

# VIDEO CODING USING VARIABLE BLOCK-SIZE SPATIALLY VARYING TRANSFORMS

Cixun Zhang<sup>\*</sup>, Kemal Ugur<sup>§</sup>, Jani Lainema<sup>§</sup>, Moncef Gabbouj<sup>\*</sup>

<sup>\*</sup>Tampere University of Technology, Tampere, Finland

<sup>§</sup>Nokia Research Center, Tampere, Finland

## ABSTRACT

In our previous work, we introduced Spatially Varying Transforms (SVT) for video coding, where the location of the transform block within the macroblock is not fixed but varying. In this paper, we extend this concept and present a novel method, called Variable Block-size Spatially Varying Transforms (VBSVT). VBSVT utilizes Variable Block-size Transforms (VBT) in the SVT framework, and is shown to be more preferable for coding prediction error with different characteristics than fixed block-size SVT and also the standard methods that use fixed or adaptive block sizes at fixed spatial locations. In addition, VBSVT has similar decoding complexity with fixed block-size SVT and lower decoding complexity compared to standard methods as only a portion of the prediction error needs to be decoded. Experimental results show that, VBSVT achieves 4.1% gain over H.264/AVC on average over a wide range of test set. Gains become more significant at high quality levels and go up to 13.5%, which makes the proposed algorithm very suitable for future video coding solutions focusing on high fidelity applications.

**Index Terms**— Video coding, H.264/AVC, Variable Block-size Transform (VBT), Spatially Varying Transform (SVT)

## 1. INTRODUCTION

H.264/AVC is the latest international video coding standard and it 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. Additional complexity of H.264/AVC becomes an issue in resource constrained applications, such as mobile video services (mobile TV, video telephony etc) and handheld consumer electronics (camcorders, digital still cameras etc). On the other hand, as display resolutions and available bandwidth/storage increases 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 implemen-

tation complexity. In our previous paper [1], we proposed a novel algorithm, named as Spatially Varying Transform (SVT), which provides coding efficiency gains over H.264/AVC while lowering the decoding complexity. The motivations leading to design of SVT are two-fold:

1. The block based transform design in most existing video coding standards does not adapt the underlying transform to the structure of the prediction error to be coded. In this case, the coding efficiency decreases.
2. Coding the entire prediction error signal may not be the best in terms of rate distortion tradeoff because the prediction error signal may contain noise which contributes little to quality but requires additional bits to code.

In [1], we used a single 8x8 transform block with varying spatial location within the 16x16 macroblock to code the prediction error. This was shown to improve the coding efficiency of standard video coders, such as H.264/AVC, as it localizes the prediction error better. However, as the block size of SVT is fixed to 8x8, certain macroblocks with prediction errors of different characteristics cannot be efficiently coded. In this paper, we extend the concept to Variable Block-size Spatially Varying Transforms (VBSVT) by using Variable Block-size Transforms (VBT) within the SVT framework. More specifically, in addition to the single 8x8 transform block as used in [1], we use two additional transform blocks with different horizontal and vertical sizes, and select the transform block size adaptively. By adapting both the size of the transform block and its spatial location, the prediction error is better localized and the underlying correlations are better exploited. In addition, as the number of transform blocks used to code the prediction error is reduced, the decoding complexity is reduced. Experimental results show that, when implemented to H.264/AVC, proposed algorithm achieves 4.1% gain on average over a wide range of test sequences. Gains become more significant at high quality levels and go up to 13.5%, which makes the proposed algorithm very suitable for future video coding solutions focusing on high fidelity applications.

This paper is organized as follows. A brief review of SVT is presented in section 2. Section 3 introduces VBSVT and presents a detailed analysis. Experimental results are given in section 4 and section 5 concludes the paper.

## 2. SPATIALLY VARYING TRANSFORM

Transform coding is widely used in video coding standards to decorrelate the prediction error and achieve increased compression rates. Typically, transform coding is applied to prediction error at fixed locations. However, this has several drawbacks that may hurt the coding efficiency and decrease visual quality. First of all, if the localized prediction error at fixed locations has a structure that is not suitable for the underlying transform, many high frequency coefficients will be generated in the transform domain and they need many bits to code. In this situation, the coding efficiency decreases. Moreover, notorious visual artifacts such as ringing may appear when these high frequency coefficients get quantized.

In our previous work [1], SVT was proposed to reduce these drawbacks of transform coding. The main idea of SVT is that the transform coding is not restricted to be at fixed locations, but instead can be applied at any location according to the characteristics of the prediction error. With this flexibility, we are also able to achieve coding efficiency improvement by selecting and coding the best portion of the prediction error in terms of rate distortion tradeoff. Generally this can be done as follows. We only code a sub-region in a certain residual region after prediction. The sub-region is found by searching inside the region according to a certain criterion. Information of the location of the selected sub-region inside the region is coded into the bitstream if necessary. In [1], one configuration of this general idea is used: a single 8x8 transform block is selected inside a 16x16 macroblock and only this 8x8 block is coded. This is shown in Fig. 1 below. Rate Distortion Optimization (RDO) is used to select the best location parameter  $(\Delta x, \Delta y)$  for SVT and determine whether SVT is used for coding each macroblock. It is suggested that  $(\Delta x, \Delta y)$  be selected from the set  $\Phi = \{(0..8, 0), (0..8, 8), (0, 1..7), (8, 1..7)\}$  which has 32 candidates. This set is chosen after performing many simulations and studying the statistical distributions of  $(\Delta x, \Delta y)$  at different bitrates and sequences. Reader is referred to [1] for more details. As in [1], in this paper, notation  $x..y$  is used to specify a range of integer values starting from  $x$  to  $y$  inclusive, with  $x, y$  being integer numbers.

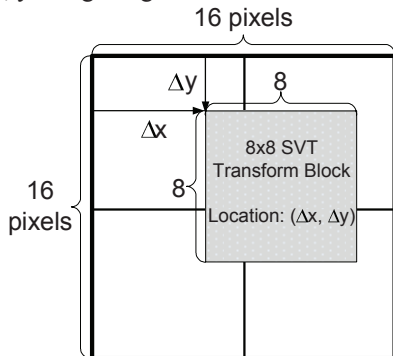


Fig.1 Illustration of 8x8 spatially varying transform

## 3. VARIABLE BLOCK-SIZE SPATIALLY VARYING TRANSFORMS

It is well established in video coding community that VBT can improve the coding efficiency [2]. Thus it is reasonable to expect improved performance by extending the fixed block-size SVT used in our previous paper [1] to cover variable block-size. In the following subsections, we will first discuss about the selection of different block-size SVT and after that we present the details of VBSVT.

### 3.1. Different block-size spatially varying transforms

Different block-size transform can be used within the SVT framework. For example, different from the 8x8 SVT used in our previous study [1], we can use 16x4 SVT and 4x16 SVT (we denote this as 16x4+4x16 SVT in the rest of the paper), which are illustrated in Fig. 2. As shown in the figure, we select and code one 16x4/4x16 block inside a 16x16 macroblock with a corresponding 16x4/4x16 transform when using 16x4/4x16 SVT respectively. Considering both of them as a whole, the location parameter of the selected 16x4/4x16 block can be represented as  $(\text{shape}, \Delta y/\Delta x)$  which can be selected from the set  $\Phi = \{(0..1, 0..12)\}$  where  $\text{shape}=0$  means 16x4 block is selected and  $\text{shape}=1$  means 4x16 block is selected. There are in total 26 location parameter candidates and statistics show that it needs 4.61 bit on average, for all the test sequences we use in our experiments, to code the index of selected location parameter in Huffman coding. For simplicity, we use 5-bit fixed length code instead. RDO is used to select the best location parameter  $(\text{shape}, \Delta y/\Delta x)$  for SVT and determine if SVT is used for coding each macroblock.

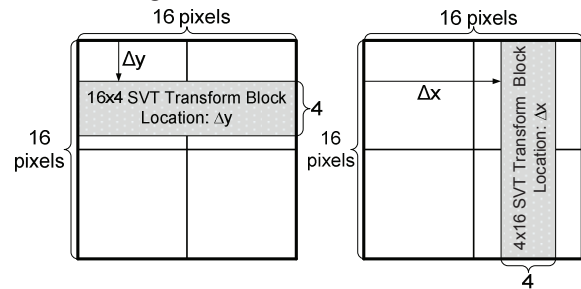


Fig.2 Illustration of 16x4 and 4x16 spatially varying transform

For the selected 16x4/4x16 block, 16x4/4x16 transform is used accordingly. In general, the separable forward and inverse 2-D transform of a 2-D signal can be written as

$$C = T_v \cdot X \cdot T_h^T, \quad (1)$$

$$X_r = T_v^T \cdot C \cdot T_h \quad (2)$$

respectively, where  $\mathbf{X}$  denotes a matrix representing  $M \times N$  pixel block of  $N$  pixels horizontally and  $M$  pixels vertically,  $\mathbf{C}$  is the transform coefficient matrix, and  $\mathbf{X}_r$  denotes a matrix representing reconstructed signal block.  $\mathbf{T}_v$  and  $\mathbf{T}_h$  are the  $M \times M$  and  $N \times N$  transform kernels in vertical and hori-

zonal direction, respectively. The superscript T denotes matrix transposition. For 16x4 and 4x16 transforms, a 4x4 and a 16x16 transform kernel need to be specified. In this paper, we use the 4x4 transform kernel in H.264/AVC [3] and the 16x16 transform kernel in [4] because of its good compatibility with H.264/AVC and low complexity. 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.

Integration of 16x4+4x16 SVT into H.264/AVC framework is in principal the same as that of 8x8 SVT proposed in [1], with modifications only because of different block size whenever needed.

Compared to 8x8 SVT in [1], the encoding complexity of 16x4+4x16 SVT proposed here is lower since there are only 26 location parameter candidates to be tested in RDO rather than 32 for 8x8 SVT. The decoding complexity of 16x4+4x16 SVT would be similar to that of 8x8 SVT since the main difference between them comes from different inverse transform and deblocking, which are not significant.

Last but not the least, we note that the reason we choose to use 16x4+4x16 SVT is because of its good compatibility with H.264/AVC, low complexity and good performance as will be shown later. However, other block-size SVT can also be used, like full or pruned 16x8+8x16 SVT with 128 or 64 coefficients coded. Directional spatially varying transforms with a directional oriented block selected and coded with a corresponding directional transform (e.g., [5]) can also be used (expected to be more efficient in intra coding). Finally, if SVT is applied to a region larger than a 16x16 macroblock, its design will be also more flexible.

### 3.2. Variable block-size spatially varying transforms

To better code the prediction error with different characteristics, VBSVT can be used instead of fixed block-size SVT. In this paper, we study VBSVT with 8x8, 16x4 and 4x16 block sizes. Again, RDO is used to select the best one among these three for VBSVT and determine if VBSVT is used for coding each macroblock.

The encoding complexity of VBSVT is about the sum of 8x8 SVT and 16x4+4x16 SVT, while decoding complexity remains similar to either 8x8 SVT or 16x4+4x16 SVT.

## 4. EXPERIMENTAL RESULTS

We implemented 16x4+4x16 SVT and VBSVT on KTA1.8 reference software [6] in order to evaluate their effectiveness. The test condition is exactly the same as that used in our previous paper for 8x8 SVT [1]. Important coding parameters used in our experiments are listed as follows:

- High Profile
- $QP_I=22, 27, 32, 37, QP_P=QP_I+1$
- CAVLC is used as the entropy coding
- Frame structure is IPPP, 4 reference frames

- Motion vector search range  $\pm 64$  pels, resolution  $\frac{1}{4}$ -pel
- RDO in the “High Complexity Mode”
- Two configurations are tested. 1) Low complexity configuration: motion estimation block sizes are 16x16, 16x8, 8x16, 8x8, and only 8x8 transform is used. In this case, when 16x4+4x16 SVT is used, only 16x4+4x16 transform is used for macroblocks that use SVT. This represents a low complexity codec with most effective tools for HD video coding; 2) High complexity configuration: motion estimation block sizes are 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4, both 4x4 and 8x8 transform are used. In this case, when 16x4+4x16 SVT is used, either 4x4 or 16x4+4x16 transform is selected for macroblocks that use SVT. This represents a high complexity codec with full usage of the tools provided in the standard. In both configurations, VBSVT is a combination of 8x8 SVT and 16x4+4x16 SVT. As in [1], we do not change the motion estimation, sub-macroblock partition decision process, even for the macroblocks that use VBSVT.

We measure the average bitrate reduction ( $\Delta$ BD-RATE) compared to H.264/AVC according to [7] for both low complexity and high complexity configurations. The results are shown in Table 1 and Table 2 respectively. Results of 8x8 SVT [1] are also presented for comparison. As we can see, 16x4+4x16 SVT performs better than 8x8 SVT while VBSVT performs best, both in low complexity configuration and high complexity configuration. 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, compared to H.264/AVC. The gain of VBSVT is lower in high complexity configuration, because the 4x4 transform in H.264/AVC (and smaller motion estimation block sizes) is more efficient in certain cases.

Fig. 3 shows the R-D curves for Panslow sequence. We note that the gain of VBSVT comes at high bitrates 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 [8], 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.

We also measure the percentage of 8x8 SVT and 16x4+4x16 SVT selected in VBSVT. The results are shown in Table 3. We can see that in VBSVT scheme, 16x4+4x16 SVT are used generally more often than 8x8 SVT, mainly because of its higher coding efficiency. However, 8x8 SVT is also used in significant portion of the cases. This shows that different block-size SVT can be suitable for different situation and VBSVT would be a more preferable algorithm for coding prediction error with different characteristics than fixed block-size SVT. Fast algorithms can be used to lower the encoding complexity (mainly due to the brute force search in RDO) of VBSVT.

## 5. CONCLUSIONS

In this paper, we presented a novel method for video coding that called Variable Block-size Spatially Varying Transforms (VBSVT). VBSVT utilizes Variable Block-size Transforms (VBT) within the SVT framework and is shown to perform better than fixed block-size SVT with similar decoding complexity. Compared to standard methods that use fixed or adaptive block sizes at fixed spatial locations, VBSVT improves the coding efficiency because it localizes the prediction error better. The proposed algorithm is implemented to H.264/AVC. We show that VBSVT achieves on average 4.11% bitrate reduction in low complexity configuration which represents a low complexity codec with most effective tools for HD video coding and on average 2.43% bitrate reduction in high complexity configuration which represents a high complexity codec with full usage of the tools provided in the standard, respectively. The coding efficiency is increased with lower decoding complexity because only a portion of the prediction error needs to be decoded. Gains become more significant at high quality levels and go up to 13.5%, which makes the proposed algorithm very suitable for future video coding solutions focusing on high fidelity applications.

## 6. REFERENCES

- [1] C. Zhang, K. Ugur, J. Lainema and M. Gabbouj, "Video coding using spatially varying transform", PSIVT 2009.
- [2] 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.
- [3] 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.
- [4] S. Ma and C.-C. Kuo, "High-definition video coding with super-macroblocks", SPIE Visual Communications and Image Processing, Vol. 6508, pp. 650816-1-650816-12, Jan. 2007.
- [5] 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, March. 2008.
- [6] KTA reference model 1.8 [online], available at <http://iphome.hhi.de/suehring/tml/download/KTA/>
- [7] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves", VCEG Doc. VCEG-M33, March 2001.
- [8] S. Pateux, J. Jung, "An excel add-in for computing Bjontegaard metric and its evolution", VCEG Doc. VCEG-AE07, Jan. 2007.

Table 1 Bitrate reduction compared to H.264 (Low complexity configuration)

Sequence	8x8 SVT	16x4+4x16 SVT	VBSVT
BigShips	-2.87%	-4.50%	-4.96%
ShuttleStart	-2.51%	-3.25%	-3.42%
City	-3.30%	-3.82%	-4.45%
Night	-2.33%	-3.70%	-4.12%
Optis	-2.65%	-3.30%	-3.97%

Spincalendar	-2.07%	-3.13%	-3.49%
Cyclists	-2.14%	-2.10%	-2.52%
Preakness	-2.34%	-2.98%	-3.71%
Panslow	-4.59%	-5.51%	-6.19%
Sheriff	-2.18%	-3.04%	-3.43%
Sailormen	-2.11%	-4.77%	-4.94%
Average	-2.64%	-3.65%	-4.11%

Table 2 Bitrate reduction compared to H.264 (High complexity configuration)

Sequence	8x8 SVT	16x4+4x16 SVT	VBSVT
BigShips	-1.68%	-2.95%	-3.04%
ShuttleStart	-0.91%	-1.86%	-1.46%
City	-1.46%	-1.99%	-2.37%
Night	-1.35%	-2.54%	-2.59%
Optis	-1.48%	-2.41%	-3.04%
Spincalendar	-1.46%	-2.17%	-2.24%
Cyclists	-1.07%	-1.21%	-1.34%
Preakness	-1.18%	-1.41%	-1.78%
Panslow	-2.46%	-3.07%	-3.16%
Sheriff	-1.20%	-2.26%	-2.26%
Sailormen	-1.34%	-3.74%	-3.48%
Average	-1.42%	-2.33%	-2.43%

Table 3 Percentage of 8x8 SVT and 16x4+4x16 SVT selected in VBSVT for different sequences

Sequence	Low complexity configuration		High complexity configuration	
	8x8 SVT	16x4+4x16 SVT	8x8 SVT	16x4+4x16 SVT
BigShips	37.3%	62.7%	39.7%	60.3%
ShuttleStart	46.1%	53.9%	42.5%	57.5%
City	47.1%	52.9%	45.1%	54.9%
Night	46.8%	53.2%	43.0%	57.0%
Optis	50.2%	49.8%	47.0%	53.0%
Spincalendar	44.3%	55.7%	43.7%	56.3%
Cyclists	52.0%	48.0%	47.6%	52.4%
Preakness	47.8%	52.2%	45.7%	54.3%
Panslow	47.0%	53.0%	48.0%	52.0%
Sheriff	48.6%	51.4%	45.0%	55.0%
Sailormen	36.6%	63.4%	35.2%	64.8%
Average	45.8%	54.2%	43.9%	56.1%

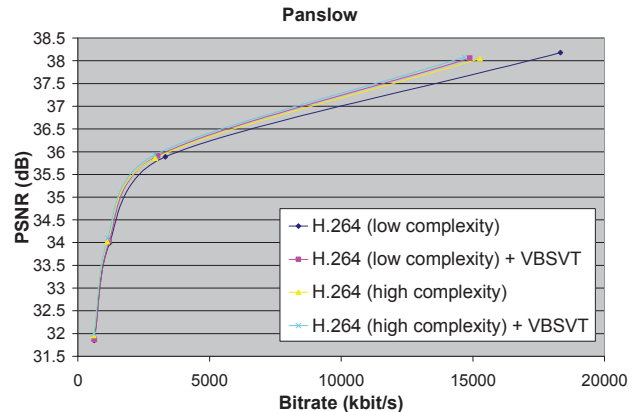


Fig.3 R-D curve for Panslow sequence