World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 1, No. 4, 162- 166, 2011

Hexagonal Based Search Pattern for Motion Estimation in H.264/AVC

Sourabh Rungta
Computer Science Department
RCET
Bhilai, India

Dr.Neeta Tripathi
Electronics Deptt.
SSITM
Bhilai,India

Kshitij Verma
ICT Department
ABV-IIITM
Gwalior, India

Prof Anupam Shukla
ICT Deptt
ABV-IIITM
Gwalior,India

Abstract—This paper introduces the performance and technical features of the new generation international video coding standard H.264 as well as discusses the implementation and optimization of H.264 on mobile device. Optimization is proposed at algorithm/code-level for both encoder and decoder to make it feasible to perform real-time H.264/AVC video encoding/decoding on mobile device for mobile multimedia applications. For encoder an improved motion estimation algorithm based on hexagonal pattern search is proposed exploiting the temporal redundancy of a video sequence. For decoder, at code level memory access minimization scheme is proposed and at algorithm level a fast interpolation scheme is proposed. The experimental result shows that we have reduced the complexity of H.264 video encoder/decoder up to 60% as compared to the reference software JM10.0.

Keywords- Component H.264; Mobile Multimedia Applications; Video encoding.

I. Introduction

H.264/AVC [1] video coding standard provides up to 50% in bit-rates savings as compared to MPEG-4 advance simple profile for the same video quality, which is resulted from the fact that H.264/AVC has exploited some advanced video coding techniques, such as variable size block motion estimation and compensation, multiple reference frames prediction, enhanced entropy coding, intra prediction, in-loop filtering, and etc. However, the outstanding performance of H.264 comes along with the overhead of extremely high algorithmic complexity, which makes it too difficult to be realized for mobile multimedia applications with low-power consumption. Due to the high complexity of H.264 and the constraints of computation resource of the mobile device, it is a challenging work to achieve a real time codec on embedded processor. We used many techniques and proposed some efficient algorithms to reduce the computational load of the encoder and decoder, including algorithm optimization and code level optimization. For algorithm optimization, an improved directive estimation algorithm for encoder is proposed which can speed up encoding time drastically. For decoder, optimizations include memory access minimization at code level along with a fast interpolation scheme at algorithm level which together reduces the decoder complexity up to 50% by using these optimization methods.

H.264 is the emerging video coding standard with enhanced compression performance when compared to other existing coding standards. It outperforms the existing standards typically by a factor of two. Its excellent performance is achieved at the expense of the heavy computational load in the encoder. Especially, for the inter-frame prediction, H.264 allows blocks of variable size seven modes of different sizes in all, which are 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4,are supported along with the SKIP mode [2]. Hence the computational complexity of motion estimation increases considerably as compared with previous standards. This is one major bottleneck for the H.264 encoder.

H.264 supports various techniques among which intramode and inter-mode prediction mostly contribute to the coding efficiency. Lagrangian RDO method is used to select the best coding mode of intra and inter prediction with highest coding efficiency [3]. In Inter prediction tree-structured multi-block sizes i.e. seven modes with different block sizes is supported by this standard. H.264 tests the encoding process with all possible coding modes of inter-coding, and calculates their RD costs to choose the mode having the minimum cost. RDO technique involves a lot of computations. The reference implementation [4] of H.264 uses a brute force search for inter mode selection which is extremely computational constraining.

II. Overview of H.264

In 1997, the ITU-T Video Coding Experts Group initiated the work on the H.264 standard (formerly known as the H.26L standard). The main objective behind the H.264 project was to develop a high performance video coding standard in comparison with former video coding standards. H.264 has some features as below [5]:

- Low bit-rate, high quality.
- Wide application fields.
- Robust (error resilient) video transmission.
- Network friendliness.

In order to achieve the above features, H.264 uses many innovative techniques: First is the intra prediction. It exploits the spatial correlation to reduce redundancy. There are 13 intra modes. 4 for 16×16 block size, and 9 for 4×4 block size in the Baseline Profile.

H.264 also uses many motion compensation techniques. One is the variable block size motion compensation. It has 7 different block sizes, from 4×4 to 16×16. When coding, it searches these modes one by one and chooses the best one from them as the final mode for the current Macroblock (MB). For example, for high-detailed block, small size mode is preferred. For some background with no motion, the encoder will choose 16×16 block size to code it. In previous video coding standards, only one or two block sizes are used. It also adopts 1/4 resolution sub-pixel motion compensation for luma component. In order to eliminate blocking artifact, an effective deblocking filter is used. In addition, it also utilizes multiple reference frames and the number of reference frames can be from 1 to 16. Previous video standards only use one reference frame, so H.264 can utilize temporal correlation more efficiently to remove temporal redundancy.

H.264 uses the integer DCT (Discrete Cosine Transform) to speed up the transform speed. ICT (Integer Cosine Transform) uses left or right shift to replace multiplication and division, so it is very easy to be implemented. H.264 also adopts new entropy coding methods, CABAC (Context-based Adaptive Binary Arithmetic Coding) [6] and CAVLC (Context-Adaptive Variable Length Coding) [7]. By using these techniques, H.264 achieves a better performance than the previous video coding standards. These techniques improve the coding performance but increase the complexity of the codec at the same time. Some literatures show that complexity of H.264 is about 5-8 times that of H.263, so it is very difficult to implement real time H.264 codec on mobile device. How to optimize H.264 codec is a very important research issue for our implementation work.

H.264/AVC reference software JM (ver. 10.0) is used and code is built with Microsoft Visual Studio 2005.

III. Implementation and Optimization of H.264

The implementation and optimization of the H.264 coding algorithm on embedded system consists of the following steps: profile selection and the code optimization. Now, we explain them respectively.

A. Profile Selection

H.264 standard contains several profiles for different applications, including Baseline Profile, Main Profile, Extended Profile and High Profile. We chose the baseline profile for implementation, there are two reasons. Firstly, the objective of Baseline Profile is to serve low bit-rate video communications and this is consistent with our application. Secondly, other profiles contain many very complex optional techniques which are not suitable to be implemented on embedded device, and the techniques adopted by the Baseline Profile are good enough to satisfy our requirement for implementation of high performance multimedia communication system.

B. Code Optimization

Since the code just ported on PDA has much redundancy and execution efficiency is too low to satisfy our requirement for real time application, it is very necessary to optimize the code to increase the encoding speed and efficiency. The optimization of embedded program has the following levels: First is the program level optimization. It is a kind of global optimization method for the program. The main methods include: One is that using the optimization option provided by the compiler to optimize the code. Another way is to modify the program structure and reduce the logical branches because they destroy the program pipeline greatly and influence the execution efficiency of the code. The second level is the algorithm level optimization. Based on the features of H.264 itself, we proposed some fast and efficient algorithms to improve the coding speed to achieve our goals we have introduced above the steps and basic methods of the implementation and optimization of H.264 for mobile device. Next, we would like to discuss the algorithm level and code level optimization for both encoder and decoder in more details.

IV. Hexagonal Pattern Based Motion Search Algorithm

Improved motion estimation algorithm for encoder

Motion estimation in H.264 follows hexagon-based search pattern which is depicted in Fig. 1. It consists of seven checking points (shaded dots) with the center surrounded by six endpoints of the hexagon with the two edge points (up and down) being excluded. Of the six endpoints in the hexagon, two horizontal points are away from the center with distance 2 and the remaining four points have a distance of $\sqrt{5}$ from the center point, respectively. The distance between any neighboring pair of endpoints among the six endpoints is either 2 or $\sqrt{5}$. From the figure, we can see the six endpoints are approximately uniformly distributed around the center. Note that the hexagonal search pattern also contains seven checking points at the beginning. Then the search process, the hexagon-based search pattern keeps advancing with the center moving to any of the six endpoints. Whichever endpoint the center of the search pattern moves to, there are always three new endpoints emerging, and the other three endpoints are being overlapped.

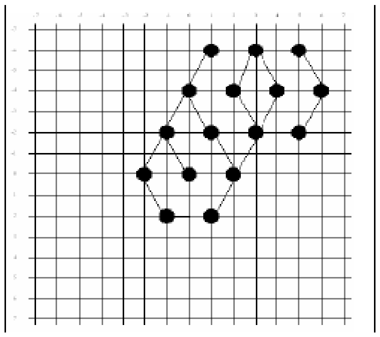

Figure 1. Hexagon-based search pattern

In order to speed up the motion estimation process, we divide the motion estimation process into two parts.

1) Dynamic motion detection

A motion detection method is suggested to separate out moving and stationary blocks within a frame. A frame is segmented in to blocks (7 modes for H.264).Each selected block is then identified as moving or stationary. The stationary block is assumed with zero motion vectors. And for the motion vector estimation of moving block temporal redundancies of video sequence are exploited to minimize the search points using standard motion estimation techniques. It has been experimentally proved that about 85% of the total blocks within frame remain stationary and that can be considered with zero motion vector. The early detection of blocks with zero motion vector leads to significant redundant computation being skipped and thus speed up the coding of video sequence. A threshold value is proposed to decide motion activity of a block. The threshold value is derived from the concept of frame variance and no assumption is adopted, ensuring that video quality is not degraded. Motion detection is carried out prior to motion estimation to avoid the heavy computational overhead.

Following method is adopted for motion detection based on frame variance.

Variance of the frame difference is calculated as:

$$v_{FD} = \frac{1}{MxN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(f_{FD}(i,j) - \overline{f_{FD}}(i,j)\right)^2$$

Where f_{FD} denotes the intensity difference between the current and previous frame. MxN is the frame size and $\overline{f_{FD}}$ is the mean of the difference frame. $\overline{f_{FD}}$ is taken as a threshold value. For a block of $px\,q$ samples, the local variance is determined as:

$$v_{BD} = \frac{1}{pxq}\sum_{i=1}^{p}\sum_{j=1}^{q}\left(f_{FD}(i,j) - \overline{f_{BD}}(i,j)\right)^2$$

Where $\overline{f_{BD}}$ denotes the mean of the block difference of current and reference frame. Then comparisons are made between v_{BD} and threshold v_{FD} to classify the block under consideration as a moving block or stationary block. If $v_{BD} > v_{FD}$, the block is considered to be moving and if $v_{BD} < v_{FD}$, the block considered to be stationary with zero motion vector. The motion activity is not constant over the entire frame of an image sequence thus the block variance (threshold), is not taken as a constant value but it is locally calculated. If a block is classified as a stationary block then no motion estimation is needed for that block. Motion vectors for that block are identical to the motion vectors of the block at same position in the previous frame. So for a stationary block.

$$MV\left[B_{pxq}(i,j,n)\right] = MV\left[B_{pxq}(i,j,n-1)\right]$$

Where $B_{pxq}(i,j,n)$ is pxq block in the current frame at position (i,j) and $B_{pxq}(i,j,n-1)$ is pxq block in the previous frame at position(i,j).

2) Motion estimation for moving block

For a moving block motion estimation is carried out by exploiting the temporal correlation between the frames in a video sequence. When matching a block from the current frame with a block from the previous frame, the matching criterion is usually evaluated using all samples within the block. Block matching is based on the assumption that all samples in a block move according to the same motion vector.

Therefore, in principle, a good estimation of the motion vector can be obtained from the direction of motion of in the previous frame. If motion vector for candidate block can be identified with useful and sufficient information from the motion vector of the block in the previous frame, the performance of the motion estimation can be improved and the total number of search points used to establish the motion vector of the current block can be significantly reduced. The proposed algorithm narrows down the search area using the direction of the same coordinate block of the previous frame. The direction of the block is decided according to the angle that is calculated using the motion vector of the block. Thus, the angle of the motion vector can be defined by the following equation.

$$radian = \operatorname{asin}\left(\frac{|y|}{\sqrt{x^2+y^2}}\right)$$

where, a sin is the arcsine, x and y are the distributions of the motion vector, and MV_{Angle} is the angle of the motion vector of the same coordinate block in previous frame. Based in this obtained MV_{Angle} using above equation following search patterns are followed. Using a hexagon based motion search as basis, proposed algorithm first checks for the MV_{Angle} and based on that following search patterns are followed.

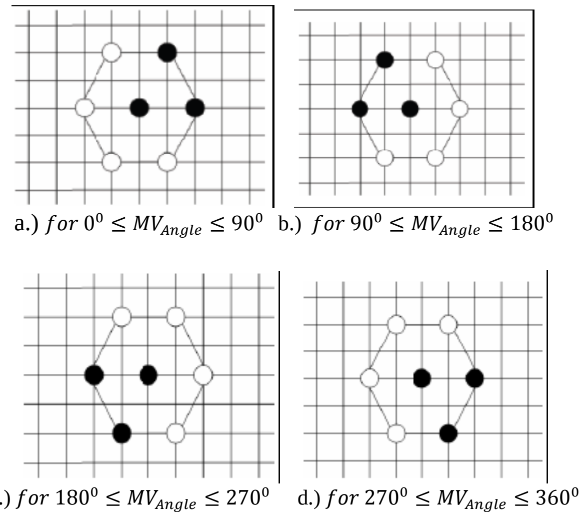
a.) $for\ 0^0 \leq MV_{Angle} \leq 90^0$
b.) $for\ 90^0 \leq MV_{Angle} \leq 180^0$
c.) $for\ 180^0 \leq MV_{Angle} \leq 270^0$
d.) $for\ 270^0 \leq MV_{Angle} \leq 360^0$

Figure 2. Search Pattern for Hexagonal Patterns

The **proposed algorithm** can be summarized as follows:
Step1. First determine the block as moving or stationary block using dynamic motion detection. If the block is stationary then its motion vectors are taken same as the motion vector of block from previous frame at same position, proceed to Step3 (ending), otherwise calculate MVAngle as described above.
Step 1 a). Then if the value of angle is between 0 to 90 the searching is done in the three points as shown above and find MDB from these three point and then proceed to Step 2.
Step 1(b). Then if the value of angle is between 90 to 180 the searching is done in the three points as shown above and find MDB from these three point and then proceed to Step 2 (Searching).
Step 1(c). Then if the value of angle is between 180 to 270 the searching is done in the three points as shown above and find MDB from these three point and then proceed to Step 2 (Searching).
Step2. With the MBD point in the previous search step as the center, a new large hexagon is formed. Two new candidate points are checked, and the MBD point is again identified. If the MBD point is still the center point of the newly formed hexagon, then go to Step 3 (Ending); otherwise, repeat this step continuously as described in Step1. (a)- (d).
Step3. Switch the search pattern from the large to the small size of the hexagon. Then MBD at two points of small hexagon is evaluated based on angle and compared with the current MBD point. The new MBD point is the final solution of the motion vector.

V. EXPERIMENTAL RESULTS

In this section, the experimental results are shown. In the experiments, four QCIF (176×144) test sequences were chosen, they are Coastguard, Container, Foreman, Salesman. H.264 reference software JM10 was applied as platform for evaluating the performance of the proposed algorithm. Then, the performance of the optimized encoder and decoder based on proposed algorithm is discussed. A range of different QP values have been used. The smallest QP used is 16 which correspond to the highest quality, but also the highest bitrate. The quantization parameter is increased with steps of four until maximum is reached at QP = 40.

The test conditions are shown in Table I. We used four Quantization Parameters while conducting the experiments on the test sequences, i.e. QP = 28, QP = 32, QP = 36 and QP = 40.

TABLE I. PARAMETERS OF THE REFERENCE SOFTWARE

Sequence type	IPPP
Codec	JM 10.0
QP	28, 32, 36, 40
ProfileIDC	66, 30
Hadmard Transform	Used
Entropy Coding method	CAVLC
Size	QCIF
Frequency	15
Search Range	16
Total No. of frames	10

The coding parameters used to evaluate the efficiency are ΔT change of average PSNR – $\Delta PSNR$ and change of average date bits $\Delta bitrate$. T_{ref} is the coding time used by JM10.0 encoder. Let $T_{proposed}$ be the time taken by the proposed algorithm.

The $\Delta T\%$ is defined as

$$\Delta T\% = \left[\frac{T_{proposed} - T_{ref}}{T_{ref}}\right] \times 100$$

The $\Delta Bitrate\%$ is defined as

$$\Delta Bitrate\% = \left[\frac{Bitrate_{proposed} - Bitrate_{ref}}{Bitrate_{ref}}\right] \times 100$$

For comparison purpose we take the salesman_qcif.yuv input stream as input for both the proposed and the reference software. QP is taken as 40 in the cases and the number of frames encoded is three. Above mentioned coding parameters are used for measuring the efficiency of proposed algorithm. The experimental results are shown in the following tables. From Table II, it is inevitable that the proposed algorithm reduces the encoding time for the four test sequences. Compared with the coding time of JM10.0 encoder, the coding time reduces by (43) %. The PSNR degradation is up to (0.06 db) which is invisible to human eye and the data bits are increased up to (0.93) %. Following are the values of coding parameters for QP values 28, 32, 36 and 40.

TABLE II. TEST RESULTS AT QP = 28

Video Sequence	ΔT%	ΔPSNR (Y)	ΔPSNR (U)	ΔPSNR (V)	ΔBitRate
coastguard_qcif.yuv	-10.81	-0.01	-0.02	0.06	0.02
container _qcif.yuv	-40.4	0.01	0.01	0	0.13
foreman _qcif.yuv	-8.29	0.05	-0.02	-0.04	0.18
salesman _qcif.yuv	-43.57	-0.01	0	0.01	-0.01

TABLE III. TEST RESULTS AT QP = 32

Video Sequence	ΔT%	ΔPSNR (Y)	ΔPSNR (U)	ΔPSNR (V)	ΔBitRate
coastguard_qcif.yuv	-10.37	-0.01	-0.01	0.02	0.34
container _qcif.yuv	-13.75	0	-0.01	0	-0.1
foreman _qcif.yuv	-11.83	0.01	0.04	-0.04	0.4
salesman _qcif.yuv	-10.12	-0.02	0.04	0.01	-0.43

TABLE IV. TEST RESULTS AT QP = 36

Video Sequence	ΔT%	ΔPSNR (Y)	ΔPSNR (U)	ΔPSNR (V)	ΔBitRate
coastguard_qcif.yuv	-42.46	0	-0.02	-0.01	-0.39
container _qcif.yuv	-41.93	0.01	0	0	0.5
foreman _qcif.yuv	-20.05	-0.04	-0.03	-0.02	-1.9
salesman _qcif.yuv	-7.32	-0.03	0.01	0.01	0.07

TABLE V. TEST RESULTS AT QP = 40

Video Sequence	ΔT%	ΔPSNR (Y)	ΔPSNR (U)	ΔPSNR (V)	ΔBitRate
coastguard_qcif.yuv	-42.46	0	-0.02	-0.01	-0.39
container _qcif.yuv	-8.85	0	-0.01	0	0.05
foreman _qcif.yuv	-6.55	0.13	0.11	0.05	4.01
salesman_qcif.yuv	-7.45	-0.01	-0.01	-0.01	-0.14

CONCLUSION

Since video coding is quite complex, the extensive statistical analysis made an excellent ground to develop the algorithm. Since this dissertation treats a lot of video coding aspects the work have given me great knowledge in this interesting field. In this paper optimizations are proposed for both encoder and decoder of H.264 video codec. For encoder an improved motion estimation algorithm is proposed which is based on the motion and direction of the macroblocks in the frames in a video sequence. The proposed algorithm reduces motion estimation time considerably with negligible bitrate degradation.

REFERENCES

[1] ITU T Recommendation H.264 | ISO/IEC 14496 10 version 4, January 2005.

[2] T.Wiegand, G. J. Sullivan, Senior Member, IEEE, G. Bjontegaard, and A. Luthra, Senior Member, IEEE. Overview of the H.264/AVC Video Coding Standard.

[3] J. Lee and B. Jeon. “Fast Mode Decision for H.264 with Variable Motion Block Sizes”. Springer - Verlag, LNCS 2869, pages 723-730, 2003.

[4] Joint Video Team (JVT) of ISO/IEC-MPEG&ITU-T VCEG,” JM Test Model CODEC,http://www.iphome.hhi.de/suehring/tml/

[5] Iain E.G. Richardson. H.264 and MPEG-4 Video Compression, Wiley2004.

[6] D. Marpe, H. Schwarz, and T. Wiegand, “Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard,” IEEE Trans. on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 620-636, 2003.

[7] J. Heo, S. H. Kim, and Y. S. Ho, “Improved CAVLC for H.264/AVC Lossless Intra-Coding,” IEEE Trans. on Circuits and Systems for Video Technology, Vol. 20, No. 2, pp. 213-222, 2010.

[8] C. Zhu, X. Lin, and L.P. Chau, “Hexagon-based search pattern for fast block motion estimation,” IEEE Trans. on Circuits and Systems for Video Technology, Vol. 12, No. 5, pp. 349–355, 2002.

[9] C. Zhu, X. Lin, and L.P. Chau, “Enhanced hexagonal search pattern for fast block motion estimation,” IEEE Trans. on Circuits and Systems for Video Technology, Vol. 14, No. 10, pp. 1210–1214, 2004.

[10] J. Lu and M. L. Liou, “A simple and efficient search algorithm for block matching motion estimation,” IEEE Trans. Circuits Syst. Video Technology, vol. 7, pp. 429–433, 1997.