4

H.264 Motion Estimation and Applications

Murali E. Krishnan, E. Gangadharan and Nirmal P. Kumar
Anand Institute of Higher Technology, Anna University, India

1. Introduction

A video signal represented as a sequence of frames of pixels contains vast amount of redundant information that can be eliminated with video compression technology enhancing the total transmission and hence storage becomes more efficient. To facilitate interoperability between compression at the video producing source and decompression at the consumption end, several generations of video coding standards have been defined and adapted by the ITU-G and VCEG etc... Demand for high quality video is growing exponentially and with the advent of the new standards like H.264/AVC it has placed a significant increase in programming and computational power of the processors. In H.264/AVC, the motion estimation part holds the key in capturing the vital motion vectors for the incoming video frames and hence takes very high processing at both encoder and the decoder. This chapter gives an overview of Motion estimation and the various search algorithms and also the scalability of parallelism in their operations to enhance the performance and improve the overall video quality. For low-end applications, software solutions are adequate. For high-end applications, dedicated hardware solutions are needed.

This chapter gives an overview of H.264/AVC video coding in general and its applications in four main sections. Section 1 deals with motion estimation and the types of algorithms one of the key modules of H.264 and the most time-consuming. Section 2 deals with the estimation criterion and their role in determining the complexiety of the estimation algorithms. Section 3 briefly discusses about the scalability of parallelism in H.264 and the final section deals with the applications of H.264 focussing on Aerial video surveillance and its advantages.

1.1 Motion estimation

Motion estimation techniques form the core of H.264/AVC (Iain Richardson, 2010) video compression and video processing applications. It extracts motion information from the video sequence where the motion is typically represented using a motion vector (x, y). The motion vector indicates the displacement of a pixel or a pixel block from the current location due to motion. This information is used in video compression to find best matching block in reference frame to calculate low energy residue to generate temporally interpolated frames. It is also used in applications such motion compensated de-interlacing, video stabilization, motion tracking etc. Varieties of motion estimation techniques are available. There are pel-recursive techniques, which derive motion vector (T.Wiegand et.al, 2003) for each pixel and

there is also the phase plane correlation technique, which generates motion vectors via correlation between current frame and reference frame. However, the most popular technique is Block Matching methodology which is the prime topic of discussion here.

1.1.1 Block matching algorithm

Block Matching Algorithm (BMA) (IEG Richardson 2003) is the most popular motion estimation algorithm. BMA calculates motion vector for an entire block of pixels instead of individual pixels. The same motion vector is applicable to all the pixels in the block. This reduces computational requirement and also results in a more accurate motion vector since the objects are typically a cluster of pixels. BMA algorithm is illustrated in figure 1.

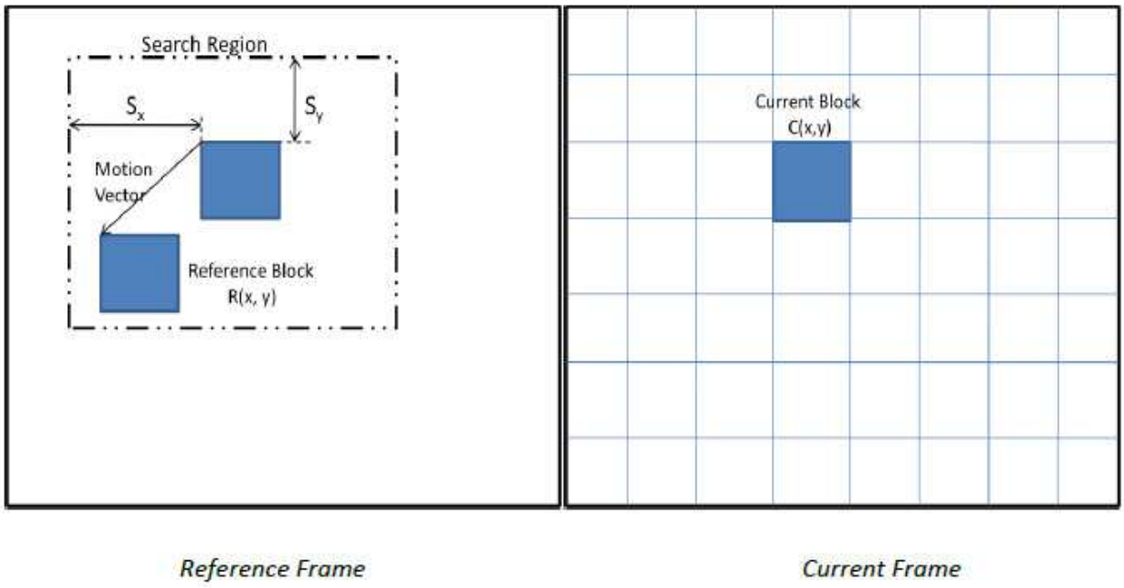


Fig. 1. Block Matching Algorithm

The current frame is divided into pixel blocks and motion estimation is performed independently for each pixel block. Motion estimation is done by identifying a pixel block from the reference frame that best matches the current block, whose motion is being estimated. The reference pixel block is generated by displacement from the current block's location in the reference frame. The displacement is provided by the Motion Vector (MV). MV consists of is a pair (x, y) of horizontal and vertical displacement values. There are various criteria available for calculating block matching.

The reference pixel blocks are generated only from a region known as the search area. Search region defines the boundary for the motion vectors and limits the number of blocks to evaluate. The height and width of the search region is dependent on the motion in video sequence. The available computing power also determines the search range. Bigger search region requires more computation due to increase in number of evaluated candidates. Typically the search region is kept wider (i.e. width is more than height) since many video sequences often exhibit panning motion. The search region can also be changed adaptively depending upon the detected motion. The horizontal and vertical search range, Sx & Sy, define the search area (+/-Sx and +/- Sy) as illustrated in figure 1.

1.1.2 Full search block matching

Full search block matching algorithm (Alois, 2009) evaluates every possible pixel block in the search region. Hence, it can generate the best block matching motion vector. This type of BMA can give least possible residue for video compression. But, the required computations are prohibitively high due to the large amount of candidates to evaluate in a defined search region. The number of candidates to evaluate are ((2*Sx) +1)*((2*Sy) +1) which is predominantly high compared to any of the search algorithms. There are several other fast block-matching algorithms, which reduce the number of evaluated candidates yet try to keep good block matching (Yu-Wen, 2006) accuracy. Note that since these algorithms test only limited candidates, they might result in selecting a candidate corresponding to local minima, unlike full search, which always results in global minima. Some of the algorithms are listed below.

1.1.3 Fast search algorithms

There are many other block matching algorithms (Nuno, 2002) and their variants available, but differs in the manner how they select the candidate for comparison and what is the motion vector resolution. Although, the full search algorithm is the best one in terms of the quality of the predicted image and its resolution of the motion vector it is very computationally intensive. With the realization that motion estimation is the most computationally intensive operation in the coding and transmitting of video streams, people started looking for more efficient algorithms. However, there is a trade-off between the efficiency of the algorithm and the quality of the prediction image. Keeping this trade-off in mind a lot of algorithms have been developed. These algorithms are called Sub-Optimal (Alois, 2009) because although they are computationally more efficient than the Full search, they do not give as good a quality as in the full search.

1.1.4 Three step search

In a three-step search (TSS) algorithm (Alan Bovik, 2009), the first iteration evaluates nine candidates as shown in figure 2. The candidates are centered on the current block's position. The step size for the first iteration is typically set to half the search range. These algorithms operate by calculating the energy measure (e.g. SAD) at a subset of locations within the search window as illustrated (TSS, sometimes described as N-Step Search) in Figure.2. SAD is calculated at position (0, 0) (the centre of the Figure) and at eight locations $\pm 2N-1$ (for a search window of $\pm$ ($2N$ −1) samples). The first nine search locations are numbered '1'. The search location that gives the smallest SAE is chosen as the new search centre and a further eight locations are searched, this time at half the previous distance from the search centre (numbered '2' in the figure). Once again, the 'best' location is chosen as the new search origin and the algorithm is repeated until the search distance cannot be subdivided further. This is the last iteration of the three-step search algorithm. The best matching candidate from this iteration is selected as the final candidate. The motion vector corresponding to this candidate is selected for the current block. The number of candidates evaluated during three-step search is very less compared to the full search algorithm. The TSS is considerably simpler than Full Search ($8N$ + 1 search compared with ($2N$+1 −1)2 searches for Full Search) but the TSS (and other fast search algorithms) do not usually perform as well as Full Search.

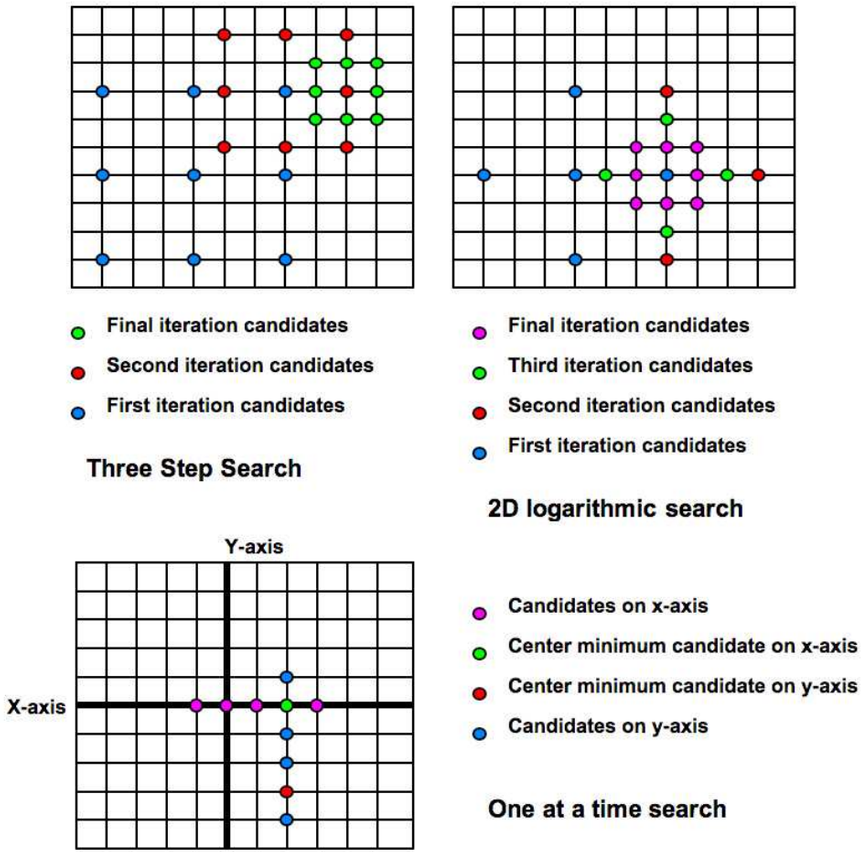


Fig. 2. Fast Search Algorithms

1.1.5 2D logarithmic search

2D Logarithmic (Alois, 2009) search is another algorithm, which tests limited candidates. It is similar to the three-step search. During the first iteration, a total of five candidates are tested. The candidates are centered on the current block location in a diamond shape. The step size for first iteration is set equal to half the search range. For the second iteration, the centre of the diamond is shifted to the best matching candidate. The step size is reduced by half only if the best candidate happens to be the centre of the diamond. If the best candidate is not the diamond centre, same step size is used even for second iteration. In this case, some of the diamond candidates are already evaluated during first iteration. Hence, there is no need for block matching calculation for these candidates during the second iteration. The results from the first iteration can be used for these candidates. The process continues till the step size becomes equal to one pixel. For this iteration all eight surrounding candidates are evaluated. The best matching candidate from this iteration is selected for the current block. The number of evaluated candidate is variable for the 2D logarithmic search. However, the worst case and best case candidates can be calculated.

1.1.6 One at a time search algorithm

The one at a time search algorithm estimates the x-component and the y-component of the motion vector independently. The candidate search is first performed along the x-axis.

During each ite ration, a set of three neighboring candidates along the x-axis are tested in Fig.2. The three-candidate set is shifted towards the best matching candidate, with the best matching candidate forming the centre of the set for the next iteration. The process stops if the best matching candidate happens to be the centre of the candidate set. The location of this candidate on the x-axis is used as the x-component of the motion vector. The search now continues parallel to the y-axis. A procedure similar to x-axis search is followed to estimate y-component of the motion vector. One-step at a time search on average tests less number of candidates. However, the motion vector accuracy is poor.

1.1.7 Sub-pixel motion estimation (Fractional Pel Motion Estimation)

Integer pixel motion estimation (also called as full search method) is carried out in the process of motion estimation that is mainly used to reduce the duplication (redundant data) among adjacent frames. But in practice, the distance of real motion is not always made by multiplier (which is constant) at the sampling interval.The actual motion in the video sequence can be much finer. Hence, the resulting object might not lie on the integer pixel (Iain Richardson, 2010) grid. To get a better match, the motion estimation needs to be performed on a sub-pixel grid. The sub-pixel grid can be either at half pixel resolution or quarter pixel resolution.

Therefore it is advantageous to use the subpixel motion estimation technique to ensure high compression with high PSNR ratio of reconstructed image. The motion vector can be calculated at 1/2, 1/4, 1/8 subpixel (Young et.al 2010) positions. The motion vector is to be calculated at 1/4 pixel gives more detailed information than at 1/2 pixel position. Since, the image has been enlarged, interpolation must be implemented to compensate for the pixel value in case of enlargement.

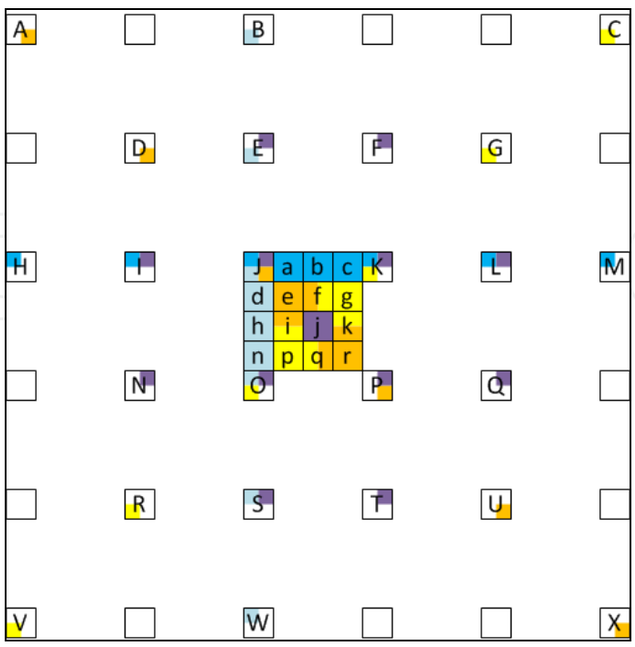


Fig. 3. 6-tap Directional interpolation filter for luma

Figure.3 shows the support pixels for each quarter-pixel position (Young et.al 2010) with different colours. For instance, the blue integer pixels are used to support the interpolation of three horizontal fractional pixels, a, b and c, the light-blue integer pixels for three vertical fractional pixels, d, h and n, the deep-yellow integer pixels for two down-right fractional pixels, e and r, the light-yellow integer pixels for two down-left fractional pixels, g and p, the purple integer pixels for the central fractional pixel j.

For each of the three horizontal fractional (Zhibo et.al.,2007) positions, a, b and c, and the three vertical fractional positions, d, h and n, which are aligned with full pixel positions, a single 6-tap filter is used and their equations are produced . The filter coefficients of DIF are {3, -15, 111, 37, -10, 2}/128 for ¼ position (and mirrored for ¾ position), {3, -17, 78, 78, -17, 3}/128 for ½ position.

$$\begin{aligned} a&=(3H-15I+111J+37K-10L+2M+64)>>7\\ b&=(3H-17I+78J+78K-17L+3M+64)>>7\\ c&=(2H-10I+37J+111K-15L+3M+64)>>7 \end{aligned} \quad (1)$$

$$\begin{aligned} d&=(3B-15E+111J+37O-10S+2W+64)>>7\\ h&=(3B-17E+78J+78O-17S+3W+64)>>7\\ n&=(2B-10E+37J+111O-15S+3W+64)>>7 \end{aligned} \quad (2)$$

For the 4 innermost quarter-pixel positions, e, g, p, and r, the 6-tap filters at +45 degree and -45 degree angles are used respectively.

$$\begin{aligned} e&=(3A-15D+111J+37P-10U+2X+64)>>7\\ g&=(3C-15G+111K+37O-10R+2V+64)>>7\\ p&=(2C-10G+37K+111O-15R+3V+64)>>7\\ r&=(2A-10D+37J+111P-15U+3X+64)>>7 \end{aligned} \quad (3)$$

For another 4 innermost quarter-pixel positions, f, i, k, and q, a combination of the 6-tap filters at +45 degree and -45 degree angles, which is equivalent to a 12-tap filter, is used.

$$\begin{aligned} f&=(e+g+1)>>1=((3A-15D+111J+37P-10U+2X)+(3C-15G+111K+37O-10R+2V)+128)>>8\\ i&=(e+p+1)>>1=((3A-15D+111J+37P-10U+2X)+(2C-10G+37K+111O-15R+3V)+128)>>8\\ k&=(g+r+1)>>1=((3C-15G+111K+37O-10R+2V)+(2A-10D+37J+111P-15U+3X)+128)>>8\\ q&=(p+r+1)>>1=((2C-10G+37K+111O-15R+3V)+(2A-10D+37J+111P-15U+3X)+128)>>8 \end{aligned} \quad (4)$$

$$j=((5E+5F)+(5I+22J+22K+5L)+(5N+22O+22P+5Q)+(5S+5T)+64)>>7 \quad (5)$$

The exception is the central position, j, where a 12-tap non-separable filter is used. The filter coefficients of DIF are {0, 5, 5, 0; 5, 22, 22, 5; 5, 22, 22, 5; 0, 5, 5, 0}/128 for the central position.

1.1.8 Hierarchical block matching

Hierarchical block matching algorithm (Alan Bovik, 2009) is a more sophisticated motion estimation technique which provides consistent motion vectors by successively refining the motion vector at different resolutions. In this, a pyramid fig.4 of reduced resolution video frame is formed from the source video. The original video frame forms the highest resolution image and the other images in the pyramid are formed by down sampling the original image. A simple bi-linear down sampling can be used. This is illustrated in figure 4. The block size of NxN at the highest resolution is reduced to (N/2) x (N/2) in the next

resolution level. Similarly, the search range is also reduced. The motion estimation process starts at the lowest resolution. Typically, full search motion estimation is performed for each block at the lowest resolution. Since the block size and the search range are reduced, it does not require large computations. The motion vectors from lowest resolution are scaled and passed on as candidate motion vectors for each block to next level. At the next level, the motion vectors are refined with a smaller search area. A simpler motion estimation algorithm and a small search range is enough at close to highest resolution since the motion vectors are already close to accurate motion vectors.

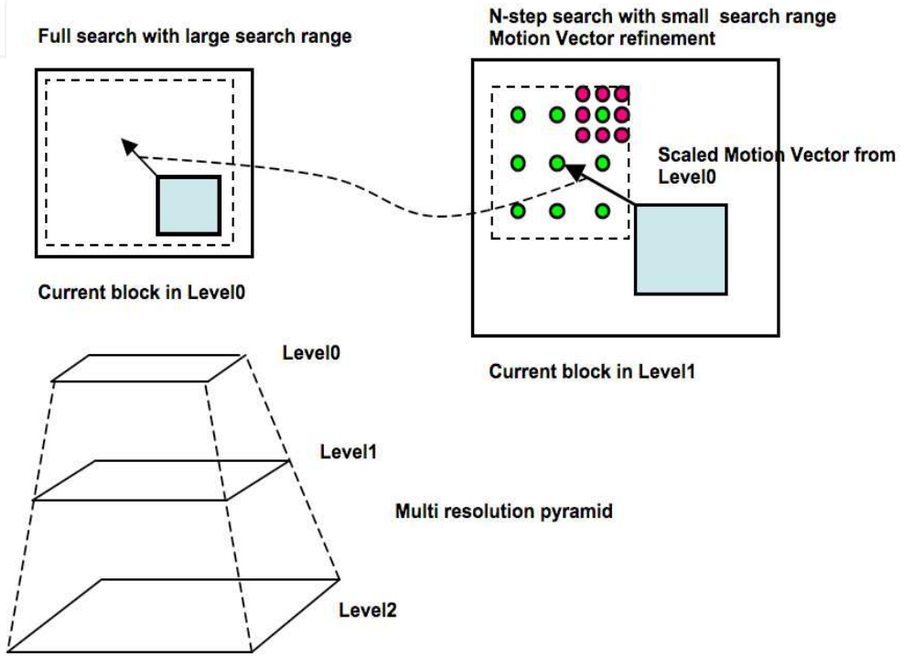


Fig. 4. Hierarchial block matching

1.1.9 Global motion estimation

There is another type of motion estimation technique known as global motion estimation. The motion estimation techniques discussed so far are useful in estimating local motion (i.e. motion of objects within the video frame). However, the video sequence can also contain global motion. For some applications, such as video stabilization, it is more useful to find global motion (Alan Bovik 2009) rather than local motion. In global motion, the same type of motion is applicable to each pixel in the video frame. Some examples of global motion are panning, tilting and zoom in/out. In all these motion, each pixel is moving using the same global motion model. The motion vectors for each pixel or pixel block can be described using following parametric model with four parameters Global motion vector for a pixel or pixel block is given (6) & (7). For pan and tilt global motion, only q0 and q1 are non-zero i.e. constant motion vector for the entire video frame. For pure zoom in/out, only p0 and p1 will be nonzero.

$$G_x = p0^* x + q0 \tag{6}$$

$$G_y = p1^* x + q1 \quad (7)$$

However a combination of all the parameters is usually present. Global motion estimation involves calculation of the four parameters in the model (p0, p1, q0, q1). The parameters can be calculated by treating them as four unknowns. Hence, ideally sample motion vectors at four different locations can be used to calculate the four run known parameters. In practice though, more processing is needed to get good estimate for the parameters. Also, note that still local motion estimation, at least at four locations, is essential to calculate the global motion estimation parameters. However, there are algorithms for global motion estimation, which do not rely on local motion estimation. The above parametric model with four parameters cannot fit rotational global motion. For rotational motion a six-parameter model is needed. However, the same four-parameter model concepts can be extended to the six-parameter model.

1.1.10 True motion estimation

For video compression applications it is enough to get a motion vector corresponding to best match. This in turns results in lower residual energy and better compression. However, for video processing applications, especially for scan rate conversion, true motion estimation is desired. In True Motion estimation, the motion vectors should represent true motion of the objects in the video sequence rather than providing best block match. Hence, it is important to achieve a consistent motion vector field rather that best possible match. True motion estimation can be achieved via both post-processing the motion vectors to get smooth motion vector field as well as building the consistency measures in motion estimation algorithm itself. Three Dimensional Recursive Search (3DRS) in Fig.5 is one such algorithm where the consistency assumption is inbuilt into the motion estimation.

The algorithm works on two important assumptions – objects are larger than block size and objects have inertia. The first assumption suggests that the neighboring block's motion vectors can be used as candidates for the current block. However, for neighboring blocks ahead in raster scan, there is no motion vectors calculated yet. Here, the second assumption is applied and motion vectors from previous frame are for these blocks. 3DRS motion estimator's candidate set consists of only spatial & temporal neighboring motion vectors. This results in a very consistent motion vector field giving true motion. To kick-start the algorithm a random motion vector is also used as a candidate as illustrated in figure 5.

1.2 Distortion metrics role of estimation criteria

The algorithms/techniques discussed above need to be incorporated into an estimation criterion that will subsequently be optimized in order to obtain the prediction error (Young et.al 2009) or the residual energy of the video frames. There is no unique criterion as such for motion estimation because its choice depends on the task/application at hand. For example, in compression an average performance (prediction error) of a motion estimator is important, whereas in motion-compensated interpolation (Philip 2009) the worst case performance (maximum interpolation error) may be of concern. Moreover, the selection of a criterion may be guided by the processor capabilities on which the motion estimation will be implemented. The difficulty in establishing a good criterion is primarily caused by the fact that motion in images is not directly observable and that particular dynamics of intensity in an image sequence may be induced by more than one motion.

Motion estimation therefore aims to find a 'match' to the current block or region that minimizes the energy in the motion compensated residual (the difference between the current block and the reference area). An area in the reference frame centered on the current macro block (Iain Richardson 2010) position (the search area) is searched and the 16 × 16 region within the search area that minimizes a matching criterion is chosen as the 'best match'. The choice of matching criterion is important since block matching might require the distortion measure for 'residual energy' affects computational complexity and the accuracy of the motion estimation process. Therefore, all attempts to establish suitable criteria for motion estimation require further implicit or explicit modeling of the image sequence of the video. If all matching criteria resulted in compressed video of the same quality then, of course, the least complex of these would always be used for block matching.

However matching criteria (IEG Richardson 2003) often differ on the choice of substitute for the target block, with consequent variation in the quality of the coded frame. The MSD, for example, requires many multiplications whereas the MAD primarily uses additions. While multiplication might not have too great an impact on a software (Romuald 2006) coder, a hardware coder using MSE could be significantly more expensive than a hardware implementation of the SAD/MAD function. Equations 8,9,10 describe three energy measures, MSD, MAD and SAD. The motion compensation block size is $N \times N$ samples; $\mathrm{Cur}_{i,j}$, $\mathrm{Ref}_{i,j}$ are current and reference area samples respectively.Fig.6 shows the image in macroblock form for the current video frame.(see photo)

Fig. 6. Macroblock view of the Frame

1.2.1 Mean squared difference

$$MSD = \frac{1}{N^2} \sum_{1=0}^{N-1} \sum_{j=0}^{N-1} \left(\mathrm{Cur}_{i,j} - \mathrm{Ref}_{i,j} \right)^2$$

MSD is also called as Mean Square Error (MSE). It is the indication of amount of difference between two macro blocks. Practically, the lower MSD value better is the match.

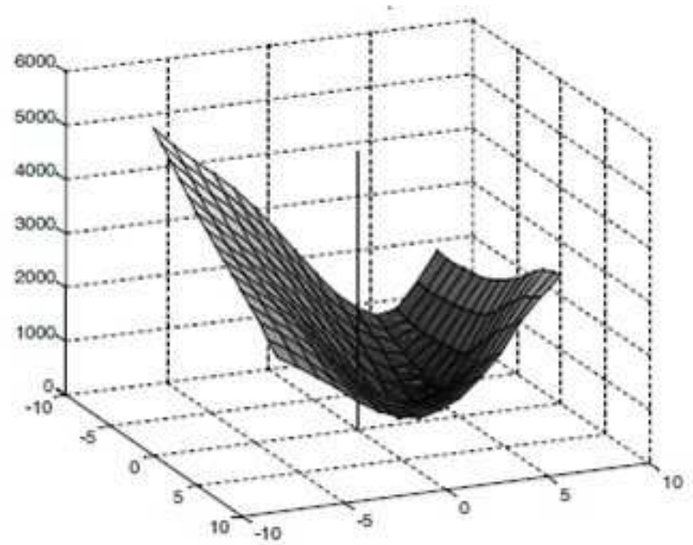


Fig. 7. MSD Map

1.2.2 Mean absolute difference

$$MAD = \frac{1}{N^2} \sum_{1=0}^{N-1} \sum_{j=0}^{N-1} \left| \mathrm{Cur}_{i,j} - \mathrm{Ref}_{i,j} \right|$$

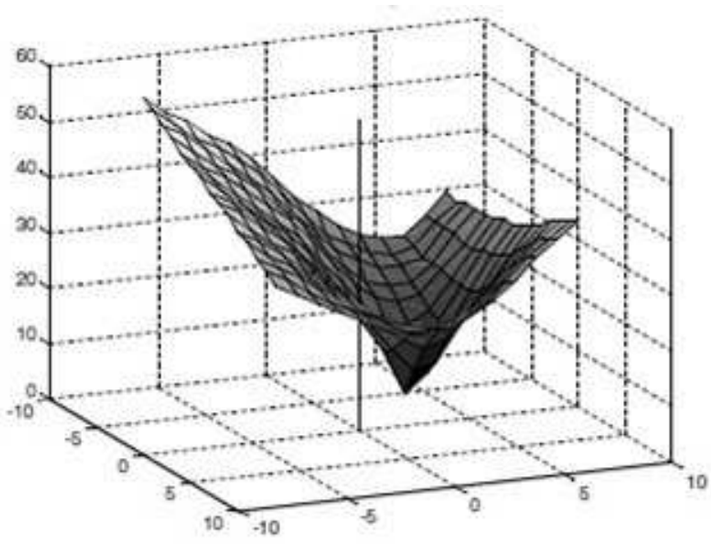


Fig. 8. MAD Map

The lower MAD the better the match and so candidate block with minimum MAD should be chosen. The function is also called as Min Absolute Error (MAE).

1.2.3 Sum of absolute difference

$$SAD = \sum_{1=0}^{N-1} \sum_{j=0}^{N-1} \left| \mathrm{Cur}_{i,j} - \mathrm{Ref}_{i,j} \right|$$

SAD is commonly used as the error estimate to identify the most similar block when trying to obtain the block motion vectoring the process of motion estimation, which requires only easy calculation as in fig.9 without the need for multiplication.

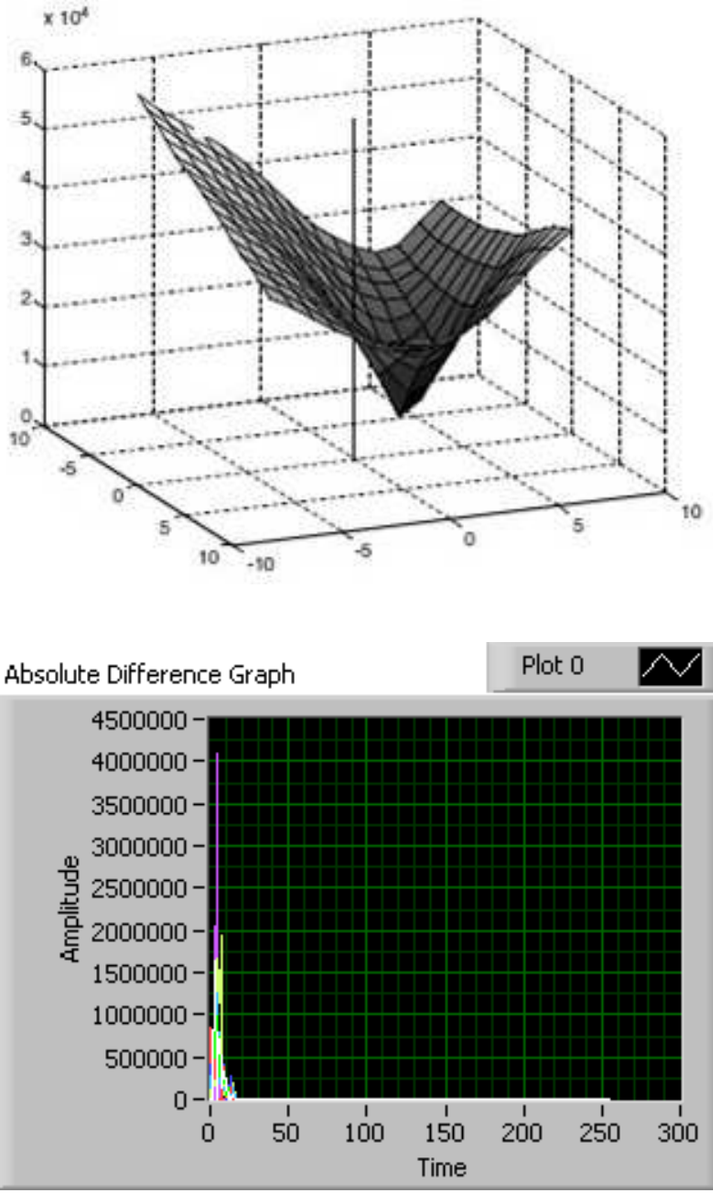


Fig. 9. SAD Map & its PSNR sketch

SAD (Young et.al 2009) is an extremely fast metric due to its simplicity; it is effectively the simplest possible metric that takes into account every pixel in a block. Therefore it is very effective for a wide motion search of many different blocks.

SAD is also easily parallelizable since it analyzes each pixel separately, making it easily implementable with hardware and software coders. Once candidate blocks are found, the final refinement of the motion estimation process is often done with other slower but more accurate metrics like which better take into account human perception. These include the sum of absolute transformed differences (SATD), the sum of squared differences (SSD), and rate-distortion optimization (RDO).

The usual coding techniques applied to moving objects within a video scene lower the compression efficiency as they only consider the pixels at the same position in the video frames. Motion estimation with SAD as the distortion metric used to capture such movements more accurately for better compression efficiency. For example in video surveillance using moving cameras, a popular way to handle translation problems on images, using template matching is to compare the intensities of the pixels, using the SAD measure. The motion estimation on a video sequence using SAD uses the current video frame and a previous frame as the target frame. The two frames are compared pixel by pixel, summing up the absolute values of the differences of each of the two corresponding pixels. The result is a positive number that is used as the score. SAD reacts very sensitively to even minor changes within a scene.

SAD is probably the most widely-used measure of residual energy for reasons of computational simplicity. The H.264 reference model software [5] uses SA (T) D, the sum of absolute differences of the *transformed* residual data, as its prediction energy measure (for both Intra and Inter prediction). Transforming the residual at each search location increases computation but improves the accuracy of the energy measure. A simple multiply-free transform is used and so the extra computational cost is not excessive. The results of the above example in Fig.6 indicate that the best choice of motion vector is (+2, 0). The minimum of the MSE or SAE map indicates the offset that produces a minimal residual energy and this is likely to produce the smallest energy of quantized transform.

1.2.4 Rate distortion optimization

These distortion metrics often play the pivotal role in deciding the quality of the videos viewed when choosing the method of Rate of Distortion Optimization (RDO) (Iain Richardson 2010)which is a technique for choosing the coding mode of a macroblock based on the rate and the distortion cost. Formulating this, we represent bitrate R and distortion cost D combined into a single cost J given by,

$$J = D + \lambda R \tag{11}$$

The bits are mathematically measured by multiplying the bit cost by the Lagrangian λ, a value representing the relationship between bit cost and quality for a particular quality level. The deviation from the source is usually measured in terms of distortion metrics in order to maximize the PSNR video quality metric.

The RDO mode selection algorithm attempts to find a mode that minimizes the joint cost J. The trade-off between Rate and Distortion is controlled by the Lagrange multiplier λ (Alan Bovik 2009). A smaller λ will give more emphasis to minimizing D, allowing a higher rate, whereas a larger λ will tend to minimize R at the expense of a higher distortion. Selecting the best λ for a particular sequence is a highly complex problem. Fortunately, empirical approximations have been developed that provide an effective choice of λ in a practical mode selection scenario.

Good results can be obtained by calculating λ as a function of QP.

$$\lambda = 0.852^{(QP-12)/3} \tag{12}$$

Distortion (D) is calculated as the Sum of Squared Distortion (SSD,

$$D_{SSD} = \sum_{1=0}^{N-1} \sum_{j=0}^{N-1} \left(\mathrm{Cur}_{i,j} - \mathrm{Ref}_{i,j} \right)^2$$

Where i,j are the sample positions in a block, Cur(i,j) are the original sample values and Ref(i,j)

are the decoded sample values at each sample position. Other distortion metrics, such as Sum of Absolute Differences (SAD), Mean of Absolute Difference (MAD) or Mean of Squared errors (MSE) may be used in processes such as selecting the best motion vector for a block [iv]. A different distortion metric typically requires a different λ calculation and indeed will have an impact in the computation process taken as a whole.

A typical mode selection algorithm might proceed as follows:

For every macroblock

- For every available coding mode m
- Code the macroblock using mode m and calculate R, the number of bits required to code the macroblock
- Reconstruct the macroblock and calculate D, the distortion between the original and decoded macroblock
- Calculate the mode cost J_m using (11), with appropriate choice of λ
- Choose the mode that gives the minimum J_m

This is clearly a computationally intensive process, since there is hundreds of possible modes combination and therefore it is necessary to code the macroblock hundreds of times to find the 'best' mode in a rate-distortion sense.

1.2.5 Conclusions and results

Thus a *matching criterion*, or *distortion function*, is used to quantify the similarity between the target block and candidate blocks. If, due to a large search area, many candidate blocks are considered, then the matching criteria will be evaluated many times. Thus the choice of the matching criteria has an impact on the success of the compression. If the matching criterion is slow, for example, then the block matching will be slow. If the matching criterion results

in bad matches then the quality of the compression will be adversely affected. Fortunately a number of matching criteria are suitable for use in video compression. Although, the number of matching criteria evaluated by block matching algorithms is largely independent of the sequence coded, the success of the algorithms is heavily dependent on the sequence coded.

1.3 Scalability of parallelism in H.264 video compression

The H.264/AVC standard provides several profiles to define the applied encoding techniques, targeting specific classes of applications. For each profile, several levels are also defined, specifying upper bounds for the bit stream or lower bounds for the decoder capabilities, processing rate, memory size for multipicture buffers, video rate, and motion vector range (Alois 2009) significantly improving the compression performance relative to all existing video coding standards [1]. To achieve the offered encoding performance, this standard incorporates a set of new and powerful techniques: 4×4 integer transform, inter-prediction with variable block-size, quarter-pixel motion estimation (ME), in-loop deblocking filter, improved entropy coding based on Context-Adaptive Variable-Length Coding (CAVLC) or on Content-Adaptive Binary Arithmetic Coding (CABAC), new modes for intra prediction, etc. Moreover, the adoption of bi-predictive frames (B-frames) along with the previous features provides a considerable bit-rate reduction with negligible quality losses.

For instance using Intel VTune software running on a Pentium IV 3 GHz CPU with H.264/AVC SD in main profile encoding solution with Arithmetic, controlling, and data transfer instructions are separated would require about 1,600 billions of operations per second. Table.1 illustrates a typical profile of the H.264/AVC encoder complexity based on the Pentium IV general purpose processor architecture. Notice that motion estimation, macroblock/block processing (including mode decision), and motion compensation modules which take up nearly the entire cycle (78%) of operations and account for higher resource usage.

	Arithmetic		Controlling		Data transfer		
Functions	MIPS	%	MIPS	%	MIPS	Mbytes/s	%
Integer-pel motion estimation	95,491.9	78.31	21,915.1	55.37	116,830.8	365,380.7	77.53
Fractional-pel motion estimation	21,396.6	17.55	14,093.2	35.61	30,084.9	85,045.7	18.04
Fractional-pel interpolation	558.0	0.46	586.6	1.48	729.7	1067.6	0.23
Lagrangian mode decision	674.6	0.55	431.4	1.09	880.7	2642.6	0.56
Intra prediction	538.0	0.44	288.2	0.73	585.8	2141.8	0.45
Variable length coding	35.4	0.03	36.8	0.09	44.2	154.9	0.03
Transform and quantization	3223.9	2.64	2178.6	5.50	4269.0	14,753.4	3.13
Deblocking	29.5	0.02	47.4	0.12	44.2	112.6	0.02
Total	121,948.1	100.00	39,577.3	100.00	153,469.3	471,299.3	100.0

(Baseline profile, 30 CIF frames/s, 5 reference frames, ±16-Pel search range, and QP = 20)

Table 1. Instruction profiling in Baseline Profile H.264

It can be observed that motion estimation, including integer-pel motion estimation, fractional-pel motion estimation, and fractional-pel interpolation in the table, takes up more than 95 percent of the computation in the whole encoder, which is a common characteristic in all video encoders. The total required computing power for a H.264 encoder is more than 300 giga instructions per second (GIPS), which cannot be achieved by existing processors. To account for this problem, several approaches have been adopted, such as the application of new low complexity ME algorithms that have been studied and developed (Yu Wen 2006), dedicated hardware (HW) structures and, more recently, multi-processor solutions.

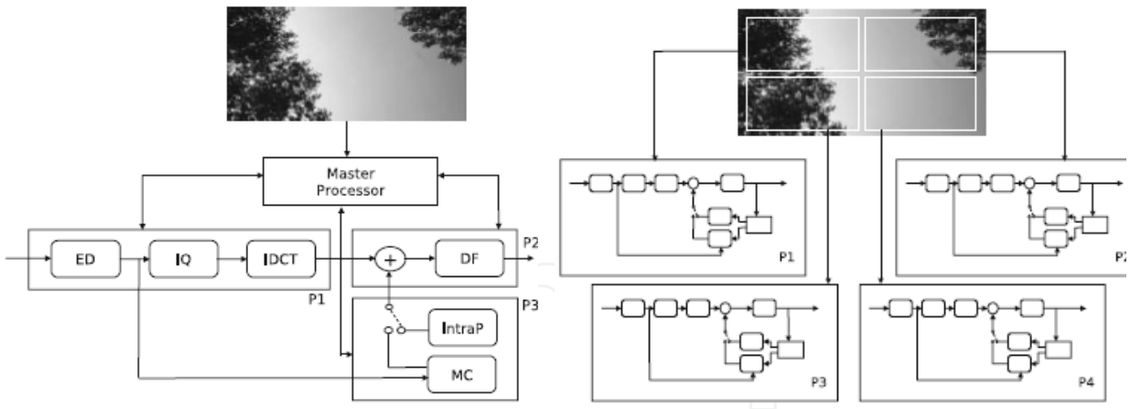
Nevertheless, the innumerous data dependencies imposed by this video standard frequently inflict a very difficult challenge in order to efficiently take advantage of the several possible parallelization strategies that may be applied. Up recently, most parallelization (Florian et.al 2010) efforts around the H.264 standard have been mainly focused on the decoder implementation [2]. When the most challenging and rewarding goal of parallelizing the encoder is concerned, it has been observed that a significant part of the efforts have been devised in the design of specialized and dedicated systems [7, 6]. Most of these approaches are based on parallel or pipeline topologies, using dedicated HWstructures to implement several parts of the encoder. When only pure software (SW) approaches are considered, fewer parallel solutions have been proposed. Most of them are based on the exploitation of the data independency between Group-of-Pictures (GOPs) of slices. For such a video encoder, it may be probably necessary to use some kind of parallel programming approach to share the encoding application execution time and also to balance the workload among the concurrent processors.

1.3.1 Parallelism in H.264

The primary aim of this section is to provide a deeper understanding of the scalability of parallelism in H.264. Several analyses and parallel optimizations have been presented about H.264/AVC encoders [3, 4, 8]. Due to the encoder's nature, many of these parallelization approaches exploit concurrent execution at: frame-level, slice-level, macroblock-level..The H.264 codec can be parallelized either by task-level and data-level decomposition. In Fig.10 the two approaches are sketched. In task-level decomposition individual tasks of the H.264 Codec are assigned to processors while in data-level decomposition different portions of data are assigned to processors running the same program.

1.3.2 Task-level decomposition

In task-level decomposition the functional partitions of the algorithm are assigned to different processors. As shown in Fig.10 the process of decoding H.264 consists of performing a series of operations on the coded input bitstream. Some of these tasks can be done in parallel. For example, Inverse Quantization (IQ) and the Inverse Transform (IDCT) can be done in parallel with the Motion Compensation (MC) stage. In Fig. 10a the tasks are mapped to a 4-processor system. A control processor is in charge of synchronization and parsing the bitstream. One processor is in charge of Entropy Decoding, IQ and IDCT, another one of the prediction stage (MC or IntraP), and a third one is responsible for the deblocking filter.



(a) (b)

Fig. 10. H.264 parallelization techniques. **a** Task-level decomposition. **b** Data-level decomposition.

Task-level decomposition requires significant communication between the different tasks in order to move the data from one processing stage to the other, and this may become the bottleneck. This overhead can be reduced using double buffering and blocking to maintain the piece of data that is currently being processed in cache or local memory. Additionally, synchronization is required for activating the different modules at the right time. This should be performed by a control processor and adds significant overhead.

The main drawbacks, however, of task-level decomposition are load balancing and scalability. Balancing the load is difficult because the time to execute each task is not known a priori and depends on the data being processed. In a task-level pipeline the executiontime for each stage is not constant and some stage can block the processing of the others. Scalability is also difficult to achieve. If the application requires higher performance, for example by going from standard to high definition resolution, it is necessary to reimplement the task partitioning which is a complex task and at some point it could not provide the required performance for high throughput demands. Finally from the software optimization perspective the task-level decomposition requires that each task/processor implements a specific software optimization strategy, i.e., the code for each processor is different and requires different optimizations.

1.3.3 Data-level decomposition

In data-level decomposition the work (data) is divided into smaller parts and each assigned to a different processor, as depicted in Fig.10b. Each processor runs the same program but on different (multiple) data elements (SPMD). In H.264 data decomposition can be applied at different levels of the data structure (see Fig.11), which goes down from Group of Pictures (GOP), to frames, slices, MBs, and finally to variable sized pixel blocks. Data-level parallelism can be exploited at each level of the data structure, each one having different constraints and requiring different parallelization methodologies.

1.3.4 GOP-level parallelism

The coarsest grained parallelism is at the GOP level. H.264 can be parallelized at the GOP-level by defining a GOP size of *N* frames and assigning each GOP to a processor. GOP-level

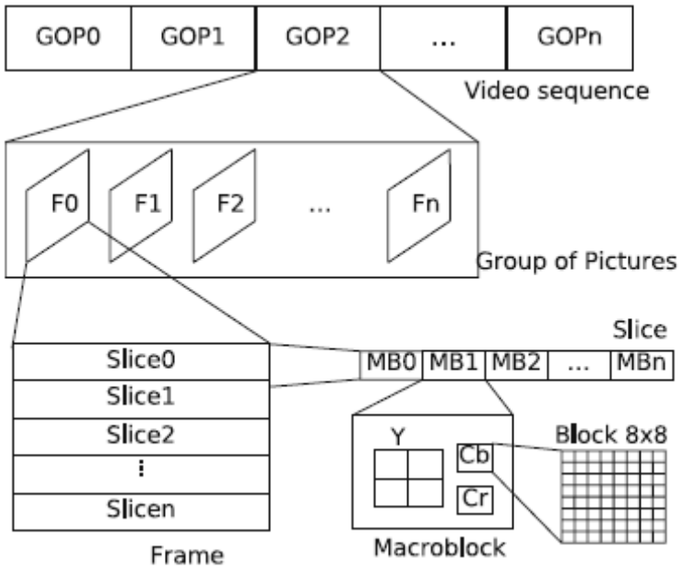


Fig. 11. GOP Structure of H.264 Video

parallelism requires a lot of memory for storing all the frames, and therefore this technique maps well to multicomputers in which each processing node has a lot of computational and memory resources. However, parallelization at the GOP-level results in a very high latency that cannot be tolerated in some applications. This scheme is therefore not well suited for multicore architectures, in which the memory is shared by all the processors, because of cache pollution.

1.3.5 Frame-level parallelism for independent frames

At frame-level in fig.12, the input video stream is divided in GOPs. Since GOPs are usually made independent from each other, it is possible to develop a parallel architecture where a controller is in charge of distributing the GOPs among the available cores.In a sequence of I-B-B-P frames inside a GOP, some frames are used as reference for other frames (like I and P frames) but some frames (the B frames in this case) might not. Thus in this case the B frames can be processed in parallel. To do so, a control/central processor assigns independent frames to different processors.

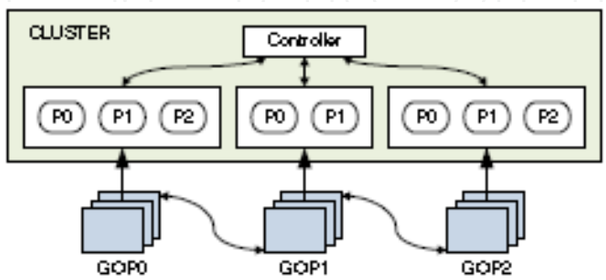


Fig. 12. Frame-level Parallelism

Frame-level parallelism has scalability problems due to the fact that usually there are no more than two or three B frames between P frames. The advantages of this model are clear:

PSNR and bit-rate do not change and it is easy to implement, since GOPs' independency is assured with minimal changes in the code. However, the memory consumption significantly increases, since each encoder must have its own Decoded Picture Buffer (DPB), where all GOP's references are stored. Moreover, real-time encoding is hardly implemented using this approach, making it more suitable for video storage purposes.However, the main disadvantage of frame-level parallelism is that, unlike previous video standards, in H.264 B frames can be used as reference [24]. In such a case, if the decoder wants to exploit frame-level parallelism, the encoder cannot use B frames as reference. This might increase the bitrate, but more importantly, encoding and decoding are usually completely separated and there is no way for a decoder to enforce its preferences to the encoder.

1.3.6 Slice-level parallelism

In slice-level parallelism (Fig. 13), frames are divided in several independent slices, making the processing of macroblocks from different slices completely independent. In the H.264/AVC standard, a maximum of sixteen slices are allowed in each frame. This approach allows exploiting parallelism at a finer granularity, which is suitable, for example, for multicore computers.In H.264 and in most current hybrid video coding standards each picture is partitioned into one or more slices. Slices have been included in order to add robustness to the encoded bitstream in the presence of network transmission errors and losses.

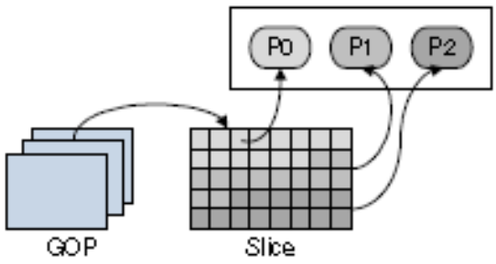


Fig. 13. Slice-level Parallelism

In order to accomplish this, slices in a frame should be completely independent from each other. That means that no content of a slice is used to predict elements of other slices in the same frame, and that the search area of a dependent frame can not cross the slice boundary [10, 16]. Although supports for slices have been designed for error resilience, it can be used for exploiting TLP because slices in a frame can be encoded or decoded in parallel. The main advantage of slices is that they can be processed in parallel without dependency or ordering constraints.This allows exploitation of slice-level parallelism (Rodriguez 2006) without making significant changes to the code.

However, there are some disadvantages associated with exploiting TLP at the slice level. The first one is that the number of slices per frame (sixteen in the H.264 standard) is determined by the encoder. That poses a scalability problem for parallelization at the decoder level. If there is no control of what the encoder does then it is possible to receive sequences with few (or one) slices per frame and in such cases there would be reduced parallelization opportunities. The second disadvantage comes from the fact that in H.264 the

encoder can decide that the deblocking filter has to be applied across slice boundaries. This greatly reduces the speedup achieved by slice level parallelism. Another problem is load balancing wherein the slices are created with the same number of MBs, and thus can result in an imbalance at the decoder because some slices are decoded faster than others depending on the content of the slice.

1.3.7 Macroblock level parallelism

There are two ways of exploiting MB-level parallelism: in the spatial domain and/or in the temporal domain. In the spatial domain MB-level parallelism can be exploited if all the intra-frame dependencies are satisfied. In the temporal domainMB-level parallelism can be exploited if, in addition to the intra-dependencies, interframe dependencies are satisfied.

1.3.8 Macroblock-level parallelism in the spatial domain

Usually MBs in a slice are processed in scan order, which means starting from the top left corner of the frame andmoving to the right, row after row. To exploit parallelism between MBs inside a frame it is necessary to take into account the dependencies between them. In H.264, motion vector prediction, intra prediction, and the deblocking filter use data from neighboring MBs defining a structured set of dependencies. These dependencies are shown in Fig. 14.

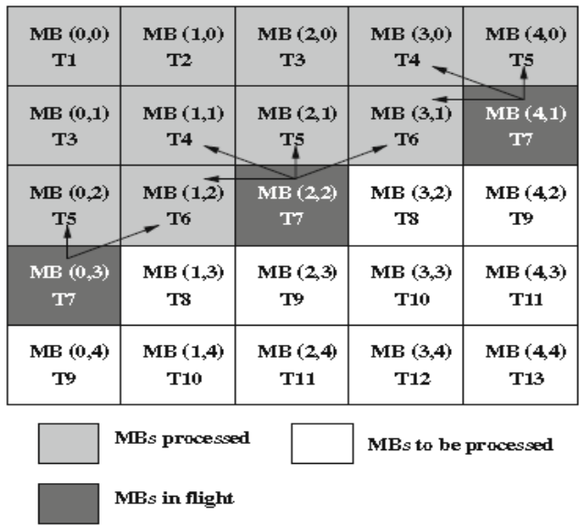


Fig. 14. 2D-Wave approach for exploiting MB parallelism in the spatial domain. The *arrows* indicate dependencies.

MBs can be processed out of scan order provided these dependencies are satisfied. Processing MBs in a diagonal wavefront manner satisfies all the dependencies and at the same time allows to exploit parallelism between MBs. We refer to this parallelization technique as 2D-Wave.

Fig.14 depicts an example for a 5×5 MBs image (80×80 pixels). At time slot T7 three independent MBs can be processed: MB (4,1), MB (2,2) and MB (0,3). The figure also shows

the dependencies that need to be satisfied in order to process each of these MBs. The number of independent MBs in each frame depends on the resolution. For a low resolution like QCIF there are only 6 independent MBs during 4 time slots. For High Definition (1920×1080) there are 60 independent MBs during 9 slots of time. Fig. 15 depicts the available MB parallelism over time for a FHD resolution frame, assuming that the time to decode a MB is constant.

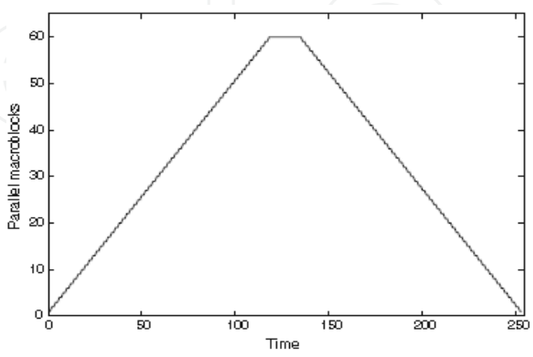


Fig. 15. MB parallelism for a single FHD frame using the 2Dwave approach.

MB-level parallelism in the spatial domain has many advantages over other schemes for parallelization of H.264. First, this scheme can have a good scalability. As shown before the number of independent MBs increases with the resolution of the image. Second, it is possible to achieve a good load balancing if a dynamicscheduling system is used. That is due to the fact that the time to decode a MB is not constant and depends on the data being processed. Load balancing could take place if a dynamic scheduler assigns a MB to a processor once all its dependencies have been satisfied. Additionally, because in MB-level parallelization all the processors/threads run the same program the sameset of software optimizations (for exploiting ILP and SIMD) can be applied to all processing elements. However, this kind ofMB-level parallelism has some disadvantages. The first one is the fluctuating number of independent MBs causing underutilization of cores and decreased total processing rate. The second disadvantage is that entropy decoding cannot be parallelized at the MB level. MBs of the same slice have to be entropy decoded sequentially. If entropy decoding is accelerated with specialized hardware MB level parallelism could still provide benefits.

1.3.9 Macroblock-level parallelism in the temporal domain

In the decoding process the dependency betweenframes is in the MC module only. MC can be regarded as copying an area, called the reference area, from the reference frame, and then to add this predicted area to the residual MB to reconstruct the MB in the current frame. The reference area is pointed to by a Motion Vector (MV). Although the limit to the MV length is defined by the standard as 512 pixels vertical and 2048 pixels horizontal, in practice MVs are within the range of dozens of pixels.

When the reference area has been decoded it can be used by the referencing frame. Thus it is not necessary to wait until a frame is completely decoded before decoding the next frame. The decoding process of the next frame can start after the reference areas of the reference

frames are decoded. Figure 16 shows an example of two frames where the second depends on the first. MBs are decoded in scan order and one at a time. The figure shows that MB *(2, 0)* of frame *i* + 1 depends on MB *(2, 1)* of frame *i* which has been decoded. Thus this MB can be decoded even though frame *i* is not completely decoded.

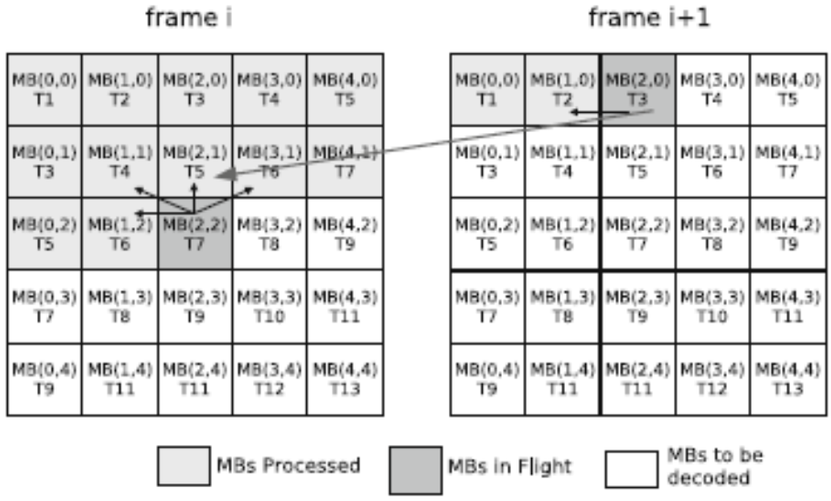


Fig. 16. MB-level parallelism in the temporal domain in H.264.

The main disadvantage of this scheme is the limited scalability. The number of MBs that can be decoded inparallel is inversely proportional to the length of the verticalmotion vector component. Thus for this scheme to be beneficial the encoder should be enforced to heavily restrict themotion search area which in far most cases is not possible. Assuming it would be possible, the minimum search area is around 3 MB rows: 16 pixels for the co-located MB, 3 pixels at the top and at the bottom of the MB for sub-sample interpolations and some pixels for motion vectors (at least 10). As a result the maximum parallelism is 14, 17 and 27MBs for STD, HD and FHD frame resolutions respectively.

The second limitation of this type of MB-level parallelism is poor load-balancing (Lai Ming che 2006) because the decoding time for each frame is different. It can happen that a fast frame is predicted from a slow frame and can not decode faster than the slow frame and remains idle for some time. Finally, this approach works well for the encoder which has the freedom to restrict the range of the motion search area. In the case of the decoder the motion vectors can have large values and the number of frames that can be processed in parallel is reduced.

In summary, parallelizing the entire process of the H.264 encoding particularly motion estimation will definitely end in optimized (Kun et.al 2009) performance provided the hardware/software requirements of the design are required. This will lead to a higher computation throughput achieved at the cost of appreciable load balance among the processor cores.

1.4 Applications

The H.264/AVC (T.Wiegand 2003) standard video format has a very broad application range that covers all forms of digital compressed video from low bit-rate Internet streaming

applications to HDTV broadcast and digital cinema applications with nearly lossless coding. With the use of H.264, bit rate savings of 50% or more are reported. H.264 video standard has a broad range of applications like broadcast television, streaming video, video storage and playback, video conferencing, aerial video surveillance, multi-sensor fusion, mobile video, medical imaging, Satellite image processing, video distribution etc..

The H.264/AVC standard was designed to suite a broad range of video application domains. However, each domain is expected to use only a subset of the available options. For this reason profiles and levels were specified to mark conformance points. Encoders and decoders that conform to the same profile are guaranteed to interoperate correctly. Profiles (G.J Sullivan et.al 2004) define sets of coding tools and algorithms that can be used while levels place constraints on the parameters of the bitstream. The standard defines 17 sets of capabilities, which are referred to as *profiles* as seen, targeting specific classes of applications and some of them are listed below in the Table.

	Baseline	Extended	Main	High	High 10	High 4:2:2	High 4:4:4 Predictive
I and P Slices	Yes	Yes	Yes	Yes	Yes	Yes	Yes
B Slices	No	Yes	Yes	Yes	Yes	Yes	Yes
Multiple Reference Frames	Yes	Yes	Yes	Yes	Yes	Yes	Yes
In-Loop Deblocking Filter	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CAVLC Entropy Coding	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CABAC Entropy Coding	No	No	Yes	Yes	Yes	Yes	Yes
Interlaced Coding (PicAFF, MBAFF)	No	Yes	Yes	Yes	Yes	Yes	Yes
8x8 vs. 4x4 Transform Adaptivity	No	No	No	Yes	Yes	Yes	Yes
Quantization Scaling Matrices	No	No	No	Yes	Yes	Yes	Yes
Separate Cb and Cr QP control	No	No	No	Yes	Yes	Yes	Yes
Separate Color Plane Coding	No	No	No	No	No	No	Yes
Predictive Lossless Coding	No	No	No	No	No	No	Yes
	Baseline	Extended	Main	High	High 10	High 4:2:2	High 4:4:4 Predictive

- **Baseline Profile (BP): T**he simplest profile mainly used for video conferencing and mobile video.
- **Main Profile (MP): I**ntended to be used for consumer broadcast and storage applications, but overtaken by the high profile.
- **Extended Profile (XP): I**ntended for streaming video and includes special capabilities to improve robustness.
- **High Profile (HiP) I**ntended for high definition broadcast and disc storage, and is used in HD DVD and Blu-ray.

Table 2. H.264 Profiles

Getting the best performance from an H.264/AVC codec generally involves selecting the best coding options or coding mode for each unit of data in the video bitstream. The key considerations in choosing a specific video application (profile/level) depends on the following parameters

- Desired image quality (resolution, frame rate)
- Available bandwidth/storage
- Available processing power
- External factors (camera motion, scene motion,transmission delays)

These variables are vital as they will ultimately impact the performance on which H.264/AVC (profile/level) compression methodology is best for your application. For low lvele applications, the industry uses H.264 Baseline, Constrained Baseline or in some cases, a much lower performance profile with many quality features simply "turned off because they do not have the computer power in the camera (Tiago 2007) to support the higher quality features. Mostly the current profile used in majority is main profile for H.264 encoding, as it provides higher video quality and high performance for the same bandwidth compared to the baseline profile.

Aerial Image Surveillance

Fig. 17. Aerial Video Surveillance

With the development of video coding technologies, digital video surveillance (Nirmal Kumar 2010) has become a very hot application in the recent few years. Specific

applications are developed with core functions that extend across civilian and military application areas. Unmanned aerial vehicle (UAV) surveillance and reconnaissance programs increasingly need methods for optimally packaging and distributing information. H.264 supports the collection, formatting, storage and dissemination of "raw" data from real time video capture and image exploitation using embedded technology for surveillance & reconnaissance application, Enhanced fusion vision for situational awareness application, Automatic vision inspection system for quick inspection of components in a manufacturing industry. Also it supports for high-end resolution for remote sensing images and data from satellite.

Airborne surveillance has been widely used in different range of applications in civilian and military applications, such as search and rescue missions, border security, resource exploration, wildfire and oil spill detection, target tracking, surveillance, etc. The unmanned airborne vehicle (UAV) is equipped with special sensors (day / night) to image objects in ground and assigns the actual recognition task (surveillance) to the crew or record image data and analyze them off-line on the ground. Pilot less airborne vehicle with sensor carrying platforms transmit data to a ground control station for analysis and data interpretation.

2. Conclusion

There is likely to be a continued need for better compression efficiency, as video content becomes increasingly ubiquitous and places unprecedented pressure on upcoming new applications in the future. At the same time, the challenge of handling ever more diverse content coded in a wide variety of formats makes reconfigurable coding a potentially useful prospect. To summarize it, we presented an overview of H.264 motion estimation and its types and also the various estimation criterion that decides the complexity of the chosen algorithm. We then probed into the available scalability of parallelism in H.264. Finally we focused on an application which is highly sought in the research environment and its advantages in a more elaborate manner.

3. References

A. Rodr´ıguez, A. Gonz´alez, and M. Malumbres, (2006), Hierarchical parallelization of an H.264/AVC video encoder. *In Int. Conf. on Parallel Computing in Electrical Engineering* (PARLEC), pages 363–368.

Alan Conrad Bovik, (2009). *The Essential Guide to Video processing,* Elsevier, 978-0-12-374456-2, USA.

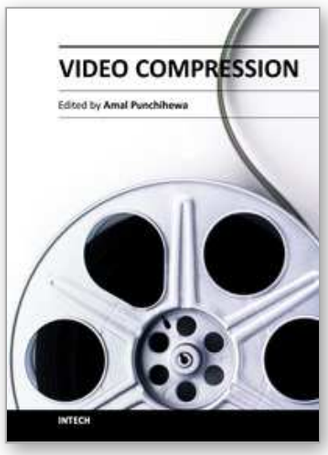
Alois M.Bock (2009), "*Video Compression Systems from first Principles to Concatenated Codecs*", The Institution of Engineering and Technology, 978-0-86431-963-6, UK

Florian H. Seitner, Michael Bleyer , Margrit Gelautz · Ralf M. Beuschel, (2010) Evaluation of data-parallel H.264 decoding approaches for strongly resource-restricted architectures*, Springer Science Business Media*, LLC,ISSN 11042-010-0501-7

Gary J. Sullivan, Pankaj Topiwala, and Ajay Luthra, (2004), The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range

Extensions, *SPIE Conference on Applications of Digital Image Processing* XXVII, August, 2004.
http://en.wikipedia.org/wiki/H.264
http://en.wikipedia.org/wiki/Video_compression
http://en.www.wikipedia.com
http://www.chiariglione.org
http:/www.vodex.com
I.E.G. Richardson, (2003). *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*, John Wiley & Sons, Ltd, 0-470-84837-5, UK
Iain E.Richardson, (2010) .*The H.264 Advanced Video Compression Standard* John Wiley & Sons ltd, 978-0-470-51692-8, UK.
Kue-Hwan Sihn, Hyunki Baik, Jong-Tae Kim, Sehyun Bae, Hyo Jung Song, (2009), Novel Approaches to Parallel H.264 Decoder on Symmetric Multicore Systems, *Proceedings of IEEE.*
Kun Ouyang Qing Ouyang Zhengda Zhou, (2009), Optimization and Implementation of H.264 Encoder on Symmetric Multi- Processor Platform, *Proceedings of IEEE Computer Society, World Congress on Computer Science and Information Engineering.*
Lai Mingche, Dai Kui, Lu Hong-yi, Wang Zhi-ying, 2006, A Novel Data-Parallel Coprocessor for Multimedia Signal Processing",IEEE Computer Society.
Nuno Roma and Leonel Sousa, 2002, Efficient and Configurable Full-Search Block-Matching Processors, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 12, December 2002.
P.Nirmal Kumar, E.MuraliKrishnan, E.Gangadharan, (2010), Enhanced Performance of H.264 using FPGA Coprocessors in Video Surveillance, *Proceedings of IEEE Computer Society, ICSAP,* India.
Philip P. Dang (2009), Architecture of an application-specific processor for real-time implementation of H.264/AVC sub-pixel interpolation, *Journal of real-time Image Processing*, Vol.4, pp. 43-53
Romuald Mosqueron, Julien Dubois, Marco Mattavelli, and David Mauvilet "Smart Camera Based on Embedded HW/SW Coprocessor", *EURASIP Journal on Embedded Systems*, Hindawi Publishing Corporation, Volume 2008, Article ID 597872, 13 pages,doi:10.1155/2008/597872.
T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, (2003), Overview of the H.264 / AVC Video Coding Standard", *IEEE Transactions on Circuit and Systems for Video Technology*, VOL. 13, NO. 7, July 2003.
Tiago Dias ,Nuno Roma , Leonel Sousa, Miguel Ribeiro (2007) Reconfigurable architectures and processors for real-time video motion estimation, *Journal of real-time Image Processing*, Vol.2, pp. 191-205
Youn-Long Steve Lin, Chao-Yang-Kao, Hung-Chih Kuo, Jian-Wen Chen, (2010) "*VLSI Design for Video Coding, H.264/AVC Encoding from Standard Specification to Chip*", Springer, 978-1-4419-0958-9, USA
Yu-Wen Huang, Ching-Yeh Chen, Chen-Han Tsai, Chun-Fu Shen And Liang-Gee Chen, (2006) "Survey on Block Matching Motion Estimation Algorithms and Architectures with New Results, *Journal of VLSI Signal Processing , Springer* Vol.42, pp.297–320, 2006.

Zhibo Chen, Jianfeng Xu, Yun He, Junli Zheng (2006) Fast integer-pel and fractional-pel motion estimation for H.264/AVC, *International Journal of Visual Communication*, Vol.17,pp. 264–29,



Video Compression

Edited by Dr. Amal Punchihewa

ISBN 978-953-51-0422-3
Hard cover, 154 pages
Publisher InTech
Published online 23, March, 2012
Published in print edition March, 2012

Even though video compression has become a mature field, a lot of research is still ongoing. Indeed, as the quality of the compressed video for a given size or bit rate increases, so does users' level of expectations and their intolerance to artefacts. The development of compression technology has enabled number of applications; key applications in television broadcast field. Compression technology is the basis for digital television. The "Video Compression" book was written for scientists and development engineers. The aim of the book is to showcase the state of the art in the wider field of compression beyond encoder centric approach and to appreciate the need for video quality assurance. It covers compressive video coding, distributed video coding, motion estimation and video quality.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Murali E. Krishnan, E. Gangadharan and Nirmal P. Kumar (2012). H.264 Motion Estimation and Applications, Video Compression, Dr. Amal Punchihewa (Ed.), ISBN: 978-953-51-0422-3, InTech, Available from: http://www.intechopen.com/books/video-compression/h-264-motion-estimation-and-applications

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen