Low Power Motion Estimation based on Non-Uniform Pixel Truncation

Yaocheng Rong, Quanhe Yu, Da An, and Yun He, Senior Member, IEEE
Tsinghua National Laboratory for Information Science and Technology
Department of Electronic Engineering, Tsinghua University
Beijing 100084, China
E-mail: {ryc08, yu-qh10, and02}@mails.tsinghua.edu.cn, hey@tsinghua.edu.cn

***Abstract*— Motion estimation consumes a large portion of computational resources in most video encoders, such as AVS, H.264/AVC and HEVC/H.265. Pixel truncation can effectively reduce the power consumption of motion estimation and it is usually performed uniformly within the search window, which unfortunately brings degradation to the compression efficiency. However, based on the observation of unequal distribution of motion vectors, we can apply different number of truncated bits in different search area to achieve a better tradeoff between compression efficiency and power consumption. The proposed algorithm saves approximately 64% of the hardware computational complexity than conventional full-bit pixel search and at the same time achieves almost the same compression efficiency, with only 0.04dB loss.** [1]

I. INTRODUCTION

Battery capacity is a critical concern for a mobile device especially in video processing applications of long duration. Users require large resolution pictures, long-time video capturing, good visual fidelity and a high compression ratio. Those requirements increase is much faster than battery capacity evolution. That is the reason why a low power video encoder design becomes more and more important.

Since H.261, the hybrid coding structure takes the dominating role in all those standards, MPEG-2/H.262, H.264/AVC, AVS and HEVC/H.265. Motion estimation plays an integral role in the hybrid encoder. It is usually conducted by checking all the search candidates block to find the best matched block in reconstructed pictures. A search window includes all the search candidates and search range is introduced to denote the horizontal and vertical range of the search window, in this paper horizontal and vertical search range remains the same. The following formula describes the block matching optimization process.

$$\{MV_x, MV_y\} = arg\left[min_{i,j}\, Cost(i,j)\right], s.t.\, i \in [-H, H], j \in [-V, V] \quad (1)$$

Where, **MV**= $\{MV_x, MV_y\}$ is the motion vector, under the minimum *Cost (i, j)*, i, j represents a pixel position, *Cost (i, j)* is the matching error function, also called the matching criterion. The sum of absolute differences (SAD) is a typical and efficient metric of *Cost (i, j)*.

$$\mathrm{SAD} = \sum\sum_{i,j=1}^{i,j=N} |C_{i,j} - R_{i,j}| \quad (2)$$

Where $C_{i,j}$ and $R_{i,j}$ are pixel values in the current block (current pixels) and the reference block (reference pixels). Besides SAD, bits used to encode the motion vector (MV) are also taken into account in *Cost (i, j)*. The matching criterion used in this paper is $Cost = \mathrm{SAD} + \lambda \cdot Bits_{MV}$ which is used in AVS, H.264/AVC and HEVC/H.265.

Motion estimation (ME) explores inter picture correlations, and provides a best prediction to the encoding block. However, in the meantime, its computational complexity takes 50%-90% of a typical video coding system [1]. Moreover, as reported in [2], 77% of the whole encoder power is consumed by the ME module.

A large amount of work has been done to alleviate the complexity and the computational load of motion estimator. On one hand, fast motion estimation (Fast ME) algorithms try to reduce the number of search candidates, e.g., Cross-search[3], three step search[4], four step search[5], UMHexagonS[6] etc.; on the other hand, some works try to reduce the computational load and the complexity of the block matching error function itself, e.g., down sampling of the matching pixels[7]and truncating of the pixel value[2, 8-10]. Among them, pixel truncation is a simple and effective way to reduce the hardware power consumption of motion estimation.

Pixel truncation masks out the least significant bits (LSBs) of a pixel value while calculating SAD (2). In this way pixel truncation saves the valid bit width for a pixel and reduces the power consumption by bringing down the switching activity of the hardware.

Reference [8] presented a fixed pixel truncation method, where all 8 bit represented pixels were uniformly truncated to 4 bits to save the power consumption at the sacrifice of compression efficiency. Reference [9] proposed an adaptive pixel truncation scheme that adjusted the number of truncated bits (NTB) according to quantization parameter (QP), Reference [2, 10] introduced a two-step search comprised of first ME on uniformly truncated pixels, and then on full-bit (non-truncated) pixels to refine the MVs. However, the methods above brought non-negligible degradation to the compression efficiency and they all perform uniform pixel truncation within the search window.

[1] This work was partially supported by 973-2009CB320903 and 2010ZX03004-003

TABLE I
RATIO OF BEST MV INSIDE 1/2 SEARCH RANGE
(SR, [-P, P]) AROUND THE CENTER

Seq.	Resolution	SR=32	SR=16	SR=8	SR=4
BQSquare	416x240	99.28%	96.33%	98.93%	92.80%
Basketball Drill	832x480	96.65%	94.33%	93.63%	86.48%
ChinaSpeed	1024x768	93.95%	90.48%	84.68%	79.95%
Parkscene	1920x1080	98.98%	97.90%	95.25%	90.70%

We note that choosing an identical number of truncated bits (NTB) within a search window can be further improved. The motivation and the detailed reasoning will be given in subsequent sections. A primary conclusion is that different numbers of truncated bits applied in different search areas of a search window can offer a better tradeoff between the compression efficiency and the hardware computational complexity. The remainder of the paper is organized as follows. Section II discusses the different characteristics of pixel truncation in distinct areas. Section III introduces the proposed non-uniform pixel truncation algorithm. Section IV presents the performance. Section V concludes the paper.

II. Analysis of Pixel Truncation in Different Areas

To set an identical NTB in the search window is a common way of implementing pixel truncation in ME. However, we find that uniform pixel truncation is not an optimal choice.

For most video encoders, optimal motion vectors (best MV) embrace an unequal distribution where most of them locate near the search center. Table I lists the ratio of best MVs locating inside the 1/2 search range around the center, which indicates that the best MVs have much higher chance to locate within 1/2 search range than outside; e.g., 99.28% of the best MVs' value are less than 16-pixel shifting from the search center in the case of search range of [-32, 32] for the "BQSquare" sequence. Similar characteristics can be seen in other test sequences under various search range.

The unequal distribution implies that we can apply different pixel truncation for the internal area and the external area of a search window, and it might achieve a better result. In subsequent tests of section II, the internal area includes search candidates locating inside 1/2 search range around the center whereas the external area includes search candidates outside the 1/2 search range.

Pixel truncation could result in inaccurate pixel value and inaccurate absolute difference; consequently an inaccurate SAD is brought about. With an inaccurate SAD the best MV could possibly drift away from the original optimum. The best MV found from full-bit pixels and the best MV found from truncated pixels are denoted as $BestMV_{full}$ and $BestMV_{trun}$, respectively. The miss ratio is defined as

$$\mathrm{P(Miss)} = \mathrm{P}(BestMV_{full} \neq BestMV_{trun}) \quad (3)$$

The internal and external miss ratio are respectively defined as the miss ratio when the best MV locates in the internal area and in the external area.

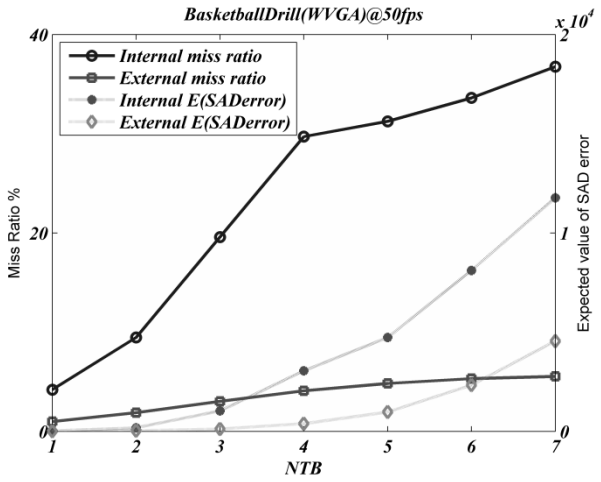

Fig. 1 Miss Ratio and expected value of SAD error Vs. NTB

In order to explore the distinct effect of pixel truncation internally and externally, the internal miss ratio and the external miss ratio are both examined in the test. An identical number of truncated bits (NTB) is performed internally and externally. In Fig. 1 for test sequence "BasketballDrill", the internal miss ratio is considerably larger than the external miss ratio no matter what value of NTB is set.

Besides the miss ratio, the consequence incurred by the failure to find the best MV needs to be considered as well. To explain the error incurred by the truncation, we define the following notations: SAD_{min1} is associated with $BestMV_{full}$. SAD_{min2} is the SAD under full-bit precision associated with $BestMV_{trun}$,. The inequality (4) holds true definitely.

$$SAD_{min2} \geq SAD_{min1} \quad (4)$$

If SAD_{min2} is much larger than SAD_{min1}, the truncation error degrades the compression efficiency a lot. We further define

$$SAD_{error} = SAD_{min2} - SAD_{min1} \quad (5)$$

$E(SAD_{error})$ is the expected value of SAD_{error}, which can indicate the degradation of pixel truncation on compression efficiency. Fig. 1 shows that $E(SAD_{error})$ in internal area is larger than that in the external area.

In summary, pixel truncation in the internal area degrades the compression efficiency more than that in the external area. Therefore, it is a better choice to perform a finer pixel truncation in the internal area than the external area. In other words, a smaller NTB is preferable in the internal area than the external area.

III. Proposed Algorithm for Pixel Truncation

Based on the above analysis, we propose an algorithm of non-uniform pixel truncation, which performs finer pixel truncation (small NTB) in the internal area and coarser pixel truncation (large NTB) in the external area. The algorithm is illustrated in Fig. 2 and described as follows.

a) The best MV search using finely truncated pixels is performed in the internal area, whereas the best MV search using coarse truncated pixels is applied in the external area. For example, as depicted in Fig. 2, when search candidates fall into the internal area (center area, filled in gray), we

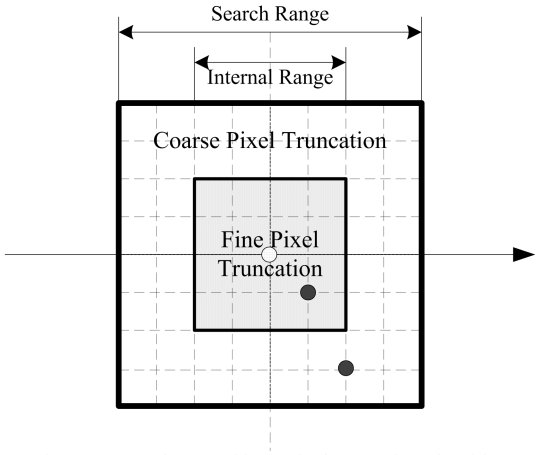

Fig. 2 Proposed non-uniform pixel truncation algorithm

denote these candidates as internal candidates, and full-bit pixels are taken. In the external area, denoted as external candidates, 8-bit full pixels are truncated to 4 bit ones. Then the best matched candidates are found separately in the internal area and the external area. As depicted in Fig. 2, (1,-1) and (2,-3) are the best MVs in the internal area and the external area, respectively.

b) Then the global best MV is decided between the two best MVs in each area ($MV_{internal}$, $MV_{external}$) (e.g., (1,-1), (2,-3)). SADs under full-bit precision associated with $MV_{internal}$ and $MV_{external}$ are both calculated and the best matched one is finally chosen to be the global best MV.

Appropriate NTB pairs ({NTB_{in}, NTB_{out}}) for the internal area and the external area should be examined for a better tradeoff between compression efficiency and hardware computational complexity. Fig. 3 demonstrates compression efficiency of several test sequences while an identical NTB within the search window is taken. Compression efficiency is compared in the form of rate-distortion performance, represented as BD-PSNR[11] in this paper. The compression efficiency drops higher when the NTB gets larger. When the NTB is smaller than two, the performance loss remains negligible. In order to minimize the power consumption, we set the NTB pairs as {NTB_{in}, NTB_{out}} = {2, 6}.

In the tests above, the internal search range is set as half of the search range, which covers a quarter of all the search candidates. However, the experiment results reveal that the compression efficiency degrades badly for test sequences with intensive motion when this fixed internal search range is applied, e.g. the compression efficiency loss for BasketballDrill (WVGA) reaches 0.107dB. Therefore, a dynamic internal search range is proposed to further improve the algorithm.

Since the predicted MV(PMV) is determined by the median value of the neighboring blocks' MVs (left: MVA, up: MVB, upright: MVC) in AVS and H.264/AVC, as illustrated in (6),

$$PMV_i = median(MVA_i, MVB_i, MVC_i), for\ i = x, y \quad (6)$$

without adding much computational load, the MVs of the neighboring blocks are further taken into account to determine the internal search range. Motion_factor which is defined in (7) and (8) is introduced to predict the motion range of the current block.

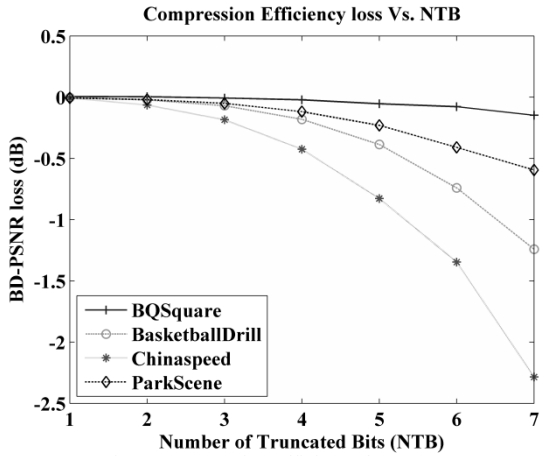

Fig. 3 Compression efficiency loss Vs. NTB

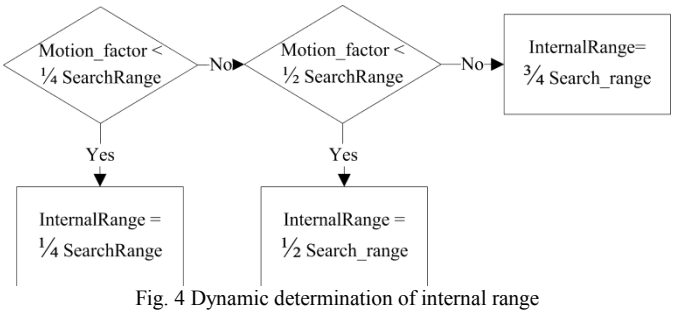

Fig. 4 Dynamic determination of internal range

$$motionfactor_i = max\{abs(MVA_i - PMV_i), abs(MVB_i - PMV_i), abs(MVC_i - PMV_i)\}, for\ i = x, y \quad (7)$$

$$motion\_factor = \max(motionfactor_x, motionfactor_y) \quad (8)$$

As illustrated in Fig. 4, the internal range is adaptively determined as a quarter, half or three quarters of the search range by the motion_factor.

The dynamic determination of the internal search range can offer better compression efficiency when compared with the fixed one. The degradation for sequence "Basketballdrill" is reduced smaller than 0.1dB (0.079dB). For simplicity, the proposed algorithm is denoted as NUPT.

IV. Performance Evaluation

The tests are performed on H.264/AVC reference software (JM18.3) with rate control option off and four QP values (22,27,32,37) and the results are presented in Table II The BD-PSNR loss shown in the table is the averaged value among different sequences (listed in Table III) of corresponding picture resolution.

The fixed pixel truncation approach, where all the pixels are truncated with the same NTB, is involved in the comparison. Fixed pixel truncation results in unacceptable performance loss, which is 0.143dB and 0.292dB for 4bit truncation and 5bit truncation, respectively. A two-step algorithm of [2] is implemented on the test platform (JM18.3) and the 0.079dB average PSNR drop remains acceptable. However, the proposed NUPT algorithm performs even better compared with [2], with only 0.039dB PSNR drop.

TABLE II
AVERAGE BD-PSNR (DB) LOSS COMPARED WITH FULL-BIT PIXEL SEARCH

Resolution	fixed 4 bit[8]	fixed 5 bit	Bahari[2]	NUPT
416x240	0.071	0.168	0.069	0.023
832x480	0.127	0.279	0.105	0.050
1280x720	0.168	0.276	0.041	0.039
1920x1080	0.205	0.445	0.103	0.043
Average	0.143	0.292	0.079	0.039

TABLE III
TEST SEQUENCES OF CORRESPONDING PICTURE RESOLUTION, SEARCH RANGE, AND NUMBER OF FRAMES

Resolution	Search Range	Sequences@Frame rate, Number of frames
416x240	[-8,8]	BasketballPass@50fps, 250 BQSquare@60fps,300 BlowingBubbles@50fps, 250 RaceHorses@30fps,300
832x480	[-16, 16]	BasketballDrill@50fps, 100 BQMall@60fps, 120 PartyScene@50fps, 100 RaceHorses@30fps,60
1280x720	[-32, 32]	Vidyo1@60fps, 120 Vidyo3@60fps, 120 Vidyo4@60fps,120
1920x1080	[-32, 32]	Kimono1@24fps, 48 ParkScene@24fps,48

The power consumption is not easy to measure until the hardware architecture design is completed. Pixel truncation can be used to disable the switching activity of the truncated bits, therefore total numbers of valid (untruncated) bits (TNVB) used in ME, as shown in (9), is a good estimate of the hardware computational complexity for power consumption.

$$TNVB = \sum(8 - NTB_i) \quad (9)$$

Normalized TNVB is further defined as TNVB divided by total numbers of full 8 bits (TNFB). For full-bit search, TNVB equals TNFB, accordingly normalized TNVB is 1.00.

In the experiments, the normalized TNVB for the test sequences varied from 0.31~0.45 and the average normalized TNVB of proposed NUPT algorithm and normalized TNVB of other algorithms are compared in Table IV. The proposed NUPT algorithm can save approximately 64% of the hardware computational complexity compared with conventional full-bit pixel (8-bit) search.

Along with Table II, results can be summarized as follows

1) Proposed NUPT algorithm gains better compression efficiency (>0.1dB) with a lower hardware computational complexity than fixed 4 bit[8] pixel truncation.

2) Moreover, proposed NUPT algorithm achieves a much better compression efficiency (>0.2dB) than fixed 5 bit with almost equal complexity.

3) Furthermore, proposed NUPT algorithm consumes 28% (0.36/0.50=72%) less complexity than Bahari[2] with still better compression efficiency.

In all, the proposed NUPT algorithm can save as much as 64% hardware computational complexity compared to conventional full-bit pixel (8 bit) search with almost the same compression efficiency (0.04dB loss).

TABLE IV
HARDWARE COMPUTATIONAL COMPLEXITY COMPARISON

	full-bit (8-bit)	Fixed 4 bit[8]	Fixed 5 bit	Bahari[2]	NUPT
Normalized TNVB	1.00	0.50	0.38	0.50	0.36

V. CONCLUSIONS

The paper proposes a hardware-oriented ME algorithm using non-uniform pixel truncation. Unequal distribution of motion vector reminds us of performing non-uniform pixel truncation within the search window. Dynamic internal range is introduced to improve the algorithm. Our proposed NUPT algorithm saves approximately 64% hardware computational complexity than conventional full-bit pixel search and performs almost the same in compression efficiency (0.04dB loss). It can be easily implemented in most motion estimation engines and is quite suitable for the low power ME applications in battery-powered mobile video devices.

REFERENCES

[1] Chen, C.Y., Chien, S.Y., Huang, Y.W., Chen, T.C., Wang, T.C., and Chen, L.G.: 'Analysis and architecture design of variable block-size motion estimation for H. 264/AVC', Circuits and Systems I: Regular Papers, IEEE Transactions on, 2006, 53, (3), pp. 578-593

[2] Bahari, A., Arslan, T., and Erdogan, A.T.: 'Low-power H. 264 video compression architectures for mobile communication', Circuits and Systems for Video Technology, IEEE Transactions on, 2009, 19, (9), pp. 1251-1261

[3] Ghanbari, M.: 'The cross-search algorithm for motion estimation [image coding]', Communications, IEEE Transactions on, 1990, 38, (7), pp. 950-953

[4] Li, R., Zeng, B., and Liou, M.L.: 'A new three-step search algorithm for block motion estimation', Circuits and Systems for Video Technology, IEEE Transactions on, 1994, 4, (4), pp. 438-442

[5] Po, L.M., and Ma, W.C.: 'A novel four-step search algorithm for fast block motion estimation', Circuits and Systems for Video Technology, IEEE Transactions on, 1996, 6, (3), pp. 313-317

[6] Chen, Z., Xu, J., He, Y., and Zheng, J.: 'Fast integer-pel and fractional-pel motion estimation for H. 264/AVC', Journal of Visual Communication and Image Representation, 2006, 17, (2), pp. 264-290

[7] Bierling, M., "Displacement estimation by hierarchical block matching," in Proc. SPIE Visual Communications and Image Processing vol. 1001, ed, 1988, pp. 942-951.

[8] He, Z., and Liou, M.L., "Reducing hardware complexity of motion estimation algorithms using truncated pixels," in Circuits and Systems, 1997. ISCAS'97., Proceedings of 1997 IEEE International Symposium on vol. 4, ed: IEEE, 1997, pp. 2809-2812.

[9] Zhong-Li, H., Chi-Ying, T., Kai-Keung, C., and Liou, M.L.: 'Low-power VLSI design for motion estimation using adaptive pixel truncation', Circuits and Systems for Video Technology, IEEE Transactions on, 2000, 10, (5), pp. 669-678

[10] Chatterjee, S.K., and Chakrabarti, I.: 'Power efficient motion estimation algorithm and architecture based on pixel truncation', Consumer Electronics, IEEE Transactions on, 2011, 57, (4), pp. 1782-1790

[11] Bjontegard, G.: 'Calculation of average PSNR differences between RD-curves', ITU-T VCEG-M33, 2001