VOL. 10, NO. 17, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

EXPLOITING LOOP PERFORATION FOR IMPROVING MOTION ESTIMATION PERFORMANCE

Ansu Mathew, Arun Iyer, Binu Joseph and Tushar Shah
AMD, Bangalore, India
E-Mail: ansu.Mathew@amd.com

ABSTRACT

Performance optimization has been an important engineering activity for decades. Many designs are proposed that trade off accuracy in return of increased performance. This paper talks about a technique, loop perforation, which helps in performance enhancement with a quality tradeoff. Loop perforation transforms loops to execute a subset of their iterations and thus reduces the amount of computational work that produces the result. Here loop perforation is performed on Motion Estimation (ME) block of video encoders. Motion Estimation block is the heart of video encoders, which occurs in Inter frame prediction, corresponds to the most computational intensive task of the whole video compression process.

In this paper, loop perforation is performed on 3 meta- functions of algorithm. The evaluation metrics used are Peak Signal to Noise Ratio (PSNR), execution time and sum of the pixels of frame (residue) as a proxy for the bitrate. The proposed approach could give 27-91% improvement in execution time without causing much degradation in quality, but with an acceptable rise in bit rate.

Keywords: PSNR, bitrate, execution time, loop perforation, residue.

INTRODUCTION

Video streaming applications have seen highest growth rate with the emerging popularity of websites and applications including video sharing, social networks, video conferencing etc. Millions of videos are being streamed everyday which causes large consumption of bandwidth as high quality video streams are widely available. These streams would also consume an abnormal amount of energy in client device and the battery will be drained at a faster rate creating a battery contingency. This can be a hindrance to the users involved in video streaming especially in low power mobile devices. Hence, it is required to have technique to control the complexity level in the computations involved in encoding and decoding of video streams.

Most of the streaming applications uses H.264 standard for video encoding/compression. Extremely computational expensive and resource hungry operation in the entire video compression process is motion estimation. More than 40% of encoding time is used by motion estimation (ME) block. Majority of the power consumption in the video encoder takes place inside the ME block. Real-time encoders may prefer to run faster and meet performance deadlines. But the computational complexity of motion estimation block can cause the video encoders to miss the real time deadlines. For devices that support transcoding, the speed of operation may be more important than saving memory. But the Motion Estimation block will be a hindrance to this. Motion estimation block slows down the video encoder and leads to more power consumption. It also causes the overheating of the battery.
Loop perforation can enable video encoder to complete its execution in a less time. Thus it improves the performance of the encoder and reduces the power consumption. In this, we address the process of implementing loop perforation in motion estimation block.

PRIOR WORK

Motion Estimation

To reduce the temporal redundancy between transmitted frames, a residual frame (difference between the predicted frame and current frame) is being constructed. The residual frame is encoded and sent to the decoder which re-creates the predicted frame, adds the decoded residual and reconstructs the current frame. The predicted frame is created from one or more past or future frames (reference frames).

Motion estimation helps in the formation of a predicted frame. During motion estimation the frame currently being encoded is divided into 16×16 regions of pixels called macroblocks. For each macroblock, the encoder tries to find out a similar 16 × 16 region in a previously encoded reference frame. Motion estimation therefore aims to find a 'best match' to the current block or region that minimizes the energy in the motion compensated residual. The choice of measure for 'energy' affects computational complexity and the accuracy of the motion estimation process. Most widely used energy measure is Sum of Absolute Difference (SAD) as in (1).

$$SAD = \sum_{j=0}^{N-1} \sum_{j=0}^{N-1} |Cij - Rij| \quad (1)$$

Motion vector is the offset between the current block and the reference block. Motion vectors are also encoded and transmitted along with residual frame. The goal of a practical motion estimation algorithm is to find a vector that minimizes the residual energy after motion compensation, whilst keeping the computational complexity within acceptable limits. Since this block is the computationally complex block, this block has highest interest in research activity. Lot of motion estimation

algorithms are existing to find out a best match within a short duration of time.

Loop Perforation

Loop perforation, which transform the program to skip loop iterations. The goal is to reduce the amount of computational work and therefore execution time and power are reduced. An example of reducing computational complexity is shown in Figure-1, in which a loop statement has to do m iterations. The number of computations can be reduced by changing the increment to larger values. This action is referred as perforation. Perforation can possibly lead the computation to produce an undesired result. But approximate computations can often tolerate such changes as long as they do not unacceptably reduce the quality of output [2]. For example, if a user is willing to tolerate a 1 dB degradation in peak signal-to-noise ratio (PSNR) for a video, loop perforation identify parts of the video encoder computation that it can skip while decreasing PSNR by no more than 1 dB . The result is a computation that performs less work (and therefore consumes fewer computational resources) while still producing acceptable output. In addition to enabling accuracy/performance tradeoffs, loop perforation can enhance robustness and enable energy savings. While it can reduce the number of computations, loop perforation enables the system to operate in a low frequency with same performance.

$$\text{Power} = CV^2\alpha f \quad (2)$$

From (2), we can say that, as frequency reduces power also gets reduced. At lower frequency, system can be operated at lower voltage. This also helps to reduce the power consumption.

Different options for Loop perforation are available. They are modulo perforation (which skips or executes every nth iteration), truncation perforation (which skips either an initial or final block of iterations), and random perforation (which skips randomly selected iterations at a mean given rate).

The perforation rate (r) determines the percentage of iterations that the perforated loop skips. If we are skipping every nth iteration as in modulo perforation, the perforation rate is r = (n-1)/n), if every nth iteration is executed if we are skipping every alternate iterations, then n=2, perforation rate=0.5.

Uses of Loop Perforation

- Improved Performance: the user can select a desired accuracy, then use loop perforation to maximize the delivered performance.
- Energy Savings: loop perforation reduces computations, thus reduces the energy required to produce the result.
- New Platforms or Contexts: Loop perforation can support effective redeployment onto a new platform (for example from desktop to mobile devices) or into a new context with different performance or accuracy requirements.
- Helps to meet real time constraints without deadline miss.

Figure-1. Method of reducing computational complexity.

PROPOSED APPROACH

Concept of Loop Perforation in Motion Estimation Block

Loop perforation cannot be applied to the applications where degradation in quality is critical [3]. In video encoding, there are different blocks that exploit the temporal redundancy and spatial redundancy. Loop perforation can be applied to these blocks. But it cannot be applied to the entropy encoder block, which converts the video sequence into compressed bit stream, as it may cause bit stream syntax violations. Here loop perforation is applied to the motion estimation block of video encoder. Motion Estimation block, which is more computationally complex has the flexibility for loop perforation to provide better performance.

By the application of loop perforation to the ME block, the block may fails to find out a best match from the reference frame and will cause a rise in the bit rate. But, as the computational complexity is reduced, the power can be saved. If a user can afford the small degradation in quality and rise in bit rate, loop perforation helps to improve the performance of the video encoder.

Flow of Motion Estimation Block and Method of Loop Perforation

Implementation of motion estimation algorithm and loop perforation over the code is performed in MATLAB environment. Pseudo code for the motion estimation block where loop perforation is performed is shown in Figure-2.

Motion compensated image is formed after motion estimation of each block in a frame. The difference of current frame and motion compensated image gives a residue. Residue is integer transformed [5] and quantized. In the decoder, inverse quantization and inverse integer transform is performed. The reconstruction of a frame is performed by the addition of residuals with the previous frame.

```
SAD_counter=0
for i= 1 to (frame_width−MBsize+1)
                step MBsize   do          //Loop perforation at macroblock level
  for j= 1 to (frame_height−MBsize+1)
                step MBsize   do
    for m= -search range to search range
                step 1   do                  // Loop perforation on full search locations
      for n= -search range to search range
                step 1 do
        if search range is outside the frame then
        continue;
      end if
            for x= 1 to MBsize
                step 1   do                  //Loop perforation on SAD calculation
              for y= 1 to MBsize
                step 1   do
                sum= sum+absolute (pixel_in_current_frame- pixel_in_reference_frame)
                SAD_counter++
                    end do
                end do
              if sum is less than cost then
                cost=sum
                save the motion vector for the corresponding.
                end if
      end do
    end do
  end do
end do
```

Figure-2. Pseudo code for motion estimation block with implementation of loop perforation.

Process of Loop Perforation in Motion Estimation Block

Loop Perforation was performed in 3 areas of Motion Estimation (ME) block.

- SAD Calculation
- Search Locations
- Macroblock Level

Loop Perforation on SAD Calculation

To find the best match between the current block and the block in the reference frame. Loops can be perforated in different perforation rates as in (3). Here the perforation rate is varied from 0.5 to 0.99609375. Perforation will affect in the calculation of absolute difference. We will not get a best match because of this perforation. Hence there will be a rise in the number of bits to be transmitted. Even though these variations will cause a quality degradation, the number of computations is reduced by a factor of 256 in the extreme case.

```
for (i=1 ; i<n ; i++) {                 for (i = 1; i<n; i=i+16) {
  for (j =1 ; j<n ; j=j+2) {..    to      for (j=1; j<n ; j=j+2) {..
        ……..}                                 ……..}
      ...}                                    ...}
```

(3)

Loop Perforation on Search Locations

Motion Estimation has to be performed in each location of search window. (±P samples about position (0, 0), the position of the current macroblock). $(2p+1)^2$ locations are there to be searched. Perforation is performed on these search locations as in (4). Perforation is applied to the loops with a perforation rate 0.5 to 0.99609375.

```
for (m=-P;m< P;P++) {          for (m=-P;m<P;m=m+16) {
  for (n=-P;n<P;n=n+2) {..  to   for (n= -P;n<P;n=n+2) {..
        ……..}                          ……..}
      ...}                             ...}
```

(4)

Loop Perforation at Macroblock Level

Instead of finding out the motion vectors for all the macroblocks, only motion vectors of the alternate macroblocks are found out. This is performed by doing the perforation at macroblock level as shown in (5). Here the perforation rate is 0.5.

```
for (i =1 ; i<row-mbsize+1 ; i=i+mbsize) {
      for (j =1 ; j<col-mbsize+1 ; j=j+mbsize) {..
                  ……..}
                  ...}        to
for (i = 1; i< row-mbsize+1 ; i=i+mbsize)  {
for (j =1 ; j<col-mbsize+1 ; j=j+2*mbsize){..
                  ……..}
            ...}
```

(5)

The motion vectors of skipped macroblocks are found out by averaging the motion vectors adjacent to it. Motion vector for the macroblock in the second row can be found out either by mean or by median of 3 motion vectors. i.e., motion vector on left, right and top of the macroblock. Motion vector of macroblock C is the mean or median of the motion vectors of A, B and D as in Figure-3.

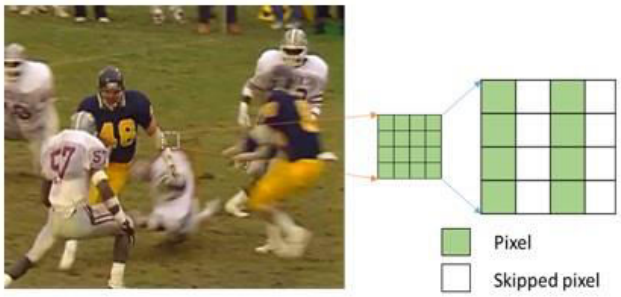

Figure-3. Loop perforation on SAD calculation (skipped pixels when perforation rate = 0.5).

	A		
B	C	D	

Figure-4. Representation of 16x16 macroblock with motion vectors A, B, C and D each for the 4x4 macroblocks.

EXPERIMENTAL SETUP AND RESULTS

Test sequences were selected based on the complexity in the contents. Three raw video sequences named Akiyo, Foreman and Football, of 30 frames with resolution 352x288, termed as CIF format, has been selected and categorized based on the amount of motion and texture content as in Table-1.

Metrics used for analysis are PSNR, Execution time, residue and No. of computation. PSNR has been considered as an objective quality measure.

$$PSNR\ (dB) = \frac{10 \log_{10} (2^n-1)^2}{MSE} \quad (6)$$

MSE is found between the original image and the reconstructed image, the square of the highest-possible signal value in the image.

Computation time for the whole MATLAB code (including motion estimation, motion compensation, integer transform, quantization and reconstruction of frame) was obtained from the profiling details of MATLAB.

Sum of absolute values in a frame after the quantization was taken as the residue. For a good search, residue will be less and for a bad search the residue will be high. Residue acts as a proxy for bitrate. Number of computations involved in SAD calculation for each frame was also calculated. Loop perforation was performed in 3 portions of the code. Each has 2 sets of loop.

Table-1. Classification of test sequences.

Index	Motion	Texture
Akiyo	Low	Low
Foreman	Medium	Medium
Football	High	High

Results are shown by the help of abbreviations shown in table 2 with x and y can be described as shown in (7).

```
for (i=1; i<n; i= i+x) {
    for (j =1; j<n; i=j+y) {...
        ..........}
```
(7)

Initially, Sum of absolute values of quantized image (residue) are found out for each frame and their average across a sequence is calculated. It is plotted against the execution time (Figure-5) and number of computations (Figure-6). By perforating the loops in SAD calculation of motion estimation, the search could not find out a best match.it leads to a rise in bit rate. Here 187% rise in residue (bit rate) for LP_16_16. The execution time is reduced by 66% for the same. For perforation LP_4_4, the bit rate increase is 32% and improvement in execution time is 62% and for perforation LP_2_1, the bit rate increase is 1% and execution time is reduced by 29%. These constitutes some of the optimal points.

Table-2. Perforation index.

Perforation index	LP_1_1	LP_1_2	LP_2_1	LP_2_2	LP_1_4	LP_4_1	LP_4_4	LP_1_8	LP_8_1	LP_8_8	LP_4_2	LP_2_4	LP_4_8	LP_8_4	LP_16_8	LP_8_16	LP_16_16	LP_8_2	LP_2_8	LP_4_16	LP_16_4	LP_16_2	LP_2_16	LP_16_1	LP_1_16
X	1	1	2	2	1	4	4	1	8	8	4	2	4	8	16	8	16	8	2	4	16	16	2	16	1
Y	1	2	1	2	4	1	4	8	1	8	2	4	8	4	8	16	16	2	8	16	4	2	16	1	16

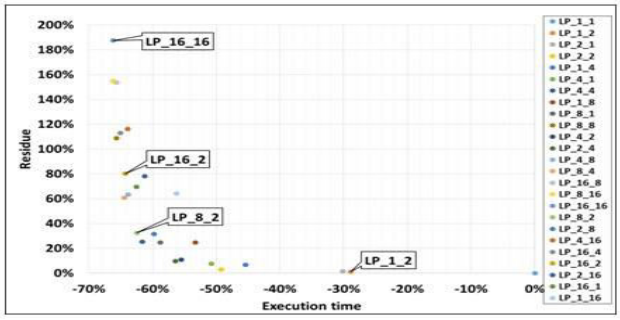

Figure-5. Loop Perforations in SAD Calculations (Residue vs Execution time).

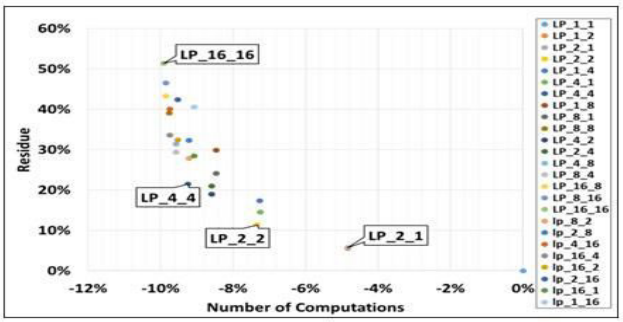

Figure-6. Loop perforation in SAD Calculations (Residue vs No. of calculations).

The results of Loop perforation in full search location is shown in Figure-7 and Figure-8. There is 51% rise in residue (bit rate) for LP_16_16. The execution time is reduced by 91% for the same. For perforation LP_4_4, the bit rate increase is 21% and improvement in execution time is 85%. For perforation LP_2_1, the bit rate increase is 6% and execution time is reduced by 45%. Similar variations can be observed in the residue versus number of computations plot.

Analysis of the quality degradation is an important factor. PSNR degradation for the loop perforation on full search location is very less. It is not varying in an ordered manner because getting a best match depends on the type of the sequence and the perforation rate. Figure-9 shows that the maximum PSNR degradation due to perforation in SAD calculations is around 0.5dB. This degradation is very negligible and is acceptable.

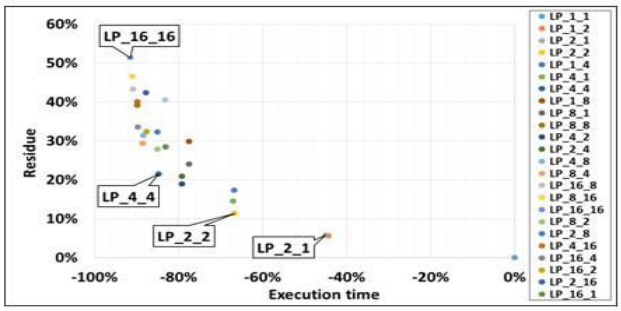

Figure-7. Loop Perforation in Full search locations (Residue vs Execution time).

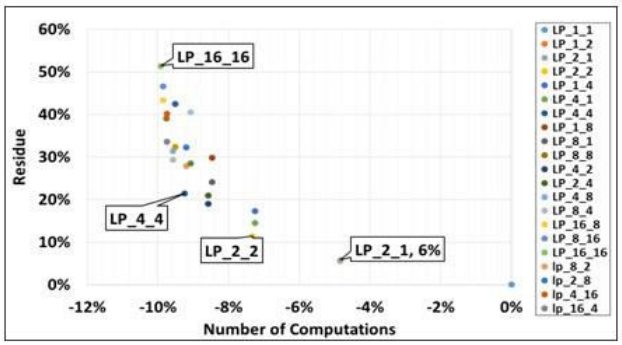

Figure-8. Loop perforation in full search locations (Residue vs No. of calculations).

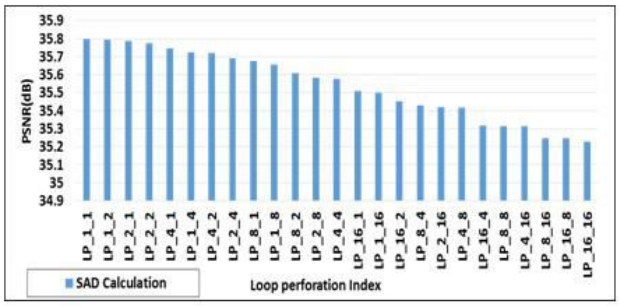

Figure-9. PSNR degradation: Loop perforation in SAD calculations.

While considering loop perforation at macro block level, the motion vector predicted by the help of median of the neighboring macro blocks are showing better bit rate. The rise in bit rate 37 % for the method that use average for the motion vector calculation, whereas or the median, it is only 27%.so we can go for the calculation of motion vector of the skipped macroblock by using median.

Table-3 shows the effect of loop perforation in encoding time and bitrate. When comparing the three methods, loop perforation on full search location is more preferable. This is because it provides more savings in computations with less degradation in bit rate. But it can be observed that we should exploit the possibility of loop perforation on all the three levels (pixel level search locations level and MB level) at the same time for better savings in computation time as well as bit rate

Acceptability of bit rate degradation is relative to the application and the threshold set by the user/architecture. In architectures where rate control is enabled, the architecture can automatically set the limit on the maximum bitrate allowed and trade-off the video quality. Other scenarios are when the system has a power constraint e.g. low battery life or if its running on power backup, then bitrate could be traded for either energy savings or data reliability (i.e. not losing data) respectively. The threshold to pick will require analysis with different sequences, resolutions and degree of energy/performance improvements.

Computation reduction results in energy savings and improved performance (which can be converted to energy savings as well, using Dynamic Voltage Frequency Scaling).

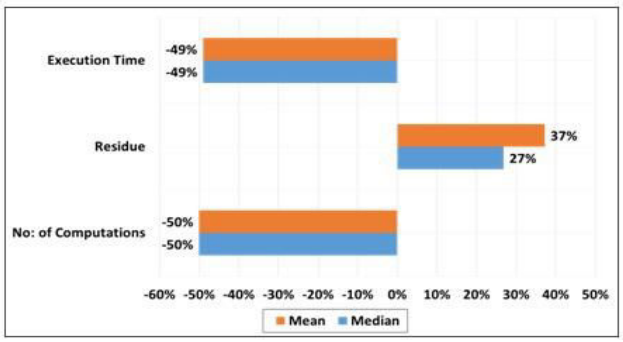

Figure-10. Loop perforation in macroblock level.

Table-3. Effect of perforation in encoding time and bitrate degradation.

Loop Perforations	Improvements in Encoding time		Degradation in Bitrate	
	Min	Max	Min	Max
SAD calculations	29%	66%	1%	187%
Full search locations	46%	92%	7%	69%
Macroblock level	47%	47%	27%	37%

CONCLUSIONS

Loop perforation will have its impact in QoS. The experimental results provides a superset of perforation rates which gives improvements in encoding time with a graceful degradation in PSNR.

Even though loop perforation affects the PSNR and bitrate, the amount of degradation in quality and rise in bitrate can always be calibrated for various configurations of perforation. Depending on the availability of disk space/bandwidth, the user can determine the perforation rate that can apply to the loops. Hence, a user with low disk space can go for a perforation rate of 0.5 and a user with enough disk space/low power can go for an aggressive perforation rate.

Loop perforation is a powerful tool over and above, any hardware level implementation (E.g. Voltage/Frequency scaling using parallel logic), as this is a high level approach and it can be made to work on any architecture at the RTL or even high level programming like algorithmic implementation using C, Java etc. This is a technique which falls under the broad Signal to Noise trade-off category, but the complexity is in applying it on a real design and in areas where the maximum benefit is seen (e.g. computations where precision can be traded for improvement in area or power or performance).

Perforating multiple loops at a time is another possible method which can be done in future. This method

might be able to provide better results without much increase in bitrate.

REFERENCES

[1] Malvar H.S., Hallapuro A., Karczewicz M. and Kerofsky L. 2003. "Low-complexity transform and quantization in H.264/AVC," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 598- 603, July.

[2] S. Misailovic, S. Sidiroglou, H. Hoffmann and M. Rinard. 2010. Quality of Service Profiling. In ICSE.

[3] H. Hoffmann, S. Misailovic, S. Sidiroglou, A. Agarwal, and M. Rinard. 2009. Using Code Perforation to Improve Performance, Reduce Energy Consumption, and Respond to Failures. Technical Report MIT-CSAIL-TR-2009-042, MIT, Sept.

[4] Sasa Misailovic, Henry Hoffmann, Martin Rinard, Stelios Sidiroglou. Managing Performance vs. Accuracy Trade-offs With Loop Perforation. In ESEC/FSE, 2011.

[5] Iain E. G. Richardson. "H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia".