A DYNAMIC PREDICTIVE SEARCH ALGORITHM FOR FAST BLOCK-BASED MOTION ESTIMATION

By

Behnaz Abdoli

Bachelor of Applied Science in

Electronics and Computer Engineering

Yazd University

Yazd, Iran, 2004

A thesis

Presented to Ryerson University

In partial fulfillment of the

requirements for the degree of

Master of Applied Science

In the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2012

Author's Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

* Signature

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

* Signature

ABSTRACT

Title of Thesis:

A DYNAMIC PREDICTIVE SEARCH ALGORITHM FOR FAST BLOCK-BASED MOTION ESTIMATION

Thesis Submitted By:

Behnaz Abdoli

Optimization Problems Research and Application Laboratory (OPR-AL)

ELCE, Master of Applied Science, Ryerson University 2012

Thesis Directed By:

Dr. Reza Sedaghat

Electrical and Computer Engineering Department,

Ryerson University

Predictive fast Motion Estimation (ME) algorithms have been widely used in video CODECs due to their performance efficiency and low computational complexity. In this thesis, a new block-based fast motion estimation technique named Dynamic Predictive Search Algorithm (DPSA) is developed, which can be considered in predictive zonal search category.

The proposed approach is based on the observation that temporally and spatially adjacent macro-blocks are not just statically correlated, but also dynamic alterations in their motion content are highly coherent. DPSA introduces a new set of six candidate predicted motion vectors. For early termination criteria, DPSA modifies termination procedure of already existing EPZS algorithm.

Performance of this newly proposed algorithm has been compared to four other state-of-the-art algorithms implemented on JVT, H.264 standard software platform.

Experimental results have proven that DPSA accomplishes up to 38% compression ratio enhancement achieved by a process with more 14.75% less computational complexity and up to

0.47 dB higher PSNR values over the EPZS. It also manages to have up to 13% speed up over EPZS algorithm.

Because of its simplicity and low computational complexity DPSA is energy efficient for portable video processing in computation- or power-constrained applications and easy to be implemented on both FPGA- and Microcontroller-based embedded systems. Also, higher compression ratio makes DPSA more compatible with limited capacity storage media, and limited band-width transmission networks.

Acknowledgement

I am very thankful to my supervisor, Dr. Reza Sedaghat for his thoughtful guidance and OPR-AL members for their endless support.

I would like to acknowledge and extend my heartfelt gratitude to my husband, Manuchehr Taghiloo who has made the completion of this research possible with his valuable comments, and supports throughout tough moments and difficulties.

I am deeply indebted to my parents for their great guidance and sacrifice throughout my whole life. Further I highly owe them for being a constant source of love and motivation throughout my life.

I am also thankful to my friends, who encouraged me to accomplish my goals.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER1

INTRODUCTION

1.1 Background

Digital video data flow from a source to a destination consists of two specific parts: compression modules, known as Encoders, in which massive data size of 'raw' digital video is reduced for transmission or storage, and decompression part, or Decoder that reconstructs the video sequence for display. Therefore, compression is an essential component of multimedia services, and good compression and decompression processes is key for providing better image quality products. Moreover, the need for better compression tools has led to developing further standards for video compression, such as the recent ones, "MPEG-4" [1] and "H.264/AVC" [2].

This chapter is a brief review on main characteristics of digital images and video signals and examines concepts such as sampling formats and quality metrics, as well as introducing structure of video coding systems.

The first subject is sampling in colour images. To represent a colour images, at least three numbers per pixel position are required for brightness, or luminance (Luma), and to indicate the colour, or chrominance (Chroma). Two methods are well-known for this purpose: RGB colour space [3] and YUV.

[4] [5] In the RGB colour space, each pixel is formed with three numbers that describe the relative proportions of three primary color components Red, Green and Blue. In this model the three colour contents are stored with the same resolution. When one of colour component has the strongest intensity, the pixel is visualized with that primary colour, and when two or more components have almost the same intensity, then the colour is a shade of a secondary colour (such as purple, yellow cyan) or it can be white or black.

The other colour model is YCbCr colour space, also known as YUV. This model is based on a fact that human vision is less sensitive to colour than to luminance. Hence, it is more efficient to

represent a colour image by luminance (Y) component, with higher resolution, and Cb; Cr; and Cg that present the colour intensity or Chrominance components, with lower resolution. This decreases size of data required to be stored or transmitted with no obvious difference on visual quality. Luma component, Y, can be calculated as an average of R, G and B with weighting factors of K_r; K_g; and K_b:

$$Y = K_r R + K_g G + K_b B$$

The chrominance is defined as the difference between R, G or B proportions and the Luma:

$$C_b = B - Y; \qquad C_r = R - Y; \qquad C_g = G - Y$$

However, since $(Cb + Cr + Cg)$ is a constant, only two Chroma components need to be stored or transmitted. The third component can always be calculated from the other two. So, in the YUV colour space, only the Luma (Y) and blue and red Chroma (Cb, Cr), also known as U and V are transmitted. Representing Chroma with a lower resolution than Luma in this way is a simple and yet, effective model of image compression.

Three patterns for sampling resolution ratio for YUV format are supported by MPEG-4 Visual and H.264/AVC standards:

4:4:4 sampling in which the three components (Y, U and V) have the same number of samples for each component at every pixel position. Thus for every four luminance samples there are four Cb and four Cr samples. 4:4:4 sampling keeps the full fidelity of the chrominance components.

In 4:2:2 sampling, the chrominance components vertical resolution is the same as the Luma but their horizontal resolution is half of Luma. In other words, for every four luminance samples in the horizontal direction there are two Cb and two Cr samples. 4:2:2 video is useful for high-quality colour reproduction.

The popular sampling format is 4:2:0, in which Cb and Cr each have half the horizontal and vertical resolution of Y. 4:2:0 sampling is suitable for consumer applications such as video conferencing, digital television and digital versatile disk (DVD) storage. Because each colour difference component contains one quarter of the number of samples in the Y component, 4:2:0 YUV video requires exactly half as many samples as 4:4:4 (or RGB) video.

For example, in a 720 x 576 pixels image resolution:

Y component is represented with eight bits for each sample in all three ratios.

In 4:4:4 Cb, Cr components are eight bits for each sample, and the total number of bits is:

$$720 \times 576 \times 8 \times 3 = 9953280 \text{ bits per frame}$$

In 4:2:2 Cb, Cr components resolution is 360 x 288 samples, each eight bits and the total number of bits:

$$(720 \times 576 \times 8) + (360 \times 288 \times 8 \times 2) = 4976640 \text{ bits per frame}$$

In 4:2:0 versions require half as many bits as the 4:4:4 versions and the total number of bits is: 4976640 bits per frame

As well, there are different intermediate formats to standardize the horizontal and vertical resolutions in pixels to capture YUV video sequences prior to compression and transmission, such as SQCIF, QCIF, CIF, 4CIF. The CIF (Common Intermediate Format), which is the basis for other formats, means 352 x 288 Y samples per frame. For example in 4:2:0 resolution there are 352 x 288 x 8 = 811008 bits for Y samples and half of this much (405504 bits) for Chroma samples. Some of the most popular formats are listed in table 1.

Table 1: Resolution formats

Format	Y resolution (Hrzntl x Vrtcl)	bits per frame for 4:2:0 resolution
4CIF	704 x 576	4866048
CIF	352 x 288	1216512
QCIF	176 x 144	304128
SCIF	128 x 96	147456

The choice of frame resolution depends on the application and available storage or transmission capacity. For example, 4CIF is appropriate for standard-definition television and DVD-video; CIF and QCIF are popular for videoconferencing applications; QCIF or SQCIF are widely used in mobile multimedia applications where the display resolution and the bitrate are limited.

Most video standards, particularly the most recent ones: MPEG-4 Visual and H.264/AVC, follow the so-called Block-based video coding [6] approach, in which each coded frame is split into fixed or variable size non-overlapping blocks of associated Luma and Chroma samples, known as Macro-blocks (MB). Usually, each MB covers a rectangular area of 16 x 16 samples of the Luma component and 8 x 8 samples of each of the two Chroma components. Macro-blocks are the basic building units of the standard for which the decoding process is specified, and all Luma and Chroma samples of a MB are encoded or decoded at a time. However, in H.264/AVC standard, the luminance component of a MB also, could be partitioned into 16 x 16, 16 x 8, 8 x 16, 8 x 8, 8 x 4, 4 x 8 or 4 x 4 blocks in a similar way as depicted in Figure 1.

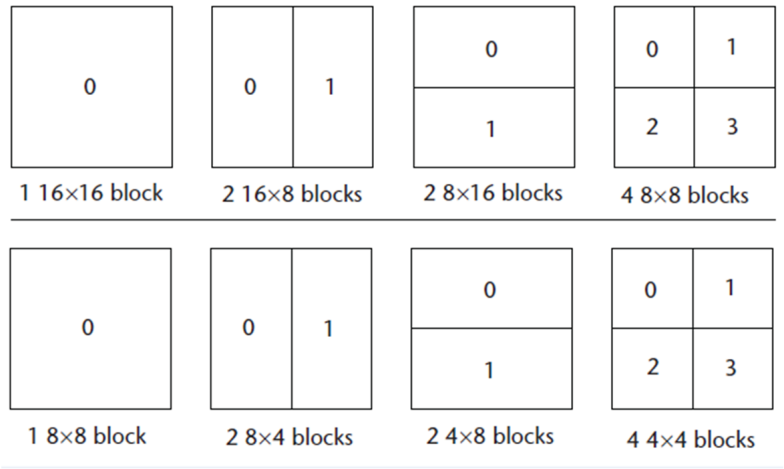

Figure 1: Variable block sizes

1.2 Video coders

As mentioned earlier, a video coding system consists of a pair of *encoder* and *decoder* components and the whole system is known as a *CODEC*. The encoder is in charge of video compression, which is the process of compacting digital video sequences into smaller number of bits. The decoder, on the other hand, converts the compressed form back into an approximation version of the original video data.

Video compression enables more efficient use of transmission and storage resources. Even with constant advances in storage and transmission capacity, compression is still likely to be an essential component of multimedia services for many years to come. An information-carrying signal may be compressed by removing redundancy from the signal. Most video compression

algorithms operate by removing redundancy in the temporal, spatial and/or frequency domains to achieve compression.

In the temporal domain, there is usually a high correlation between following frames of video that were captured at around the same time. Especially in high sampling rates (the frame rate) temporally adjacent frames are often highly correlated. For example, when the sequence is captured from a camera at 30 frames per second, there is little change between the two frames in the short interval of 1/30 of a second. There is clearly significant temporal redundancy, and most of the image remains unchanged between successive frames.

In the spatial domain, there is usually a high correlation between pixels that are close to each other, inside a frame. Because neighbouring samples most possibly are part of the same object and have the same or very close colour intensity and luminance, hence their values are often very similar.

Frequency domain compression is based on the fact that human eyes and brain are more sensitive to lower frequencies [4]. In a video frame, if we low-pass filter the background region, by removing some of the higher-frequency content, the image becomes smoother and less information is required to store or transmit. However, the image is still recognisable for human eye, with no visible quality distortion.

By removing spatial, frequency and/or temporal redundancies it is possible to compress the data significantly at the expense of an acceptable range of distortion. A video encoder consists of three main functional units: a temporal model, a spatial model and an entropy encoder. The H.264 and MPEG-4 Visual standards assume a CODEC *model* that uses block-based motion compensation, transformation, quantisation and entropy coding. Each of these components processes one MB at a time.

There are some specific orders to code the MBs. One of the most popular ones is called raster scan. If the frame is processed in raster order, then coding starts from the most top-left pixel in the frame and continues to the right first. When one whole line is finished, it jumps to the most left pixel in the next line. Raster scan order is demonstrated in figure 2.

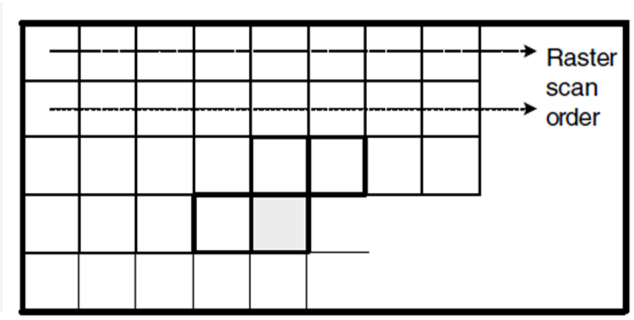

Figure 2: Raster Scan Order

The first functional unit in encoder is temporal model. The input to the temporal model is an uncompressed video sequence. The temporal model attempts to reduce temporal redundancy by exploiting the similarities between neighbouring video frames, usually by constructing a prediction of the current video frame. In MPEG-4 Visual and H.264, the prediction is formed from one or more previous or future frames and is improved by compensating for differences between the frames (motion compensated prediction). The output of the temporal model is a residual frame, created by subtracting the prediction from the actual current frame, and a set of motion vectors describing how the motion was compensated.

The residual frame forms the input to the spatial model which makes use of similarities between neighbouring samples in the residual frame to reduce spatial redundancy. In MPEG-4 Visual and H.264 this is achieved by applying a transform to the residual samples and quantizing the results. The transform converts the samples into another domain in which they are represented by transform coefficients. The coefficients are quantised to remove insignificant values, leaving a small number of significant coefficients that provide a more compact representation of the residual frame. The output of the spatial model is a set of quantized transform coefficients.

Temporal model parameters are usually motion vectors and spatial model parameters are coefficients. These parameters are compressed by the entropy encoder to remove statistical redundancy in the data such as representing common vectors and coefficients by short binary codes. A compressed sequence consists of coded motion vector parameters; coded residual coefficients; and header information form the bit stream exiting from encoder.

The video decoder reconstructs a video frame from the compressed bit stream. After decoding the spatial model, coefficients and motion vectors are decoded by an entropy decoder to

reconstruct an estimated version of the residual frame. Then the decoder uses the motion vector parameters and one or more reference frames, to create a prediction of the current frame. Eventually, the frame itself is reconstructed by adding this prediction to the residual frame.

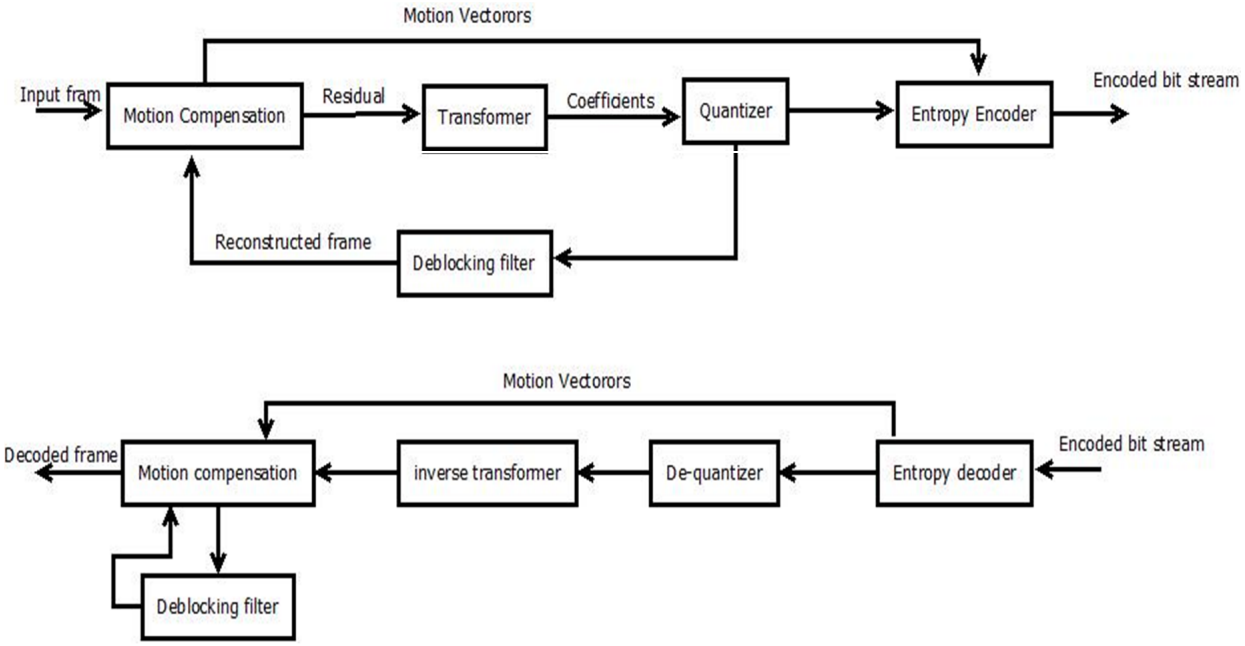

Figure 3: Digital video CODEC block-diagram

1.3 Quality measurement

In order to evaluate and compare video processing systems, determining the quality of the encoded video images is required. The most popular scale to measure the so-called objective quality of compressed video sequences is a logarithmic parameter, known as Peak Signal to Noise Ratio (PSNR) which depends on the mean squared error (MSE) between an original and a compressed video frame, as in equation (1-1).

$$PSNR_{dB} = 10\ Log_{10}[(2^n-1)^2/MSE] \quad n:\#\ of\ bits/\ image\ sample. \qquad (1\text{-}1)$$

PSNR is a very popular, and widely used to evaluate the fidelity between compressed and decompressed video images, because it can be calculated easily and quickly. To have a general perspective, an image with PSNR of 30.6 dB reflects a good quality, while that same image with PSNR of 28.3 dB is considered the poorer image quality.

In a colour image, PSNR is attributed to three different values, for Y; U and V samples. As explained, Chroma components can be represented with a lower resolution than Y, to reduce the amount of data required to be stored or transmitted with no obvious difference on visual quality. That is why in many studies Y samples illustrate the quality of encoded images. However, in some cases where the color content in a frame is very diverse and it changes dramatically block to block (figure 4). In these cases, it is important to evaluate U and V samples to determine the quality of encoded sequence.

Figure 4: "Flower" and "Mobile"; two examples of colourful sequences

Another factor that describes the performance of a CODEC is compression ratio. For a fixed resolution and on a single image, the higher the compression ratio is in an encoder the lower number of bits is required to be stored or transmitted and lower band-width network or smaller media storage is needed.

CHAPTER2

Block based Motion Compensation

2.1. Motion compensation

The first compression module in encoders is Motion Compensation (MC) which exploits temporal redundancy between frames and describes a picture in terms of the transformation of a reference picture to the current picture. MC is based on the fact that usually, for most of the frames of a sequence, the only difference between one frame and the next one is the result of either the camera moving or an object in the frame moving. With sampling rates like 30 frames per second, the motion of objects or camera in just 1/30 of a second is clearly very small. This means much of the information that represents one frame will be the same as the information used in the next frame.

In most recent visual coding standards including MPEG-4 Visual, and H.264/AVC, the macro-block (MB), corresponding to a MxN-pixel region of a frame, is the basic unit for motion compensated. For instance, in a video material in 4:2:0 format, a macro block is organised as a 16 x 16-pixel region representing 256 luminance samples, 64 blue chrominance samples and 64 red chrominance samples, giving a total of six 8 x 8 blocks. An MPEG-4 Visual or H.264 CODEC processes each MB at a time and provides a Motion Vector (MV) associated to that MB.

Finding the MV of a MB involves Motion Estimation (ME) process, in which, a macro block of MxN-sample region in a reference frame is found that closely matches the current macro block. The reference frame is a previously encoded frame from the sequence that can be before or after the current frame in display order. Inside an area centred on that MB position in a previously encoded frame, called reference frame (search window) is searched, to find a MxN -pel region that closely matches the current MB. This is carried out by comparing the *current MB* in the current frame with the possible MxN-samples regions in the search window to find the block that

minimises a matching criterion, such as most widely-used Sum of Absolute Differences (SAD) value.

Then MC process continues with subtracting the selected the best matching MB in the reference frame from current MB to produce a residual MB of Luma and Chroma samples that will be encoded and transmitted together with a Motion Vector (MV), describing the position of the best matching region relative to the current MB's position.

SAD is the most popular distortion criterion to measure the residual energy. For an MxN block, it can be described as equation (2-1):

$$SAD\ (d) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (|C(x, y) - R(x + dx, y + dy)|) \qquad (2\text{-}1)$$

Where d = (dx, dy) is the MV, $C(x, y)$ and $R(x, y)$ are intensity of given pixel in current and reference MB, respectively.

The region with minimum SAD indicates the offset that produces a minimal residual energy and this is likely to produce the most matching MB to the current MB.

The decoder uses the received motion vector to re-create the predictor region and decodes the residual block, adds it to the predictor and reconstructs a version of the original block.

Block-based motion compensation is popular for a number of reasons. It is relatively straightforward and computationally tractable, it fits well with rectangular video frames and with block-based image transforms (e.g. the Discrete Cosine Transform) and it provides a reasonably effective temporal model for many video sequences.

However, there are a some disadvantages: [4] for example real objects rarely have neat edges that match rectangular boundaries; objects often have movements that are fractional number of pixel positions between frames and many types of object motion are hard to compensate for using block-based methods, such as deformable objects, rotation and warping, and complex motions. In spite of these drawbacks, block-based motion compensation is the base of the temporal model used by all current video coding standards.

Motion estimation is the most computationally intensive and time consuming module of encoders. Many studies are conducted to find an algorithm to reduce the computational complexity of ME process. These algorithms can be generally categorized in three groups: Full search method; fast search algorithms; and predictive search algorithms.

2.2. Full search method

Full Search algorithm [7] is the conventional motion estimation technique that calculates SAD value at all $(2w+1)^2$ possible MBs in the search window with size of w pixels (–w to w samples around position (0,0), the position of the current MB) to find the best matching block. This method is guaranteed to find the minimum SAD in the search window but it is computationally intensive.

The conventional full search process starts with searching the most top-left of the window (position [–w, –w]) and the search proceeds in raster order until all positions have been evaluated. In a typical video sequence, most motion vectors are found around (0,0). That is why some researchers recommend simplifying the computation of the full search algorithm by starting the search from (0,0) position, and proceeding in a spiral pattern around this location. They define an early termination condition such as a SAD value threshold. If the calculated SAD for zero MV is less than that threshold, the computation is stopped from further searching.

These new Full search approaches are still popular due to their accuracy, but even with the use of early termination, Full Search motion estimation is very computationally intensive and can be undesirable and very expensive for some applications especially where real time encoding is required.

2.3. Fast search algorithms

Fast motion estimation algorithms, are introduced in effort to reduce the computational complexity of intensive full search method. These algorithms operate by calculating the SAD criterion at a subset of locations, with a specific pattern, within the search window, instead of all over the search window. In computation- or power-limited applications, fast ME algorithms are preferable.

Many fast ME patterns have been proposed over the last decades. The most popular ones include: Three Step Search (TSS) [8], New Three Step Search (NTSS) [9], Cross Search [10][11], Diamond Search [12][13], Hexagon search [14] and the hybrid combinations of them. In the next chapter some of these algorithms are described to illustrate how fast ME techniques work.

Each fast ME technique can be considered as a trade-off between three main criteria: encoded image quality; compressed bit-rate; and computational complexity. As mentioned in chapter 1, image quality is measured with PSNR parameter. Computational cost in a fast ME technique is measured by the factor of the required number of searching points to find the MV of each MB, whilst the bit-rate of encoded streams represents compression efficiency of the encoder. Another parameter to compare different algorithms is ME process time. Clearly, full search method has extremely long process time, and all fast search algorithms are significantly effective in terms of speed up over full search.

Ideally, in a constant sampling frame rate, all ME algorithms are meant to achieve as few bit-rate as possible and the highest possible quality (PSNR), with as low number of search points as possible. Most fast ME patterns achieve lower number of search points with the price of lower quality and compression ratio. In fact, in case of constant frame rate systems, PSNR and data bit-rate are anti-correlated.

The more accurate MV estimation leads to further fidelity between encoded and original videos. As a result, the energy content of residual frames is much lower, and fewer data bits are required to be transmitted or stored, which means lower bit-rate. Also, it results in higher quality images, measured in high PSNR values for Luma and Chroma samples. Hence, both parameters represent the accuracy of ME process.

Fast search methods just search a few positions inside the search window and assume the minimal SAD verdict is close enough to the ultimate MV. Therefore there is a chance of getting trapped in a local minima, and having a less optimal answer. Hence, these plain fast search algorithms could not compete with the high accuracy of full search.

2.4. Predictive search algorithms

Predictive search algorithms are a new category of fast ME techniques that introduce some solutions for lowering the computational intensity of ME, with a performance much closer to conventional full search method. These solutions can be classified into two steps: one is predicting the most likely initial sub-set of search points based on the temporal and spatial correlation between MV of current MB and previously coded MBs; and the second step is defining some early termination conditions to avoid being trapped in local minima and enhance the speed up.

In these approaches, a set of correlated MB to the current MB is introduced. All these blocks must be already encoded at the time of coding current block. In most existing predictive algorithms, this set includes some adjacent MBs in current frame and, some blocks in reference frame, or other close frames in time order, which are already encoded. MV of these blocks is used to predict a set of initial search points. Then, SAD parameter for all initial points are calculated and the verdict with minimum SAD is set as the origin of a local search pattern, which could be a diamond, square or hexagon shaped pattern.

Most recent techniques define an early stop criterion, and in each step, that criterion is checked. If the condition is met the processor stops further searching and returns a found vector as the final MV of current MB. This termination condition can be a SAD value threshold. As soon as a verdict meets the defined threshold, further searching is terminated and location of that verdict will define the ultimate MV.

Many proposed predictive algorithms first examine (0,0) position, due to the high possibility of zero MV [15]. This is why they are called "Predictive Zonal Search Algorithms".

Some examples of existing predictive zonal search approaches are described in chapter 4, to clarify how this type of coding techniques functions. In chapter 5, a new predictive algorithm will be proposed, based on dynamic and static correlated macro-blocks, which improves performance of other methods, significantly.

CHAPTER3

Fast Motion Estimation

In this chapter four main fast motion estimation patterns, including three-step search; cross search; diamond search; and hexagon search patterns are explained.

3.1. Three-step search (TSS) algorithm

Three-step search was first introduced around 1993, and became one of the most popular algorithms for MC, due to its simplicity and efficiency. Since then many studies were conducted to improve this algorithm including a new three-step search; and enhanced three-step search that were based on the original TSS algorithm. Two of these proposed ideas are described here.

3.1.1. New Three-Step Search (NTSS)

Initial search points in New Three-step Search (NTSS) algorithm [9] for a search window size of 7 is shown in figure 5. In the first step, 17 points including the center; 8 points with 1 pixel distance from the center; and 8 points on the larger 9x9 grid are checked. If the minimum SAD happens to be found at the center of the search window, the search will stop. If the minimum SAD point is one of the eight points on the 3x3 grid, again another 3x3 grid pattern is formed around that center, to check three or five extra points and the minimum SAD found in this step is final MV. Otherwise the large 9x9 search window size is reduced by half and the center moves to the minimum SAD point in Step1, the algorithm is repeated until the search distance cannot be subdivided further.

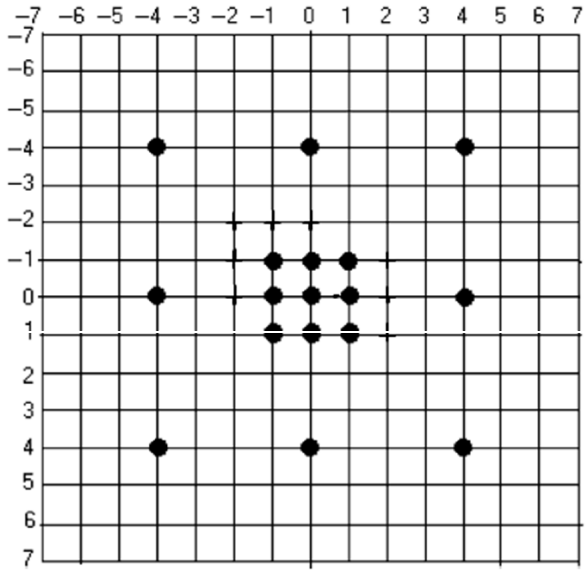

Figure 5: 17 search points in NTSS

3.1.2. Efficient Three-Step Search (ETSS)

In order to exploit the center-biased characteristics of motion vector distribution in real-world video sequences, ETSS utilizes a small diamond search pattern in the search window center [8]. Figure 6 shows the search pattern used in the first step of ETSS for a search window size of 7. Thus, in the first step, a total of 13 points will be searched instead of 17 points in N3SS. If the winning point is found to be on the center of search window, it will stop further searching. If the minimum SAD point is one of the eight points on the large 9x9 grid, the following process will be the same as in NTSS. If the minimum is one of the four points on the small diamond, the small diamond center is set to the origin of another small diamond pattern, and another three points will be examined. The center of the small diamond is moved to the minimum SAD point each time until the minimum is found in the center of small diamond. Two main differences between ETSS and NTSS include:

1- A small diamond pattern is used instead of a square pattern in the central area
2- Unrestricted search step for the small diamond rather than a single movement for the small square.

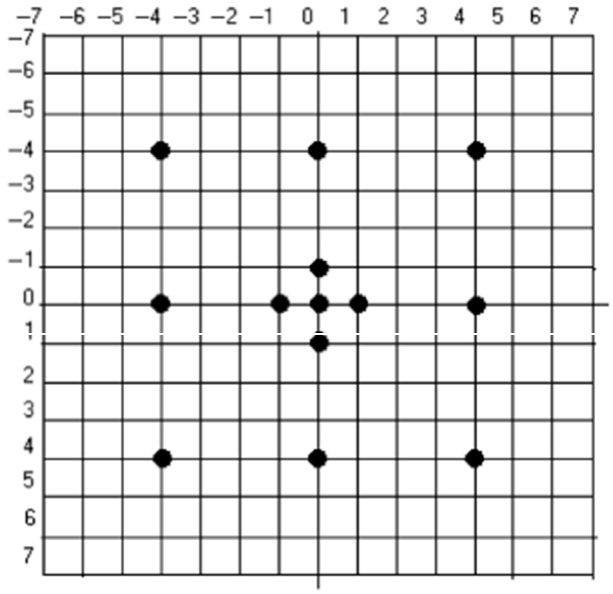

Figure 6: ETSS search point

3.2. Diamond search algorithm

3.2.1. Unrestricted Center-biased Diamond Search

A basic diamond search pattern is named UCBDS [17]. This fast search pattern can be summarized in following steps:

Step1: The original diamond pattern is formed at the center of the search window, that we call it (c,c). The SAD is evaluated for each of the *nine* candidate search points. If the minimum SAD point is found to be at the center, go to step5; otherwise, go to step2.

Step2: If the minimum SAD point in the previous search step is located at one of the four vertices (c-2,c); (c+2,c) ; (c,c-2) ; (c,c+2), then go to step3. Else, if it is located at one of the four possible faces of the previous diamond (c-1,c-1); (c+1,c+1) ; (c-1,c+1) ; (c+1,c-1), then go to step4.

Step3: Another diamond pattern is formed around the minimum SAD point (updating the center (c,c)). *Five* new candidate search points are evaluated.

Step4: Another diamond pattern is formed around the minimum SAD point (updating the center (c,c)). *Three* new candidate search points are checked. Note that any candidate point that extends

beyond the search window is ignored. If the minimum SAD is found at (c,c), then go to step5 otherwise, go to step2 to continue the next search step.

Step5: The shrunk diamond pattern is used with the same center (c,c). Now, the final *four* internal points of the previous diamond are searched. Similarly, any internal candidate point that extends beyond the search window is also ignored. The candidate point that gives the minimum SAD is chosen as the final motion vector (m_x, m_y).

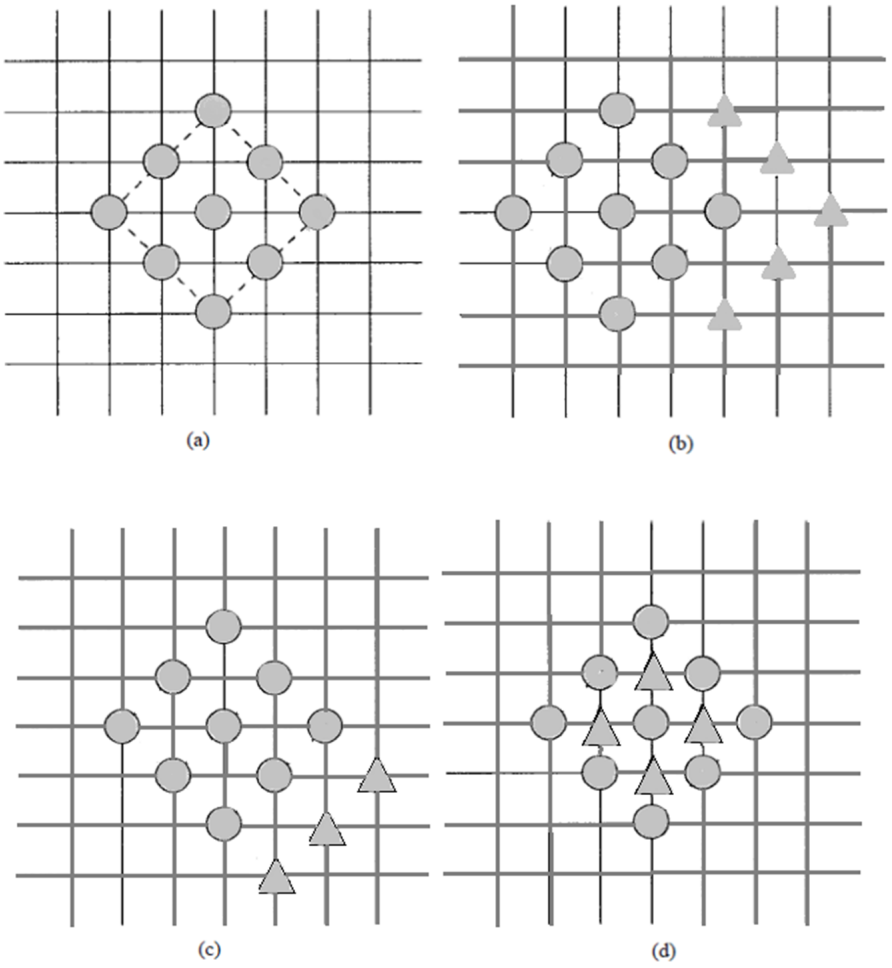

Figure 7: Simple diamond search path

3.2.2. Improved Diamond Search

[13][18] From the analysis of different sequences, it has been found that nearly 80% MVs are horizontal and vertical [16], caused by objects' movement or camera's tracking, panning or tilting, etc. It can therefore be concluded that most motion vectors contained in real-world

sequences are horizontal and vertical. Thus, authors in [13] suggest eliminating the points located at the face of the large diamond in the DS to speed up the computation. They call this new search pattern "Improved LDSP" (ILDSP), as shown in figure 8.

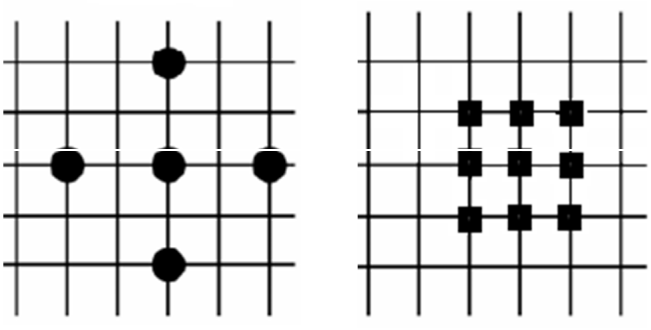

Figure 8: a) Improved Large Diamond Search Pattern
b) Small Square Search Pattern

Besides, to ensure the global optimum point being reached, the Improved Diamond Search (IDS) algorithm expand the original small diamond from five check points to nine and change its name from SDSP (small diamond search pattern) to "Small Square Search Pattern" (SSSP). IDS procedure can be summarized as follows:

Step1: The initial ILDSP is centered at the origin of the search window, and distortion criterion for five points including the origin is calculated. If the minimum SAD is found at the center position, go to Step 3; otherwise, go to Step 2.

Step2: The origin is repositioned to the minimum SAD point found in the previous search step, to form a new ILDSP. As the prior step, SAD values for five points are calculated. If the new minimum SAD point is obtained at the center position, go to Step 3; otherwise, recursively repeat this step.

Step3: Switch the search pattern from ILDSP to SSSP. The minimum SAD point found in this step is the final solution of the motion vector which points to the best matching block.

3.3. Cross search algorithm

One recent example of cross patterns [18][19] for motion estimation is Zero-MV biased cross-diamond search algorithm (ZCDS) [15]. Here, an adaptive dynamic threshold (*Thrs.*) is defined

based on the block motion content via a linear model. In this model the ratio between SAD of zero MV macro block and average minimum SAD is used. The ZCDS process consists of the following steps:

Step1 (*Thrs* computation): If the *SAD* of zero MV block is less than the threshold *Thrs*, then the search stops, otherwise, go to Step2 with a LCSP.

Step 2 (Large Cross Shape Pattern LCSP): Examine nine search points of the LCSP located at the center of search window. If the SAD of any candidate block is less than *Thrs*, the search stops; otherwise go to Step (3).

Step 3 (Half Diamond Searching): Two additional search points closest to the current minimum SAD in the central LCSP are checked. If the SAD value is less than *Thrs*, the search stops immediately. If the minimum SAD found in previous step located at the middle wing of the LCSP and the new minimum SAD found in this step still coincides with this point, go to Step (4).

Step 4 (Diamond Searching Pattern DSP): A DSP is formed by repositioning the center of DSP to the minimum SAD found in previous step. The DSP points are checked. If the *SAD* of any candidate is less than *Thrs*, the search stops immediately; If the new minimum SAD point is at the center of the newly formed DSP, then go to Step (5) for converging to the final solution; otherwise, this step is repeated.

Step 5 (Ending – SCSP Converging step): With the minimum SAD point in the previous step as the center, a SCSP is formed. The SCSP points are checked one-by-one. If the *SAD* of any candidate is less than *Thrs*, the search stops immediately; otherwise, identify the new minimum SAD point for the SCSP, which is the final motion vector.

3.4. Hexagon search pattern

[14][19]In the original hexagonal search (HEXBS) algorithm, two search patterns are involved. First the large hexagon search pattern in figure 9, consisting of six endpoints is formed to find the region in which the optimal motion vector is expected to be in. This coarse search continues based on a gradient scheme until the center point of the hexagon has the minimum SAD value.

After a coarse hexagonal search the some fine-resolution search looks into the small area inside the large hexagon, as shown in figure 9.

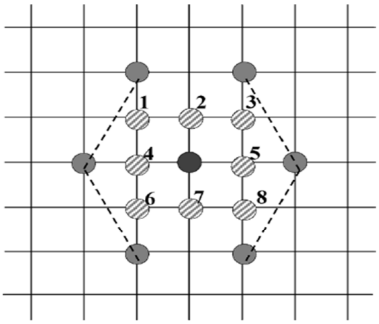

Figure 9: HEXBS initial search point [14]

This method considers grouping the search points in the six sides of the hexagon, resulting in six pairs of points, as shown in figure 10. For each group, a group distortion is defined by summing the distortions of all the points within the group. The area near to the group with the minimum group distortion is considered as the final solution for the search. Two or three search points are examined in the focused inner search, depending on the position of the group.

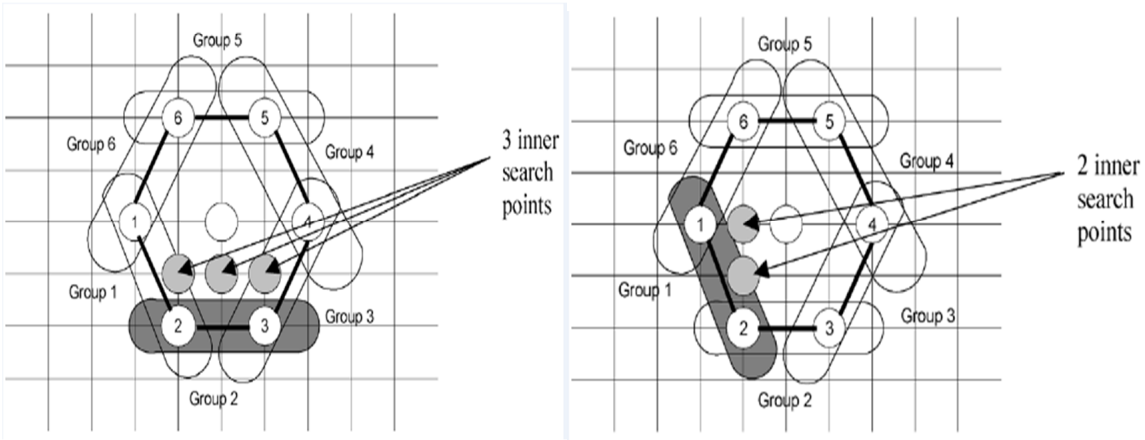

Figure 10: Inner search points in HEXBS [14]

Most of these fast motion estimation algorithms accomplish a speed up improvement over full search technique. However, this is mostly at the cost of PSNR and compression ratio.

CHAPTER4

Predictive search algorithms

To alleviate the computation burden in a video encoder while keeping the quality and compression ratio, Predictive Zonal search algorithms are proposed some solutions for lowering the computational intensity of ME, with a quality and compression performance closer, to conventional full search method. In these approaches, the idea is basically to predict the most likely initial sub-set of search points based on the temporal and spatial correlation between MV of current MB and previously coded MBs.

Afterwards, they measure the SAD parameter, for all initial points and set the verdict with minimum SAD as the origin of a local checking pattern, which could be diamond, square or hexagon shape pattern. Also they usually define some early termination conditions to avoid being trapped in local minima and enhance the speed up. In this chapter some examples of predictive search algorithms are listed.

One very common initial point is median MV of adjacent blocks in the current frame which has been used in many predictive algorithms [20]. In H.264/AVC standard, after finishing the ME process the encoder needs to calculate this spatial median MV for other modules any ways, hence using that in initial step of ME does not add any extra calculation to the process. Also, because this MB is located at the same location as the current MB, it is highly possible that their MV is very similar. Therefore, many proposed approaches suggest having it in the list of predicted MVs.

Also, most of predictive techniques take advantage of the fact that zero motion vector probability is considerably high and put that in their prediction set. [15] In gentle videos, on average, more than 70% of MVs can be located inside a 3x3 window around zero MV. Even in the case of high motion content videos, the possibility of zero motion is more than 25%. Thus, the second predicted initial MV is MV (0, 0).

4.1. Motion Vector Field Adaptive Search Techniques

The first predictive algorithm named *Motion Vector Field Adaptive Search Technique* (MVFAST) [21] suggests using MV of left; top; and top-right adjacent blocks in current frame, and zero motion vector (0,0) as initial predictors, combined with a two stage diamond search pattern and a fixed early-stopping criterion after examining the (0,0) predictor.

Then, another algorithm known as *Predictive Motion Vector Field Adaptive Search Technique* (PMVFAST) [22][23][24] introduced two initial predicted MVs, including: the median MV of three spatially adjacent blocks in current frame, and the MV of the collocated block in the reference frame. This algorithm used an adaptive early-stopping criterion, which were more reliable, sequence independent, and calculated based on correlations between adjacent blocks.

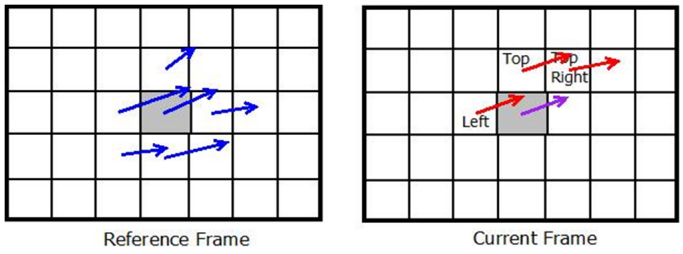

Figure 11: Correlated MBs in current and reference frames

The next proposed algorithm was *Advanced Predictive Diamond Zonal Search* (APDZS)[25][26], which modified search pattern of PMVFAST algorithm to multiple stage diamonds, and achieved higher PSNR than PMVFAST at an insignificant cost in speed up, but still better than MVFAST.

4.2. Un-Symmetrical Multi-Hexagon Search algorithms

Another example of predictive algorithms, *Un-Symmetrical Multi-Hexagon Search* (UMHEX) [27] has defined 13 initial search points in 2 sets. Five predictors in set1 including: the zero MV (0,0); median MV of the spatially adjacent blocks on the left, top, and top-right of the current block; MV of co-located block in reference frame; neighbouring reference frame prediction , obtained through multiple frames by scaling of the current block's motion vector in the reference frame; and upper layer prediction, which is a larger size block. For example, the 16 x 16 is used

as upper layer predictor for 16x8 and 8x16 blocks. Set 2 consists of four MVs around the spatial median prediction and four MVs around the zero MV (0, 0).

After testing all these predictors, the region with min cost becomes the center of an Un-symmetrical cross search with a horizontal search window size of W and vertical search range of W/2. Again, the motion vector with the minimum cost is chosen as the origin of a full search in a 2X2 area. There are a total of 25 search points in this step.

In the next step, first a 16-points hexagon search pattern is carried out as the basic search pattern followed by an uneven-multi-hexagon-grid, performed by extending the 16-points hexagon pattern with different scale factors and starting the search process from the inner hexagon to the outer one. The final step is an Extended-hexagon-based search including a hexagon and a diamond search pattern.

This algorithm also proposed floating point calculation-based early termination criterion in each step of the searching process. [28] Because of quite large number of predictors; complex searching steps; and a computationally intensive early-stop criterion this algorithm, as it will be shown in experimental result section, is not highly successful in speed up.

Authors in [29] proposed "*Simplified-Un-Symmetrical Multi-Hexagon Search*" (SUMHEX) algorithm, which only uses spatial median and upper layer predictors of UMHEX to reduce the memory space of MV prediction module. Combining with an adaptive search pattern for various levels of motion contents in different images, SUMHEX was able to improve UMHEX speed up, by 57%.

4.3. Enhanced Predictive Zonal Search (EPZS)

Enhanced Predictive Zonal Search (EPZS) [30][31], is the most famous predictive method and currently implemented in JVT, H.264/AVC standard software platform. EPZS considered initial set points introduced by previously developed algorithms, and added 5 more predictions to them. Four of new predictors come from MV of four adjacent blocks in up, right, down and left side of co-located MB in reference frame. And the other one is *accelerator* MV (figure 12), which is the differentially increased or decreased MV of co-located MB in reference frame and one frame

before that. The idea behind it is that a block MV may be following the acceleration instead of a constant velocity.

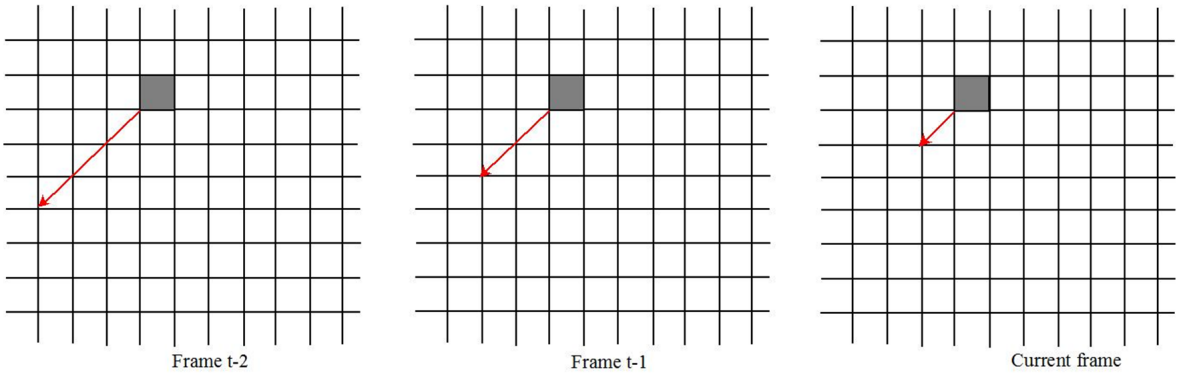

Figure 12: Acceleration predicting

EPZS's termination criterion, based on SAD value of adjacent MBs in current frame, was more efficient than all previous algorithms and could achieve a significant improvement in computational complexity. For early-stop criterion, EPZS essentially defines some constant threshold range for each valid block size in H.264 standard. In particular for 16x16-samples blocks in H.264 reference model software this range included Min-threshold of zero, Med-threshold of 8192 and Max-threshold of 48192.

[32]In the first step of EPZS procedure, SAD of median predictor is calculated and compared with an initial Stop-criterion which is equal to the defined Med-threshold. If it is smaller than this threshold, it stops further searching and takes median predictor as the final MV. Otherwise, at this point a new Stop-Criterion is calculated based on equations (4-1) to (4-4).

In the next step, SAD values of other spatial and temporal predictors are calculated. Then the minimum SAD predictor is compared with the new Stop-Criterion. If it is smaller, further searching is terminated and that minimal SAD predictor is taken as final MV, otherwise it starts a local diamond search around MB with minimum SAD, and it continues till the final MV is found.

$$\text{Min-SAD} = \text{Min}\,(\text{SAD}\,(A_{cur}), \text{SAD}\,(B_{cur}), \text{SAD}\,(C_{cur})) \quad (4\text{-}1)$$

$$\text{Step1-Criterion} = \text{Min}\,(\text{Min-SAD}, \text{Max-threshold}) \quad (4\text{-}2)$$

$$\text{Step2-Criterion} = \text{Max}\,(\text{Step2-Criterion}, \text{Med-threshold}) \quad (4\text{-}3)$$

$$\text{Stop-Criterion} = [9*(\text{Step2-Criterion}) + 2*\text{Med-threshold}]/8 \quad (4\text{-}4)$$

The author in [33] suggests that it is better to increase the early-stop criterion of EPZS, to reduce the searching predictor processing without loss of quality. This method modifies EPZS early-termination process by introducing a motion factor, based on the distortions difference of the three adjacent blocks in current frame, and adding this adaptive motion factor to the criterion. Then it replaces equation (4-3) by the following formula, (4-5).

$$\text{Step2-Criterion} = \text{Step1-Criterion} + \text{motion\_factor} * ((\text{Max\_SAD}) - (\text{Min\_SAD})) \quad (4\text{-}5)$$

Where, Max_SAD and Min_SAD are respectively the maximum and the minimum values of the three distortions of adjacent blocks.

The prediction set of all these Zonal algorithms seems to be the most principal feature and key to their performance. In the next section a *Dynamic Predictive Search algorithm* (DPSA) is proposed, which could be considered as an improvement of EPZS. The aim of this new approach is to introduce the most effective spatial and temporal correlated MBs for a simple but optimum initial sub-set that leads to higher performance (PSNR), lower computational complexity (search points), and lower bit rates compared with EPZS.

CHAPTER5

Proposed Algorithm

In this chapter we propose a Dynamic Predictive Search Algorithm (DPSA), which introduces the most effective spatially and temporally correlated MBs for a simple but optimum initial sub-set, combining with an early termination criterion that leads to higher quality (PSNR), lower computational complexity (search points), higher speed up, and lower bit rates compared to other existing predictive algorithms.

5.1. Prediction sub-set

The most prominent feature that results in high performance in zonal search algorithms is their set of predictors associated to spatial and temporal correlation between adjacent MBs in current and reference frames. These blocks are consisting of:

- Left; top; and top-right MBs in current frame, which in this document, henceforth, are called spatial MBs, since they are spatially correlated to the current MB, (Sptl_left_MB, Sptl_up_MB, and Sptl_upright_MB)
- Left; top; and top-right MBs in reference frame, which are called temporal MBs, since they are temporally correlated to their co-positioned blocks in the current frame (tmprl_left_MB, tmprl_up_MB, and tmprl_upright_MB). These are shown in figure 13.

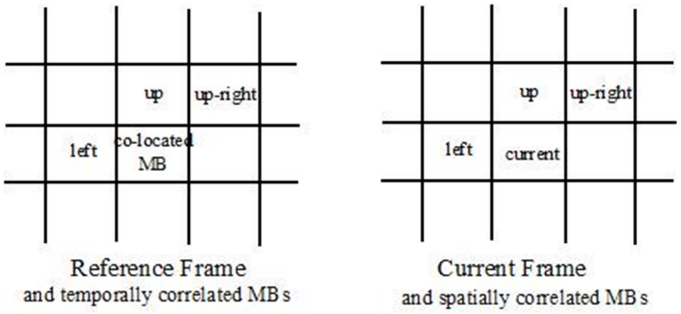

Figure 13: Spatial and temporal MBs

The first two predictors in DPSA are what previously have been used and proved to be efficient in other techniques, including spatial median MV of adjacent blocks in current frame; as well as zero motion vectors.

One closely correlated MB to current one is the co-located block in reference frame (Tmprl_MB). These two blocks are most probably associated to the same object and therefore have similar motion behaviour. One similarity is the direction and speed of their movement. Another aspect is dynamic correlation between them. For example, if MV of co-located MB (Tmprl_MB) differs from its neighbouring blocks in reference frame in terms of direction and velocity, then this difference might be repeated for MV of current MB in current frame, too.

It is possible that the object included Tmprl_MB has stopped or accelerated or even changed its direction from reference frame to current frame. To consider these possibilities, proposed DPSA algorithm suggests that if direction and velocity of tmprl_left_MB; tmprl_up_MB; tmprl_upright_MB in reference frame have changed in previously calculated MV of Sptl_left_MB; Sptl_up_MB; Sptl_upright_MB in current frame then we can assume that this alteration has happened between MV of current MB in current frame, and its co-located MB in reference frame, too.

However, it has been observed that not all neighbouring blocks have equal correlation to tmprl_MB. As a matter of fact, two top and left adjacent blocks appeared to be the highest correlated neighbours, while adding tmprl_upright_MB in search options, in most cases, only increases the number of search points without any improvement in quality or compression ratio. For that reason in proposed DPSA, only MVs of left and top MBs are considered.

At the moment of calculating MV for current macro block, motion vectors for left and top MBs are already calculated (because of raster scan order) (figure 14). A new predicted MV is created by finding the changes (Δ) between MVs of top and left adjacent blocks in reference frame (tmprl_left_MV; tmprl_up_MV respectively) with their co-positioned blocks Sptl_left_MV; Sptl_up_MV in current frame and then applying that on MV of Tmprl_MB, a new predicted MV is created. This predictor is referred to here as the spatial-temporal predictor (Sptl_tmprl_prdctor), since all temporal and spatial correlations of adjacent macro blocks are consider at the same time.

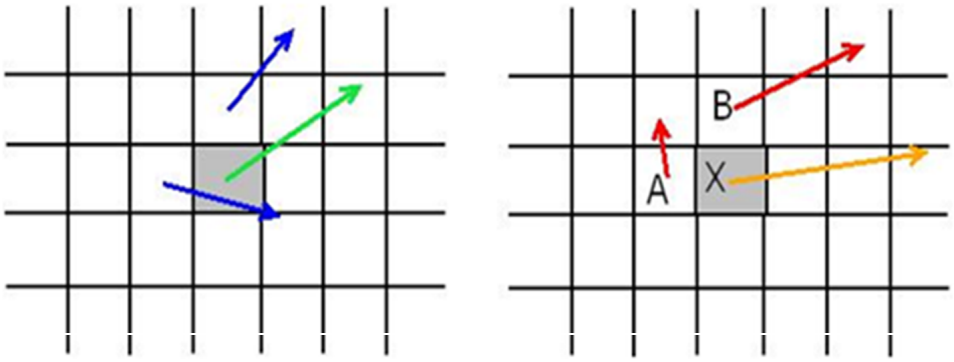

Figure 14: a) Reference frame b) Current frame

$$Avrg\_\Delta = \frac{(Sptl\_left\_MV)-(Tmprl\_left\_MV)}{2} + \frac{(Sptl\_up\_MV)-(Tmprl\_up\_MV)}{2} \quad (5\text{-}1)$$

$$Avrg\_\Delta = \frac{[left\_\Delta] + [up\_\Delta]}{2} \quad (5\text{-}2)$$

$$Sptl\_tmprl\_prdctor\ 1 = (Tmprl\_MB\_MV) + Avrg\_\Delta \quad (5\text{-}3)$$

To enhance the performance of initial set points, two other predictors are defined using $\Delta/2$. By adding and subtracting $\Delta/2$ to and from MV of co-located MB in reference frame, second and third spatial-temporal predictors (Sptl_tmprl_prdctor2 and Sptl_tmprl_prdctor3) are created. These predicted MVs cover the possibility that Tmprl_MB_MV changing speed or direction is not exactly as its neighbours. This way, the algorithm sweeps vaster area to get closer to the real MV. Consequently, it will result in higher performance with an insignificant increasing in search points, especially in complex scenes and motion activities.

$$Sptl\_tmprl\_prdctor2 = (Tmprl\_MB\_MV) + (Avrg\_\Delta)/2 \quad (5\text{-}4)$$

$$Sptl\_tmprl\_prdctor3 = (Tmprl\_MB\_MV) \_ (Avrg\_\Delta)/2 \quad (5\text{-}5)$$

To cover the spatial correlation between neighbouring MBs, many algorithms suggest using MV of three adjacent MBs, left; top; and top-right blocks in current frame as three initial search points.

Again, with the same logic as discussed, it can be assumed that blocks in left and top, are more redundant to current MB than top-right block. Consequently, Sptl_upright_MV can be

eliminated in the process. Now, two neighbouring blocks in each frame are remained, which are considered the most efficient MBs to predict the MV of Current_MB.

sptl_prdctor1 = (Sptl_left_MV) (5-6)

sptl_prdctor2= (Sptl_up_MV) (5-7)

As it is seen in equation (5-1) in Avrg_Δ definition Sptl_left_MV and Sptl_up_MV have been used. On the other hand, the redundancy between these two MBs and the current MB has been considered in defining sptl_prdctor1 and sptl_prdctor2. To minimize computation required for the prediction process only one of these adjacent blocks can be used in spatial prediction. In other words, spatial predictor can be defined as:

Up_sptl_prdctor = (Sptl_up_MV) (5-8)

Then, for spatial-temporal prediction there are two options:

Either Δ can be defined as it was in equation (5-1), or it can be only one of the neighbours, which would be left MB. Therefore:

$$Left\_\Delta = (Sptl\_left\_MV) - (Tmprl\_left\_MV) \quad (5\text{-}9)$$

Sptl_tmprl_prdctor 1= (Tmprl_MB_MV) + Left_Δ (5-10)

And two other predictors have to be obtained by:

Sptl_tmprl_prdctor2= (Tmprl_MB_MV) + (left_Δ)/2 (5-11)

Sptl_tmprl_prdctor3= (Tmprl_MB_MV) - (left_Δ)/2 (5-12)

This way, the proposed algorithm can be divided in two branches. One is called DPSA1, in which Avrg_Δ and (Avrg_Δ)/2 are used to define the initial sub-set of MVs. The second branch is named DPSA2, and uses Left_Δ and (left_Δ)/2 in the process.

Generally, after defining the predictors, the processor starts calculating SAD value for all of predicted regions. Then, the verdict with minimal SAD is set as the origin of a local search to find the most matching MB. In DPSA algorithm the local search pattern is chosen to be small diamond search due to its well-known high complexity-performance ratio. In each step minimum

SAD value is compared to an early stop criterion. If it satisfies the criterion the processor stops further searching and take that MV as the best description of current MB movement.

5.2. Early termination strategy

For early termination conditions, EPZS criteria are base of proposed DPSA, with adding one more condition. In DPSA algorithm, after testing all initial predicted MVs, if one of three spatial-temporal predictors using Δ and $\Delta/2$ has minimum SAD value then it can skip the local search step and take that predictor as the final MV of the current MB. This claim is based on the fact that, in the definition of these three initial predicted MVs, all possible correlations between adjacent blocks in current frame, and co-located MB, as well as its surrounding neighbours in reference frame have been considered.

In other words, if the minimum SAD of initial predictors is associated to one of the zero MV, median spatial MV, or Up_sptl_prdctor, and this minimum SAD value does not satisfy the calculated Stop-Criterion in equation (4-4), it starts small diamond search around that minimal SAD verdict to find the accurate MV. This process continues until either reaching a MB with smaller SAD value than the criterion, or the center of the search turns out to be the minimum SAD between the others, as in ordinary diamond search algorithm.

All these six predictors, along with the new early stop condition, have been tested and evaluated in JVT software platform, which will be elaborated further in the next section.

5.3. Algorithm block-diagram

DPSA1 algorithm process can be summarized in the following steps:

Step1: SAD value for median MV of three adjacent MBs in current frame including Left; Top; and Top-Right MBs and zero MV (0,0) are calculated.

Step2: MB pointed by Up_sptl_prdctor is found and its SAD value is calculated.

Step3: Avrg_Δ and (Avrg_Δ)/2 are measured with equation (5-2)

Step4: Three spatial-temporally predicted MV are found with equations (5-3), (5-4), and (5-5) and their SAD values are calculated.

Step5: The Min. SAD value between previous steps is found

Step6: If the Min. SAD is corresponding to MBs in step4, then it stops search. Otherwise, it goes to step7.

Step7: Based on equations (4-1) to (4-4), Stop-Criterion is measured.

Step8: If the Min. SAD value is less than this criterion, then it stops search. Otherwise, it goes to step9.

Step9: The verdict with Min. SAD becomes center of a small diamond search pattern. SAD value for four new points in small diamond pattern is calculated, and the Min. SAD is found.

Step10: If the Min. SAD is associated to the center of the search, then it stops search. Otherwise, it goes back to step9.

For DPSA2, steps 3 and 4 are as follow, and remaining steps are the same as DPSA1:

Step3: Left_Δ and (Left_Δ)/2 are measured with equation (5-9)

Step4: Three spatial-temporally predicted MV are found with equations (5-10), (5-11), and (5-12) and their SAD values are calculated.

All these steps are shown in the block-diagram of figure 15.

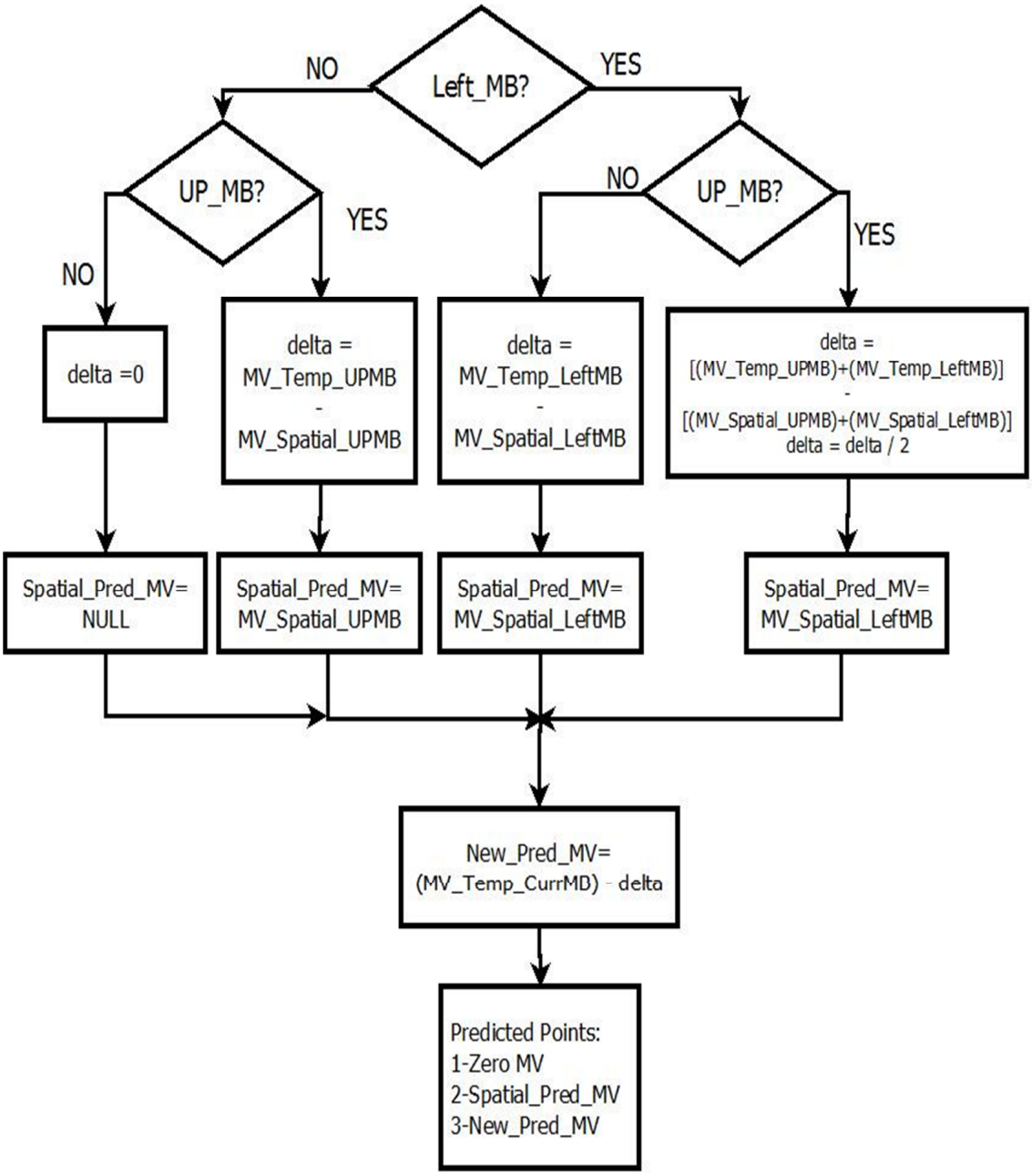

Figure 15: DPSA block-diagram

5.4. Example

To make the algorithm more clear one example of DPSA1 is shown in this part (figure 16).

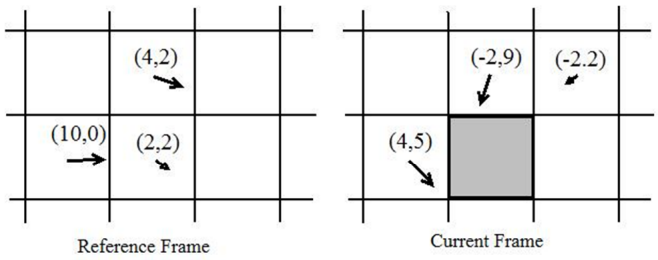

Figure 16: One example for DPSA1 algorithm

Median MV = median [(4,5); (-2,9); (-2,2)] = (-2,5)

Up_sptl_prdctor = (-2, 9)

Avrg_Δ = [(4, 5)-(10, 0)]/2 + [(-2, 9)-(4, 2)]/2 = (-6, 6)

(Avrg_Δ)/2 = (-3, 3)

Sptl_tmprl_prdctor1 = (2, 2) + (-6, 6) = (-4, 8)

Sptl_tmprl_prdctor2 = (2, 2) + (-3, 3) = (-1, 5)

Sptl_tmprl_prdctor3= (2, 2) - (-3, 3) = (5, -1)

In first step, six predictors are examined to find the Min SAD. Based on experimental results, in majority of cases, one of the three spatial-temporal predictors turn to be the min SAD, in that case the verdict with min SAD is final MV. In this example to clarify all other steps, it is assumed that the spatial predictor (-2, 9) is Min SAD verdict. Thus, in step 2, a small diamond search is formed around (-2, 9) vector. At this stage, vector (-2, 9) points to the minimal SAD parameter. Again, in step 3, another small diamond search pattern centered on (-2, 9) is formed. Since the center of this diamond is the minimum one, at this step, search is stopped and (-2, 9) is returned as final MV. These steps are shown in figure 17.

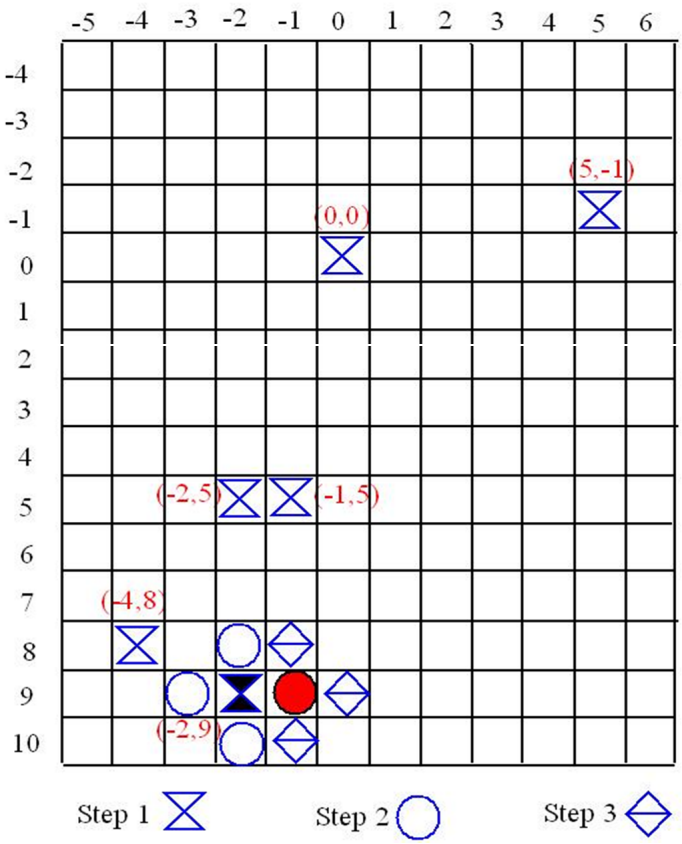

Figure 17: DPSA1 algorithm search steps for example1

CHAPTER6

Experimental results

Several experiments have been conducted to investigate performance of our proposed algorithm. Amongst, over 250 frames of Six 4:2:0 YUV, QCIF (176 x144), shown in figure 18, and five CIF (352x288) format test video sequences, shown in figure 19, are encoded with DPSA1 and DPSA2 along with 4 accepted algorithms in H.264 reference software, including FS; UMHEX; SUMHEX; and EPZS, all implemented in common test conditions. A 2.4GHZ dual-core CPU, with 4G RAM are used in this experiments.

Each time, ME process is carried on search window size of 32x32 pixels finding minimum SAD value as distortion criterion. The encoder is set to 30 frames per second for I and P frames, disabled skip mode and intra blocks in P frames, each frame divided into non-overlapping blocks of 16x16 samples H.264/144096-10 reference software version 18.1 from JVT [34][35] is chosen to be the test reference platform.

(a) *Akiyo* (b) *News* (c) *Coastguard*

(d) *car-phone* (e) *Mother& daughter* (f) *Foreman*

Figure 18: YUV video sequences; QCIF format (176x144)

(a)*BUS* (b) *Stefan* (c) *Water fall*

(d) *Tempete* (e) *Hall-monitor*

Figure 19: YUV video sequences; CIF format (352x288)

Software is developed to display four main criteria: total number of search points (sp); ME process time; data bit-rate; and PSNR of each encoded test sequence, to be evaluated. The results of these essential parameters are demonstrated and compared with 4 other algorithms, including EPZS and Full search techniques, in tables (2) to (12) and figures (20) to (36).

In all above tables, two different versions of DPSA have been implemented. In the sixth row, Left_Δ and (Left_Δ)/2 are calculated with equation (5-9). In seventh row Avrg_Δ and (Avrg_Δ)/2 are measured with equation (5-1).

Between 4 already existing algorithms applied in this study, Full search method reaches the highest quality encoded images with the cost of extensive computational complexity. UMHEX and SUMHEX decrease full search number of search points by 10 times with an insignificant cost of PSNR. EPZS manages to have the least number of search points, significantly, with acceptable range of PSNR. Hence, between these algorithm, EPZS is the most reasonable base to compare the newly proposed algorithm with. Though, EPZS does not have in positive effect on bit-rate.

As these experimental results indicate, DPSA not only improves the compression ratio, but decreases EPZS computational complexity too. Also, unlike most other fast ME algorithms,

DPSA not only does not reduce the quality of compressed videos, but also, boost the PSNR by 0.47dB.

Table 2: Performance comparison for "News"

Algorithms	SP/ frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR(kb/s)
EPZ	396.86	0.405	36.599	39.222	40.001	113.94
UHEX	2838.01	1.121	36.629	39.259	40.034	114.01
SUHEX	3185.94	1.001	36.623	39.234	40.001	113.80
Full	112141.00	18.427	36.629	39.250	40.060	113.55
DPSA2	358.38	0.371	36.615	39.253	40.093	107.73
DPSA1	361.16	0.374	36.620	39.249	40.047	105.31

Table 3: Performance comparison for "Akiyo"

Algorithms	SP/ frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR(kb/s)
EPZ	354.60	0.367	37.967	40.704	41.754	57.58
UHEX	1950.83	0.812	37.968	40.709	41.764	57.70
SUHEX	1998.23	0.716	37.968	40.708	41.764	57.69
Full	113180.65	17.410	37.963	40.692	41.768	57.84
DPSA2	302.31	0.343	37.969	40.713	41.774	57.83
DPSA1	305.46	0.342	37.983	40.715	41.761	57.16

Table 4: Performance comparison for "Car-phone"

Algorithms	SP/frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR (kb/s)
EPZS	582	1.594	37.182	39.997	40.477	265.52
UHEX	7085	6.696	37.172	39.963	40.478	262.85
SUHEX	8897	7.020	37.199	39.975	40.428	265.94
Full	410967	22.517	37.192	39.893	40.449	260.60
DPSA2	504	1.446	37.301	39.970	40.472	245.35
DPSA1	512	1.463	37.341	39.965	40.534	241.09

Table 5: Performance comparison for "Mother & daughter"

Algorithms	SP/frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR (kb/s)
EPZS	388	1.252	36.659	41.278	41.956	88.23
UHEX	4766	4.816	36.672	41.248	42.014	87.44
SUHEX	7001	5.960	36.682	41.272	42.009	87.85
Full	399621	178.000	36.701	41.249	41.968	87.13
DPSA2	334	1.137	36.836	41.325	42.111	83.68
DPSA1	338	1.133	36.919	41.353	42.072	81.01

Table 6: Performance comparison for "Foreman"

Algorithms	SP/frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR (kb/s)
EPZS	638	1.764	36.15	40.19	41.04	339.80
UHEX	9382	8.847	36.18	40.17	41.05	332.55
SUHEX	13310	10.734	36.16	40.18	41.07	341.88
Full	413380	208.077	36.16	40.23	41.05	327.84
DPSA2	571	1.605	36.44	40.21	41.12	289.01
DPSA1	582	1.648	36.49	40.17	41.11	279.49

Table 7: Performance comparison for "Hall-monitor"

Algorithms	SP/ frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR (kb/s)
EPZS	1208.396	4.269	37.96	39.72	41.64	363.58
UHEX	12777.78	14.684	37.96	39.72	41.64	364.61
SUHEX	12615.68	11.038	37.96	39.71	41.66	363.93
Full	1655893	819.869	37.96	39.72	41.65	363.09
DPSA2	1080.192	3.944	37.98	39.71	41.66	361.17
DPSA1	1083.192	3.967	37.99	39.72	41.66	360.52

Table 8: Performance comparison for "Coast-guard"

Algorithms	SP/ frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR(kb/s)
EPZ	486.46	0.462	34.274	43.256	43.995	394.83
UHEX	6607.41	2.439	34.283	43.307	43.998	392.95
SUHEX	10925.11	3.566	34.287	43.299	44.017	390.21
Full	113201.55	26.602	34.292	43.301	44.009	389.93
DPSA2	429.91	0.446	34.453	43.259	44.128	358.09
DPSA1	444.69	0.461	34.482	43.277	44.097	348.01

Table 9: Performance comparison for "Waterfall"

Algorithms	SP/ frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR (kb/s)
EPZS	1402	5.192	34.00	35.04	36.93	1186.96
UHEX	24477	25.254	34.05	35.11	36.98	1161.78
SUHEX	71623	58.133	34.05	35.09	36.97	1163.36
Full	1666408	1024.884	34.05	35.09	36.96	1161.34
DPSA2	1264	4.608	34.37	35.13	36.98	798.35
DPSA1	1289	4.879	34.41	35.13	36.97	736.03

Table 10: Performance comparison for “Tempete”

Algorithms	SP/ frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR (kb/s)
EPZS	2286	6.344	35.60	36.78	38.40	2745.44
UHEX	39434	38.196	35.62	36.79	38.42	2738.47
SUHEX	65470	56.336	35.61	36.78	38.43	2742.37
Full	1659335	1028.996	35.62	36.80	38.44	2733.29
DPSA2	2213	6.277	35.60	36.92	38.53	2061.98
DPSA1	2227	6.215	35.61	36.93	38.54	1988.52

Table 11: Performance comparison for “Stefan”

Algorithms	SP/ frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR(kb/s)
EPZ	2696.54	2.378	36.179	38.029	38.034	2454.71
UHEX	23420.49	9.225	36.190	38.049	38.033	2474.40
SUHEX	32947.06	9.938	36.176	38.024	38.030	2469.40
Full	452803.90	98.346	36.177	38.046	38.033	2450.41
DPSA2	2457.39	2.150	38.437	38.082	38.103	2035.02
DPSA1	2481.31	2.203	36.449	38.095	38.081	1980.05

Table 12: Performance comparison for “Bus”

Algorithms	SP/ frame	ME Time	Y-PSNR	U-PSNR	V-PSNR	BR(kb/s)
EPZ	2819.54	2.625	35.359	39.723	41.331	2617
UHEX	29063.56	12.867	35.384	39.691	41.269	2640.69
SUHEX	41113.46	14.430	35.368	39.738	41.302	2630.63
Full	452806.20	113.952	35.361	39.720	41.267	2602.51
DPSA2	2580.34	2.305	35.599	39.745	41.420	2107.95
DPSA1	2567.53	2.285	35.591	39.745	41.385	2077.09

6.1. PSNR

As it has been discussed in chapter1 in our tables PSNR is associated to three different values, Y; U and V samples .It has been observed that DPSA improves almost all components, considerably, compare to EPZS algorithm.

For instance, "Tempete" sequence is a good example of detailed images which is very colourful and the color content changes sharply between MBs. In this type of sequences, PSNR values associated to U- and V- samples are as important as Y components in the quality of encoded image. In figures 20, U-PSNR values are compared between DPSA1 and EPZS, frame by frame in CIF "Tempete" file, to demonstrate DPSA performance.

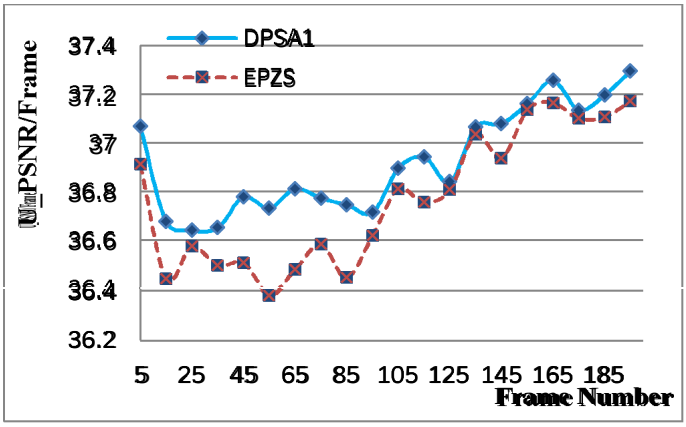

Figure 20: Frame by frame PSNR comparison for EPZS and DPSA1 on "Tempete"

However, since in most image samplings Y components have the higher resolution, and are more important, only this part of samples are compared for the rest of test videos.

Some tested sequences, such as "Hall-monitor"; "Mother & daughter"; "Akiyo"; and "News", are representing small to medium motion content videos. In simple and smaller MV images most ME algorithms are quite accurate, Still figure 21 shows better PSNR curve in DPSA, compare to EPZS.

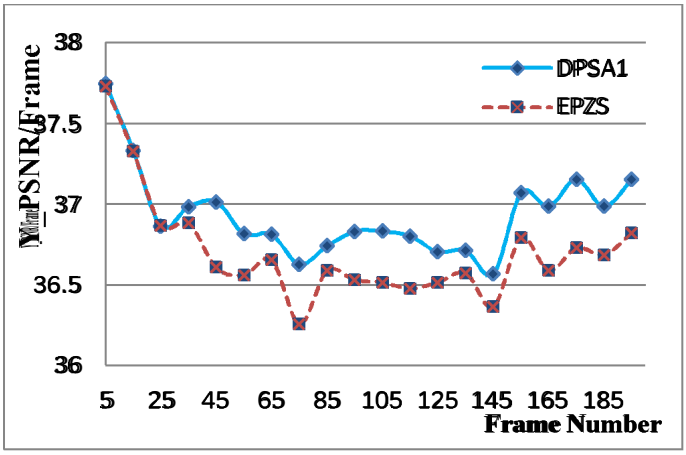

Figure 21: Frame by frame PSNR comparison for EPZS and DPSA1 on"Mother & daughter"

On the other hand, in medium to large motion content images, such as "Bus"; "Stefan"; "Waterfall"; and "Coast-guard", image quality is a critical key. In particular, "Waterfall"; "Bus"; and "Tempete" are three good examples of detailed images. In this category, usually blurring of features due to a crude de-blocking filter is very obvious. Hence, the accuracy of motion compensation module is essential to have good quality encoded images.

Comparing PSNR values earned in different algorithms in tables (8) to (12) indicates that almost in all cases DPSA has higher PSNR values over EPZS and other techniques. Also, frame by frame comparisons for PSNR of three detailed images are shown in figures (22) to (24).

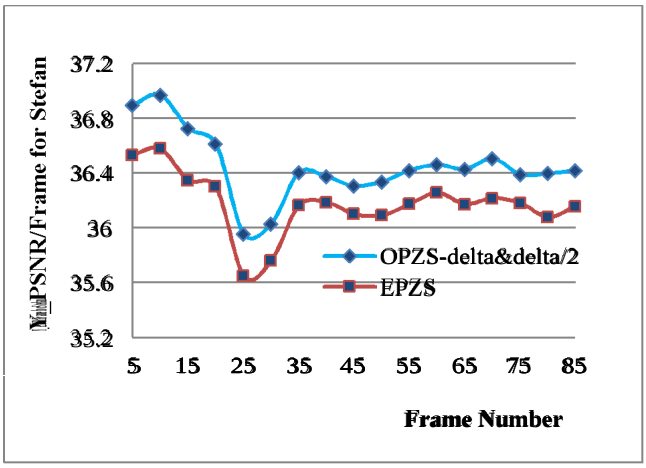

Figure 22: Frame by frame Y-PSNR comparison for "Stefan"

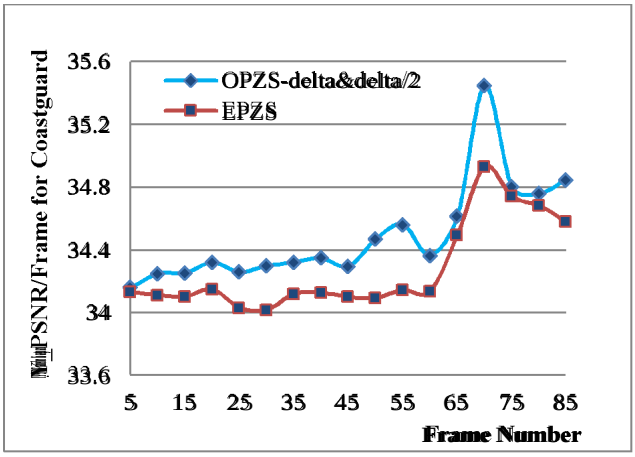

Figure 23: Frame by frame Y-PSNR comparison for *"Coast-guard"*

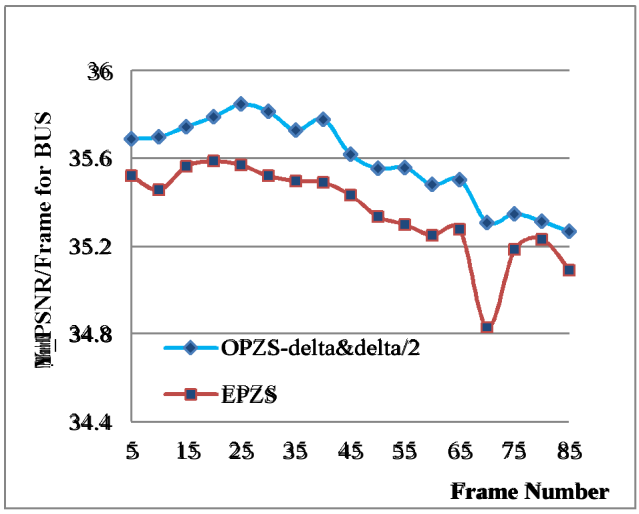

Figure 24: Frame by frame Y-PSNR comparison for *"BUS"*

6.2. Compression ratio

One of the most significant achievements of proposed DPSA algorithm is the high compression ratio. Somehow, this criterion represents the level of motion compensation accuracy. The more accurate MV estimation leads to further fidelity between encoded and original videos. As a result, the energy content of residual frames are much lower, and fewer data bits are required to be transmitted or stored, which means lower bit-rate or higher compression ratio.

Reviewing result-tables (2) to (12) shows that amongst 4 existing algorithms, EPZS has improved the computational complexity, but it does not have any effect on compression ratio. DPSA on the other hand, creates an outright reduction in bit-rates between all tested techniques.

Specifically, in sophisticated pictures, with lots of details, this factor can be clearly compared. For instance, Tables (8) to (12) are corresponding to five CIF streams with medium to large MVs. In this scale typically an encoder with constant parameters will produce higher bit-rates. These tables prove a significant improvement on data bit-rate over the state-of-the-art EPZS algorithm.

In figures 25 and 26 one video sequence of QCIF category and one from detailed CIF format category are chosen to illustrate frame by frame comparison on generated data-bits between DPSA1 and EPZS algorithms. It can be seen that in both scales, proposed DPSA has better curve in all individual frames.

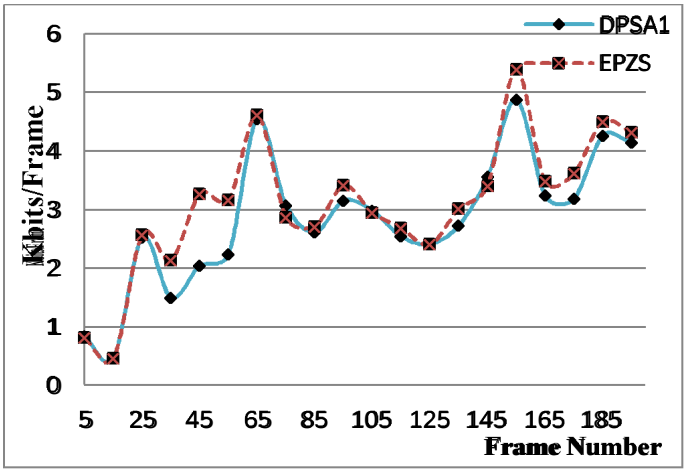

Figure 25: Frame by frame data Bits/frame comparison for EPZS and DPSA1 on "Mother & daughter"

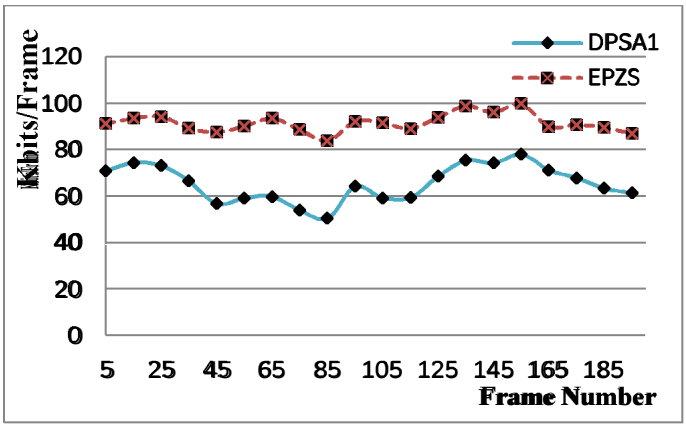

Figure 26: Frame by frame data Bits/frame comparison for EPZS and DPSA1 on "Tempete"

6.3. Computational complexity

6.3.1. Motion Estimation Process time

One way to evaluate an algorithm's computational complexity is to compare ME process time, or speed up. As third columns in tables (2) to (12) show, intensive FS algorithm has most definitely the longest process time. UHEX and SUHEX algorithms considering the fact that are categorized in fast ME techniques, are not that much helpful in terms of speed up. EPZS algorithm, on the other hand, managed to have considerably good improvement in speed up over other techniques. Still, proposed DPSA has improved EPZS motion estimation process time by 13%.

Having said that, because, especially in software implementation, algorithms are implemented on multi-purpose computers, and the processor might be involved in other tasks and interrupts, we believe ME process time is not the ultimate reliable parameter. Hence, we developed the software to display total number of search points for much more accurate evaluation. This parameter is discussed in next section.

6.3.2. Search points

The other criterion that indicates computational complexity of an algorithm is the number of search points. Between 4 already existing algorithms applied in this study, Full search method reaches the highest quality encoded images with the cost of extensive computational complexity. UMHEX and SUMHEX decrease full search number of search points by 10 times with an insignificant cost of PSNR. EPZS manages to have the least number of search points, significantly. As these experimental results indicate, DPSA not only improves the compression ratio, but even it decreases EPZS computational complexity too.

In figures (27) and (28) frame by frame number of search points per frame comparison is conducted for two QCIF vide sequences; "Mother & daughter" and "Coast-guard". These two files represent streams with small to medium MVs and details. In simple and smaller MV images, most ME algorithms are quite accurate, and produce few bits, but the point is which algorithm can perform faster and with less number of search points.

It can be seen that DPSA has the lowest number of search points in most frames.

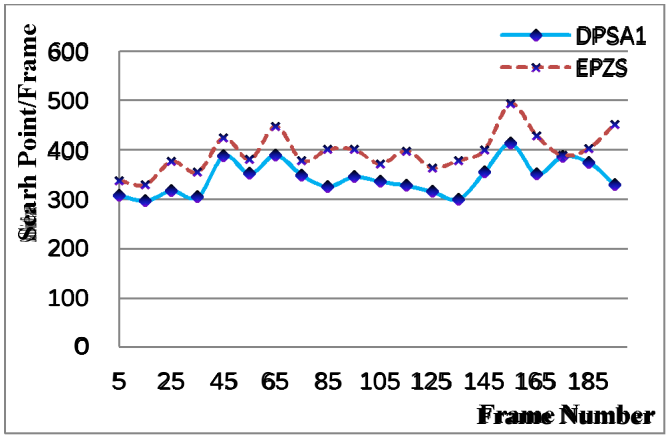

Figure 27: Frame by frame number of search point's comparison for EPZS and DPSA1 on "Mother & daughter"

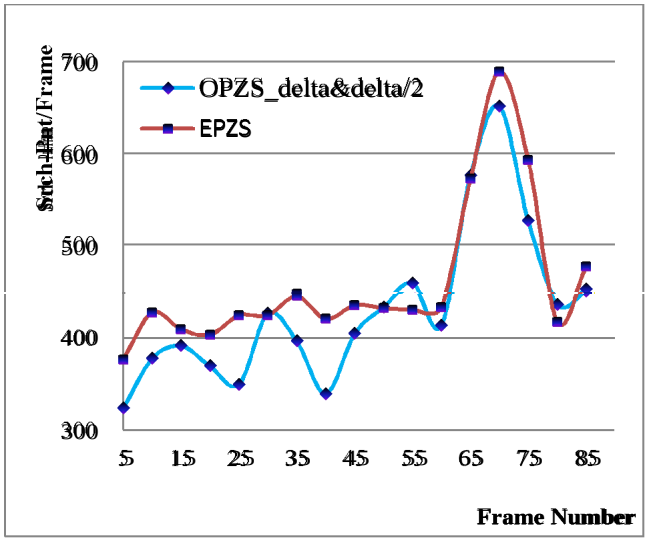

Figure 28: Frame by frame search point comparison for *"Coast-guard"*

The same experiment has been conducted on two CIF, medium to large MV videos: "Tempete" and "Bus". They also prove considerably good improvement on computational complexity over well-known EPZS algorithm, shown in figure (29) and (30).

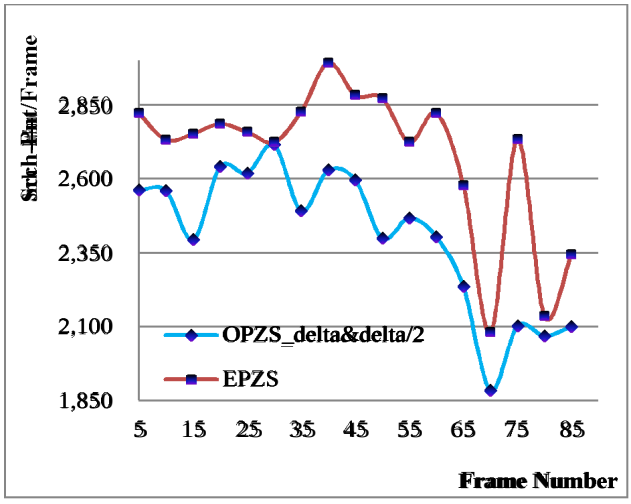

Figure 29: Frame by frame search point comparison for *"BUS"*

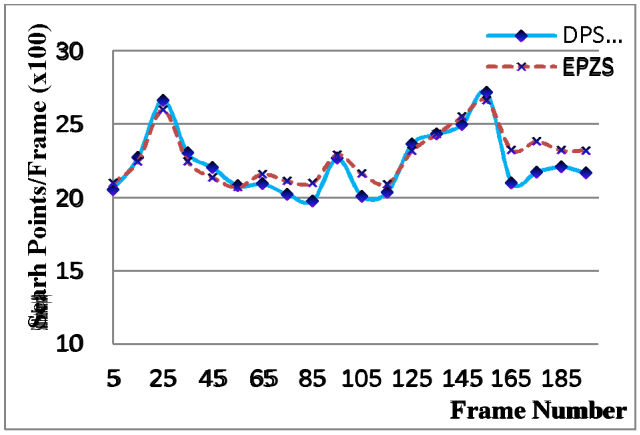

Figure 30: Frame by frame number of search point comparison on "Tempete"

Based on these tables (2) to (12), although EPZS has much lower number of search points than other three already existing approaches, DPSA1 and 2 have improved its result by about 14.7%.

6.4. Variable Quantization parameter

Another factor that has been studied in this research is step size of the quantization, carried out on the residual frames (quantization parameter). In figure (31) to (34), three main parameters: number of search points; PSNR; and data bit rate, are compared for a QCIF test video: "Car-phone", encoded by DPSA1 and EPZS with different quantization parameters (QP). As we can see for all three parameters, almost always, DPSA1 curves show better results than EPZS.

Figures (34) to (36) show the same results for a CIF format file: "Waterfall", as an example of detailed images. DPSA has considerably good performance in these cases, as well.

As evident in these graphs, *step size* (QP) is a critical parameter. If the step size is large, the range of quantised values is small and it leads to highly compressed data, during transmission. However, the re-quantised values in decoder are not a close approximation of the original ones and create lower PSNR values.

With smaller step size, the re-quantised values match the original signal more closely, which means higher PSNR, with the price of lower compression efficiency. QP of 28 seems to be the most optimum step size, and has been applied in all other experimental steps.

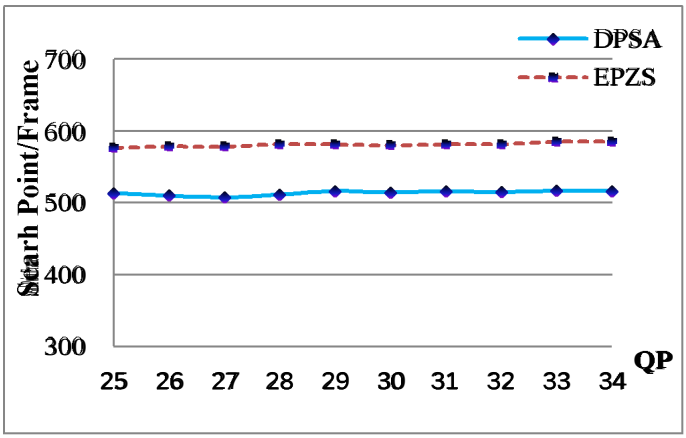

Figure 31: Number of search points comparison for EPZS and DPSA1with various quantization parameters on "Car-phone"

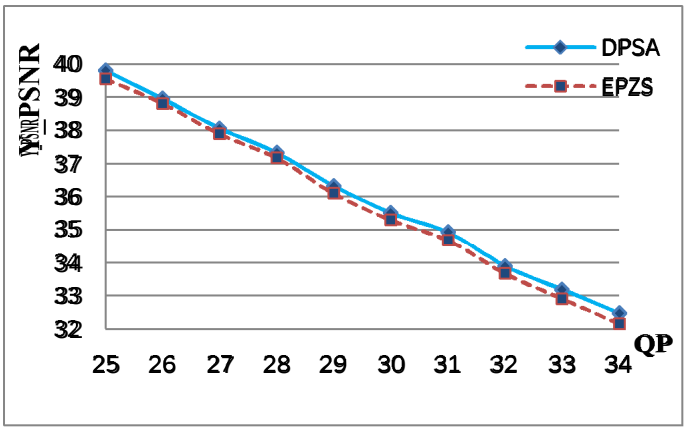

Figure 32: PSNR comparison for EPZS and DPSA1with various quantization parameters on "Car-phone"

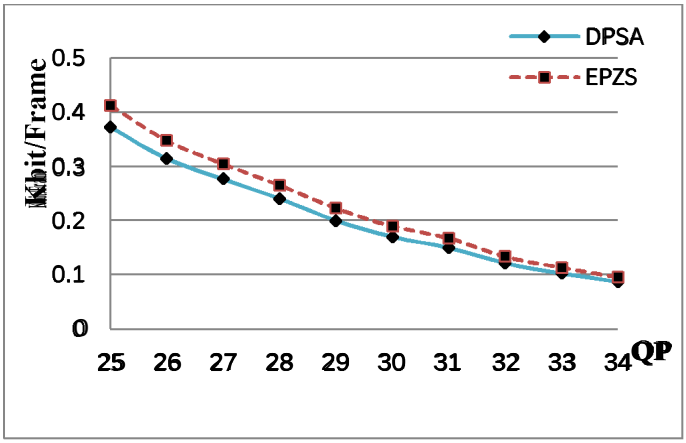

Figure 33: Bit-Rate comparison for EPZS and DPSA1with various quantization parameters on "Car-phone"

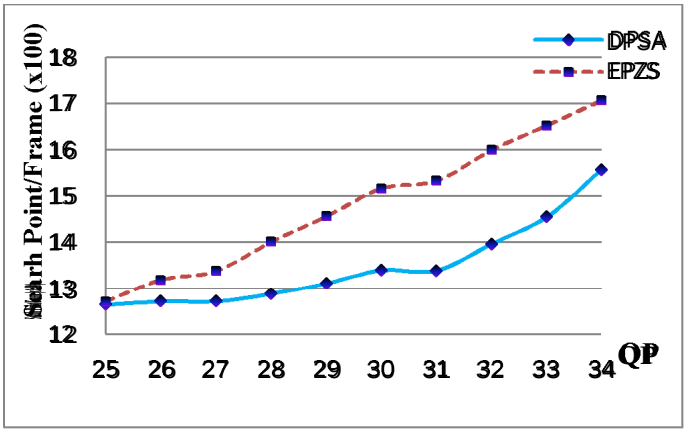

Figure 34: Number of search points comparison for EPZS and DPSA1with various quantization parameters on "Waterfall"

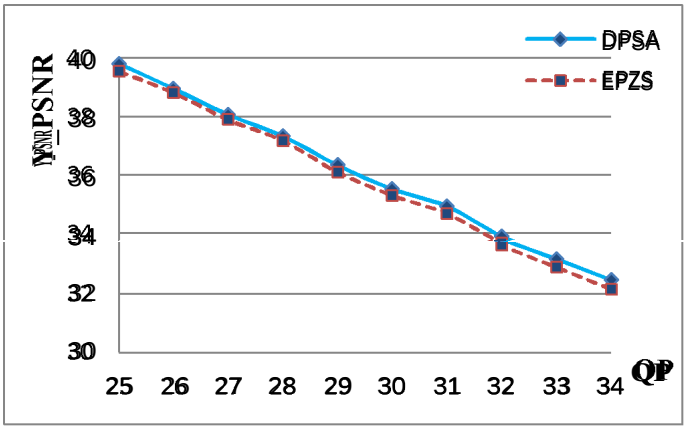

Figure 35: PSNR comparison for EPZS and DPSA1with various quantization parameters on "Waterfall"

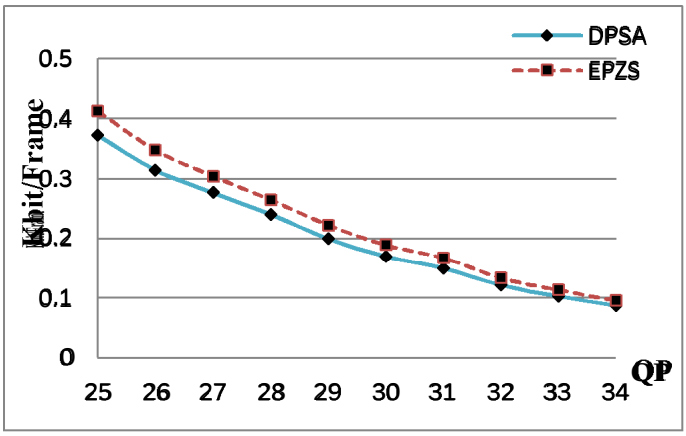

Figure 36: Bit-Rate comparison for EPZS and DPSA1with various quantization parameters on "Waterfall"

6.5. Comparison between DPSA1 and DPSA2

Comparing two proposed versions: DPSA1 and DPSA2, overall both methods improve all main criteria in EPZS. DPSA1, using Avrg_Δ and (Avrg_Δ)/2, in all test videos creates the least number of bit-rate with the highest PSNR values. However, the number of search points and ME process time in this approach is higher than DPSA2.

DPSA2, on the other hand, uses Left_Δ and (Left_Δ)/2. It manages to have the least number of search points, even up to 14.7% less than EPZS which is well-known for its computational complexity, and the highest speed up among all tested algorithms. Though, it does not increase the compression ratio, as much as DPSA1 does.

Therefore, although the differences between DPSA1 and DPSA2 performance are insignificant, they can be recommended in this way:

- For applications, with computational complexity constraint, such as real time encoders, in which, the computational cost must be low enough to ensure encoding of at least intended frames per second (*in this experiment: 30 fps*) DPSA2 is highly recommended.

- Instead, for applications like online video conferencing, in which the network's channel band-width is limited, or the capacity of storage is the main concern, DPSA1 with the lowest required bit-rate, insures the much needed compression ratio.

CHAPTER7

Conclusion

Fast ME algorithms are patterns which try to reduce the computational complexity of ME process time. Predictive zonal search algorithms are a category of block-based fast ME methods that promise to keep the encoded images quality and compression ratio besides lowering the computational complexity. In this thesis a new block-based dynamic predictive search algorithm (DPSA) is proposed, for video encoders.

7.1. Advantages

Encoding several test sequences with DPSA along with 4 other existing algorithms indicates that this technique clearly outperforms other methods in terms of PSNR; compression ratio; and computational complexity. All achieved improvements through DPSA1 and DPSA2 compare to EPZS algorithm for the 11 tested video sequences are listed in table 13.

Unlike most other fast ME algorithms, DPSA not only produces up to 38% lower bit-rate, but also, increases the PSNR parameter by 0.47dB. This confirms finding more accurate MVs and minimizing the energy content of residual images. This is a highly desirable feature in many applications, particularly in limited bit-rate and band-path channel networks and fixed capacity storage media like CDs and DVDs.

Moreover, based on experimental results, DPSA accomplishes up to 14.75% lower number of search points than state-of-the-art EPZS with up to 13% speed up. This accomplishment makes DPSA energy efficient for portable video processing in computation- or power-constrained applications.

One advantage of DPSA, over the other algorithms is simplicity. Unlike well-known EPZS that uses over 10 predictors, proposed DPSA takes advantage of the most optimum correlated MBs for prediction, and only with six initial search points, manages to achieve the highest

performance and the shortest process time. This feature makes it easier to be implemented on both hardware and software platforms.

Another feature of DPSA is its scalability for various levels of motion contents, small to medium to large. It also has considerably good performance over different sampling resolutions, as we have presented experimental results for both QCIF and CIF movie streams in this paper. This makes DPSA generally beneficial for different applications from offline film processing to online video-conferencing.

7.2. Future works

Experimental measurements prove that DPSA improves PSNR values as a factor of quality, over EPZS algorithm, on almost all encoded videos. Watching encoded frames in single colour areas of these images, it is even visually distinctive that DPSA creates smoother frames. However, in some colourful images such as waterfall, with high frequency samples, DPSA results in some kind of washed out colours.

One suggestion on future works in this area might be finding a solution to resolve this issue in order to improve the sharpness of colours in encoded images.

Table 13: Performance improvement in DPSA1 and DPSA2 over EPZS

Sequence name	DPSA algrthm	PSNR improvement (dB)	Bit-Rate improvement (%)	Search point improvement (%)	ME process time (%)
Car-phone	DPSA1	0.16	9.20	12.03	8.2
	DPSA2	0.12	7.60	13.41	9.3
Mother & Daughter	DPSA1	0.26	8.18	12.91	9.5
	DPSA2	0.177	5.16	14.05	9.2
Foreman	DPSA1	0.345	17.75	8.77	6.6
	DPSA2	0.29	14.95	10.42	9
Waterfall	DPSA1	0.410	37.99	8.1	6
	DPSA2	0.374	32.74	9.85	11.25
Tempete	DPSA1	0.1	27.57	2.58	2
	DPSA2	0.1	24.89	3.17	1
Hall monitor	DPSA1	0.031	0.84	10.36	0.3
	DPSA2	0.017	0.66	10.61	0.3
Stefan	DPSA1	0.270	19.34	7.98	7.4
	DPSA2	0.258	17.10	8.87	9.6
Bus	DPSA1	0.232	20.63	8.94	13.0
	DPSA2	0.240	19.45	8.48	12.2
Coast-guard	DPSA1	0.208	11.86	8.59	0.2
	DPSA2	0.179	9.31	11.62	3.5
News	DPSA1	0.021	7.57	9.00	7.7
	DPSA2	0.016	5.45	9.70	8.4
Akiyo	DPSA1	0.016	0.73	13.86	6.8
	DPSA2	0.002	-0.03	14.75	6.5

PUBLICATIONS

1. Abdoli, B; Sedaghat, R, "A Dynamic Predictive Fast Motion Estimation Technique", IEEE Transaction on multimedia, submitted, 2012

2. Abdoli, B; Sedaghat, R, "Optimized Predictive Zonal Search (OPZS) for Block-based Motion Estimation", International journal of Signal Processing: Image communication, Elsevier, submitted, 2012

BIBLIOGRAPHY

[1] Kneip, J.; Robert Bosch GmbH; Hildesheim Bauer, S. ; Vollmer, J. ; Schmale, B. ; Kuhn, P. ; Reissmann, M. ,"*The MPEG-4 video coding standard – a VLSI point of view*", in IEEE conference on signal processing systems 98, pp.43-52, October 1998

[2] Wiegand, Thomas; Sullivan, Gary J.; Bjontegaard, Gisle; Luthra, Ajay, *"Overview of the H.264 /AVC Video Coding Standard"*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, NO. 7, pp. 560-575, JUL 2003.

[3] Ahirwal,B.; Mechatron. Pune Khadtare, M.; Mehta, R., "*FPGA based system for Color Space Transformation RGB to YIQ and YCbCr*", in IEEE international conference on Intelligent and advanced systems, pp. 1345-1349, November 2007

[4] Iain E.G. Richardson, "H.264 and MPEG-4 video compression", (Book), 2003

[5] Xiao-Fan Feng, Daly, S., "*Vision-Based Strategy to Reduce the Perceived Color Mis-registration of Image-Capturing Devices*", PROCEEDINGS OF THE IEEE, VOL. 90, NO. 1, pp. 18-27, January 2002

[6] Kalva, Hari, *"The H.264 Video Coding Standard"*, in IEEE Multimedia, vol.13, NO.4, pp. 86-90, October 2006

[7] Jong-Nam Kim ; Sung-Cheal Byun ; Yong-Hoon Kim ; Byung-Ha Ahn, "*Fast Full Search Motion Estimation Algorithm Using Early Detection of Impossible Candidate Vectors*", IEEE Transactions on Signal processing, vol. 50, NO.9, pp. 2355 – 2365, September 2002

[8] Jing, Xuan; Chau, Lap-Pui, *"An Efficient Three-Step Search Algorithm for Block Motion Estimation"*, IEEE Transaction on Multimedia, vol. 6, NO.3, pp. 435-438, June 2004

[9] Li, Reoxiang; Zeng, Bing; Liou, M.L.; *"A new three-step search algorithm for block motion estimation"*, IEEE Transactions on Circuits and Systems for Video Technology, vol.4, NO.4, pp. 438-442, Aug1994.

[10] Chi-Wai Lam ; Lai-Man Po ; Chun Ho Cheun "*A Novel Kite-Cross-Diamond Search Algorithm For Fast Video Coding and video conferencing Applications*", in IEEE international conference on Acoustics, Speech, and Signal Processing, vol.3, pp. 365-368, May 2004

[11] Liang Yaling ; Liu Jing; Du Minghui "*A Cross Octagonal search algorithm for fast block motion estimation*", international symposium on Intelligent Signal Processing and Communication Systems 2005, pp. 357 – 360, December 2005

[12] Cheung, Chun-Ho; Po, Lai-Man, *"A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation"*, IEEE Transactions on Circuits and Systems for Video Technology, vol.12, NO.12, pp. 1168-1177, Dec 2002

[13] Improved Diamond Search Algorithm for H.264/AVC Video Coding Standard

[14] Zhu, Ce; Lin, Xiao; Chau, Lappui; Po, Lai-Man, *"Enhanced hexagonal search for fast block motion estimation"*, IEEE Transactions on Circuits and Systems for Video Technology, vol.14, NO.10, pp. 1210-1214, October 2004

[15] X. Yi and N. Ling, "Zero motion vector-biased cross diamond search for rapid motion estimation", in Image and Video Communications and Processing, San Jose, SPIE vol. 5685, pp. 995-1006, Jan 2005

[16] Zhiru Shi, W.A.C. Fernando and D.V.S.X. De Silva "*A motion estimation algorithm based on predictive intensive direction search for H.264/AVC*", in IEEE International Conference on Multimedia and Expo (ICME), pp. 667-672, September 2010

[17] Jo Yew Tham ; Ranganath, S. ; Ranganath, M. ; Kassim, A.A., "*A Novel unrestricted center-biased diamond search algorithm for block ME*", IEEE Transactions on Circuits and Systems for Video Technology, vol.8, NO.4, pp. 369 - 377, Aug 1998

[18] Tourapis, A.M. ; Au, O.C. ; Liou, M.L. ; Shen, G. ; Ahmad, I. "*Optimizing the MPEG-4 Encoder - Advanced Diamond Zonal Search*", IEEE international symposium on Circuits and Systems 2000, vol.3, pp. 674 - 677, 2000

[19] Li Hong-ye; Liu Ming-jun, "*Cross-Hexagon-based Motion Estimation Algorithm Using Motion Vector Adaptive Search Technique*", International Conference on wireless communication and signal processing, pp. 1-4, November 2009

[20] Ahmed, Zaynab ; Hussain, Abir Jaafar ; Al-Jumeily, Dhiya, " *Mean Predictive Block Matching (MPBM) for fast Block-Matching motion estimation*", in 3rd European Workshop on Visual Information Processing (EUVIP), pp. 67-72, October 2011

[21] Hosur, Prabhudev I.; Ma, Kai-Kuang, *"Motion Vector Field Adaptive Fast Motion Estimation"* Second International Conference on Information, Communications and Signal Processing (ICICS 1999), Singapore, Dec1999.

[22] Tourapis, Alexis Michael; Au, Oscar C.; Liou, Ming L., *"Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) - Enhancing Block Based Motion Estimation,"* in proceedings of Visual Communications and Image Processing 2001(VCIP-2001), pp.883-892, San Jose, CA, January 2001.

[23] Lihui Yang "*Research on the Motion Estimation Algorithm in AVS*", second international conference on networking and digital society, vol. 2, pp. 625 – 628, May 2010

[24] Hoi-Ming Wong, "*Enhanced predictive motion vector field adaptive search technique (E-PMVFAST)– based on future MV prediction*", IEEE International Conference on multimedia and Expo, July 2005

[25] Tourapis, Alexis Michael; Au, Oscar C.; Liou, Ming L.; Shen, Guobin; Ahmad, Ishfaq, *"Optimizing the MPEG-4 Encoder - Advanced Diamond Zonal Search"*, IEEE International Symposium on Circuits and Systems, Geneva, Switzerland , vol.3, pp. 674 – 677, August 2002

[26] Tourapis, A.M.; Au, O.C.; Liou, M.L. "*New Results on Zonal Based Motion Estimation Algorithms -Advanced Predictive Diamond Zonal Search*", IEEE International conference on Circuits and Systems, vol.5, pp. 183 - 186, 2001

[27] Lifen, Xie; Chunqing, Huang; Bihui, Chen, "*UMHexagonS Search Algorithm for Fast Motion Estimation*", in 3rd Computer Research and Development International Conference, pp. 483-487, May 2011

[28] Chou, Lei-Chun; Ye, Cheng-Da; Liu, Yuan-Chen; Jhao, Bin-Cheng, "*Fast Predictive Search Algorithm for Video Motion Estimation*", in 14th International Conference of Image Analysis and Processing, pp. 399 – 406, October 2007

[29] Yoon, Hyosun; Kim, Hyesuk; Kim, Miyoung; Nga, Lai; Lee, Gueesang, *"Hierarchical Integer pixel and Adaptive Fractional pixel Motion Estimation"*, IEEE 8th International Conference on Computer and Information Technology Workshops , Sydney, QLD, pp. 391 – 395, July 2008

[30] Marcelino S., Faria S., Assuncao P., Moiron S., Ghanbari M., *"Efficient MV Prediction for Zonal Search In Video Transcoding"*, in IEEE International Workshop on Multimedia Signal Processing (MMSP), pp. 228 – 232, Dec 2010

[31] Tourapis, Alexis Michael; Au, Oscar C.; Liou, Ming L., *"Highly efficient predictive zonal algorithms for fast block-matching motion estimation"* IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, NO.10, pp. 934-947, October 2002

[32] Tourapis, Alexis Michael, "*Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation*", in proceedings of Visual Communications and Image Processing 2002 (VCIP-2002), pp. 1069-79, San Jose, CA, Jan 2002

[33] Ezzedine, Tahar, "*Enhanced Adaptive Early Termination for Enhanced Predictive Zonal Search Algorithm in motion estimation*", in International Journal of Computer Science and Network Security (IJCSNS), vol. 8 NO. 6, pp. 236 -240, June 2008

[34] Tourapis, Alexis Michael; Sühring, Karsten; Sullivan, Gary, *"H.264/14496-10 AVC Reference Software Manual"*, Joint Vide Team (JVT) of ISO iIEC and ITU-T VCEG, Input document to JVT, June-July2009

[35] Chen, Zhibo; He, Yun, "*Fast Integer and Fractional Pel Motion Estimation*", Joint Vide *Team (JVT) of ISO iIEC and ITU-T VCEG, Input document to JVT, 5th Meeting, Geneva, Switzerland, October 2002.*

Nomenclature

DPSA	Dynamic Predictive Search Algorithm
EPZS	Enhanced Predictive Zonal Search
PSNR	Peak Signal to Noise Ratio
ME	Motion Estimation
MC	Motion Compensation
SAD	Sum Of Absolute Difference
SDSP	Small Diamond Search Pattern
LDSP	Large Diamond Search Pattern
MV	Motion Vector
MB	Macro Block
SP	Number of Search Points
BR	Data Bit Rate
QP	Quantization Parameter