جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

تحسين طريقة البحث مره واحده بالأعتماد على الأرتباط بين المقاطع

رساله
مقدمه الى كلية العلوم في جامعة النهرين كجزء من
متطلبات نيل درجة الماجستير في علوم الحاسبات

من قبل

راوية فائق محمود
(بكالوريوس جامعة النهرين ٢٠٠٤)

أشراف

د. لؤي أدور جورج

نيسان ٢٠٠٧ ربيع الأول ١٤٢٨

Chapter One
General Introduction

1.1 Introduction

Image compression addresses the problem of reducing the amount of data required to represent a digital image. The underlying basis of the reduction process is the removal of redundant data. From a mathematical point of view some transforms converts a 2D-pixel array into a statistically uncorrected data set. The transformation is applied prior to storage or transmission of the image. At some time later, the compressed image is decompressed to reconstruct the original image or an approximation to it [Gon00].

In digital image compression, three basic data redundancies can be identified and exploited. Image compression methods try to eliminate some of these redundancies to produce more compact code that preserve the essential information contained in the image, these data redundancies are [Umb98]:

1. **Coding Redundancy:** It occurs when the data used to represent the image are not utilized in an optimal manner. That is, it occurs when the gray levels of an image are coded in a way that uses more codes than absolutely necessary to represent each gray level. For example, if an 8-bit/pixel image, which allows 256 different intensity levels, is used to represent a 16-color image, actually only 4-bit/pixel is needed to represent the image. In general, coding redundancy is perfect when the

codes assigned to the set of gray levels have not been selected to take the full advantage of the probabilities of gray levels.

2. **Inter-Pixel Redundancy:** It occurs because the adjacent pixels tend to be highly correlated. This is a result of the fact that in most images the brightness levels do not change rapidly, but change gradually, so that adjacent pixels values tend to be relatively close to each other in value (for video, or motion images), this concept can be extended to include interframe redundancy (i.e. redundancy between frames of image data).
3. **Psycho-Visual Redundancy:** refers to the fact that some information is more important to the human visual system (HVS) than other types of information. For example, the HVS can only perceive spatial frequencies below about 50 cycles per degree, any higher frequency information is of little interest to us.
4. **Temporal Redundancy:** exits due to the similarity between the sequential neighboring frames, this type of redundancy exists in video and removed by using motion estimation method [Nic03].

1.2 Standard Image Compression Methods

Standardization of still images and video compression techniques has become a high priority issue, because only a standard can reduce the high cost and resolve the critical problem of interoperability of equipment's from different manufactures [Kad02].

In the following a summary of the most commonly known compression standards is presented.

JPEG

The JPEG format is a standard developed by Joint Photographic Experts Group for compressing continuous-tone still picture (e.g., photographs), the group worked under the joint auspices of International Television Union

(ITU), International standard Organization (ISO), and International Electro technical commission (IEC). JPEG had worked toward establishing the first international digital image compression standard for continuous-tone still image (both grayscale and color) [Kad02].

JPEG 2000

The JPEG 2000 is the next generation of still image compression standard. The main advantages of JPEG 2000, compared to JPEG, is in its better compression ratio (high compression rates), and its progressive compression output [Lau02].

MPEG

MPEG (Moving Picture Experts Group) is developed by an ISO/IEC working group for developing international standards for compression, decompression, and representation of moving pictures and audio [Kad02].

MPEG is used for monitoring applications where a stream of high-quality video and audio are needed, while limiting the amount of bandwidth used, relative to the quality level [Lau02].

MPEG-1

The first finalized international standard was MPEG-1 (International Standard). Its goal was to produce video recorder-quality output (352x240) using a bit rate of 1.2 Mbps. The uncompressed video alone can run to 472 Mbps, MPEG getting it down to 1.2 Mbps [Kad02]. MPEG-1 is basically a standard for storing and playing video on a single computer at low bit-rates (that is, requiring low transfer capacity and consequently, lower bandwidth) [Lau02].

MPEG-2

MPEG-2 (International Standard) was originally designed for compressing broadcast quality video into 4 to 6 Mbps. Later, MPEG-2 was expanded to support higher resolutions, including High-Definition Television (HDTV) [Kad02].

MJPEG 2000

Motion JPEG 2000 (ISO/IEC) as with JPEG, can also be used to represent a video sequence as a series of still JPEG 2000 images. At high compression rates, the image quality is better preserved with MJPEG2000 than using JPEG 2000 [Lau02].

1.3 Videoconferencing Compression Standards

In the past years, a number of compression standards have emerged and still this number is being developed. A growing number of standards were developed because of enhanced processing power, dedicated hardware, new compression technique [Aal96].

In the following a summary of the most commonly known video compression standards is presented.

H.261

H.261 was designed for videoconferencing and video-telephone applications over ISDN telephone lines. It uses a fairly old and simplified MPEG-1 technique, with a focus on bandwidth consumption over image quality. H.261 is not a standard, but a recommendation by the International Television Union (ITU) [Lau02].

H.263

The H.263 video coding standards is specifically designed for very low bit rate applications such as video conferencing. Its technical content was completed in 1995 and the standard was approved in early 1996. It based on the H.261 standard with several added features (like, unrestricted motion vectors, syntax-based arithmetic coding, and advanced prediction) [Shi00].

H.263 Version 2 (H.263+), (H.263++) and (H.26L)

H.263+ was approved in January 1998 by ITU-T. H.263+ includes a number of new optional features based on the H.263. These new optional features were added to provide improved coding efficiency, and to offer a flexible video format and scalability. H.263++ is the extension of H.263 and is currently scheduled to be completed in the year 2000. H.26L is a long-term project to seek more-efficient video coding algorithms that will be much better than the current H.261 and H.263 standards. H.261 and H.263 standards include several aspects (like, higher coding efficiency, more functionality, and low complexity permitting software implementation). H.26L addresses very low-bit-rate, real time, and end-to-end-delay applications [Shi00].

1.4 Wavelet Transform

Wavelet transform provides a multi-scale representation of images and video in the space-frequency domain. Aside from the energy compaction and decorrelation properties that facilitate compression, the major advantage of the wavelet transform is its inherent scalability [Xio04].

Discrete Wavelet Transform (DWT) is an efficient and useful tool for signal and image processing applications and it was adopted in many emerging standards, starting with the new compression standard JPEG2000. This growing "success" is due to the achievements reached in the field of mathematics, to its multiresolution processing capabilities, and also to the wide range of filters that can be provided. These features allow DWT to be tailored to suit a wide range of applications [Xia01].

1.5 Motion Estimation (ME) and Motion Compensation (MC) for Video Compression

Motion estimation is the process of estimating the motion of moving objects in a video scene. It is accomplished by determining the motion by which an object moves from one frame to the next. A given frame in video sequence may be predicted from its previous frame by displacing all the moving objects in the previous frame by the estimated motion, this is called motion compensation. The motion compensated frame difference is then coded and transmitted [Gal03]. Motion compensation is an important function of most video coding schemes, because it allows taking into account the high degree of correlation usually existing between consecutive frames [Cap99].

Motion estimation/compensation is the standard approach to reduce the temporal redundancy during the process of coding real-world video sequences. Due to its simplicity, the block-matching algorithm (BMA) has been widely used in motion estimation [Mir02].

In the past two decades, extensive researches were conducted to develop the motion estimation techniques. Many motion estimation techniques like pel-recursive techniques, gradient techniques, frequency-domain techniques and block based matching techniques have evolved. Among these motion estimation techniques, block-based matching has been widely adopted by international standards such as the H.26111, H.26312, MPEG-113 and MPEG-214 due to its effectiveness and robustness.

Most of the proposed motion estimation algorithms tend to be biased towards achieving speed by sacrificing visual quality. A lot of research projects were motivated to find a good trade-off between speed and quality, that is, to increase the speed as much as is consistent with good visual results. A focus on the block-based motion estimation technique was given, since it is widely adopted in international standards [Chu01].

1.6 Block Based Motion Estimation

Block-based motion estimation is fast, easy-to-implement, and provides reasonable results across a wide range of sequences. Also, it has the advantage of combining well with block-based techniques for coding the residuals [Ser84].

The principle of block-based motion estimation utilized in most of the video standards is that the video image frame is partitioned into blocks, where each block is the elementary unit. Motion estimation is performed by matching each block in the current frame against a region in a reference frame to find the best match (see Figure 1.1). Different matching criteria have been used in the search region, however the minimum mean (or sum of) absolute differences (respectively Mean Absolute Difference (MAD) or Sum Absolute Difference (SAD) by using one of these matching criteria, the best match can be located [Chu01].

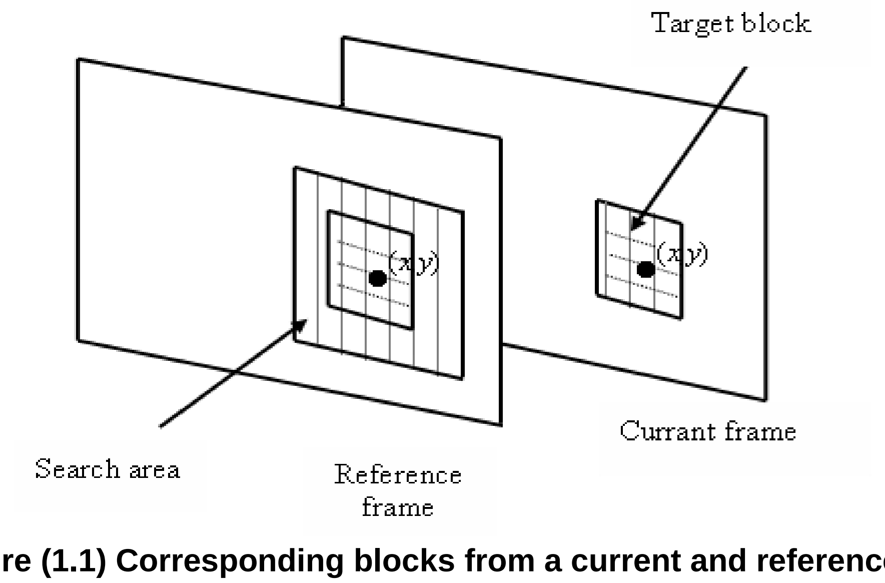


Figure (1.1) Corresponding blocks from a current and reference frame, and the search area in the reference frame.

1.7 Related Works

1. Meyer, et al (1997), [Mey97], they aimed to construct an algorithm for motion compensation of wavelet coefficients by using block matching techniques, the results of the new prediction method showed higher PSNR, and better perceptual quality. The algorithm works directly on the wavelet coefficients and the researcher recommend it to be successfully used for wavelet based very low bit rate video coding.
2. Liu, et al (1998), [Liu98], they introduced several block-based fast motion estimation methods for video compression. Two classes of algorithms have been summarized. The three step search algorithm and once at a time search algorithm were examined along with the improvements. The advantages of each algorithm were given in terms of motion estimation computational complexity.
3. Xiao, (2001), [Xia01], his project studied image compression with wavelet transforms. As a necessary background, the basic concepts of graphical image storage and currently used compression algorithms are discussed. The mathematical properties of several types of wavelets, including Haar, Daubechies, and biorthogonal wavelets are covered, and then analyzed the compression results to compare the wavelet types, then found that the biorthogonal wavelet gave a good compression results than the other types.
4. Kadhim, (2002), [Kad02], her project was aimed to investigate the performance of the H.263 video compression. She developed all the required programs. Also, an adaptive mechanism was proposed and implemented to handle the time delay associated with some estimation methods. The proposed speed up mechanism had not

significantly affected the compression efficiency and image quality.

5. Ibraheem, (2004), [Ibr04], he introduced an image compression system based on wavelet transform coding. In his project, some additional coding techniques were implemented such as Differential Pulse Code Modulation (DPCM), and S-shift coding to improve the compression performance. His test results indicated that the degradation in image quality was kept as minimum loss as possible (i.e. got minimum MSE with good PSNR (from 24 to 40), and C_r from (2 to 5)).
6. Kotteri, (2004), [Kot04], had designed and implemented image compression by using biorthogonal tap9/7 DWT and performed quantization on wavelet coefficients. He utilized the fidelity measure (MSE and PSNR) to asses of quality of the compressed image. To avoid wasted in computation he improve efficiency of the filter bank.
7. Piella, (2005), [Pie05], they presented a new class of adaptive wavelet decompositions that can capture the directional nature of picture information. This method exploited the properties of built lifting structures able to choose between different update filters, the choice being triggered by a local gradient of the input. In order to discriminate between different geometrical information, the system makes use of multiple criteria, giving rise to multiple choices of update filters.

1.8 Aim of Thesis

This research work aimed to design and implement a video compression scheme that based on utilizing both wavelet transform (to compress anchor frames) and inter-block correlation that based on enhanced the performance of motion estimation process, which in turn will speed up the inter-frame compression stage.

In the proposed work, wavelet transform (TAP9/7) is planned to be utilized in addition to some other coding techniques; whose implementation require low-complexity computation power.

Also, in this work an enhancement version of OTS motion estimation is proposed, the enhancement is based on utilizing the benefits of the existing correlation between the motion vectors of the adjacent blocks. This method was implemented to speed up the motion estimation task, and to preserve the quality of the coded image.

1.9 Thesis Layout

Beside to this chapter the remaining part of thesis consists of the following four chapters:

- **Chapter Two:** this chapter discusses the image and video compression techniques in details, including wavelet transform, and the methods of motion estimation.
- **Chapter Three:** this chapter includes in details the designed and implemented video compression models. All the developed algorithms that used in this research work are presented.
- **Chapter Four:** this chapter contains the results of the conducted tests on some samples of movies that used as test material in this work. The used performance criteria are the fidelity measures (MSE, PSNR) beside to compression rations.
- **Chapter Five:** this chapter includes the derived conclusions and some suggestions for future works.

Chapter Two
Image and Video Compression

2.1 Introduction

Image compression had been pushed to the forefront of the image processing field. This is largely a result of the rapid growth in computer power, the corresponding growth in the multimedia market, and the advent of the world wide web (WWW), which makes the internet easily accessible for every one. Additionally, the advances in video technology, including high definition television, have been created a demand for new, better and faster image compression algorithms.

The increasing demand to incorporate video data into telecommunic-ations services, the corporate environment, the entertainment industry, and even at home had made digital video technology a necessity. A problem, however, is that still image and digital video data rates are very large, typically in the range of 150Mbits/sec. Data rates of this magnitude would consume a lot of bandwidth, storage and computing resources in the typical personal computer. For this reason, video compression standards have been developed to eliminate picture redundancy [Arr97].

The development of compression algorithm had started with certain applications to two-dimensional (2D) still images. Since video and television signal consists of consecutive forms of image data, the development of compression methods for 2D still data is of paramount importance. After the development of some still image compression methods, they were extended to video (motion imaging) [Umb98].

2.2 Classification of Compression Techniques

Compression of digital data is based on various computational algorithms, which can be implemented either by software or by hardware. There are two primary types of image compression methods: Some compression methods preserve the data, while the other allows some loss of data. Therefore, compression techniques are classified into two categories [Umb98]:

1. Lossless compression methods.
2. Lossy compression methods.

2.3 Lossless Compression Methods

Lossless compression techniques provide the guarantee that no pixel difference will occur between the original and the decompressed image, in other words lossless schemes result in reconstructed data that exactly matches the original. It is generally used for applications that cannot allow any difference between the original and reconstructed data. The most popular lossless compression methods are: ***Huffman coding, Arithmetic coding, S-shift coding, and Run length coding,*** [Avc02].

2.3.1 Huffman Coding

Huffman coding was developed by D. Huffman in 1952, has a minimum average length of codewords. This near optimality occurs when the statistical distribution of the gray levels (the histogram) is given. Huffman algorithm generates a code that is as close as possible to the entropy (minimum bound) of the distribution.

This method results in a *variable length code*, where the codewords are of unequal length. For complex images, Huffman coding alone will typically reduce the file by 10 to 50% (1.1:1 to 1.5:1), but this ratio can be improved to reach 2:1 or 3:1 by preprocessing for the irrelevant information removal [Dra06] [Umb98].

The flow of Huffman algorithm can be described by the following five steps:

1. Find the gray-level frequency of occurrence (histogram), and then determine the probability density function (PDF).
2. Order the PDF values (or histogram magnitudes) from smallest to largest.
3. Combine the smallest two by addition.
4. Goto step 2, until only two probabilities are left.
5. By working backward along the tree, generate code by alternating assignment of (0 and 1).

2.3.2 Arithmetic Coding

This type of coding was introduced in 1976. It assigns variable length codes to variable length blocks of symbols. Because it does not require assignment of integer length codes to fixed length blocks of symbols, the average bit length required for the codes can approach more closely the lower bound or (the entropy) of the symbols. Therefore, it achieves higher compression rates than other variable length coding methods. Implementations of arithmetic coding are very complicated and need to overcome the precision problem. The other disadvantage is its decoding speed [Chr00].

2.3.3 S-Shift Coding

The idea of this method is to encode the sequence of numbers by codewords whose bit length is less than the bit length required to represent the maximum value of the sequence of numbers to be coded. The numbers whose values are large may splitted into a sequence of codewords, by using the following formula:

$$X = nW_m + W_r \quad \ldots \quad (2.1)$$

Where:

X is the number to be coded.

n is the number of codewords that used to encode the number X.

W_m is the lowest value which cannot be coded by using a single codeword.

W_r is the value of the last codeword used to encode X.

The values of W_m, W_r, and n are determined by using the following equations:

$$W_m = 2^b - 1 \quad \ldots \quad (2.2)$$

$$W_r = X \bmod W_m \quad \ldots \quad (2.3)$$

$$n = X \operatorname{div} W_m \quad \ldots \quad (2.4)$$

Where b is the number of bits used to represent each single shift codeword.

The performance of Huffman coding and shift coding are better when the sequence of numbers has a histogram whose shape is highly peaked. The performance of shift coding is better than Huffman and arithmetic coding when the histograms have long tails [Ibr04, Gon00].

2.4 Lossy Compression

In order to achieve high compression ratio for the cases of complex images lossy compression methods are required. Lossy compression provides tradeoffs between image quality and degree of compression, which allow the compression algorithms to be customized to the application. The lossy compression methods are the basis of the available tools for compression algorithm development, and they provide a wide variety of compression ratios and image quality. In general a higher compression ratio results in a poorer image, lossy compression could be performed in both the spatial and transform domains, the most popular lossy compression methods are [Umb98]:

1. Predictive Coding.
2. Quantization.
3. Transform coding.
4. Sub-band Coding.
5. Fractal Image Compression.

2.4.1 Predictive Coding

Predictive coding uses a set of known pixels that previously transmitted, to predict the current pixel. When the prediction model is good then the difference between the prediction and the actual pixel value can be transmitted or stored more efficiently than the actual pixel intensity. The simplest predictive coding is the differential pulse code modulation (DPCM). Differential pulse code modulation uses a model of pixel statistics to determine the prediction coefficients in order to minimize the squared error of the pixel prediction [Smi92]. This coder predicts that the next pixel has the same value as the previous pixel. Then, the predicted value is subtracted from the actual value, and the difference is coded. Higher order predictors use more than a single past pixel in the prediction [Til94].

2.4.2 Quantization

Quantization involved in image processing. Quantization techniques generally compress a range of values to a single quantum value. By reducing the number of discrete symbols in a given stream, the stream becomes more compressible. There are number of quantization methods:

1. ***Scalar Quantization (SQ)***

   Scalar quantization is used to reduce the number of bits needed to store a set of real numbers. This can be done by dividing each number by a quantization factor and rounding it to the nearest integer before it is stored. To retrieve the number again, the stored quantized integer (quantization index) is multiplied by the quantization factor again. This step is not lossless, because the retrieved number doesn't have the exact values of the originals, the degree of closeness depends on the value of quantization factor [Aal96].

2. ***Vector Quantization (VQ)***

   In the recent years, vector quantization (VQ) has been found to be an efficient for image compression in comparison with the old classical scalar quantization (SQ) techniques. Vector quantization is a lossy compression method. It uses a codebook containing pixel patterns with corresponding indexes on each of them. The main idea of VQ is to represent arrays of pixels by an index in the codebook. In this way, compression is achieved because the size of the index is usually a small fraction of that of the block of pixels.

   The main advantages of VQ are the simplicity of its idea and the possible efficient implementation of the decoder. Moreover, VQ is theoretically an efficient method for image compression, and a superior performance will be gained for large vectors. However, in order to use large vectors, VQ becomes complex and requires many computational resources (e.g. memory, and computations per pixel) in order to efficiently construct and search a codebook. More research on reducing this complexity has to be done in order to make VQ a practical image compression method with superior quality [Xia01].

2.4.3 Transform Coding (TC)

The fact that pixels are highly correlated with their neighbors also suggests that pictures contain an important low-frequency component. Transform coding methods utilize this fact by transforming the image into the frequency domain, followed by efficient coding of the resulting coefficients. Since most of the higher frequency components lack energy, they can be coded with fewer bits or eliminated completely. Various fast algorithms were developed to perform the necessary transformations [Til94].

1. Fourier Transform.
2. Cosine Transform.
3. Wavelet Transform.

2.5 Wavelet Transform (WT)

During the last decade, a number of signal processing applications have emerged using wavelet theory. Among these applications widespread developments have been occurred in the area of data compression. Wavelet techniques have demonstrated the ability to provide not only high coding efficiency, but also spatial and quality scalability features [Shi00].

The importance of wavelet comes from its ability to decompose the image data into multilevel of independent information with changing the scale (like a geographical map in which the image has non-redundant information due to the change of the scale). Due to wavelet transform every image will be transformed in each level of decomposition to a one low information sub-image and three detail sub-images along the horizontal, vertical and diagonal axis of image, as shown in Figures (2.1- 2.2- 2.3). Also the low information image can be decomposed into another four sub-images. This approach of decomposition process provides us with a number of unrealizable features in the original image, which appear in their levels after the application of wavelet transformation. So, the wavelet can be regarded as the most efficient transform that dealing with image, sound or other patterns since it provides a powerful time-frequency representation [Gon00].

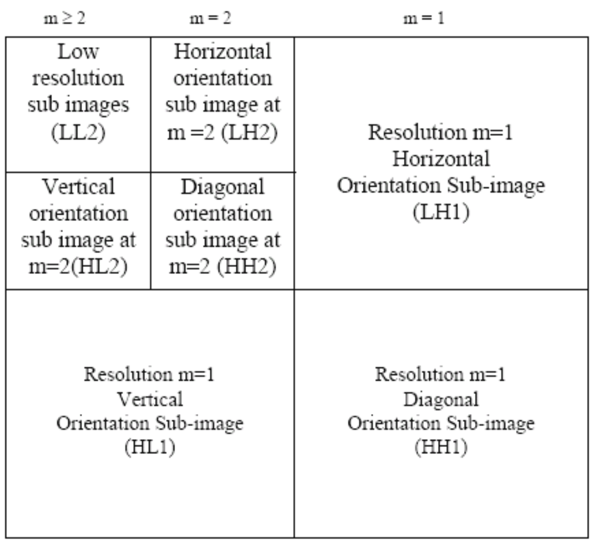
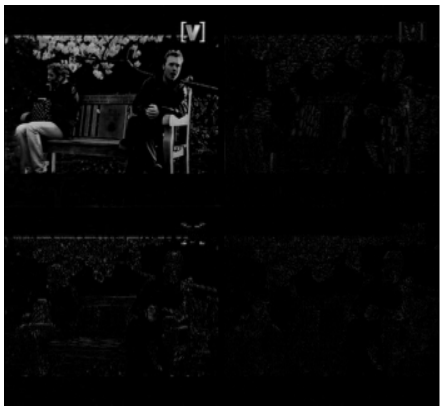

m ≥ 2
m = 2
m = 1
Low resolution sub images (LL2)
Horizontal orientation sub image at m =2 (LH2)
Resolution m=1 Horizontal Orientation Sub-image (LH1)
Vertical orientation sub image at m=2(HL2)
Diagonal orientation sub image at m=2 (HH2)
Resolution m=1 Vertical Orientation Sub-image (HL1)
Resolution m=1 Diagonal Orientation Sub-image (HH1)

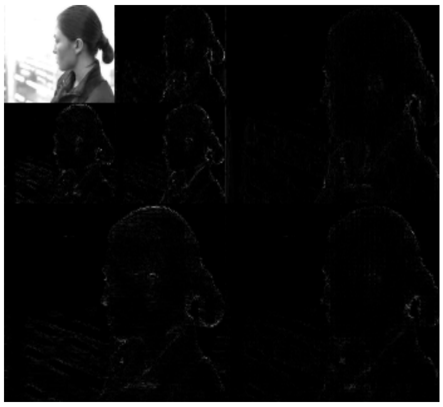
Figure (2.1) The subbands of wavelet decomposition.

The WT is a family of transforms that satisfies specific conditions. It is a transform that has the two basic mother functions (shifting and expansion). To satisfy the conditions of a WT, the filters must reconstruct the signal (or image), which means that any distortion introduced by the forward transform will be canceled in the inverse transform [Umb98].



(a) Original image. **(b) Its wavelet decomposition.**

Figure (2.2) The original image and its wavelet subbands after one decomposition Level.



(a) Original image. **(b) Its wavelet decomposition.**

Figure (2.3) The original image and its wavelet subbands after two decomposition Levels.

2.5.1 The Lifting Scheme

The lifting scheme is an efficient implementation of a wavelet transform algorithm. It was primarily developed as a method to improve the implementation of wavelet transform, and then it was extended to a generic method to create the so-called second-generation wavelets. Second-generation wavelets are much more flexible and powerful than first generation wavelets. It is shown that any discrete wavelet transform can be done through a finite number of lifting steps. So, the lifting scheme is an implementation of the filtering operations through a sequence of phases. It can be described in three phases, namely: Split phase, Predict phase and Update phase, as illustrated in Figure (2.4) [Mah05]

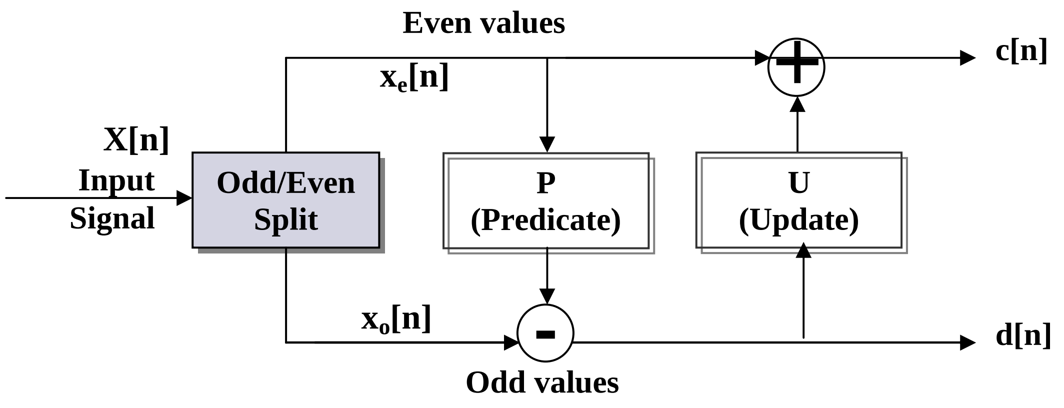


Figure (2.4) Lifting scheme transform.

In the ***split phase*** the original data is partitioned into 2-subbands: the first subband contains the samples of odd indices and the second one contains the samples of even indices. The ***predict phase*** (***P***) is applied to obtain the coefficient of the high frequency (***HF***) subbands details (these coefficient are called wavelet coefficients), and the purpose of this step is to predict values of the even samples based on the values of the odd samples. After the predict step, the ***update phase (U)*** is applied to obtain the coefficients of low frequency (***LF***) subband by making the average value of the output of low-pass coefficient equal to the average values of the original input data. So the purpose of ***(U)*** is to update the even samples by using the previously calculated detail (wavelet) coefficients [Pay05] [Vra04].

2.5.2 The Integer Wavelet Transform (TAP 5/3)

Wavelet transforms operate on integer values to produce integer valued wavelet coefficients. Integer wavelet transforms have been effectively used for lossless compression of images, the results of the invertible integer wavelet transforms are integer, while the computations are done with floating point numbers. Rounding each filter output to an integer value, a transform that maps integers to integers can be obtained. This kind of transform is named IWT which is based on the Lifting Scheme (LS) [Dro01].

2.5.3 Float Wavelet Transform (TAP 9/7)

The (9/7) biorthogonal filter was chosen as the basis of the JPEG2000 lossy image compression standard for still images. The coefficients of this filter are given as floating-point numbers. The float filters can be lifted (factorized) in order to speed up the convolution step. It is primarily suited to high visual quality compression. The use of floating-point arithmetic in the DWT, and the associated rounding errors, make it unsuitable for strictly lossless compression [Mah05].

Let us denote a row of pixels in a tile by P_k, P_{k+1}, through P_m. Because of nature of the wavelet transforms, a few pixels with indices less than k or greater than m may need to be used. The (9/7) floating-point wavelet transform is computed by executing four "lifting" steps followed by two "scaling" steps on the extended pixel values P_k through P_m . Each step is performed over all the pixels in the tile before the next step starts [Sal02].

The transform is computed as follow (the symbol “$\leftarrow$” means takes the value of) [Mah05]:

step 1: $X(k) \leftarrow X(k) + \alpha[X(k-1)+X(k+1)]$, for all odd k in the range $0 \le k < K$.
step 2: $X(k) \leftarrow X(k) + \beta[X(k-1) +X(k+1)]$, for all even k in the range $0 \le k < K$.
step 3: $X(k) \leftarrow X(k) + \gamma[X(k-1) + X(k+1)]$, for all odd k in the range $0 \le k < K$.
step 4: $X(k) \leftarrow X(k)+ \delta[[X(k-1) + X(k+1)]$, for all even k in the range $0 \le k < K$.
step 5: $F(k) \leftarrow X(k)^* \xi$, for all odd k in the range $0 \le k < K$.
step 6: $F(k) \leftarrow X(2i)/ \xi \;\; X(k)$, for all even k in the range $0 \le k < K$.

The values of X beyond [0, k] have been generated by using mirroring method.

The values of coefficients $\{\alpha, \beta, \gamma, \delta, \xi\}$ are listed in Table (2.1)

Table (2.1) Lifting coefficients of 9/7 biorthogonal analysis filter

Coefficient	Value
α	- 1.586134342
β	- 0.05298011854
γ	0.8829110762
δ	0.4435068522
ξ	1.230174105

The inverse transform implies same above steps but they arranged in reverse order and the coefficients $\{-\alpha, -\beta, -\gamma , -\delta , 1/\xi\}$ are used instead of their correspond $\{\alpha, \beta, \gamma, \delta, \xi\}$.

2.6 Video Compression

Video compression had attracted considerable attention over the last decade. In recent year, video compression had played a vital role in data storage and transmission. Several standards for video coding have been established, such as those of the Motion Picture Expert Group: MPEG-1 and MPEG-2, H.261, and most recently H.263. There is a large redundancy in any video sequence that has to be exploited by every efficient video coding scheme. Most of the popular compression schemes are involved with the removal of the spatial and temporal redundancies [Soo01] [Tur01] [Sch97].

1. ***Spatial redundancy:*** refers to the correlation present between different parts of a frame. Removal of spatial redundancy thereby involves looking within a frame. The spatial redundancy left in the prediction error is commonly reduced by a transform coding or a quantization.
2. ***Temporal redundancies:*** is found between successive frames of a video sequence. A lot of information present in a frame is also present in the frame that preceded it. Hence, removal of such temporal redundancy involves looking between frames by compressing frame differences instead of complete frames. Usually, the temporal redundancy is usually reduced by motion-compensated prediction of the current frame from a previously reconstructed frame.

A higher compression rate is achieved by predicting spatial frequencies using motion estimation (ME) and compensation (MC) techniques [Lag03].

2.6.1 Video Compression Technologies

Video compression technologies can be divided into two groups:

1. Inter frame compression technologies
2. Intra frame compression technologies

Table (2.2) below describes the main algorithms for each group and its main characteristics.

Table (2.2) Compression Technique

	Inter Frame	**Intra Frame**
Technology Name	MPEG-4 Main Profile, H.263+,MPEG-x	Wavelet, M_JPEG
Main Characteristics	Use both spatial redundancy and temporal redundancy	Use only spatial redundancy

The main characteristics could be summarized by the following two remarks:

1. Using only ***spatial redundancy*** means that each frame is compressed separately.
2. Using ***spatial redundancy together with temporal redundancy*** means utilizing the similarity between consequent frames and motion estimations.

In order to better understand the differences in video quality between the different compression algorithms, it is important to review the conceptual differences between inter and intra frame algorithms.

Inter-frame processing is the key to exploit and reduce the temporal redundancy in digital video compression (i.e. utilizing the similarity between consequent frames by motion estimations). Temporal redundancy exits due to the similarity between the sequential neighboring frames. In video compression, knowledge of motion helps to exploit this similarity and remove the temporal redundancy between neighboring frames in addition to the spatial redundancies. Motion estimation (ME) or motion compensation (MC) are the basic approaches to find out and represent the motion between frames. These techniques are widely used in video standards (including H.26x and MPEG) to achieve high data compression rate.

Compression algorithms, such as MPEG-4 main profile and H.263+, use temporal redundancy as well as spatial redundancy. Actually, it means that compression algorithms process subsequent frames and estimate the motion within the frames. Then the algorithm codes only the difference between the frames instead of the whole frame. This coding method results in higher video quality per given bit-rate, since the given bit-rate is used for representing a smaller amount of video data (i.e. the ***frame difference***).

Intra-frame compression algorithms process each frame separately, without analyzing the correlation between consequent frames (using only spatial redundancy). There are different kinds of compression methods applied for processing the frame and creating a compact representation of it (either by filters – Wavelet, or by a DCT transform – M-JPEG) [Nic03].

2.6.2 Video Coding Structure

The video coding structure is build from the original video data which represented as sequence of frames, most of the well known inter-frame compression techniques divide the whole frames into two types, the first type consists of the ***anchor (reference) frames,*** and the second type consists of the ***estimated frames***, as shown in Figure (2.5). The Anchor frames are compressed independently and separately without any considerations to the correlations may exist within the neighboring frames, while the estimated frames are compressed using motion estimation methods, which encode the estimated frames according to the correlations with the neighboring anchor (reference) frames, [Jab05].

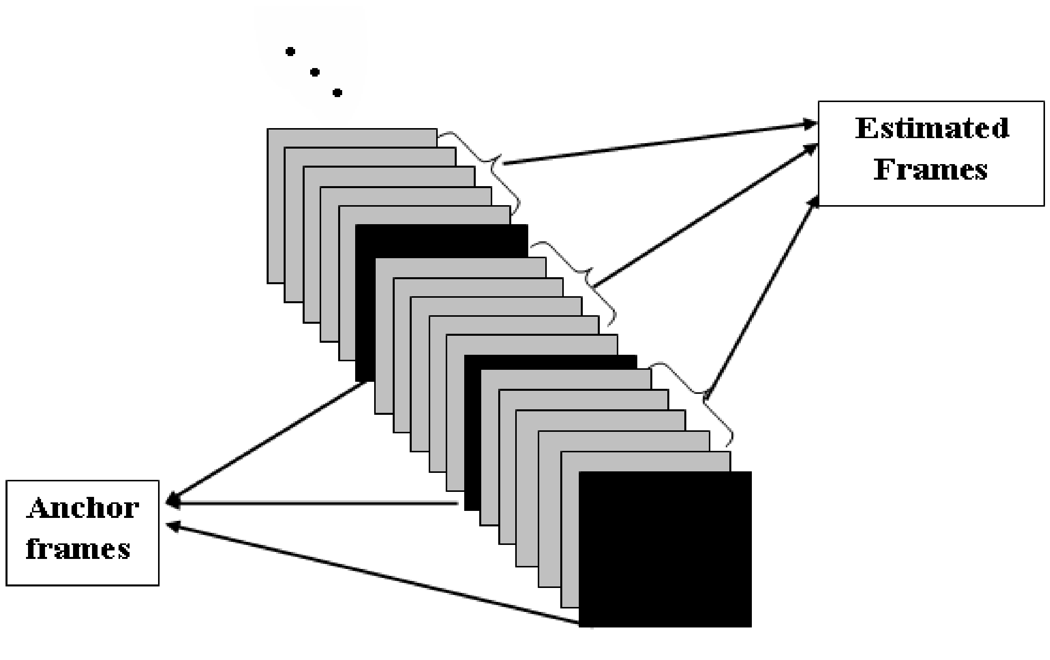


Figure (2.5) Video coding structure

2.6.3 Motion Estimation (ME)

Motion estimation (ME) is a powerful method utilized in a variety of video processing systems. Many methods have been developed for different applications over the last 30 years, including motion compensated (MC) ***filtering*** for noise reduction, MC ***prediction*** for coding, and MC ***interpolation*** for video format conversion (MC-VFC) [Haa00].

Motion estimation is computationally intensive, but is pivotal to the success of video coding. A good motion estimation algorithm can lead to efficient reduction of the temporal redundancies and a fast implementation, which is essential to real-time video coding applications, this is accomplished by estimating the displacement between frames of picture elements, which may be either uniformly sized blocks or individual pixels [Chu98] [Zak93].

2.6.4 Block-Based Motion Estimation Methods

Block-based motion estimation methods are the most popular methods for motion estimation because of their simplicity and ease of implementation. The block matching is seemingly used by the video compression standards because it can achieve a good balance between complexity and coding efficiency. It involves finding a candidate block in a specified search area, in the previous frame that is most similar to the current block in the current frame. The block matching method can be categorized into frame based block matching and object based block matching method. There are many ME techniques used nowadays, the conventional frame based block matching technique is considered to be the full search or exhaustive search. This technique is often used for motion searching in relatively small search ranges due to the heavy computation and extension data fetching between the frame buffer and ME [Soo01].

In the last decade, many fast-searching algorithms have been developed the choice of an algorithm for motion estimation is governed by the ‘speed-quality-bitrate’ tradeoff. A good algorithm is one that has a low computational complexity, provides a high quality of motion compensation and also ensures that the bit-stream is as small as possible.

There are a large number of different near-optimal search algorithms for block based motion estimation. These include [Alk99]:

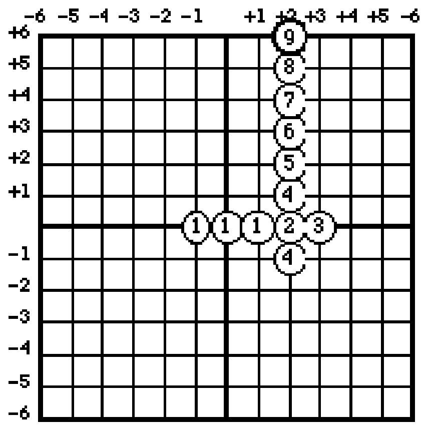
1. Once-at- a Time Search (OTS).
2. Three Step Search (TSS).
3. Two-D Logarithmic Search.
4. Four Step Search (FSS).
5. Orthogonal Search.
6. Cross Search.

However, most of these fast hierarchical algorithms use the origin of the searching window as the initial search center, and they have not exploited the motion correlation of the blocks among the same image moving object [Jie99].

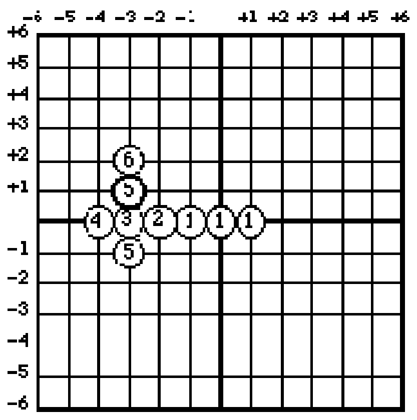
In matching criteria, the ME scheme have to use a criteria to search for. Most ME schemes look for a minimum mean square error (MSE), and the sum of absolute errors (SAE) between blocks [Are04].

A. Once at a Time Search Method (OTS)

This is a simple and effective way of trying to find a point with the optimal block. During the horizontal stage, the point on the horizontal direction with the minimum distortion is found. Then, starting with this point, the minimum distortion in the vertical direction is found [Tura98]. OTS consists of estimating the motion on the X axis by searching for the position of the least error (i.e. best sum of absolute difference (SAD)) on the considered point and the left and right points. The point with lowest error is taken as the result of the present step. The process is repeated until the minimum becomes the central examined point the search area. Then, the process is repeated on the Y axis, examining the up and down points, as shown in Figure (2.6), [Zan92], [Shi00].



Match is found at position [+2,+6]



Match is found at position [-3,+1]

Figure (2.6) OTS Method

Figure (2.6) above illustrates the OTS finding positions of minimum distortion. In the left example the position of minimum distortion is at [+2,+6] and in the right example it is at [-3,+1].

B. Three Step Search Method (TSS)

This algorithm was introduced by Koga et al in 1981. It became very popular because of its simplicity, robustness and near optimal performance. It searches for the best motion vectors in a coarse to fine search pattern. The algorithm may be described as [Tura98]:

Step 1: An initial step size is picked. Eight blocks at a distance of step size from the centre (around the centre block) are picked for comparison.

Step 2: The step size is halved. The centre is moved to the point with the minimum distortion.

Steps 1 and 2 are repeated till the step size equal (1). A particular path for the convergence of this algorithm is shown in Figure (2.7).

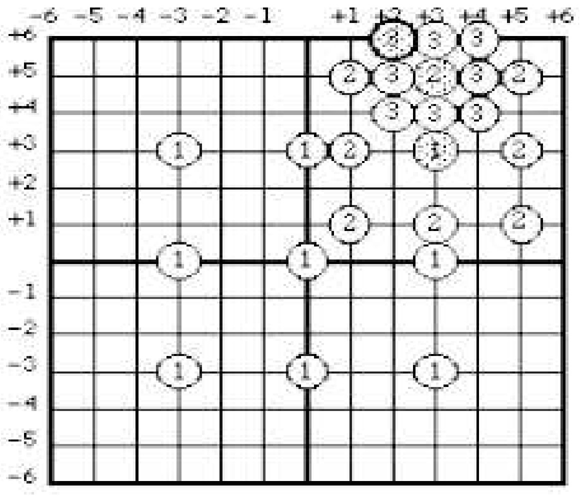


Figure (2.7) TSS method

The Three Step Search (TSS) converging on a position of minimum distortion at [+2, +6].

C. Two Dimensional Logarithmic Search (TDL)

This algorithm was introduced by Jain & Jain around the same time that the three step search was introduced, and is closely related to it. Although this algorithm requires more steps than the three step search, it can be more accurate, especially when the search window is large. The algorithm may be described as follows, [Tura98]:

Step 1: Pick an initial step size. Look at the block at the centre of the search, and at four positions at a distance (s) from the centre, and located on the X and Y axes.

Step 2: If the position of best match is at the centre, halve the step size. If however, one of the other four points is the best match, then it becomes the center and step 1 is repeated.

Step 3: When the step size becomes 1, all the nine blocks around the center are chosen for the search and the best among them is picked as the required block.

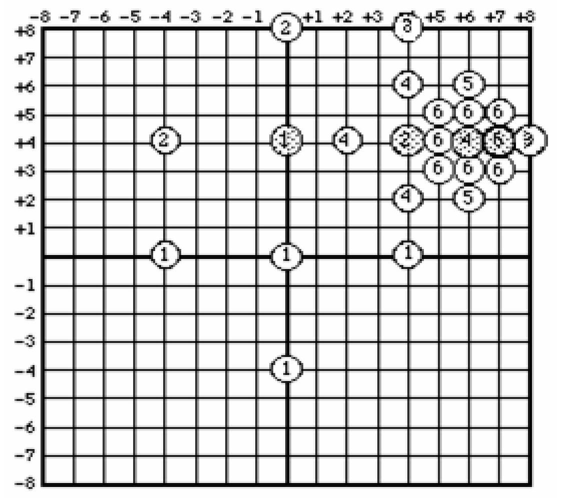


Figure (2.8) TDL method

Figure (2.8) above illustrates the 2-D logarithmic (TDL) search converging on a position of minimum distortion. The points [0,+4], [+4,+4], [+6,+4] are the minima at each stage and finally [+7,+4] is chosen as the matching block.

A lot of variations of this algorithm exist and they differ mainly in the way in which the step size is changed.

Chapter Three
Proposed Video Compression System

3.1 Introduction

In this chapter the layouts of the two proposed video compression schemes are investigated and the implementation steps are illustrated. The first video coding scheme the video frames are handled either as still images or by applying motion compensation. While is the second scheme the interpolation coding method is implemented to handle some frames lay between two successive anchor frames.

Anchor frames are compressed by using wavelet transform, the wavelet coefficient are quantized and finally coded by using some entropy encoders.

In this research work, the standard method OTS, in addition to a new designed method are used to determine the motion vector of the blocks. The new motion estimation method (MOTS) can be considered as a modified version of the standard OTS method.

3.2 Media File

The multimedia file that used in this research project as uncompress media file is the Audio/Video Interleaved (AVI) file. The established system takes an AVI file as input media file and produce a compact file that represents the compressed video file.

Microsoft AVI file format is a Resource Interchange File Format (RIFF), it is used with applications that capture, edit, and play back audio-video sequences. In general, AVI files contain multiple streams of different types of data. Most AVI sequences contain both audio and video streams. See appendix-A (AVI file format).

In this project the audio had been separated from the video by using (*VCDCutter ver. 4.04*) program, then only the video sequences have been used, they consist of a sequence of frames (bitmap images). Each bitmap (BMP) image consists of two parts header and data, the color of each pixel in the image consists of three color components (Red, Green, and Blue). See appendix-B (BMP file structure).

3.3 The Proposed Compression System

In the proposed work, the loaded video frames have been coded in two different ways: reference or anchor frames compression (intraframe coding), and motion estimation or compensation coding. See figure (3.1).The compression method based on intraframe coding is used to remove the spatial redundancy. In this research work, the wavelet transform, hierarchal scale quantization and shift coding methods have been used to encode the anchor frames. While the rest frames are compressed by using interframe coding, this is based on removing the temporal redundancy by implementing block motion estimation method. Two different methods of motion estimation (ME) have been considered, the first one of them is the standard OTS method, and the other is a modified (enhanced) version of OTS method, the proposed modification is based on the idea of arranging the steps of OTS methods such it is capable to utilize the benefits of the existing correlation between the motion vectors of the adjacent blocks. So the second method is called, a modified once at a time search (MOTS) method.

Two video coding schemes were implemented. The difference between the two schemes is the use of interpolation process, where the second scheme implies interpolation process to encode some intermediate frames, between successive reference frames, while in the first scheme no interpolation was utilized. Both schemes contain the coding processes: anchor frame compression (AFC) and motion estimation (ME).

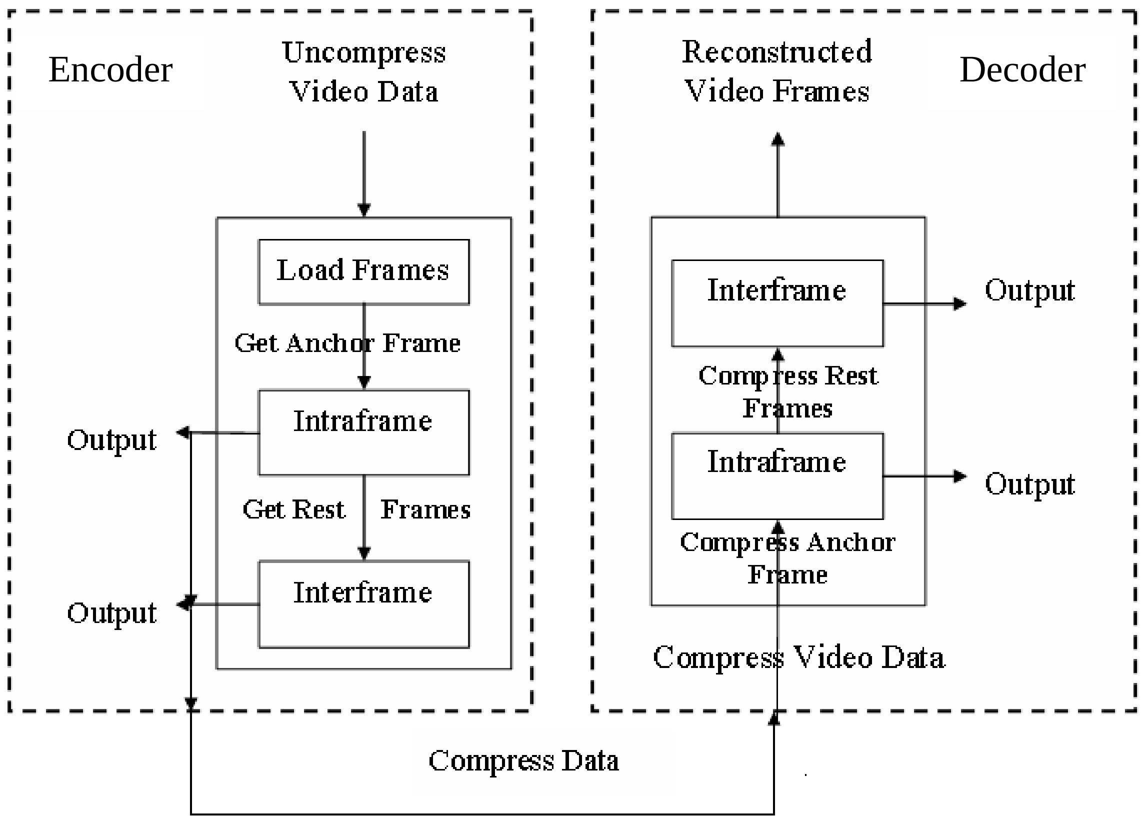


Figure (3.1) The proposed system structure

3.3.1 The First Video Coding Scheme (without interpolation)

As shown in Figure (3.2), the proposed video coding system compress the input video sequence as groups of frames (each group consist of N-frames). The first and last frame, in the group, are compressed as still images (intra frames), while the in-between frames are coded using motion estimation methods.

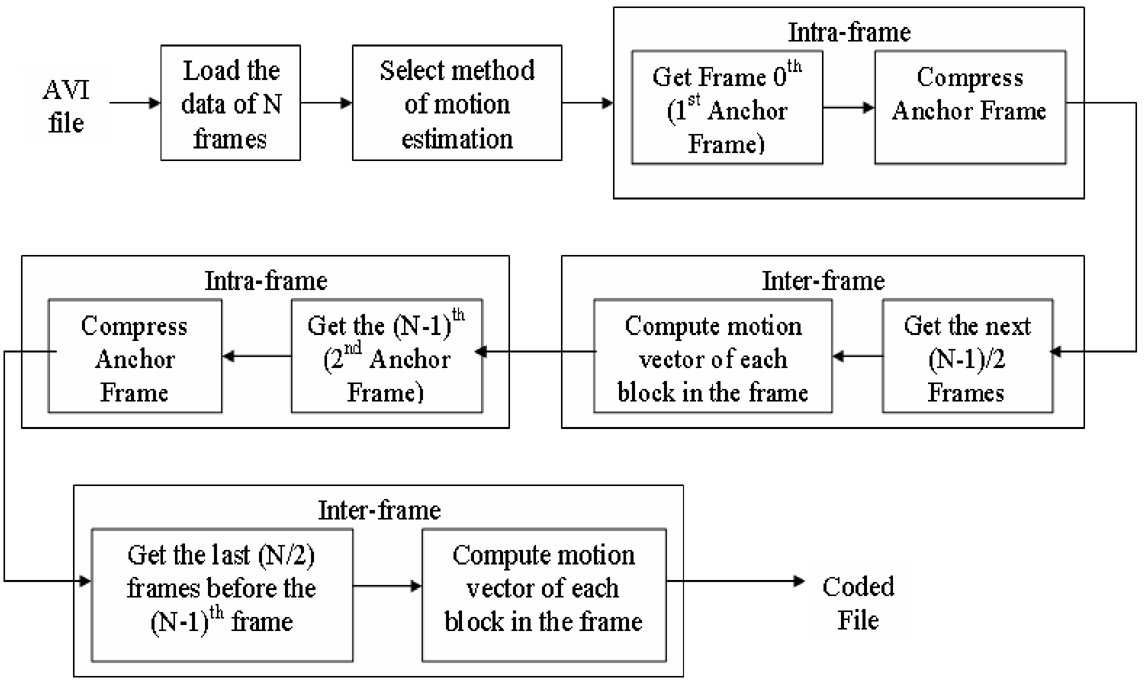

AVI file
Load the data of N frames
Select method of motion estimation
Intra-frame
Get Frame 0th (1st Anchor Frame)
Compress Anchor Frame
Inter-frame
Get the next (N-1)/2 Frames
Compute motion vector of each block in the frame
Intra-frame
Get the (N-1)th (2nd Anchor Frame)
Compress Anchor Frame
Inter-frame
Get the last (N/2) frames before the (N-1)th frame
Compute motion vector of each block in the frame
Coded File

Figure (3.2) The structure of the first proposed video coding scheme

The system flow is illustrated by the following major steps:

1. As a first step, the system will load the header of AVI file, and encode its necessary contents (i.e. header compression) after reducing the unnecessary information.
2. Load the data of a group of N-frames, the first frame in the group is considered as an anchor frame (AF), and it is compressed by using wavelet transform, hierarchal uniform quantization, DPCM, shift coding.
3. Then one of the two adopted motion estimation (OTS, MOTS) method is applied to encode the next sequence (N-1)/2 frames. The determined motion is done relative to the previous anchor frame.
4. Get the second anchor frame (the last loaded video frame in the group) and compress it as still image, by using wavelet transform.

5. The frames sequences, starting from N/2 to N-2, are coded by applying the same motion estimation method but relative to the second anchor frame.

For example, if the size of the frame group was taken 12 frames, then the 1st and 12th frame are considered as the 1st and 2nd anchor (references) frames, respectively. Then the frames (2-6) are coded using ME method relative to the first anchor frame (AF), while the frames (7-11) are coded relative to the second anchor frame (AF), see figure (3.3).

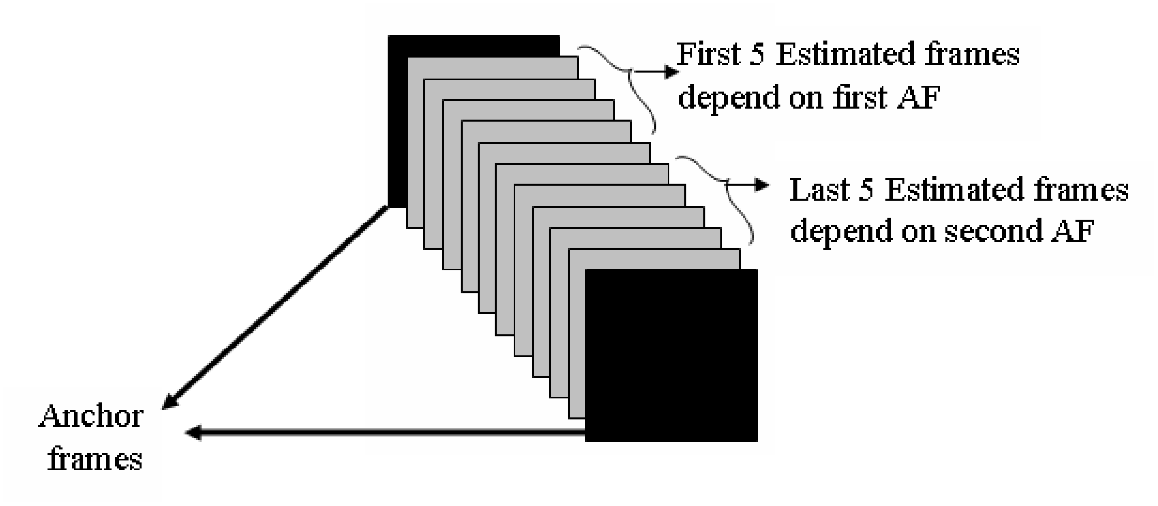

First 5 Estimated frames depend on first AF
Last 5 Estimated frames depend on second AF
Anchor frames

Figure (3.3) The types of coded frames by using the first proposed scheme

3.3.2 The Second Video Coding Scheme (with interpolation)

Figure (3.4) illustrates the proposed system structure after adding the interpolation coding method to improve the system compression performance by extending the number of frames coded within each group of frames or pictures (GOP). The interpolation process is applied by using the following linear interpolation method:

$$I_j(x,y) = \left(1 - \frac{j}{M+1}\right) A_i(x,y) + \frac{j}{M+1} A_{i+1}(x,y), \ldots\ldots\ldots(3.1)$$

Where,

I_j is the jth interpolation frame {j=1, 2, 3,…, M}.

M is the number of interpolated frames.

A_i represents the first anchor frame.

A_{i+1} represents the second anchor frame.

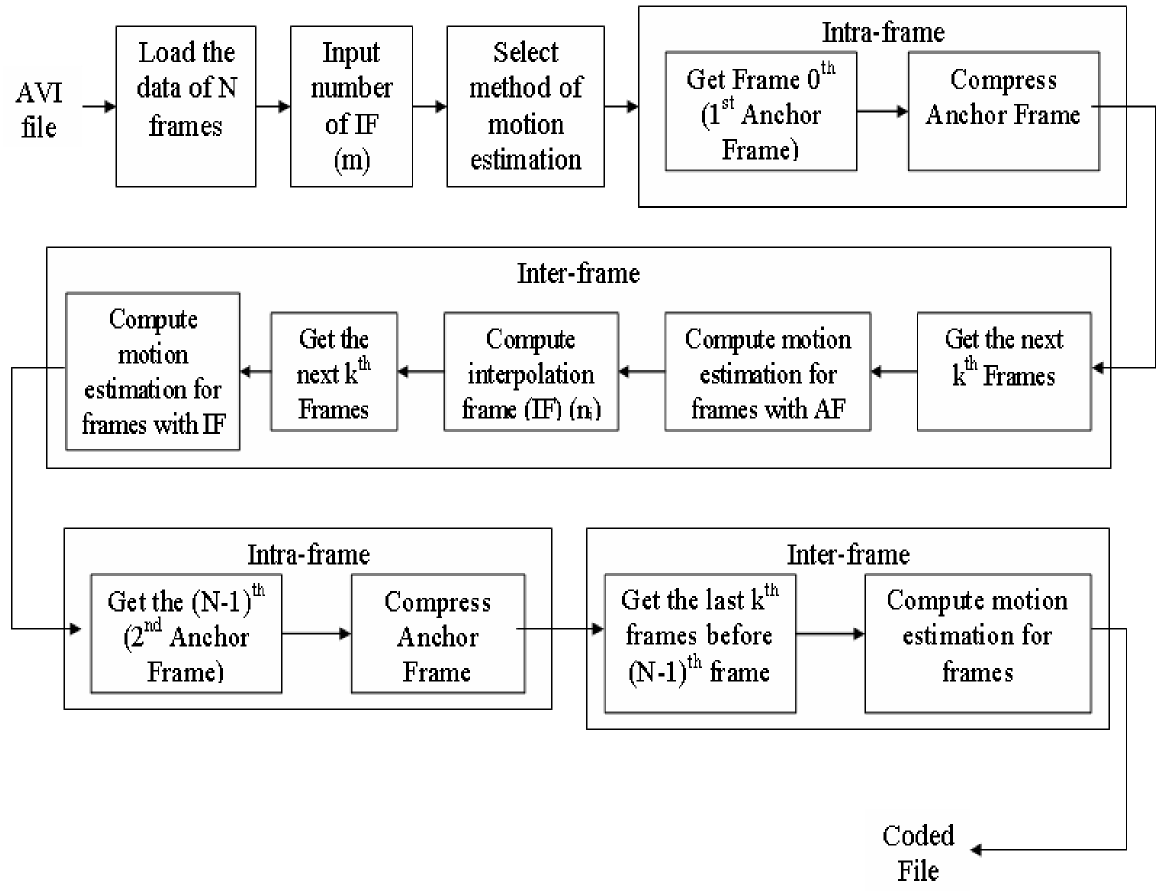

AVI file
Load the data of N frames
Input number of IF (m)
Select method of motion estimation
Intra-frame
Get Frame 0th (1st Anchor Frame)
Compress Anchor Frame
Inter-frame
Get the next kth Frames
Compute motion estimation for frames with AF
Compute interpolation frame (IF) (ni)
Get the next kth Frames
Compute motion estimation for frames with IF
Intra-frame
Get the (N-1)th (2nd Anchor Frame)
Compress Anchor Frame
Inter-frame
Get the last kth frames before (N-1)th frame
Compute motion estimation for frames
Coded File

Figure (3.4) The structure of the second proposed coding scheme

The system flow is illustrated by the following major steps:

1. As a first step, the system will load the header of AVI file and encode its contents (i.e. header compression) after reducing the unnecessary information.

2. Load a group of N-frames, the value of N is predefined by the user.
3. The first frame (anchor frame) is compressed as a still image, (i.e. by using wavelet transform).
4. The involved computations of the adopted motion estimation (ME) method are performed on the next sequence $k = \frac{(N - M - 2)}{(M + 1)}$ frames, where M is the number of interpolated frames. The value of M is predefined by the user. The determined motion vectors are computed relative to the previous anchor frame.
5. Get frames that have count number, $n_i=i(N-1)/(M+1)$, where $1 \le i \le M$, represents the index of the interpolated frames. Then, find the difference images produced by subtracting the established interpolation frame from their associated actual frames. These difference images are coded as still images (by using wavelet transform).
6. Get the next sequence of k frames and compress them using the adopted ME method relative to the nearest interpolation frame (IF).
7. Repeat step (5) for the next sequences of k frames till reaching the last k frames in GOP.
8. Get the second anchor frame (i.e. the last loaded video frame), then apply the ME on the last k sequence of frames relative to the second anchor frame.

The whole sequence of video frame will be divided into groups of frames, each group consists of N frames (in this work it was taken either 22 or 34 or 45). The first and last N^{th} frames in the group are considered as anchor frames, while the frames whose index is i(N-1)/(M+1) (where M is the number of interpolation frames in the group; and i is the index of the interpolation frame, it takes the values 1,2,….M) are coded using interpolation, see equation (3.1), relative to the two anchor frames, the number of interpolation frames (M) depends on the size of the group. The remaining frames in the group are compressed by using the motion estimation relative to the nearest anchor frames (AF) or interpolation frames (IF). For example, if the group size was taken 34 frame, and two interpolation frames are used, then the frames (2-6) are coded using ME method applied relative to the first anchor frame (AF), while the last five frames (29-33) are coded by using ME method relative to the second anchor frame (AF), the remaining 20 frames are compressed by using motion estimation relative to the two interpolation frames (IF), see Figure (3.5).

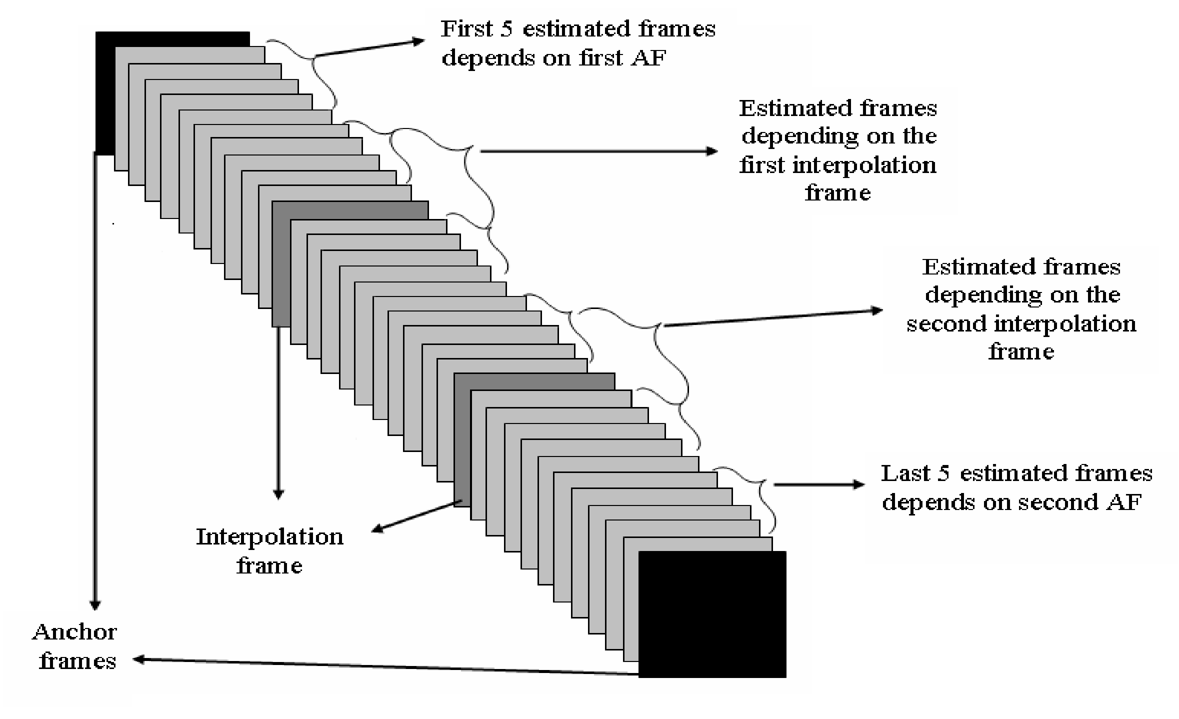
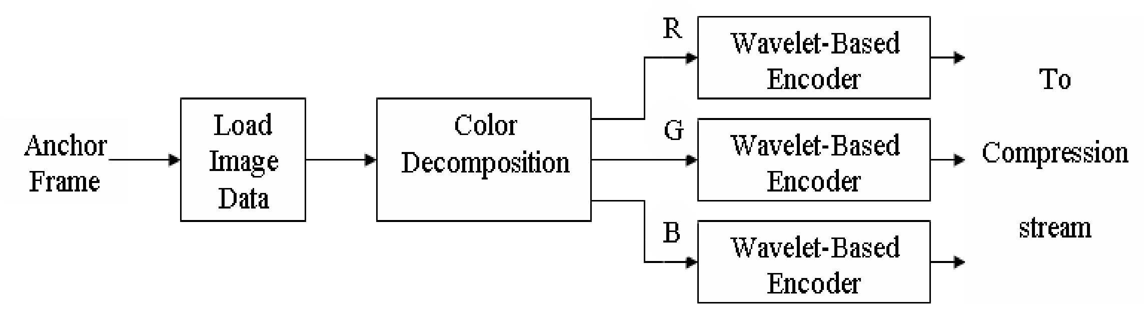

First 5 estimated frames depends on first AF
Estimated frames depending on the first interpolation frame
Estimated frames depending on the second interpolation frame
Last 5 estimated frames depends on second AF
Interpolation frame
Anchor frames

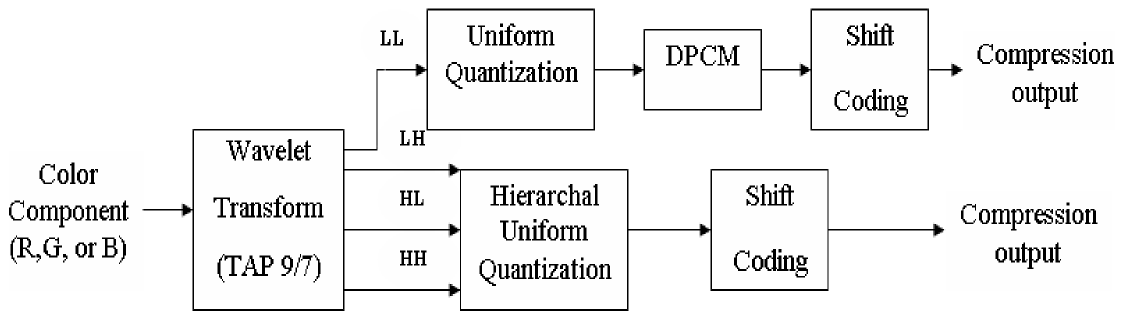
Figure (3.5) The types of the coded frames by using the second proposed scheme

3.4 Compression of Anchor Frames

Figure (3.6) illustrates the steps of the intraframe compression scheme.



a. General structure



b. Wavelet Based Encoder

Figure (3.6) The compression scheme for anchor frames

The video frames are 2D matrices, each element consists of three numbers; one for each color component (i.e., red, green, and blue components). The involved steps of the established compression scheme are summarized in the following:

1. Load image data: as a first step the image data is loaded.
2. Color decomposition: in this step the image data are decomposed into three matrices, each matrix holds the values of single color component (red, green, blue).

3. Wavelet transform: in this step the wavelet transform is applied on each color matrix, individually, to get four subbands (LL, LH, HL, HH) for each color band. The goal of wavelet analysis is to map the image data into alternative representation in which the image data is decomposed into subbands each hold certain kind of image information; such that most of the image information energy is concentrated in the lowest frequency subband (LL), and in a few high frequency coefficients. Such kind of data decomposition reduces the image data correlation and provides a useful data structure.
4. Quantization: a uniform quantization is applied on the WT coefficients to reduce the number of bits needed to represent, approximately, the coefficients.
5. Predictive Coding: the DPCM method is applied only on the quantized coefficients of the (LL) subband, this coding step takes the advantage of the fact that the adjacent coefficients are highly correlated, which means that the difference between adjacent pixels is typically small, and consequently, it will need a small number of bits to represent it.
6. Integer Mapping: the negative to positive mapping is applied to keep all coefficients values (DPCM coefficients and the quantized indices of the detail subbands) to positive numbers.
7. Shift Coding Optimizer: firstly the shift-coding optimizer is applied to search for the smallest codeword size needed to represent the coefficients.
8. Shift Coding: secondly encode the output of DPCM for (LL) subband and the quantized coefficients of the subbands (LH, HL, HH) by using shift encoder; then send the output (i.e. the codeword) to the compression stream.

3.5 Wavelet Transform (TAP 9/7)

In this research project, the wavelet transform (TAP9/7) was used as the transform engine. The implementation of this transform implies two stages. In the first stage each column of the image is transformed individually, where the odd and even elements of each column are separated from each other, by applied the equation as mentioned in chapter two on these two sets of elements:

In the second stage each row is transformed individually, where the row elements were divided into odd sequence and even sequence, then the same above equations have been applied on these two sequences of the row data. After the completion of the two stages the transform results put in a 2D matrix, the resultant matrix consists of four subbands (LL, LH, HL, HH), respectively. The WT can be applied more than one time (pass), in each next pass the transform is done on the LL coefficients to get another four subbands. Algorithms (3.1) and (3.2) shows the implemented steps of the wavelet transform.

Algorithm (3.1) Two Dimensional Wavelet Transform

Input:

```
Img() is the original image data
W is the image width
H is the image height
NoPass is the number of wavelet transform levels or passes
```

Output:

```
Timg() an array consists, at least, of four subbands (LL,LH,HL,HH)
```

Procedure:

```
Wid ← W:  Hgt ← H
Construct an array Timg() has the same size of Img()
Put all contents of Img in Timg
For all k   where   $1 \le k \le NoPass$
  For I= the index of each column in Timg (where $0 \le I \le H-1$)
      Move the contents of the I^th column of Timg in the array A
      Call wavelet TAP9/7(A, Hgt, B)
      Move the contents of the array B in the I^th column of Timg
  End loop I
  For J= the index of each row in the Timg array (where $0 \le J \le W-1$)
      Move the contents of the J^th row of Timg in the array A
      Call wavelet TAP9/7(A, Wid, B)
      Move the contents of the array B in the J^th column of Timg
  End loop J
Wid=Wid\2
Hgt=Hgt\2
End loop k
```

Algorithm (3.2) Wavelet TAP9/7

Input:

A() is the input array

N is the number of the input (or output) elements

Output:

B() is the output of one dimension wavelet transform

Procedure:

```
$N_{even} \leftarrow (N+1)/2 : N_{odd} \leftarrow N - N_{even}$
$N_m \leftarrow N - 1 : N_{p1} \leftarrow N + 1 : N_{p2} \leftarrow N + 2$
For all k where    $0 \le k \le N_m$
  $C(k) \leftarrow A(k)$
End loop k
For all k where    $1 \le k \le 4$
  $C(-k) \leftarrow C(k)$
  $C(N_m + k) \leftarrow C(N_m - k)$
End loop k
For all k where    $(k = -3, -1, 1, 3, \ldots\ldots, N_{p2})$
  $C(k) \leftarrow C(k) - 1.586134342 * (C(k-1) + C(k+1))$
End loop k
For all k where    $(k = -2, 0, 2, 4, \ldots\ldots, N_{p1})$
$C(k) \leftarrow C(k) - 0.05298011854 * (C(k-1) + C(k+1))$
End loop k
For all k where    $(k = -1, 1, 3, 5, \ldots\ldots, N)$
  $C(k) \leftarrow C(k) - 0.8829110762 * (C(k-1) + C(k+1))$
End loop k
```

To be continue

```
For all k where    $(k=0, 2, 4, 6, \ldots\ldots, N_m)$
  $C(k) \leftarrow C(k) - 0.4435068522 * (C(k-1) + C(k+1))$
End loop k
$Aa \leftarrow 1/1.230174105$
$i \leftarrow 0$
For all k where    $0 \le k \le N_{even}-1$
  $B(k) \leftarrow C(i) * Aa$
  $i \leftarrow i+2$
End loop k
$j \leftarrow 1$
For all k where    $0 \le k \le N_{odd}-1$
  $B(k+N_{even}) \leftarrow C(j) * 1.230174105$
  $j \leftarrow j+2$
End loop k
```

3.6 Subbands Coefficients Quantization

Image quantization is the process of reducing the number of possible values of a quantity and consequently reducing the number of bits needed to represent it.

In this research work, the uniform quantization was adopted to quantize the coefficients of each subband individually, where the quantization step used to quantize the coefficients of each subband was determined according to the following equation:

$$Q_{step} = \begin{cases} Q_L & \text{for LL coefficient} \\ Q\,\alpha^{n-1} & \text{for LH, HL in n}^{\text{th}}\text{ level} \\ Q\,\beta\,\alpha^{n-1} & \text{for HH in n}^{\text{th}}\text{ level} \end{cases} \quad , \ldots\ldots\ldots\ldots\ldots(3.2)$$

Where n is the wavelet level number (i.e., the pass number), (Q, α, β) are quality numbers (such that, $Q \geq 1$, $Q_0 \geq 1$, $\alpha \leq 1$, $\beta \geq 1$). According to the above equation the value of the quantization step is reduced with the increase of the wavelet level, and its value for HH-subband is greater than its value for the corresponding HL and LH subbands.

So, the quantization index for each approximate and wavelet coefficient is determined by using the following equation:

$$I_q(x,y) = round\left(\frac{Timg(x,y)}{Q_{step}}\right) ,\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.3)$$

Where

Timg is the array of the wavelet transform coefficients.

B_q its quantization index.

Algorithm (3.3) illustrates the implemented steps of the heirarical uniform quantization method.

Algorithm (3.3) Image Quantization

Input:

Timg() is the array of the wavelet transform coefficients.

W is the image width.

H is the image height.

Q_0 is quantization step for (LL).

Q is the initial quantization step for (LH, HL, HH).

α, β are the quality numbers.

Output:

Quantization indicies $I_q()$

Procedure:

```
$W_m \leftarrow W$: $H_m \leftarrow H$
For all k      where   $1 \leq k \leq NoPass$
  $W_m \leftarrow (W_m+1)$ div 2
  $H_m \leftarrow (H_m+1)$ div 2
End loop k
```

To be continue

```
'''' Quantization of (LL) coefficients
 $X_s \leftarrow 0$:  $X_e \leftarrow W_m - 1$:  $Y_s \leftarrow 0$: $Y_e \leftarrow H_m - 1$
 For all x, y   where   $X_s \le x \le X_e$  and  $Y_s \le y \le Y_e$
   $I_q(x, y) \leftarrow round\ (Timg(x, y) / Q_L)$
 End loop x, y
''' Quantization of (LH, HL, HH) coefficients
 $W_w \leftarrow W : H_h \leftarrow H$
 For all  j  where   $1 \le j \le NoPass$
   $W_m \leftarrow (W_w+1)\ div\ 2$
   $H_m \leftarrow (H_h+1)\ div\ 2$
   $W_{m1} \leftarrow W_w - 1$
   $H_{m1} \leftarrow H_h - 1$
   $X_s \leftarrow W_m$:  $X_e \leftarrow W_{m1}$:  $Y_s \leftarrow 0$: $Y_e \leftarrow H_m - 1$
   $Q_{step} \leftarrow Q * \alpha^{j-1}$
   For all x, y   where   $X_s \le x \le X_e$  and  $Y_s \le y \le Y_e$
     $I_q (x, y) \leftarrow round\ (Timg(x, y) / Q_{step})$
   End loop x, y
   $X_s \leftarrow 0$:  $X_e \leftarrow W_m - 1$:  $Y_s \leftarrow H_m$ : $Y_e \leftarrow H_{m1}$
   For all x, y   where   $X_s \le x \le X_e$  and  $Y_s \le y \le Y_e$
     $I_q(x, y) \leftarrow round\ (Timg(x, y) / Q_{step})$
   End loop x, y
   $X_s \leftarrow W_m$:  $X_e \leftarrow W_{m1}$ : $Y_s \leftarrow H_m$ : $Y_e \leftarrow H_{m1}$
   $Q_{step} \leftarrow Q_{step} * \beta$
   For all x, y   where   $X_s \le x \le X_e$  and  $Y_s \le y \le Y_e$
     $I_q(x, y) \leftarrow round\ (Timg(x, y) / Q_{step})$
   End loop x, y
   $W_w \leftarrow W_m$: $H_h \leftarrow H_m$
 End loop j
```

3.7 Differential Pulse Code Modulation (DPCM)

DPCM implies predicting the next pixel value according to the previous pixels values, and then encoding the difference between the predicted values and actual value. This technique takes the advantage of the fact that adjacent approximation (LL) coefficients are highly correlated, which means that the difference between adjacent coefficients is typically small. Because this difference is small, it will take only a small number of bits to represent it.

In this research work, the following steps where followed to apply the DPCM (first order) on the LL-subband:

1. Compute the difference between each two adjacent coefficients in the first rows:

   $D(x,0) = I_q(x,0) - I_q(x-1,0)$, …………..………..……….…….…...(3.4)

   Where, W is the image width.

   x is the row index ($0<x\leq W/2^n-1$)

2. Compute the difference between each two adjacent coefficients in the first column:

   $D(0,y) = I_q(0,y) - I_q(0,y-1)$, ………………………………..(3.5)

   Where, H is the image height.

   y is the column index ($0<y\leq H/2^n-1$)

3. Apply the following difference equation on the remaining coefficients of the LL-subband.

$$D(x,y) = I_q(x,y) - \left(\frac{I_q(x,y-1) + I_q(x-1,y-1) + I_q(x-1,y)}{3} \right), ..(3.6)$$

   Where, $1<x\leq W/2^n-1$ and $1<y\leq H/2^n-1$, n is the wavelet level number.

This method is applied on LL (approximate) coefficients, exclusively, to reduce the number of bits needed to encode the coefficients of this subband. Algorithm (3.4) shows the implemented steps to perform the above mentioned method of DPCM.

Algorithm (3.4) Differential Pulse Code Modulation (DPCM)

Input:

```
  $B_q()$ is the array of (LL) subband from quantization
  $W$   is the image width
  $H$   is the image height
```

Output:

```
  The difference coefficients $D()$
```

Procedure:

```
  $W_m \leftarrow W : H_m \leftarrow H$
  For all i       where $1 \le i \le NoPass$
    $W_m \leftarrow (W_m+1)$ div 2
    $H_m \leftarrow (H_m+1)$ div 2
  End loop i
  $X_e \leftarrow W_m -1:\ Y_e \leftarrow H_m -1$
  For all x   where     $1 \le x \le X_e$
    $D(x,0) \leftarrow I_q(x,0)- I_q(x-1,0)$
  End loop x
  For all  y   where     $1 \le y \le Y_e$
    $D(0,y) \leftarrow I_q(0,y)- I_q(0, y-1)$
  End loop y
  For all x, y   where     $1 \le x \le X_e$  and  $1 \le y \le Y_e$
    $D(x,y) \leftarrow I_q(x, y)- (I_q(x, y-1) + I_q(x-1, y-1) + I_q(x-1, y)) / 3$
  End loop x, y
```

3.8 Mapping and S-Shift Optimizer

In this stage, all the determined quantization indices of the subbands (LH, HL, HH) and the outputs of the DPCM process are mapped to be positive, the following mapping equation had been used to convert the signed integer into positive integers:

$$C' = \begin{cases} \mathbf{2C} & \textbf{if } \mathbf{C \geq 0} \\ \mathbf{2|C|+1} & \textbf{if } \mathbf{C < 0} \end{cases} \quad ,\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.7)}$$

Where, C represents the signed integer value of the quantization index. This kind of mapping insure that all coefficients values are mapped as positive integers, and is to keep the optimal number of bits needed to use shift code the quantized coefficient as small as possible into consideration that the histogram of the coded coefficients is highly peaked around zero.

The mechanism applied to compute the optimal length (b), in bits of the shift codewords is based on scanning all possible lengths, starting from 2 bits and proceeding more till the numbers k; which represent the minimum number of bits required to represent the maximum coefficient value (L) in the set of wavelet quantization indices and DPCM output. This number (k) is considered as the length (in bits) of the second (auxiliary) codeword. The scan method was applied to test all possible values of bits that can be assigned to the first (shortest) codeword, so the length range of the first codeword is [2, k].

The scanning mechanism implies iteration over all possible numbers of bits (b) that can be assigned to the first codeword; for each possible number of bits (b) the total number of all bits (T) required encoding the quantization coefficients and DPCM output is determined by using the following equation:

$$k = \lceil \log_2(L) \rceil, \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.8)$$

$$T = b\sum_{i=0}^{L} His(i) + k \sum_{i=2^b-1}^{L} His(i), \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.9)$$

Then the value of (b) which leads to the lowest determined value of (T) is considered as the optimal length of the first codeword.

Algorithm (3.5) illustrates the implemented steps to perform the positive mapping step and to determine the optimal length of the shift codewords.

```
Algorithm (3.5) Mapping and S-Shift Optimizer
Input:
  D()  is represent the output of applying DPCM on (LL) coefficients
  $I_q()$  array of quantization indices of (LH,HL,HH) coefficients
  W  is the image width
  H  is the image height
Output:
  Number of required bits ($N_b$, $N_{bmax}$)
Procedure:
'' Do the positive mapping
    Construct an array $Img_1()$ has size W*H
    Put all contents of D() and $I_q()$ in $Img_1()$
    For all coefficients (x,y)  where  $0 \le x \le W-1$ and $0 \le y \le H-1$
      If $Img_1(x,y)>0$ then
         $Img_1(x,y) \leftarrow 2*Img_1(x,y)$

                                                        To be continue
```

```
        Else If $Img_1(x,y)<0$ then
                $Img_1(x,y) \leftarrow 2* abs(Img_1(x,y))+1$
        End if
      End loop
'''' Compute the max number
  $Max \leftarrow Img_1(0,0)$
  For all coefficients (x,y)  where  $0 \le x \le W-1$ and $0 \le y \le H-1$
    If $Max < Img_1(x,y)$ then
     $Max \leftarrow Img_1(x,y)$
  End loop
'''' Compute maximum number of required bits
    $k \leftarrow 1$
    $I \leftarrow 1$
    While $Max \ge I$
      $I \leftarrow 2 * I + 1$
      $k \leftarrow k + 1$
    Wend
''''Compute the histogram Hist() of the quantized transform coefficients
      Set $Hist(i) \leftarrow 0$  For all i  where  $0 \le i \le Max$
      For all x, y   where   $0 \le x \le W-1$ and $0 \le y \le H-1$
        $i \leftarrow Img_1(x,y)$
        $Hist(i) \leftarrow Hist(i)+1$
      End loop x, y
''''Shift coding optimizer to compute the number of required bits ($N_b$, $N_{bmax}$)
    $noc \leftarrow W * H$
    $N_{bmax} \leftarrow k$
    $MinBits \leftarrow N_{bmax} * noc$
    $N_b \leftarrow N_{bmax}$
```

To be continue

```
 For all I          where    $2 \le I \le N_{bmax} - 1$
    $M \leftarrow 2^I - 1$
    $S_m \leftarrow 0$
    For all J        where    $0 \le J \le Max$
      If $J < M$ then
        $S_m \leftarrow S_m + I * Hist(J)$
      Else
        $S_m \leftarrow S_m + (I + N_{bmax}) * Hist(J)$
      End if
    End loop J
    If $S_m < MinBits$ then
      $MinBits \leftarrow S_m$
      $N_b \leftarrow I$
    End If
End loop I
```

3.9 S-Shift Encoder

In this stage, the input data are the results of the DPCM, and the quantization coefficients of the subbands (LH, HL, HH), both results have been gathered in a single array named S(). The codewords produced by applying shift-coding on the array S() are send to the compression bit stream, which represents the compressed data file, see algorithm (3.6).

Algorithm (3.6) S-Shift Encoder

Input:

S() 2D array represent the mapped values of the subband coefficients (LL, LH,HL, HH) of record from (R,G,B).

W is the image width.

H is the image height.

N_{bmax} number of bits required to encode the largest number by using fixed length binary representation.

N_b the length (in bits) of the first shift codeword.

Output:

A set of integers whose lengths are either N_b or N_{bmax}.

Procedure:

```
Max ← $2^{Nb}$-1
TnoBits ← 0
For all x, y where   $0 \le x \le W-1$ and $0 \le y \le H-1$
    If S(x, y) < Max then
        Output the value S(x, y) as an integer has a length $N_b$ bits.
        Tnobits ← Tnobits+$N_b$
    Else
        Output the value of Max as an integer has a length $N_b$ bits.
        Output the value of (S(x, y) – Max) as an integer has a length $N_{bmax}$ bits.
        Tnobits ← Tnobits+$N_b$+$N_{bmax}$
    End if
End loop x, y
```

3.10 Video Motion Estimation

To handle the motion estimation task the standard method (OTS) was implemented in this research work. Also, some enhancements were suggested to improve the performance of OTS method, the enhanced method was called modified OTS (i.e. MOTS) method.

The proposed enhancment on the OTS algorithm is based on the fact that adjacent blocks in the image are correlated (inter-block correlation). So, if some of the blocks in the frame are coded by using the traditional steps of the OTS method. Then, the initialized motion vectors of other blocks in the frame could be estimated from averaging the motion vectors of some adjacent blocks, whose motion vectors were estimated before this step. The determination of average motion vector could be considered as the initial step followed by other steps of OTS method.

3.10.1 Modified OTS Method

The suggested modified method implies the steps of the standard OTS method, mentioned in chapter two, the mechanism of this method consist of two steps: (1) the original steps of the standard OTS method are applied on the blocks belong to the first row and column to determine their motion vectors, (2) for the remaining blocks on the rest part of the frame, the initial step will involve the determination of the initial motion vector by using the previous three adjacent blocks (top, left, top-left blocks) surrounding the tested block, the initial value of the motion vector will be taken as the averages of both Δx, Δy to the three blocks. Then, start making search for the best matched block around the initial position of the tested block which defined by the relative average vector (Δx, Δy).

3.10.2 Motion Estimation Computations

Figure (3.7) illustrates the general scheme of the motion estimation mechanism.

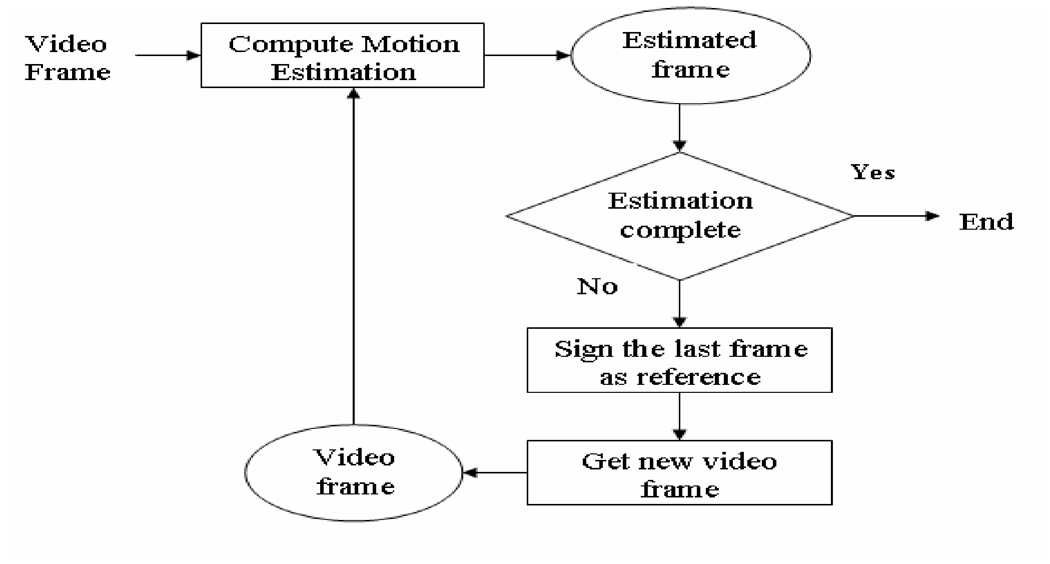


Figure (3.7) General scheme of motion estimation

The main steps of any motion estimation method are the followings:

1. As a first step, load the video frame, which is going to be coded by using ME method.
2. Then the motion vectors of the video frame are computed by using the motion estimation technique that selected by the user. The computations of the motion vectors are done relative to an anchor frame (reference frame), where the shift vectors of the blocks are determined. The system utilizes the nearest reference frame to estimate the motion vectors of the predicted frames.
3. If the motion vectors of all frames within the group (GOP) were estimated, then the systems loads the next group of frames, encode their anchor and interpolation frames by using intra-frame coding method, and then starts the motion estimation stage to encode the remaining (predicted) frames.

4. Step-3 will repeated till reaching the end of video sequence.

3.10.3 Implementation of OTS Method

The code list (3.7) illustrates the implemented steps of OTS motion estimation method, in this algorithm the blocks of the two input frames (i.e., the anchor frame and the frame wanted to be coded) are motion compensated by using OTS method. This method is simple but effective, it implies two stages (i.e., a horizontal scanning stage followed by the vertical scanning stage). During the horizontal stage the horizontal shift of the position of the block (in anchor frame) that shows minimum distortion along the horizontal axis is found. Then, the vertical shift in the position of most similar block (has minimum distortion) is found, the search would be applied on all blocks of the frame.

Algorithm (3.7) OTS Motion Estimation

```
Input:
  Video frames
  W is the image width
  H is the image height
Output:
  Estimated frame
Procedure:
  L ← block size
  $W_x$ ← W/L  :  $H_y$ ← H/L
  """searching for all blocks in the frame
  For each block (ix, iy;   where $0 \le ix \le W_x - 1$  and  $0 \le iy \le H_y - 1$)
```

To be continue

```
$x_s \leftarrow ix*L$ : $x_e \leftarrow x_s + (L-1)$
$y_s \leftarrow iy*L$ : $y_e \leftarrow y_s + (L-1)$
Compute the absolute difference (AD) for the blocks whose coordinates are $(x_s+1, y_s)$, $(x_s-1, y_s)$ in the reference frame.
    If AD$(x_s, y_s)$ is smaller than AD$(x_s+1, y_s)$ and AD$(x_s-1, y_s)$ then
        Stop searching along the horizontal direction
    Else
        If AD$(x_s+1, y_s)$ is smaller than AD$(x_s-1, y_s)$ then
            Continue searching for the smallest AD on the right hand side of the block $(x_s, y_s)$, by continually incrementing $x_s=x_s+1$ where $x_s < W-L$.
        Else
            Continue searching for the smallest AD on the left hand side of the block $(x_s, y_s)$, by continually decrementing $x_s=x_s-1$ where $x_s \geq 0$.
        End if
    End if
Compute the absolute difference (AD) for the blocks whose coordinates are $(x_s, y_s+1)$,$(x_s, y_s-1)$ in the reference frame.
    If AD$(x_s, y_s)$ is smaller than AD$(x_s, y_s+1)$ and AD$(x_s, y_s-1)$ then
        Stop searching along the vertical direction
    Else
        If AD$(x_s, y_s+1)$ is smaller than AD$(x_s, y_s-1)$ then
            Continue searching for the smallest AD on the down ward side of the block $(x_s, y_s)$, by continually incrementing $y_s=y_s+1$ where $y_s < H-L$.
```

To be continue

```
        Else
            Continue searching for the smallest AD on the up ward
            side of the block ($x_s$, $y_s$), by continually decrementing
            $y_s=y_s-1$ where $y_s \geq 0$.
        End if
    End if
End loop ix, iy
```

3.10.4 Implementation of MOTS Method

The code list (3.8) illustrates the implemented steps of MOTS method. After loading the video frame; the standard OTS method is applied on the blocks arranged as a first row and first column. Then, the OTS method is applied on the remaining blocks but the initial reference position of the matching process is assigned as the average of the motion vectors of the three previously motion compensated blocks (top, left, top-left).

Algorithm (3.8) Modified OTS Motion Estimation

Input:

```
  Video frames
  $W$   is the image width
  $H$   is the image height
```

Output:

```
  Estimated frame
```

Procedure:

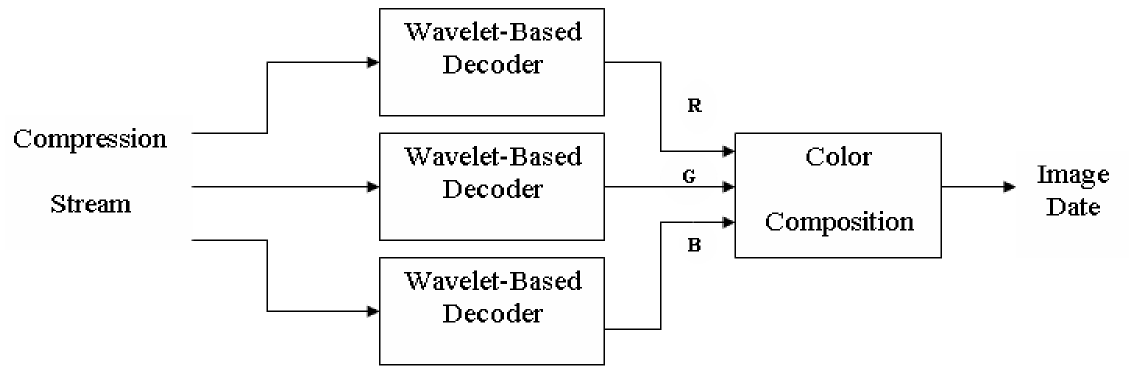
```
  $L \leftarrow$ block size
  $W_x \leftarrow W/L$
  $H_y \leftarrow H/L$
  For all blocks belong to the first row are motion compensated by using OTS method, described in (algorithm 3.5).
  For all blocks belong to the first column are motion compensated by using OTS method, described in (algorithm 3.5).
  For each block (ix, iy;   where   $1 \le ix \le W_x$ and $1 \le iy \le H_y$)
      $x_s \leftarrow ix*L$ : $x_e \leftarrow x_s + (L-1)$
      $y_s \leftarrow iy*L$ : $y_e \leftarrow y_s + (L-1)$
      $x_1 \leftarrow \Delta x$ of block $(x_s-L, y_s-L)$: $y_1 \leftarrow \Delta y$ of block $(x_s-L, y_s-L)$
      $x_2 \leftarrow \Delta x$ of block $(x_s, y_s-L)$ : $y_2 \leftarrow \Delta y$ of block $(x_s, y_s-L)$
      $x_3 \leftarrow \Delta x$ of block $(x_s-L, y_s)$ : $y_3 \leftarrow \Delta y$ of block $(x_s-L, y_s)$
      $\Delta x \leftarrow (x_1 + x_2 + x_3) / 3$
      $\Delta y \leftarrow (y_1 + y_2 + y_3) / 3$
      Compute the absolute difference (AD) for the blocks whose coordinates are $(x_s+\Delta x+1, y_s+\Delta y)$, $(x_s+\Delta x-1, y_s+\Delta y)$ in the reference frame.
        If $AD(x_s+\Delta x, y_s+\Delta y)$ is smaller than $AD(x_s+\Delta x+1, y_s+\Delta y)$ and $AD(x_s+\Delta x-1, y_s+\Delta y)$ then
              Stop searching along the horizontal direction
```

To be continue

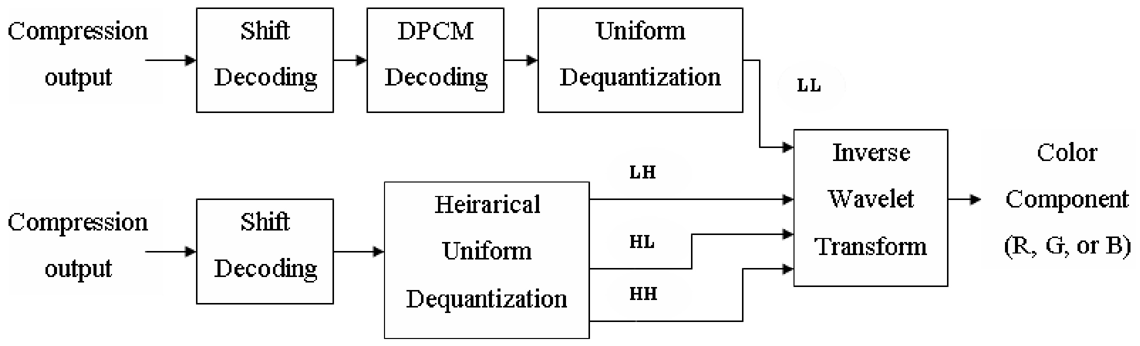
```
        Else
          If AD($x_s$+$\Delta x$+1, $y_s$+ $\Delta y$) is smaller than AD( $x_s$+ $\Delta x$ -1, $y_s$+ $\Delta y$ )
          then
              Continue searching for the smallest AD on the right hand side
              of the block ($x_s$+$\Delta x$, $y_s$+$\Delta y$), by continually increasing
              $x_s$=$x_s$+1 where $x_s$+$\Delta x$ <W-L.
          Else
              Continue searching for the smallest AD on the left hand side
              of the block ($x_s$+$\Delta x$, $y_s$+$\Delta y$), by continually decrementing
              $x_s$=$x_s$-1 where $x_s$+$\Delta x$ $\geq$0.
          End if
        End if
      Compute the absolute difference (AD) for the block whose
      coordinates are (xs+$\Delta x$, ys+ $\Delta y$+1), (xs+ $\Delta x$, ys+ $\Delta y$-1) in the
      reference frame
        If AD($x_s$+$\Delta x$, $y_s$+$\Delta y$) is smaller than AD($x_s$+$\Delta x$, $y_s$+$\Delta y$+1) and
        AD($x_s$+$\Delta x$, $y_s$+$\Delta y$-1) then
              Stop searching along the vertical direction
        Else
          If AD ($x_s$+ $\Delta x$, $y_s$+ $\Delta y$+1) is smaller than AD ( $x_s$+ $\Delta x$, $y_s$+ $\Delta y$-1)
          then
              Continue searching for the smallest AD on the down ward
              side of the block ($x_s$+$\Delta x$, $y_s$+$\Delta y$), by continually increasing
              ys=ys+1 where ys+$\Delta y$<H-L.
          Else
              Continue searching for the smallest AD on the up ward side of
              the block ($x_s$+$\Delta x$, $y_s$+$\Delta y$), by continually decrementing
              ys=ys-1 where ys+$\Delta y$ $\geq$ 0.
          End if
        End if
    End loop ix, iy
```

3.11 Decoding of Anchor Frames

Figure (3.8) illustrates the decoding scheme for anchors frame.



a. General structure



b. Wavelet-Based Decoder

Figure (3.8) The decompression scheme of anchor frame

The implemented decoder consists of the following steps:

1. Loading the compressed data as a one dimensional array of bits.
2. Decoding the compressed data of LL coefficients: first the s-shift decoding is applied to get the DPCM codewords, then decoding the DPCM codewords to get the quantization indices of (LL) coefficients, and then dequantize the quantization indices to get the quantized values of LL-coefficients.
3. Decoding the compressed data of LH, HL, HH coefficients: first the s-shift decoding is applied to get the quantization indices, and then

apply the hieratical dequantization process to get the quantization coefficients.

4. Inverse wavelet transform: apply the inverse wavelet transform on the constructed LL, LH, HL, HH coefficients to produce the reconstructed image.

3.11.1 Shift Decoder

The code list (3.9) illustrates the steps of implemented shift decoder.

```
Algorithm (3.9) S-Shift Decoder
Input:
  Ww, Hh is the subband width and height, respectively.
  NoPass is the number of wavelet levels.
  A()  is the array of shift-codewords.
  $N_b$  is Number of bits of the first (short) shift codeword.
  $N_s$ is number of bits of the second (long) shift codeword.
Output:
  NA() the shift decoder output (quantization indices and DPCM codewords)
Procedure:
  Max ← $2^{Nb}$ -1
  For all x, y where   $0 \le x \le Ww-1$ and $0 \le y \le Hh-1$
      A ← GetBits($N_b$)
      If abs(A)< Max then
        NA(x, y) ← A
      Else
        NA(x, y) ← A+ GetBits($N_s$)
    End if
  End loop x, y
```

3.11.2 DPCM Decoding

The code list (3.10) illustrates the steps of the implemented DPCM decoder.

Algorithm (3.10) Differential Pulse Code Modulation Decoding

```
Input:
  NA() is the array of (LL) subband from S-Shift decoder.
  W is the image width.
  H is the image height.
  NoPass is the number of wavelet levels.
Output:
  Decoded array D()
Procedure:
  $W_m \leftarrow W$: $H_m \leftarrow H$
  For all i    where  $1 \le i \le NoPass$
      $W_m \leftarrow (W_m+1)$ div 2
      $H_m \leftarrow (H_m+1)$ div 2
  End loop i
  $X_e \leftarrow W_m - 1$ : $Y_e \leftarrow H_m - 1$
  $D(0,0) \leftarrow NA(0,0)$
  For all  x  where   $1 \le x \le X_e$
      $D(x, 0) \leftarrow NA(x, 0) + D(x-1, 0)$
  End loop x
  For all  y  where   $1 \le y \le Y_e$
      $D(0,y) \leftarrow NA(0,y) + D(0, y-1)$
  End loop y
  For all  x, y  where   $1 \le x \le X_e$ and $1 \le y \le Y_e$
      $D(x,y) \leftarrow (D(x, y-1) + D(x-1, y-1) + D(x-1, y)) / 3) + NA(x,y)$
  End loop x, y
```

3.11.3 Image Dequantization

The code list (3.11) illustrates the implemented steps to perform the dequantization tasks.

Algorithm (3.11) Image Dequantization

Input:

```
  D()  is an array of (LL) subband from DPCM
  NA() is an array of (LH,HL,HH) subband from S-shift decoder
  W  is the image width
  H  is the image height
  $Q_0$  is quantization step for (LL) coefficient
  Q  is  quantization step for (LH,HL,HH) coefficients
  $\alpha, \beta$  is quality factors (or decay factors)
```

Output:

```
  Dimg() is the dequanted wavelet coefficients.
```

Procedure:

```
  $W_m \leftarrow W$: $H_m \leftarrow H$
  $W_{m1} \leftarrow W$: $H_{m1} \leftarrow H$
  For all i      where   $1 \le i \le NoPass$
    $W_m \leftarrow (W_m+1)$ div 2
    $H_m \leftarrow (H_m+1)$ div 2
  End loop i
''''for (LL) coefficient
  $X_s \leftarrow 0$:  $X_e \leftarrow W_m -1$:  $Y_s \leftarrow 0$: $Y_e \leftarrow H_m -1$
  For all x, y   where      $X_s \le x \le X_e$  and  $Y_s \le y \le Y_e$
      $Dimg(x, y) \leftarrow D(x, y) * Q_0$
  End loop x, y
```

To be continue

```
''''for (LH, HL, HH) coefficient
  $W_m \leftarrow W$:  $H_m \leftarrow H$
  For all  j  where  $1 \le j \le NoPass$
      $W_{m1} \leftarrow W_m$:  $H_{m1} \leftarrow H_m$
      $W_m \leftarrow (W_m+1)$ div 2
      $H_m \leftarrow (H_m+1)$ div 2
      $X_s \leftarrow W_m$:  $X_e \leftarrow W_{m1}-1$:  $Y_s \leftarrow 0$: $Y_e \leftarrow H_m -1$
      $Q_{step} \leftarrow Q * \alpha^{j-1}$
      For all  x, y  where  $X_s \le x \le X_e$  and  $Y_s \le y \le Y_e$
          Dimg(x, y) $\leftarrow$ NA(x, y)* $Q_{step}$
      End loop x, y
      $X_s \leftarrow 0$:  $X_e \leftarrow W_m -1$:  $Y_s \leftarrow H_m$: $Y_e \leftarrow H_{m1}-1$
      For all  x, y  where  $X_s \le x \le X_e$  and  $Y_s \le y \le Y_e$
          Dimg(x, y) $\leftarrow$ NA(x, y) * $Q_{step}$
      End loop x, y
      $X_s \leftarrow W_m$:  $X_e \leftarrow W_{m1} -1$:  $Y_s \leftarrow H_m$: $Y_e \leftarrow H_{m1}-1$
      $Q_{step} \leftarrow Q_{step} * \beta$
      For all  x, y  where  $X_s \le x \le X_e$  and  $Y_s \le y \le Y_e$
          Dimg(x, y) $\leftarrow$ NA(x, y) * $Q_{step}$
      End loop x, y
  End loop j
```

3.11.4 TAP9/7 Wavelet Transform Inverse

The code list (3.12) and (3.13) illustrates the steps of the implemented inverse wavelet transform.

Algorithm (3.12) Two Dimensional Wavelet Transform Encoder

Input:

Dimg() is the wavelet coefficients(LL, LH, HL, HH)

W is the image width

H is the image height

NoPass is the number of wavelet transform levels or passes

Output:

*Frm() is Image frame of W*H*

Procedure:

```
For all k   where   $1 \le k \le NoPass$
    $W_w \leftarrow W$: $H_h \leftarrow H$
    For I=1 to NoPass-k
        $W_w \leftarrow (W_w+1)$ div 2
        $H_h \leftarrow (H_h+1)$ div 2
    End loop I
      For all   I=each column in Dimg  (where   $0 \le I \le W_w-1$)
          Move the contents of the $I^{th}$ column of Timg in array A
          Call inverse wavelet TAP9/7(A, Hh, B)
          Move the contents of the array B in the $I^{th}$ column of Frm
      End loop I
      For J=each row in Dimg array  (where   $0 \le J \le H_h-1$)
          Move the contents of the $J^{th}$ row of Dimg in the array A
          Call inverse wavelet TAP9/7(A, Ww, B)
          Move the contents of the array B in the $J^{th}$ column of Frm
      End loop J
End loop k
```

Algorithm (3.13) Inverse Wavelet TAP9/7

Input:

A() is the input array (wavelet coefficients)

N is the number of elements

Output:

B() is the output array of one dimension wavelet transform

Procedure:

```
$N_{even} \leftarrow (N+1)/2$: $N_{odd} \leftarrow N - N_{even}$
$N_m \leftarrow N - 1$: $N_{p1} \leftarrow N + 1$:  $N_{p2} \leftarrow N + 2$
$Aa \leftarrow 1/ 1.230174105$
$i \leftarrow 0 : j \leftarrow 1$
For all k where    $0 \le k \le N_{even}-1$
    $C(i) \leftarrow 1.230174105 * A(k)$
    $i \leftarrow i+2$
End loop k
For all k where    $0 \le k \le N_{odd}-1$
    $C(j) \leftarrow Aa * A(N_{even}+k)$
    $j \leftarrow j+2$
End loop k
For all k where    $1 \le k \le 4$
    $C(N_m+ k) \leftarrow C(N_m -k)$
    $C(-k) \leftarrow C(k)$
End loop k
For all k   where    $(k= -2, 0, 2, 4, \ldots\ldots, N_{p2})$
    $C(k) \leftarrow C(k)- 0.4435068522 * (C(k-1) + C(k+1))$
End loop k
```

To be continue

```
For all k   where   $(k= -1, 1, 3, 5, \ldots, N_{p1})$
    $C(k) \leftarrow C(k) - 0.8829110762 * (C(k-1) + C(k+1))$
End loop k
For all k   where   $(k=0, 2, 4, 6, \ldots, N)$
    $C(k) \leftarrow C(k) - 0.05298011854 * (C(k-1) + C(k+1))$
End loop k
For all k   where   $(k=1, 3, 5, 7, \ldots, N_m)$
    $C(k) \leftarrow C(k) - 1.586134342 * (C(k-1) + C(k+1))$
End loop k
For all k   where   $(k=0, 1, 2, 3, 4, \ldots, N_m)$
    $B(k) = C(k)$
End loop k
```

3.12 Motion Estimation with Decoding

Figure (3.9) illustrates the scheme of the constructed inter-frame decoder (motion compensation) by using the estimated motion vectors.

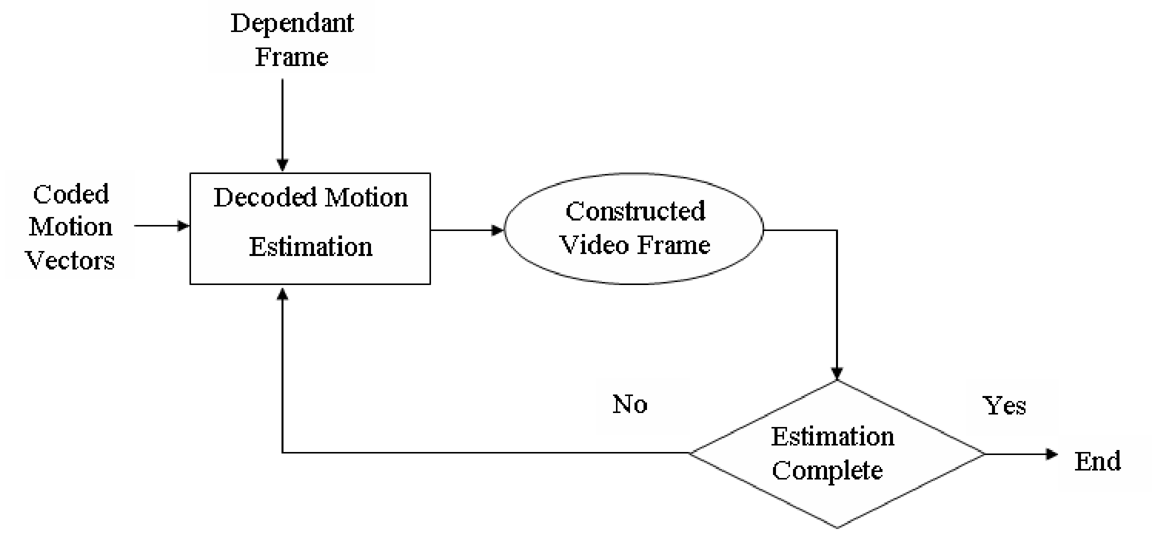


Figure (3.9) Frame construction using its motion vectors

The code list (3.14) illustrates the step of the implemented algorithm to construct the inter-frames by using the corresponding motion vectors set.

Algorithm (3.14) Motion Estimation Decoding

Input:

*Estimated frames (EF) of H*W, and Dependant frames (DF) of blocks of x, y coordinates.*

W is the image width

H is the image height

shftx(), shfty() is an array represented the shifted values with respect to x and y

Output:

*Transformed video frame H * W*

Procedure:

```
Set L equal to the block size
$W_x \leftarrow W/L$:  $H_y \leftarrow H/L$
blk_no=0
For all ix, iy where    $0 \le ix \le W_x-1$ and  $0 \le iy \le H_y-1$
  $x_s \leftarrow ix*L$ : $x_e \leftarrow x_s + (L-1)$
  $y_s \leftarrow iy*L$ : $y_e \leftarrow y_s + (L-1)$
  If the motion Flag=1 then
    For all y1, x1  where  $y_s \le y1 \le y_e$  and  $x_s \le x1 \le x_e$
       TF(ix+x1,iy+y1) ← DF(ix+x1–shftx(blk_no),iy+y1-shfty(blk_no))
    End loop y1, x1
  Else
      For all y1, x1 where   $y_s \le y1 \le y_e$  and  $x_s \le x1 \le x_e$
         TF(ix+x1,iy+y1) ← EF(ix+x1,iy+y1)
      End loop y1, x1
    End if
  blk_no ← blk_no+1
End loop ix, iy
```

Chapter Four
Tests and Results

4.1 Introduction

This chapter is devoted to present the results of the conducted tests to study the compression performance of the suggested video compression schemes. Some of the well known fidelity measures (i.e. MSE, PSNR) have been used to assess the quality of the reconstructed video frames.

The effects of some involved coding parameters on the performance of the two searching methods of motion estimation (OTS, and MOTS), and on the compression method of the anchor frames and interpolation frames have been investigated.

The developed systems have been established by using Visual Basic (version 6.0) programming language under windows XP operating system. The tests have been conducted by using personal computer (processor Pentium 4, 2.40 GHz), dual cash memory.

4.2 Video Test Material

Two different video sequences have been taken as test samples, each video sequence consists of different numbers of frames and its specifications are listed in table (4.1). Figure (4.1) and (4.2) show some selected frames extracted from the two video sequences.

Table (4.1) Video Properties

Samples Number	No. of Frames	Width	Height
1	126	352	288
2	745	352	288

Frame 0

Frame 3

Frame 11

Frame 18

Frame 20

Frame 21

Frame 22

Frame 29

Frame 33

Frame 36

Figure (4.1) Some of the original frames for first video sequence.

Frame 0

Frame 1

Frame 11

Frame 12

Frame 21

Frame 22

Frame 23

Frame 29

Frame 33 Frame 36

Figure (4.2) Some of the original frames for second video sequence.

4.3 Performance Parameters

A lot of key parameters have been utilized in the literature to describe the performance of various compression methods. In this research the fidelity criteria (MSE and PSNR) in addition to the compression ratio have been used to describe the performance of the proposed video compression scheme at different coding conditions.

4.3.1 Objective Fidelity Criteria

This kind of criteria borrowed from digital signal processing and information theory, they provide equations that can be used to describe the amount of error in the reconstructed (decompressed) frame. Beside to their common usage, the objective criteria are not necessarily correlated with the subjective perception of frame quality. However, they are useful as relative measures for making comparison between different versions of the same frame. The commonly used objective measures are the Mean-Square-Error (MSE), the Peak-Signal-to-Noise ratio (PSNR). The value of MSE is determined by using the following equation **[**Soo01]:

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left(O(i,j) - R(i,j)\right)^2 \text{ ,……….........……………..(4.1)}$$

The smaller value of the MSE means the compressed frame is much closer to the original frame. As an alternative fidelity measure is the PSNR, it considers the decompressed frame to be the "signal" and the difference from the original as the "noise". The PSNR is defined as **[**Soo01]:

$$PSNR = 10\log_{10}\left(\frac{(L-1)^2}{MSE}\right) \text{ ,……….......…………………… (4.2)}$$

Where L is the number of gray levels (e.g., for 8 bits per pixel L=256).

4.3.2 Compression Compactness

Although there are different measures to describe the achieved reduction in data size due to compression (like compression ratio, compression factor, and bit rate), in this research work the compression ratio (C_r) had been adopted.

Compression ratio refers to the degree of video file size (data) reduction due to compression process. This measure is determined as the ratio between the sizes of the original uncompressed video file to the size of the overall compressed data file [Umb98].

$$C_r = \frac{uncompression\ file\ size}{compression\ file\ size}, \quad (4.3)$$

The (C_r) parameter is an indicator for the compactness ability of the compression process.

4.4 Testing Strategy

The testing operations have been implemented on the above mentioned two video samples. In the first established coding scheme two of the video frames are coded as anchor frames (AF), and the remaining frames are treated as predictive frames (PF). While in the second established coding scheme two video frames are treated as anchor frames (AF), other frames as interpolation frames (IF), and the remaining (in-between frames) as predictive frames (PF).

For the anchor frames (AF) and the interpolation frames (IF), the effects of involved intra-frame coding parameters (like, the number of wavelet transform levels, the quantization parameters) on the compression performance has been studied. While for the predicted frames (PF), the effects of the involved interframe parameters (like, the size GOP, Minimum MSE) have been studied, the range of PSNR is taken equal to 33 for intra-frame compression.

4.5 Intra-Frame Compression Performance Tests

1. Anchor Frames Compression

The listed tables in this section illustrates the compression results of the first anchor frame in the first video sequences, in these tables the values of MSE, PSNR, and compression ratio (C_r) are listed. It is obvious that their values are significantly affected by the number of wavelet passes (NoPass), and the quantization parameters value:

1. Table (4.2) shows the effects of the number of wavelet passes on the compression performance.

Table (4.2) The effect of NoPass parameters (where, Q=40; α=0.6; β=1.5).

NoPass	Q_L	MSE	PSNR	C_r	Time in sec.
1	2	13.08	36.96	3.86	2.86
2		22.72	34.57	5.52	2.97
3		28.26	33.62	6.17	3.21
1	3	13.38	36.87	4.24	2.76
2		23.02	34.51	5.73	2.90
3		28.54	33.58	6.22	3.11

2. Table (4.3) illustrates the effects of the quantization step (Q_L) on the compression performance, where the number of wavelet passes was taken (NoPass=3).

Table (4.3) The effect of Q_L (where, Q=40; α=0.6; β=1.5).

Q_L	MSE	PSNR	C_r
1	28.03	33.65	6.09
2	28.26	33.62	6.17
3	28.54	33.58	6.22
4	29.02	33.50	6.26
5	29.58	33.42	6.28
6	30.30	33.32	6.30
7	30.84	33.24	6.32
8	31.55	33.14	6.35

3. Table (4.4) illustrates the effects of the quantization parameters (Q) on the compression performance, where the number of wavelet passes was taken (NoPass=3).

Table (4.4) The effect of Q (where, Q_L=3; α=0.6; β=1.5).

Q	MSE	PSNR	C_r
5	1.76	45.67	2.95
10	4.07	42.04	3.93
20	10.68	37.85	5.10
30	19.05	35.33	5.78
40	28.54	33.58	6.22
50	38.14	32.32	7.56

4. Table (4.5) shows the effect of increasing the quantization parameter (α), where the number of wavelet passes was taken (NoPass=3).

Table (4.5) The effect of α (where, Q_L=3; Q=40; β=1.5).

α	MSE	PSNR	C_r
0.1	13.98	36.67	4.30
0.2	15.12	36.33	4.86
0.3	16.96	35.84	5.33
0.4	19.61	35.21	5.74
0.5	23.49	34.42	5.98
0.6	28.54	33.58	6.22
0.7	34.64	32.74	6.37
0.8	43.01	31.80	6.59
0.9	52.69	30.91	6.69

5. Table (4.6) shows the effects of the quantization parameter (β) on the compression performance, where the number of wavelet passes was taken (NoPass=3).

Table (4.6) The effect of β (where, Q_L=3; Q=40; α=0.6).

β	MSE	PSNR	C_r
1.1	28.49	33.58	6.22
1.3	28.53	33.58	6.22
1.5	28.54	33.58	6.22
1.7	28.55	33.57	7.14
1.9	28.55	33.57	7.14

Table (4.7) illustrates some samples compression results of the anchor frames belong to the two video sequences. These results, also, indicate that the compression performance parameters (MSE, PSNR, C_r) are affected by the number of wavelet passes (NoPass), and the values of the quantization parameters.

Table (4.7) Test results of anchor frames (where, Q_L=3; Q=40; α=0.6; β=1.5).

	Anchor No.	Frame No.	NoPass	MSE	PSNR	C_r
First video sequences	First	0	1	13.38	36.87	4.24
		0	2	23.02	34.51	5.73
		0	3	28.54	33.58	6.22
	Second	11	3	28.01	33.66	7.18
		21	3	27.93	33.67	6.23
		33	3	28.68	33.56	6.24
		36	3	28.67	33.56	6.23
Second video sequences	First	0	1	15.44	36.24	4.01
		0	2	24.91	34.17	5.56
		0	3	30.55	33.28	6.08
	Second	11	3	30.89	33.23	6.19
		21	3	30.11	33.34	6.20
		33	3	31.42	33.16	6.16
		36	3	30.06	33.35	6.18

Figure (4.3) shows some samples of the reconstructed anchor frames belong to the first video sequence, and figure (4.4) shows samples of the reconstructed anchor frames belong to the second video sequence.

Frame 0 first anchor
NoPass= 3
Cr=6.22
MSE=28.54
PSNR=33.58

Frame 11 second anchor
NoPass=3
Cr=7.18
MSE=28.01
PSNR=33.66

Frame 21 second anchor
NoPass=3
Cr=6.23
MSE=27.93
PSNR=33.67

Frame 33 second anchor
NoPass=3
Cr=6.24
MSE=28.68
PSNR=33.56

Frame 36 second anchor
NoPass=3
Cr=6.23
MSE=28.67
PSNR=33.56

Figure (4.3) Some samples of the reconstructed anchor frames of the first video sequence.

Frame 0 first anchor
NoPass = 3
Cr=6.08
MSE=30.55
PSNR=33.28

Frame 11 second anchor
NoPass=3
Cr=6.19
MSE=30.89
PSNR=33.23

Frame 21 second anchor
NoPass=3
Cr=6.20
MSE=30.11
PSNR=33.34

Frame 33 second anchor
NoPass=3
Cr=6.16
MSE=31.42
PSNR=33.16

Frame 36 second anchor
NoPass=3
Cr=6.18
MSE=30.06
PSNR=33.35

Figure (4.4) Some samples of the reconstructed anchor frames of the second video sequence.

2. Compression of Interpolation Frames

The tables given in this section illustrates the results of the compression of some interpolated frames belong to two video sequences. In these tables the values of MSE, PSNR, and C_r are listed. The tested parameters are significantly affected by the number of wavelet passes, quantization parameters, and the number of interpolated frames:

1. Tables (4.8) shows the results of compression performance parameters (MSE, PSNR, C_r), where the number of wavelet passes was taken (NoPass=3) and two interpolated frames were taken.

Table (4.8) The effect of GOP size (where, Q_L=3; Q=40; α=0.6; β=1.5).

	GOP Size	IF Frame No.	MSE	PSNR	C_r
First video sequences	22	7	27.92	33.67	6.22
		14	28.02	33.66	6.22
	34	11	28.01	33.66	7.18
		22	27.65	33.71	6.23
Second video sequences	22	7	30.41	33.30	6.20
		14	30.69	33.26	6.11
	34	11	30.89	33.23	6.19
		22	29.69	33.40	6.20

Figures (4.5 and 4.6) shows some of the reconstructed interpolated frames, where the size of GOP was taken (22 and 34) frame and 2 interpolated frames were taken.

GOP=22, IF1 (Frame 7)
NoPass=3
Cr=6.22
MSE=27.92
PSNR=33.67

GOP=22, IF2 (Frame 14)
NoPass=3
Cr=6.22
MSE=28.02
PSNR=33.66

GOP=34, IF1 (Frame 11)
NoPass=3
Cr=7.18
MSE=28.01
PSNR=33.66

GOP=34, IF2 (Frame 22)
NoPass=3
Cr=6.23
MSE=27.65
PSNR=33.71

Figure (4.5) Some samples of the reconstructed interpolation frames belong to the first video sequence.

GOP=22, IF1 (Frame 7)
NoPass=3
Cr=6.20
MSE=30.41
PSNR=33.30

GOP=22, IF2 (Frame 14)
NoPass=3
Cr=6.11
MSE=30.69
PSNR=33.26

GOP=34, IF1 (Frame 11)
NoPass=3
Cr=6.19
MSE=30.89
PSNR=33.23

GOP=34, IF2 (Frame 22)
NoPass=3
Cr=6.20
MSE=29.69
PSNR=33.40

Figure (4.6) Some samples of the reconstructed interpolation frames belong to the second video sequence.

2. Table (4.9) shows the compression results that affected by the number of interpolated frames, where the number of wavelet passes was taken (NoPass=3), and (GOP size=37).

Table (4.9) The effect of number of interpolated frames parameter (where, Q_L=3; Q=40; α=0.6; β=1.5).

	No. of IF	IF Frame No.	MSE	PSNR	C_r
First video sequences	1	18	28.38	33.60	6.20
	2	12	28.30	33.61	6.27
		24	28.58	33.57	6.17
	3	9	28.32	33.61	6.63
		18	28.38	33.60	6.20
		27	28.48	33.59	6.23
Second video sequences	1	18	29.83	33.38	6.20
	2	12	31.77	33.11	6.16
		24	30.43	33.30	6.14
	3	9	31.64	33.13	6.17
		18	29.83	33.38	6.20
		27	30.74	33.25	6.03

Figures (4.7 and 4.8) shows some samples of the reconstructed interpolated frames, where the size of GOP was taken 37 frames. Three tests have been conducted, in the first test one interpolated frame were taken, in the second test two interpolation frames were taken, while in the third test three interpolation frames were taken from the two video sequences.

GOP=37, IF1 (Frame 18)
NoPass=3
Cr=6.20
MSE=28.38
PSNR=33.60

GOP=37, IF1 (Frame 12)
NoPass=3
Cr=6.27
MSE=28.30
PSNR=33.57

GOP=37, IF2 (Frame 24)
NoPass=3
Cr=6.17
MSE=28.58
PSNR=33.57

GOP=37, IF1 (Frame 9)
NoPass=3
Cr=6.63
MSE=28.32
PSNR=33.61

GOP=37, IF2 (Frame 18)
NoPass=3
Cr=6.20
MSE=28.38
PSNR=33.60

GOP=37, IF3 (Frame 27)
NoPass=3
Cr=6.23
MSE=28.48
PSNR=33.59

Figure (4.7) Some samples of the reconstructed interpolation frames from the first video sequence.

GOP=37, IF1 (Frame 18)
NoPass=3
Cr=6.20
MSE=29.83
PSNR=33.38

GOP=37, IF1 (Frame 12)
NoPass=3
Cr=6.16
MSE=31.77
PSNR=33.11

GOP=37, IF2 (Frame 24)
NoPass=3
Cr=6.14
MSE=30.43
PSNR=33.30

GOP=37, IF1 (Frame 9)
NoPass=3
Cr=6.17
MSE=31.64
PSNR=33.13

GOP=37, IF2 (Frame 18)
NoPass=3
Cr=6.20
MSE=29.83
PSNR=33.38

GOP=37, IF3 (Frame 27)
NoPass=3
Cr=6.03
MSE=30.74
PSNR=33.25

Figure (4.8) Some samples of the reconstructed interpolation frames from the second video sequence.

4.6 Inter-Frame Compression Performance Tests

Some of the conducted tests have been directed to make comparisons between performance of the motion estimation methods (OTS) and (MOTS). The effects of some involved compression parameters were taken into account:

1. Table (4.10) shows the effect of two motion estimation methods on the performance parameters, the value of all involved coding parameters are kept same.

Table (4.10) The effect of ME method without IF Frames (where, GOP size=12 ; Threshold=150 ; Block size=8×8).

	Method type	MSE	PSNR	C_r	Time in sec.
first video sequences	OTS	59.54	30.38	57.90	4.86
	MOTS	39.23	32.20	58.52	4.01
second video sequences	OTS	99.54	28.15	53.44	5.13
	MOTS	25.70	34.03	54.29	4.08

Figure (4.9) shows two samples of the reconstructed video frames which have been motion compensated by OTS method. While figure (4.10) shows the corresponding two reconstructed video frames which have been motion compensated by MOTS method.

Frame 3 from 12 frames
MSE=37.31
PSNR=32.41

Frame 1 from 12 frames
MSE=56.60
PSNR=30.60

Figure (4.9) Some of the reconstructed video frames by using OTS method.

Frame 3 from 12 frames
MSE=37.26
PSNR=32.42

Frame 1 from 12 frames
MSE=7.37
PSNR=39.46

Figure (4.10) Some of the reconstructed video frames by using MOTS method.

2. Tables (4.11 and 4.12) illustrate the effects of GOP size parameter on the performance of the two motion estimation methods.

Table (4.11) The effect of GOP size parameters on performance of OTS method (where, Th=150; No. of IF=2; Block size=8×8).

	GOP Size	MSE	PSNR	C_r	Time in sec.
First video sequences	22	45.29	31.57	57.90	7.00
	34	53.97	30.81	57.95	10.53
Second video sequences	22	42.51	31.85	57.61	7.06
	34	61.89	30.21	57.66	10.59

Table (4.12) The effect of GOP size parameters on performance of MOTS method (where, Th=150; No. of IF=2; Block size=8×8).

	GOP Size	MSE	PSNR	C_r	Time in sec.
First video sequences	22	19.36	35.26	58.25	6.47
	34	28.07	33.56	58.52	9.87
Second video sequences	22	12.01	37.34	57.70	6.46
	34	47.41	31.37	57.75	9.88

Figure (4.11) shows some of the reconstructed video frames when different values of GOP size are taken, they have been motion compensated by OTS method. While figure (4.12) shows the corresponding reconstructed video frames which have been motion compensated by MOTS method.

Frame 18 from 22 frames
MSE=8.87
PSNR=38.65

Frame 20 from 34 frames
MSE=8.23
PSNR=38.98

Frame 12 from 22 frames
MSE=22.40
PSNR=34.63

Frame 23 from 34 frames
MSE=44.52
PSNR=31.65

Figure (4.11) Some of the reconstructed video frames at different values of GOP size (using OTS method).

Frame 18 from 22 frames
MSE=7.46
PSNR=39.41

Frame 20 from 34 frames
MSE=6.28
PSNR=40.15

Frame 12 from 22 frames
MSE=8.25
PSNR=38.97

Frame 23 from 34 frames
MSE=38.63
PSNR=32.26

Figure (4.12) Some of the reconstructed video frames at different values of GOP size (using MOTS method).

3. Tables (4.13 and 4.14) illustrate the effects of numbers of interpolated frames parameters on the performance of two motion estimation methods (OTS and MOTS).

Table (4.13) The effect of IF number parameters on performance of OTS method (where, GOP size=37; Th=150; Block size=8×8).

	No. of IF	MSE	PSNR	C_r	Time in sec.
First video sequences	0	250.71	24.14	53.58	11.01
	1	145.71	26.50	55.38	11.10
	2	86.85	28.74	55.22	11.23
	3	101.41	28.07	57.40	11.45
Second video sequences	0	181.89	25.53	49.58	11.00
	1	104.57	27.94	54.12	11.10
	2	69.64	29.70	55.27	11.21
	3	49.64	31.17	58.10	11.39

Table (4.14) The effect of IF number parameters on performance of MOTS method (where, GOP size=37; Th=150; Block size=8×8).

	No. of IF	MSE	PSNR	C_r	Time in sec.
First video sequences	0	174.20	25.72	55.53	10.05
	1	86.33	28.77	56.59	10.15
	2	43.71	31.73	55.85	10.30
	3	62.87	30.15	58.45	10.50
Second video sequences	0	136.51	26.80	52.66	10.04
	1	77.65	29.23	56.05	10.14
	2	52.84	30.90	56.71	10.28
	3	37.57	32.38	59.25	10.45

Figure (4.13 and 4.14) show some of the reconstructed video frames at different values of the parameter number of interpolated frames, they have been motion compensated by OTS and MOTS methods, respectively.

Frame 3 from 37 frames
No. of IF=0
MSE=37.31
PSNR=32.41

Frame 11 from 37 frames
No. of IF=2
MSE=18.20
PSNR=35.53

Frame 21 from 37 frames
No. of IF=1
MSE=63.98
PSNR=30.07

Frame 23 from 37 frames
No. of IF=3
MSE=24.46
PSNR=34.25

Figure (4.13) Some of the reconstructed video frames at different values of the no. of IF (using OTS method).

Frame 3 from 37 frames
No. of IF=0
MSE=37.26
PSNR=32.42

Frame 11 from 37 frames
No. of IF=2
MSE=6.71
PSNR=39.86

Frame 21 from 37 frames
No. of IF=1
MSE=54.08
PSNR=30.80

Frame 23 from 37 frames
No. of IF=3
MSE=19.86
PSNR=35.15

Figure (4.14) Some of the reconstructed video frames at different values of the no. of IF (using MOTS method).

4. Tables (4.15 and 4.16) show the effect of changing threshold values on the performance parameters of two motion estimation methods.

Table (4.15) The effect of threshold values (for OTS method), (where, GOP size=34; No. of IF=2; Block size=8×8).

	Thresholds	MSE	PSNR	C_r	Time in sec.
First video sequences	10	40.59	32.05	54.16	10.12
	15	42.84	31.81	55.26	10.24
	50	53.97	30.80	57.95	10.30
	75	53.97	30.80	57.95	10.41
	150	53.97	30.80	57.95	10.53
Second video sequences	10	19.23	35.29	50.33	10.11
	15	20.08	35.10	51.34	10.29
	50	60.94	30.28	57.63	10.35
	75	61.89	30.21	57.66	10.41
	150	61.89	30.21	57.66	10.59

Table (4.16) The effect of threshold values (for MOTS method), (where, GOP size=34; No. of IF=2; Block size=8×8).

	Thresholds	MSE	PSNR	C_r	Time in sec.
First video sequences	10	33.06	32.94	59.10	9.21
	15	31.50	33.15	58.45	9.35
	50	28.07	33.65	58.42	9.39
	75	28.07	33.65	58.42	9.67
	150	28.07	33.65	58.42	9.87
Second video sequences	10	46.54	31.45	59.76	9.19
	15	48.34	31.29	58.96	9.30
	50	17.46	35.71	57.75	9.57
	75	17.46	35.71	57.75	9.65
	150	17.46	35.71	57.75	9.88

Figure (4.15) shows some sample of the reconstructed video frames constructed at different threshold values, they have been motion compensated by OTS method. Figure (4.16) shows the corresponding reconstructed video frames which have been motion compensated by MOTS method.

Frame 3 from 34 frames
Threshold value=15
MSE=36.44
PSNR=32.41

Frame 18 from 34 frames
Threshold value=75
MSE=34.38
PSNR=32.77

Frame 21 from 34 frames
Threshold value=50
MSE=8.78
PSNR=38.70

Frame 23 from 34 frames
Threshold value=150
MSE=44.52
PSNR=31.65

Figure (4.15) Some of the reconstructed video frames at different threshold values (using OTS method).

Frame 3 from 34 frames
Threshold value=15
MSE=37.89
PSNR=32.35

Frame 18 from 34 frames
Threshold value=75
MSE=7.90
PSNR=39.15

Frame 21 from 34 frames
Threshold value=50
MSE=3.90
PSNR=42.22

Frame 23 from 34 frames
Threshold value=150
MSE=10.79
PSNR=37.80

Figure (4.16) Some of the reconstructed video frames at different threshold values (using MOTS method).

4.7 Overall System Performance Tests

In this section the overall performance results of the two established video compression schemes are given. In other words, the results of intraframe compression (AF and IF) and interframe compression (PF by using the two ME methods OTS, MOTS) are presented, the test results have been listed in tables (4.17 and 4.18) in each table the average values of MSE, PSNR, and C_r of all video frames belong to the first video sequence are listed. The results indicate that the overall results are significantly affected by size of GOP, threshold values (Th), number of wavelet passes, number of interpolation frames, and the value of quantization parameters, as shown in tables (4.17 and 4.18).

Table (4.17) The overall test results of the video compression scheme based on using OTS method (for the first video sequence)

GOP size	No. of AF	No. of IF	No. of PF	Th	NoPass	Q_L	Q	α	β	MSE	PSNR	C_r
12	2	0	10	150	3	3	40	0.6	1.5	38.70	32.54	64.60
22	2	2	18	150	3	3	40	0.6	1.5	31.54	33.23	70.34
34	2	2	30	150	3	3	40	0.6	1.5	33.37	33.06	70.89
37	2	0	35	150	3	3	40	0.6	1.5	102.64	30.43	59.81
37	2	1	34	150	3	3	40	0.6	1.5	57.83	31.81	67.81
37	2	2	33	150	3	3	40	0.6	1.5	33.49	32.61	67.67
37	2	3	32	150	3	3	40	0.6	1.5	40.63	32.67	69.98
34	2	2	30	15	3	3	40	0.6	1.5	31.14	33.26	68.20
34	2	2	30	50	3	3	40	0.6	1.5	33.37	33.06	70.89
34	2	2	30	75	3	3	40	0.6	1.5	33.37	33.06	70.89
34	2	2	30	150	3	3	40	0.6	1.5	33.37	33.06	70.89

Table (4.18) The overall test results of the video compression scheme based on using MOTS method (for the first video sequence)

GOP size	No. of AF	No. of IF	No. of PF	Th	NoPass	Q_L	Q	α	β	MSE	PSNR	C_r
12	2	0	10	150	3	3	40	0.6	1.5	31.93	33.15	65.22
22	2	2	18	150	3	3	40	0.6	1.5	26.35	33.97	70.69
34	2	2	30	150	3	3	40	0.6	1.5	28.19	33.61	71.46
37	2	0	34	150	3	3	40	0.6	1.5	77.14	30.95	61.76
37	2	1	34	150	3	3	40	0.6	1.5	42.98	32.38	56.59
37	2	2	33	150	3	3	40	0.6	1.5	31.56	33.21	68.30
37	2	3	32	150	3	3	40	0.6	1.5	34.21	33.02	71.03
34	2	2	30	15	3	3	40	0.6	1.5	28.88	33.53	71.39
34	2	2	30	50	3	3	40	0.6	1.5	28.19	33.63	71.36
34	2	2	30	75	3	3	40	0.6	1.5	28.19	33.63	71.36
34	2	2	30	150	3	3	40	0.6	1.5	28.19	33.63	71.36

4.8 Tables Notes

1. The increase in the number of wavelet passes (level) causes an increase in C_r, and decreasing in Quality.
2. The increase in the quantization step causes a decrease in the frame quality and an increase in value of C_r, and vice versa.
3. The results of using MOTS method refer to good compression ratio and PSNR, when they are compared with the results of OTS method.
4. The increase of GOP size without using interpolation coding lead to a decrease in C_r and quality, but when using interpolation coding this increase leads to an increase in C_r.
5. Increasing the number of interpolated frames lead to increasing both quality and compression ratio.
6. The increase in threshold values of motion estimation leads to increase in quality and decrease in compression ratio, and this increase or decrease will stop at certain threshold values.

Chapter Five
Conclusions and Future Works

5.1 Conclusions

From the test results presented in previous chapter, some remarks related to the behavior and performance of the two investigated video coding schemes are stimulated. Among these remarks are the following:

1. The quantization parameters mainly affects on the (MSE, PSNR, and C_r). It was found that the suitable values are (Q_L=3, Q=40, α=0.6, β=1.5) which led to good PSNR (less distortion) and C_r.
2. The new proposed MOTS method of ME has better performance in terms of image quality and compression ratio, and it is faster than the traditional OTS method.
3. When the size of GOP is taken small then the number of frames coded as anchor frames will increase, and consequently the error level of the compressed frames decreases and the attained compression ratio become less.
4. The use of interpolation frames with large GOP size will increase in compression ratio of estimated frames, but it will cause a decrease in compression performance (PSNR).
5. The value of the parameter (number of interpolated frames) affects the performance of the inter-coding methods (OTS and MOTS), where in the increase in the number of interpolation frames leads to less error and good quality and better compression ratio.

5.2 Future Works

1. Utilize another wavelet-based compression schemes (like packet wavelet transform, Quadtree wavelet transform) to compress anchor frames.
2. Using vector quantization to improve the compression performance of the detail subbands.
3. Using another types of coding (like, Huffman coding, and run length coding) instead of shift coding.
4. Using the same enhancement mechanism to improve the performance of other motion estimation methods (like, modified to orthogonal search algorithm (OSA), and four step search (FSS)).
5. Improve the performance of motion estimation by searching the most similar anchor frame (instead of using nearest anchor frame).
6. Add further coding step to handle those blocks which their motion compensation led to high error, such blocks could be compressed by using one of the still image coding methods.