AN OPTIMALLY SCALABLE AND COST-EFFECTIVE FRACTIONAL-PIXEL MOTION ESTIMATION ALGORITHM FOR HEVC

[1]*Huang LI,* [2]*Yihao ZHANG,* [1]**Hongyang CHAO*

[1] School of Software & [2] School of Info. Sci. and Tech., Sun Yat-sen University, China 510275

ABSTRACT

Fractional-pixel motion compensation is still one of the most time-consuming parts in the upcoming High Efficiency Video Coding (HEVC) standard. In this paper, we propose an optimally scalable and cost-effective fractional-pixel motion estimation (FPME) algorithm to optimally fit to different and varying constrains of computing resources. Our main contribution include two aspects. Firstly, an optimally scalable and cost-effective FPME algorithm based on a cost-benefit analysis is proposed, where we present a improved fractional-pixel MV prediction method and a new cost-effective priority for each search point in HEVC. Secondly, a complexity adjustment strategy is delivered to enable the ability for the FPME to adjust its complexity to match different given constraints on time. Experiments show that the proposed algorithm can achieve best R-D performance while optimally adjust its complexity, based on any given time constraints. As a side product, the proposed algorithm can also serve as the best fast algorithm which has already reduced computing complexity by a factor 74% with almost no loss on PSNR and bitrates.

Index Terms— motion estimation, fractional pixel, cost-effective priority, video coding, HEVC

1. INTRODUCTION

High Efficiency Video Coding (HEVC) [1] [2] is the next-generation video coding standard which is currently under development. Aiming to improve the coding efficiency, more and more technologies are incorporated in HEVC. But at the same time the encoder complexity is greatly increased. Fractional-pixel motion compensation, according to our analysis, is the most time-consuming process for the encoder, which occupies 49% of the total encoding time on average. Hence it is important to optimize the fractional-pixel motion compensation for the overall encoder performance.

In practice, available computational resources for encoder are often constrained and varying, especially for portable devices or real-time visual communications. Moreover, different devices are likely to have different computing capacities. Even for the same device, the available computing capacity for video encoder varies from time to time because of multitasking. How to adjust the encoder complexity adaptively with best R-D performance under constraint and varying computing resources is important for video encoder in practice. Therefore, researches on optimally scalable and cost-effective algorithms, which can adaptively control their speeds while keeping optimal performance under arbitrary computational resources constraint, would likely be a must. This paper is just focusing on optimally scalable algorithms for FPME.

Currently there are not many researches on HEVC aiming to propose scalable algorithms. In the related results on H.264, for the most time-consuming fractional-pixel motion estimation (FPME), some fast algorithms have been proposed. These fast FPME algorithms can be classified into two categories. The first category is the model-based algorithms [3]-[5]. These algorithms establish a mathematical model for fractional-pixel error surface and predict the optimal MV by finding the minimal of the model. Take Hill's method [3] for example, it tries to use a quadratic surface to fit the subpixel SAD surfaces. But the problem is that not all actual SAD surfaces can be well fitted. Moreover, these model-based algorithms is not scalable. That means we are unable to get better R-D performance if we have more time. The second category is the neighboring-MV based algorithms [6]-[10], which derive a predicted fractional-pixel MV as start position and then refine the MV by a small pattern such as diamond pattern. For these algorithm, almost all of the time is spent on refinement search. However, these algorithms are not designed to be optimally scalable. Optimally scalable means that it can effectively utilize every piece of time to achieve as much R-D gains as possible. The main reason for this defect is that different sub-pixels have quite different cost on interpolation, especially in HEVC, but most fast FPME algorithms ignore it.

Actually, a fast algorithm based on cost-effective concept has been proposed for H.264 [11]. This method belongs to the neighboring-MV based algorithms. However, it remedy the inefficiency in refinement search by a cost-effective search order. The search order is defined according to cost-performance ratios, by simultaneously considering the interpolation costs and probabilities of being optimal match point for fractional pixels. It was proved that it can outperform the other existing fast algorithms in H.264. However, since HEVC introduced quite a different interpolation filter, we cannot directly apply the method in [11] to HEVC. Furthermore, algorithm in [11] does not have the mechanism to automatically adjust the complexity of FPME to match a given time for FPME, which is essential to solve above problems. Therefore, it is basically still a fast algorithm which may not fit to different constrains on computing resources for different video contents.

In this paper, we try to solve all the problems mentioned above and thus proposed an optimally scalable and cost-effective FPME algorithm for HEVC. Differing from most fast algorithm, a special search order is adopted to check the sub-pixels with higher cost-effective priority first. Moreover, the proposed algorithm will actually has the ability to automatically adjust its complexity in order to match any given FPME time on the fly.

There are two major contributions in this paper: the first one is that we address an improve version of the optimally scalable and cost-effective algorithm by introducing a better fractional motion

This work was partially supported by NSF of China under Grant 61173081 and Guangdong Natural Science Foundation, P.R.China, under Grant S2011020001215.

*Corresponding author: Hongyang CHAO (isschhy@mail.sysu.edu.cn)

vector prediction method which increases 10% prediction accuracy , and delivering a new priority table for refinement search according to the new interpolation process in HEVC. The second one is that we propose a strategy to automatically adjust the complexity of FPME, which is just a problem and shortage not solved by [11]. After this, we can really have an optimally scalable and cost-effective FPME to fit the different constrains on computing resources.

The rest of this paper is organized as follows. Section 2 introduces our first contribution on addressing an improve version of the optimally scalable and cost-effective algorithm for HEVC. Section 3 presents our second contribution on automatic FPME complexity adjusting strategy. Experimental results and conclusions are given in section 4 and section 5 respectively.

2. OPTIMALLY SCALABLE AND COST-EFFECTIVE ALGORITHM FOR HEVC

In this section, we will introduce the first contribution in this paper: an improved version of the optimally scalable and cost-effective algorithm for HEVC. It includes two major improvements compared with the one in [11]. The first improvement is that we bring a better fractional-pixel MV prediction method which increases 10% prediction accuracy. And the second improvement is that we deliver a new priority table for refinement search according to the new interpolation process in HEVC.

Our proposed algorithm belongs to the second category of fractional-pixel motion estimation (FPME) algorithms, i.e. the neighboring-MV based algorithms, which consist of a fractional-pixel motion vector (FPMV) prediction and a refinement search.

2.1 The improved FPMV prediction method

According to our analysis, the accuracy of the fractional-pixel MV prediction method in [11] is no more than 30% for sequence with intense motion in HEVC. In addition, we have following observation: the fractional-pixel part of the MV predictor which is obtained by AMVP [2] method has a high correlation with the final optimal fractional-pixel MV. However, it is infeasible to get the MV predictor before FPME, because the MV predictor is selected from the MV predictor candidate list [2] as the one closest to the final optimal MV which is obtained after integer-pixel and fraction-pixel motion estimation. In order to solve this dilemma, we select a 'rough' MV predictor $\mathrm{MV}_{\mathrm{rough}}$ from the candidate list as the one closest one to $\mathrm{MV}_{\mathrm{integer}}$ which is obtained by integer-pixel motion estimation. The fractional-pixel part of $\mathrm{MV}_{\mathrm{rough}}$ is then used as the predicted FPMV, which can be extracted by the formula below:

$$\mathrm{MV}_{\mathrm{frac\_pred}} = \left(\mathrm{MV}_{\mathrm{rough}} - \mathrm{MV}_{\mathrm{integer}}\right) \bmod 4 \qquad (1)$$

where $\mathrm{MV}_{\mathrm{frac\_pred}}$ is the predicted FPMV that we used as a start position in our proposed FPME algorithm.

To prove that the proposed predicted method is better, the proposed predicted method is compared with the one in [11]. In the experiments, we use the final fractional-pixel motion vector (FPMV) by Full Fractional Pixel Search (FFPS) as a baseline to evaluate the accuracy of the predictors obtained by our proposed method and the one in [11].

The experiment has been performed on 20 sequences in Table 1. Table 2 shows that how many percentages of the predicted FPMVs are exactly the final FPMVs obtained by FFPS. We can see that the proposed prediction method improve the prediction accuracy by 10% on average. Especially for the sequences with intensive motion in class A and B, the proposed prediction method improves prediction accuracy by 15%-20%.

Table 1. Sequences for analyzing the accuracy of predicted fractional-pixel motion vector

Class	Resolution	Sequence
A	2560x1600	PeopleOnStreet, Traffic
B	1920x1080	BasketballDrive, BQTerrace, Cactus, Kimono1, ParkScene
C	832x480	BasketballDrill, BQMall, PartyScene, RaceHorses
D	416x240	BasketballPass, BlowingBubbles, BQSquare, RaceHorses
E	1280x720	Vidyo1, Vidyo3, Vidyo4

Table 2. Prediction accuracy comparison

Class	Predicted FPME equals final FPME		
	proposed	method in [11]	accuracy improvement
A	45.81%	29.84%	15.97%
B	47.65%	23.28%	24.37%
C	40.76%	40.07%	0.69%
D	33.65%	32.91%	0.74%
E	75.57%	67.60%	7.97%
Average	48.69%	38.74%	9.95%

2.2 The new cost-effective refinement search order for HEVC

The cost-effective algorithm in [11] gives a refinement search order for the fractional-pixel search points in the same refinement search pattern. But we cannot directly apply the order to HEVC because the new interpolation filter in HEVC changes it. The search order is relative to the cost-performance ratios of every fractional-pixel search point. The search point with highest performance and lowest cost will be searched first. Here we will give a clear definition to the cost-performance ratio of each fractional pixel, which is implicit in [11]. The cost-performance ratio of the fractional pixel at (x, y) is defined as:

$$\mathrm{CP}_{(\mathrm{x,y})} = \mathrm{Cost}_{(\mathrm{x,y})} / \mathrm{Probability}_{(\mathrm{x,y})} \qquad (2)$$

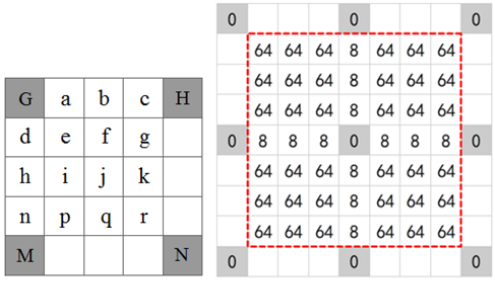
where $\mathrm{Probability}_{(\mathrm{x,y})}$ is the probability of being the best MV for search position (x, y). $\mathrm{Cost}_{(\mathrm{x,y})}$ is the computational complexity of interpolation for the point (x, y).

However, the new 8-tap DCT-based interpolation filter in HEVC changes the interpolation costs and probabilities, which makes the cost-performance ratios different from those in H.264. Hence the search order is different form that in [11].

In order to update the cost-effective refinement search order for HEVC, we need to calculate the cost-performance ratio define in formula (3). It involves two aspects of work: obtain the new cost table and the new probability table for each fractional-pixel search position in HEVC.

We start with the new cost table. All 1/4 accuracy fractional pixels are depicted in Fig. 1(a). For better description of the new interpolation process in HEVC, we classify those fractional pixels into two categories based on their computational complexities on interpolation. The first category includes 6 points: a, b, c, d, h and n. All these points are interpolated from 8 integer pixels horizontally or vertically. The second category includes 9 points: e, f, g, i, j, k, p, q and r. These points are interpolated from 8 fractional pixels in the first category. For example, if we want to interpolate the point e, we must interpolate 8 extra fractional pixels

in advance as an intermediate step. That is to say, the computational complexity of the points in the second category is 8 times that of the points in the first category. The cost to generate each fractional pixel is shown in Fig. 1(b), which is quite different from the one for H.264 in [11].

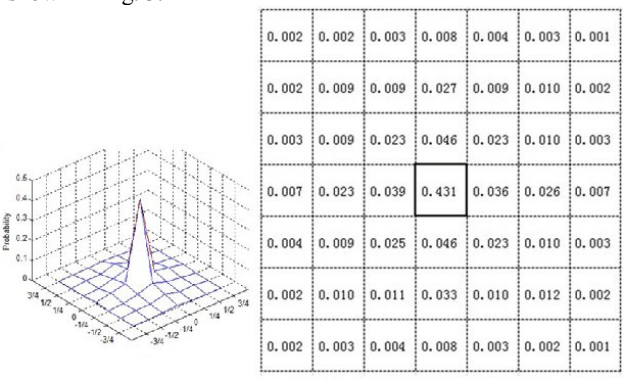


(a) (b)

Fig. 1. (a) Integer pixels (blocks with upper-case letter) and fractional pixels (blocks with lower-case letter). (b) Computational complexity of interpolation for fractional pixels.

A new interpolation method will generate different fractional pixel values, which may lead to the differences between the best match points in HEVC and H.264. To ensure we get the correct probabilities for each fractional pixel of being best match point, we update the probability table for HEVC, based on the distribution of optimal fractional-pixel MV (FPMV) of the sequences in Table 1. The new probability table is shown in Fig. 2.

Combining the new cost table in Fig. 1(b) and the new probability table in Fig. 2, we can recalculate the cost-performance ratio for each search point by formula (2) and thus give a new search order, i.e. search priority. The new search order for HEVC is shown in Fig. 3.



(a) (b)

Fig. 2. The distribution of optimal fractional-pixel MV (FPMV) of the sequences in Table 1. (a) Graph (b) Table

2.3 Algorithm description

Step 1: Choose start search point with smaller R-D cost between the predicted position and integer-pixel position.

Step 2: Refinement search around the start search point.

Step 2.1: In the diamond pattern, take one search point that have the smallest value according to Fig. 3 as current search point.

Step 2.2: Interpolate before checking current search point.

Step 2.3: Match the current search point.

Step 2.4: If current search point have smaller cost than the center point, then the opposite search point is skipped (based on the error surface unimodal assumption in [7]).

Step 2.5: Go back to Step 2.1 until all search points in the current diamond pattern are checked.

Step 3: If the optimal search point is in the center of the diamond pattern, the search stops. Otherwise, the diamond pattern moves to a new center and go back to Step 2.1.

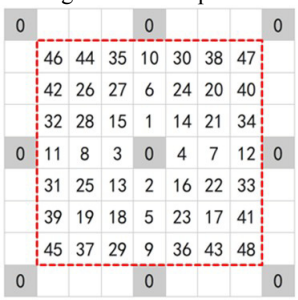


Fig. 3. Cost-effective priority table for fractional-pixel search points in HEVC (smaller value means higher priority).

3. FPME COMPLEXITY ADJUSTING STRATEGY

In section 2, we have proposed an optimally scalable and cost-effective fractional-pixel motion estimation (FPME) algorithm for HEVC. However, if we cannot find a FPME complexity adjusting strategy to fit to any given FPME time, it is basically still a fast algorithm which may not fit to different constrains on computing resources and different video contents. In this section, we will introduce our second contribution: a complexity adjustment strategy. It can precisely control and optimally allocate the time consumed by the proposed FPME, in order to match arbitrary time constraints and achieve best R-D performance.

To develop such a complexity adjusting strategy, there are two questions needed to be answered: the first one is how to control the time consumed by FPME; the second one is how to optimally allocate the given time among different blocks with different motion intensity to achieve best R-D performance. For example, more time should be allocated a active block rather than a still block. Generally, there are two ways for early termination:

(1) Use a fixed number of search points for each cycle of FPME on each block. This method can control the time consumed, but it is unable to optimally allocate the time because it will waste too much time on still blocks.

(2) Use a fixed threshold for R-D cost, i.e. SATD plus the bits used by MV. This method can optimally allocated the resources among different blocks, but the time consumed is not controllable because active video sequences will consume more time to reach the threshold. That is to say, for a given FPME time, different video contents require different threshold values.

From the analysis above, both two methods are unable to solve the two questions at the same time. Therefore, we proposed a dynamic threshold on R-D cost with a automatic adjusting mechanism. The basic idea is to adjust the threshold by learning the relationship between the threshold and FPME time in previous encoded frames.

The dynamic threshold *th* is defined as follow: if the average R-D cost over all pixels in one block is smaller than *th*, search stops.

$$\mathrm{Cost}/(\mathrm{P_{width}} * \mathrm{P_{heigth}}) < th \qquad (3)$$

where $\mathrm{P_{width}}$ and $\mathrm{P_{heigth}}$ are the width and height of current PU, respectively.

The threshold *th* is adjusted frame by frame: assuming the given time on FPME for certain sequence is $\mathrm{T_{total}}$, the sequence

has N frames. Firstly we dynamically allocate the time for each frame by formula (4).

$$T^{i}_{frame,alloc} = \frac{T_{total} - \sum_{j=0}^{i-1} T^{j}_{frame,used}}{N-i} \quad (4)$$

where $T^{i}_{frame,alloc}$ is the time allocated for the FPME in *i*th frame, and $T^{j}_{frame,used}$ is the actual time used by the FPME in *j*th frame. Then we adjust the threshold for each frame by formula (5).

$$th^{i+1} = th^{i} * (T^{i}_{frame,alloc} / T^{i}_{frame,used}) \quad (5)$$

By applying this adjusting mechanism, if the actual time spent by FPME for one frame $T^{i}_{frame,alloc}$ is larger than expected $T^{i}_{frame,used}$, the threshold will increase, or decrease otherwise. Based on the FPME time given , *th* will finally automatically get a balanced value for different video contents.

4. EXPERIMENTAL RESULT

In order to evaluate the performance of the proposed FPME algorithm and complexity adjustment strategy, we have three experiments in this section.

All the experiments were run on Windows Server 2003 with Intel Xeon E5420 at 2.50GHz. We have implemented the proposed algorithm based on HM-4.0 (latest HM version has the same FPME structure) and the coding structure is IPPP with QP = 32. The test sequences are the 20 official test sequences in Table 1 [12].

A. Optimal scalability and cost-effectiveness

In order to evaluate the improved optimally scalable and cost-effective algorithm proposed in Section 2, we record the average R-D cost on every number of search points for several algorithms. The results are shown in Fig. 4. CBFPS is a widely used algorithm in [7]. RFSME is the algorithm in [10]. The curves in Fig. 4 show that our proposed algorithm can achieve the least R-D costs, i.e. best R-D performances, at an arbitrary number of search points. Hence whenever the complexity adjusting strategy stop the proposed algorithm, it will achieve the best R-D performance.

B. Testing the automatic complexity adjustment strategy

In order to show the scalability of the encoder, which is not implemented in [11], we give a target FPME time T_{target} to the encoder. By using the proposed FPME algorithm in section 2 and the automatic adjusting strategy in section 3, the FPME algorithm can automatically fit to the given time T_{target}, as shown in Table 3. The results show that the automatic complexity strategy can control the time consumed by FPME, with 0.50% error on average.

C. Using the proposed algorithm as a fast algorithm

As a side product, the proposed algorithm is also the best fast algorithm if we set a fixed value to the threshold *th* in section 3. Comparing with Full Fractional Pixel Search (FFPS), the time spent on fractional-pixel motion estimation is reduced by a factor of 54% on average. The average PSNR loss and bitrate increment are 0.01 and 0.21% respectively. ΔPSNR, ΔBitrate and ΔTime in Table 4 are calculated as follow:

$$\Delta PSNR = PSNR_{proposed} - PSNR_{FFPS}$$
$$\Delta Bitrate = (Bitrate_{proposed} - Bitrate_{FFPS}) / Bitrate_{FFPS} \quad (6)$$
$$\Delta Time = (Time_{proposed} - Time_{FFPS}) / Time_{FFPS}$$

As shown in Table 5, the average number of search points for each FPME is only 4.5 when keeping almost the same performance with FFPS. By contrast, the number of search point is 10 for CBFPS [9].

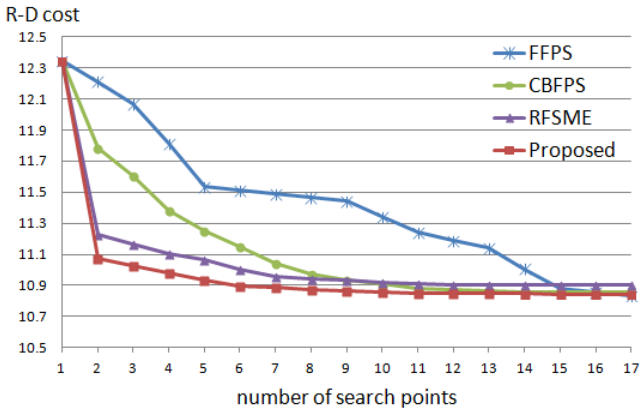


Fig. 4. Comparison of R-D costs with other algorithms on each search point

Table 3. Complexity adjusting result

Class	Time spent on FPME	Error on time control
A	99.97% * T_{target}	0.03%
B	100.53% * T_{target}	0.53%
C	100.68% * T_{target}	0.68%
D	100.68% * T_{target}	0.68%
E	100.63% * T_{target}	0.63%
Average	100.50% * T_{target}	0.50%

Table 4. Comparison of PSNR, bit-rate and time with FFPS

Class	Average ΔPSNR	Average ΔBitrate	Average FPME ΔTime
A	-0.01	0.24%	-51.85%
B	-0.01	0.14%	-54.66%
C	-0.01	0.12%	-43.87%
D	-0.01	0.13%	-37.29%
E	-0.02	0.48%	-81.91%
Average	-0.01	0.22%	-53.92%

Table 5. Comparison of Search Points(SPs) with FFPS

Class	Average Search Points		
	Proposed	FFPS	Saved SPs
A	4.32	16	-74.61%
B	4.32	16	-74.60%
C	5.22	16	-69.29%
D	5.80	16	-65.91%
E	2.24	16	-86.82%
Average	4.50	16	-73.52%

5. CONCLUSIONS

This paper proposed a optimally scalable fractional-pixels motion estimation algorithm based on cost-effective approach for HEVC, where we have two major contributions.

According to our experimental results, the proposed algorithm can outperform existing fast algorithms at any complexity constrain. Moreover, the proposed algorithm can automatically adjust its complexity to match different given time for FPME with optimal R-D gain.

6. REFERENCES

[1] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, and T. Wiegand, "WD4: Working Draft 4 of High-Efficiency Video Coding," Document of Joint Collaborative Team on Video Coding, JCTVC-F802, July 2011.

[2] K. McCann, K. McCann, S. Sekiguci, B. Bross, and W.-J. Han, "HM4: High Efficiency Video Coding (HEVC) Test Model 4 Encoder Description," Document of Joint Collaborative Team on Video Coding, JCTVC-F802, July 2011.

[3] P. Hill, T. K. Chiew, D. Bull, and N. Canagarajah, "Interpolation free subpixel accuracy motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 12, pp. 1519-1526, Dec. 2006.

[4] S. Dikbas, T. Arici, and Y. Altunbasak, "Fast motion estimation with interpolation-free sub-sample accuracy," IEEE Trans. Circuits Syst. Video Technol., vol. 20, no. 7, pp. 1047-1051, July 2010.

[5] Q. Zhang, Y. Dai, and C. Kuo, "Direct techniques for optimal subpel motion resolution estimation and position prediction," IEEE Trans. Circuits Syst. Video Technol., vol. 20, no. 12, pp. 1735-1744, Dec. 2010.

[6] Y.-J. Wang, C.-C. Cheng, and T.-S. Chang, "A fast algorithm and its VLSI architecture for fractional motion estimation for H.264/MPEG-4 AVC video coding," IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 5, pp. 578-583, May 2007.

[7] Z. Chen, J. Xu, Y. He, and J. Zheng, "Fast integer-PEL and fractional-PEL motion estimation for H.264/AVC," J. Vis. Commun. Image Representation, vol. 17, no. 2, pp. 264–290, 2006.

[8] H. Chao and J. Lu, "A high accurate predictor based fractional pixel search for H.264," in Proc. IEEE Int. Conf. Image Process., pp. 2365-2368, Oct. 2006.

[9] Libo Yang, Keman Yu, Jiang Li and Shipeng Li, "Prediction-based directional fractional pixel motion estimation for H.264 video coding," ICASSP 2005, vol. 2, pp. ii/901-ii/904, Mar. 2005.

[10] Weiyao Lin, Panusopone, K., Baylon, D.M.; Ming-Ting Sun, Zhenzhong Chen, Hongxiang Li, "A Fast Sub-Pixel Motion Estimation Algorithm for H.264/AVC Video Coding," Circuits and Systems for Video Technology, IEEE Transactions on , vol.21, no.2, pp.237-242, Feb. 2011.

[11] Jiyuan Lu, Peizhao Zhang, Hongyang Chao and Fisher, P.S., "On Combining Fractional-Pixel Interpolation and Motion Estimation: A Cost-Effective Approach," IEEE Trans. Circuits Syst. Video Technol., vol. 21, no. 6, pp. 717-728, June 2011.

[12] Frank Bossen, "Common test conditions and software reference configurations," Document of Joint Collaborative Team on Video Coding, JCTVC-F900, July 2011.