

Video Coding Using Spatially Varying Transform

Cixun Zhang, *Student Member, IEEE*, Kemal Ugur, Jani Lainema, Antti Hallapuro, and Moncef Gabbouj, *Fellow, IEEE*

Abstract—In this paper, a novel algorithm called spatially varying transform (SVT) is proposed to improve the coding efficiency of video coders. SVT enables video coders to vary the position of the transform block, unlike state-of-art video codecs where the position of the transform block is fixed. In addition to changing the position of the transform block, the size of the transform can also be varied within the SVT framework, to better localize the prediction error so that the underlying correlations are better exploited. It is shown in this paper that by varying the position of the transform block and its size, characteristics of prediction error are better localized, and the coding efficiency is thus improved. The proposed algorithm is implemented and studied in the H.264/AVC framework. We show that the proposed algorithm achieves 5.85% bitrate reduction compared to H.264/AVC on average over a wide range of test set. Gains become more significant at medium to high bitrates for most tested sequences and the bitrate reduction may reach 13.50%, which makes the proposed algorithm very suitable for future video coding solutions focusing on high fidelity video applications. The gain in coding efficiency is achieved with a similar decoding complexity which makes the proposed algorithm easy to be incorporated in video codecs. However, the encoding complexity of SVT can be relatively high because of the need to perform a number of rate distortion optimization (RDO) steps to select the best location parameter (LP), which indicates the position of the transform. In this paper, a novel low complexity algorithm is also proposed, operating on a macroblock and a block level, to reduce the encoding complexity of SVT. Experimental results show that the proposed low complexity algorithm can reduce the number of LPs to be tested in RDO by about 80% with only a marginal penalty in the coding efficiency.

Index Terms—H.264/AVC, spatially varying transform (SVT), transform, variable block-size transforms (VBT), video coding.

I. INTRODUCTION

H.264/AVC is the latest international video coding standard which provides up to 50% gain in coding efficiency compared to previous standards. However, this is achieved at the cost of both increased encoding and decoding complexity. It is estimated in [1] that the encoder complexity increases

by more than one order of magnitude between Moving Picture Experts Group (MPEG)-4 Part 2 (Simple Profile) and H.264/AVC (Main Profile) and by a factor of 2 for the decoder. For mobile video services (video telephony, mobile TV, and others) and handheld consumer electronics (digital still cameras, camcorders, and others), additional complexity of H.264/AVC becomes an issue due to the limited resources of these devices. On the other hand, as display resolutions and available bandwidth/storage increase rapidly, high definition (HD) video is becoming more popular and commonly used, making the implementation of video codecs even more challenging.

To better satisfy the requirements of increased usage of HD video in resource-constrained applications, two key issues should be addressed: coding efficiency and implementation complexity. In this paper, we propose a novel algorithm, namely, spatially varying transform (SVT), which provides coding efficiency gains over H.264/AVC. The concept of SVT was first introduced by the authors in [2] and later extended in [3]. The technique is developed and studied mainly for coding HD resolution video, but also proved to be useful for coding lower resolution video. The motivations leading to the development of SVT are two fold:

- 1) The typical block-based transform design in most existing video coding standards does not align the underlying transform with the possible edge location. In this case, the coding efficiency decreases. In [4], directional discrete cosine transforms (DCTs) is proposed to improve the efficiency of transform coding for directional edges. However, efficient coding of horizontal, vertical, and nondirectional edges was not addressed, whereas typically, in natural video sequences, such edges are more common than directional edges. This is the main reason why only very marginal overall gain has been achieved, as stated in [4]. Some toy examples are given below in (1)–(3), which represent a vertical edge, a nondirectional edge, and a directional edge, respectively, where \mathbf{E}_1 , \mathbf{E}_3 , and \mathbf{E}_5 are the original prediction error matrices, \mathbf{E}_2 , \mathbf{E}_4 , and \mathbf{E}_6 are the corresponding prediction error matrices after we align the transform (by shifting the transform horizontally and vertically) to the edge location, and \mathbf{C}_i ($i = 1, 2, 3, 4, 5, 6$) is the corresponding transform coefficient matrix. Fig. 1 shows the energy distribution of the transform coefficients, where the x -axis denotes the sum of x and y components of the coefficients in the transform coefficient matrix, and the y -axis denotes the percentage of the energy of the transform coefficients.

Manuscript received April 16, 2009; revised September 29, 2009, February 1, 2010 and May 10, 2010; accepted September 11, 2010. Date of current version March 2, 2011. This work was supported by the Academy of Finland, Application 129657, Finnish Program for Centers of Excellence in Research 2006–2011. This paper was recommended by Associate Editor C.-W. Lin.

C. Zhang and M. Gabbouj are with the Tampere University of Technology, FI-33720 Tampere, Finland (e-mail: cixun.zhang@tut.fi; moncef.gabbouj@tut.fi).

K. Ugur, J. Lainema, and A. Hallapuro are with Nokia Research Center, FI-33720 Tampere, Finland (e-mail: kemal.ugur@nokia.com; jani.lainema@nokia.com; antti.hallapuro@nokia.com).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCSVT.2011.2105595

We can see that by aligning the transform to the edge location, more energy in the transform domain concentrates in the low frequency part and this facilitates the entropy coding and improves the coding efficiency. It is interesting that this is true not only for horizontal, vertical, and nondirectional edges but also for directional edges as follows:

$$\begin{aligned}
 E_1 &= \begin{bmatrix} 0 & 2 & 1 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 2 & 1 & 0 \end{bmatrix} \xrightarrow{\text{after } 2-D \text{ DCT}} \\
 C_1 &= \begin{bmatrix} 3.0000 & 0.5412 & -3.0000 & -1.3066 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
 E_2 &= \begin{bmatrix} 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix} \xrightarrow{\text{after } 2-D \text{ DCT}} \\
 C_2 &= \begin{bmatrix} 3.0000 & 3.1543 & 1.0000 & -0.2242 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (1) \\
 E_3 &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{after } 2-D \text{ DCT}} \\
 C_3 &= \begin{bmatrix} 1.5000 & -0.1913 & -1.0000 & -0.4619 \\ 0.1913 & -0.1464 & 0.0793 & -0.1464 \\ -1.0000 & 0.4619 & 0.5000 & -0.1913 \\ 0.4619 & 0.8536 & -1.1152 & -0.8536 \end{bmatrix} \\
 E_4 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix} \xrightarrow{\text{after } 2-D \text{ DCT}} \\
 C_4 &= \begin{bmatrix} 1.5000 & 1.1152 & 0 & 0.0793 \\ -1.1152 & -0.8536 & 0.0793 & 0.1464 \\ 0 & 0.4619 & 0.5000 & -0.1913 \\ -0.0793 & -0.8536 & -1.1152 & -0.1464 \end{bmatrix} \quad (2) \\
 E_5 &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{after } 2-D \text{ DCT}} \\
 C_5 &= \begin{bmatrix} 2.5000 & 0.0793 & -2.0000 & -1.1152 \\ -0.0793 & 0.3536 & -0.1913 & -0.3536 \\ -2.0000 & 0.1913 & 1.5000 & 0.4619 \\ 1.1152 & -0.3536 & -0.4619 & -0.3536 \end{bmatrix} \\
 E_6 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{after } 2-D \text{ DCT}} \\
 C_6 &= \begin{bmatrix} 2.5000 & 2.2304 & 0.5000 & 0.1585 \\ -0.0793 & 0.2500 & 0.4619 & 0.1036 \\ -2.0000 & -1.5772 & 0 & 0.1121 \\ 1.1152 & 0.6036 & -0.1913 & 0.2500 \end{bmatrix} \quad (3)
 \end{aligned}$$

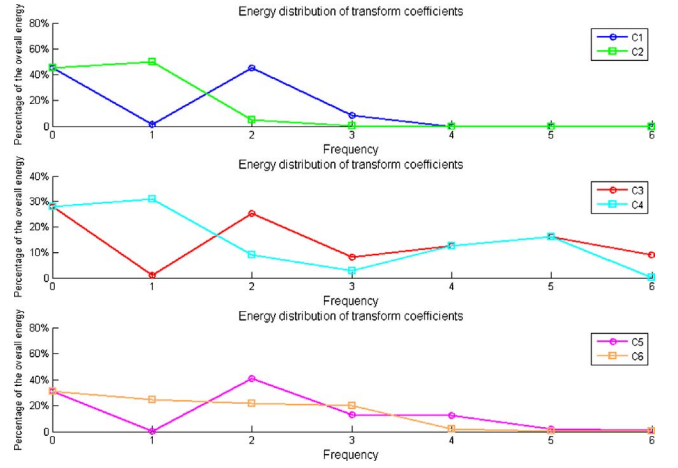


Fig. 1. Energy distribution of the transform coefficients before and after shifting the transform.

- 2) Coding the entire prediction error signal may not be the best in terms of rate distortion (RD) tradeoff. An example is the SKIP mode in H.264/AVC [5], which does not code the prediction error at all. We note that this fact is useful in improving the coding efficiency especially for HD video coding since, generally, a certain image area in HD video sequences contains less detail than in lower resolution video sequences, and the prediction error tends to have lower energy and more noise like after better prediction. More importantly, as we will see, utilizing this fact, our proposed algorithm can be implemented elegantly, e.g., on a macroblock basis, without boundary problem.

Both of these two factors contribute to the coding efficiency improvement achieved by the SVT [2]. The basic idea of SVT is that we do not restrict the transform coding inside regular block boundaries but adjust it to the characteristics of the prediction error. With this flexibility, we are able to achieve coding efficiency improvement by selecting and coding the best portion of the prediction error in terms of RD tradeoff. Generally, this can be done by searching inside a certain residual region after intra prediction or motion compensation, for a subregion and only coding this subregion. Note that, when the region refers to a macroblock, the proposed algorithm can be considered as a special SKIP mode, where *part* of the macroblock is “skipped.” Finally, the location parameter (LP) indicating the position of the subregion inside the region is coded into the bitstream if there are nonzero transform coefficients.

It is worth noting that the idea of shifting the transform, as used in the SVT, has been used in video and image denoising, where typically several denoised estimates produced for each shift of the transform are combined in a certain manner to produce the final denoised output [6]–[9]. It is often used in post-processing of a decoded image or a video frame since its complexity is generally too high to be incorporated in an image or video decoder, for instance, as an in-loop filter. Also, the boundary problem of this algorithm has not received much attention, and the performance tends to become worse at the

boundary and when it is applied to a smaller area such as a macroblock rather than a whole image or a video frame. By contrast, our proposed algorithm has no boundary problem and can be applied elegantly, e.g., on a macroblock basis. Also, using the proposed algorithm, there is no complexity penalty to the decoder since the LP of the subregion inside the region is coded in the bitstream and the decoder can simply use this information to reconstruct the region. For these reasons, the proposed algorithm can be easily incorporated in an image or a video decoder. In this case, various denoising algorithms [6]–[9] may still be efficiently used as post-processing algorithms.

In this paper, the proposed algorithm is studied and implemented in H.264/AVC framework. We show that 5.85% bitrate reduction is achieved compared to H.264/AVC on average over a wide range of test sets. Gains become more significant at medium to high bitrates for most tested sequences and the bitrate reduction can reach 13.50%, which makes the proposed algorithm very suitable for future video coding solutions focusing on high fidelity video applications.

The main drawback of the proposed algorithm is that the encoding complexity is higher mainly due to the brute force search process to select the best LP for the coded subregion. Nevertheless, the additional encoding complexity of the proposed algorithm is acceptable, for offline encoding application cases. For other cases with strict requirement of encoding complexity, a fast algorithm should be used. For such applications, we propose a fast algorithm operating on macroblock and block level to reduce the encoding complexity of SVT [10]. The proposed fast algorithm first skips testing SVT for macroblock modes for which SVT is unlikely to be useful by examining the RD cost of macroblock modes without SVT on a macroblock level. For the remaining macroblocks that SVT may be useful, the proposed fast algorithm selects available candidate LP based on motion difference and utilizes a hierarchical search algorithm to select the best available candidate LP at the block level. Experimental results show that the proposed fast algorithm can reduce the number of candidate LPs that need to be tested in search process by about 80% with only a marginal penalty in the coding efficiency.

The remainder of this paper is organized as follows. The proposed algorithm is introduced in Section II and its integration into H.264/AVC framework is described in Section III. The fast algorithm is introduced in Section IV. Experimental results are given in Section V. Section VI concludes this paper.

II. SVT

Transform coding is widely used in video coding standards to decorrelate the prediction error and achieve high compression rates. Typically, in video coding, a transform is applied to the prediction error at fixed locations. However, this has several drawbacks that may hurt the coding efficiency and decrease the visual quality. First of all, if the localized prediction error at a fixed location has a structure that is not suitable for the underlying transform, many high frequency transform coefficients will be generated thus requiring a large number of bits to code. Consequently, the coding efficiency decreases. Moreover, notorious visual artifacts such as ring-

ing may appear when these high frequency coefficients get quantized. In this paper, SVT is proposed to reduce these drawbacks of transform coding. The basic idea of SVT is that the transform coding is not restricted inside regular block boundary but can be applied to a portion of the prediction error according to the characteristics of the prediction error.

In this paper, we select and code a single block inside a macroblock when applying SVT. Extension to multiple blocks is straightforward. The selection is due to the fact that coding on a macroblock basis reduces complexity, delay, and throughput, which facilitate the implementation especially for hardware. This means that the position and shape of the transform block within a macroblock is variable, and information about its shape and location is signaled to the decoder, when there are nonzero transform coefficients. However, it should be noted that there is no restriction on the size and shape of a “subregion” and a “region” and thus the proposed idea could easily be extended to cover arbitrary sizes and shapes. For instance, SVT can be applied to an extended macroblock [11] of size larger than 16×16 . Directional SVTs with a directional oriented block selected and coded with a corresponding directional transform [4] can also be used.

In the following sub-sections, we further discuss three key issues of SVT in more detail: selection of SVT block-size, selection and coding of candidate LP, and filtering of SVT block boundaries.

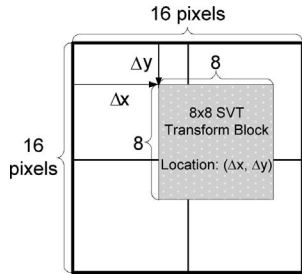
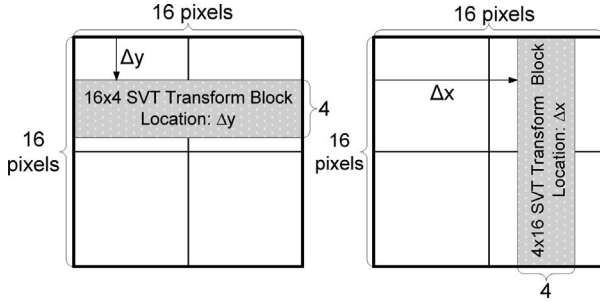
A. Selection of SVT Block-Size

In this paper, an $M \times N$ SVT is applied on a selected $M \times N$ block inside a macroblock of size 16×16 , and only this $M \times N$ block is coded. It is easy to see that there are in total $(17 - M) \times (17 - N)$ possible LPs if the $M \times N$ block is coded by an $M \times N$ transform. The following factors are taken into account when choosing suitable M and N .

- 1) Generally, larger M and N will result in fewer possible LPs and less overhead, and vice versa.
- 2) Also, larger M and N will result in lower distortion of the reconstructed macroblock but need more bits in coding the transform coefficients, and vice versa.
- 3) Finally, a larger block-size transform is more suitable in coding relatively flat areas or non-sharp edges, without introducing coding artifacts such as ringing, while a smaller block-size transform is more suitable in coding areas with details or sharp edges.

To facilitate the transform design, we can further assume $M = 2^m$, $N = 2^n$, where m and n are integers ranging from 0 to 4 inclusive in our context.¹ In this paper, we use 8×8 SVT, 16×4 SVT, and 4×16 SVT, which are illustrated in Figs. 2 and 3, respectively, to show the effectiveness of the proposed algorithm. Nevertheless, it should be noted that, for different sequences with different characteristics, other block-size SVT can be more suitable in terms of coding efficiency, complexity, and others. For instance, larger block-size SVT,

¹Note that SKIP mode is a special case when m and n are both equal to 0 and the macroblock partition is 16×16 (and the motion vector equals the predicted value in the context of H.264/AVC), and 16×16 transform is another special case when m and n are both equal to 4.

Fig. 2. Illustration of 8×8 SVT.Fig. 3. Illustration of 16×4 and 4×16 SVT.

such as 16×8 SVT or 8×16 SVT, may be more preferable in coding flat areas and when using a higher quantization parameter (QP). It is well established in video coding community that variable block-size transforms (VBT) can improve the coding efficiency [12]. Using VBT in the framework of SVT, namely, variable block-size SVT (VBSVT), we will show that the prediction error is better localized and the underlying correlations are better exploited; thus, the coding efficiency is improved compared to a fixed block-size SVT. However, when the number of different block-size SVT used becomes large, the coding gain of VBSVT tends to saturate since more overhead bits are needed to code the LPs. In this paper, we adaptively choose among 8×8 SVT, 16×4 SVT, and 4×16 SVT in VBSVT. It may also prove beneficial to choose different block-size SVT at different QPs and different bitrates. It is also possible and beneficial for the encoder to code SVT parameters, such as M , N (or m , n), in the bitstream, and transmit them in the slice header in H.264/AVC framework.

The transforms used for 8×8 , 16×4 , and 4×16 SVTs are described in the following. In general, the separable forward and inverse 2-D transform of a 2-D signal can be written, respectively, as follows:

$$\mathbf{C} = \mathbf{T}_v \cdot \mathbf{X} \cdot \mathbf{T}_h^t \quad (4)$$

$$\mathbf{X}_r = \mathbf{T}_v^t \cdot \mathbf{C} \cdot \mathbf{T}_h \quad (5)$$

where \mathbf{X} denotes a matrix representing $M \times N$ pixel block, \mathbf{C} is the transform coefficient matrix, and \mathbf{X}_r denotes a matrix representing the reconstructed signal block. \mathbf{T}_v and \mathbf{T}_h are the $M \times M$ and $N \times N$ transform kernels in the vertical and the horizontal direction, respectively. The superscript “ t ” denotes matrix transposition. For 8×8 SVT, we simply reuse the 8×8 transform kernel in H.264/AVC [5]. For 16×4 SVT and 4×16 SVT, a 4×4 and a 16×16 transform kernels need

to be specified. Here, we use the 4×4 transform kernel in H.264/AVC [5], [13] and the 16×16 transform kernel in [14] because it is simple and can reuse the butterfly structure of the existing 8×8 transform in H.264/AVC. It is noted that different 16×16 transform kernels could be used, which, however, is not the main focus of our proposed algorithm. In all cases, normal zig-zag scan (for frame-based coding which is used in our experiments) is used to represent the transform coefficients as input symbols to the entropy coding. Different scan patterns were also tested but not much gain in coding efficiency was achieved.

The block-size of SVT for luma component can be used to derive the corresponding block-size of SVT for the corresponding chroma components. Take 4:2:0 chroma format as an example, if $M \times N$ SVT is used for the luma component, then $M/2 \times N/2$ SVT can be used for chroma components.

B. Selection and Coding of Candidate LPs

When there are nonzero transform coefficients of the selected SVT block, its location inside the macroblock needs to be coded and transmitted to the decoder. For 8×8 SVT, as shown in Fig. 2, the location of the selected 8×8 block inside the current macroblock can be denoted by $(\Delta x, \Delta y)$ which can be selected from the set $\Phi_{8 \times 8} = \{(\Delta x, \Delta y), \Delta x, \Delta y = 0, \dots, 8\}$. There are in total 81 candidates for 8×8 SVT. For 16×4 SVT and 4×16 SVT, as shown in Fig. 3, the locations of the selected 16×4 and 4×16 block inside the current macroblock can be denoted as Δy and Δx , respectively, which can be selected from the set $\Phi_{16 \times 4} = \{\Delta y, \Delta y = 0, \dots, 12\}$ and $\Phi_{4 \times 16} = \{\Delta x, \Delta x = 0, \dots, 12\}$, respectively. There are in total 26 candidates for 16×4 SVT and 4×16 SVT.

The “best” LP is then selected according to a given criterion. In this paper, rate distortion optimization (RDO) [15] is used to select the best LP in terms of RD tradeoff by minimizing the following:

$$J = D + \lambda \cdot R \quad (6)$$

where J is the RD cost of the selected configuration, D is the distortion, R is the bitrate, and λ is the Lagrangian multiplier. In our implementation, the reconstruction residual for the remaining part of the macroblock is simply set to be 0, but different values can be used and might be beneficial in certain cases (luminance change, and others). It may be useful to also utilize the reconstructed SVT block and neighboring macroblock to derive the reconstructed value for the remaining part of the macroblock.

The set of candidate LPs is also important since it directly affects the encoding complexity and the performance of SVT. Larger number of candidate LPs provides more room and is more robust for coding efficiency improvement for different sequences with different characteristics, but adds more encoding complexity and also more overhead. Experimentally, according to criterion (6), we choose to use $\Phi_{8 \times 8}' = \{(\Delta x, \Delta y), \Delta x = 0, \dots, 8, \Delta y = 0; \Delta x = 0, \dots, 8, \Delta y = 8; \Delta x = 0, \Delta y = 1, \dots, 7; \Delta x = 8, \Delta y = 1, \dots, 7\}$ for 8×8 SVT and $\Phi_{16 \times 4} = \{\Delta y, \Delta y = 0, \dots, 12\}$ and $\Phi_{4 \times 16} = \{\Delta x, \Delta x = 0, \dots, 12\}$ for 16×4 SVT and 4×16 SVT, which shows

good RD performance over a large test set [2]. There are in total 58 candidate LPs, and statistics show that the measured entropy of the LP index is 5.73 bits, for all test sequences used in the experiments. Accordingly, the LP index is represented by a 6-bit fixed length code in our implementation. More advanced representation methods can be designed to achieve a better compression performance. These LPs are shown to have a good tradeoff in coding efficiency and complexity for different test sequences. As the overhead bits to code the LPs become significant at low bitrates, it would be useful to choose different LPs at different QPs and different bitrates. Similarly, an indication of available LP can also be coded and transmitted in the slice header in H.264/AVC framework.

The LP for the luma component can be used to derive the corresponding LPs for chroma components. Consider the 4:2:0 chroma format as an example, LPs for chroma components can be derived from the LP for the luma component as follows:

$$\Delta x_C = (\Delta x_L + 1) \gg 1, \quad \Delta y_C = (\Delta y_L + 1) \gg 1 \quad (7)$$

where $\Delta x_L, \Delta y_L$ are the LPs for the luma component, while $\Delta x_C, \Delta y_C$ are the LPs for the chroma components.

C. Filtering of SVT Block Boundaries

Due to the coding (transform and quantization) of the selected SVT block, coding artifacts may appear around its boundary with the remaining non-coded part of the macroblock. A deblocking filter can be designed and applied to improve both the subjective and objective quality. An example in the framework of H.264/AVC will be described in detail in Section III-D.

III. IMPLEMENTING SVT IN THE H.264/AVC FRAMEWORK

The proposed technique is implemented and tested in the H.264/AVC framework. Fig. 4 shows the block diagram of the extended H.264/AVC encoder with SVT. The encoder first performs motion estimation and decides the optimal mode to be used, and then searches for the best LPs (illustrated as ‘‘SVT Search’’ in the diagram) if SVT is used for this macroblock. The encoder then calculates the RD cost for using SVT using (6), and if lower RD cost can be achieved by using SVT then that macroblock is selected to be coded with SVT. For macroblocks that use SVT, the LPs are coded and transmitted in the bitstream when there are nonzero transform coefficients of the selected SVT block. The LPs for the macroblocks that use SVT are decoded in the box marked as ‘‘SVT L.P. Decoding’’ in the diagram. However, we note that the normal criteria, e.g., sum of absolute differences or sum of squared differences, which is used in these encoding processes for macroblocks that do not use SVT, may not be optimal for macroblocks that use SVT. Better encoding algorithms are under study.

Several key parts of the H.264/AVC standard [5], for instance, macroblock types, coded block pattern (CBP), entropy coding, and deblocking, need to be adjusted. The proposed modifications aiming at good compatibility with H.264/AVC are described in the following sub-sections.

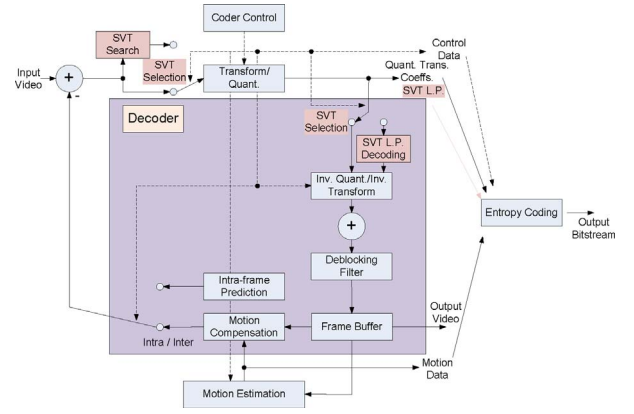


Fig. 4. Block diagram of the extended H.264 encoder with SVT.

TABLE I
EXTENDED MACROBLOCK TYPES FOR P SLICES IN H.264/AVC WITH SVT

mb_type	Name of mb_type
0	P_16 × 16
1	P_16 × 16_SVT
2	P_16 × 8
3	P_16 × 8_SVT
4	P_8 × 16
5	P_8 × 16_SVT
6	P_8 × 8
7	P_8 × 8_SVT
8	P_8 × 8ref0 ^a
9	P_8 × 8ref0_SVT
Inferred ^b	P_Skip

^aP_8 × 8ref0 is a macroblock mode defined in H.264/AVC with 8 × 8 macroblock partition and the reference index of this mode shall be inferred to be equal to 0 for all sub-macroblocks of the macroblock [5].

^bNo further data is present for P_Skip macroblock in the bitstream and it can be inferred [5].

A. Macroblock Type

We focus our study of SVT on coding the inter prediction error in P slices although the idea can be easily extended for I and B slices. Table I shows the extended macroblock types for P slices in H.264/AVC [5] (original intra macroblock types in H.264/AVC are not included). We use SVT for 16 × 16, 16 × 8, 8 × 16, and 8 × 8 macroblock partitions, based on the observation that SVT is used in a considerable portion when the same macroblock partition is concerned. This is shown in Fig. 5, where the statistics is based on all the test sequences we use and the overall test QP when context adaptive variable length coding (CAVLC) is used as the entropy coding. The macroblock type index is coded in a similar way as H.264/AVC. The sub-macroblock types are kept unchanged and therefore not shown in the table.

B. CBP

In this implementation, SVT is used for coding only the luma component. As shown in Figs. 2 and 3, since only one block is selected and coded in macroblocks that use SVT, we can either use 1 bit for luma CBP or jointly code it with chroma CBP as is done in H.264/AVC [5] when CAVLC is used as the entropy coding. However, in our experiments with

many test sequences, we found that luma CBP is often equal to 1 in high fidelity video coding, which is our main focus in this paper. Based on this observation, we restrict the new macroblock modes to have luma CBP equal to 1 and thus there is no need to code it. Chroma CBP is not changed and it is represented in the same way as in H.264/AVC [5]. An alternative way would be to infer the luma CBP according to QP or bitrate.

C. Entropy Coding

In H.264/AVC [5], when CAVLC is used as the entropy coding, a different coding table for the total number of nonzero transform coefficients and trailing ones of current block is selected depending on the characteristics (the number of nonzero transform coefficients) of the neighboring blocks. For macroblocks that use SVT, a fixed coding table is used for simplicity without loss of coding efficiency. Besides, we may also need to derive the information about the number of nonzero transform coefficients contained in each 4×4 luma block. When the selected SVT block aligns with the regular block boundaries, no special scheme is needed. Otherwise, the following two-step scheme with negligible complexity increase is used in our implementation as follows.

- Step 1: A 4×4 luma block is marked to have nonzero transform coefficients if it overlaps with a coded block that has nonzero transform coefficients in the selected SVT block, and marked not to have nonzero transform coefficients otherwise. This information may also be used in other process, e.g., deblocking.
- Step 2: The number of nonzero transform coefficients, nc_B , for each 4×4 block that is marked to have nonzero transform coefficients, is empirically set to

$$nc_B = \lfloor (nc_{MB} + \lfloor n_B/2 \rfloor) / n_B \rfloor \quad (8)$$

where nc_{MB} is the total number of nonzero transform coefficients in the current macroblock and n_B is the number of blocks marked to have nonzero transform coefficients, and $\lfloor \cdot \rfloor$ denotes the floor operator. In (8), we (approximately) distribute the total number of nonzero transform coefficients evenly to all the blocks that are marked as having nonzero transform coefficients. This is easy to calculate and it is shown experimentally to have similar coding efficiency compared to a more sophisticated scheme, for instance, distributing the total number of nonzero transform coefficients to all the blocks that are marked as having nonzero transform coefficients according to the overlapping size of the SVT block and each underlying 4×4 block.

Note that due to the floor operation in (8), a 4×4 block may be marked to have nonzero transform coefficients according to Step 1, but has zero nonzero transform coefficients according to (8) in Step 2.² In our implementation, when only the information about whether a block has nonzero transform

²Alternatively, one can set the number of nonzero transform coefficients to be 1 in this case. However, experimentally this does not show benefit over (8) in terms of coding efficiency.

TABLE II
INCREASE OF NUMBER OF EDGES TO BE CHECKED IN DEBLOCKING FOR LUMA COMPONENT WHEN SVT IS USED (CABAC, 720P)

Sequence	Increase of Number of Edges to Be Checked in Deblocking (%)
<i>BigShips</i>	32.8
<i>ShuttleStart</i>	31.3
<i>City</i>	39.5
<i>Night</i>	15.9
<i>Optis</i>	26.6
<i>Spincalendar</i>	27.6
<i>Cyclists</i>	20.2
<i>Breakness</i>	21.7
<i>Panslow</i>	33.6
<i>Sheriff</i>	23.6
<i>Sailormen</i>	27.2
Average	27.3

coefficients or not is needed (e.g., in deblocking), we use the result of Step 1; while when the information about how many nonzero transform coefficients a block has is needed, we use the result of Step 2 (e.g., in CAVLC).

The coding of the transform coefficients in CAVLC and context-based adaptive binary arithmetic coding (CABAC) is not changed, and thus no additional complexity is introduced.

D. Deblocking

For macroblocks that use SVT, the deblocking process in H.264/AVC [5] needs to be adjusted because the selected SVT block may not align with the regular block boundaries. Two cases are shown in Fig. 6: one is when 4×4 transform is not used as an optional transform and the other is when 4×4 transform is also used as an optional transform. The edges of the selected SVT block and the macroblock shown in Fig. 6 may be filtered. The filtering criteria and process of these edges are similar to those used in H.264/AVC [5], [16] by first deriving the necessary information of the blocks on both side of the boundary such as having nonzero transform coefficients or not (using Step 1 in Section III-C), motion difference, and others, with minor modifications.

The existing regular structure of deblocking in H.264/AVC is made less regular by SVT because the location of the edges of selected SVT block are not fixed. This could be an important issue especially in hardware implementation because it affects the data flow. Meanwhile, when SVT is used, the number of edges to be checked in deblocking would increase due to the fact that some of the macroblocks originally coded in SKIP mode will now be coded with SVT. According to our preliminary results, for instance, when 4×4 transform is also used as an optional transform, the number of edges to be checked in deblocking increases on average 27.3% for luma component over the 720p test set and all the tested QPs used in our experiment. Detailed results are shown in Table II.

IV. FAST ALGORITHMS FOR SVT

As described in Section III, we note that even though we do not change the motion estimation, sub-macroblock

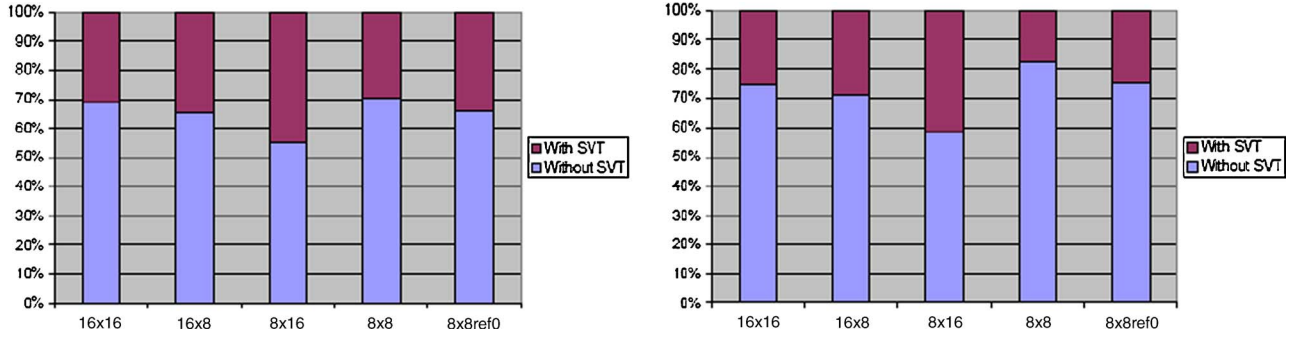


Fig. 5. Percentage of the use of SVT for different macroblock partitions when CAVLC is used. (a) Without using 4×4 transform. (b) Using 4×4 transform.

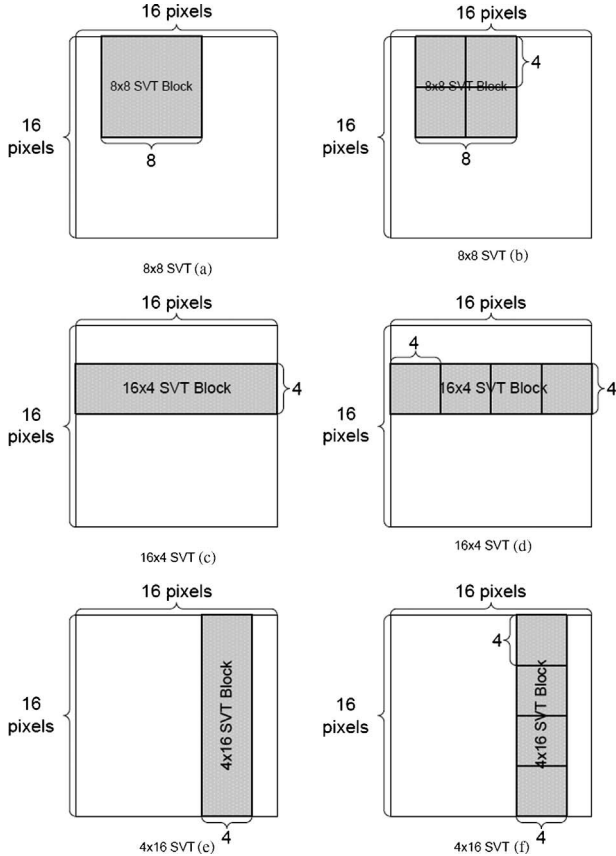


Fig. 6. Illustration of edges that may be filtered when SVT is used. Bold lines represent the edges of the SVT block. (a), (c), (e) When 4×4 transform is not used as an optional transform. (b), (d), (f) When 4×4 transform is also used as an optional transform.

partition decision process for the macroblocks that use SVT, the encoding complexity of SVT is higher due to the brute force search process in RDO. For example, in case of VBSVT described in Section II, there are a total of 58 candidate LPs for one macroblock mode and we need to conduct RDO for each candidate LP to select the best one. Note that typically we need to conduct transform, quantization, entropy coding, inverse transform, inverse quantization, and others to calculate the RD cost in (6) and this complexity is high.³ Thus, for application

³Of course, many fast algorithms can be used to reduce the complexity of transform, quantization, entropy coding, and in estimating the RD cost.

cases that have strict requirement of encoding complexity, fast algorithm of SVT should be developed and used.

In this section, we propose a simple yet efficient fast algorithm operating on macroblock and block level to reduce the encoding complexity of SVT. The basic idea to reduce the encoding complexity of SVT is to reduce the number of candidate LPs tested in RDO. The proposed fast algorithm first skips testing SVT for macroblocks for which SVT is unlikely to be useful by examining the RD cost of macroblock modes without SVT on a macroblock level. For the remaining macroblocks that SVT may be useful, the proposed fast algorithm selects available candidate LPs based on the motion difference and utilizes a hierarchical search algorithm to select the best available candidate LP on a block level. The macroblock level and block-level algorithms are described in Sections IV-A and IV-B, respectively.

A. Macroblock Level Fast Algorithm

The basic idea of macroblock level fast algorithm is to skip testing SVT for macroblock modes for which SVT is unlikely to be useful. This is done by examining the RD cost of macroblock modes that do not use SVT and are already available prior to SVT coding. In the proposed implementation, SVT is only applied for the macroblock modes in RDO process, when the two criteria are met as follows:

$$\min(J_{\text{inter}}, J_{\text{skip}}) \leq J_{\text{intra}} \quad (9)$$

$$J_{\text{mode}} \leq \min(J_{\text{inter}}, J_{\text{skip}}) + th \quad (10)$$

where J_{inter} and J_{intra} are the minimum RD cost of available inter and intra macroblock modes in regular (without SVT) coding, respectively, J_{skip} is the RD cost of the SKIP coding mode, and J_{mode} is the RD cost of the current macroblock mode to be tested with SVT (e.g., if SVT is being tested for INTER_16x8 mode, then J_{mode} refers to RD cost of the regular INTER_16x8 without SVT). In (9), by comparing inter, intra, and SKIP modes, we assume that if the RD cost for intra modes is the lowest, then the probability for inter modes with SVT to have a lower RD cost will be very small, so there is no need to check SVT for this macroblock. The basis of this assumption is that intra modes are often used in coding detailed areas for which SVT is unlikely to be useful since part of the macroblock is not coded when it is used. In (10), we assume that if the RD cost for the current mode is much

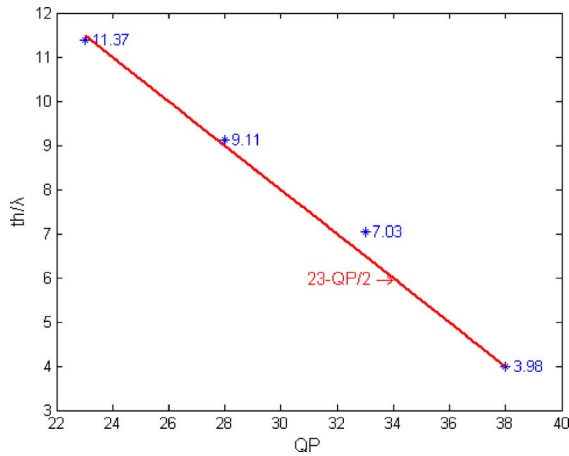


Fig. 7. Derivation of (11).

larger than the RD cost for the best mode, then the probability for this mode with SVT to have a lower RD cost is also expected to be very small. The threshold th in (10) represents the empirical upper limit of bitrate reduction when SVT is used assuming the reconstruction of macroblock remains the same when SVT is not used. It is calculated using (11), which is derived empirically using the following statistical analysis. We used four HD sequences, *Crew*, *Harbour*, *Raven*, and *Jets* as train set. The sequences are first encoded and the values J_{mode} , J_{inter} , J_{intra} , and J_{skip} are recorded for each mode in each macroblock that is coded with SVT. According to resulting costs, a threshold value th in (10) is calculated for each QP so that a certain number of the macroblocks satisfy the corresponding inequality in (9) and (10). This results for a different threshold value for each QP and using these threshold values (11) is derived by fitting a first order polynomial to the data. This simple derivation is also illustrated in Fig. 7 as follows:

$$th = \lambda \cdot \max(23 - QP/2, 0) \quad (11)$$

where λ is the Lagrangian multiplier used in (6) and the max function returns the maximum value of its arguments. As reflected in (11), the reason that the threshold value th , which represents the upper limit of bitrate reduction, tends to get lower when QP is larger, is mainly because details/edges are likely to be quantized and thus the benefit of SVT to align the transform to the underlying details/edges is compromised.

B. Block-Level Fast Algorithm

The proposed block-level fast algorithm includes two steps: the selection of available candidate LPs based on the motion difference in the first step and a hierarchical search algorithm in the second step. These two steps are described in detail below.

1) *Selection of Available Candidate LPs Based on Motion Difference*: SVT is used here for 16×16 , 16×8 , 8×16 , and 8×8 macroblock partitions, as described in Section III-A. One straightforward approach to reduce the encoding complexity is to restrict the transform block in a candidate SVT block to be

inside the same motion compensation block boundary, since applying the transform across motion compensation block boundaries is usually considered not efficient. In other words, it is typically assumed that edges exist at different motion compensation block boundaries that the transform should align to. This approach is simple because only macroblock partition information is used. However, some penalty was observed in coding efficiency for sequences with slow motion and rich detail, for instance, *Preakness* and *Night*. This is because prediction from different motion compensation blocks is not the major reason for causing blocking effect [16] which is likely to degrade the efficiency of the transform coding. Actually, it may not even cause blocking effect if the motion difference of different motion compensation blocks is small enough and/or the prediction is good.⁴ In this case, edges do not necessarily appear at different motion compensation block boundaries where the transform should align to and SVT would be still useful to improve coding efficiency. Taking this into account, we use a more general approach in this paper, by skipping the testing of candidate LPs when and only when the transform block(s) in the SVT block at that position overlaps with different motion compensation blocks in which the motion difference is significant according to a certain criterion. In order to measure the motion difference, we use a similar method to the one used in deriving the boundary strength parameter in deblocking filter of H.264/AVC [5], [16]. Specifically, in our implementation, we skip testing a candidate LP and mark it unavailable if at least one of the following conditions is true.

- If the transform applied to the SVT block at that position overlaps with at least two neighboring motion compensation blocks and the motion vectors of these blocks are larger than or equal to a predefined threshold which is set to be one integer pixel in this paper.
- If the transform applied to the SVT block at that position overlaps with at least two neighboring motion compensation blocks and the reference frames of these two neighboring blocks are different.

Since the number of available candidate LPs varies from one macroblock to another and the information to derive the available candidate LPs can be obtained both by the encoder and the decoder, the index of the selected LP is coded as follows. Assume there are N ($N > 0$) available candidate LPs and the index of the selected candidate is n ($0 \leq n < N$), then it is coded as follows:

$$\begin{cases} V = n, L = \lfloor \log_2 N \rfloor + 1, & \text{if } n < 2(N - 2^{\lfloor \log_2 N \rfloor}) \\ V = n - (N - 2^{\lfloor \log_2 N \rfloor}), L = \lfloor \log_2 N \rfloor, & \text{otherwise} \end{cases} \quad (12)$$

where V represents the binary value of the coded bit string and L represents the number of bits coded. This is a near fixed-length code which is chosen based on the observation that all available candidate LPs are similarly likely to be used.

⁴We would also like to point out that it is shown in a recent paper that “despite the transform residual block being composed of two parts that were predicted from different areas of the reference picture, correlation within mixed blocks is very similar to that of normal blocks. The DCT is only marginally suboptimal with respect to Karhunen-Loeve transform” [17].

This approach shows a stable coding efficiency for sequences with different characteristics and achieves similar gain over a wide range of test sets compared to the original algorithm without using the information of motion difference. Finally, we note that the additional complexity introduced by this approach is marginal when it is carefully implemented because: 1) the decision is only conducted when necessary and can be skipped for macroblock modes with 16×16 partition and some LPs, e.g., the LP (0, 0) for 8×8 SVT; 2) the decision is simple and only uses the motion vector and the reference frame information; and 3) generally several candidate LPs representing spatially consecutive blocks can be marked available or unavailable at the same time in one decision.

2) *Hierarchical Search Algorithm*: The basic idea of the hierarchical search algorithm is similar to that of the motion estimation algorithm, i.e., to first find the best LP in a relatively coarse resolution and then refine the results in a finer resolution. This assumes that the prediction error structure remains similar in the coarse resolution as in the original resolution, which is often the case for a macroblock of size 16×16 especially in HD video material. In our implementation, we define the candidate LP in coarser resolution as *key candidate LP*, which are marked as squares in Fig. 8. Other candidate LPs are also called as *non-key candidate LP*, which are marked as circles in Fig. 8. The hierarchical search algorithm can be summarized as follows.

- a) Let Φ_1 denote the set of all available key candidate LPs. Select the best one in Φ_1 with the lowest RD cost and let Φ_2 denote its available neighboring candidate LPs which are marked as triangles in Fig. 8.
- b) The key LPs are divided into 14 LP zones as shown in Fig. 8. Select the best LP zone which is available and has the lowest RD cost. A LP zone is available if and only if all three key candidate LPs in that zone are available and the RD cost of an LP zone is defined to be the sum of the RD costs of the three key candidate LPs in that zone. Let Φ_3 denote the additional available non-key candidate LPs which are inside the best LP zone (marked as stars in Fig. 8).
- c) Select the best LP which has the lowest RD cost among all the candidates in Φ_1 , Φ_2 , and Φ_3 .

More sophisticated algorithms using the same basic idea can be designed, for instance, by using different definitions of key position and zone, and/or examining different number of good key candidate LPs and zones.

V. EXPERIMENTAL RESULTS

We implemented the proposed algorithm VBSVT and its fast version with the proposed fast algorithm, which we denote as FVBSVT, in KTA1.8 reference software [18] in order to evaluate their effectiveness. Although our primary interest is in HD video coding, we also test the proposed algorithm in lower resolution [common intermediate format (CIF)] video coding. The most important coding parameters used in the experiments are listed as follows.

- 1) High Profile.

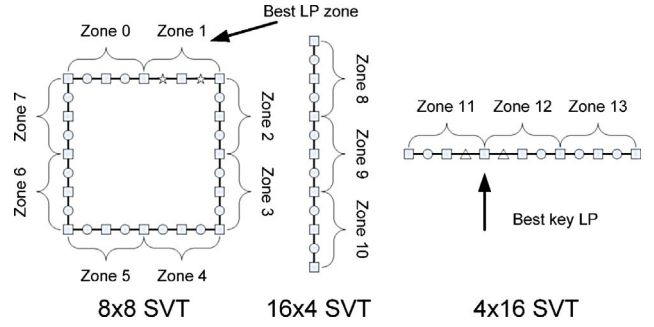


Fig. 8. Illustration of hierarchical search algorithm.

- 2) $QP_I = 22, 27, 32, 37$, $QP_P = QP_I + 1$, fixed QP is used and rate control is disabled according to the common conditions that are set up by ITU-T VCEG and ISO-IEC/MPEG community for coding efficiency experiments [19].
- 3) CAVLC/CABAC is used as the entropy coding.
- 4) Frame structure is IPPP, four reference frames.
- 5) Motion vector search range: $\pm 64/\pm 32$ pels for 720p/CIF sequences, resolution $\frac{1}{4}$ -pel.
- 6) RDO in the “high complexity mode.”

Two configurations are tested as follows.

- 1) *Low complexity configuration*: Motion estimation block sizes are 16×16 , 16×8 , 8×16 , 8×8 , and 4×4 transform is not used for macroblocks coded with regular transform or SVT. This represents a low complexity codec with most effective tools for HD video coding.
- 2) *High complexity configuration*: Motion estimation block sizes are 16×16 , 16×8 , 8×16 , 8×8 , 8×4 , 4×8 , 4×4 , and 4×4 transform is also used as an optional transform for both macroblocks coded with regular transform or SVT. This represents a high complexity codec with full usage of the tools provided in H.264/AVC.

We measure the average bitrate reduction (Δ BD-RATE) compared to H.264/AVC using Bjontegaard tool [20] for both low complexity and high complexity configurations. The Bjontegaard tool measures the numerical average in data rate or quality differences between two rate–distortion curves. This is a more compact way to present the performance comparison and required by MPEG and VCEG community. The results are shown in Tables III–VII. We also show the results of 8×8 SVT and $16 \times 4 + 4 \times 16$ SVT for comparison. As we can see, $16 \times 4 + 4 \times 16$ SVT performs better than 8×8 SVT while VBSVT performs best, both in low and high complexity configurations. VBSVT achieves on average 4.11% (up to 6.19%) bitrate reduction in low complexity configuration and on average 2.43% (up to 3.48%) bitrate reduction in high complexity configuration, respectively, when CAVLC is used, and achieves on average 5.85% (up to 7.92%) bitrate reduction in low complexity configuration and on average 4.40% (up to 6.25%) bitrate reduction in high complexity configuration, respectively, when CABAC is used, compared to H.264/AVC. The gain of VBSVT is lower in high complexity configuration, because the overhead to code the LP becomes more significant

TABLE III
 Δ BD-RATE COMPARED TO H.264/AVC (LOW COMPLEXITY CONFIGURATION) (CABAC, 720P)

Sequence	8×8 SVT (%)	$16 \times 4 + 4 \times 16$ SVT (%)	VBSVT (%)	FVBSVT (%)	Reduction of Candidate LPs (%)
<i>BigShips</i>	-4.27	-5.31	-6.22	-5.96	82.5
<i>ShuttleStart</i>	-4.09	-4.70	-5.76	-5.22	91.4
<i>City</i>	-6.98	-6.76	-7.92	-7.61	81.6
<i>Night</i>	-3.53	-4.86	-5.41	-4.87	84.7
<i>Optis</i>	-5.28	-5.17	-6.70	-5.95	83.3
<i>Spincalendar</i>	-3.24	-4.02	-4.87	-4.01	84.4
<i>Cyclists</i>	-4.56	-4.28	-5.68	-5.32	85.5
<i>Preakness</i>	-2.78	-3.23	-4.04	-3.34	78.7
<i>Panslow</i>	-4.15	-4.91	-5.62	-5.09	86.1
<i>Sheriff</i>	-4.46	-5.15	-5.93	-5.35	84.5
<i>Sailormen</i>	-3.88	-5.86	-6.22	-5.69	84.8
Average	-4.29	-4.93	-5.85	-5.31	84.3

TABLE IV
 Δ BD-RATE COMPARED TO H.264/AVC (LOW COMPLEXITY CONFIGURATION) (CABAC, CIF)

Sequence	8×8 SVT (%)	$16 \times 4 + 4 \times 16$ SVT (%)	VBSVT (%)	FVBSVT (%)	Reduction of Candidate LPs (%)
<i>Bus</i>	-2.27	-3.34	-3.87	-3.13	81.0
<i>Container</i>	-3.87	-4.27	-5.31	-4.79	84.4
<i>Foreman</i>	-2.61	-2.86	-3.52	-3.07	82.7
<i>Hall</i>	-4.23	-4.99	-5.98	-5.49	84.9
<i>Mobile</i>	-2.30	-2.73	-3.81	-2.86	81.3
<i>News</i>	-2.84	-3.98	-4.52	-3.81	86.7
<i>Paris</i>	-3.28	-4.81	-5.69	-4.32	85.8
<i>Tempete</i>	-2.91	-3.94	-4.60	-4.03	81.8
Average	-3.04	-3.87	-4.66	-3.94	83.6

TABLE V
 Δ BD-RATE COMPARED TO H.264/AVC (HIGH COMPLEXITY CONFIGURATION) (CABAC, 720P)

Sequence	8×8 SVT (%)	$16 \times 4 + 4 \times 16$ SVT (%)	VBSVT (%)	FVBSVT (%)	Reduction of Candidate LPs (%)
<i>BigShips</i>	-3.05	-3.62	-4.46	-4.34	80.9
<i>ShuttleStart</i>	-3.56	-4.02	-4.69	-4.33	90.9
<i>City</i>	-5.41	-4.92	-6.25	-5.65	80.0
<i>Night</i>	-2.46	-3.65	-4.00	-3.60	82.8
<i>Optis</i>	-4.20	-3.93	-5.36	-4.71	82.1
<i>Spincalendar</i>	-2.44	-2.76	-3.28	-2.96	82.2
<i>Cyclists</i>	-3.87	-3.24	-4.30	-4.24	83.8
<i>Preakness</i>	-1.22	-1.45	-2.00	-1.60	77.9
<i>Panslow</i>	-2.26	-2.60	-3.50	-3.02	84.0
<i>Sheriff</i>	-3.98	-4.25	-5.14	-4.63	82.8
<i>Sailormen</i>	-3.40	-5.07	-5.44	-5.19	83.2
Average	-3.26	-3.59	-4.40	-4.03	82.8

TABLE VI
 Δ BD-RATE COMPARED TO H.264/AVC (HIGH COMPLEXITY CONFIGURATION) (CABAC, CIF)

Sequence	8×8 SVT (%)	$16 \times 4 + 4 \times 16$ SVT (%)	VBSVT (%)	FVBSVT (%)	Reduction of Candidate LPs (%)
<i>Bus</i>	-1.33	-1.99	-2.32	-1.83	78.8
<i>Container</i>	-1.35	-1.42	-2.12	-1.80	83.0
<i>Foreman</i>	-1.58	-2.02	-2.51	-1.83	80.5
<i>Hall</i>	-2.79	-3.42	-4.01	-3.63	83.3
<i>Mobile</i>	-0.99	-1.23	-1.58	-1.22	79.3
<i>News</i>	-0.90	-1.99	-2.31	-1.81	85.8
<i>Paris</i>	-1.45	-1.79	-2.15	-1.69	84.5
<i>Tempete</i>	-1.79	-2.39	-2.62	-2.24	79.7
Average	-1.52	-2.03	-2.45	-2.01	81.9

TABLE VII
AVERAGE RESULTS OF Δ BD-RATE COMPARED TO H.264/AVC

Average Results	8×8 SVT (%)	$16 \times 4+4 \times 16$ SVT (%)	VBSVT (%)	FVBSVT (%)	Reduction of Candidate LPs (%)
Average (CAVLC, 720p, low complexity configuration)	-2.64	-3.65	-4.11	-3.69	83.4
Average (CAVLC, CIF, low complexity configuration)	-2.44	-3.68	-4.14	-3.55	83.6
Average (CAVLC, 720p, high complexity configuration)	-1.42	-2.33	-2.43	-2.34	81.1
Average (CAVLC, CIF, high complexity configuration)	-1.28	-1.88	-2.08	-1.72	81.4
Average (CABAC, 720p, low complexity configuration)	-4.29	-4.93	-5.85	-5.31	84.3
Average (CABAC, CIF, low complexity configuration)	-3.04	-3.87	-4.66	-3.94	83.6
Average (CABAC, 720p, high complexity configuration)	-3.26	-3.59	-4.40	-4.03	82.8
Average (CABAC, CIF, high complexity configuration)	-1.52	-2.03	-2.45	-2.01	81.9

TABLE VIII
PERCENTAGE OF 8×8 SVT AND $16 \times 4+4 \times 16$ SVT SELECTED IN VBSVT FOR DIFFERENT SEQUENCES (CABAC, 720P)

Sequence	Low Complexity Configuration		High Complexity Configuration	
	8×8 SVT (%)	$16 \times 4+4 \times 16$ SVT (%)	8×8 SVT (%)	$16 \times 4+4 \times 16$ SVT (%)
<i>BigShips</i>	51.4	48.6	54.4	45.6
<i>ShuttleStart</i>	48.9	51.1	52.8	47.2
<i>City</i>	50.1	49.9	53.0	47.0
<i>Night</i>	49.7	50.3	49.8	50.2
<i>Optis</i>	57.0	43.0	59.1	40.9
<i>Spincalendar</i>	44.8	55.2	53.7	46.3
<i>Cyclists</i>	52.3	47.7	50.0	50.0
<i>Preakness</i>	50.2	49.8	49.9	50.1
<i>Panslow</i>	49.4	50.6	55.2	44.8
<i>Sheriff</i>	54.3	45.7	51.2	48.8
<i>Sailormen</i>	48.3	51.7	46.4	53.6
Average	50.2	49.8	52.3	47.7

TABLE IX
PERCENTAGE OF 8×8 SVT AND $16 \times 4+4 \times 16$ SVT SELECTED IN VBSVT FOR DIFFERENT SEQUENCES (CABAC, CIF)

Sequence	Low Complexity Configuration		High Complexity Configuration	
	8×8 SVT (%)	$16 \times 4+4 \times 16$ SVT (%)	8×8 SVT (%)	$16 \times 4+4 \times 16$ SVT (%)
<i>Bus</i>	38.6	61.4	39.9	60.1
<i>Container</i>	54.1	45.9	56.0	44.0
<i>Foreman</i>	53.1	46.9	53.1	46.9
<i>Hall</i>	48.9	51.1	48.8	51.2
<i>Mobile</i>	49.2	50.8	47.2	52.8
<i>News</i>	46.4	53.6	44.6	55.4
<i>Paris</i>	45.2	54.8	44.6	55.4
<i>Tempete</i>	44.3	55.7	43.6	56.4
Average	47.5	52.5	47.2	52.8

and the 4×4 transform in H.264/AVC (and smaller motion estimation block sizes) is more efficient in certain cases. The gain of VBSVT is higher in HD video coding because typically less detail is contained in a certain image area in HD video sequences than in lower resolution video sequences and thus the non-coded part of the macroblock can be more safely skipped without introducing much degradation in reconstructed quality.

Fig. 9 shows the R–D curves for sequences *Night* and *Panslow*. We note that the gain of VBSVT comes at medium to high bitrates for many tested sequences and can be much more significant than the average gain over all bitrates reported above. This is true for most sequences we tested. Take *Panslow* sequence as an example, by using the performance evaluation tool provided in [21], we are able to show that VBSVT achieves 13.50% and 6.90% bitrate reduction at 37 dB compared to H.264/AVC, in low and high complexity configuration, respectively, when CAVLC is used.

We also measure the percentage of 8×8 SVT and $16 \times 4+4 \times 16$ SVT selected in VBSVT. The results are shown in Tables VIII–X. We can see that in VBSVT scheme, $16 \times 4+4 \times 16$ SVT are used generally more often than 8×8 SVT. However, 8×8 SVT is also used in a significant portion of the cases. This shows that different block-size SVT can be suitable for different situations and VBSVT would be a more preferable algorithm for coding prediction error with different characteristics than fixed block-size SVT.

In Figs. 10 and 11, we show the distribution of the macroblocks coded with SVT and corresponding SVT blocks in a prediction error frame for *Night* and *Sheriff* sequences, respectively. It can be observed that, as expected, SVT is mostly useful for macroblocks where residual signals of larger magnitude are likely to gather and locate at certain regions, which frequently happens, as also observed by other researchers [22]. However, when using RDO, the selection of the SVT depends on the distortion of the whole macroblock and

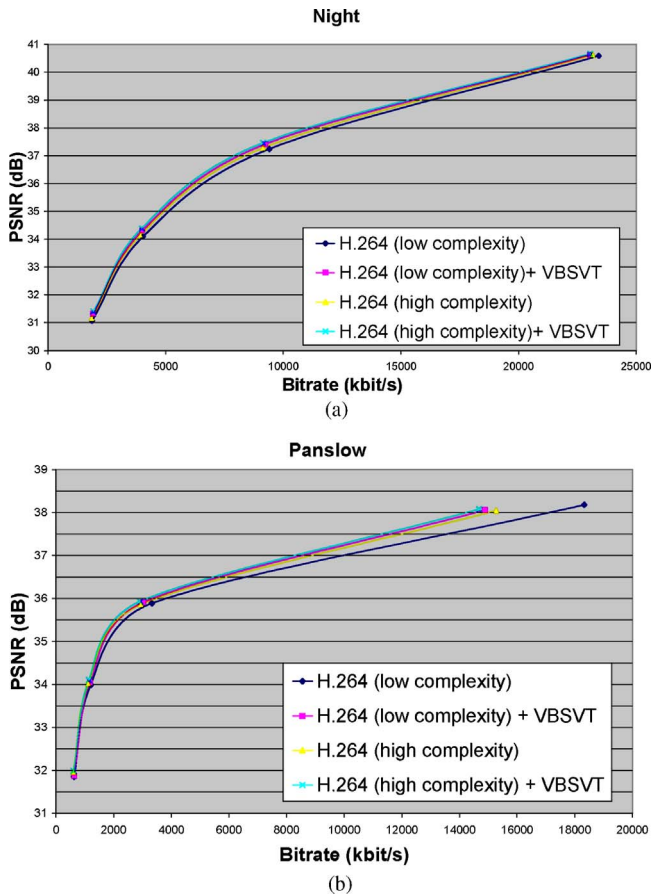


Fig. 9. R–D curves for (a) *Night* sequence (CABAC) and (b) *Panslow* sequence (CAVLC).

the coded bits of the selected SVT block, so it is also possible that SVT can also be chosen to be used for macroblocks with relatively evenly distributed residual signal of medium magnitude. Macroblocks with residual signal of low magnitude are often coded in SKIP mode.

We also view the reconstructed video sequences in order to analyze the impact of SVT on subjective quality. We conclude that using SVT does not introduce any special visual artifacts and the overall subjective quality remains similar, while the coded bits are reduced.

Similarly, we measure the average bitrate reduction (Δ BD-RATE) of FVBSVT compared to H.264/AVC. We also measure the average reduction of candidate LPs tested in RDO over all the tested QPs using FVBSVT. The results are also shown in Tables III–VII. We can see that with the proposed fast algorithm we can reduce by a little more than 80% the number of candidate LPs tested in RDO while retaining most of the coding efficiency, for both low and high complexity configurations. The reduction in high complexity configuration is slightly less than that in low complexity configuration because when the 4×4 transform is used, the number of candidate LPs is increased according to the selection method based on motion difference described in Section IV-B. This means that on average about 10–11 candidate LPs are tested in RDO (each RD cost calculation includes transform, quantization, entropy coding, inverse transform, inverse quantization, and others) when

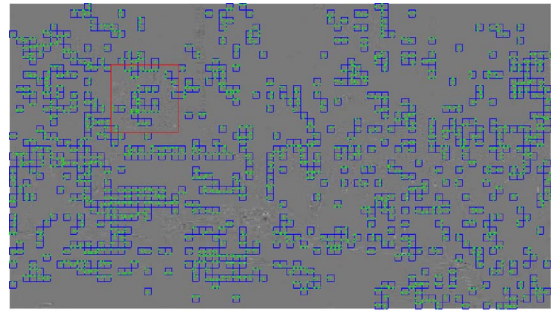


Fig. 10. Distribution of the macroblocks coded with SVT and corresponding SVT blocks for 32nd prediction error frame (*Night* sequence).

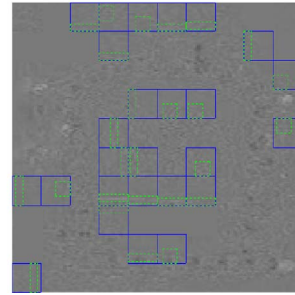


Fig. 11. Distribution of the macroblocks coded with SVT and corresponding SVT blocks for 65th prediction error frame (*Sheriff* sequence).

FVBSVT is used. We also measured the execution time of FVBSVT to estimate the computational complexity. Compared to H.264/AVC, the encoding and decoding time of FVBSVT increases on average 2.49% and 2.05%, respectively, over 720p test set and all the tested QPs in high complexity configuration. The encoding and decoding time increase vary for different test sequences and detailed results are shown in Table XI. The running time increase is marginal because only a relatively small part of the whole coding process is affected. The memory usage remains similar when SVT is used because it needs to store the number of nonzero transform coefficients, access the pixels

TABLE X
AVERAGE RESULTS OF PERCENTAGE OF 8×8 SVT AND $16 \times 4 + 4 \times 16$ SVT SELECTED IN VBSVT FOR DIFFERENT SEQUENCES

Average Results	Low Complexity Configuration		High Complexity Configuration	
	8×8 SVT (%)	$16 \times 4 + 4 \times 16$ SVT (%)	8×8 SVT (%)	$16 \times 4 + 4 \times 16$ SVT (%)
Average (CAVLC, 720p)	45.8	54.2	43.9	56.1
Average (CAVLC, CIF)	44.4	55.6	43.1	56.9
Average (CABAC, 720p)	50.2	49.8	52.3	47.7
Average (CABAC, CIF)	47.5	52.5	47.2	52.8

TABLE XI

ENCODING/DECODING TIME INCREASE OF FVBSVT COMPARED TO H.264/AVC (HIGH COMPLEXITY CONFIGURATION) (CABAC, 720P)

Sequence	Encoding Time Increase of FVBSVT (%)	Decoding Time Increase of FVBSVT (%)
<i>BigShips</i>	2.88	1.79
<i>ShuttleStart</i>	1.16	0.63
<i>City</i>	2.94	3.85
<i>Night</i>	2.62	2.08
<i>Optis</i>	2.18	2.29
<i>Spincalendar</i>	2.74	2.07
<i>Cyclists</i>	1.90	4.40
<i>Preakness</i>	3.71	1.89
<i>Panslow</i>	2.17	0.49
<i>Sheriff</i>	2.41	0.99
<i>Sailormen</i>	2.64	2.07
Average	2.49	2.05

in neighboring blocks in deblocking and store the motion vector and reference frame information etc, which is similar to the case when SVT is not used. The simulation PC we use is Intel(R) Core(TM)2 Quad CPU Q6600 @2.40 GHz 2.39 GHz, 2.00 GB of RAM, and the operating system is Microsoft Windows XP Professional Version 2002 Service Pack 2.

VI. CONCLUSION

In this paper, we proposed a novel algorithm, called SVT to improve the coding efficiency of video coders. The main idea of SVT is to vary the position of the transform block. In addition to changing the position of the transform block, the size of the transform can also be varied within the SVT framework. We showed in this paper that by varying the position of the transform block and its size, the characteristics of the prediction error is better localized, and the coding efficiency is improved. The proposed algorithm is studied and implemented in the H.264/AVC framework. We showed that the proposed algorithm achieves on average 5.85% bitrate reduction in a low complexity configuration which represents a low complexity codec with most effective tools for HD video coding which we focus on and on average 4.40% bitrate reduction in high complexity configuration, which represents a high complexity codec with full usage of the tools provided in the standard, respectively. Gains become more significant at medium to high bitrates for most tested sequences and the bitrate reduction may reach up to 13.50%, which makes the proposed algorithm very suitable for future video coding solutions focusing on high fidelity video applications. The subjective quality remains similar when SVT is used while the coded bits are reduced.

The decoding complexity remains similar when incorporating SVT in video codecs with the proposed implementation. However, the encoding complexity of SVT can be relatively high because of the need to perform a number of RDO steps to select the best position and size for the transform. To address this issue, a novel fast algorithm was also proposed operating on macroblock and block levels to reduce the encoding complexity of SVT. The proposed fast algorithm first skips testing SVT for macroblocks for which SVT is unlikely to be useful by examining the RD cost of macroblock modes without SVT at a macroblock level. For the remaining macroblocks for which SVT may be useful, the proposed fast algorithm selects available candidate LPs based on the motion difference and utilizes a hierarchical search algorithm to select the best available candidate LP at a block level. A reduction of about 80% in the number of candidate LPs that need to be tested in RDO has been achieved, with only a marginal penalty in coding efficiency. Compared to H.264/AVC, the fast SVT achieves on average 5.31% and 4.03% bitrate reduction for low complexity configuration and high complexity configuration, respectively. Nevertheless, we note that still SVT may influence the data flow to make hardware implementation more difficultly than regular block boundary transform. We believe efficient hardware implementation could be studied further and is an interesting topic.

The proposed method can also be potentially used in many other situations, for instance, in inter-layer prediction in scalable video coding (SVC) and inter-view prediction in multiview video coding (MVC).

ACKNOWLEDGMENT

The authors would like to thank the reviewers and the Associate Editor for their valuable comments which helped improve the manuscript.

REFERENCES

- [1] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: Tools, performance, and complexity," *IEEE Circuits Syst. Mag.*, vol. 4, no. 1, pp. 7–28, Apr. 2004.
- [2] C. Zhang, K. Ugur, J. Lainema, and M. Gabbouj, "Video coding using spatially varying transform," in *Proc. 3rd PSIVT*, Jan. 2009, pp. 796–806.
- [3] C. Zhang, K. Ugur, J. Lainema, and M. Gabbouj, "Video coding using variable block-size spatially varying transforms," in *Proc. IEEE ICASSP*, Apr. 2009, pp. 905–908.
- [4] B. Zeng and J. Fu, "Directional discrete cosine transforms: A new framework for image coding," *IEEE Trans. Circuits, Syst. Video Technol.*, vol. 18, no. 3, pp. 305–313, Mar. 2008.
- [5] *Advanced Video Coding for Generic Audiovisual Services*, ITU-T Rec. H.264/ISO/IEC IS 14496-10 v3, 2005.

- [6] A. Nosratinia, "Denoising JPEG images by re-application of JPEG," in *Proc. IEEE Workshop MMSP*, Dec. 1998, pp. 611–615.
- [7] R. Samadani, A. Sundararajan, and A. Said, "Deringing and deblocking DCT compression artifacts with efficient shifted transforms," in *Proc. IEEE ICIP*, Oct. 2004, pp. 1799–1802.
- [8] J. Katto, J. Suzuki, S. Itagaki, S. Sakaida, and K. Iguchi, "Denoising intra-coded moving pictures using motion estimation and pixel shift," in *Proc. IEEE ICASSP*, Mar. 2008, pp. 1393–1396.
- [9] O. G. Guleryuz, "Weighted averaging for denoising with overcomplete dictionaries," *IEEE Trans. Image Process.*, vol. 16, no. 12, pp. 3020–3034, Dec. 2007.
- [10] C. Zhang, K. Ugur, J. Lainema, A. Hallapuro, and M. Gabbouj, "Low complexity algorithms for spatially varying transform," in *Proc. PCS*, May 2009.
- [11] S. Naito and A. Koike, "Efficient coding scheme for super high definition video based on extending H.264 high profile," in *Proc. SPIE Vis. Commun. Image Process.*, vol. 6077, 607727, Jan. 2006, pp. 1–8.
- [12] M. Wien, "Variable block-size transforms for H.264/AVC," *IEEE Trans. Circuits, Syst. Video Technol.*, vol. 13, no. 7, pp. 604–613, Jul. 2003.
- [13] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Trans. Circuits, Syst. Video Technol.*, vol. 13, no. 7, pp. 598–603, Jul. 2003.
- [14] S. Ma and C.-C. Kuo, "High-definition video coding with super-macroblocks," in *Proc. SPIE Vis. Commun. Image Process.*, vol. 6508, 650816, Jan. 2007, pp. 1–12.
- [15] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits, Syst. Video Technol.*, vol. 13, no. 7, pp. 688–703, Jul. 2003.
- [16] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Trans. Circuits, Syst. Video Technol.*, vol. 13, no. 7, pp. 614–619, Jul. 2003.
- [17] K. Vermeirsch, J. D. Cock, S. Notebaert, P. Lambert, and R. V. de Walle, "Evaluation of transform performance when using shape-adaptive partitioning in video coding," in *Proc. PCS*, May 2009.
- [18] *KTA Reference Model 1.8* [Online]. Available: <http://iphome.hhi.de/suehring/tml/download/KTA/jm11.0kta1.8.zip>
- [19] T. K. Tan, G. Sullivan, and T. Wedi, "Recommended simulation common conditions for coding efficiency experiments," document VCEG-AE10, ITU-T Q.6/SG16 VCEG, Jan. 2007.
- [20] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," document VCEG-M33, ITU-T SG16/Q6 VCEG, Mar. 2001.
- [21] S. Pateux and J. Jung, "An excel add-in for computing Bjontegaard metric and its evolution," document VCEG-AE07, ITU-T Q.6/SG16 VCEG, Jan. 2007.
- [22] F. Kamisli and J. S. Lim, "Transforms for the motion compensation residual," in *Proc. IEEE ICASSP*, Apr. 2009, pp. 789–792.



Cixun Zhang (S'09) received the B.E. and M.E. degrees in information engineering from Zhejiang University, Hangzhou, China, in 2004 and 2006, respectively. He is currently pursuing the Ph.D. degree from the Department of Signal Processing, Tampere University of Technology, Tampere, Finland.

Since August 2006, he has been a Researcher with the Department of Signal Processing, Tampere University of Technology, Tampere. His current research interests include video, image coding, and communication. He is currently working on standardization

of high efficiency video coding standard.

Mr. Zhang was the recipient of the AVS Special Award from the AVS Working Group of China in 2005, the Nokia Foundation Scholarship in 2008, and the Chinese Government Award for Outstanding Self-Financed Students Abroad in 2009.



Kemal Ugur received the M.S. degree in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2003, and the Ph.D. degree from the Tampere University of Technology, Tampere, Finland, in 2010.

He was with Nokia Research Center, Tampere, in 2004, where, currently, he is a Principal Researcher and leading a project on next generation video coding technologies. Since he joined Nokia, he has been actively participating in several standardization forums, such as Joint Video Team for

the standardization of multiview video coding extension of H.264/AVC, Video Coding Experts Group for exploration toward next generation video coding standard, 3GPP for mobile broadcast and multicast standard, and, recently, Joint Collaborative Team on Video Coding for standardization of high efficiency video coding standard. He has more than 25 publications in academic conferences and journals and around 30 patent applications.

Mr. Ugur is a member of the research team that won the highly prestigious Nokia Quality Award in 2006.



Jani Lainema received the M.S. degree in computer science from the Tampere University of Technology, Tampere, Finland, in 1996.

He was with the Visual Communications Laboratory, Nokia Research Center, Tampere, in 1996. Since then he has contributed to the designs of ITU-T and MPEG video coding standards as well as to the evolution of different multimedia service standards in 3GPP, digital video broadcasting, and digital living network alliance. Recently, he has been working as a Principal Scientist with Nokia Research

Center. His current research interests include video, image and graphics coding, and communications, as well as practical applications of game theory.



Antti Hallapuro received the M.S. degree in computer science from Tampere University of Technology, Tampere, Finland, in 2010.

He was with Nokia Research Center, Tampere, in 1998, where he has been working on video coding-related topics. He participated in H.264/AVC video codec standardization and is the author and co-author of several input documents and related academic papers. He has contributed significantly to productization of high performance H.264/AVC codecs for various computing platforms. His current

research interests include practical video coding and processing algorithms and their high performance implementations. He is currently working on topics of next generation video-coded standardization.



Moncef Gabbouj (S'86–M'91–SM'95–F'11) received the B.S. degree in electrical engineering from Oklahoma State University, Stillwater, in 1985, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1986 and 1989, respectively.

He has been an Academy Professor with the Academy of Finland, Helsinki, Finland, since January 2011. He was a Professor with the Department of Signal Processing, Tampere University of Technology, Tampere, Finland. He was the Head of

the department from 2002 to 2007. He was a Visiting Professor with the American University of Sharjah, Sharjah, UAE, from 2007 to 2008, and a Senior Research Fellow with the Academy of Finland from 1997 to 1998 and from 2007 to 2008. His current research interests include multimedia content-based analysis, indexing and retrieval, nonlinear signal and image processing and analysis, voice conversion, and video processing and coding.

Dr. Gabbouj served as the Distinguished Lecturer for the IEEE Circuits and Systems Society from 2004 to 2005. He served as an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, and was a Guest Editor of *Multimedia Tools and Applications*, *European Journal Applied Signal Processing*. He was the Past Chairman of the IEEE Finland Section, the IEEE CAS Society, the Technical Committee on DSP, and the IEEE SP/CAS Finland Chapter. He was the recipient of the 2005 Nokia Foundation Recognition Award. He was the co-recipient of the Myril B. Reed Best Paper Award from the 32nd Midwest Symposium on Circuits and Systems and the co-recipient of the NORSIG'94 Best Paper Award from the 1994 Nordic Signal Processing Symposium. He is a member of the IEEE Signal Processing Society and the IEEE Circuits and Systems Society.