Fast Coding Unit Partition Search

Satish Lokkoju#1, Dinesh Reddy#2

#Samsung India Software Operations Private Ltd
Bangalore, India.
#1.satish@samsung.com
#2.Dinesh.reddy@samsung.com

Abstract— Quad tree based encoders do brute force search for finding out the best partition for Coding Unit (CU). This brute force search performs encoding for all the possible block sizes and selects the partition size that gives best compression. This search along with inherent complexity of the latest encoders makes it extremely difficult to attain real time performance of 30 fps and low power. The solution to this problem is to perform a low complexity analysis of the Coding Unit and suggest the partition of the CU based on the available CU characteristics without performing entire encoding to estimate the cost. The present paper describes a method to do this using Sum of Absolute Difference, hereby SAD, and gradient information of the Coding Unit. We show that the presented method results in 3x faster encoding when compared to the brute force algorithm with small increase in bitrate (approximately 5% increase in worst case) and no change in subjective quality. The complexity bitrate trade off and the result BD-PSNR values of this method are also presented.

Keywords— Quad tree, HEVC, Coding Unit Partition, SAD, Mode decision.

I. INTRODUCTION

Video encoders have always been one of the resource consuming processes in the modern consumer electronics devices. The demand for more compression for sustaining a number of streaming solutions has lead to an increase in the complexity of the encoders. Though there are lot of advancements in processor technologies that directly result in more power it is not sufficient enough to get real time performance with latest encoders. Thus, there is a need for devising a method that results in reduction in complexity with less or no increase in bitrate. All modern encoders have a number of tools to attain the desired compression. One among them is the selection of mode based on the spatial and temporal characteristics, such as texture and motion vectors respectively, that give best compression. The selection is usually done using brute force search by encoding and calculating the cost of all different possible modes and selecting the best among them. Due to huge complexity of the brute force search this has been the target area of our algorithm. A generic multi depth quad tree based video codec gives us a lot of flexibility in terms of partitioning of the block. It also increases complexity. For example, for a Coding Unit of size 64x64 with minimum possible partition size 4x4, there are 18446744073709551616 ways in which the CU can be partitioned and we need to perform a minimum of 340 encoder-decoder cycles, on blocks of different sizes, to get the best possible partition information. This clearly is a time consuming exercise.

This paper presents a generic method to partition a Coding Unit to different blocks based on the spatial and temporal characteristics of the unit. The complexity bitrate trade-off that can be achieved is also explained. The rest of the paper is divided into the following topics. Section II gives a brief overview of the work that is done in this field and also the challenges that are faced when extrapolating those methods to the latest quad tree based video encoders. Section III will explain the proposed method in detail. Section IV will give the results of the tests and Section V will give some details about the future directions and conclusion.

II. CODING UNIT PARTITION

Fast mode detection is one of the important ways in which encoder complexity can be reduced without compromising in the bitrate. The Coding Unit is partitioned into blocks of different sizes to generate a layout that gives best compression. Discrete cosine transform (DCT) is one of the main tools that are present in almost all the modern encoders. The DCT exploits the spatial redundancy in a Coding Unit. Areas with uniform texture are best possible candidates for DCT. Thus our problem reduces to finding the uniform areas of a Coding Unit and partitioning them into blocks and performing the encoding on that layout. Texture analysis has been used to gauge the continuity of the Coding Unit. Filtering operation using Sobel operator is used in [1]. The gradient information thus generated is used to select the block size. The problem with this approach is that an area of the Coding Unit with small variations in the surrounding pixels is still marked as non-uniform. Logarithm of the ratio of energy of pixels to energy of perfectly non-uniform block [2] may also be used to decide on identifying uniform areas. This information is used against a threshold to calculate the block partition layout. These methods may not be used in quad tree based video codecs directly as the relative variations in the energies of the blocks of different sizes may be small and it becomes difficult to differentiate one block from another.

III. PARTITION SIZE DETERMINATION AND TWO PASS SEARCH

The proposed methods employs two different techniques, based on the type of the Coding Unit i.e. whether intra or inter.
For the intra Coding Units, prediction within the frame is possible. Thus the spatial characteristics of the Coding Unit will decide on the way it is partitioned. The gradient information of the Coding Unit is a nice representation of the uniformity/continuity of the Coding Unit. The gradient of the Coding Unit is obtained by sobel filtering on the Coding Unit. The sobel filter is a derive operator that provides gradient information in either of the directions based on the filter coefficients.

\[ \text{Sobel}_x = \begin{pmatrix}  1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \]

\[ \text{Sobel}_y = \begin{pmatrix}  1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \]

\[
G_x = p_{i-1,j+1} + 2 * p_{i,j+1} + p_{i+1,j+1} - p_{i-1,j-1} -2 * p_{i,j-1} - p_{i+1,j-1} \\
G_y = p_{i+1,j+1} + 2 * p_{i+1,j+1} + p_{i+1,j+1} - p_{i-1,j-1} -2 * p_{i-1,j} - p_{i-1,j+1} \\
\text{Gradient} = G_x + G_y
\]

Once the gradient is obtained for 64x64 Coding Unit, the uniformity of the block is decided based on the standard deviation of the constituent blocks. For example, to decide if a 16x16 area in a Coding Unit is uniform, the standard deviation of the constituent 8x8 blocks are calculated and if it is less than a threshold, the blocks are marked as 16x16. The same process is applied for all the block sizes starting for 4x4 to 64x64 iteratively. The threshold is different for different block sizes and it is calculated after extensive tests on a variety of video streams. This method overcomes the problem with small variations in the surrounding pixels by using standard deviation values of the constituent blocks to arrive at the decision. The said method takes care of the intra frames where only spatial prediction, and thus the gradient analysis, is used.

This method may not be used for inter frames as the blocks in the inter frame can use either spatial or temporal prediction. Thus, a combination of SAD and gradient analysis is appropriate. First consider the inter blocks in a inter frame. Best matching block from the previous reference frame is subtracted from the current frame and the residual in transformed and encoded. The best partition information cannot be obtained unless all the possible blocks are inter coded and cost determined. This includes motion vector prediction, motion estimation, transform and entropy coding which is cumbersome. Our method uses a combination of motion vector prediction, SAD estimation and gradient information to arrive at a best block partition. This method is described below.

The motion vectors of a Coding Unit are closely related to its surrounding units. This information can be used to provide an approximate location in the reference frame where the best match of the current Coding Unit can be found. Thus, the first step in estimating the inter mode block partition is to calculate the motion vector prediction of the surrounding Coding Units. The predicted motion vectors thus generated are added to the collocated Coding Unit in the reference frame. The prediction unit generated is subtracted from the current Coding Unit and the SAD is generated. For example, consider a block ‘r’ as shown in figure 1. The collocated SAD for this block is calculated by obtaining the motion vector prediction from the surrounding blocks and adding it to the coordinates of the collocated block in the previous frame, in this case ‘s’.

Fig. 1 shows an example motion vector prediction for calculating the inter mode SAD.

Standard deviations of the blocks are calculated using the SAD of the constituent blocks in a quad tree fashion. This gives information regarding the continuity that is the main criterion for deciding the block size. The standard deviation calculated is used against a threshold to decide the block size for inter mode. The threshold for inter mode blocks is calculated by extensive tests on different streams taking into account the quantization parameter used that is directly related to the quality of reference frame reconstructed and thus the SAD. As the Coding Units in the inter frame may also be coded with intra modes, a check is performed on the gradient of the Coding Unit using the same method that is used in the case of Intra frames. The initial partition layout is thus generated.

Once the block partition layout is determined, we need to perform inter mode and intra mode search on the constituent blocks of the layout to fine tune the partition layout. This method offers the flexibility of controlling the complexity with respect to bitrate. For example, consider the layout that is generated after the inter mode and intra mode partition search is performed as is shown in the figure below.

Fig. 2 shows an example Coding Unit partition layout with individual blocks color coded.
The brown coloured blocks of the 64x64 Coding Unit are the 32x32 blocks which are hereby named as x, y and z as shown in the figure 2. Similarly, green coloured blocks are 16x16 blocks of the 2nd 32x32 blocks and named as p, r and s as shown in the figure 2. Same is the case with 4 blue coloured 8x8 blocks of the 2nd 16x16 block which are named as a, b, c and d.

Once the Coding Unit layout is determined, the search is performed on the block size according to the layout. The first block is marked as 32x32 (x in figure 2) so the search is performed for the best intra mode if it is Intra frame and among intra or inter if it is Inter frame. The search is performed by doing one complete encoder – decoder cycle on the block and by finding the least cost. Now we move to the second block that has a combination of 8x8 (a, b, c and d) and 16x16 blocks (p, r and s). The first sub block of the second block has a depth of 3* that corresponds to 8x8. So the brute force search is performed on this block for all prediction types, modes and the best mode is selected. This same process is performed on all 8x8 blocks of the 16x16 block.

Finally, these blocks are combined to form a 16x16 block and the cost is calculated. This cost is compared to the sum of the costs of the best modes of the 8x8 blocks. This comparison is done up to two levels, i.e. the best depth given by the analyse module and a depth plus one i.e. block size just greater than the one that is obtained using the partition mode detection described above. This significantly decreases the time consumption because the number of brute force searches is reduced. Now we move on to the 2nd sub block of the second block. As this block depth is 2 corresponding to 16x16 (p in figure 2), the search is performed at this level and best is chosen. The same process takes places for 3rd (r in figure 2) and 4th (s in figure 3) blocks that are 16x16 blocks. Now all these 16x16 blocks are combined to form a 32x32 block and search is performed at that level. These costs are compared and best block size is chosen.

Finally search is performed at 64x64 level (as 3 blocks of the 64x64 Coding Unit are marked as 32x32 blocks) and mode with least cost is chosen. This is compared to the sum of the costs of best modes of the individual blocks. “Two pass” search significantly reduces the number of brute force searches performed, because initial information about the partition size is obtained by intra inter partition size search described above. For example the types of searches performed for the Coding Unit with layout as shown in the figure are,

- 32x32 for block x, y and z
  - 64x64 search for the complete Coding Unit and compare the cost with the sum of the least costs of the constituent sub blocks.
- 4 8x8 searches for sub blocks a, b, c and d
  - 16x16 search that includes the 4 8x8 blocks.
    - The least cost of 16x16 block is compared with sum of least costs of the constituent 8x8 blocks.
- 3 16x16 searches for p, q and r
  - 32x32 search that includes 3 16x16 sub-blocks p, q and r and 4 8x8 sub-blocks a, b, c and d. The least cost of 32x32 is compared with sum of least costs of 4 8x8 sub blocks and 3 16x16 sub blocks.

A total of 13 brute force searches are performed when compared to 340 searches that need to be performed for a full search.

The least cost obtained by intra inter partition search is compared with the best block size. The best possible scenario for this algorithm is when the partition search module gives the correct layout map i.e. in which blocks with modes 32x32 are marked with either depth 1 (32x32) or 2(16x16) by the partition search module. As in both these cases search is performed for 32x32 block. The worst case is when a 32x32 block is marked as 4x4 or 8x8 by the partition search module that results in less compression.

This performance degradation is more in the case when a 4x4 block is marked as 32x32 than a case where a 32x32 block is flagged as 8x8 or lower sizes.

From the analysis we performed on different streams, it is observed that a higher block size with uniform texture is most likely correctly marked in the partition search module that results in better compression.

The complexity vs. bitrate trade off can be achieved by manipulating the block sizes for which “two pass” search is enabled. By disabling the two pass search for 32x32 i.e. (for blocks that are marked as 32x32 no further search at 64x64 level is performed because the partition search module is sensitive enough to mark 32x32 block correctly in most of the cases). So, once a block is marked as 32x32 no further search is performed for 64x64 block in. This same technique can be applied to all the other block sizes according to the performance required. The results with different settings are discussed in the next section.

IV. RESULTS

The tests are performed on a set of test streams with different motion and texture variations. The results of the test streams with high motion namely BasketballDrill, BQMall, PartyScene, RaceHorses is presented here. Two different configuration settings, Random access and Low delay, are used to cover a broad range of use cases. These are similar to the settings “random access and low delay” configurations of the HEVC Test Model (HM) Reference Software. Random access configuration includes support for B frames and large motion search area and low delay corresponds to only I and P frames with low coding delay. The results are shown in Table 1 and Table 2. The Table 1 corresponds to configuration with hierarchical B coding enabled and other with hierarchical B coding disabled.

On an average random access configuration gives a 1.23 increase in bit rate with a 0.19 dB decrease in Y PSNR and 0.12 dB decrease in U and V PSNR values. The complexity is reduced by 3x on an average.

The quad tee partition of the Coding unit is partitioned recursively. So the 64x64 block of the coding unit is said to have a depth of 0. Similarly, the 4 32x32 blocks of the 64x64 have a depth of 1, 16x16 blocks have a depth of 2, 8x8 blocks have a depth of 3 and 4x4 blocks have a depth of 4.
Presented in Table 3 and 4.

Table 2: Reduced search (discussed algorithm) for Random access configuration (hierarchical B-frames).

One the other hand, the low delay configuration results in a 5.88 percent increase in bit rate with 0.28 dB decrease in PSNR value for Y and 0.12 and 0.13 dB decrease in PSNR values for U and V respectively. These results are presented in Table 3 and 4.

Table 3: Full brute force search for low delay configuration.

Table 4: Reduced search (discussed algorithm) for Low Delay configuration.
V. CONCLUSION

The test results clearly show that there is a significant reduction in complexity for a small trade-off in bitrate and PSNR values. There is a 3x decrease in complexity on an average with less than 5% increase in bitrate on an average. The complexity can be further reduced to \( \frac{1}{4} \) of the present by disabling the two pass search for 32x32 and 64x64 block sizes. This average decrease in PSNR values in this case is less than 1 dB. In the future this method can be extended to find the global best possible partition by taking into account the overall layout of the Coding Units in a frame or multiple frames thus reducing the bitrate.

REFERENCES

[1] Efficient Block-Size Selection Algorithm for Inter frame coding in H.264/MPEG-4 AVC, Andy C. Yu

[2] Advanced Block Size Selection Algorithm For Inter Frame Coding in H.264/MPEG-4 AVC, Andy C. Yu and Graham R. Martin


[4] Low Complexity H.264 Video Encoding, Paula Carrillo†, Hari Kalva†, and Tao Pint‡. †Dept. of Computer Science and Technology, Tsinghua University, Beijing, China ‡Dept. of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL, USA