Hash Based Fast Local Search for Intra Block Copy (IntraBC) Mode in HEVC Screen Content Coding

Sik-Ho Tsang, Yui-Lam Chan*, and Wan-Chi Siu
Centre for Signal Processing, Department of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong.
E-mail: {sik-ho.tsang, *enylchan, enwcsiu}@polyu.edu.hk *Tel: +852-27666213

***Abstract*— The Intra Block Copy (IntraBC) mode is a very efficient coding tool for the screen content coding (SCC) extension in High Efficiency Video Coding (HEVC) due to the frequently occurrences of repeating patterns and noiseless characteristics of screen content videos. There are local search and hash search for the IntraBC mode to increase coding efficiency. However, the IntraBC mode also brings along high computational complexity for SCC as exhaustive block matching is done within the same frame. Though there are already some constraints applied to IntraBC mode to reduce its complexity, it is still very high. To further reduce the complexity, we propose to speed up the local search by checking the hash values of both current block and block candidates. With our proposed methods, the encoding time is reduced by up to 25% and 23% for YUV and RGB sequences respectively while coding efficiency can still be maintained with only negligible bitrate increased.**

I. INTRODUCTION

With the evolution of thin-client devices as well as cloud technology, computer screen sharing applications such as virtual desktop and video conferencing with slideshows sharing have become widespread nowadays [1]. Screen content coding (SCC) is highly demanded and has been introduced as an extension of High Efficiency Video Coding (HEVC) [2] for handling the screen content videos generated by those computer screen sharing applications. As a result, SCC for limited network bandwidth has emerged as one of the hot research topics in the aspect of video coding.

Screen content videos are the videos with the compound of camera-captured contents and computer generated contents such as texts and graphical user interface. The camera-captured contents have already been efficiently encoded by HEVC already. However, the characteristics of computer generated contents are different from those of camera-captured contents such as complex structure, sharp edges with high contrast, noiseless smooth regions and repeating patterns. HEVC cannot handle these computer generated contents well. Therefore numerous coding tools have been suggested for SCC. Complex structure and sharp edges with high contrast are well handled by palette mode [3-4] whereas noiseless smooth regions can be handled by single intra mode [5]. It is noted that there were research works [6-7] for coding smooth regions, their purpose is only for camera-captured contents and depth maps where smooth blocks always contain camera noise or depth estimation noise which is not the case in SCC. In this paper, we would focus on Intra Block Copy (IntraBC) mode [8] which is one of the SCC coding tools used for finding repeating patterns within the same frame.

IntraBC was firstly introduced in [8] to find the repeating pattern within the same frame by performing intra motion estimation (ME) and intra motion compensation (MC). It is considered as an additional mode besides the conventional HEVC intra mode in intra coding. If IntraBC is used by one particular coding unit (CU), each prediction unit (PU) within the CU is encoded with an intra motion vector (MV), or equivalently block vector (BV), as well as the residual signal of that CU. The experimental results shown in [8] for IntraBC are very impressive that bitrates are largely reduced. This means that there are plentiful repeating patterns within the same frame. However, computational complexity for the encoder is also increased essentially because exhaustive block matching for every PU candidate is done during intra ME and MC. As a consequence, the technique in [8] has suggested several constraints for balancing the coding performance and coding efficiency. First, it is suggested that MV/BV has to be integer-pel. It means that there are no fractional-pel ME and MC but only integer-pel ME and MC are performed. With this restriction, interpolation process can then be skipped. Second, local search is recommended that only the area near the current PU to be encoded is searched. Last, ME and MC for IntraBC is only applied for small CU sizes of 16×16 and 8×8 as repeating patterns usually appeared for small CU size rather than large CU size. [9] proposed to further speed up the encoding process by skipping the testing of IntraBC based on the mode of the parent CU and the CU activity. However, if the matched repeating pattern is out of the local search area, the repeating pattern cannot be located. On the other hand, if full-frame search is used, the encoder complexity would be impractically high. Hence, in addition to the local search, there is also a full-frame hash search suggested in the IntraBC mode [10].

For the full-frame hash search, block matching is only done for those CU candidates which have the same hash value as the current CU to be encoded. Zhu *et al.* [11-12] suggested to estimate the hash value based on the DC value of the CU as well as the number of color transitions along the row and column while Li *et al.* [13] estimate the hash value based on the Cyclic Redundancy Check (CRC) value of the CU to find the repeating patterns in the hash search. In this paper, we will review the hash value estimated in [10] as it is used in the

The work described in this paper is partially supported by the Centre for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, and a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Grant No. PolyU 152016/14E).

HEVC reference software Model version 16.2 and the Screen Content Model version 3.0 (HM-16.2+SCM-3.0, hereafter called SCM-3.0 for the sake of simplicity) [14] for the IntraBC mode . In fact, the hash values calculated by [11-13] can also be adopted into our proposed approach for fast local search. It is noted that the full-frame hash search in SCM-3.0 is used for 8×8 CU with 2N×2N PU only with consideration of the tradeoff between computational complexity and coding performance while our proposed algorithm is going to support any PU sizes. In this paper, we further describe the local and hash searches in details in Section II. Next, complexity analysis for the local and hash searches is discussed in Section III. Our fast local search utilizing the hash value estimated in the hash search is then proposed in Section IV. Finally, experimental results are shown in Section V with conclusions drawn in Section VI.

II. Conventional Local and Hash Searches

A. Local Search

Different CU and PU sizes have different search strategies in the IntraBC mode. Fig. 1 illustrates the search areas for various searching strategies. For 16×16 CU, only 2N×2N PU with full vertical and horizontal searches are performed. It is due to the fact that large CU size tends to have fewer repeating patterns found within the same frame. For 8×8 CU, there are different PU sizes for block matching in the IntraBC mode because 8×8 CU, which has a smaller CU size compared with 16×16 CU, is more likely to find the repeating pattern. Then, for 8×8 CU, if it is 2N×2N, 2N×N, or N×N PU, local vertical, local horizontal and 2D searches within the left coding tree unit (CTU) and the current CTU are performed. If it is N×2N PU, only full vertical and full horizontal searches are performed. And it is noted that before doing the above searches, predictors are tested for all sizes of CU and PU where predictors are generated by the last two coded IntraBC MV/BV and default MV/BV.

B. Hash Search

As aforementioned in the previous section, for hash search, block matching is only done for those CU candidates which have the same hash value as the current CU to be encoded. And the hash search is only performed for 8×8 CU with 2N×2N PU.

The hash value is a 16-bit value calculated based on the DC of four sub-partitions of the CU, DC_k with $k=0,1,2,3$, and gradient of the whole CU, *Grad*, as follows:

$$DC_k = \frac{1}{4\times 4}\sum_{j=0}^{3}\sum_{i=0}^{3} p(i,j)$$
$$Grad = \sum_{j=1}^{7}\sum_{i=1}^{7}\frac{|p(i,j)-p(i-1,j)|+|p(i,j)-p(i,j-1)|}{2} \quad (1)$$

where $p(i,j)$ is the sample value at the corresponding position (i,j) within the CU. And the hash value is formed by concatenating DC_k with $k=0,1,2,3$, and *Grad* as follows:

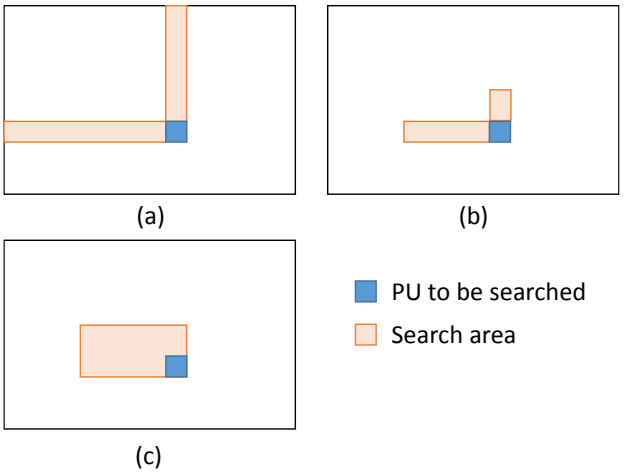


Fig. 1 Illustrations of (a) full vertical and horizontal searches, (b) local vertical and horizontal searches, and (c) local 2D search.

$$Hash = (MSB_3(DC_0) \ll 13) + (MSB_3(DC_1) \ll 10) + (MSB_3(DC_2) \ll 7) + (MSB_3(DC_3) \ll 4) + Grad \quad (2)$$

where $\ll$ represents the left shift operation and $MSB_m(X)$ means that m most significant bits for X are obtained. In this case, $m=3$. By (2), the hash value is formed for each 8×8 CU candidate. Thus, once hash search is used, block matching is only done for those searching points with the same hash value as the current CU. After that, hash values of all the newly searching points within the reconstructed area are estimated and can be re-used for the coming CUs. By this mean, CU candidates within the same frame but with different hash values from the current CU are filtered out. Full-frame search can be carried out with less computational complexity when the hash values are adopted.

III. Complexity Analysis of Local and Hash Searches

To perform block matching for one searching point, sum of absolute difference (SAD) between the CU candidate and the current CU, $SAD(x,y)$, as well as the cost of MV/BV, $MV/BVCost(x,y)$, are estimated based on the rate distortion (RD) cost, $RDCost(x,y)$, as follows:

$$RDCost(x,y) = SAD(x,y) + MV/BVCost(x,y) \quad (3)$$

where (x,y) is the position of an CU candidate. So SAD and MV/BV cost are estimated once for each searching point. And the one with minimum RD cost is selected as the optimal solution within the IntraBC mode. For MV/BV cost estimation, it occupies very small amount of time compared with SAD estimation as it is just simply estimated by table look-up plus few operations. The complexity of block matching is mainly come from the SAD estimation. Therefore, to analyze the complexity of local search and hash search, we collect the statistics of the number of searching points on the testing sequences with the first 100 frames encoded, quantization parameters (QP) {22, 27, 32, 37} and All Intra

(AI) configuration used which are the settings recommended in the Common Test Conditions (CTC) for SCC [15]. TABLE I tabulates the number of searching points in percentage for each sequence. It can be observed that the local search occupies over 90% in average within the IntraBC mode while the hash search only occupies less than 10% of the complexity. Hence, it is highly motivated to reduce the complexity for the local search.

IV. PROPOSED HASH BASED FAST LOCAL SEARCH

In order to reduce the computational complexity for the local search, we propose to reduce the number of searching points to speed up the local search by checking the hash values of the PU candidates and the current PU. If their hash values are not the same, those PU candidates most likely is not the repeating pattern of that current PU. Thus, RD cost estimated by (3) would be skipped. As a result, irrelevant search area is not searched and the encoding time can be reduced. Fig. 2 illustrates the idea for our proposed hash based fast local search using the Programming sequence. In this figure, the blue rectangle is the current PU while the orange rectangles are the corresponding search areas. We can see that large amount of searching points can be skipped if we first check the hash values of PU candidates and the current PU.

Yet we cannot directly apply the hash value estimated by (1) and (2) into the local search for checking whether the hash values of the PU candidates and the current PU are the same. One of the reasons is that the conventional hash search is used for 8×8 CU with 2N×2N PU only. On the other hand, the local search can be used for 16×16 CU with 2N×2N PU as well as 8×8 CU with 2N×2N, 2N×N, N×2N, and N×N PU. The second reason is that the shorter the distance between the PU candidate and the current PU, the higher the tolerance of the distortion can be accepted. In other words, we can have larger SAD when MV/BV Cost is low, as shown in (3). Hence, we need to have different hash values for different PU sizes with relaxed constraint. The following shows our proposed hash value used for speeding up the local search for 8×8 CU with 2N×2N PU, the hash value is:

$$Hash_{2N\times 2N} = (MSB_m(DC_0) \ll 3m) + (MSB_m(DC_1) \ll 2m) + (MSB_m(DC_2) \ll m) + MSB_m(DC_3) \quad (4)$$

For 8×8 CU with 2N×N PU, the hash value is:

$$Hash_{2N\times N} = (MSB_m(DC_k) \ll m) + MSB_m(DC_{k+2}) \quad (5)$$

where k can be 0 or 1. Similarly, for 8×8 CU with N×2N PU, the hash value is:

$$Hash_{N\times 2N} = (MSB_m(DC_k) \ll m) + MSB_m(DC_{k+1}) \quad (6)$$

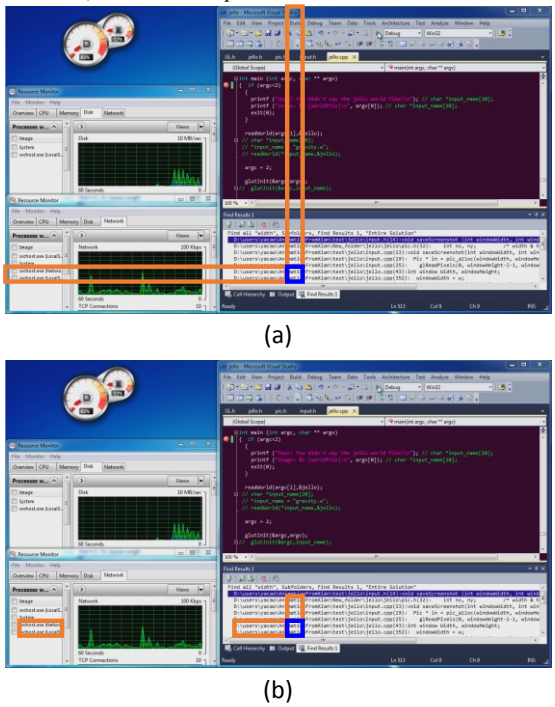
where k can be 0 or 2. Lastly, for 8×8 CU with N×N PU, the hash value is:

$$Hash_{N\times N} = MSB_m(DC_k) \quad (7)$$

TABLE I
NUMBER OF SEARCHING POINTS IN PERCENTAGE (%) OF LOCAL AND HASH SEARCHES WITH YUV AND RGB SEQUENCES

Sequence	Type*	YUV		RGB	
		Local	Hash	Local	Hash
BasketballScreen	M	90.81	9.19	90.06	9.94
MissionControlClip2	M	91.27	8.73	93.74	6.26
FlyingGraphics	TGM	85.54	14.46	88.66	11.34
Desktop	TGM	88.36	11.64	90.87	9.13
Console	TGM	81.66	18.34	77.85	22.15
MissionControlClip3	M	93.85	6.15	95.81	4.19
EBURainFruits	CC	96.65	3.35	97.37	2.63
Kimono1	CC	99.93	0.07	99.93	0.07
WebBrowsing	TGM	94.59	5.41	94.61	5.39
Map	TGM	84.91	15.09	77.09	22.91
Programming	TGM	95.82	4.18	95.50	4.50
SildeShow	TGM	98.53	1.47	97.87	2.13
Robot	A	97.69	2.31	95.50	4.50
Average		**92.28**	**7.72**	**91.91**	**8.09**

* TGM: Text and graphics with motion, M: mixed content
A: animation, CC: camera-captured content



(a)

(b)

Fig. 2 Illustrations of the idea of our proposed hash based fast local search using the Programming sequence where (a) shows the search area of the conventional full vertical and horizontal searches, and (b) shows the search area of the full vertical and horizontal searches using our proposed approach.

where k can be 0, 1, 2 or 3. With (4) to (7), we can adopt our proposed hash values to the local search by checking the hash values of PU candidates with different PU sizes. It is noted that there is no hash based fast local search for 16×16 CU as only 2N×2N PU is supported which occupies much less encoding time compared with 8×8 CU. In addition, gradient is not included in (4) to (7) so that more PU candidates can be included for RD cost estimation by (3). It is important especially when the PU candidates have a low MV/BV Cost.

V. Experimental Results

To evaluate the performance of the proposed algorithm, we perform simulations using the HEVC reference software SCM-3.0 [14] with the coding conditions already mentioned in Section III. Fig. 3 depicts the average Bjontegaard delta bitrate (BDBR) [16] and average encoding time of our proposed algorithm against the conventional SCM-3.0 without IntraBC in percentage (%) using various *m* values from 1 to 8. Note that *m*=0 actually means the conventional SCM-3.0 with IntraBC. From Fig. 3, it can be seen that with higher the value of *m*, smaller the increase in encoding time but with similar bitrate reduction. Thereby, *m* is suggested to be 3. Whether hash value is estimated in the conventional hash search by (3), the DC values can be stored directly before concatenation for estimating the hash values of (4) to (7) for our proposed hash based fast local search. Table II tabulates the BDBR and encoding time against the conventional SCM-3.0 in percentage (%) using *m*=3 for each testing sequence. From the table, it can be observed that the conventional SCM-3.0 obtains 25.23% and 24.06% bitrate reduction with 42.62% and 35.52% of encoding time increased for YUV and RGB sequences respectively whereas our proposed hash based fast local search can obtain 24.41% and 23.30% bitrate reduction with only 29.85% and 24.30% of encoding time increased for YUV and RGB sequences respectively. That means compared with SCM-3.0 with the IntraBC mode, our proposed approach can achieve about 10% encoding time reduction with bitrate only increased by less than 1% which is negligible.

VI. Conclusions

The Intra Block Copy (IntraBC) mode helps to increase the coding efficiency of HEVC screen content coding by finding the repeating patterns within the same frame using local and hash searches but encoding time is largely increased. By using our proposed hash based fast local search, checking of hash values between PU candidates and the current PU is done before RD cost estimation such that the encoding time is reduced by up to 25% and 23% for YUV and RGB sequences respectively compared with SCM-3.0 with IntraBC while bitrate is increased negligibly by less than 1% in average.

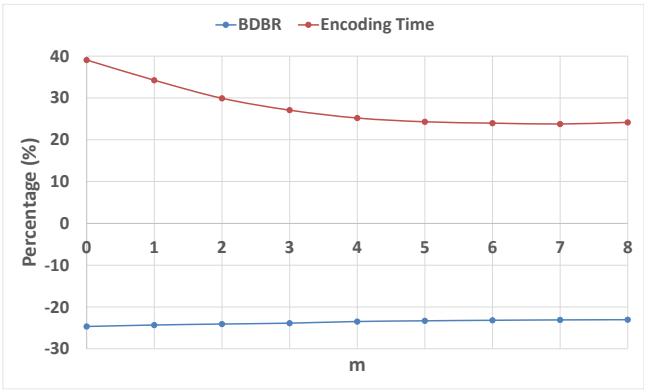


Fig. 3 BDBR and encoding time of our proposed algorithm against conventional SCM-3.0 without IntraBC in percentage (%) using various *m* values.

Table II
BDBR and Encoding Time of Proposed Algorithm against Conventional SCM-3.0 Without IntraBC in Percentage (%) Using m=3.

Sequence	YUV				RGB			
	SCM-3.0 with IntraBC		Proposed		SCM-3.0 with IntraBC		Proposed	
	BDBR	Time	BDBR	Time	BDBR	Time	BDBR	Time
BasketballScreen	-29.61	41.90	-28.89	27.70	-29.29	34.68	-28.54	22.02
MissionControlClip2	-32.59	38.19	-31.72	25.83	-31.38	31.45	-30.58	20.54
FlyingGraphics	-40.03	28.79	-37.86	17.31	-41.04	11.62	-38.76	2.69
Desktop	-55.45	6.22	-54.56	1.02	-54.03	-4.35	-53.24	-8.82
Console	-25.79	6.68	-24.68	1.63	-22.44	-6.35	-21.42	-9.58
MissionControlClip3	-44.17	34.28	-43.16	21.23	-43.49	25.77	-42.43	14.70
EBURainFruits	-0.34	84.76	-0.17	65.56	-0.34	84.76	-0.17	65.56
Kimono1	-0.08	69.71	-0.04	59.19	-0.08	69.71	-0.04	59.19
WebBrowsing	-51.84	11.78	-51.28	2.77	-47.09	0.32	-46.56	-6.27
Map	-7.84	95.14	-6.72	70.60	-5.44	93.15	-4.67	71.87
Programming	-29.10	28.42	-28.05	16.91	-30.15	15.05	-29.11	5.92
SildeShow	-9.42	20.02	-8.79	13.42	-6.77	16.64	-6.29	11.53
Robot	-1.78	88.19	-1.43	64.88	-1.30	89.27	-1.04	66.58
Average	**-25.23**	**42.62**	**-24.41**	**29.85**	**-24.06**	**35.52**	**-23.30**	**24.30**

References

[1] Y. Lu, S. Li, and H. Shen, "Virtualized screen: A third element for cloud-mobile convergence," *IEEE Multimedia*, vol. 18, no. 2, pp. 4–11, February 2011.

[2] G. J. Sullvian, J. Ohm, W.-J. Han, and T. Wiegand. "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649-1668, December 2012.

[3] Z. Ma, W. Wang, M. Xu, and H. Yu, "Advanced screen content coding using color table and index map," *IEEE Trans. Image Process.*, vol.23, no.10, pp. 4399-4412, October 2014.

[4] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Exploiting inter-layer correlations in scalable HEVC for the support of screen content videos," in *Proc. of Digital Image Process. (DSP)*, pp.888-892, Hong Kong, August 2014.

[5] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Fast and efficient intra coding technique for smooth regions in screen content coding based on boundary prediction samples," in *Proc. of IEEE International Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, pp. 1409-1413, Brisbane, Australia, April 2015.

[6] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Efficient intra prediction algorithm for smooth regions in depth coding," *IET Electronics Lett.*, vol.48, no.18, pp.1117-1119, August 2012.

[7] L.-L. Wang, and W.-C. Siu, "Novel adaptive algorithm for intra prediction with compromised modes skipping and signaling processes in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol.23, no.10, pp. 1686-1694, October 2013.

[8] M. Budagavi, and D.-K. Kwon, "Intra motion compensation and entropy coding improvements for HEVC screen content coding," in *Proc. of Picture Coding Symposium (PCS)*, pp. 365-368, San Jose, U.S.A., December 2013.

[9] M. Budagavi, and D.-K. Kwon, "Fast intra block copy (IntraBC) search for HEVC screen content coding," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 9-12, Melbourne, Australia, June 2014.

[10] J. Chen, Y. Chen, T. Hsieh, R. Joshi, M. Karczewicz, and W.-S. Kim *et al.*, "Description of screen content coding technology proposal by Qualcomm," *JCT-VC*, JCTVC-Q0031, Valencia, Spain, April-March 2014.

[11] W. Zhu, W. Ding, J. Xu, Y. Shi, and B. Yin, "2-D Dictionary Based Video Coding for Screen Contents," in *Proc. of Data Compression Conference (DCC)*, pp.43-52, March 2014.
[12] W. Zhu, W. Ding, J. Xu, Y. Shi, and B. Yin, "Hash Based Block Matching for Screen Content Coding," *IEEE Trans. Multimedia*, vol. 17, no. 7, pp. 935-944, July 2015.
[13] B. Li, J. Xu, and F. Wu, "A unified framework of hash-based matching for screen content coding," in *Proc. of IEEE Visual Communications and Image Processing Conference*, pp.530-533, December 2014.
[14] R. Joshi, J. Xu, R. Cohen, S. Liu, Z. Ma, and Y. Ye, "Screen content coding test model 3 (SCM 3)," *JCT-VC*, JCTVC-S1014, Strasbourg, France, October 2014.
[15] H. Yu, R. Cohen, K. Rapaka, J. Xu, "Common conditions for screen content coding tests," *JCT-VC*, JCTVC-S1015, Strasbourg, France, October 2014.
[16] G. Bjontegaard, "Calculation of average PSNR differences between RD curves," *Video Coding Experts Group (VCEG)*, VCEG-M33, Austin, Texas, U.S.A., April, 2001.