INCREMENTAL REFINEMENT OF COMPUTATION FOR THE DISCRETE WAVELET TRANSFORM

Yiannis Andreopoulos[*] *and Mihaela van der Schaar*[+]

[*]Queen Mary University of London
Dept. of Electronic Engineering
Mile End Road, London E1 4NS, United Kingdom
yiannis.a@elec.qmul.ac.uk

[+]University of California, Los Angeles
Dept. of Electrical Engineering
420 Westwood Plaza, Los Angeles, CA, 90095
mihaela@ee.ucla.edu

ABSTRACT

Contrary to the conventional paradigm of transform decomposition followed by quantization, we investigate the computation of two-dimensional discrete wavelet transforms (DWT) under quantized representations of the input source. The proposed method builds upon previous research on approximate signal processing and revisits the concept of incremental refinement of computation: Under a refinement of the source description (with the use of an embedded quantizer), the computation of the forward and inverse transform refines the previously-computed result thereby leading to incremental computation of the output. We study for which input sources (and computational-model parameters) can the proposed framework derive identical reconstruction accuracy to the conventional approach without any incurring computational overhead. This is termed successive refinement of computation, since all representation accuracies are produced incrementally under a single (continuous) computation of the refined input source and with no overhead in comparison to the conventional calculation approach that specifically targets each accuracy level and is not refinable.

Index Terms— *Approximate Signal Processing, Discrete Wavelet Transform, Computational Complexity, Incremental Refinement of Computation*

1. INTRODUCTION

A common use of multidimensional transform representations is in decorrelating input data and aiding the processing or compression of multimedia information, such as images or video sequences [1] [2]. Conventional approaches produce the transform representation to a maximum degree of precision needed and then quantize and process (or compress) the transform coefficients. Even though this approach follows the conceptual design of such systems in a straightforward manner, it wastes system resources in many cases where lossy representations of the input are required, e.g. at medium to low-bitrate coding, as noted by several authors [3] [5]. This realization led to schemes for computational optimization of transforms [3] [6], or scalable computation schemes [5], where the computational resources increase monotonically under increased precision requirements. These studies focused mainly on the mainstream case of the discrete cosine transform (DCT) [5] [6], but similar studies can be envisaged for other popular transforms [3] such as the discrete wavelet transform (DWT) or the Karhunen-Loève transform. However, it is hard to precisely estimate in advance the exact required quantization precision for the coding or decoding of a certain source [5]. Hence, rate-distortion tradeoffs are introduced to the problem and one cannot achieve the same representation accuracy as the original (albeit wasteful) computation of the transform. In addition, from a system perspective, these schemes produce an "all or nothing" representation: the computation cannot be interrupted when resources become unavailable and retrieve a meaningful approximation of the final result. One exception to this rule is the work of Winograd and Nawab [4] on incremental refinement of the discrete Fourier transform (DFT) where they display a first realization of incremental refinement of computation. However, apart from the DFT, their work did not attempt to extend these schemes to other global transforms, such as the DWT, where one faces different challenges.

We propose a novel approach that tackles these issues. We focus on the case of the DWT, as it is a popular choice for embedded coding applications [1] [2]. Section 2 proposes a bitplane-based computation for the 2-D DWT and analyzes its design for the multilevel decomposition and reconstruction. In order to analyze the results, a computational model is introduced in Section 3. Together with a stochastic modeling framework of the input source data from our extended work [10], this model is used to test the proposed computation with real video data produced by a motion-compensated wavelet video coder in order to practically validate the feasibility and the benefits of incremental refinement of computation for real-world multimedia data.

2. LIFTING-BASED DWT UNDER INCREMENTAL REFINEMENT OF COMPUTATION

State-of-the-art entropy coding schemes in scalable coders are naturally providing complexity-scalability with respect to the compression bitrate. A common misconception about rate-scalable video codecs is that they are inherently complexity scalable. While this is true to a certain extent, the provided complexity scalability range is limited [5]. The fundamental reason behind the lack of complexity scalability is the fact that the conventional approaches for the computation of the transform and motion compensation do not scale in function of the bitrate (which corresponds to a certain quantization precision) and do not adapt to content variations and content features. We investigate a systematic way of converting the traditional computation of the DWT decomposition and reconstruction into incremental refinement structures that are not only scalable, but can also refine the computation under refined quantization of the input.

A. Description of the Basic Algorithm for Incremental Refinement of Computation of the DWT

For incremental refinement of multidimensional transforms, it is essential that the appropriate transform computation scheme is used. Separable transforms such as the DWT are conventionally implemented via a separable approach consisting of rowwise filtering followed by columnwise filtering of the intermediate results [1]. However, this approach is not suitable for incremental refinement structures, as incremental refinement of the intermediate results does not represent a successive approximation

of the final transform coefficients [4]. Hence, we need to focus on the direct 2-D computation of the DWT. In addition, the most-efficient computational scheme for the transform realization should be used, in order for the results to be relevant with the current state-of-the-art. In the case of the DWT, this corresponds to the factorization of the analysis and synthesis polyphase matrices using the lifting scheme [1]. Consider the 2-D DWT decomposition via K 2-D "predict" and "update" lifting steps performed for a 2-D image block $\mathbf{X}$ (of $R \times C$ pixels) [7]:

$$\mathbf{M} = \mathbf{\Upsilon}\Big(\mathbf{A}_{\mathrm{u}(K)}\Big(\mathbf{A}_{\mathrm{p}(K)}\Big(\cdots\Big(\mathbf{A}_{\mathrm{u}(1)}\Big(\mathbf{A}_{\mathrm{p}(1)} \cdot \mathbf{X} \cdot {\mathbf{A}_{\mathrm{p}(1)}}^{T}\Big){\mathbf{A}_{\mathrm{u}(1)}}^{T}\Big)\cdots\Big){\mathbf{A}_{\mathrm{p}(K)}}^{T}\Big){\mathbf{A}_{\mathrm{u}(K)}}^{T}\Big){\mathbf{\Upsilon}}^{T}. \quad (1)$$

where $\mathbf{A}_{\mathrm{p}(k)}$, $\mathbf{A}_{\mathrm{u}(k)}$ are the kth prediction and update steps ($1 \le k \le K$), respectively, and $\mathbf{\Upsilon}$ is a diagonal scaling matrix in order to approximate an orthonormal decomposition.

The last equation is computing the 2-D DWT as a series of basic computation steps in two dimensions, starting from the inner part of (1), i.e. $\mathbf{A}_{\mathrm{p}(1)} \cdot \mathbf{X} \cdot {\mathbf{A}_{\mathrm{p}(1)}}^{T}$, and working outwards to the final result $\mathbf{M}$. If we use embedded double-deadzone quantization of the input $\mathbf{X}$ with basic partition cell Δ=1 [1], each quantized coefficient $x_{r,c}^{\mathrm{quant}}$ of the input $\mathbf{X}$ ($0 \le r < R$, $0 \le c < C$) is represented by:

$$x_{r,c}^{\mathrm{quant}} = (-1)^{\{x_N\}_{r,c}} \sum_{n=0}^{N-1} \{x_n\}_{r,c} \cdot 2^n \quad (2)$$

with $\{x_n\}_{r,c}$ the nth bit of quantized coefficient $x_{r,c}^{\mathrm{quant}}$ (where $\{x_0\}_{r,c}$ is the least-significant bit), and $\{x_N\}_{r,c}$ is the sign bit. This is the popular case of successive approximation quantization (SAQ), where, starting from the sign bit, each additional bitplane corresponds to increased precision in the approximation of $x_{r,c}$.

For incremental refinement under the SAQ approximation of each input sample $x_{r,c}$, the proposed computation of the first predict step for each bitplane n, $0 \le n < N$, is:

$$m_{2i,2j}^{\mathrm{quant,p(1)}} = (\text{-}1)^{\{x_N\}_{2i,2j}} \{x_n\}_{2i,2j} \quad (3)$$

$$m_{2i,2j+1}^{\mathrm{quant,p(1)}} = 2^n (\text{-}1)^{\{x_N\}_{2i,2j+1}} \{x_n\}_{2i,2j+1} + \sum_{l=0}^{T_{\mathrm{p}(1)}-1} \Big(a_l^{\mathrm{quant,p(1)}} \cdot 2^n (\text{-}1)^{\{x_N\}_{2i,2(j+l)-P_{\mathrm{p}(1)}}} \{x_n\}_{2i,2(j+l)-P_{\mathrm{p}(1)}} \Big) 2^{-s_l^{\mathrm{p}(1)}} \quad (4)$$

$$m_{2i+1,2j+1}^{\mathrm{quant,p(1)}} = 2^n (\text{-}1)^{\{x_N\}_{2i+1,2j+1}} \{x_n\}_{2i+1,2j+1} + \sum_{l=0}^{T_{\mathrm{p}(1)}-1} \Big[a_l^{\mathrm{quant,p(1)}} \cdot \Big(2^n (\text{-}1)^{\{x_N\}_{2i+1,2(j+l)-P_{\mathrm{p}(1)}}} \{x_n\}_{2i+1,2(j+l)-P_{\mathrm{p}(1)}} + m_{2(i+l)-P_{\mathrm{p}(1)},2j+1}^{\mathrm{quant,p(1)}} \Big) \Big] 2^{-s_l^{\mathrm{p}(1)}} \quad (5)$$

$$m_{2i+1,2j}^{\mathrm{quant,p(1)}} = 2^n (\text{-}1)^{\{x_N\}_{2i+1,2j}} \{x_n\}_{2i+1,2j} + \sum_{l=0}^{T_{\mathrm{p}(1)}-1} \Big(a_l^{\mathrm{quant,p(1)}} \cdot 2^n (\text{-}1)^{\{x_N\}_{2(i+l)-P_{\mathrm{p}(1)},2j}} \{x_n\}_{2(i+l)-P_{\mathrm{p}(1)},2j} \Big) 2^{-s_l^{\mathrm{p}(1)}} \quad (6)$$

where $a_l^{\mathrm{quant,p(1)}}$ are the coefficients of the prediction matrix $\mathbf{A}_{\mathrm{p}(1)}$ and $m_{2i+\{0,1\},2j+\{0,1\}}^{\mathrm{quant,p(1)}}$ is the output quadrant of coefficients, as computed from the signed quantized values (signed bitplanes) $(\text{-}1)^{\{x_N\}_{r,c}} \{x_n\}_{r,c}$ of the input. Equation (3) is a simple copy operation between input and output, while (4)-(6) predict the input coefficient bit of each case by an addition with a factor that depends on: i) the nth signed bit values of the coefficients in the spatial neighborhood of $(\text{-}1)^{\{x_N\}_{2i+\{0,1\},2j+\{0,1\}}} \{x_n\}_{2i+\{0,1\},2j+\{0,1\}}$; ii) the quantized representation of the lifting taps $a_l^{\mathrm{quant,p(1)}}$ corresponding to matrix $\mathbf{A}_{\mathrm{p}(1)}$, which, for the case of the 9/7 filter-pair, can be found in [1]. All scalings with 2^w, $w \in \mathbb{Z}$, are implemented with bit-shifting operations and amount to negligible complexity in relation to multiplications or additions. Notice that there are only a limited number of different inputs possible for the input coefficients of (4)–(6), and consequently only a limited number of distinct values can be produced. For the case of the 9/7 filter-bank for example, after a few straightforward calculations we find that only 15 input and output values are possible for (4) and (6), and only 99 possible input/output values are possible for (5). These values can be pre-computed in advance and stored in a small lookup table with negligible memory cost. As a result, the entire calculation required for the first prediction step given by (3)-(6) is performed with lookup tables containing the precomputed result for each possible value of a coefficient $m_{2i+[0,1],2j+[0,1]}^{\mathrm{quant,p(1)}}$ of $\mathbf{M}$. Once the possible values of both lookup tables have been pre-computed, there is no further computational overhead for the calculation of (3)-(6).

The second matrix product, $\mathbf{A}_{\mathrm{u}(1)} \cdot \mathbf{M}^{\mathrm{p}(1)} \cdot {\mathbf{A}_{\mathrm{u}(1)}}^{T}$, is produced by reusing the results of (3)-(6):

$$m_{2i+1,2j+1}^{\mathrm{quant,u(1)}} = m_{2i+1,2j+1}^{\mathrm{quant,p(1)}} \quad (7)$$

$$m_{2i+1,2j}^{\mathrm{quant,u(1)}} = m_{2i+1,2j}^{\mathrm{quant,p(1)}} + \sum_{l=0}^{T_{\mathrm{u}(1)}} \Big(a_l^{\mathrm{quant,u(1)}} \cdot m_{2i+1,2(j+l)-P_{\mathrm{u}(1)}}^{\mathrm{quant,p(1)}} \Big) 2^{-s_l^{\mathrm{u}(1)}} \quad (8)$$

$$m_{2i,2j}^{\mathrm{quant,u(1)}} = m_{2i,2j}^{\mathrm{quant,p(1)}} + \sum_{l=0}^{T_{\mathrm{u}(1)}} \Big[a_l^{\mathrm{quant,u(1)}} \cdot \Big(m_{2i,2(j+l)-P_{\mathrm{u}(1)}}^{\mathrm{quant,p(1)}} + m_{2(i+l)-P_{\mathrm{u}(1)},2j}^{\mathrm{quant,u(1)}} \Big) \Big] 2^{-s_l^{\mathrm{u}(1)}} \quad (9)$$

$$m_{2i,2j+1}^{\mathrm{quant,u(1)}} = m_{2i,2j+1}^{\mathrm{quant,p(1)}} + \sum_{l=0}^{T_{\mathrm{u}(1)}} \Big(a_l^{\mathrm{quant,u(1)}} \cdot m_{2(i+l)-P_{\mathrm{u}(1)},2j+1}^{\mathrm{quant,p(1)}} \Big) 2^{-s_l^{\mathrm{u}(1)}} \quad (10)$$

where $m_{2i+\{0,1\},2j+\{0,1\}}^{\mathrm{quant,u(1)}}$ is the output quadrant of coefficients, as computed for the output coefficients $m_{r,c}^{\mathrm{quant,p(1)}}$ of the prediction step of (3)-(6). Again, (7) is a simple copy operation, while for (8)-(10) the update filter coefficients $a_l^{\mathrm{quant,u(1)}}$ update the output (see [7] for an example instantiation of these coefficients). We do not use lookup tables for (8)-(10) because the dynamic range grows larger with each matrix product. The remaining steps $k = 2,\ldots,K$ to complete the single-level decomposition using the 2-D lifting formulation of (1) are performed as in (3)-(6) and (7)-(10) with the replacement of the input with the output of each previous step and the replacement of the coefficients $a_l^{\mathrm{quant,p(1)}}$, $a_l^{\mathrm{quant,u(1)}}$ by $a_l^{\mathrm{quant,p}(k)}$, $a_l^{\mathrm{quant,u}(k)}$, respectively. All steps can be performed in-place as in the conventional lifting decomposition, with the reuse of the memory for the $\mathbf{M}^{\mathrm{p}(1)}$ and $\mathbf{M}^{\mathrm{u}(1)}$ arrays. Once all the steps have been completed for the current decomposition level, the final computation is reordered in the output matrix $\mathbf{M}$ in binary-tree ("Mallat") form and the produced results of each bitplane are added to the results of the previous (more significant) bitplanes, if existing. Finally, the scaling performed at the end of each decomposition level in the conventional decomposition can be skipped altogether, or incorporated into the encoding stage of each bitplane and as a result it is not explicitly considered.

B. Extension to Multiple Levels and Transform Inversion

In order to extend the bitplane-based computation to multiple levels, we propose a "depth-first" incremental refinement of computation by continuing the bitwise lifting-scheme computations for all subsequent levels after the termination of one bitplane at the first level. This is achieved by reformulating the first prediction step of (3)-(6) for all levels beyond one as ($0 \le i < R/2^l$, $0 \le j < C/2^l$, $2 \le l \le L_{\max}$):

$$m_{2i,2j}^{\mathrm{quant,p(1)}} = m_{4i,4j}^{\mathrm{quant,u}(K)} \quad (11)$$

$$m_{2i,2j+1}^{\mathrm{quant,p(1)}} = m_{4i,4j+2}^{\mathrm{quant,u}(K)} + \sum_{l=0}^{T_{\mathrm{p}(1)}-1} \Big(a_l^{\mathrm{quant,p(1)}} \cdot m_{4i,4(j+l)-2P_{\mathrm{p}(1)}}^{\mathrm{quant,u}(K)} \Big) 2^{-s_l^{\mathrm{p}(1)}} \quad (12)$$

$$m_{2i+1,2j+1}^{\text{quant,p}(1)} = m_{4i+2,4j+2}^{\text{quant,u}(K)} + \sum_{l=0}^{T_{\text{p}(1)}-1}\Big[a_l^{\text{quant,p}(1)} \cdot\Big(m_{4i+4,4(j+l)-2P_{\text{p}(1)}}^{\text{quant,u}(K)} + m_{2(i+l)-P_{\text{p}(1)},2j+1}^{\text{quant,p}(1)}\Big)\Big]2^{-s_l^{\text{p}(1)}} \quad (13)$$

$$m_{2i+1,2j}^{\text{quant,p}(1)} = m_{4i+2,4j}^{\text{quant,u}(K)} + \sum_{l=0}^{T_{\text{p}(1)}-1}\Big(a_l^{\text{quant,p}(1)} \cdot m_{4(i+l)-2P_{\text{p}(1)},4j}^{\text{quant,u}(K)}\Big)2^{-s_l^{\text{p}(1)}} \quad (14)$$

where we use the outputs $\mathbf{M}^{\text{u}(k)}$ of the last update step of the previous level and, importantly, at the end of the previous level we only perform the reordering and addition to the previous results for the high-frequency wavelet coefficients ($m_{2r,2c+1}^{\text{quant,u}(k)}$, $m_{2r+1,2c}^{\text{quant,u}(k)}$, $m_{2r+1,2c+1}^{\text{quant,u}(k)}$, $0 \le r < R/2^{l-1}$, $0 \le c < C/2^{l-1}$) and do the reordering of the low-frequency coefficients ($m_{2r,2c}^{\text{quant,u}(k)}$) only for the final (coarsest) decomposition level $L_{\max}$. This is due to the fact that each intermediate level receives the necessary low-frequency coefficients of the previous level with the change of indexing performed in (11)-(14) in comparison to (3)-(6).

Notice that, under the changed computation proposed in (11)-(14), the use of lookup tables becomes cumbersome for the first prediction step of all levels beyond one, as the inputs $m_{2r,2c}^{\text{quant,u}(K)}$ may have high dynamic range. Hence the computation is performed without the use of lookup tables for these cases.

Concerning the inverse transform, the process is exactly anti-symmetric as in the conventional lifting computation: all lifting steps are performed in reverse order by solving the forward bitwise lifting equations for the coefficients being predicted or updated during the forward transform. All the steps are executed in reverse order and from the coarsest level to the finest one.

3. MODELING OF INCREMENTAL REFINEMENT OF COMPUTATION

The proposed algorithm for incremental refinement of computation of the DWT present a data-adaptive computation where each non-zero input bit causes a certain amount of computation towards the completion of the transform decomposition. In order to understand the behavior of such approaches better, we analyzed the expected performance using stochastic source models in our extended work [10].

We are quantifying the benefits of conventional transform calculation versus incremental refinement in terms of the computational effort required to complete the decomposition or reconstruction task, whether it is for a single bitplane or for the entire set of bitplanes. Since arithmetic operations in the proposed incremental refinement approaches deal with data with significantly-reduced bitwidth in comparison to the conventional computation of the DWT that processes all bitplanes at once, we propose the following metrics used for the area-time complexity of binary multiplication and addition [8].

Definition 1: Addition and multiplication of two numbers represented with N_1 and N_2 bits, each having an additional bit as the sign bit as in (2), requires the following number of operations:

$$\text{Cost}_{\text{add}} = \begin{cases} 0, & \text{if one of the two numbers is zero} \\ \max\{N_1, N_2\} + 1, & \text{otherwise} \end{cases} \quad (15)$$

$$\text{Cost}_{\text{mult}} = \begin{cases} 0, & \text{if one of the two numbers is zero} \\ (\max\{N_1, N_2\}+1)(\min\{N_1, N_2\})^{1+\xi}, & \text{otherwise} \end{cases} \quad (16)$$

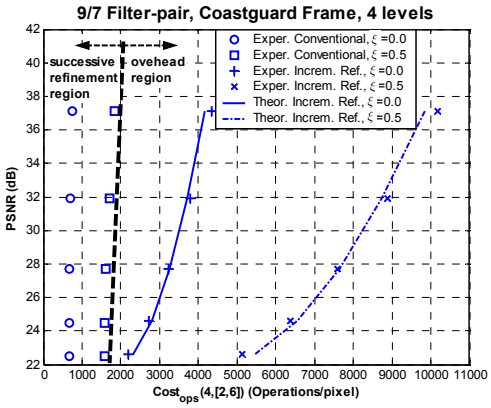
with $\xi \ge 0$ a system parameter indicating how "hard" is binary multiplication in comparison to binary addition. ■

This definition can be intuitively viewed as follows: Assume a virtual processing element (PE) able to perform signed addition between two bits and the carry information. Addition is (maximally) requiring $\max\{N_1, N_2\} + 1$ activations of the PE for two numbers with N_1 and N_2 bits. Similarly, by viewing multiplication as cumulative additions, the number of activations of the PE is given by (16), with the system parameter ξ indicating the cost of accumulating the intermediate results. If any of the two operands is zero, no operations are required (apart from a minimal "zero detection" effort), since the result is trivial.

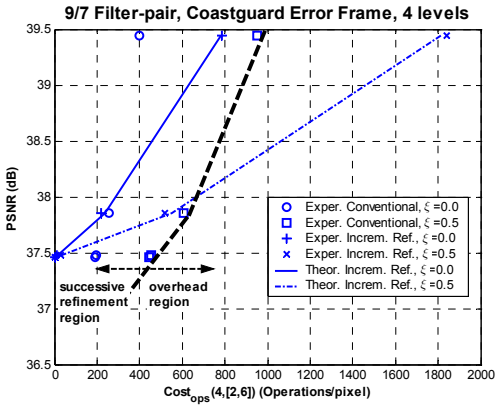
Based on the complexity definitions presented here, we derived the expected number of arithmetic operations based on stochastic source modeling [10]. Due to lack of space, we are only presenting a short validation of our results in this paper.

4. EXPERIMENTAL RESULTS

We first examine the performance of incremental refinement in individual cases of IDWT applied to video frames as well as error frames produced by motion-compensated prediction. Subsequently, we test the performance of incremental refinement of computation for the case of entire video sequences decompressed with a scalable video coder.



(a)



(b)

Figure 1. Computation-Distortion results for the original (conventional) approach for the computation of the IDWT versus the proposed approach. We display two cases: $\xi = 0.0$ (solid lines) and $\xi = 0.5$ (dashed lines). A representative video frame (part a) and an error frame (part b) were used. We also indicate the model results and the successive refinement region for $\xi = 0.5$.

Figure 1 presents representative results from a video sequence in terms of computation (measured based on (15), (16) in conjunction with our experimental software implementation) versus distortion achieved by stopping at several bitplanes (from $n = 6$ to $n = 2$) and focusing directly at the multilevel decomposition with $L_{\max} = 4$. The results of the conventional (non-refinable) approach are measured also based on (15), (16) and by performing multiple IDWTs. The 9/7 filter-pair was used for

these results and throughout this section. We set the precision for fixed-point software implementation to 13 bits for the fractional part and 13 bits for the integer part (including the sign bit). In the results of this section we focus on the "depth-first" approach for the proposed incremental refinement since we determined experimentally that it performs almost identically to the "breadth-first" approach and it also represents the generic multilevel incremental refinement paradigm where each input source bitplane refines the final result of the IDWT.

Starting with the case of regular (intra) video frames (part (a) of Figure 1), we observe that the proposed approach does not achieve performance close or inside the successive refinement region marked by the computation required by the conventional approach. To the contrary, it introduces significant computational overhead for both cases of ξ values tested. However, it appears that the incremental refinement case with $\xi = 0$ is close to the conventional case for $\xi = 0.5$ in the high-distortion (low PSNR) region. This is a scenario that could be feasible in practice, since the overhead of multiplication versus addition (expressed by increased values of ξ) increases with increased dynamic range, and this is indeed the case of the conventional approach versus the proposed incremental refinement of computation: the conventional computation performs less operations but with a higher dynamic range on average, in comparison to the proposed approach. The theoretical results of our extended work [10] agree well with the experimental results for the case of intra frames seen in Figure 1.

In general, the results of Figure 1 indicate that the proposed approach appears to provide benefits mostly for the case of error frames. As a result, it would be interesting to evaluate its performance in a coding framework using motion-compensated prediction. Representative results are presented in Figure 2 in terms of mean PSNR for embedded coding of the Y, U, V channels of CIF-resolution video using a scalable video coder [10] with a group-of-pictures structure consisting of one intra-frame and 23 error frames. The corresponding bitrates are within the range of 128~1024 kbps. We have included as a reference the computation required for the conventional separable lifting calculation [1], because it is commonly used in practical applications [1] [2]. The results indicate that the proposed approach becomes beneficial for the medium to low-rate regime corresponding to medium to high distortion (28 db - 31 dB in PSNR).

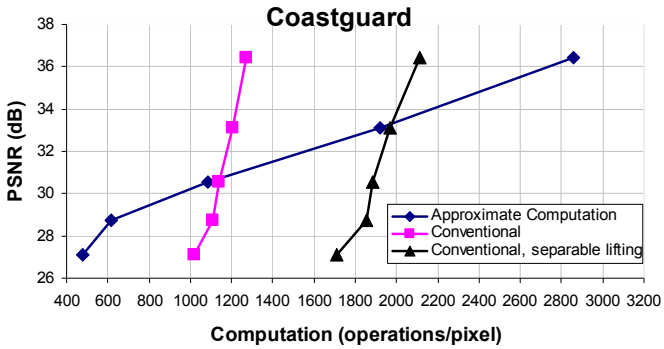


Figure 2. Computation-Distortion results for the conventional approach versus the proposed approach in the case of video sequences. The computation required by the non-refinable separable lifting is included as well as a reference.

Figure 2 shows the striking difference in complexity scalability between the proposed approach and the conventional computation (both the two-dimensional and the separable lifting approach). Concerning the low-distortion (high-rate) regime, an overhead ranging up to three-fold increase in computation is imposed due to the incremental refinement property. We remark though that, since the comparison is carried out under the direct 2-D lifting-based realization, which is already reducing computation by 40% in comparison to the conventional separable lifting-based DWT [7], the proposed approach performs significantly better when compared to the conventional separable lifting realization. Notice that all the results presented for each sequence in Figure 2 are produced incrementally by the proposed approach (with respect to the IDWT part), while both conventional computations require reinitiating the computation for each experimental point reported in the figure. This indicates that under dynamic resource allocation, the proposed approach can seamlessly refine the output representation. Moreover, when system resources become scarce, the proposed approach can still produce a high-distortion, albeit meaningful, result unlike the conventional computation that is constrained to a certain range of computation.

5. CONCLUSIONS

We propose a method for incremental computation of the forward and inverse DWT under a bitplane-based formulation of the 2-D lifting decomposition. This results in a continuous computation for the output where, under any termination point, the representation corresponding to the provided input source accuracy can be retrieved. In addition, to fine-granular and scalable computation, the proposed DWT calculation ensures that, should additional computational resources be provided, the transform calculation can be enhanced from the previously-computed result. A first exploration of the subspace of operational parameters for which incremental refinement provides comparable or even superior computational efficiency in comparison to the conventional (non-refinable) computation is presented. The results demonstrate that successive refinement of computation is feasible for the medium to high distortion regime.

REFERENCES

[1] *JPEG2000: Image Compression Fundamentals, Standards and Practice*, D. Taubman, M. Marcellin, Kluwer Acad. Pubs, 2002.

[2] J.-R. Ohm, "Advances in scalable video coding," *Proc. of the IEEE*, vol. 93, pp. 42-56, Jan. 2005.

[3] V. K. Goyal, and M. Vetterli, "Computation-distortion characteristics of block transform coding," *Proc. IEEE Int. Conf. on Accoust., Speech, and Signal Proc.*, ICASSP-97, vol. 4, pp. 2729-2732, April 1997.

[4] J. Winograd and S. H. Nawab, "Incremental refinement of DFT and STFT approximations," *IEEE Signal Proc. Letters*, vol. 2, no. 2, pp. 25-27, Feb. 1995.

[5] K. Lengwehasatit and A. Ortega, "Scalable variable complexity approximate forward DCT," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 14, no. 11, pp. 1236-1248, Nov. 2004.

[6] Z. Wang, "Pruning the fast discrete cosine transform," *IEEE Trans. on Comm.*, vol. 39, no. 5, pp. 640-643, May 1991.

[7] H. Meng and Z. Wang, "Fast spatial combinative lifting algorithm of wavelet transform using the 9/7 filter for image block compression," *IEE Electronics Letters*, vol. 36, no. 21, pp. 1766-1767, Oct. 2000.

[8] R. P. Brent and H. T. Kung, "The area-time complexity of binary multiplication," *J. of the Assoc. for Comp. Machin.*, vol. 28, no. 3, pp. 512-534, Jul. 1981.

[9] W. H. R. Equitz and T. Cover, "Successive refinement of information," *IEEE Trans. on Inform. Theory*, vol. 37, no. 2, pp. 269-275, Mar. 1991.

[10] Y. Andreopoulos and M. van der Schaar "Incremental refinement of computation for the discrete wavelet transform," *IEEE Trans. on Signal Process.*, to appear.