DISTORTION/DECODING TIME TRADEOFFS IN SOFTWARE DCT-BASED IMAGE CODING

Krisda Lengwehasatit *Antonio Ortega*

Integrated Media Systems Center
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, California 90089-2564
Phone: (213) 740-4679, Fax: (213) 740-4679
E-mail: lengweha@sipi.usc.edu and ortega@sipi.usc.edu

ABSTRACT

We present a general framework for variable complexity algorithms (VCA) and study the related issue of defining a minimum *average* complexity implementation. As an example we consider implementations of the inverse DCT (IDCT) which minimize the average computation time by taking advantage of the sparseness of the quantized input data. Since the decoding speed depends on the number of zeros in the input we then present a formulation that enables the encoder to optimize its quantizer selection so as to meet a prescribed "decoding time budget". This leads to a complexity-distortion optimization technique which is analogous to well known techniques for rate-distortion optimization. In our experiments we demonstrate significant reductions in decoding time.

1. INTRODUCTION

The Discrete Cosine Transform (DCT) is by far the most popular transform used for image compression applications [1, 2]. Reasons for its popularity include not only its good performance in terms of energy compaction for typical images but also the availability of several fast algorithms [3]. While we concentrate on the DCT, most of our developments are directly applicable to other orthogonal transforms. One common factor in all the fast algorithms proposed for hardware implementation of DCT, see for example [4], is that they aim at reducing the complexity of a generic direct or inverse DCT, *regardless of the input to be transformed*. Therefore complexity is estimated by the number of operations which is the same for every input. However it is easy to verify that for typical scenarios the image blocks on which IDCT has to be performed are very sparse.

A question naturally arises in evaluating a compression algorithm, namely, whether the performance achieved (e.g. in terms of distortion for a given rate) justifies the required complexity. Initial steps in evaluating these complexity, rate and distortion trade-offs can be found in [5], where block transforms of various block sizes are compared. A more complete theoretical framework is described in [6] where a distortion-computation function is defined and examples are given for block transforms of different sizes. In both cases the complexity estimates were not input dependent and the study concentrated on the encoder complexity.

In this work we focus on the decoder complexity and study the achievable performance *for a given input* to the decoder[1]. As will be seen, in order to minimize the average complexity one may need to resort to *variable complexity algorithms (VCA)*, i.e. algorithms which require a variable, input-dependent, amount of time to complete a task. The analogy with variable length coding is quite obvious. As in the coding case, a VCA matched to the input data will result in reduced average complexity as compared to a fixed complexity implementation. The only requirement to achieve this gain is then the existence of classes of inputs for which the operation at hand can be computed faster. In this work we consider an example of such a scenario, namely, that of the computation of the inverse DCT (IDCT) transform, and show how a VCA can be designed, similar to those used in public domain software implementations of image/video codecs (e.g. [7, 8, 9]). Note that the sparseness of the quantized blocks in the DCT domain has been used to speed compressed domain manipulation [10], however the statistical input dependences have been left unexplored.

Our motivation comes from observing the increased importance of software implementations for various compression applications (e.g., software-only decoders for H.263 video, JPEG coders/decoders images used for Internet applications). This trend is likely to continue as faster hardware becomes available and innovative uses of software, for example usage of JAVA applets, become widespread. In these circumstances optimizing the performance of the algorithms for the specific case of software operation is all the more important. Another example of an application of our framework is the design of Asynchronous circuits which allow operations to be completed in a variable time and where the decision on when to complete an operation should be driven by the type of input [11].

The paper is organized as follows. Section 2. introduces a formal framework for VCAs. Section 3. shows an example of optimization of average complexity in a VCA implementation of the IDCT. In Section 4. we introduce the idea of optimizing the distortion of the source for a given total complexity at the decoder. Experimental results are provided to demonstrate the performance.

2. VARIABLE COMPLEXITY ALGORITHMS

Assume a generic algorithm which takes an input from a set $\mathcal{I}$ and maps it into an output from a set $\mathcal{O}$. In the case of a size 8 IDCT $\mathcal{I} = \mathcal{O} = \mathcal{R}^8$. In order to have a VCA it is necessary to define different versions of the algorithm which are matched to the various inputs. For example, in the IDCT case one can classify inputs depending on how many zeros they contain so that simpler versions of the algorithm are available for those inputs with more zeros.

For simplicity assume a discrete set of inputs $\{1, 2, \ldots N\}$ each with a corresponding probability of occurrence $p(i)$. Assume that the set of available implementations of our algorithm is also finite $\mathcal{A} = \{A_0, A_1, \ldots, A_k\}$ and each algorithm has a fixed complexity $c(A_j)$. Note that not all the implementations may be suitable for all inputs, i.e. some implementations may not yield exact results[2]. For example, an IDCT algorithm which assumes that the first 4 co-

[1] i.e. given a statistical characterization of the input

[2] The exactitude of the algorithm could be made part of the trade-offs by considering it to be an additional source of

© IEEE 1997

efficients in a vector are zero is not suitable for inputs not having this property. Let $\mathcal{A}_i \subset \mathcal{A}$ be the set of algorithms providing an exact result for input i. We will assume that A_0 is an algorithm which gives an exact result for all possible inputs. Our goal is thus to outperform this baseline algorithm approach which will have a fixed complexity of $c(A_0)$ for every input. Note that this may not always be possible depending on the characteristics of the input.

Given $c(A_j)$ and $p(i)$, call $\bar{T}$ the average computation cost for an implementation achieving exact results. Before we deal with the IDCT case we concentrate on determining bounds on $\bar{T}$. As will be seen, the key lies in the classification of the input. Assume that we exactly know what input, out of the N possible ones, has to be processed, and this without any classification computation cost. Then, clearly, for a given input i we should use

$$A_i^\star = \arg \min_{A_k \in \mathcal{A}_i} c(A_k), \quad (1)$$

that is, the minimum cost algorithm among those giving an exact result for input i. Thus we can write that

$$\bar{T} \;\geq\; \sum_{i=1}^{N} p(i) \cdot c(A_i^\star). \quad (2)$$

This will be unlikely to be a tight bound since classification cost was not considered.

To obtain a more realistic cost estimate one needs to take into account the input classification. Indeed, the problem of finding the optimal average complexity is really a classification problem. Assume that we partition the input set $\mathcal{I}$ into L classes, $\mathcal{C} = \{C_1, C_2, \ldots, C_L\}$, s.t. as usual, $\cup_{l=1}^{L} C_l = \mathcal{I}$ and $C_i \cap C_j = \emptyset$, $\forall i \neq j$. Let s_l be the cost of classifying an input in class l. We assume that it costs the same to classify any input for a given class. For each set C_l in the partition we will use a single algorithm and we impose the constraint that it is an exact algorithm for all the elements of the set. Thus we define the allowable algorithms for set C_l so that $\mathcal{A}'_l = \cap_{j \in C_l} \mathcal{A}_j$, which will at least contain A_0. Thus, as before, each class should use the fastest available algorithm

$$A_l'^\star = \arg \min_{A'_k \in \mathcal{A}'_l} c(A'_k), \quad (3)$$

and the overall cost will be

$$\bar{T}(\mathcal{C}) = \sum_{i=l}^{L} \Big(\sum_{i \in C_l} p(i)\Big) \cdot (c(A_l'^\star) + s_l), \quad (4)$$

where clearly the expected average cost depends on the classification. Thus finding the optimal cost for given $\mathcal{I}$ and $\mathcal{A}$ requires finding the classification $\mathcal{C}^\star$ which minimizes $\bar{T}(\mathcal{C})$. In general it may not be possible to find general bounds, we now consider the particular case of classification for the IDCT where optimal results can be achieved.

3. IDCT OPTIMIZATION

To the best of our knowledge, only one previous published work, by Froitzheim and Wolf [12] (FW), has formally addressed the problem of minimizing the IDCT complexity in an input dependent manner, based on detecting zero coefficients in the input block. Numerous software implementations of IDCT available in the public domain do take into account the sparseness to achieve image decoding speed-up [7, 8, 9]. However, these methods seem to be based on trial

distortion.

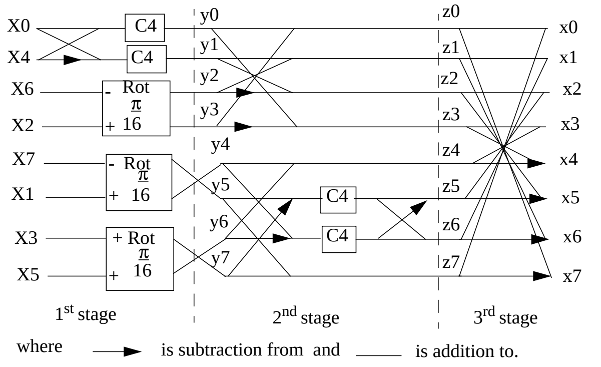


Figure 1. Inverse DCT algorithm by Vetterli-Ligtenberg where Rot(Q) is a rotation operation mapping $[x, y]$ to $[X, Y]$ such that $X = x \cos Q + y \sin Q$ and $Y = -x \sin Q + y \cos Q$.

and error for typical images. In [13] we presented a systematic procedure to obtain the fastest, in the average sense, IDCT algorithm for a given image or set of images. A brief description of the algorithm follows; additional details can be found in [13].

As discussed above, input classification is key in VCA design. Out of the many different classification methods, we define an input class to be a set of vector inputs having zero coefficients at certain positions. For example all the DCT inputs where all the coefficients are zero belong to a class and all the inputs which are zero in the first 4 components and non-zero elsewhere belong to a different class. Thus, for a size 8 DCT we have 256 different classes of inputs. To each class i, where $i \in \{0, \ldots 2^b - 1\}$ is the integer value corresponding to the position of the zeros for a vector of size b, we can associate an empirical probability p_i based on typical compressed images. Clearly, inputs with more zeros result in potentially faster IDCT implementations as operations involving zeros can be skipped. For example an all zero input (i.e. Class 0) would require no operations while a Class 255 input would require the maximum number of operations. We first select the Chen-Wang (CW) algorithm [14][3] as our baseline algorithm. For each input class one can find an exact IDCT implementation with minimum number of operations which will be referred to as the *reduced IDCT* version for a given class.

The baseline IDCT (see Fig. 1) structure involves several stages at which groups of 2, 4 and 8 coefficients are the input to a basic operation. We thus select a tree-structured classification which successively tests to determine whether sets of 8, 4 and 2 input coefficients are zero (see Fig 2 for a size 4 IDCT example). Let $S_{x_0 x_2 x_1 x_3}$ represent a class of inputs where subscript x_j provides the available information for the j-th coefficient in the IDCT and can take the following values: (i) $n \in \{1, 2, \ldots\}$ when at least one of the coefficients indexed with this number is nonzero, (ii) 0 when x_j is known to be zero and (iii) d when x_j can be either zero or nonzero (don't care). As an example of S_{11d0} means that at least one of $\{x_0, x_2\}$ must be nonzero, x_1 is a "don't care", i.e. we have no information about it, and x_3 is known to be zero. The classification tree in Fig 2 operates in a hierarchical (testing successively on 4, 2 and 1 coefficients) but does not indicate in which order the tests should be performed. The goal is then to find the best classification, that is, to determine (i) the best order in which to apply the tests and (ii) which tests it is worth performing (in the sense of reducing the average complexity) given that testing for zeros in the input carries an associated cost.

[3]Both the reduced number of operations and the fixed point arithmetic implementation motivate this choice.

We now present experimental results obtained with the optimized tree-pruning algorithm proposed in [13].

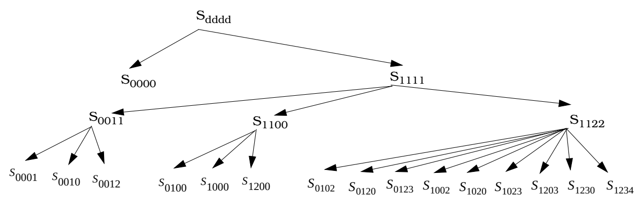


Figure 2. Classification diagram for a given input into 16 possible classes.

In our experiments we implement a simple software image decoder based on the tree-structured classication scheme described above. For the sake of praticality, we use a set of approximated weights for the different operations involved (additions, multiplications, logical operations, etc) in our optimization. Fig. 3 indicates that the experimental results in (b) are sufficiently close to those predicted by the model (a)[4]. In Fig. 3 all the values are normalized by the complexity of the baseline CW algorithm [14] without any zero tests. Note that significant reductions in complexity are possible. Other than our optimized algorithm and FW [12] we give results for the CW algorithm with all-zero test, and CW with all-zero test for the first 1-D IDCT and ac-zero test for the second 1-D IDCT (since after the first 1-D IDCT, it is more likely for typical images that only the DC coefficient in the 1D vector is non-zero). Results are given for various MSE values. Note that our algorithm is robust even used at an MSE different from what it was designed for (see Fig. 3(b)).

4. OPTIMIZING ENCODING FOR A GIVEN DECODING TIME

As described in the previous section a VCA implementation of the IDCT can be obtained given a "typical" set of input images. As is clear from the results, the coarser the quantization the faster the decoder can run. Thus, a decoder based on a VCA IDCT (or several different IDCT algorithms) is inherently *computation scalable.* The encoder can control the decoding speed by assigning to each block one out of several available quantizers (coarser quantizers result in faster operation).

The question arises then of how to optimally select those quantization steps *for a given decoding time budget.* This leads to a formulation where the encoder operates based on a complexity-distortion (C-D) trade-off, rather than on the traditional rate-distortion (R-D) trade-off. As an application, one could for instance store images to be downloaded by different types of hosts so that the slower hosts can access lower quality images but which can be decoded more quickly. Similarly, the quality of the decoded images can be selected based on the load of the shared host. The problem can be formalized as finding the quantizer assignment $\{j,i\}_{opt}$ such that

$$\sum_j D_j(i) \text{ is minimized while } \sum_j T_j(i)) < T_{budget}.$$

Where T_{budget} is the total time (or complexity) budget, $D_j(i)$ and $T_j(i)$ are, respectively, the distortion and decoding time when quantizer i is used for block j. The problem can be solved using the well-known Lagrangian multiplier method [15]

$$\{j,i\}_{opt} = arg \min_{\{j,i\}} (\sum_j D_j(i) + \lambda \cdot \sum_j T_j(i))$$

where $\lambda \geq 0$ is the Lagrange multiplier, which will have to be adjusted selected so that the budget is met.

Figs. 4 and 5 summarize our results. Fig. 4 indicates that appropriate quantizer allocation for each block results in very significant reductions in complexity. As expected the C-D allocation outperforms the other methods. Note that in Fig. 4 we compare the C-D allocation when multiple algorithms are used (one for each quantizer, 'x') and when a single algorithm is used for all quantizers ('+' and '*'). When a coarse quantizer is used in the design ('*') the performance is very close to that of the multiple algorithm approach. For the actual decoding tests we thus use a single algorithm. Each point in the C-D optimized curves is obtained with a given parameter λ. Figs. 5 (a) and (b) demonstrate how a complexity driven quantizer allocation ('*') differs from its rate driven counterpart ('x'). As expected better R-D performance is expected if a rate budget, rather than a complexity budget, is used.

[4] Note that in Fig. 3(b) we depict the normalized *total* decoding time and thus reduction in complexity is not as significant in relative terms as in Fig. 3(a).

REFERENCES

[1] W. Pennebaker and J. Mitchell, *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, 1994.

[2] J. Mitchell, W. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard.* New York: Chapman and Hall, 1997.

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

[4] A. Ligtenberg and M. Vetterli, "A discrete Fourier/cosine transform chip," *IEEE J. on Sel. Areas in Comm.*, vol. SAC-4, pp. 49–61, Jan 1986.

[5] M. J. Gormish, *Source Coding with Channel, Distortion and Complexity Constraints.* PhD thesis, Stanford University, Mar. 1994.

[6] V. Goyal and M. Vetterli, "Computation distortion characteristics of block transform coding," in *Proc. of ICASSP'97*, (Munich, Germany), Apr. 1997.

[7] "The independent JPEG's group software JPEG, version 6." `ftp://ftp.uu.net`.

[8] "MPEG video software decoder, v2.2." `http://bmrc.berkeley.edu/projects/mpeg/`.

[9] "vic:UCB/LBL video conferencing tool." `ftp://ftp.ee.lbl.gov/conferencing/vic/`.

[10] N. Merhav and V. Bhaskaran, "A transform domain approach to spatial domain image scaling," in *Proc. of ICASSP'96*, (Atlanta, GA), pp. 2405–2409, May 1996.

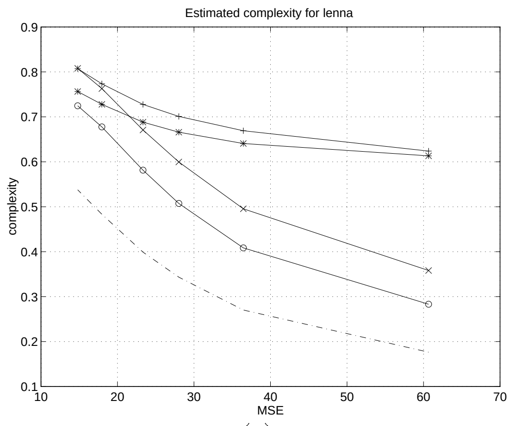
[11] S. M. Nowick, K. Y. Yun, P. A. Beerel, and A. E. Dooply., "Speculative completion detection for the design of high-performance asynchronous dynamic adders," in *Proc. of Intl. Symp. on Advanced Res. in Async. Circuits and Systems (ASYNC)*, 1997.

[12] K. Froitzheim and H. Wolf, "A knowledge-based approach to JPEG acceleration," in *Proc. of IS&T/SPIE*, (San Jose), Feb. 1995.

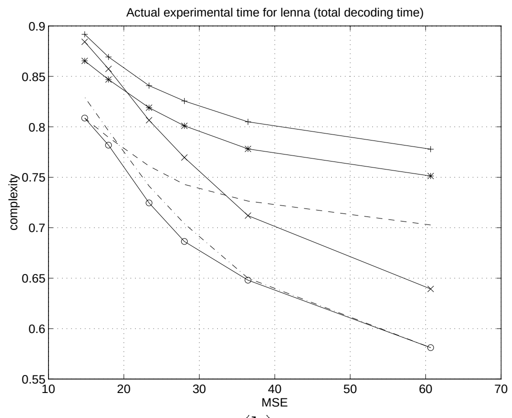
[13] K. Lengwehasatit and A. Ortega, "DCT computation with minimal average number of operations," in *Proc. of VCIP'97*, (San Jose, CA), Feb. 1997. To appear.

[14] Z. Wang, "Fast algorithms for the discrete w transform and for the discrete fourier transform," *IEEE Trans. on Signal Proc.*, vol. ASSP-32, pp. 803–816, Aug. 1984.

[15] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. on Signal Proc.*, vol. 36, pp. 1445–1453, Sept. 1988.

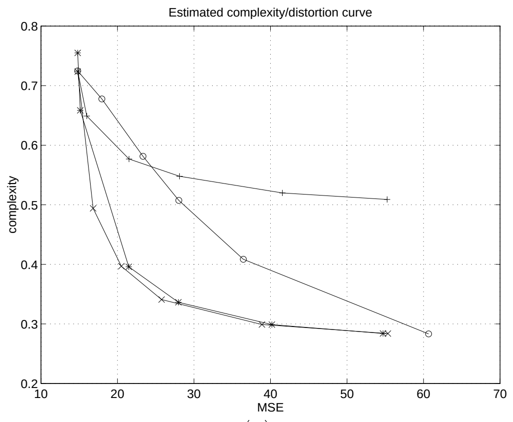


(a)

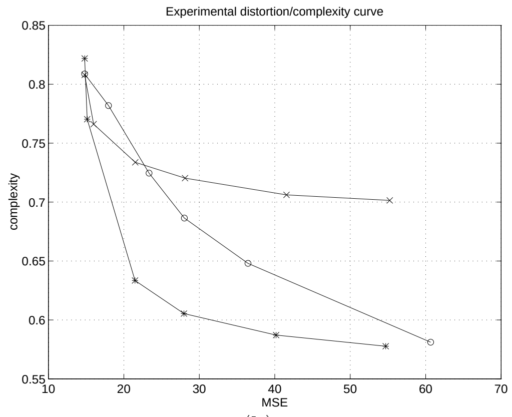


(b)

Figure 3. Normalized results for lenna (CW=1): (a) estimated IDCT complexity, (b) experimental decoding complexity. In (a) we compare, CW with all-zero test ('+'), CW with all-zero test for the first 1-D IDCT and ac-zero test for the second 1-D DCT ('x'), FW ('*'), tree-structured classification ('o'), and algorithm with no classification cost ('-·'). For (b) we also have results for optimized trees designed with different inputs witj MSE=14.8 ('--') and MSE=60.66 ('-·').

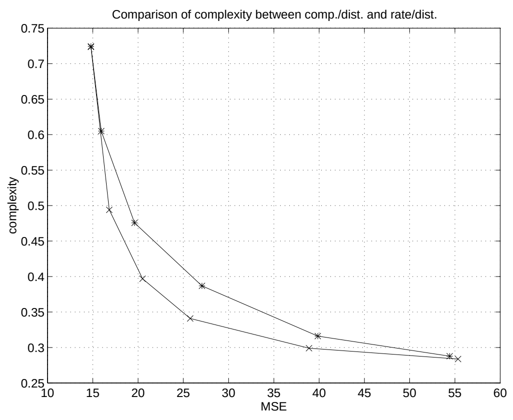


(a)

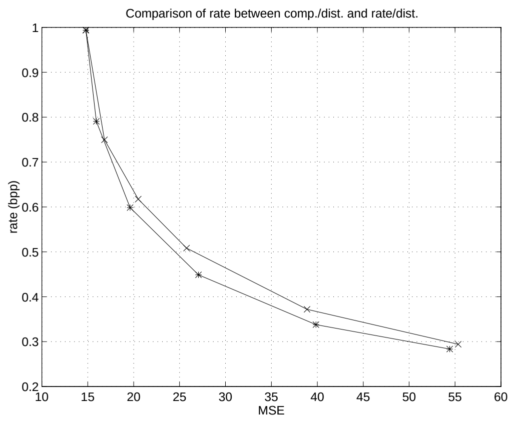


(b)

Figure 4. (a) estimated IDCT complexity (b) experimental decoding complexity. Results for optimized tree-structured algorithm for each quantizer ('o'), C-D curve with different algorithms for each quantizer ('x'), C-D curves using a single algorithm optimized for MSE=60.66 ('*') and MSE=14.80 ('+').



(a)



(b)

Figure 5. C-D (a) and R-D (b) curves achieved with either a complexity ('x') or a rate ('*') budgets.