-quantization stage at the decoder. Figure 4.(a) shows the

butterfly of the 8-point AAN IDCT. The k-point IDCT (where $y(m) = 0, \forall m > k$ and $m \leq 8$) can be obtained by pruning and merging some data paths of (a). A 2-point IDCT is illustrated as an example in Figure 4.(b).

As an estimate, assume that the complexity of the k-point reduced 1D IDCT is C_k, then the $k \times k$ IDCT on class-k block has an overall complexity:

$$C_{k \times k} = (k + 8) \times C_k \tag{1}$$

The complexity of the PS-IDCT is tabulated in Table 1, according to Eq.(1). Figure 3 gives the 2×2 IDCT as an example, which applies the 2-point 1D DCT 10 times. We can see that the PS-IDCT achieves finer granularity in complexity than the dyadic IDCT by allowing more IDCT input block classes. Furthermore, the PS-DCT can still maintain low side information: at most 3bits per block is necessary to distinguish all eight possible classes if we use fixed length code for the side information. On the other hand, the bandwidth can also be made scalable, based on the important observation that an $m \times m$ IDCT can be applied safely to a class k block as long as $m \geq k$. There exist many ways the proxy can map a set of eight classes to a smaller set containing, say, four classes, thereby reducing the side information overhead to around 2 bits per block. The operations performed by such an active proxy are presented in Sec.5.

Table 1. Complexity of the IDCTs.

Class k	**k-point IDCT**		**$k \times k$ IDCT**	
	mult	**add**	**mult**	**add**
DC	0	0	0	0
2	3	8	30	80
3	4	12	44	132
4	5	20	60	240
5	5	23	65	299
6	5	25	70	350
7	5	27	75	405
8	5	29	80	464

4. BLOCK CLASSIFICATION AT THE PROXY

One straightforward example to demonstrate the effectiveness our framework is to assign the block classification task to the proxy. In this fashion, the client is relieved of the burden to perform the zero testing, thereby further speeding up the IDCT operations.

4.1. Side Information

For the case of 8 block classes, 3 bits/block are required for coding the class index of each block. Only 2 bits/block are necessary for the 4-class case. Such side information overhead is not significant relatively unless the coded image has very low bit rates.

We would be able to lower the side information by introducing the entropy coding. However, it may be impractical to apply entropy coding directly to the block indices, since our study (see Figure 5) shows that the class index statistics change from image to image and rate to rate, and thus one would need either to have a set of variable-length coding (VLC) tables for different images/rates or use an arithmetic coder to adaptively code the class indices.

One potential alternative is to use differential-coding of the class indices. In our study, the statistics of the index differences of the neighboring blocks show that the index differences tend to take small values (around zero), regardless of the bit rates and images. This would be more amenable to using a fixed VLC table rather than a set of tables.

Figure 6 illustrates the side information required when we perform differential entropy coding of the block indices. The side information is measured by the entropy E as follows.

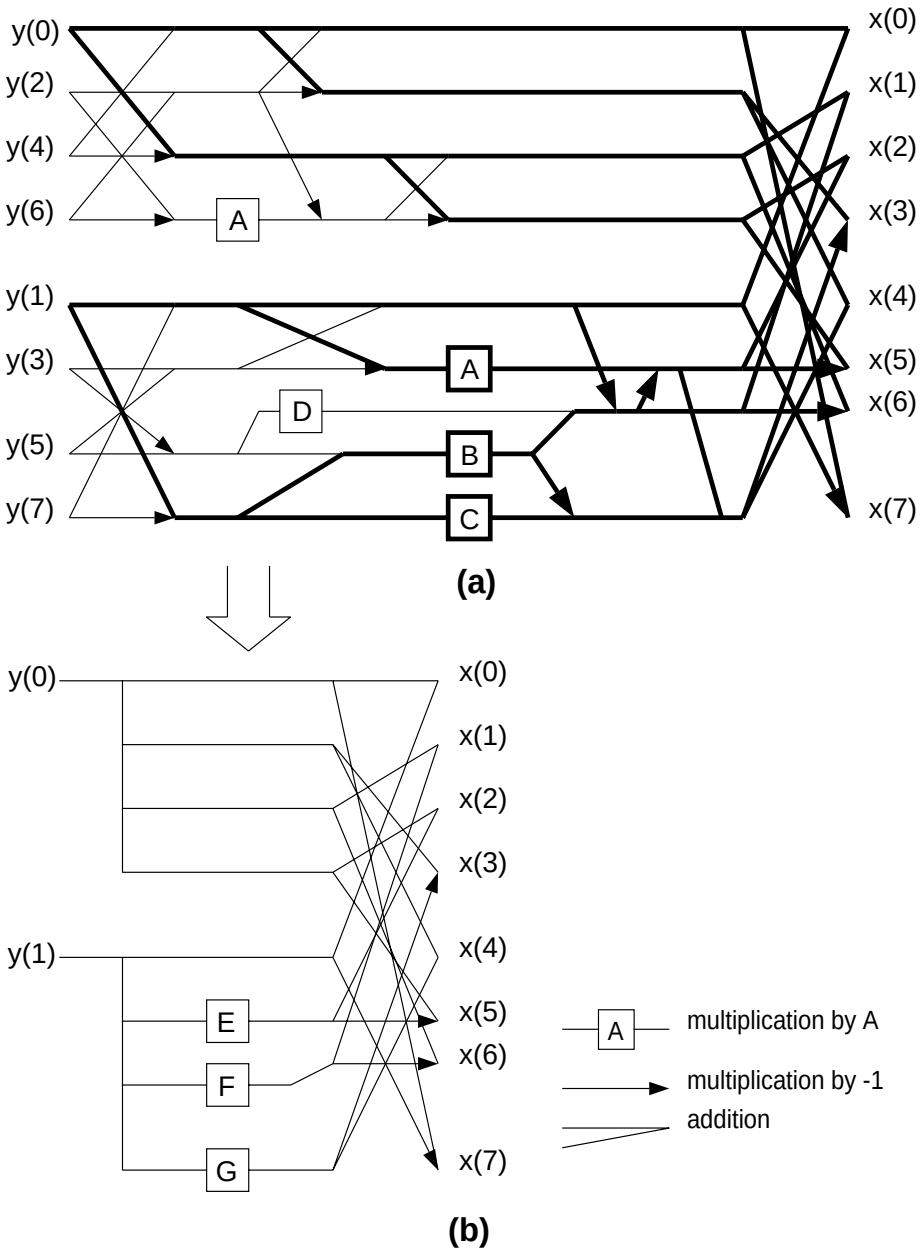

Figure 4. Example of reduction of complexity: pruned AAN IDCT algorithm. (a)Baseline full 8-point IDCT with 5 multiplications and 29 additions. The multiplier values are: $A = 1.414213562$, $B = 1.847759065$, $C = 1.00823922$ and $D = -2.61312593$. If only $y(0)$ and $y(1)$ are non-zeros, then only the highlighted paths are contributing to the outputs $x(0)$ to $x(7)$; (b) A 2-point IDCT is obtained from (a), after we simplify the paths by exploiting the sparseness of the inputs. Only 3 multiplications and 8 additions are required. The three multipliers are given as $E = A - B + 1$, $F = B - 1$ and $G = A - 2B + C + 1$.

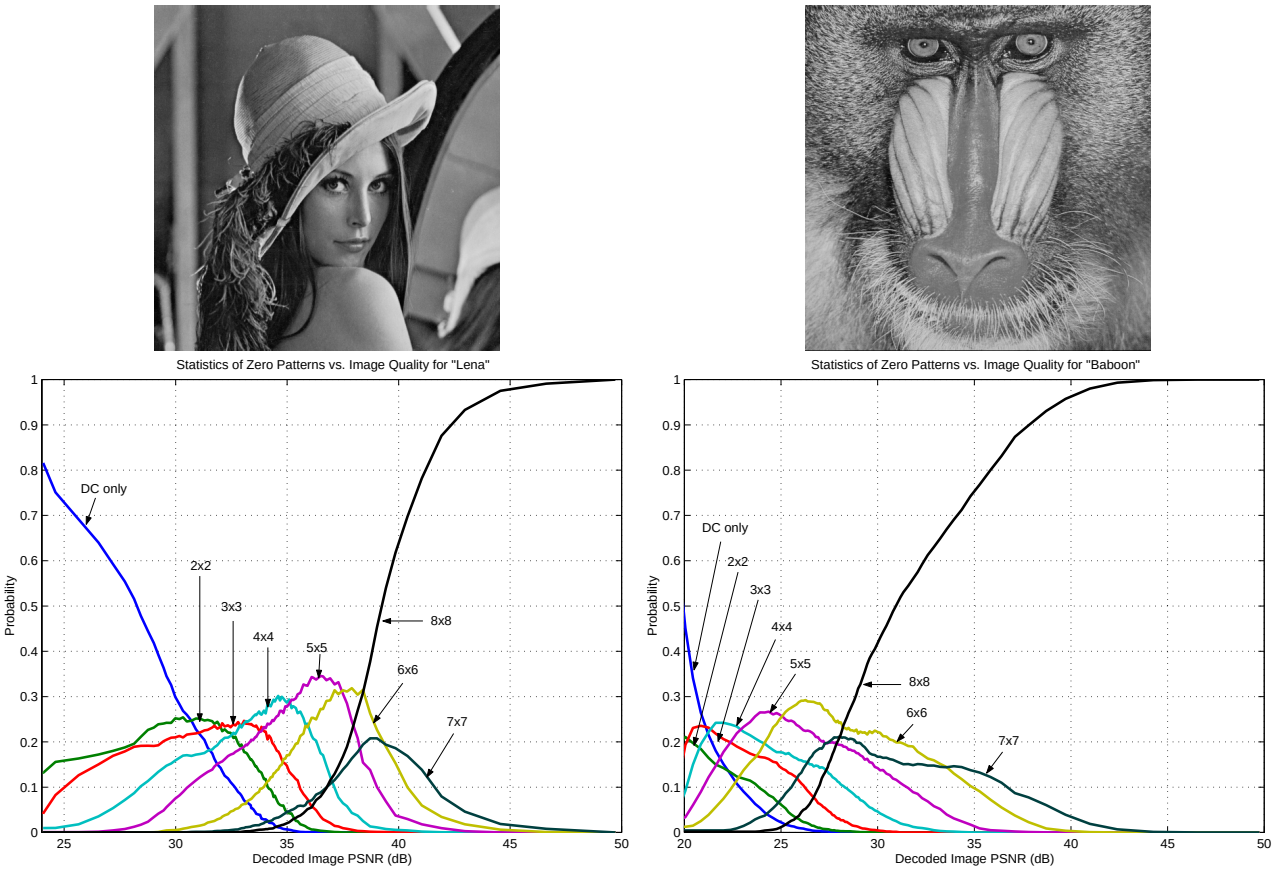

Figure 5. Statistics of classes of IDCT input blocks. Probability of class 1 (DC only), the sparsest class, decreases with increasing image PSNR. The least sparse case, class 8 (full 8×8) is the opposite. In between are class 2 to class 7, which reach their peaks at PSNR values in ascending order. "Baboon" has fewer sparse IDCT blocks than "Lena" due to the fact that "Baboon" has more large high-frequency DCT coefficients.

$$E = \sum_{k=1}^{N} (-P_k \times \log P_k) \tag{2}$$

where P_k is the probability of symbol k occurring in an image, and N is the cardinality of the set of symbols. In PS-IDCT, where 8 classes of blocks are possible, the difference of block indices belongs to the set $[-7, 7]$ of 15 symbols. In the 4-class case such as the dyadic IDCT, we have the set $[-3, 3]$ of 7 symbols.

4.2. Complexity Reduction by the Proxy

The task of block classification essentially involves testing the zeros present in the IDCT inputs. An efficient method to classify blocks is to use their EOB marker value, which can be obtained by parsing the compressed bit stream.[16,17] A simple zero testing scheme is adopted in the IDCT subroutines of the IJPEG software,[20] where the IDCT outputs are set to the DC term when all AC terms are found to be zero. For the purpose of comparison, we adopt the algorithm described below in pseudo code, which starts zero testing from the outside (lower right) to the inside (upper left) of a block $\mathbf{Y}$.

```
k=8
Repeat {
test row Y(i,k) and column Y(k,j) against zero, where i,j = 1,2,...,k
if (at least one nonzero found) break
k = k - 1
```

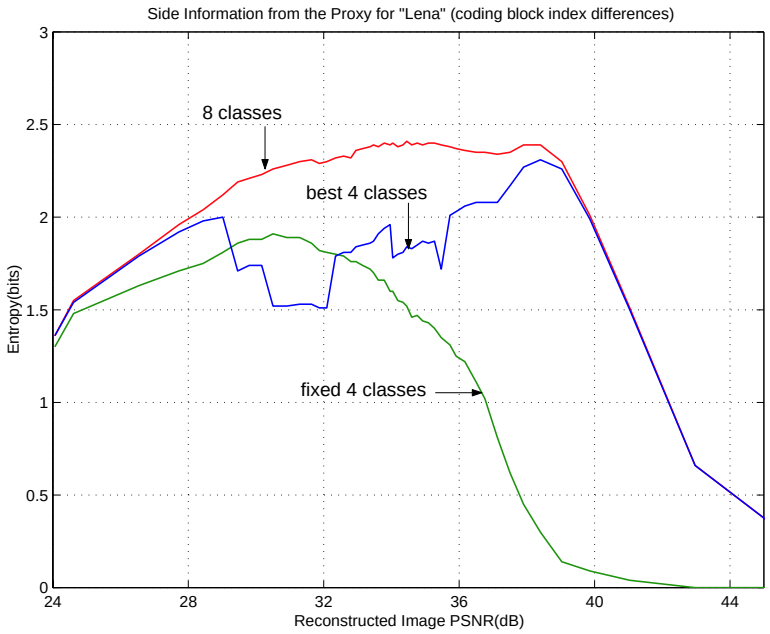

Figure 6. Side information from the proxy for "Lena". Comparison is made of three schemes: the proxy indicates to the client one class out of the set of classes $\{1, 2, 4, 8\}$ as in dyadic IDCT; or out of the set of the best 4 classes found by an active proxy described in Sec.5; or it uses the full classification including 8 classes.

} `Until` $(k == 1)$
`return` k `as the block class`

This zero test algorithm is also a variable complexity algorithm. If block $\mathbf{Y}$ is not so sparse (of larger class index), it terminates earlier. On the other hand, if $\mathbf{Y}$ is very sparse, the algorithm will go all the way towards the upper left corner of the block, hence taking longer to finish.

We compare the complexity of the three schemes: (1) IJPEG IDCT performed at the client; (2) PS-IDCT embedded into the IJPEG software and replacing the original AAN IDCT subroutines. The set of classes is chosen to be $\{1, 2, 4, 8\}$ as in the dyadic IDCT. Block classification is performed at the client and the complexity of these zero tests is included in the overall complexity computation; and finally scheme (3), which is the same as (2) except that zero test is carried out at the proxy. The test is thus not counted towards the complexity of the client. Note that the client needs to spend time in decoding the 2 bits/block side information, either obtained locally in scheme (2) or received from the proxy in scheme (3), so that one out of the four reduced IDCT AAN subroutines can be selected accordingly. The complexity of switching among the four reduced IDCT algorithms is also taken into account in both schemes (2) and (3).

The experimental results of the IDCT complexity at the client are shown in Figure 7. We can observe that the proxy scheme (3) has significantly lower complexity than the client scheme (2), over the whole range of PSNR values. This clearly demonstrates the effectiveness of our proxy framework, which incurs a slight increase of the proxy-client bandwidth, in exchange for an improved client complexity performance.

The client scheme (2) outperforms the IJPEG when PSNR is smaller than 35dB. This is expected since at low PSNR's, many blocks are of small index classes, we can still have an overall gain by invoking reduced IDCT of smaller classes, although more time is spent in the "greedy" zero testing in the client scheme than in the "shallow" zero testing in IJPEG. However, the return is diminishing with increasing PSNR values. The two curves crosses each other at 35dB, implying that the overhead of zero testing in the client scheme more than offsets the gain achieved by using reduced IDCT subroutines of larger class index. Nevertheless, the proxy scheme (3) outperforms the IJPEG consistently. They converge at very high image quality (PSNR=40dB) because both schemes degenerate to the full 8×8 IDCT under such circumstances.

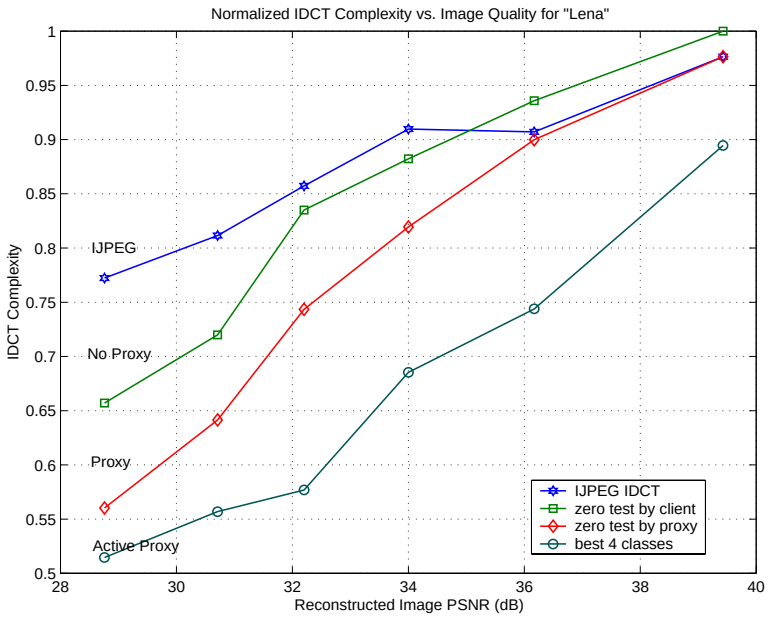

Figure 7. Comparison of the client complexity of the four schemes. The proxy scheme outperforms the IJPEG and the dyadic IDCT (without proxy) scheme consistently. The active proxy scheme achieves further reduction in complexity.

5. ACTIVE PROXY UNDER A BANDWIDTH CONSTRAINT

As mentioned in Sec.4.1, making the set of classes smaller will help reduce the side information between the proxy and the client. In Sec.4, a set of fixed classes is pre-determined and known by both the proxy and the client. In fact, the proxy can play a more active role in adaptively selecting the best four classes out of the eight possible classes, based on the statistics of the image to be decoded at the client. The side information about this selection can be sent to the client in the header field for the entire image. For instance, if the proxy finds that the best set of classes is $\{2, 3, 5, 8\}$, a total of 12 bits (001010100111) is attached in the image header destined to the client, which decodes the field and learns about the decision made by the proxy. The succeeding IDCT operations for the entire image at the client is then tied to this best four classes. We can see that the best-four scheme causes negligible increase in the bandwidth for the whole image, compared with its fixed-four class counterpart.

5.1. Formulation of the Best 4 Classes Problem

The problem can be formalized as follows.

Each element i ($i \in \{1, 2, ..., 8\}$) in the set $\mathcal{A}$ of cardinality 8 is to be mapped to an element j ($j \in \{1, 2, 3, 4\}$) of the set $\mathcal{B}$ of cardinality 4. The one-to-one mapping $i \mapsto j$ is given by $j = \min(n), \forall n \in \{1, 2, 3, 4\}$ that satisfy

$$f(n-1) < i \leq f(n) \tag{3}$$

where,

$$k = f(n) \tag{4}$$

i.e., the set $\mathcal{B}$ is so chosen that $k \times k$ reduced IDCT is applied to class n in set $\mathcal{B}$. Intuitively speaking, out of the eight classes of $\{1, 2, ..., 8\}$, the mapping is such that those that belong to set $\mathcal{A}$ are mapped to the nearest classes in set $\mathcal{B}$. See Figure 8 for two possible mapping schemes.

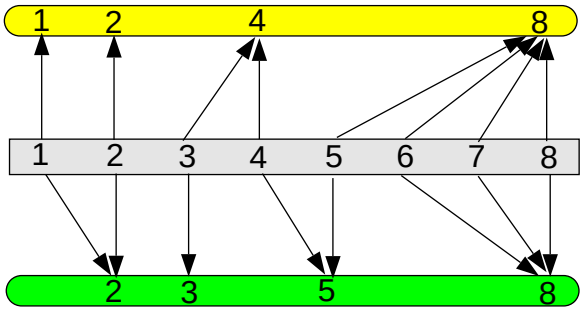

Figure 8. Two possible mapping schemes from a set of eight elements to a smaller set of four elements. In the mapping upwards, blocks of classes from 5 to 8 are merged and mapped to the fourth element in the smaller set on which the full 8×8 IDCT is applied, whereas in an alternative mapping downwards, blocks of class 5 are mapped to the third element that corresponds exactly to a reduced 5×5 IDCT.

There are altogether $\binom{8}{4} = 70$ possible smaller sets the proxy can map the larger set to. The proxy can perform an exhaustive search and choose the best set that minimizes the following average complexity C_{avg},

$$C_{avg} = \sum_{j=1}^{4} C_{k \times k} \times P_k, \tag{5}$$

where $k = f(j)$, as defined in Eq.(4). And P_k is the probability given by $P_k = \frac{N_j}{N_t}$, where N_j is the number of blocks mapped to j, and N_t is the total number of blocks in the image. For a 512×512 image with block size being 8×8, N_t is 4096. $C_{k \times k}$ can be approximated by Eq.(1). In our experiment, the number of operations listed in Table 1 are used. This information can be readily made available to the proxy prior to the transmission of source content. we can also measure the real running time (or power consumption) of each reduced IDCT subroutines at the client, so that the proxy can perform the optimization procedure based on more accurate estimations.

5.2. The Greedy Method

In order to reduce the complexity of exhaustively searching all the possible reduced sets in the optimization process, we propose a more efficient greedy algorithm that consists of four phases. The algorithm starts with a set of 8 classes $\{1, 2, ...8\}$. Each class i has its associated IDCT complexity $C_{i \times i}$, and the number of blocks falling into this class (occupation) N_i. In each phase, we examine any two neighboring classes i and j. Without loss of generality, we assume $j > i$. Note that $j = i + 1$ may not hold in other phases than the phase 1. We define the cost of merging the class pair (i, j) into one class j as ΔC_{ij}, which can be expressed as the increased complexity caused by the merging:

$$\Delta C_{ij} = (N_i + N_j) \times C_{j \times j} - (N_i \times C_{i \times i} + N_j \times C_{j \times j}) = N_i \times (C_{j \times j} - C_{i \times i}) \tag{6}$$

Among all possible merging operations in a phase, we choose the pair of neighboring classes (i, j) found to have the minimal cost. As a result, class i is removed from the set, while class j is updated to a new class by increasing its occupancy to $(N_i + N_j)$. The class complexity $C_{j \times j}$ remains unchanged. It follows that the next phase will operate on a smaller set that has one fewer class than the current phase. After four phases of merging the minimal cost pairs, there are only four classes left in the set. Our simulation results suggest that they are identical to the best four classes found by the exhaustive search method.

As to the complexity, in phase i, where $i = \{1, 2, 3, 4\}$, at most $(8-i)$ subtractions, multiplications and comparisons are required. The subtraction operation can be replaced by the table lookup if we pre-compute all the class complexity difference $(C_{j \times j} - C_{i \times i)}$. Hence the total complexity of the greedy algorithm can be reduced to $22 (= 7 + 6 + 5 + 4)$ multiplications and 18 comparisons. By contrast, the forgoing exhaustive search method has to compare the cost of $\binom{8}{4} = 70$ possible smaller sets, with each set involving 4 multiplications. Thus a total of up to 280 multiplications and 69 comparisons are required. The advantage of the greedy method is self evident. Moreover, the existence

of a greedy algorithm makes it conceivable to perform this class selection among more than 8 classes, without the significant increase in computation required by an exhaustive search.

5.3. Results

Table 2 gives the best 4 classes generated by the active proxy algorithm. Two observations can be made: (1) the classes shift to the right (larger index) as PSNR value increases. This is due to blocks becoming less and less sparse as the quantization gets finer; (2) When the image quality is as low as below 30dB, the blocks are so sparse that even the class-7 and class-8 blocks are absent. Hence the proxy can instruct the client to apply 6×6 IDCT to class-5 blocks rather than the wasteful full 8×8 IDCT, as dictated by the fixed-four scheme of $\{1, 2, 4, 8\}$.

Table 2. Best 4 classes for "Lena" under several decoded image PSNR values.

PSNR(dB)	Classes							
28.76	1	2	3			6		
30.71	1		3		5		7	
32.20		2	3		5			8
34.68			3	4		6		8
36.17				4	5	6		8
37.94					5	6	7	8
39.43					5	6	7	8

Simulation results agree well with our theoretical prediction. The complexity of the best-four class scheme is remarkably lower than the fixed-four class scheme, regardless of the reconstructed image quality (Figure 7). This clearly demonstrates the client performance improvement made possible by the adaptation at an active proxy.

Both the static and dynamic schemes require 2 bits/block overhead. If differential entropy coding of block indices are used, the static scheme consumes less side information bits than the dynamic one at most cases (Figure 6). This, again, exhibits the interesting trade-off between complexity reduction and side information increase.

This optimization process requires a global view of the statistics of the whole image, hence the processing delay would be unacceptable if it is performed by the client device. Besides, even the less complex greedy approach would become an excessive overhead for thin client devices that typically are not accessible to powerful computational resources. Therefore, the optimization only becomes practical when the proxy comes into play.

6. CONCLUSIONS AND ONGOING WORK

We have proposed a novel proxy framework in wire/wireless multimedia communication, which is capable of help accelerating the IDCT operations performed at the client. We introduce a fast IDCT algorithm that is conducive to the client-proxy-server architecture. We demonstrate that the effectiveness of the proxy in providing the useful trade-off between client-proxy bandwidth and IDCT complexity at the client. We also show that through adaptation, an active proxy can further improve the client complexity performance to a great extent, without violating the bandwidth constraint.

Currently, we are investigating ways in which a more active proxy can selectively set some high-frequency DCT coefficients to zero so that we can save overhead bits for the block class information, and at the same time, we can speed up the IDCT computation. As expected, the received image suffers from quality degradation. our goal is to optimize the decoded image quality subject to either the client complexity, or the proxy-client bandwidth constraints.

ACKNOWLEDGMENTS

Most of this work was performed while the first author was a summer research student at AST San Diego lab of STMicroelectronics. This research has also been funded in part by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152.

REFERENCES

1. R. Mohan, J. Smith, and C.-S. Li, "Adapting multimedia internet content for universal access," *IEEE Transactions on Multimedia* , pp. 104–114, March 1999.
2. A. Fox, S. Gribble, Y. Chawathe, and E. Brewer, "Adapting to network and client variation using infrastructural proxies: lessons and perspectives," *IEEE Personal Communications* , pp. 10–19, 1998.
3. R. Han *et al.*, "Dynamic adaptation in an image transcoding proxy for mobile web browsing," *IEEE Personal Communications* , pp. 8–17, Dec. 1998.
4. I. Bharadvaj, A. Joshi, and S. Auephanwiriyakul, "An active transcoding proxy to support mobile web access," in *17th IEEE Symposium on Reliable Distributed Systems (SRDS'98)*, (West Lafayette, Indiana), Oct. 1998.
5. The ICAP forum, http://www.i-cap.org/index.cfm.
6. The WAP forum, http://www.wapforum.org/.
7. K. Ham, S. Jung, S. Yang, K. Lee, and K. Chung, "Wireless-adaptation of WWW content over CDMA," in *IEEE International Workshop on Mobile Multimedia Communications (MoMuC'99)*, pp. 368–372, 1998.
8. J. Vass, S. Zhuang, J. Yao, and X. Zhuang, "Efficient mobile video access in wireless environments," in *IEEE Wireless Communications and Networking Conference (WCNC'99)*, pp. 354–358, 1999.
9. W. Pennebaker and J. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, 1994.
10. A. Hung and T.-Y. Meng, "Statistical inverse discrete cosine transform for image compression," *Proc. of SPIE* **2187**, pp. 196–207, 1994.
11. A. Hung and T.-Y. Meng, "A comparison of a fast inverse discrete cosine transform algorithms," *ACM & Springer International Journal on Multimedia Systems* **2**, pp. 204–217, 1994.
12. K. Froitzheim and H. Wolf, "Knowledge-based approach to JPEG acceleration," in *Proc. of IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, (San Jose), Feb. 1995.
13. K. Lengwehasatit and A. Ortega, "Distortion/decoding time tradeoffs in software DCT-based image coding," in *Proc. of ICASSP'97*, (Munich, Germany), 1997.
14. K. Lengwehasatit, *Complexity-distortion tradeoffs in image and video compression.* PhD thesis, University of Southern California, April 2000.
15. K. Lengwehasatit and A. Ortega, "Rate-complexity-distortion optimization for quadtree-based DCT coding," in *Prof. of Intl' Conf. on Image Processing (ICIP'00)*, Sep. 2000.
16. L. Winger, "Source adaptive software 2D IDCT with SIMD," in *Proc. of IEEE ICASSP'2000*, (Istanbul, Turkey), June 2000.
17. E. Murata, M. Ikekawa, and I. Kuroda, "Fast 2D IDCT implementation with multimedia instructions for a software MPEG2 decoder," in *Proc. of IEEE ICASSP'98*, May 1998.
18. L. McMillan and L. Westover, "A forward-mapping realization of the inverse discrete cosine transform," in *Data Compression Conference (DCC'92)*, pp. 219–228, 1992.
19. Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. IEICE* **71**, pp. 1095–1097, 1988.
20. The independent JPEG's group, JPEG software version 6b, http://www.ijg.org/.