

Optimal Tiling Algorithms for Inter-frame Video Compression.

Kai-Lung Hua, Mary Comer, and Ilya Pollak*

Abstract

We propose the use of large dictionaries of tilings for video compression and develop fast algorithms to select the optimal tiling for both the motion compensation and transform stages of a video coder. We illustrate the effectiveness of this approach by showing that our tiling selection method results in up to 23% savings in bit rate as compared to the H.264/AVC tiling selection, for several standard video sequences.

Index Terms

Large tree-structured dictionaries, motion compensation, transform, H.264/AVC.

I. INTRODUCTION

Algorithms for best basis search in tree-structured dictionaries have been effectively used for many signal processing problems, including noise removal and other estimation tasks [2]–

This work was supported in part by a National Science Foundation (NSF) CAREER award CCR-0093105, an NSF grant IIS-0329156, and a MediaTek fellowship.

Parts of this work have been presented at the 2006 IEEE International Conference on Image Processing [5] and the 2006 Asilomar Conference on Signals, Systems, and Computers [6].

The authors are with the School of Electrical and Computer Engineering, Purdue University, 465 Northwestern Ave., West Lafayette, IN 47907, phone 765-494-3465, 3486, and 5916, fax 765-494-3358, e-mail khua,comerm,ipollak@ecn.purdue.edu. Corresponding author's e-mail: ipollak@ecn.purdue.edu.

[4], [10], [11], [18], [22], [30], image compression [7], [8], [12]–[15], [17], [23]–[27], audio compression [20], [21], and image segmentation [19]. An important class of best basis algorithms are methods that search for an optimal rectangular tiling of an image [1]–[3], [7], [8], [12], [13], [17], [19], [23]–[27]. Such methods can significantly improve image coding strategies based on block or lapped transforms, through adapting the sizes and shapes of transform blocks to the structure of an image. The utility of using variable block sizes in video compression has recently been recognized, as well. For example, the H.264/AVC video compression standard [16] incorporates variable block sizes both in the motion compensation stage [28] and in the transform stage [29].

In the present paper, we show that compression performance can be further improved by searching for the optimal tiling in our *dyadic* and *multitree* dictionaries introduced in [8], which are both much larger than the set of tilings allowed for motion compensation and transform in H.264/AVC. We define a rate-distortion cost and use the efficient tiling algorithm of [8] to minimize the cost over our dictionaries, for both the motion compensation and transform stages of our video coder. We show that, the proposed algorithms result in bit rate reductions of up to 23% at typical PSNRs, compared to H.264/AVC, on several video sequences commonly used to evaluate the performance of video coders. This is accomplished through only a modest increase in the computational complexity as compared to H.264/AVC. Note that, in general, the use of finer tilings results in more overhead bits and therefore may increase the overall bit rate. We show, however, that our optimal tiling search is so effective that the increase in the number of overhead bits is more than compensated for by the reduction in the number of bits required to encode the motion vectors and transform coefficients. Our experimental results demonstrate that the additional overhead bits for the more complex multitree tilings lead to only a small improvement in rate-distortion performance over the dyadic algorithm, making the more

computationally efficient dyadic scheme perhaps more attractive in applications.

The rest of this paper is organized as follows. Section II overviews the tiling selection for the motion compensation stage in H.264/AVC and introduces our proposed tiling selection methods for motion compensation. In Section III we introduce optimal tiling selection methods for the transform stage. Sections IV and V describe our evaluations of the proposed methods: Section IV describes our partial implementations of H.264/AVC and our own proposed video coders; and Section V presents the experimental rate-distortion curves. Section VI draws conclusions.

II. OPTIMAL TILINGS FOR MOTION COMPENSATION

A. Motion Compensation in H.264/AVC

There are two basic ways of encoding a frame of video: *intra* mode and *inter* mode. Intra-frame compression only uses the information contained within the current frame. Inter-frame mode predicts the current frame from one or several reference frames, and encodes the error between the predicted frame and the actual one. The ability to accurately predict the current frame is therefore crucial to the success of inter-frame compression. The prediction is typically done by motion estimation methods which partition a frame into blocks and match every block with a similar block in the reference frame. Conventional methods use square blocks of fixed size. The choice of the block size is problematic for such methods: if the size is too small, too many bits are spent on encoding the motion vectors; whereas if it is too large, the prediction of complicated motion sequences is poor. Generally speaking, areas with no motion should use large block sizes and areas with complicated motion should use small block sizes. Therefore, H.264/AVC incorporates variable block sizes in the motion compensation stage.

Specifically, H.264/AVC allows partitioning each 16×16 macroblock into rectangular sub-blocks and performing motion estimation separately for each subblock. The specific tilings

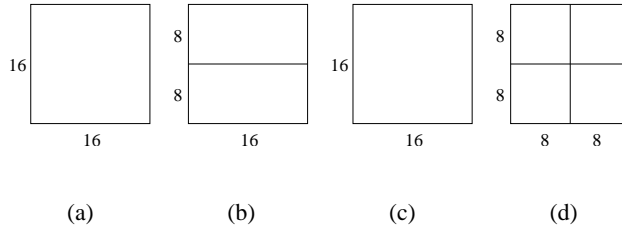


Fig. 1. Possible tilings of a 16×16 macroblock for the H.264/AVC motion estimation and compensation.

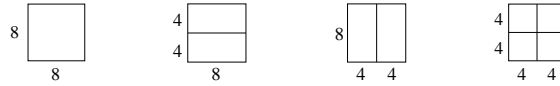


Fig. 2. Possible tilings of a 8×8 sub-macroblock for the H.264/AVC motion estimation and compensation.

allowed by the standard are shown in Figs. 1 and 2. One of the four tilings in Fig. 1 may be used for a 16×16 macroblock. If the tiling of Fig. 1(d) is used, then each of the four 8×8 sub-macroblocks can be further tiled using the four tilings of Fig. 2. The encoder in the H.264/AVC reference software [9] selects a tiling for each macroblock based on a cost function. Given a motion vector \mathbf{v} for such a subblock, [9] defines the distortion $d(\mathbf{v})$ as the sum, over all the pixels in the subblock, of the absolute differences between the pixel luminance and the luminance of the corresponding pixel in the reference frame, for integer pixel search; and the sum of absolute transformed differences for fractional pixel search. The rate $r(\mathbf{v})$ is defined as the number of bits required to encode the difference between the motion vector and the predicted¹ motion vector, as tabulated in [9]. The cost of a motion vector \mathbf{v} is defined as

$$d(\mathbf{v}) + \lambda_m r(\mathbf{v}),$$

where the Lagrange multiplier λ_m is

$$\lambda_m = \sqrt{0.85 \cdot 2^{(QP-12)/3}}, \quad (1)$$

and QP is the quantization parameter.

After determining the optimum motion vector for each subblock, the cost of each tiling of a

¹A motion vector for a block is predicted based on the neighboring blocks, and then the difference between the prediction and the actual motion vector is encoded.

macroblock into subblocks is defined as a weighted sum of the tiling's distortion D and rate R : $\text{COST} \equiv D + \lambda R$. Both the distortion and the rate are assumed to be additive over the subblocks in the tiling, i.e., it is assumed that the cost of a tiling that consists of subblocks P_1, \dots, P_d can be calculated as follows:

$$\text{COST} = \sum_{i=1}^d (D_{P_i} + \lambda R_{P_i}), \quad (2)$$

where D_{P_i} and R_{P_i} are the distortion and rate, respectively, for the subblock P_i , and the summation is taken over all subblocks in the tiling. Following [9], we define the distortion D_{P_i} to be the sum of the squared differences between each pixel² in the subblock P_i and its motion-compensated reconstruction.³ We define the rate R_{P_i} as the the sum of three terms: the number of bits $R_{P_i}^{(t)}$ to encode the selected tiling, the number of bits $R_{P_i}^{(e)}$ for the predicted motion vector error, and the number of bits $R_{P_i}^{(c)}$ for the quantized transform coefficients for subblock P_i :

$$R_{P_i} = R_{P_i}^{(t)} + R_{P_i}^{(e)} + R_{P_i}^{(c)}. \quad (3)$$

We estimate $R_{P_i}^{(c)}$ from training data, as explained in Appendix. Finally, the Lagrange multiplier in Eq. (2) is defined as $\lambda \equiv \lambda_m^2$ where λ_m is given by Eq. (1).

In our implementation of an H.264/AVC-compliant tiling scheme, we also follow [9]: the costs of the four tilings of Fig. 1 are evaluated; if one of the tilings (a), (b), or (c) has the lowest cost, this tiling is selected; if (d) has the lowest cost, the possibilities shown in Fig. 2 are evaluated for each of the four 8×8 sub-macroblocks. It is easily seen that the dictionary of tilings that this scheme selects from, contains a total of $3 + 4^4 = 259$ tilings. Note also that the procedure that selects a tiling out of these 259 tilings is not globally optimal, since the global

²Here, each pixel value is motion-compensated, i.e., it is the difference between the pixel value in the original frame and its prediction.

³The reconstruction is the result of the inverse quantization and inverse transform of the transformed, quantized pixel values.

minimum of the cost (2) may be achieved by a tiling which is finer than (d) even if tiling (a), (b), or (c) has a lower cost than (d).

B. Proposed Optimal Tiling Algorithm

We propose to use large dictionaries of tilings and to extract globally optimal tilings from these dictionaries using the algorithm developed in [8]. The proposed dictionaries are not H.264/AVC-compliant but result in improved compression ratios. We presently review the optimal tiling algorithm of [8], and then discuss its adoption to our motion compensation and transform task.

The algorithm constructs optimal tree-structured tilings of a rectangular image domain into smaller rectangles via recursive bipartitioning. During this process, a rectangle may either be used as a single tile or split further into two subrectangles. For example, the tiling of Fig. 3(a) may be obtained through such a recursive bipartitioning process, as illustrated in Fig. 3(c,d). In Fig. 3(c,d), a vertical (horizontal) line through a tree node signifies a vertical (horizontal) split of the corresponding rectangle into two subrectangles. Note that in this case, two different trees correspond to the same tiling. On the other hand, the rectangular tiling of Fig. 3(b) cannot be obtained through such a recursive binary splitting process.

Referring back to Fig. 3(c,d), we point out that in some applications the important object is the tiling produced by the leaves of the tree, and thus there is no distinction between the different trees that may have produced the tiling. In our application, however, the various tilings will be encoded by encoding the corresponding trees and therefore we will select the tree that corresponds to the most efficient encoding. This motivates defining the following cost function for a tree with leaves P_1, \dots, P_d and intermediate nodes Q_1, \dots, Q_{d-1} :

$$C(\text{tree}) = \sum_{i=1}^d e(P_i) + \sum_{i=1}^{d-1} s(Q_i), \quad (4)$$

where e and s are cost functions for individual tiles and intermediate tree nodes, respectively.

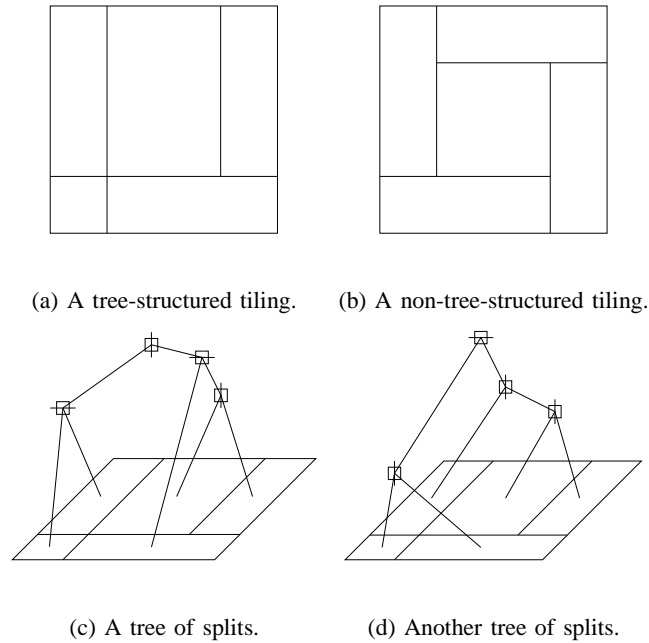
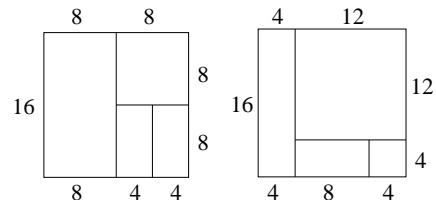


Fig. 3. An illustration of tilings and trees of splits. (a) An admissible tiling—i.e., a tiling that can be obtained via recursive binary splitting. (b) An inadmissible tiling. (c) A tree of splits that leads to the tiling in (a). (d) Another tree of splits that leads to the tiling in (a).



(a) Dyadic tiling (b) Multitree tiling

Fig. 4. Example of possible dyadic and multitree tilings.

The smallest tile size that we use is 4×4 . We define two dictionaries of trees: the dyadic dictionary and the multitree dictionary. The former contains all trees that produce tilings by only splitting rectangles in the middle, horizontally or vertically. The latter allows splits at arbitrary locations that are multiples of four. A tiling from the dyadic dictionary and a tiling from the multitree dictionary are illustrated in Fig. 4.

An efficient algorithm for finding the globally optimal tree is developed in [8]. Specifically,

	H.264/AVC	Dyadic	Multitree
number of tilings	259	6857	68480
number of tiles	41	49	100

TABLE I: NUMBER OF TILINGS AND TILES

this algorithm performs the following bottom-up optimization:

$$C_P^* = \min\{e(P), \min(C_{P'}^* + C_{P''}^*) + s(P)\}, \quad (5)$$

where C_P^* is the cost of the optimal tree with root P , and the inner minimization is performed over all pairs of subblocks P', P'' which partition P . This efficient search algorithm exploits the fact that although the number of possible trees and tilings is very large, the number of rectangular tiles is much smaller and manageable, as shown in Table I.

When adopting the optimal tiling algorithm to motion compensation, we define the cost function $e(P_i)$ of Eq. (4) as a weighted sum of the distortion D_{P_i} and the rate R_{P_i} :

$$e(P_i) = D_{P_i} + \lambda R_{P_i},$$

where, as in our implementation of H.264/AVC, D_{P_i} is defined as the sum of the squared differences between each pixel in the subblock P_i and its motion-compensated reconstruction, and $\lambda = \lambda_m^2$ with λ_m given by Eq. (1). We define the rate R_{P_i} as the the sum of two terms: the number of bits $R_{P_i}^{(e)}$ for the predicted motion vector error for subblock P_i and the number of bits $R_{P_i}^{(c)}$ for the quantized transform coefficients for subblock P_i :

$$R_{P_i} = R_{P_i}^{(e)} + R_{P_i}^{(c)}. \quad (6)$$

We estimate $R_{P_i}^{(c)}$ from training data, as explained in Appendix. The cost function $s(Q_i)$ is defined as λR_{Q_i} , where R_{Q_i} is the number of bits for encoding the split of the rectangle Q_i . Thus, the

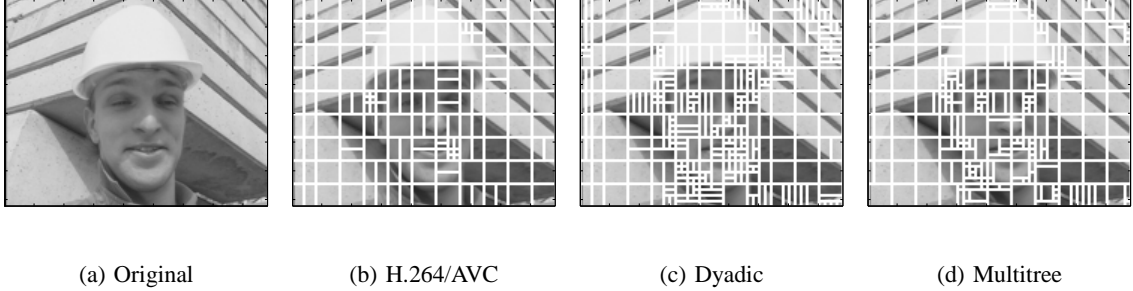


Fig. 5. A frame of the *Foreman* sequence (left) and its three tilings for the motion compensation stage produced by three different tiling methods.

overall cost of a tree is:

$$C(\text{tree}) = \sum_{i=1}^d \left[D_{P_i} + \lambda(R_{P_i}^{(e)} + R_{P_i}^{(c)}) \right] + \lambda \sum_{i=1}^{d-1} R_{Q_i}, \quad (7)$$

Fig. 5 shows examples of the different tiling selection methods for the motion compensation stage.

III. OPTIMAL TRANSFORM TILINGS

In this section, we first overview the transform stage of H.264/AVC and then explain how to apply the optimal tiling algorithm described above to extract the best transform tiling.

A. Transform Stage for H.264/AVC

Typically, a fixed 4×4 integer transform is used in H.264/AVC encoders. However, adaptive block size transforms (ABT) are also supported by H.264/AVC [29]. The basic idea of ABT is to associate the block size of the transform matrix with the block size of motion compensation subblocks. In general, for a rectangular $m \times n$ image $B_{m,n}$, the transform $\hat{B}_{m,n}$ is defined by

$$\hat{B}_{m,n} = T_v \cdot B_{m,n} \cdot T_h^T, \quad (8)$$

where T_v and T_h are the $m \times m$ and $n \times n$ transform matrices in the vertical and horizontal directions, respectively. The maximum allowed block size in H.264/AVC is 8×8 . If larger subblocks (i.e., 16×8 , 8×16 , and 16×16) are encoded, 8×8 transform matrix is used.

B. Optimal Transform Algorithm

We improve the compression algorithm further by using dyadic or multitree dictionary of tilings for the transform stage. We extract the optimal sub-tiling for each *motion compensation subblock* into *transform subblocks*, and jointly optimize the tiling for the motion compensation stage and the sub-tiling for the transform stage. Note that we use the 2D discrete cosine transform (DCT) for all block size transform matrices in this transform.

For each motion compensation subblock P_i , we select the tree with intermediate nodes $\{Q_{i,1}, \dots, Q_{i,k-1}\}$ which generates the sub-tiling $\{P_{i,1}, \dots, P_{i,k}\}$, so as to minimize

$$\sum_{j=1}^k (D_{P_{i,j}} + \lambda R_{P_{i,j}}) + \sum_{j=1}^{k-1} \lambda R_{Q_{i,j}}, \quad (9)$$

where $D_{P_{i,j}}$ is the sum of the squared differences (SSD) for the transform subblock $P_{i,j}$, λ is a Lagrange multiplier, $R_{P_{i,j}}$ is the number of bits for encoding the quantized transform coefficients for the transform subblock $P_{i,j}$, and $R_{Q_{i,j}}$ is the number of bits to encode the split of the rectangle $Q_{i,j}$. We estimate $R_{P_{i,j}}$ from training data, as explained in Appendix.

The set of all valid dyadic transform splits is defined the set of all splits of a valid transform subblock into two congruent valid transform subblocks. When using a multitree dictionary at the motion compensation, two extra types of splits are allowed, in addition to the dyadic splits:

- A $12 \times m$ transform subblock may be split horizontally into an $8 \times m$ and a $4 \times m$ transform subblocks, where m is 4, 8, 12, or 16;
- A $m \times 12$ transform subblock may be split vertically into an $m \times 8$ and an $m \times 4$ transform subblocks, where m is 4, 8, 12, or 16.

Fig. 6 shows examples of the different tiling selection methods for the transform stage with H.264/AVC motion compensation scheme. White and black lines indicate the tilings for motion compensation and transform stages, respectively.



Fig. 6. A frame of the *Carphone* sequence (left) and its two tilings: dyadic transform tiling with H.264/AVC motion compensation tiling (center) and multitree transform tiling with H.264/AVC motion compensation tiling (right). The motion compensation tiling and transform tiling are indicated with white and black lines, respectively.

IV. IMPLEMENTATION

A. H.264/AVC-Compliant Motion Estimation and Compensation, Transform, and Quantization

We implement H.264-compliant motion estimation and compensation, integer 4×4 transform, and quantization. Our search range for motion estimation is 16 pixels, and we use half-pixel and quarter-pixel refinements. We use DCT for optimal dyadic and multitree transforms.

B. Entropy Coding

The entropy coding losslessly encodes the source symbols into a bitstream. Two modes of entropy coding are used in H.264/AVC standard: context-based adaptive variable length coding (CAVLC) and context-based adaptive binary arithmetic coding (CABAC).

In our simplified implementation, instead of performing CAVLC or CABAC, we estimate the bit rate as the sum of the following three terms.

- The number of bits for the motion vectors, taken from a table in [9].
- The number of bits for the quantized transform coefficients, estimated as the entropy.
 - Let discrete random variable X represents quantized and transformed coefficients, with possible values x_1, \dots, x_n ; the entropy, the number of bits per symbol, then is calculated

as:

$$H(X) = - \sum_{i=1}^n p(x_i) \log_2 p(x_i)$$

where $p(x_i) = Pr(X = x_i)$.

- In order to capture the content-based adaptivity of the encoder, $p(x_i)$ is updated after every frame.
- Thus, if the number of quantized and transformed coefficient is M , the number of bits needed to encode these coefficients is estimated as $M \times H(X)$.
- The number of bits to encode the selected tiling:
 - for the H.264/AVC-compliant scheme, we use two bits to encode the four possibilities shown in Fig. 1, and eight additional bits if a refinement of Fig. 1(d) is selected;
 - for the optimal dyadic tiling scheme, we use “0” to encode that a subblock is not split, and “10” and “11” to encode a horizontal and a vertical split, respectively;
 - for the optimal multitree scheme, we use one bit to encode whether or not a subblock is split, and, if it is split, we use additional $\lceil \log_2(M + N - 2) \rceil$ bits to encode the location of the split, where $4M \times 4N$ is the size of the subblock.

V. EXPERIMENTAL RESULTS

The coding algorithms are evaluated using *Mother and Daughter*, *Salesman*, *Highway*, *Foreman*, and *Carphone* video sequences with 110 frames and ± 16 search range [Quarter common intermediate format (QCIF), 176×144 , luminance only]. Among these sequences, there are ones with little motion (*Mother and Daughter* and *Salesman*), with a medium amount of motion (*Highway*), and with a relatively large amount of motion (*Foreman* and *Carphone*). We use a 15-frame group of pictures (GOP) with coding pattern IPPPPPPPPPPPPPP, where I and P represent intra-frame and inter-frame respectively. Since our implementation works with single-channel

frames, and since our compression strategy only differs from H.264/AVC for inter-frames, all our rate and distortion calculations are done with the luminance component of the inter-frames only.

A. Experiment 1: Dyadic and Multitree Motion Compensations with Fixed 4×4 Transform.

In this experiment, the H.264/AVC fixed 4×4 integer transform is used for all compression algorithms. We use our implementation of H.264/AVC as a baseline, and evaluate our new compression strategies which incorporate our proposed adaptive tiling algorithms in the motion compensation stage only. Fig. 7 shows the rate-distortion curves for five test sequences. The right column of the figure shows the rate-distortion curves for three schemes with the bit rates displayed as percentages of the baseline bit rate.

It can be seen from Fig. 7 that the proposed dyadic and multitree motion compensation algorithms result in up to 19% savings in bit rate as compared to the H.264/AVC motion compensation. Note that for sequences that do not contain much motion, i.e., *Mother and Daughter* and *Salesman*, dyadic motion compensation achieves more savings in bit rate than multitree motion compensation at low PSNRs. The explanation for this is that when the transformed motion-compensated pixels for such sequences are quantized with a large quantization step, the overhead bits required to encode a split location will dominate the decision whether to split a rectangular block or keep it whole. Since fewer bits are required to encode a split location under the dyadic scheme than under a multitree scheme, the dyadic scheme will tend to select more complicated tiling structures, to result in less distortion. Similarly, for the sequence *Highway* which has moderate amount of motion, the dyadic scheme has a slight advantage over the multitree scheme at low PSNRs.

When applying our optimal tiling algorithm to the motion compensation stage, the running

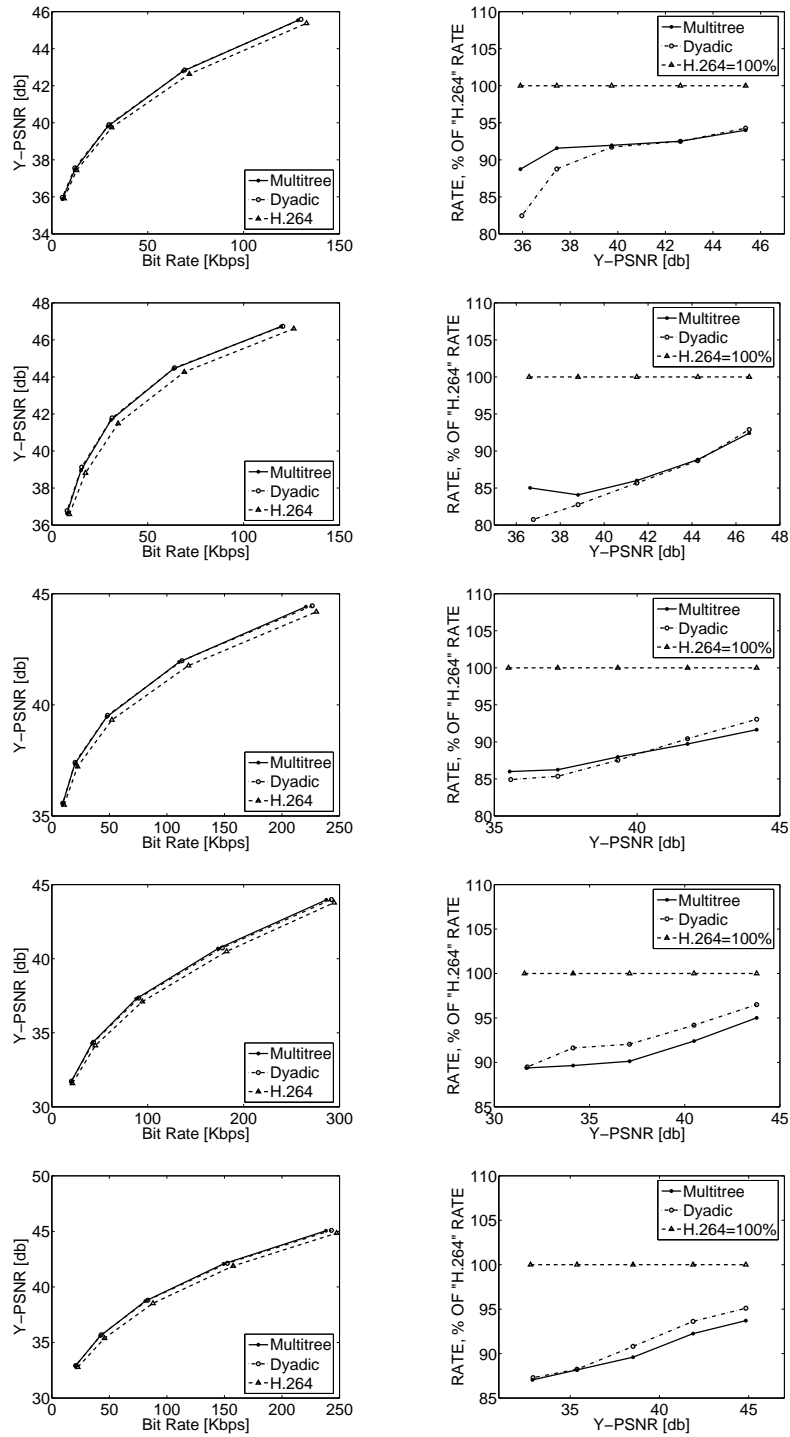


Fig. 7. The rate-distortion curves for compression with motion compensation using the H.264/AVC tiling scheme (dashed lines with triangles) and our proposed dyadic and multitree schemes (respectively: dashdot and solid lines). The right column shows bit rates as percentages of the bit rate for the baseline algorithm. The results are shown for frames 100 through 210 of “mother and daughter” (top), “salesman” (second row), “highway” (third row), “foreman” (forth row), and “carphone” (bottom).

times of our dyadic and multitree algorithms are only about a factor of 1.13 and 1.55, respectively, slower than our implementation of the H.264-compliant scheme.⁴

B. Experiment 2: H.264/AVC Motion Compensation Tiling with Dyadic and Multitree Transforms.

In this experiment, conventional H.264/AVC motion compensation is used for all compression algorithms. We use our implementation of H.264/AVC as a baseline, and evaluate our new compression strategies which incorporate our proposed adaptive tiling algorithms in the transform stage only. Fig. 8 shows the rate-distortion curves for the test sequences. The right column of the figure shows the rate-distortion curves for three schemes with the bit rates displayed as percentages of the baseline bit rate.

It can be seen from Fig. 8 that the proposed dyadic and multitree transforms result in up to 13% savings in bit rate as compared to the H.264/AVC 4×4 integer transform. The figure also shows that the rate-distortion curves for the dyadic and multitree transforms are very close in this case: the percentage difference in bit rate is within 3%. For sequences with little motion, 4×4 integer transform performs better than our optimal transform at low PSNRs, due to the overhead bits required to encode our transform tiling. In addition, comparing Fig. 8 with Fig. 7, we can see that the average gain from using our optimal tiling algorithms in the motion compensation stage is higher than the average gain from using them in the transform stage.

When we apply our optimal tiling algorithms to the transform stage, the running times of our dyadic and multitree algorithms are only about a factor of 1.43 and 1.66, respectively, slower

⁴Note that these factors are different from those reported in an earlier version of this work [5]. This is due to our improved implementation of the motion vector search.

than our implementation of the H.264-compliant scheme.⁵

C. Experiment 3: Dyadic Motion Compensation Tiling with Dyadic and Multitree Transforms.

In this experiment, our adaptive tiling algorithm with dyadic dictionary is used in the motion compensation stage. Our baseline is an H.264-compliant scheme which uses a fixed 4×4 transform. We evaluate our new compression strategies which, in addition to the dyadic tiling scheme in the motion compensation stage, incorporate the dyadic and multitree schemes in the transform stage. Fig. 9 shows the rate-distortion curves for the test sequences. The right column of the figure shows the rate-distortion curves for four schemes with the bit rates displayed as percentages of the baseline bit rate.

The combined use of optimal dyadic tilings for motion compensation and optimal multitree tilings for the transforms results in up to 21% savings in bit rate as compared to the baseline. However, again, we do not observe significant difference between using the optimal multitree tiling and the optimal dyadic tiling for the transform stage: the differences between the bit rates for the two are within 3%. Just like in the previous experiment, we observe that, for sequences with little motion, the 4×4 integer transform is the best choice at low PSNRs.

D. Experiment 4: Multitree Motion Compensation Tiling with Dyadic and Multitree Transforms.

In this experiment, our adaptive tiling algorithm with multitree dictionary is used in the motion compensation stage. Our baseline is an H.264-compliant scheme which uses a fixed 4×4 transform. We evaluate our new compression strategies which, in addition to the multitree tiling scheme in the motion compensation stage, incorporate the dyadic and multitree schemes in the transform stage. Fig. 10 shows the rate-distortion curves for the test sequences. The right

⁵Note that these factors are lower than reported in an earlier version of this work [6]. This is due to our improved implementation of the motion vector search.

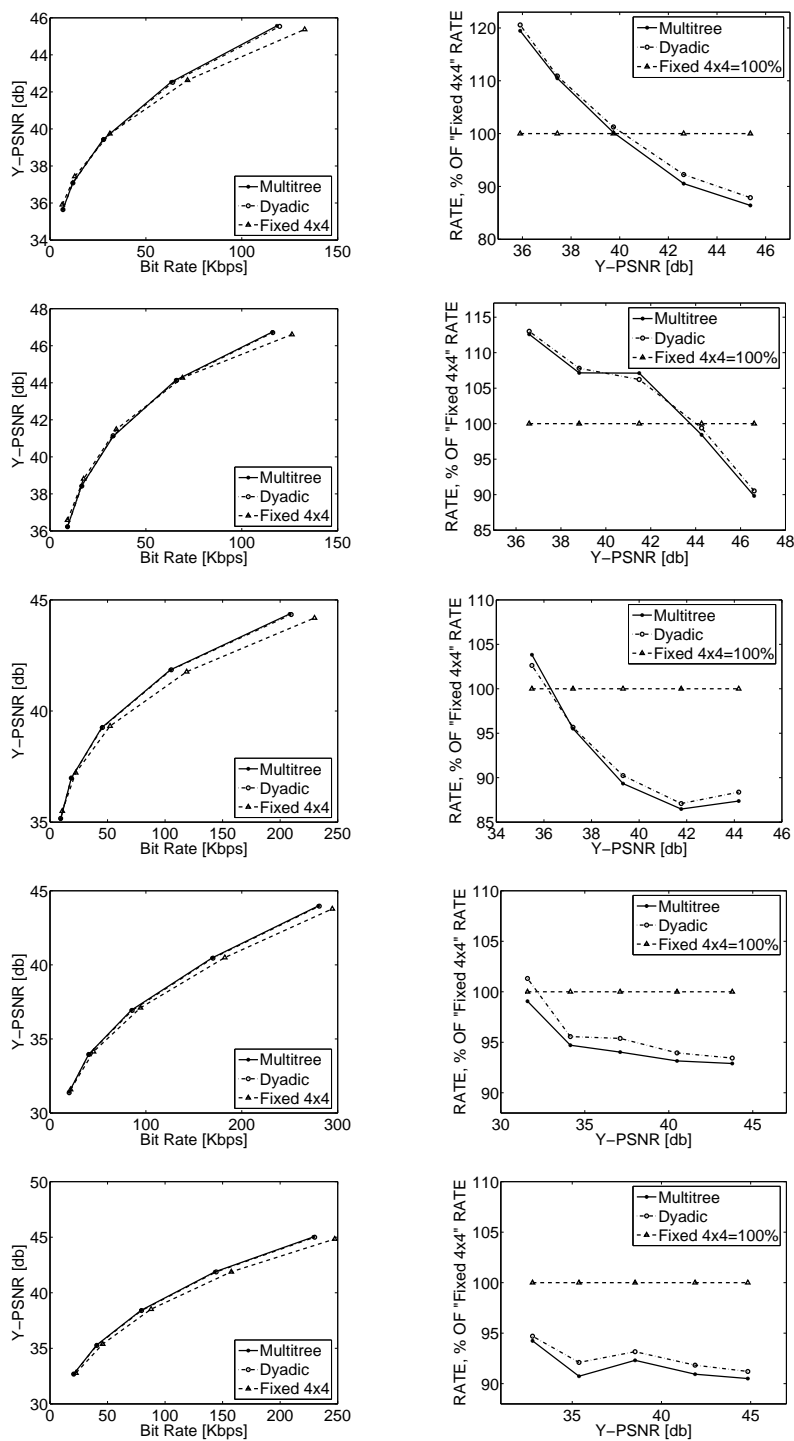


Fig. 8. The rate-distortion curves for compression with motion compensation using the H.264/AVC tiling scheme in conjunction with: fixed 4×4 transform (dashed lines with triangles), proposed optimal dyadic transform tiling (dashdot), and proposed optimal multitree transform tiling (solid). The right column shows bit rates as percentages of the bit rate for the baseline algorithm. The results are shown for frames 100 through 210 of “mother and daughter” (top), “salesman” (second row), “highway” (third row), “foreman” (fourth row), and “carphone” (bottom).

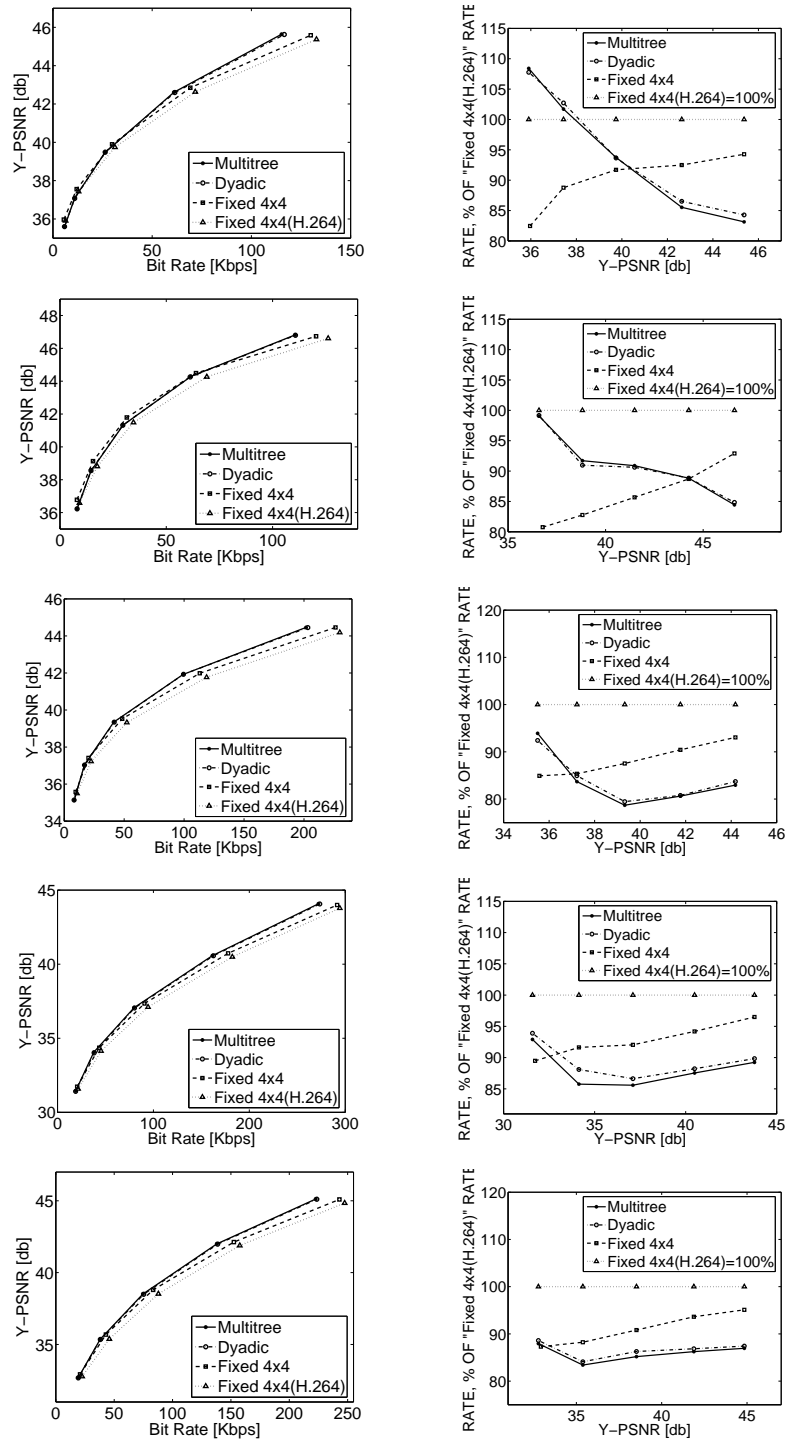


Fig. 9. The rate-distortion curves for compression with motion compensation using the dyadic tiling scheme in conjunction with: fixed 4×4 transform (dashed), proposed optimal dyadic transform tiling (dashdot), and proposed optimal multitree transform tiling (solid). Our baseline is an H.264-compliant scheme which uses fixed 4×4 transform (dotted lines with triangles). The right column shows bit rates as percentages of the bit rate for the baseline algorithm. The results are shown for frames 100 through 210 of “mother and daughter” (top), “salesman” (second row), “highway” (third row), “foreman” (fourth row), and “carphone” (bottom).

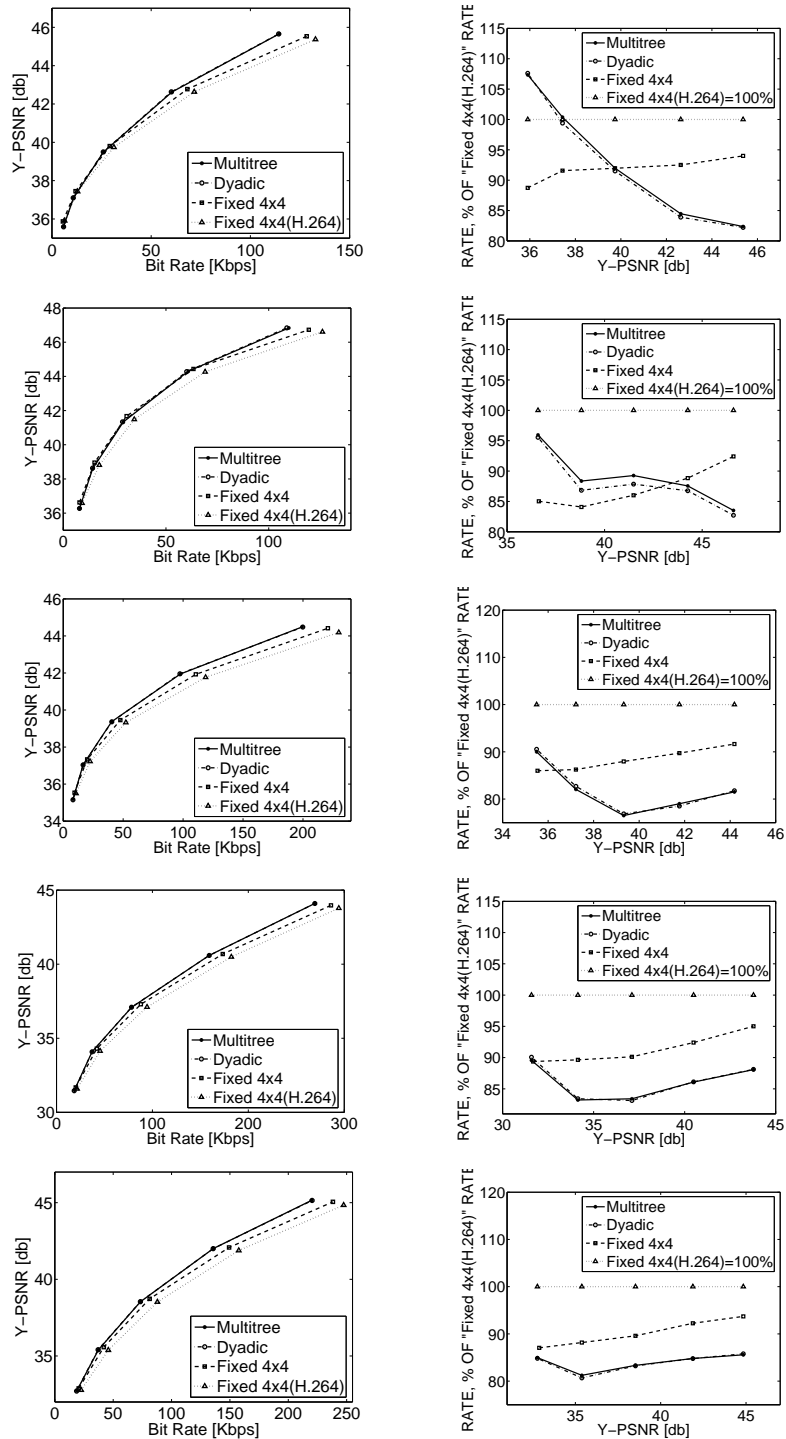


Fig. 10. The rate-distortion curves for compression with motion compensation using the multitree tiling scheme in conjunction with: fixed 4×4 transform (dashed), proposed optimal dyadic transform tiling (dashdot), and proposed optimal multitree transform tiling (solid). Our baseline is an H.264-compliant scheme which uses fixed 4×4 transform (dotted lines with triangles). The right column shows bit rates as percentages of the bit rate for the baseline algorithm. The results are shown for frames 100 through 210 of “mother and daughter” (top), “salesman” (second row), “highway” (third row), “foreman” (fourth row), and “carphone” (bottom).

column of the figure shows the rate-distortion curves for four schemes with the bit rates displayed as percentages of the baseline bit rate.

The combined use of optimal multitree tilings for motion compensation and transform results in up to 23% savings in bit rate as compared to the baseline. However, again, we do not observe significant difference between using the optimal multitree tiling and the optimal dyadic tiling for the transform stage: the differences between the bit rates for the two are within 2%. Just like in the previous experiment, we observe that, for sequences with little motion, the 4×4 integer transform is the best choice at low PSNRs.

Note that the shapes of the curves in the right columns of Figs. 10 and 9 are quite similar. Comparing the two sets of curves, it can be inferred that, when combined with an optimal transform tiling, the multitree motion compensation can save up to 4% in bit rate compared to the dyadic motion compensation.

VI. CONCLUSION

In this paper we proposed algorithms for the selection of the optimal tiling from large dictionaries of tilings and illustrated their application to both motion compensation and transform stages of H.264/AVC. Specifically, we constructed an implementation of an H.264-compliant baseline and replaced the motion compensation and transform tiling modules for the encoding of inter-frames with our proposed techniques.

We conclude that significant gains (up to 19% savings in bit rate) result from replacing the standard H.264/AVC motion compensation tiling with our proposed optimal dyadic tiling selection method. We observe that this replacement results in consistent gains (between 4% and 19% of bit rate) across a wide range of typical PSNRs and for all the five standard video sequences that we used in our testing. These gains come at a cost of only about 13% in running

time.

The more complex multitree tiling algorithm, when applied to the motion compensation stage, results in a 55% increase in running time, however, does not provide significant bit savings as compared to the dyadic algorithm. In fact, for sequences for low amounts of motion, the multitree algorithm works worse than the dyadic algorithm. This is due to a significantly larger overhead bit rate incurred by the multitree algorithm.

We also observe the similarity of the rate-distortion curves of the multitree and dyadic tiling algorithms when applied to the transform stage of the coder. We therefore conclude that the lower complexity of the dyadic algorithm makes it a more attractive choice. The use of the optimal dyadic transform tiling is beneficial for sequences with moderate and high amount of motion, although it may result in worse rate-distortion performance than the baseline H.264 method for sequences with low amounts of motion at low PSNRs. However, note that the present paper does not take into account the encoding of intra-frames. For such frames, our still image compression results from [8] suggest that the dyadic and multitree transform tilings can result in very substantial gains.

Our results suggest that the use of our dyadic tiling algorithm in the motion compensation stage is beneficial, in general. In addition, high-motion videos such as sports sequences or action movies will benefit from using our dyadic tiling algorithm in the transform stage. Low-motion videos such as newscasts will not benefit from our optimal tiling algorithms in the transform stage for the encoding of inter-frames; however, they may still benefit from these algorithms in the transform stage of the intra-frame encoding, as suggested by [8].

VII. ACKNOWLEDGMENT

The authors would like to thank Dr. Yan Huang, Prof. Minh Do, Prof. Charles Bouman, Prof. Edward Delp, and Prof. Fernando Pereira for stimulating discussions.

APPENDIX I

ESTIMATION OF THE NUMBER OF BITS FOR THE TRANSFORM COEFFICIENTS

In our optimal tiling algorithms for motion compensation, the cost functions both for our implementation of an H.264 baseline and for our proposed algorithms involve the estimates of the number of bits $R_{P_i}^{(c)}$ required to encode the quantized transform coefficients for the subblock P_i —see Eqs. (2), (3), (6), and (7). Our optimal transform tiling algorithms also involve the estimates of these rates, denoted $R_{P_i,j}$ in Eq. (9). This Appendix describes how these estimates are constructed.

We estimate $R_{P_i}^{(c)}$ using the first 99 frames of four sequences (*Mother and Daughter*, *Salesman*, *Carphone*, and *Foreman*). Note that these frames are disjoint from the frames that we use to test our algorithms in Section V. We first run our multitree algorithm on the first 99 frames of the sequence *Mother and Daughter*, using the cost of Eq. (7) and assuming that $R_{P_i}^{(c)} = 0$. We run our algorithm five times, with five different values of the quantization parameter (specifically, 20, 25, 30, 35, and 40). Since the quantization steps in our experiments are all larger than one, there are 511 possible different values for the quantized transform coefficients (i.e., all the integers from -255 to 255). We take all the quantized transform coefficients from these five experiments and construct their histogram with 511 bins, $h_k^{(1)}$, $k = -255, \dots, 255$. We then estimate the number of bits required to encode the quantized transform coefficient whose value is k , as $-\log h_k^{(1)}$. We let j be the index of a pixel in the subblock P_i , and we let $c(j)$ be the value of the quantized

transform coefficient at that pixel. We then form the following estimate of the rate $R_{P_i}^{(c)}$:

$$R_{P_i}^{(c,1)} = - \sum_{j \in P_i} \log h_{c(j)}^{(1)}.$$

Using this estimate in place of $R_{P_i}^{(c)}$ in the cost of Eq. (7), we rerun the five compression experiments with the same sequence, and the same five quantization parameters. Since the cost is different from the first run, the quantized transform coefficients may be different. We again form their histogram, call it $h_k^{(2)}$, and re-estimate the rate $R_{P_i}^{(c)}$ as

$$R_{P_i}^{(c,2)} = - \sum_{j \in P_i} \log h_{c(j)}^{(2)}.$$

We then rerun the five experiments again, to re-estimate the histogram one more time.

We repeat the same procedure for all four video sequences and three compression methods (H.264 baseline, dyadic algorithm, and multitree algorithm), to generate a total of twelve histograms. These histograms are combined into a single histogram h_k by averaging the twelve values for each bin k . Our final estimate of $R_{P_i}^{(c)}$ which we use in Eqs. (3), (6), and (7) is as follows:

$$R_{P_i}^{(c)} = - \sum_{j \in P_i} \log h_{c(j)}.$$

A similar procedure is used to produce the estimate of $R_{P_{i,j}}$ of Eq. (9) in the transform stage.

REFERENCES

- [1] N. N. Bennett. Fast algorithm for best anisotropic Walsh bases and relatives. *J. of Appl. and Comput. Harmonic Analysis*, 8(1):86–103, Jan. 2000.
- [2] D. L. Donoho. CART and best-ortho-basis: A connection. *Ann. Stat.*, 25(5):1870–1911, Oct. 1997.
- [3] D. L. Donoho. Wedgelets: Nearly minimax estimation of edges. *Ann. Statist.*, 27(3):859–897, Jun. 1999.
- [4] D. L. Donoho and I. M. Johnstone. Ideal denoising in an orthonormal basis chosen from a library of bases. *Comptes Rendus Acad. Sci., Ser. I*, 319:1317–1322, 1994.
- [5] K.-L. Hua, I. Pollak, and M. Comer. Optimal image tilings with application to video compression. In *Proc. ICIP*, Atlanta, Georgia, Oct. 2006.

- [6] K.-L. Hua, I. Pollak, and M. Comer. Optimal tilings for image and video compression. In *Proc. 40th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA., Nov. 2006.
- [7] Y. Huang and I. Pollak. MLC: A novel image coder based on multitree local cosine dictionaries. *IEEE Sig. Proc. Lett.*, 12(12):843-846, Dec. 2005.
- [8] Y. Huang, I. Pollak, M.N. Do, and C. A. Bouman. Fast search for best representations in multitree dictionaries. *IEEE Trans. Im. Proc.*, 15(7), Jul. 2006.
- [9] JVT reference software version 10.2.
iphome.hhi.de/suehring/tml/download/jm10.2.zip
- [10] H. Krim and J.-C. Pesquet. On the statistics of best bases criteria. In A. Antoniadis, editor, *Wavelets and statistics*, Lecture Notes in Statistics, pages 193–207. Springer-Verlag, 1995.
- [11] H. Krim, D. Tucker, S. Mallat, and D. Donoho. On denoising and best signal representation. *IEEE Trans. Inf. Th.*, 45(7):2225–2238, Nov. 1999.
- [12] R. Leonardi and M. Kunt. Adaptive split-and-merge for image analysis and coding. In *Proc. SPIE*, volume 594, pages 2–9, Dec. 1985.
- [13] E. Le Pennec and S. G. Mallat. Sparse geometric image representations with bandelets. *IEEE Trans. Im. Proc.*, 14(4):423–438, Apr. 2005.
- [14] M. Lindberg. *Two-Dimensional Adaptive Haar-Walsh Tilings*. Licentiat Thesis in Applied Mathematics, Åbo Akademi University, Åbo, Finland, Oct. 1999.
- [15] M. Lindberg and L. F. Villemoes. Image compression with adaptive Haar-Walsh tilings. In *Wavelet Applications in Signal and Image Processing VIII, Proc. SPIE 4119*, 2000.
- [16] A. Luthra, G.J. Sullivan, and T. Wiegand, Eds. *Special Issue on the H.264/AVC Video Coding Standard*. *IEEE Trans. Ckts. Syst. Vid. Tech.*, , 13(7), Jul. 2003.
- [17] F. Meyer. Image compression with adaptive local cosines: A comparative study. *IEEE Trans. Im. Proc.*, 11(6):616-629, Jun. 2002.
- [18] P. Moulin. Signal estimation using adapted tree-structured bases and the MDL principle. In *Proc. IEEE-SP Int. Symp. TFTS*, pages 141–143, Paris, Jun. 1996.
- [19] U. Ndili, R. D. Nowak, and M. A. T. Figueiredo. Coding theoretic approach to image segmentation. In *Proc. ICIP-2001*, Thessaloniki, Greece, Oct. 2001.
- [20] O.A. Niamut and R. Heusdens. Flexible frequency decompositions for cosine-modulated filter banks. In *Proc. ICASSP-2003*, Hong Kong, Apr. 2003.
- [21] O.A. Niamut and R. Heusdens. RD optimal time segmentations for the time-varying MDCT. In *Proceedings of European*

- Signal Processing Conference (Eusipco)*, Vienna, Austria, Sep. 6-10, 2004.
- [22] J.-C. Pesquet, H. Krim, D. Leporini, and E. Hamman. Bayesian approach to best basis selection. In *Proc. ICASSP-96*, pages 2634–2638, Atlanta, USA, May 1996.
- [23] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Trans. Im. Proc.*, 2(2):160–175, Apr. 1993.
- [24] R. Shukla, P. L. Dragotti, M. N. Do, and M. Vetterli. Rate-distortion optimized tree structured compression algorithms for piecewise smooth images. *IEEE Trans. Im. Proc.*, 14(3):343–359, Mar. 2005.
- [25] G. J. Sullivan and R. L. Baker. Efficient quadtree coding of images and video. *IEEE Trans. Im. Proc.*, 3(3):327–331, May 1994.
- [26] J. Vaisey and A. Gersho. Image compression with variable block size segmentation. *IEEE Trans. Sig. Proc.*, 40(8):2040–2060, Aug. 1992.
- [27] M. B. Wakin, J. K. Romberg, H. Choi, and R. G. Baraniuk. Rate-distortion optimized image compression using wedgelets. In *Proceedings of ICIP-2002*, Rochester, New York, Sep. 2002. u
- [28] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G.J. Sullivan. Rate-constrained coder control and comparison of video coding standards. *IEEE Trans. Ckts. Syst. Vid. Tech.*, 13(7):688–703, Jul. 2003.
- [29] M. Wien. Variable block-size transforms for H.264/AVC. *IEEE Trans. Ckts. Syst. Vid. Tech.*, 13(7):604–613, Jul. 2003.
- [30] R. M. Willett and R. D. Nowak. Platelets: a multiscale approach for recovering edges and surfaces in photon-limited medical imaging. *IEEE Trans. Medical Imaging*, 22(3):332–350, Mar. 2003.