

Video Coding Using Spatially Varying Transform

Cixun Zhang¹, Kemal Ugur², Jani Lainema², and Moncef Gabbouj¹

¹Tampere University of Technology, Tampere, Finland
{cixun.zhang, moncef.gabbouj}@tut.fi

²Nokia Research Center, Tampere, Finland
{kemal.ugur, jani.lainema}@nokia.com

Abstract. In this paper, we propose a novel algorithm, named as Spatially Varying Transform (SVT). The basic idea of SVT is that we do not restrict the transform coding inside normal block boundary 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 rate distortion tradeoff. The proposed algorithm is implemented and studied in the H.264/AVC framework. We show that the proposed algorithm achieves 2.64% bit-rate reduction compared to H.264/AVC on average over a wide range of test set. Gains become more significant at high bit-rates and the bit-rate reduction can be up to 10.22%, which makes the proposed algorithm very suitable for future video coding solutions focusing on high fidelity applications. The decoding complexity is expected to be decreased because only a portion of the prediction error needs to be decoded.

Keywords: H.264/AVC, video coding, transform, spatially varying transform (SVT).

1 Introduction

H.264/AVC (H.264 for short hereafter) 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. It is estimated in [1] that the encoder complexity increases with more than one order of magnitude between MPEG-4 Part 2 (Simple Profile) and H.264 (Main Profile) and with a factor of 2 for the decoder. For mobile video services (video telephony, mobile TV etc.) and handheld consumer electronics (digital still cameras, camcorders etc), additional complexity of H.264 becomes an issue due to the limited resources of these devices. 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 implementation complexity. In this paper, we propose a novel algorithm, named as Spatially Varying Transform (SVT), which provides coding efficiency gains over

H.264 and is expected to lower the decoding complexity. The technique is developed and studied mainly for coding HD resolution video, but it could be extended also for other resolutions. The motivations leading to design of SVT are two-fold:

1. The 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 [2], directional discrete cosine transforms is proposed to improve the efficiency of transform coding for directional edges. However, efficient coding of horizontal/vertical edges inside the blocks and non-directional edges was not addressed.
2. Coding the entire prediction error signal may not be the best in terms of rate distortion tradeoff. An example is the SKIP mode in H.264 [3], which does not code the prediction error at all.

The basic idea of SVT is that we do not restrict the transform coding inside normal block boundary 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 rate distortion tradeoff. This is done by searching inside a certain residual region after intra prediction or motion compensation, for a sub-region and only coding this sub-region. The location parameter of the sub-region inside the region is coded into the bitstream if there are non-zero coefficients.

The proposed algorithm is implemented and studied in H.264 framework. Extensive experimental results show that it can improve the coding efficiency of H.264. In addition decoding complexity is expected to be lowered a little mainly because only a portion of the prediction error needs to be decoded. Encoding complexity of the proposed technique is higher mainly due to the brute force search process. Fast encoding algorithms are being studied to alleviate this aspect of the proposed technique.

The paper is organized as follows: The proposed algorithm is introduced in section 2 and its integration into H.264 framework is described in section 3. Experimental results are given in section 4. Section 5 concludes the paper and also presents future research directions.

2 Spatially Varying Transform

The basic idea of SVT is that the transform coding is not restricted inside normal block boundary but applied to a portion of the prediction error according to the characteristics of the prediction error. We only code a sub-region in a certain residual region after intra prediction or motion compensation. 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 there are non-zero coefficients. Fig. 1 shows an illustrative example of the idea: one 8x8 block inside a 16x16 macroblock is selected and only this 8x8 block is coded. In this paper, we focus our discussion on this particular configuration, which turns out to be promising, as we will see later. However, we note that there is no restriction of the “sub-region” and “region”, for example, on their size, shape, etc when using the idea

in a general sense. Other possible configurations of the idea to achieve further gain in coding efficiency are under study.

In the following, we further discuss two key issues of SVT in more detail: selection of location parameter candidates and filtering of block boundaries.

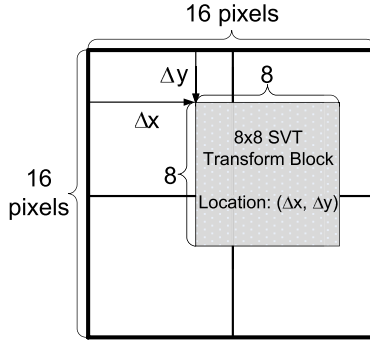


Fig. 1. Illustration of spatially varying transform

2.1 Selection of Location Parameter Candidates

When there are non-zero coefficients of the selected 8x8 block, its location inside the macroblock needs to be coded and transmitted to the decoder. As shown in Fig. 1, the location of the selected 8x8 block inside the current macroblock is denoted by (Δx, Δy) where Δx and Δy each can take integer value from 0 to 8, if the selected block is restricted to have the same size (which facilitates the transform design) for all locations. There are in total 81 possible combinations and we need to select the best one according to a certain criterion. In this paper, Rate-Distortion Optimization (RDO) is used to select the best (Δx, Δy) in terms of RD tradeoff by minimizing the following:

$$J = D + \lambda \cdot R . \tag{1}$$

where J is the RD cost of the selected combination, D is the distortion, R is the bit rate and λ is the Lagrangian multiplier. The reconstruction residue for the remaining part of the 16x16 residual macroblock is simply set to be 0 in our implementation, but different values can be used and might be beneficial in certain cases (luminance change, etc). Similarly, RDO can also be used to decide if SVT should be used for a macroblock.

Selection of location parameter candidates is important since it directly affects the encoding complexity and the performance of SVT. We study the frequency distribution of (Δx, Δy) and it is observed that the most frequently selected (Δx, Δy), are (0..8,0), (0..8,8), (0,1..7), (8,1..7)¹, which takes up a percentage around 60% of all 81 combinations. According to extensive experiments, this is generally true for

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.

different sequences, macroblock partitions and Quantization Parameters (QP). Fig. 2 below shows the distributions of $(\Delta x, \Delta y)$ for different macroblock partitions of BigShips sequence when QP equal to 23. As we will see in section 4, using this subset of location parameters turns out to be an efficient configuration of the proposed algorithm.

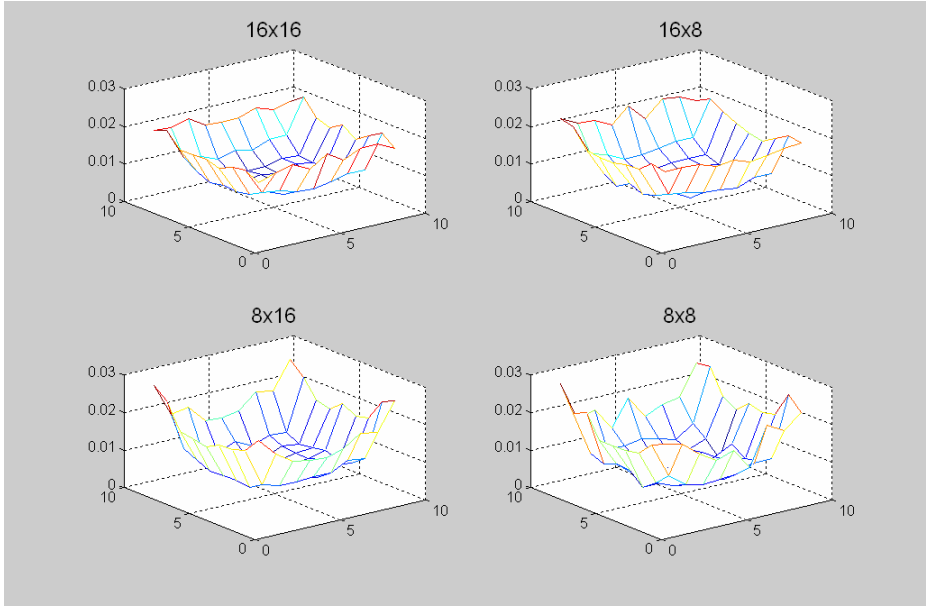


Fig. 2. Frequency distribution of $(\Delta x, \Delta y)$ for different macroblock partitions of BigShips sequence, at QP=23 (Z axis denotes the frequency)

2.2 Filtering of Block Boundaries

Due to the coding (transform and quantization) of the selected 8x8 block, blocking artifacts may appear around its boundary with the remaining non-coded part of the macroblock. A deblocking filter can be applied to improve the subjective quality and possibly also the objective quality. An example in the framework of H.264 will be described in detail later in section 3.4.

3 Integration of Spatially Varying Transform into H.264 Framework

In this paper, we study the proposed technique in H.264 framework. Fig. 3 below is the block diagram of extended H.264 encoder with SVT. As shown in Fig. 3, encoder needs to search the best 8x8 block inside macroblocks that use SVT, which is marked as “SVT Search” in the diagram. Then encoder decides whether to use SVT for the current macroblock, using RDO in our implementation. The location parameter is

coded and transmitted in the bitstream. A corresponding decoder needs to decode the location parameter for macroblocks that use SVT, which is marked as “SVT L.P. Decoding” in the diagram. One thing that is worth mentioning here is, in this paper and also the experimental results in section 4, we do not change the motion estimation, sub-macroblock partition decision process, even for the macroblocks that use SVT. After the residual macroblock is generated as normal, RDO is used to decide whether SVT should be used. The reason is to keep the encoding complexity low. However, we note that the normal criteria used in these encoding processes for normal macroblocks may not be optimal for macroblocks that use SVT. Better encoding algorithms are under study.

Several key parts of the H.264 standard [3], for example, macroblock types, Coded Block Pattern (CBP), entropy coding, deblocking, also need to be adjusted. Proposed modifications aiming at good compatibility with H.264 are described in the following sub-sections.

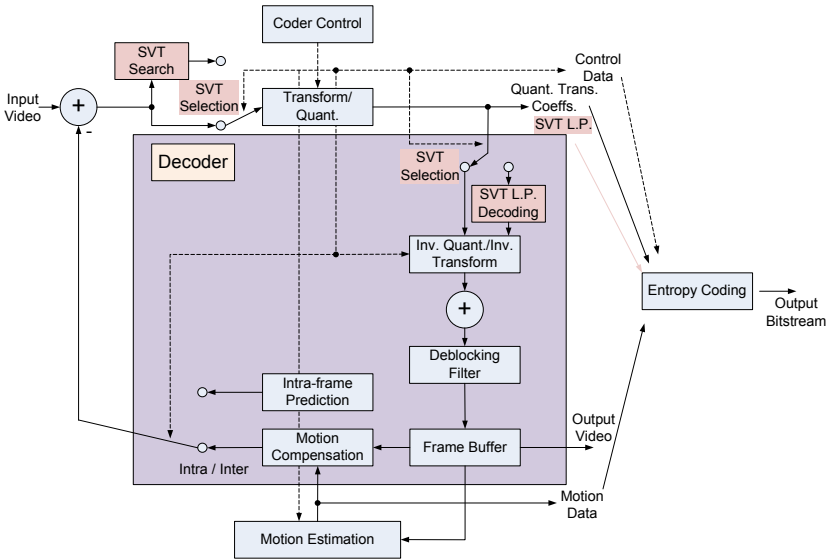


Fig. 3. Block diagram of extended H.264 encoder with spatially varying transform

3.1 Macroblock Types

In this work, we focus our study of SVT on coding inter prediction error in P slices although the idea can be easily extended to be also used in I and B slices. Table 1 below shows the extended macroblock types for P slices in H.264 [3] (original intra macroblock types in H.264 are not included), with the name of new macroblock types that use SVT in *italics*. The macroblock type index is coded using Exp-Golomb codes in the same way as H.264. The sub-macroblock types are kept unchanged and therefore not shown in the table.

Table 1. Extended macroblock types for P slices in H.264 with spatially varying transform

mb_type	Name of mb_type
0	P_16x16
1	P_16x16_SVT
2	P_16x8
3	P_16x8_SVT
4	P_8x16
5	P_8x16_SVT
6	P_8x8
7	P_8x8_SVT
8	P_8x8ref0
9	P_8x8ref0_SVT
Inferred	P_Skip

3.2 Coded Block Pattern

In this work, we only use SVT for luma component coding. As shown in Fig. 1, since only one 8x8 block is selected and coded in macroblocks that use SVT, we can use 1 bit for luma CBP or jointly code it with chroma CBP as H.264 does [3]. However, in our experiments with many test sequences, we found that luma CBP is probably 1 when QP is low (and probably 0 when the QP is high) where most gain of SVT comes from, so we restrict the new macroblock modes to have luma CBP equal to 1 and there is no need to code this information. Chroma CBP is represented in the same way as H.264 [3]. An alternative way would be to infer the luma CBP according to QP.

3.3 Entropy Coding

In H.264 [3], when Context Adaptive Variable Length Coding (CAVLC) is used as the entropy coding, different coding table for total number of non-zero transform coefficients and trailing ones of current block is selected depending on the characteristics (the number of non-zero transform coefficients) of the neighboring blocks. For macroblocks that use SVT, for simplicity, a fixed coding table is used. Besides, we may also need to derive the information about the number of non-zero transform coefficients every luma 4x4 block has. When the selected 8x8 block aligns with the normal block boundaries, no special scheme is needed. Otherwise, the following scheme is used in our implementation:

1. A luma 4x4 block is marked to have non-zero coefficients if it overlaps with a coded block that has non-zero coefficients in the selected 8x8 block, and marked not to have non-zero coefficients otherwise. This information may also be used in other processes, e.g., deblocking.
2. The number of non-zero transform coefficients for each 4x4 block that is marked to have non-zero coefficients, is empirically set to the same to

$$(nC+nB/2)/nB . \quad (2)$$

where nC is the total number of non-zero transform coefficients in the current macroblock and nB is the number of blocks marked to have non-zero

coefficients. Operator “/” is integer division with truncation of the result toward zero.

We note that due to the truncation in (2), a 4x4 block may be marked to have non-zero coefficients according to step 1 but has no non-zero coefficients according to (2) in step 2 at the same time. In our implementation, when only the information about whether a block has non-zero transform coefficients or not is, we use the result of step 1; while when the information about how many non-zero transform coefficients a block has is needed, we use the result of step 2.

3.4 Deblocking

As shown in Fig. 4 below, for macroblocks that use SVT, the deblocking process in H.264 [3] needs to be adjusted because the selected 8x8 block may not align with the normal block boundaries. The following scheme is used in our implementation:

1. First, the boundary edges of the selected 8x8 block and the remaining part of the macroblock are filtered. The filtering criteria and process of these edges are similar to those used in H.264, with minor modifications based on empirical tests to produce visually pleasing results for a variety of content.
2. Second, the normal internal edges and macroblock boundary edges are filtered except those which are inside the selected 8x8 block or overlap with the boundary edges which have already been filtered in the first step. The filtering criteria and process of these edges are kept unchanged as in H.264.

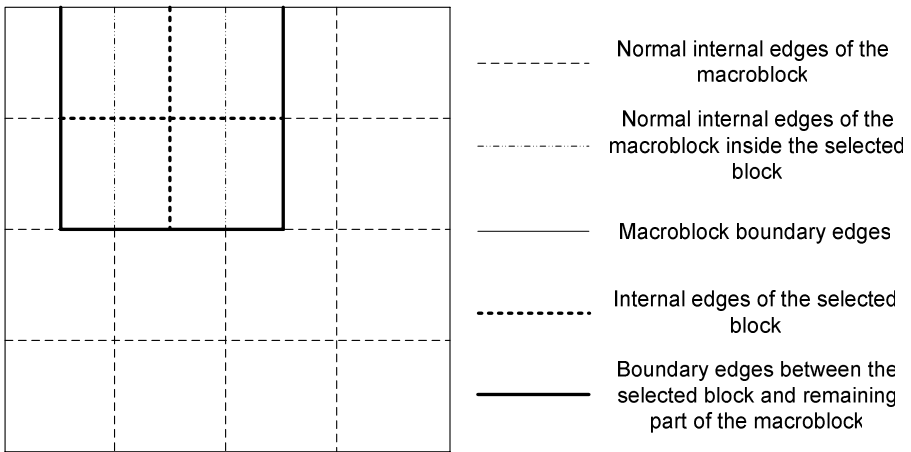


Fig. 4. Illustration of different edges of macroblocks that use spatially varying transform

4 Experimental Results

We implemented SVT on KTA1.8 reference software [4] in order to evaluate its effectiveness. Important coding parameters used in our experiments are listed as follows:

- High Profile
- QPI=22, 27, 32, 37, QPP=QPI+1
- CAVLC is used as the entropy coding
- Frame structure is IPPP, 4 reference frames
- Motion vector search range ± 64 pels, resolution $\frac{1}{4}$ -pel
- RDO in the “High Complexity Mode”
- Two configurations are tested. 1) Low complexity configuration: motion compensation block size are 16×16 , 16×8 , 8×16 , 8×8 , only 8×8 transform is used. In this case, also only 8×8 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 compensation block size are 16×16 , 16×8 , 8×16 , 8×8 , 8×4 , 4×8 , 4×4 , both 4×4 and 8×8 transform are used. In this case, either 4×4 or 8×8 transform is selected for macroblocks that use SVT. This represents a high complexity codec with full usage of the tools provided in the standard.

We test three configurations of the proposed algorithm in our experiments:

1. SVT32: The location parameter $(\Delta x, \Delta y)$ is selected in the set: $\Phi_{32} = \{(0..8,0), (0..8,8), (0,1..7), (8,1..7)\}$ which has 32 candidates. The index is coded using 5-bit fixed length code. As we will see, this turns out to be an efficient configuration of SVT.
2. SVT4: The location parameter $(\Delta x, \Delta y)$ is selected in the set: $\Phi_4 = \{(0,0), (0,8), (8,0), (8,8)\}$ which has 4 candidates. The index is coded using 2-bit fixed length code. This serves as a comparison to show the effectiveness and necessity of the searching process of SVT.
3. SVT81: The location parameter $(\Delta x, \Delta y)$ is selected in the set: $\Phi_{81} = \{(0..8,0..8)\}$ which has 81 candidates. The index is coded using 7-bit fixed length code. Although this overhead may be reduced a little by using variable length code, SVT81 serves as a meaningful comparison to show the performance of a configuration of the proposed algorithm with a selected subset of location parameters, which is SVT32 in our case.

Table 2. Experimental results (Low complexity configuration)

Sequence (1280x720/60p)	Δ BD-RATE		
	SVT32	SVT4	SVT81
BigShips	-2.87%	-0.92%	-2.41%
ShuttleStart	-2.51%	-1.76%	-2.18%
City	-3.30%	-1.77%	-2.97%
Night	-2.33%	-0.55%	-1.95%
Optis	-2.65%	-0.12%	-2.04%
Spincalendar	-2.07%	-0.84%	-1.84%
Cyclists	-2.14%	-1.41%	-1.48%
Preakness	-2.34%	-0.17%	-2.26%
Panslow	-4.59%	-2.17%	-4.14%
Sheriff	-2.18%	-0.68%	-1.77%
Sailormen	-2.11%	-0.60%	-1.81%
Average	-2.64%	-1.00%	-2.26%

Table 3. Experimental results (High complexity configuration)

Sequence (1280x720/60p)	Δ BD-RATE		
	SVT32	SVT4	SVT81
BigShips	-1.68%	-0.14%	-0.96%
ShuttleStart	-0.91%	-0.45%	-0.62%
City	-1.46%	-0.31%	-1.19%
Night	-1.35%	-0.21%	-0.99%
Optis	-1.48%	+0.06%	-1.25%
Spincalendar	-1.46%	-0.59%	-1.23%
Cyclists	-1.07%	-0.32%	-0.60%
Preakness	-1.18%	+0.12%	-0.93%
Panslow	-2.46%	-0.96%	-2.18%
Sheriff	-1.20%	-0.13%	-0.76%
Sailormen	-1.34%	+0.07%	-1.14%
Average	-1.42%	-0.26%	-1.08%

Table 4. Percentage of Φ_{32} and Φ_4 selected in Φ_{81} for different sequences

Sequence (1280x720/60p)	Low complexity configuration		High complexity configuration	
	Φ_{32}/Φ_{81}	Φ_4/Φ_{81}	Φ_{32}/Φ_{81}	Φ_4/Φ_{81}
BigShips	59.8%	6.6%	56.7%	4.7%
ShuttleStart	63.1%	6.7%	58.4%	4.3%
City	58.7%	6.8%	55.8%	4.7%
Night	67.2%	12.0%	65.1%	10.2%
Optis	57.7%	5.4%	55.3%	3.3%
Spincalendar	57.9%	8.5%	54.6%	5.9%
Cyclists	63.2%	7.4%	61.0%	5.4%
Preakness	68.1%	10.3%	64.9%	6.1%
Panslow	59.0%	8.5%	56.3%	6.0%
Sheriff	58.9%	5.9%	56.4%	4.1%
Sailormen	57.5%	7.1%	55.0%	5.3%
Average	61.0%	7.7%	58.1%	5.5%

We calculate the average bit-rate reduction (Δ BD-RATE) according to [5] for both low complexity and high complexity configurations. The results are shown in Table 2 and Table 3 below. We also measure the percentage of Φ_{32} and Φ_4 selected in SVT81. The results are shown in Table 4.

As we can see from Table 2 and Table 3, SVT32 performs better than both SVT4 and SVT81. It achieves on average 2.64% (up to 4.59%) bit-rate reduction in low complexity configuration and on average 1.42% (up to 2.46%) bit-rate reduction in high complexity configuration, respectively. The percentage of Φ_{32} selected in Φ_{81} is around 60%. Fig. 5 and Fig. 6 show the R-D curves for City and Panslow sequences. We can see that the gain of SVT comes mainly at high bit-rates and can be quite significant. This is true for most sequences we tested. Take Panslow sequence as an example, by using the performance evaluation tool provided in [6], we are able to show that VBSVT achieves 10.22% and 5.59% bit-rate reduction at 37 dB and 38 dB compared to H.264/AVC, in low and high complexity configuration, respectively.

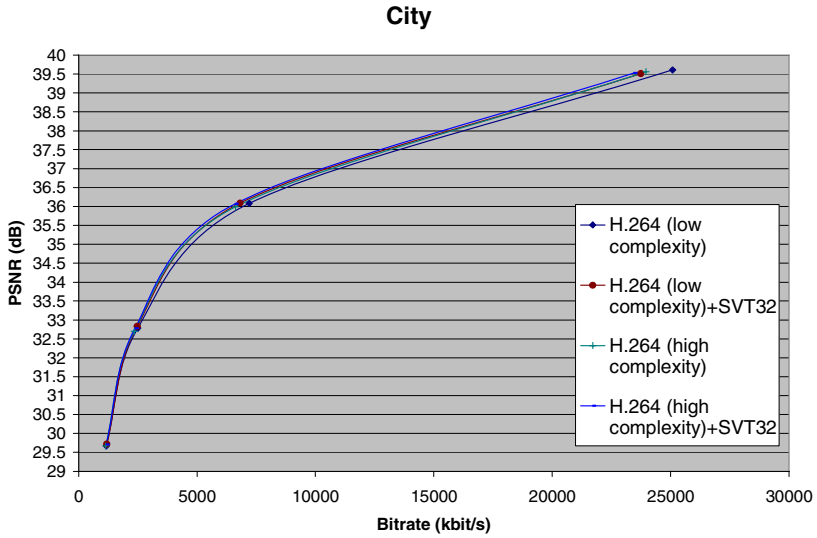


Fig. 5. R-D curve for City sequence

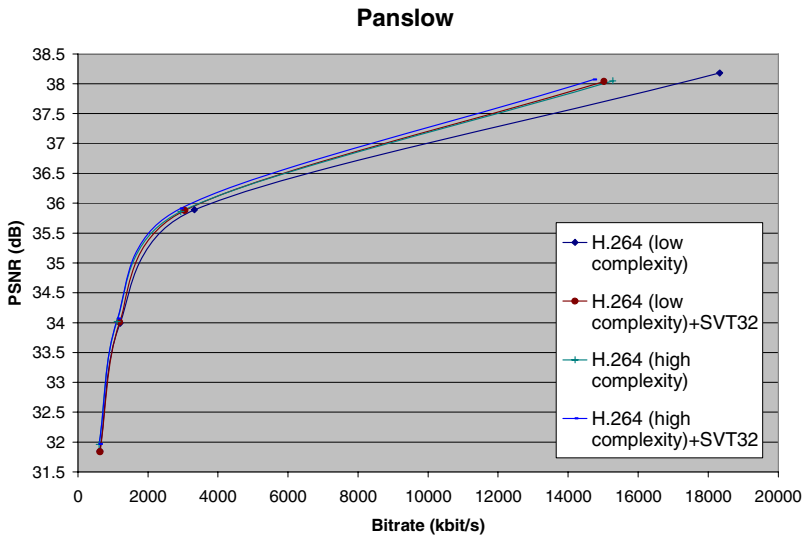


Fig. 6. R-D curve for Panslow sequence

5 Conclusions

In this paper, we propose a novel algorithm, named as Spatially Varying Transform (SVT). The basic idea of SVT is that we do not restrict the transform coding inside normal block boundary 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 rate distortion tradeoff. The proposed algorithm is implemented and studied in H.264/AVC framework. Two key issues of SVT: selection of location parameter candidates and filtering of block boundaries, are addressed in detail. It is shown that a configuration of the proposed algorithm achieves on average 2.64% bit-rate reduction in low complexity configuration which represents a low complexity codec with most effective tools for HD video coding and on average 1.42% bit-rate reduction in high complexity configuration which represents a high complexity codec with full usage of the tools provided in the standard, respectively, compared to H.264/AVC. Gains become more significant at high bit-rates and the bit-rate reduction can be up to 10.22%, which makes the proposed algorithm very suitable for future video coding solutions focusing on high fidelity applications. The decoding complexity is expected to be decreased because only a portion of the prediction error needs to be decoded.

Future studies include: 1) Better configuration of the proposed algorithm to achieve better performance, e.g., by selecting different portions inside a macroblock. 2) Better encoding algorithms especially in motion estimation and macroblock partition decision to generate residuals more suitable to code for the proposed algorithm. 3) Encoding algorithms to reduce the encoding complexity.

References

1. Ostermann, J., Bormans, J., List, P., Marpe, D., et al.: Video Coding with H.264 / AVC: Tools, Performance, and Complexity. *IEEE Circuits and Systems Magazine* 4(1), 7–28 (2004)
2. Zeng, B., Fu, J.: Directional discrete cosine transforms – A new framework for image coding. *IEEE Trans. Circuits Syst. Video Technol.* 18(3), 305–313 (2008)
3. Advanced video coding for generic audiovisual services, ITU-T Recommendation H.264 (March 2005)
4. KTA reference model 1.8m, <http://iphone.hhi.de/suehring/tml/download/KTA/jm11.0kta1.8.zip>
5. Bjontegaard, G.: Calculation of average PSNR differences between RD-curves, VCEG Doc. VCEG-M33 (March 2001)
6. Pateux, S., Jung, J.: An excel add-in for computing Bjontegaard metric and its evolution, VCEG Doc. VCEG-AE 2007 (January 2007)